Syllabus

**UNIT 3**

**SOA & Cloud Management**

* 1. Definition of Service Oriented Architecture
  2. Basic concepts of SOA
  3. Web Services: SOAP and REST
  4. Cloud APIs (RESTful)
  5. Relating SOA and Cloud Computing.
  6. Cloud Availability
  7. Cloud Governance
  8. Service Level Agreement

**Extra Reading**: Pricing Model: Usage Reporting, billing and metering (AWS), Cloud Statistics -- Ask students to give presentation on the same

**Service-Oriented Architecture**

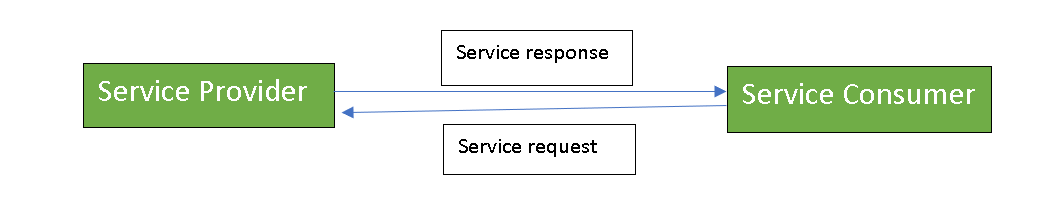
Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration. It defines a way to make software components reusable using the interfaces.

Formally, SOA is an architectural approach in which applications make use of services available in the network. In this architecture, services are provided to form applications, through a network call over the internet. It uses common communication standards to speed up and streamline the service integrations in applications. Each service in SOA is a complete business function in itself. The services are published in such a way that it makes it easy for the developers to assemble their apps using those services. Note that SOA is different from micro service architecture.

* SOA allows users to combine a large number of facilities from existing services to form applications.
* SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
* SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

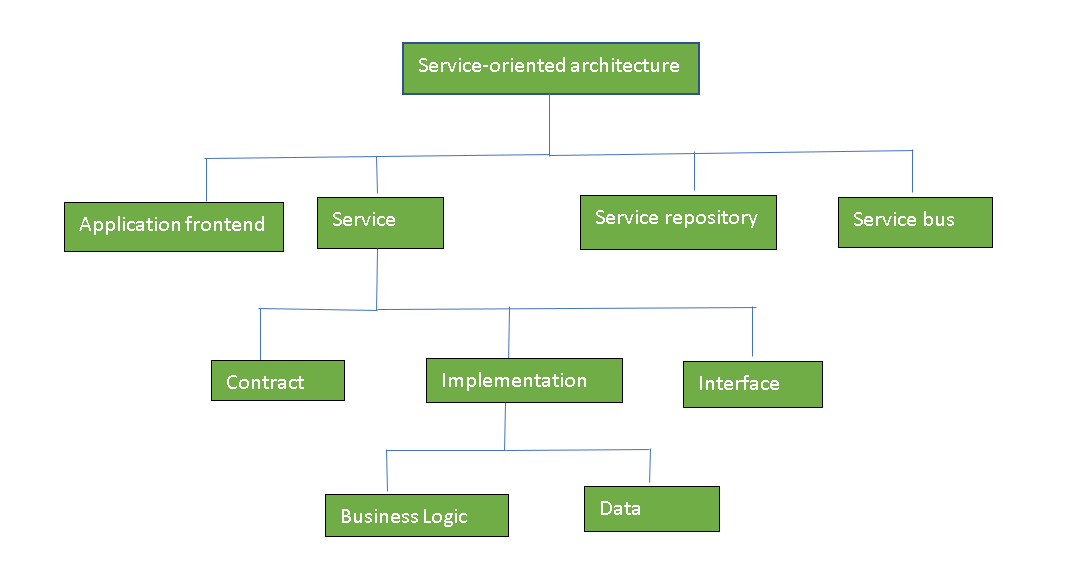
There are two major roles within Service-oriented Architecture:

1. **Service provider:** The service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.
2. **Service consumer:** The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.



**Services might aggregate information and data retrieved from other services or create workflows of services to satisfy the request of a given service consumer. This practice is known as service orchestration Another important interaction pattern is service choreography, which is the coordinated interaction of services without a single point of control.**

**Components of SOA:** 



**Guiding Principles of SOA:**

1. **Standardized service contract:**Specified through one or more service description documents.
2. **Loose coupling:** Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.
3. **Abstraction:** A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.
4. **Reusability:** Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.
5. **Autonomy:** Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.
6. **Discoverability:** Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered. Service discovery provides an effective means for utilizing third-party resources.
7. **Compos ability:** Using services as building blocks, sophisticated and complex operations can be implemented. Service orchestration and choreography provide a solid support for composing services and achieving business goals.

**Advantages of SOA:**

* **Service reusability:** In SOA, applications are made from existing services. Thus, services can be reused to make many applications.
* **Easy maintenance:** As services are independent of each other they can be updated and modified easily without affecting other services.
* **Platform independent:** SOA allows making a complex application by combining services picked from different sources, independent of the platform.
* **Availability:** SOA facilities are easily available to anyone on request.
* **Reliability:** SOA applications are more reliable because it is easy to debug small services rather than huge codes
* **Scalability:**Services can run on different servers within an environment, this increases scalability

**Disadvantages of SOA:**

* **High overhead:** A validation of input parameters of services is done whenever services interact this decreases performance as it increases load and response time.
* **High investment:** A huge initial investment is required for SOA.
* **Complex service management:** When services interact they exchange messages to tasks. the number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.

**Practical applications of SOA:** SOA is used in many ways around us whether it is mentioned or not.

1. SOA infrastructure is used by many armies and air forces to deploy situational awareness systems.
2. SOA is used to improve healthcare delivery.
3. Nowadays many apps are games and they use inbuilt functions to run. For example, an app might need GPS so it uses the inbuilt GPS functions of the device. This is SOA in mobile solutions.
4. SOA helps maintain museums a virtualized storage pool for their information and content.

**What is SOAP?**

**SOAP** is a protocol which was designed before REST and came into the picture. The main idea behind designing SOAP was to ensure that programs built on different platforms and programming languages could exchange data in an easy manner. SOAP stands for Simple Object Access Protocol.

**What is REST?**

**REST** was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principles of REST can be called a RestFul web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components. REST stands for Representational State Transfer.

**When to use REST?**

One of the most highly debatable topics is when REST should be used or when to use SOAP while designing web services. Below are some of the key factors that determine when REST and SOAP API technology should be used for web services **REST services should be used in the following instances**

* **Limited resources and bandwidth** – Since SOAP messages are heavier in content and consume a far greater bandwidth, REST should be used in instances where network bandwidth is a constraint.
* **Statelessness** – If there is no need to maintain a state of information from one request to another then REST should be used. If you need a proper information flow wherein some information from one request needs to flow into another then SOAP is more suited for that purpose. **We can take the example of any online purchasing site.** These sites normally need the user first to add items which need to be purchased to a cart. All of the cart items are then transferred to the payment page in order to complete the purchase. This is an example of an application which needs the state feature. The state of the cart items needs to be transferred to the payment page for further processing.
* **Caching**– If there is a need to cache a lot of requests then REST is the perfect solution. At times, clients could request for the same resource multiple times. This can increase the number of requests which are sent to the server. By implementing a cache, the most frequent queries results can be stored in an intermediate location. So whenever the client requests for a resource, it will first check the cache. If the resources exist then, it will not proceed to the server. So caching can help in minimizing the amount of trips which are made to the web server.
* **Ease of coding**– Coding REST Services and subsequent implementation is far easier than SOAP. So if a quick win solution is required for web services, then REST is the way to go.

**When to use SOAP?**

SOAP should be used in the following instances

1. **Asynchronous processing and subsequent invocation** – if there is a requirement that the client needs a guaranteed level of reliability and security then the new SOAP standard of SOAP 1.2 provides a lot of additional features, especially when it comes to security.
2. **A Formal means of communication** – if both the client and server have an agreement on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction. An example is an online purchasing site in which users add items to a cart before the payment is made. Let’s assume we have a web service that does the final payment. There can be a firm agreement that the web service will only accept the cart item name, unit price, and quantity. If such a scenario exists then, it’s always better to use the SOAP protocol.
3. **Stateful operations –**ifthe application has a requirement that state needs to be maintained from one request to another, then the SOAP 1.2 standard provides the structure to support such requirements.

There are many differences between SOAP and REST web services. The important 10 differences between SOAP and REST are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services. |
| 6) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 7) | SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. |
| 8) | SOAP **defines its own security**. | RESTful web services **inherits security measures** from the underlying transport. |
| 9) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| 10) | SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |

**Cloud Application Programming Interface (Cloud API)**

What Does Cloud Application Programming Interface (Cloud API) Mean?

A Cloud API is a software interface that allows developers to link [cloud computing](https://www.techopedia.com/definition/2/cloud-computing) services together. Application programming interfaces ([APIs](https://www.techopedia.com/definition/24407/application-programming-interface-api)) allow one computer program to make its data and functionality available for other programs to use. Developers use APIs to connect software components across a network.

Cloud APIs are often categorized as being **vendor-specific** or **cross-platform**. Vendor-specific cloud APIs are written to support the cloud services of one specific provider, while cross-platform APIs allow developers to connect functionalities from two or more cloud providers.

Cloud APIs are often categorized by type:

* **PaaS APIs:** Platform as a Service APIs provide access to back-end services such as databases.
* **SaaS APIs:** Software as a Service APIs facilitate connections between cloud services at the [application layer](https://www.techopedia.com/definition/6006/application-layer).
* **IaaS APIs**: Infrastructure as a Service APIs enable cloud-based compute and storage resources to be [provisioned](https://www.techopedia.com/definition/26531/cloud-provisioning) and de-provisioned as quickly as possible.

When a cloud provider creates an application or service, they also create APIs so that other software can communicate with that software or service. The components of an API ecosystem include the following:

Assets - Information to be shared internally and/or externally with end users.

APIs - Connect assets to end users.

Developers - Create applications that can use APIs to share assets.

Software - The applications that use APIs to provide services for end users.

End Users - People or programs that request and are granted access to assets through an API.

Cloud API Protocols

The protocols that support Cloud APIs include:

[REST](https://www.techopedia.com/definition/1312/representational-state-transfer-rest) -- RESTful APIs use the HTTP protocol to perform functions such as creating, reading, updating, archiving and deleting records. RESTful APIs use Uniform Resource Identifiers (URIs) to exchange data.

[SOAP (Simple Object Access Protocol)](https://www.techopedia.com/definition/1313/simple-object-access-protocol-soap) -- SOAP APIs use the [XML](https://www.techopedia.com/definition/24387/extensible-markup-language-xml) protocol in addition to HTTP to transfer data. Generally speaking, SOAP APIs are stricter and more heavyweight than RESTful APIs.

JSON-RPC -- This type of [remote procedure call](https://www.techopedia.com/definition/24022/remote-procedure-call-rpc) uses [JSON](https://www.techopedia.com/definition/3930/javascript-object-notation-json) formatting instead of XML to transfer data. Because RPC APIs are more challenging to maintain and update than RESTful APIs, their use has declined over years.

Cloud API Security

Many of the threats to cloud APIs are the same for other technologies in terms of threat actors and [attack vectors](https://www.techopedia.com/definition/15793/attack-vector). Unprotected APIs introduce cybersecurity risks and can be used as an unauthorized entry point into an organization’s network and databases.

Authentication and authorization mechanisms such as [OAuth2.0](https://www.techopedia.com/definition/28983/oauth-20) and [OpenID Connect](https://www.techopedia.com/definition/4951/openid) can mitigate the chance that a vulnerable API can be used to launch a [lateral attack](https://www.techopedia.com/definition/34772/lateral-movement-cybersecurity-cybersecurity).

Popular Cloud APIs

Popular cloud APIs and platform bundles include the following:

**Google Cloud APIs**

[Google Compute Engine API](https://cloud.google.com/compute/docs/api/libraries) - used to create and run virtual machines (VMs) on Google Cloud.

[Google Storage Transfer API](https://cloud.google.com/storage-transfer/docs/apis) - used to transfer data from an external source to Google Cloud.

[AutoML API](https://cloud.google.com/automl/docs) - helps citizen developers to create [machine learning models](https://www.techopedia.com/definition/8181/machine-learning-ml) for specific business requirements.

**AWS APIs**

[AWS Cloud Control](https://www.techopedia.com/definition/26437/cloud-application-programming-interface-cloud-api) - an integrated set of APIs designed to make it easy for developers to manage these five operations: create, read, update, delete and list ([CRUD](https://www.techopedia.com/definition/25949/create-retrieve-update-and-delete-crud)-L).

[Amazon EventBridge](https://docs.aws.amazon.com/eventbridge/latest/APIReference/Welcome.html) (formerly CloudWatch Events) - a cloud-based [bus service](https://www.techopedia.com/definition/2162/bus) for connecting applications with data from disparate sources.

[Amazon API Gateway](https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html) - a cloud service for service creating [REST](https://www.techopedia.com/definition/1312/representational-state-transfer-rest), HTTP, and [WebSocket](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API) APIs at scale and maintaining them throughout their lifecycle.

**Azure APIs**

[Azure Communication Services](https://azure.microsoft.com/en-us/services/communication-services/) - provides APIs for voice, video, chat, SMS and email. Requires applications to use the same infrastructure as Microsoft Teams.

[Azure Cognitive Services](https://azure.microsoft.com/en-us/services/cognitive-services/) - provides APIs for AI services, including speech to text, text to speech, speech translation, speaker recognition, sentiment analysis, natural language understanding (NLU), computer vision, facial recognition and anomaly detection.

[Azure API Management](https://azure.microsoft.com/en-us/services/api-management/#overview) - provides developers with an API gateway, management plane and developer portal so they can expose services hosted on Azure as APIs.

**CLOUD Availability:**

**3 best practices to achieve high availability in cloud computing**

**High availability is the crucial test of whether a business can continue to access data and applications when things go wrong in a cloud-based IT infrastructure.**

Availability is an important part of service-level agreements in cloud computing to ensure that infrastructure can continue to function even if a component fails. If there is poor availability, a business is unable to access its data or applications -- and potentially loses revenue.

Availability addresses points of failure within systems, databases and applications. High availability, sometimes referred to as HA, better protects companies from disruptions, and it supports productivity and reliability.

Follow these three best practices to achieve high availability in cloud computing.

**1. Determine how much uptime you need**

Uptime is a measurement of **how long a system properly functions.** A service-level agreement (SLA) between a cloud service provider and a customer will state the cloud's **expected availability and potential consequences for failing to meet it.**

Large providers, such as AWS, Microsoft Azure and Google Cloud, have SLAs of at least 99.9% availability for each paid service. **The provider promises its customers that they will experience less than nine hours of downtime over the course of a year.** The more nines in the number, the less downtime the customer can expect to experience in a year.

Application complexity can affect uptime. For example, simple websites could see availability of 99.9999% -- approximately 31.6 seconds of downtime each year -- because there are very few points of failure. On the other hand, a more complex monolithic web application that has more components, such as caching servers or object storage, creates more points of failure and may make high availability difficult. Enterprises can employ additional redundancies to ensure uptime, but that increases costs.

The [required amount of uptime](https://www.techtarget.com/searchcloudcomputing/answer/How-much-cloud-uptime-do-you-need) an application needs largely depends on how important it is. For example, users visiting a lawn care e-commerce giant's site may be more forgiving if downtime occurs than users of an emergency services provider. When negotiating an SLA with a cloud service provider, a business should weigh the consequences of downtime for its users and what it can afford. Not everything needs 99.999999% availability.

**2. Understand core high availability components**

High availability may cost a lot of time and money, but it is essential for mission-critical applications. However, the key to high availability is to apply the right amount of resources to a workload. There are many tools to ensure that workloads remain accessible during internal or external disruptions. Organizations should apply the right resources and availability requirements to a given workload to balance reliability and performance with costs.

There are several [components to a public cloud platform](https://www.techtarget.com/searchcloudcomputing/tip/Know-the-tools-needed-to-bake-cloud-HA-into-apps) that organizations should understand to weigh the benefits and costs of high availability:

**Physical locations.** Organizations achieve high availability through finding and eliminating single points of failure and by distributing redundant instances across [availability zones](https://www.techtarget.com/searchaws/definition/availability-zones).

**Networking.** A good network connection is essential when transferring data between the cloud and local storage. Some workloads require dedicated connectivity.

**Compute instances.** In public clouds, servers take the form of compute instances. A cloud customer can organize those instances into clusters or create backup instances for failover, which can cost more.

**Storage instances.** Data from applications is kept in storage instances, and cloud storage services are highly available. This removes the need for replication. However, be wary of storage becoming a single point of failure for applications.

**Load balancing.** Load balancing is how organizations direct traffic to multiple compute instances to accommodate for more load on the instances. Load balancers are often the first component to discover, report and modify an instance failure.

**IP cutover.** When an instance fails, the IP address of the failed instance must be remapped to the alternate instance to redirect traffic.

**Monitoring.** In terms of SLAs, monitoring can help to validate uptime availability. It also serves to reveal availability complications as well as track cloud resource usage.

**Plan cloud capacity for unexpected spikes**

To handle fluctuations in demand for cloud resources, users can scale an application.

**Vertical scaling.** Also known as scaling up, this method helps avoid failures or slowdowns and processes data at the speed at which it comes in. Cloud admins can handle increased flow by adding more capacity to an instance. While this method is good for a predictable increase, it does not address unexpected spikes beyond the threshold.

**Horizontal scaling.** Also called scaling out, this process adds instances to a resource pool. It helps solve the problem of increased demand with resources used in larger quantities, rather than increasing the size of individual instances. It should cost less than scaling vertically.

Cloud scaling should be an automated process. In addition to handling spikes, automated scaling should help control costs.

**3. Assess application needs before adding HA**

It's easy to apply services such as load balancing and IP addressing schemes to the cloud. But every application is different, and cloud users should [assess their needs](https://www.techtarget.com/searchcloudcomputing/tip/Ask-these-5-questions-before-deploying-cloud-HA) before applying high availability. Before adding high availability to an application, ask these questions:

**Does the workload benefit from HA?** High availability isn't always the best fit, in terms of cost and complexity. An admin might select a high availability workload type even when it is not necessary.

**Does cloud HA justify the cost?** Consider the amount of expected downtime and how users will react to this. Then determine the maximum allowable downtime and implement the right high availability strategies to make certain that requirement is met. Monitoring and recording cloud availability and downtime is a way to know the acceptable performance.

**Is HA applied to the right assets?** Figure out what the organization's goals are, such as optimal performance and workload availability. Evaluate what is the most valuable aspect for the cloud workload and how uptime requirements will benefit these goals.

**Is HA more complex than necessary?** High availability comes from a wide range of technologies and procedures that can be used or combined. Evaluate whether there is a simpler way to achieve protection from downtime that would cost less money.

**Does cloud HA work as intended?** Evaluate the high availability setup to make certain the deployment was successful. Review performance against disruption from physical events, such as natural disasters. Audit the infrastructure to ensure the established requirements are being met. If instances fail, they should bounce back within a justifiable time frame and without data loss, as specified in the SLA.

**Cloud Governance and Its Need**

[**Cloud**](https://www.geeksforgeeks.org/cloud-based-services/)**Governance** **:**

It is the set of policies or principles that act as the guidance for the adoption use, and management of cloud technology services.

It is an ongoing process that must sit on top of existing governance models.

It is a set of rules you create to monitor and amend as necessary in order to control costs, improve efficiency, and eliminate security risks.

[Need for Cloud](https://www.geeksforgeeks.org/why-cloud-computing-is-important-in-data-science/) Governance :

By implementing cloud governance, organizations can avoid the following issues as follows.

1. Security and privacy risks :

This issue may arise due to unauthorized downloads/ installation of software, storage of illegal data, and access to restricted sites by users.

Cloud Governance solutions cover multiple cloud security components. For example, Encryption, Security groups, Audit trails, Application access rules, Access controls.

2. Vendor lock-in :

Many vendors opt for this, as this clause causes organizations to depend on the cloud service provider (or vendor) for products and services.

This can be avoided by making changes to the SLA suitably and reduce dependencies on a single vendor, thus ensuring freedom to the organization.

3. Cloud Sprawl :

This happens when employees of different departments use different programs and cloud infrastructure from third-party providers without involving the IT department and getting necessary approvals.

If not detected and restricted, crowd sprawl may lead to fragmented, redundant, inefficient, and unmanaged cloud programs sitting on the enterprise cloud and unnecessarily creating trouble.

4. Shadow IT and unwarranted usage of cloud resources :

This happens when employees in various departments do not follow the rules and regulations as imposed by the IT department on cloud usage resulting in security breaches and fragmented control throughout the organization.

This leads to not getting sufficient results from the cloud in the long run.

5. Lack of data portability and interoperability :

This happens when the cloud service provider or the inbuilt cloud infrastructure is incapable of connecting well with other software and products outside the organization.

This may also lead to modules not compatible with each other and hence chaos in the cloud due to an inefficient system.

What is an SLA?

A service level agreement (SLA) is a [documented agreement](https://www.bmc.com/blogs/sla-best-practices/) between a service provider and a customer that identifies both the services required and the expected level of service. The agreement varies between vendors, services, and industries.

Before subscribing for an IT service, the SLA should be carefully evaluated and designed to realize maximum service value from an end-user and business perspective. Service providers should pay attention to the differences between [internal outputs and customer-facing outcomes](https://www.bmc.com/blogs/outcomes-vs-outputs/), as these can help define the service expectations.

An introduction to service level agreements

Success in business depends heavily on an organization’s ability to understand and meet customer expectations. But when those expectations are not clear, or when customers are not fully informed about what they can expect from a service provider, managing customer expectations may become extremely difficult. To counter this issue, businesses of all kinds rely on SLAs.

An SLA functions as a documented understanding between the entity providing the service and the one receiving the benefits of the service. Although traditional SLAs define service expectations between vendors and customers, they may also be employed between departments within the same organization. And while the SLA may consist of as little as a few sentences or as much as entire documents’ worth of provisions and stipulations, they are always a critical component of modern service contracts. It’s also important to note that SLAs should not be thought of as immutable; they should change and grow to meet evolving business needs. With this in mind, SLAs should incorporate a clear framework for introducing revisions or modifications during the course of the contract.

Who provides the SLA?

The SLA is created and provided by the service vendor. This allows the organization to customize their various SLAs to meet specific service and customer requirements. In fact, there might be cases in which a company may provide several SLAs for a single service, with each SLA reflecting different levels of service at different price points. However, because SLAs are usually prepared by the vendor, they may favor the service provider over the customer. As such, it can be helpful to encourage clients to review SLAs—and to even consider bringing in legal counsel, where appropriate—to ensure that the SLA is satisfactory before making any formal commitments. This will help prevent issues where customers feel as though they have been misled.

Why are SLAs so Important?

Although they may seem straightforward, SLAs serve a variety of functions.

**Ensuring everyone involved is on the same page**

At their most basic, SLAs are designed to keep both parties honest. If there is an issue with the service, the SLA acts as vital documentation detailing all the metrics, responsibilities, and expectations that were originally agreed upon. Both parties have access to the SLA, which means that neither may claim that they were unaware of expectations or agreed-upon standards after the fact. A legally reviewed and approved SLA effectively eliminates the possibility of misinterpretation of the contract and provides protection to both the customer and the service provider. Clients enjoy the peace of mind that comes from knowing that they will receive the service they are paying for, and providers have a contract to refer their customers back should they start to demand services that are not included in the agreement.

**Providing clear metrics**

More than simply defining what the services themselves will consist of an SLA also establishes the metrics by which service levels will be measured. These guidelines allow both parties to better understand and gauge the effectiveness of the service, and whether it is fulfilling the terms of the contract. The metrics outlined in the SLA must be clearly stated and fully quantifiable, so that there is no room for debate about whether vital KPIs are being met.

**Offering recourse for unmet obligations**

SLAs not only help define standards, set expectations, and offer insight into important metrics, they also act as a roadmap of next steps and consequences if obligations are not met. Should one of the parties fail to meet the standards outlined in the contract, the SLA will define the consequences—potentially including legal penalties or forms of financial restitution. Clearly defined repercussions help hold both parties accountable.

Improving provider-business partnerships

Finally, an effectively implemented SLA helps manage commitments between service providers and customers to improve the service provider-business partnership. Clients and businesses enjoy more productive and successful relationships, and neither has to worry about the other failing to meet their obligations. At the same time, the right SLA can help assuage the concerns of potential new customers, bringing in new business opportunities and improving the brand reputation.

What are the benefits of an SLA?

The importance of SLAs in ensuring that expectations are being effectively managed cannot be overstated. Additionally, SLAs bring with them several clear advantages , including the following:

**Improved customer experience**

Customers who invest in a service provider are taking a risk—they are acting in good faith with the expectation that the provider will be able to meet their needs. The SLA provides these customers with a sort of safety net; the customer knows that should the provider fail to deliver on the agreed-upon services or otherwise fail to meet obligations, then the customer will have legally binding documentation to assist them in seeking restitution. By managing expectations and giving customers some much-needed insurance, SLAs can improve the overall customer experience.

**Improved employee experience**

When obligations are clearly defined and metrics are transparent and quantifiable, it benefits everyone involved. This includes employees, who have a clearer understanding of what is expected of them and how their performance will be measured.

**Established and trusted source of information**

In many ways, the SLA acts as a mediator, ensuring that everyone’s best interests are being looked after. It is something that can be trusted by both parties, acting as a source of reliable, legally vetted information relevant to service standards and other guidelines.

**Increased productivity and performance**

The metrics established within SLAs offer their own internal advantages. With clear expectations, employees have a ‘north star’ by which to guide their performance. This leads to improved productivity and increases personal accomplishment.