SYSFORE®

# INTRODUCTION TO AZURE STORAGE, BACKUP & RECOVERY

PART 2

Sysfore Technologies is a leading Azure Accelerate Partner with Microsoft for South India and a prime enabler for Microsoft Azure adoption.

# CONTENTS

## AN OVERVIEW OF AZURE STORAGE

Azure is cloud storage that allows you to have anywhere and anytime access as long as you have internet connectivity. Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support the big data scenarios required by scientific, financial analysis, and media applications. Or you can store the small amounts of data required for a small business website. Wherever your needs fall, you pay only for the data you're storing. Azure Storage currently stores tens of trillions of unique customer objects, and handles millions of requests per second on average.

Azure Storage is elastic, so you can design applications for a large global audience, and scale those applications as needed - both in terms of the amount of data stored and the number of requests made against it. You pay only for what you use, and only when you use it.

Azure Storage uses an auto-partitioning system that automatically load-balances your data based on traffic. This means that as the demands on your application grow, Azure Storage automatically allocates the appropriate resources to meet them.

Azure Storage is accessible from anywhere in the world, from any type of application, whether it's running in the cloud, on the desktop, on an on-premises server, or on a mobile or tablet device. You can use Azure Storage in mobile scenarios where the application stores a subset of data on the device and synchronizes it with a full set of data stored in the cloud.

Azure Storage supports clients using a diverse set of operating systems (including Windows and Linux) and a variety of programming languages (including .NET, Java, and C++) for convenient development. Azure Storage also exposes data resources via simple REST APIs, which are available to any client capable of sending and receiving data via HTTP/HTTPS.

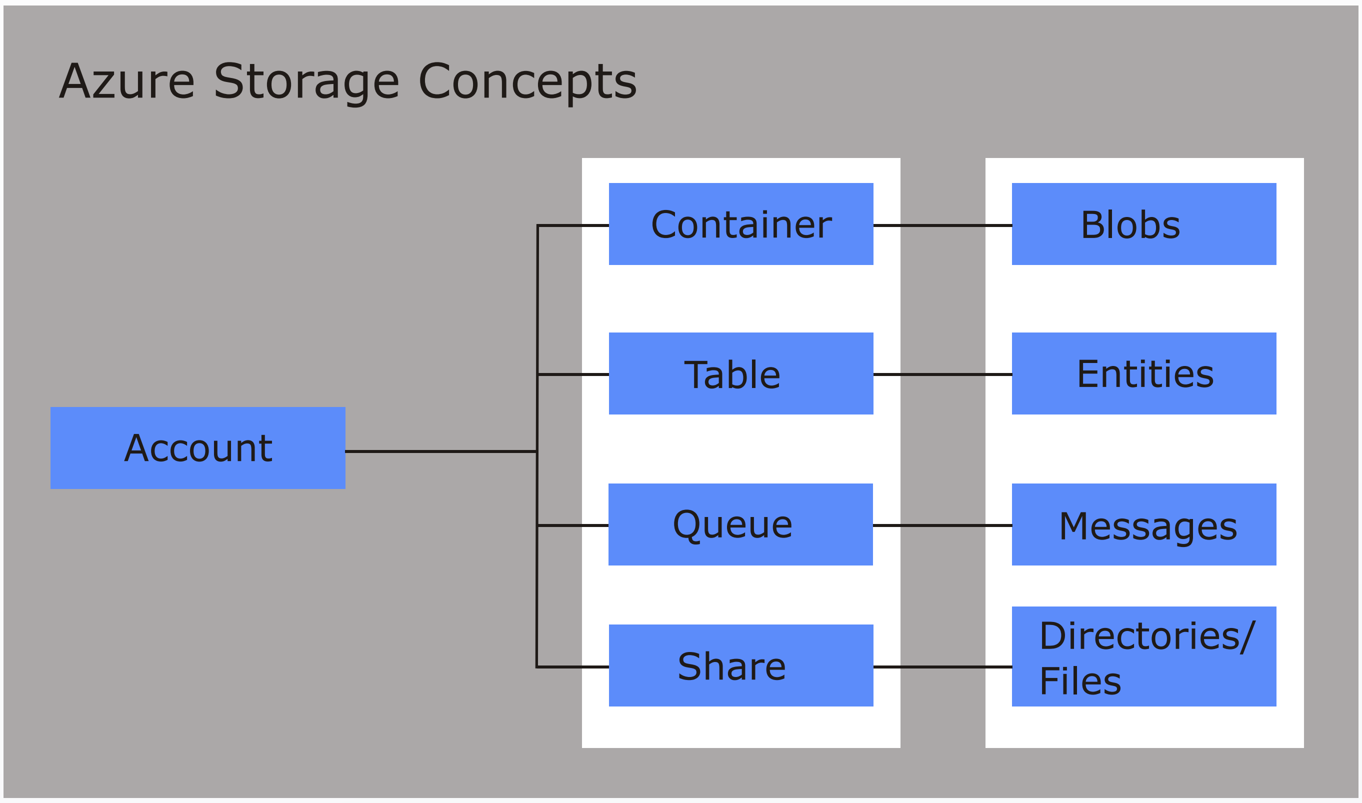
### Regions:

There will be 16 regions available in 2014, which by far is the highest number of regions as compared to any major cloud service provider. This enables you to build your cloud storage applications based on where your consumers are.

## INTRODUCING THE AZURE STORAGE SERVICES

**The Azure Storage services are Blob storage, Table storage, Queue storage, and File storage:**

- Blob storage stores file data. A blob can be any type of text or binary data, such as a document, media file, or application installer.
- Table storage stores structured datasets. Table storage is a NoSQL key-attribute data store, which allows for rapid development and fast access to large quantities of data.
- Queue storage provides reliable messaging for workflow processing and for communication between components of cloud services.
- File storage offers shared storage for legacy applications using the standard SMB 2.1 protocol. Azure virtual machines and cloud services can share file data across application components via mounted shares, and on-premise applications can access file data in a share via the File service REST API.



## ABSTRACTIONS:

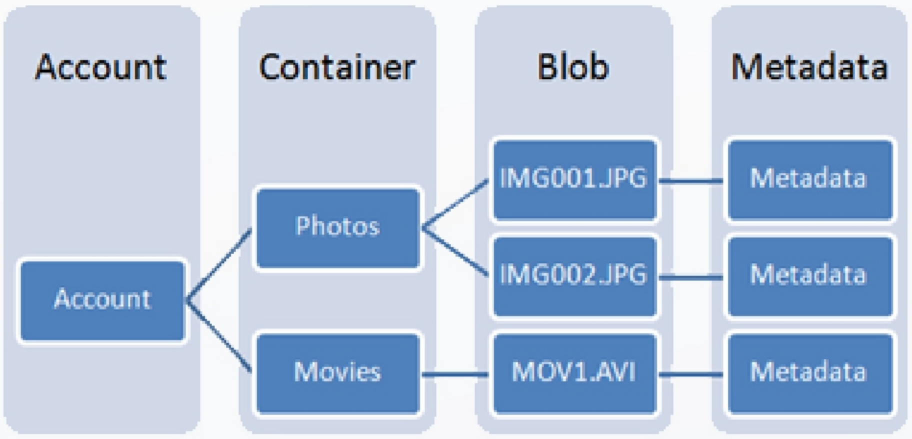
**Blob Storage:**

For users with large amounts of unstructured data to store in the cloud, Blob storage offers a cost-effective and scalable solution. You can use Blob storage to store content such as:

- Data sharing- Documents, photos, videos, music, blogs etc
- Simple REST Interface for putting, getting or deleting blobs.
- Backups of files, computers, databases, and devices.
- Configuration data for cloud applications.
- Big data, such as logs and other large datasets.

Every blob is organized into a container. Containers also provide a useful way to assign security policies to groups of objects. A storage account can contain any number of containers, and a container can contain any number of blobs, up to the 500 TB capacity limit of the storage account.

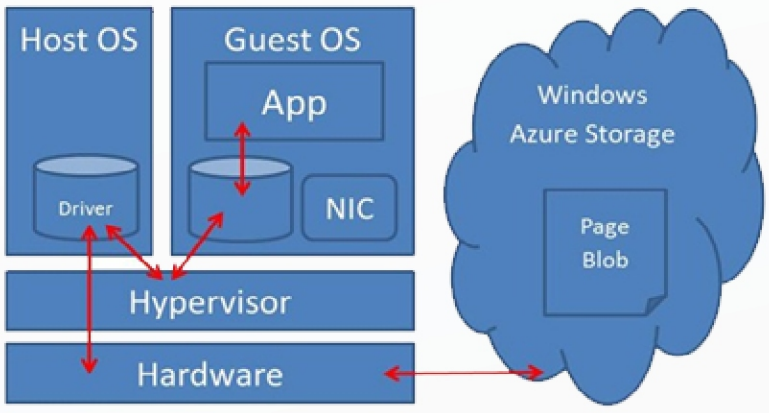
Blob storage offers two types of blobs, block blobs and page blobs (disks). Block blobs are optimized for streaming and storing cloud objects, and are a good choice for storing documents, media files, backups etc. A block blob can be up to 200 GB in size. Page blobs are optimized for representing IaaS disks and supporting random writes, and may be up to 1 TB in size. An Azure virtual machine network attached IaaS disk is a VHD stored as a page blob.



## Disk Storage:

When you create a Windows Azure Virtual Machine, the platform will attach at least one disk to the VM for your operating system disk. This disk will also be a VHD stored as a page blob in storage. As you write to the disk in the VM, the changes to the disk will be made to the page blob inside storage. You can also attach additional disks to your VM as data disks, and these will be stored in storage as page blobs as well.

Unlike for drives, the code that communicates with storage on behalf of your disk is not within your VM, so doing IO to the disk will not cause network activity in the VM, although it will cause network activity on the physical node. The following diagram shows how the driver runs in the host operating system, and the VM communicates through the disk interface to the driver, which then communicates through the host network adapter to storage.



## Table Storage:

Table storage offers highly available, massively scalable storage, so that your application can automatically scale to meet user demand. Table storage is Microsoft's NoSQL key/attribute store  it has a schemaless design, making it different from traditional relational databases. With a schemaless data store, it's easy to adapt your data as the needs of your application evolve. Table storage is easy to use, so developers can create applications quickly. Access to data is fast and cost-effective for all kinds of applications. Table storage is typically significantly lower in cost than traditional SQL for similar volumes of data.

Table storage is a key-attribute store, meaning that every value in a table is stored with a typed property name. The property name can be used for filtering and specifying selection criteria.

A collection of properties and their values comprise an entity. Since Table storage is schema less, two entities in the same table can contain different collections of properties, and those properties can be of different types.

You can use Table storage to store flexible datasets, such as user data for web applications, address books, device information, and any other type of metadata that your service requires. You can store any number of entities in a table, and a storage account may contain any number of tables, up to the capacity limit of the storage account.

Like Blobs and Queues, developers can manage and access Table Storage using standard REST protocols, however Table Storage also supports a subset of the OData protocol, simplifying advanced querying capabilities and enabling both JSON and AtomPub (XML based) formats.

## File Storage:

File storage offers cloud-based file shares, so that you can migrate legacy applications to Azure quickly and without costly rewrites.

Applications running in Azure virtual machines or cloud services can mount a File storage share to access file data, just as a desktop application would mount a typical SMB share. Any number of application components can mount and access the File storage share simultaneously.

Since a File storage share is a standard SMB 2.1 file share, applications running in Azure can access data in the share via file sytem I/O APIs. Developers can therefore leverage their existing code and skills to migrate existing applications. IT Pros can use PowerShell cmdlets to create, mount, and manage File storage shares as part of the administration of Azure applications.

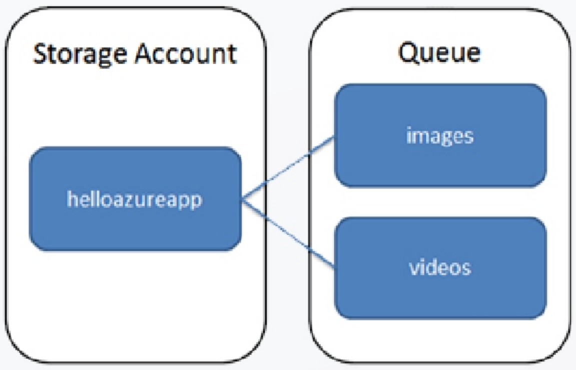
Like the other Azure storage services, File storage exposes a REST API for accessing data in a share. On-premise applications can call the File storage REST API to access data in a file share. This way, an enterprise can choose to migrate some legacy applications to Azure and continue running others from within their own organization. Note that mounting a file share is only possible for applications running in Azure; an on-premise application may only access the file share via the REST API.

Distributed applications can also use File storage to store and share useful application data and development and testing tools. For example, an application may store configuration files and diagnostic data such as logs, metrics, and crash dumps in a File storage share so that they are available to multiple virtual machines or roles. Developers and administrators can store utilities that they need to build or manage an application in a File storage share that is available to all components, rather than installing them on every virtual machine or role instance.

## Queue Storage:

Queue storage provides a reliable messaging solution for asynchronous communication between application components, whether they are running in the cloud, on the desktop, on an on-premises server, or on a mobile device. Queue storage also supports managing asynchronous tasks and building process workflows.

A storage account can contain any number of queues. A queue can contain any number of messages, up to the capacity limit of the storage account. Individual messages may be up to 64 KB in size.



## Additional Services, Tools and Libraries

**Azure Import/ Export**

- Move TBs of data into and out of Azure Blobs by shipping disks.
- Submit and monitor jobs via REST and Portal.
- All disks are encrypted with BitLocker.

**Azure Import/ Export**

- Client libraries like .NET, Java, C++, Node.js
- Windows phone and windows runtime
- Powershell commands
- CLI Tools for those who are not on windows platform but can manage azure storage on existing platform
- AZ Copy copy for blobs and disks. For backups, copying between accounts and between on premise and cloud.

## BACKUP & RECOVERY ON AZURE

Every enterprise needs to backup and restore data. You can use Azure to backup and restore your application whether in the cloud or on-premises. Azure offers different options to help depending on the type of backup.

### Site Recovery

Azure Site Recovery (formerly Hyper-V Recovery Manager) can help you protect important applications by coordinating the replication and recovery of Hyper-V images across sites. You can back up to your own secondary site, a hoster's site, or use Azure and avoid the expense and complexity of building and managing your own secondary location. Azure encrypts data and communications and you have the option enable encryption for data at-rest too.

It monitors the health of your services continuously and helps automate the orderly recovery of services in the event of a site outage at the primary datacenter. Virtual machines can be brought up in an orchestrated fashion to help restore service quickly, even for complex multi-tier workloads.

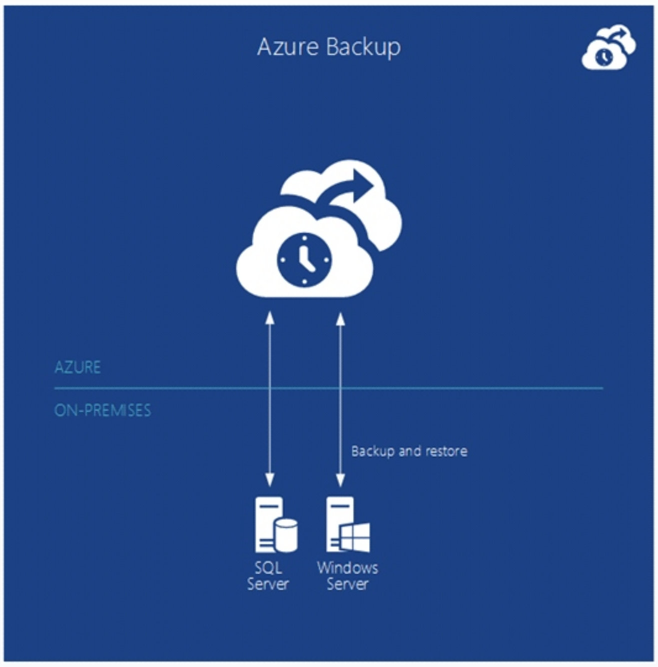


Figure: Azure Backup backs up data from on-premises Windows Servers into the cloud.

Azure Backup backs up data from on-premises servers running Windows Server into the cloud. You can manage your backups directly from the backup tools in Windows Server 2012, Windows Server 2012 Essentials, or System Center 2012 - Data Protection Manager. Alternatively, you can use a specialized backup agent.

Data is safer because backups are encrypted before transmission and stored encrypted in Azure and protected by a certificate that you upload. The service uses the same redundant and highly available data protection found in Azure Storage. You can back up files and folders on a regular schedule or immediately, running either full or incremental backups. After data is backed up to the cloud, authorized users can easily recover backups to any server. It also offers configurable data retention policies, data compression, and data transfer throttling so you can manage the cost to store and transfer data.

### Scenarios for Azure Backup

If you already using Windows Server or System Center, Azure backup is a natural solution for backing up your servers file system, virtual machines, and SQL Server databases. It works with encrypted, sparse and compressed files.

## GEO REDUNDANT STORAGE (GRS)

**Overview:**

With geo-replication, Windows Azure Storage now keeps your data durable in two locations. In both locations, Windows Azure Storage constantly maintains multiple healthy replicas of your data.

The location where you read, create, update, or delete data is referred to as the 'primary' location. The primary location exists in the region you choose at the time you create an account via the Azure Portal (e.g., North Central US). The location where your data is geo-replicated is referred to as the secondary location. The secondary location is automatically determined based on the location of the primary; it is in the other data center that is in the same region as the primary. In this example, the secondary would be located in South Central US. The primary location is currently displayed in the Azure Portal, as shown below. In the future, the Azure Portal will be updated to show both the primary and secondary locations. To view the primary location for your storage account in the Azure Portal, click on the account of interest; the primary region will be displayed on the lower right side under Country/Region, as highlighted below.

| Primary | Secondary |
|---|---|
| North Central US | South Central US |
| South Central US | North Central US |
| East US | West US |
| West US | East US |
| North Europe | West Europe |
| West Europe | North Europe |
| South East Asia | East Asia |
| East Asia | South East Asia |

## How Geo-Replication Works:

When you create, update, or delete data to your storage account, the transaction is fully replicated on three different storage nodes across three fault domains and upgrade domains inside the primary location, then success is returned back to the client. Then, in the background, the primary location asynchronously replicates the recently committed transaction to the secondary location. That transaction is then made durable by fully replicating it across three different storage nodes in different fault and upgrade domains at the secondary location. Because the updates are asynchronously geo-replicated, there is no change in existing performance for your storage account.

Our goal is to keep the data durable at both the primary and secondary location. This means we keep enough replicas in both locations to ensure that each location can recover by itself from common failures (e.g., disk, node, rack, TOR failing), without having to talk to the other location. The two locations only have to talk to each other to geo-replicate the recent updates to storage accounts. They do not have to talk to each other to recover data due to common failures. This is important, because it means that if we had to failover a storage account from the primary to the secondary, then all the data that had been committed to the secondary location via geo-replication will already be durable there.

With this first release of geo-replication, we do not provide an SLA for how long it will take to asynchronously geo-replicate the data, though transactions are typically geo-replicated within a few minutes after they have been committed in the primary location.

## SHARED ACCESS SIGNATURES (SAS)

**Overview:**

A shared access signature provides delegated access to resources in your storage account. This means that you can grant a client limited permissions to your blobs, queues, or tables for a specified period of time and with a specified set of permissions, without having to share your account access keys. The SAS is a URI that encompasses in its query parameters all of the information necessary for authenticated access to a storage resource. To access storage resources with the SAS, the client only needs to pass in the SAS to the appropriate constructor or method.

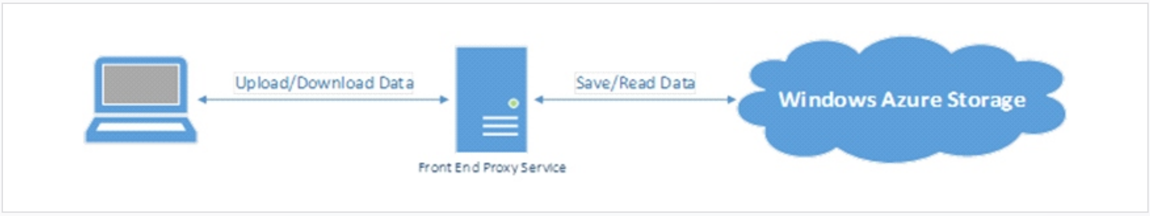
**When Should You Use a Shared Access Signature?**

You can use a SAS when you want to provide access to resources in your storage account to a client that can't be trusted with the account key. Your storage account keys include both a primary and secondary key, both of which grant administrative access to your account and all of the resources in it. Exposing either of your account keys opens your account to the possibility of malicious or negligent use. Shared access signatures provide a safe alternative that allows other clients to read, write, and delete data in your storage account according to the permissions you've granted, and without need for the account key.

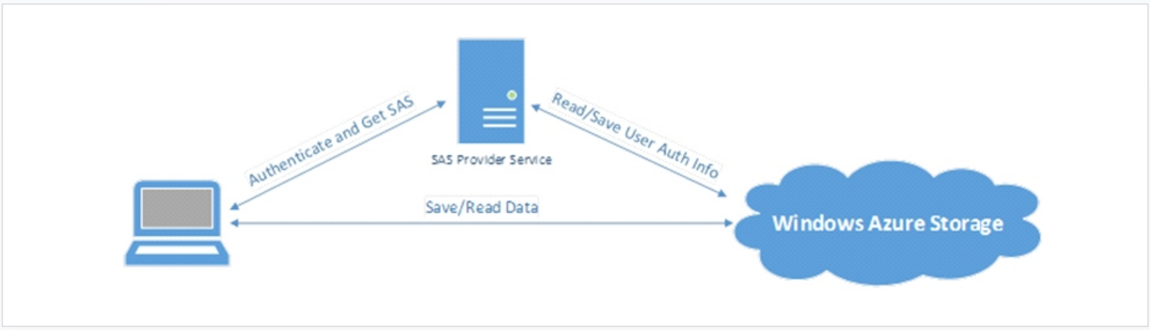
A common scenario where a SAS is useful is a service where users read and write their own data to your storage account.

In a scenario where a storage account stores user data, there are two typical design patterns:

**1.** Clients upload and download data via a front-end proxy service, which performs authentication.This front-end proxy service has the advantage of allowing validation of business rules, but for large amounts of data or high-volume transactions, creating a service that can scale to match demand may be expensive or difficult.



**2.** Clients upload and download data via a front-end proxy service, which performs authentication.This front-end proxy service has the advantage of allowing validation of business rules, but for large amounts of data or high-volume transactions, creating a service that can scale to match demand may be expensive or difficult.



Many real-world services may use a hybrid of these two approaches, depending on the scenario involved, with some data processed and validated via the front-end proxy while other data is saved and/or read directly using SAS.

**How a Shared Access Signature Works:**

A shared access signature is a URI that points to a storage resource and includes a special set of query parameters that indicate how the resource may be accessed by the client. One of these parameters, the signature, is constructed from the SAS parameters and signed with the account key. This signature is used by Azure Storage to authenticate the SAS.

A shared access signature has the following constraints that define it, each of which is represented as a parameter on the URI:

- The storage resource. Storage resources for which you can delegate access include containers, blobs, queues, tables, and ranges of table entities.
- Start time. This is the time at which the SAS becomes valid. The start time for a shared access signature is optional; if omitted, the SAS is effective immediately.
- Expiry time. This is the time after which the SAS is no longer valid. Best practices recommend that you either specify an expiry time for a SAS, or associate it with a stored access policy (see more below).
- Permissions. The permissions specified on the SAS indicate what operations the client can perform against the storage resource using the SAS.

**Controlling Shared Access Signatures with a Stored Access Policy**

A shared access signature can take one of two forms:

**Ad hoc SAS:** When you create an ad hoc SAS, the start time, expiry time, and permissions for the SAS are all specified on the SAS URI (or implied, in the case where start time is omitted). This type of SAS may be created on a container, blob, table, or queue.

**SAS with stored access policy:** A stored access policy is defined on a resource container - a blob container, table, or queue - and can be used to manage constraints for one or more shared access signatures. When you associate a SAS with a stored access policy, the SAS inherits the constraints - the start time, expiry time, and permissions - defined for the stored access policy.
The difference between the two forms is important for one key scenario: revocation. A SAS is a URL, so anyone who obtains the SAS can use it, regardless of who requested it to begin with. If a SAS is published publically, it can be used by anyone in the world.

**A SAS that is distributed is valid until one of four things happens:**

- The expiry time specified on the SAS is reached.
- The expiry time specified on the stored access policy referenced by the SAS is reached (if a stored access policy is referenced, and if it specifies an expiry time). This can either occur because the interval elapses, or because you have modified the stored access policy to have an expiry time in the past, which is one way to revoke the SAS.
- The stored access policy referenced by the SAS is deleted, which is another way to revoke the SAS. Note that if you recreate the stored access policy with exactly the same name, all existing SAS tokens will again be valid according to the permissions associated with that stored access policy (assuming that the expiry time on the SAS has not passed). If you are intending to revoke the SAS, be sure to use a different name if you recreate the access policy with an expiry time in the future.
- The account key that was used to create the SAS is regenerated. Note that doing this will cause all application components using that account key to fail to authenticate until they are updated to use either the other valid account key or the newly regenerated account key.

## Best Practices for Using Shared Access Signatures

When you use shared access signatures in your applications, you need to be aware of two potential risks:

- If a SAS is leaked, it can be used by anyone who obtains it, which can potentially compromise your storage account.
- If a SAS provided to a client application expires and the application is unable to retrieve a new SAS from your service, then the application's functionality may be hindered.

**The following recommendations for using shared access signatures will help balance these risks:**

- Always use HTTPS to create a SAS or to distribute a SAS. If a SAS is passed over HTTP and intercepted, an attacker performing a man-in-the-middle attack will be able to read the SAS and then use it just as the intended user could have, potentially compromising sensitive data or allowing for data corruption by the malicious user.
- Reference stored access policies where possible. Stored access policies give you the option to revoke permissions without having to regenerate the storage account keys. Set the expiration on these to be a very long time (or infinite) and make sure that it is regularly updated to move it farther into the future.
- Use near-term expiration times on an ad hoc SAS. In this way, even if a SAS is compromised unknowingly, it will only be viable for a short time duration. This practice is especially important if you cannot reference a stored access policy. This practice also helps limit the amount of data that can be written to a blob by limiting the time available to upload to it.
- Have clients automatically renew the SAS if necessary. Clients should renew the SAS well before the expected expiration, in order to allow time for retries if the service providing the SAS is unavailable. If your SAS is meant to be used for a small number of immediate, short-lived operations, which are expected to be completed within the expiration time given, then this may not be necessary, as the SAS is not expected be renewed. However, if you have client that is routinely making requests via SAS, then the possibility of expiration comes into play. The key consideration is to balance the need for the SAS to be short-lived (as stated above) with the need to ensure that the client is requesting renewal early enough to avoid disruption due to the SAS expiring prior to successful renewal.
- Be careful with SAS start time. If you set the start time for a SAS to now, then due to clock skew (differences in current time according to different machines), failures may be observed intermittently for the first few minutes. In general, set the start time to be at least 15 minutes ago, or don't set it at all, which will make it valid immediately in all cases. The same generally applies to expiry time as well - remember that you may observe up to 15 minutes of clock skew in either direction on any request. Note for clients using a REST version prior to 2012-02-12, the maximum duration for a SAS that does not reference a stored access policy is 1 hour, and any policies specifying longer term than that will fail.
- Be specific with the resource to be accessed. A typical security best practice is to provide a user with the minimum required privileges. If a user only needs read access to a single entity, then grant them read access to that single entity, and not read/write/delete access to all entities. This also helps mitigate the threat of the SAS being compromised, as the SAS has less power in the hands of an attacker.
- Understand that your account will be billed for any usage, including that done with SAS. If you provide write access to a blob, a user may choose to upload a 200GB blob. If you've given them read access as well, they may choose do download it 10 times, incurring 2TB in egress costs for you. Again, provide limited permissions, to help mitigate the potential of malicious users. Use short-lived SAS to reduce this threat (but be mindful of clock skew on the end time).

- Validate data written using SAS. When a client application writes data to your storage account, keep in mind that there can be problems with that data. If your application requires that that data be validated or authorized before it is ready to use, you should perform this validation after the data is written and before it is used by your application. This practice also protects against corrupt or malicious data being written to your account, either by a user who properly acquired the SAS, or by a user exploiting a leaked SAS.

- Don't always use SAS. Sometimes the risks associated with a particular operation against your storage account outweigh the benefits of SAS. For such operations, create a middle-tier service that writes to your storage account after performing business rule validation, authentication, and auditing. Also, sometimes it's simpler to manage access in other ways. For example, if you want to make all blobs in a container publically readable, you can make the container Public, rather than providing a SAS to every client for access.

## AZURE FILES

**Overview:**
The Azure File service exposes file shares using the standard SMB 2.1 protocol. Applications running in Azure can now easily share files between VMs using standard and familiar file system APIs like ReadFile and WriteFile. In addition, the files can also be accessed at the same time via a REST interface, which opens a variety of hybrid scenarios. Finally, Azure Files is built on the same technology as the Blob, Table, and Queue Services, which means Azure Files is able to leverage the existing availability, durability, scalability, and geo redundancy that is built into our platform.

**SMB 2.1: pros and cons**
Azure Files doesn't just sound like SMB  it's using the SMB 2.1 protocol for compatibility, so you'll be able to access it from Linux systems as well as Windows, in VMs on Azure or running in your own business (although you'll have to make a VPN connection to Azure first). You get standard file commands like moving and renaming files, writing content to a file or setting it to read only, or getting notifications when files are modified  all the things you're used to when working with files.

Picking SMB 2.1 means it will be compatible with far more systems and applications, because operating systems and tools and libraries understand files already. However, it also means it doesn't have built-in encryption.
That's why you can only access Azure Files storage from VMs and Azure roles in the same Azure region, not from another Azure data centre. Unless Microsoft comes up with a way to wrap encryption around a cross-data-centre connection, the transfer wouldn't be secure. But file storage can be geo-replicated within a region (in the same way as Azure blobs, tables and queues), so you do get redundancy.

You can also access your storage through a REST API, which lets you access individual files or the entire share, and from anywhere without needing a VPN. That means you can build a connection into an Azure Files storage account into a web application, giving you a lightweight way to build a website that lets you upload files like images from a smart phone.

**Therefore:**

- It enables moving on premises applications that rely on shared file storage to Azure.

- Natively supported by OS APIs, libraries and tools.

- Supports standard file system semantics

## File Rest APIs

This allows internet access to the same shared file system, for applications built in hybrid either on cloud or on premises.

It supports a variety of common APIs

- Create/ delete files and directories
- Write/ read files
- Get file and directory properties
- List file

## Scenarios

**"Lift and Shift" applications**
Azure Files makes it easier to "lift and shift" applications to the cloud that use on-premise file shares to share data between parts of the application. To make this happen, each VM connects to the file share (see "Getting Started" below) and then it can read and write files just like it would against an on-premise file share.

**Shared Application Settings**
A common pattern for distributed applications is to have configuration files in a centralized location where they can be accessed from many different virtual machines. Such configuration files can now be stored in an Azure File share, and read by all application instances. These settings can also be managed via the REST interface, which allows worldwide access to the configuration files.

**Diagnostic Share**
An Azure File share can also be used to save diagnostic files like logs, metrics, and crash dumps. Having these available through both the SMB and REST interface allows applications to build or leverage a variety of analysis tools for processing and analyzing the diagnostic data.

**Dev/Test/Debug**
When developers or administrators are working on virtual machines in the cloud, they often need a set of tools or utilities. Installing and distributing these utilities on each virtual machine where they are needed can be a time consuming exercise. With Azure Files, a developer or administrator can store their favorite tools on a file share, which can be easily connected to from any virtual machine.

## Azure Storage Architecture:

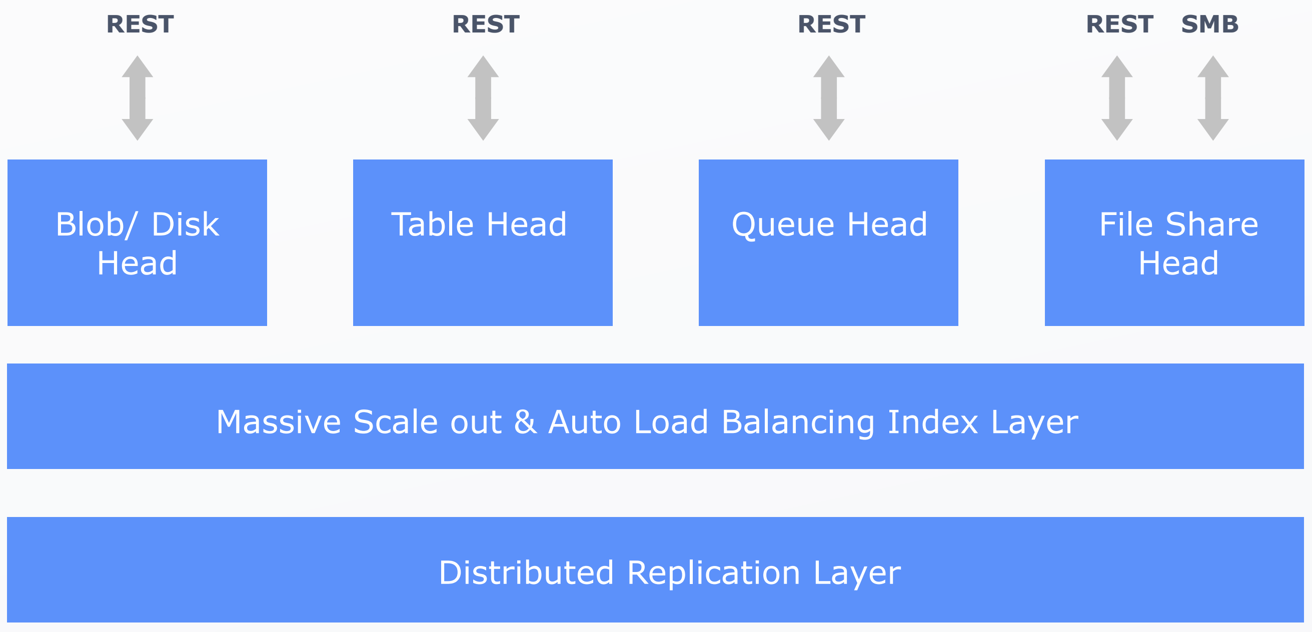
Windows Azure Storage is a distributed storage software stack built completely by Microsoft for the cloud.

**3 Layer Architecture**
The storage access architecture has the following 3 fundamental layers:

**Front-End (FE) layer** This layer takes the incoming requests, authenticates and authorizes the requests, and then routes them to a partition server in the Partition Layer. The front-ends know what partition server to forward each request to, since each front-end server caches a Partition Map. The Partition Map keeps track of the partitions for the service being accessed (Blobs, Tables or Queues) and what partition server is controlling (serving) access to each partition in the system.

**Partition Layer** This layer manages the partitioning of all of the data objects in the system. All objects have a partition key. An object belongs to a single partition, and each partition is served by only one partition server. This is the layer that manages what partition is served on what partition server. In addition, it provides automatic load balancing of partitions across the servers to meet the traffic needs of Blobs, Tables and Queues. A single partition server can serve many partitions.

**Distributed and replicated File System (DFS) Layer** This is the layer that actually stores the bits on disk and is in charge of distributing and replicating the data across many servers to keep it durable. A key concept to understand here is that the data is stored by the DFS layer, but all DFS servers are (and all data stored in the DFS layer is) accessible from any of the partition servers.


REST
REST
REST
REST
SMB
Blob/ Disk Head
Table Head
Queue Head
File Share Head
Massive Scale out & Auto Load Balancing Index Layer
Distributed Replication Layer

## Azure Files Vs Blobs

| Description | Azure Blobs | Azure Files |
|---|---|---|
| Durability Options | LRS,ZRS, GRS (RA- GRS for higher availability) | LR , GRS |
| Accessibility | REST APIs | SMB 2.1, REST APIs |
| Connectivity | REST- Worldwide | SMB 2.1- Within region REST- Worldwide |
| Directories | Flat namespace however prefix listing can simulate virtual directories | True directory objects |
| Case Sensitivity of Names | Case Sensitive | Case sensitive, but case preserving |
| Capacity | Up to 500TB containers | 5TB file shares |
| Throughput | Up to 60 MB/s per blob | Up to 60 MB/s per share |
| Object Size | Up to 1 TB/Blob | Up to 1 TB/File |
| Billed Capacity | Based on bytes written | Based on file size |

## Azure Files Vs Disks

| Description | Azure Blobs | Azure Files |
|---|---|---|
| Relationship with Azure VMs | Required for booting (OS Disk) | LR , GRS |
| Scope | Exclusive/ Isolated to a single VM | Shared across multiple VMs |
| Snapshots and Copy | Yes | No |
| Configuration | Configured via portal/ management APIs and available at boot time | Connect after boot (via net use on windows) |
| Built in authentication | Built in authentication | Set up authentication on net use |
| Cleanup | Resources can be cleaned up with VM if needed | Manually via standard file APIs or REST APIs |
| Access via REST | Can only access as fixed formatted VHD (single blob) via REST. Files stored in VHD cannot be accessed via REST | Individual files stored in share are accessible via REST |
| Max size | 1 TB | 5 TB File Share<br>1 TB File within Share |
| Max 8kb IOps | 500 IOps | 1000 IOps |

## Replication for Durability and High Availability

Data in your storage account is replicated to ensure durability that is also highly available, meeting the Azure Storage SLA even in the face of transient hardware failures. Azure Storage is deployed in 15 regions around the world and also includes support for replicating data between regions. You have several options for replicating the data in your storage account:

**Locally redundant storage (LRS)** maintains three copies of your data. LRS is replicated three times within a single facility in a single region. LRS protects your data from normal hardware failures, but not from the failure of a single facility. LRS is offered at a discount.

**Zone-redundant storage (ZRS)** maintains three copies of your data. ZRS is replicated three times across two to three facilities, either within a single region or across two regions, providing higher durability than LRS. ZRS ensures that your data is durable within a single region. ZRS provides a higher level of durability than LRS. ZRS is currently available only for blobs. Once you have created your storage account and selected zone redundant replication, you cannot convert it to use to any other type of replication or vice versa.

**Geo-redundant storage (GRS)** is enabled for your storage account by default when you create it. GRS maintains six copies of your data. With GRS, your data is replicated three times within the primary region, and is also replicated three times in a secondary region hundreds of miles away from the primary region, providing the highest level of durability. In the event of a failure at the primary region, Azure Storage will failover to the secondary region. GRS ensures that your data is durable in two separate regions.

**Read-access geo-redundant storage (RA-GRS)** provides all of the benefits of geo-redundant storage noted above, and also allows read access to data at the secondary region in the event that the primary region becomes unavailable. Read-access geo-redundant storage is recommended for maximum availability in addition to durability.

For more information and details, please feel free to reach us at Info@sysfore.com and our sales consultants will be in touch with you.