Windows Azure Drive

Brad Calder and Andrew Edwards

February 2010

Table of Contents

[1 Introduction 1](#_Toc252485426)

[2 Windows Azure Drive Overview 2](#_Toc252485427)

[3 Windows Azure Page Blob as a Drive 3](#_Toc252485428)

[4 Selecting the Guest OS 3](#_Toc252485429)

[5 Windows Azure Drive Local On-Disk Cache 4](#_Toc252485430)

[5.1 Windows Azure VM Size 4](#_Toc252485431)

[5.2 Role Local Resource 4](#_Toc252485432)

[5.3 Cache Directory 4](#_Toc252485433)

[5.4 Per Drive Cache 5](#_Toc252485434)

[6 Using the Windows Azure Drive in an Azure Application 5](#_Toc252485435)

[6.1 Credentials to Access the Page Blob as a Drive 7](#_Toc252485436)

[6.1.1 Account and Key 7](#_Toc252485437)

[6.1.2 Shared Access Signatures 8](#_Toc252485438)

[6.2 Creating and Mounting a Drive Example 8](#_Toc252485439)

[7 How the Windows Azure Drive Works 9](#_Toc252485440)

[8 Usage Patterns and Best Practices 9](#_Toc252485441)

[8.1 Application Drive Allocation, Failover, and Upgrade 9](#_Toc252485442)

[8.1.1 Drive Allocation 10](#_Toc252485443)

[8.1.2 Failover 10](#_Toc252485444)

[8.1.3 Upgrade 11](#_Toc252485445)

[8.2 Snapshotting a Drive and Using Snapshots 11](#_Toc252485446)

[8.3 NTFS Semantics and ChkDsk 12](#_Toc252485447)

[8.4 Using Windows Azure Drive within a Geo-Location 12](#_Toc252485448)

[8.5 Known Issues 12](#_Toc252485449)

[8.6 Troubleshooting 13](#_Toc252485450)

[9 Summary 13](#_Toc252485451)

# Introduction

Windows Azure is the foundation of Microsoft’s Cloud Platform. It is the “Operating System for the Cloud” that provides essential building blocks for application developers to write scalable and highly available services. Windows Azure provides:

* Virtualized Computation
* Scalable Storage
* Automated Management
* Rich Developer SDK

Windows Azure Storage allows application developers to store their data in the cloud, so the application can access its data from anywhere at any time, store any amount of data for any length of time, and be confident that the data is durable and will not be lost. Windows Azure Storage provides a rich set of data abstractions:

* Windows Azure Blob – provides storage for large data items.
* Windows Azure Table – provides structured storage for maintaining service state.
* Windows Azure Queue – provides asynchronous work dispatch to enable service communication.
* Windows Azure Drive – provides durable NTFS volumes for Windows Azure applications

This document describes Windows Azure Drive, and how to use it. To use Windows Azure Drives and Blobs, a user needs to create a storage account via the Windows Azure Developer Portal. The user will receive a 256-bit secret key once the account is created. This secret key is then used to authenticate user requests to the storage system as described in this document.

# Windows Azure Drive Overview

Customers have told us that one of the challenges is taking their already running Windows applications and running them in the cloud while making sure their data is durable while using the standard Windows NTFS APIs. With Windows Azure Drive, your Windows Azure applications running in the cloud can use existing NTFS APIs to access a durable drive. This can significantly ease the migration of existing Windows applications to the cloud. The Windows Azure application can read from or write to a drive letter (e.g., X:\) that represents a durable NTFS volume for storing and accessing data. The durable drive is implemented as a [Windows Azure Page Blob](http://msdn.microsoft.com/en-us/library/ee691964.aspx) containing an NTFS-formatted Virtual Hard Drive (VHD).

The Page Blob can be mounted as a drive only within the Windows Azure cloud, where all non-buffered/flushed NTFS writes are made durable to the drive (Page Blob).   If the application using the drive crashes, the data remains persistent via the Page Blob, and can be remounted when the application instance is restarted or remounted elsewhere for a different application instance to use.   Since the drive is an NTFS formatted Page Blob, you can also use the standard blob interfaces to upload and download your NTFS VHDs to the cloud.

For the Beta Release of Windows Azure Drive, the billing for using a drive is only in terms of the storage space used by the Page Blob and the read/write transactions to the Page Blob. Therefore, the billing for using the drive during Beta is completely incorporated into the Windows Azure Storage billing for the user’s storage account and subscription.

Windows Azure Drive can optionally cache the drive data on a local disk on the VM. Caching the data on the local drive will reduce the read traffic to the page blob, which will reduce the transaction cost. This is because there is no additional transaction charge for reads that are to the local disk cache, whereas the transactions against the Page Blob are counted towards billing. Note, even when the cache is enabled, all non buffered and flushed writes are committed transactions to the Page Blob in durable storage.

# Windows Azure Page Blob as a Drive

Windows Azure Blob now provides a new type of blob called a Page Blob. A Page Blob is optimized for random read/write workloads, making it particularly suited for being used as a drive. Each Page Blob consists of an array of pages, and each page is identified by its offset from the start of the blob. The size limit on a Page Blob is 1TB, and hence the size limit of a Windows Azure Drive is also 1 TB.

One benefit of Page Blobs is that a storage account is only charged for the pages that are stored. For instance, you can allocate a 1TB Page Blob and if you are only storing 1GB of pages in the blob, then you are only charged for 1GB of storage. This means for a large durable drive, you are only charged for the VHD metadata overhead, the metadata maintained for formatting and using the Page Blob as an NTFS volume, and the data pages in the blob stored for the NTFS VHD.

Windows Azure Page Blobs also support the following functionality, which Windows Azure Drive takes advantage of:

* **Lease Blob** – This allows an exclusive modify lease to be taken on a blob, which allows the client that knows the lease-id to be the only one that can update the blob while the lease is held. The ability to renew the lease is also provided, in order to allow long leases to be held. Lease Blob is what Windows Azure Drive uses to maintain exclusive access to a mounted NTFS formatted Page Blob. This means that a Page Blob can only be mounted by a single VM instance at a time.
* **Snapshot Blob** – Allows the creation of read-only versions of a blob, which can be used for creating blob backups or blob versioning. Snapshots are read-only and an account is only charged for the unique pages; pages shared across snapshots and the base blob from which they were derived do not accrue additional storage charges.
* **Copy Blob** – This provides the ability to copy a blob or a snapshot from one blob name to a new blob name within a storage account. When you copy a snapshot to a new blob, the new blob is read/writeable.

# Selecting the Guest OS

Windows Azure Drive is now accessible starting with the release of Windows Azure Guest OS 1.1. In order to mount a Windows Azure Drive you will need to use Feb 2009 version or later of the Windows Azure SDK and the Windows Azure Guest OS 1.1 or later. [See here for how to configure your service to use Guest OS 1.1](http://msdnstage.redmond.corp.microsoft.com/en-us/library/ee907419.aspx) or later.

# Windows Azure Drive Local On-Disk Cache

Windows Azure Drive provides an optional local on-disk cache of the drive’s data. Employing the cache is useful to reduce the transaction cost of accessing the blob service. The local on-disk cache is allocated from the VM’s local resources in several steps.

## Windows Azure VM Size

Windows Azure provides four different VM sizes, named small, medium, large, and extra large. The local drive space allocated to each of these VM sizes is 250GB, 500GB, 1000GB, and 2000GB, respectively. The local on-disk cache space used by a Windows Azure Drive will be allocated from this drive space. Your application will choose which size to use with a setting in the role’s service \*.csdef definition file:

 <ServiceDefinition name="<service-name>"

 upgradeDomainCount="<number-of-upgrade-domains>">

 <WebRole name="<web-role-name>"

 vmsize="Small|Medium|Large|ExtraLarge">

## Role Local Resource

From the local disk space allocated to the VM, the role can create named “local resources” that are available for the role to use as a path to the drive (e.g., “z:\”). To use a local on-disk cache for the Windows Azure Drive, you should create a “LocalStorage” local resource. The following example creates a local resource called “MyAzureDriveCache” that is 220GB in size:

<LocalResources>

 <LocalStorage name="MyAzureDriveCache"

 cleanOnRoleRecycle="false"

 sizeInMB="220000" />

</LocalResources>

Setting “CleanOnRoleRecycle” to false means that when the role is restarted by the Windows Azure Fabric after a crash e the local storage will not be wiped.

## Cache Directory

The application is responsible for retrieving the local resource from the Windows Azure library, and passing that resource to the Windows Azure Drive. The API to retrieve the local resource is:

 LocalResource localCache =

 RoleEnvironment.GetLocalResource("MyAzureDriveCache");

The application then needs to tell the VM instance the path for where to store the cache and the maximum amount of local disk space to use for the cache, as shown here:

 CloudDrive.InitializeCache(localCache.RootPath,

 localCache.MaximumSizeInMegabytes);

Note, this local cache disk space is shared among all of the mounted drives by the application on that VM instance that want to use the local on-disk cache.

## Per Drive Cache

When mounting a drive, you can specify the maximum amount of the total cache space to use for a particular drive. For all of the drives mounted for a VM instance, the total cache space requested by all of the mounted drives has to sum up to be less than or equal to the amount of cache space set aside with InitializeCache call described above. Using the example above, after creating the 220GB of local on disk cache using InitializeCache, we can then choose to mount one drive Blob1 using 200GB of local on disk cache, one drive Blob2 using 20GB of local on disk cache, and one drive Blob3 with no on-disk cache, as shown here:

CloudDrive drive1 =

 new CloudDrive(new Uri(“http://account.blob.core.windows.net/container/Blob1”), credentials);

CloudDrive drive2 =

 new CloudDrive(new Uri(“http://account.blob.core.windows.net/container/Blob2”), credentials);

CloudDrive drive3 =

 new CloudDrive(new Uri(“http://account.blob.core.windows.net/container/Blob3”), credentials);

drive1.Mount(200000, DriveMountOptions.None);

drive2.Mount(20000, DriveMountOptions.None);

drive3.Mount(0, DriveMountOptions.None);

Note that the local on-disk cache will grow dynamically up to the total space allocated for that drive as the drive is used. Thus, we do not recommend sharing the local on disk cache local resource (drive) with other data from your application.

In addition, there can be some workloads that perform better without using a local on-disk cache, so it may be worthwhile to test the performance of your application with and without (setting the cache size to 0 when mounting) the local on-disk cache.

# Using the Windows Azure Drive in an Azure Application

In this section, we give an overview of the programming interface for using Windows Azure Drive. Please see the Windows Azure Drive client library for complete details.

The following is the list of APIs provided for your Windows Azure application to use Windows Azure Drives:

* **Create Drive** - Creates a Page Blob formatted as a single partition NTFS volume VHD.
	+ When creating a drive, you specify the URL of the page blob and the credentials needed for the client library to access the drive. The container has to already exist, but the blob cannot exist. If the blob already exists, when calling create, you will get an error. The URL will be of the form:
		- [http://<accountname>.blob.core.windows.net/<containername>/<blobname>](http://..../blobname)
	+ When creating the drive you pass in the size of the NTFS drive to create. The maximum drive size allowed is 1 TB, and the minimum size is 16 MB. Note that the VHD and NTFS formats have some overhead, so you will not be able to use the full 1TB.
	+ The drive is created as a “Fixed” VHD inside the Page Blob. Even though it is a Fixed VHD, only a small fraction of the Page Blob pages have data stored in them when the formatting of the drive is complete. This saves storage capacity, since your account is only charged for the pages that are used for the Page Blob.
* **Mount Drive** – This takes a formatted Page Blob and mounts it to a drive letter for the Windows Azure application to start using.
	+ A Page Blob can only be mounted by a single VM/application at a time.
	+ The Page Blob must be formatted as a Fixed VHD with a single volume NTFS partition.
	+ Before mounting the drive you create a CloudDrive object specifying the URL of the page blob and the credentials needed for the client library to access the drive. The URL will be of the form:
		- [http://<accountname>.blob.core.windows.net/<containername>/<blobname>](http://..../blobname)
	+ Then when mounting the drive you can specify the maximum amount the VM’s local disk cache space to use as a cache for the mounted drive as described above. You can specify anywhere from 0 (meaning do not use the local disk cache) to MAX(size of mounted Page Blob drive, max size of local resource disk space allocated minus the disk space already allocated for other mounted drives).
	+ When the mount drive completes successfully it returns the drive letter the page blob was mounted to chosen by the system.
	+ A snapshot can be mounted read-only. The only read-only mounts allowed are for snapshot Page Blobs.
	+ A VM can dynamically mount up to 16 drives.
* **Initialize Cache** – This allows an application to specify the location and size of the local data cache for all Windows Azure Drives mounted for that VM instance.
	+ The location should be created using a LocalStorage entry in the service definition (\*.csdef) of your application, retrieved on the VM using the appropriate API, and then passed to this API.
	+ The size will limit the amount of total local disk space a VM will be able to use for all mounted drives for caching.
* **Get Mounted Drives** – This returns the list of mounted drives. It consists of a list of the drive letter and Page Blob URLs for each mounted drive.
* **Unmount Drive** – This is used to unmount the drive.
	+ This releases the Page Blob lease and deallocates the drive letter.
* **Snapshot Drive** – This allows the client application to create a backup of the drive (Page Blob).
	+ Snapshotted drives can later be copied to become a read/writeable version of the drive if need be.
	+ A snapshot can be made of a drive while it is mounted.
* **Copy Drive** – This provides the ability to copy a snapshot or an unmounted drive to another drive (Page Blob) name to be used as a read/writable drive.
	+ Mounted drives cannot be copied directly. To copy from a mounted drive, snapshot it, then copy from the snapshot. Afterward, delete the snapshot.
	+ At this time, Copy Drive only works with Account/Key credentials (see below), and does not work with Shared Access Signatures (SAS).

## Credentials to Access the Page Blob as a Drive

When mounting a drive, the application has to specify credentials that allow it to access the Page Blob in the Windows Azure Blob service. Windows Azure Drive support two different authorization schemes, account and key, as well as Shared Access Signatures (SAS).

### Account and Key

To use a Windows Azure Drive with the account name and key as credentials, you construct a StorageCredentialsAccountAndKey object and specify your account and key.

StorageCredentialsAccountAndKey credentials =

 new StorageCredentialsAccountAndKey("myaccount", storageAccountKey);

You can then use these credentials to create a CloudDrive object, and then perform operations on the cloud drive.

CloudDrive drive = new CloudDrive(blobURI, credentials);

drive.Mount(200000, DriveMountOptions.None);

With this approach, Windows Azure Drive generates temporary Shared Access Signatures (see below) from your private key to allow the OS driver to access the Page Blob. An important point here is that your private key is kept only within the process that called Mount(), and the OS driver accesses it from there to **create** the signatures. However, this means that the process that calls Mount() must be running while the drive is mounted. If that process exits, the next instance of the process needs to call Mount() again with the same parameters to keep the drive mounted. If all processes that have called Mount() for a drive exit, then the drive will stay mounted for only an additional 20 to 30 minutes.

### Shared Access Signatures

Another way to mount a drive is to directly pass in a Shared Access Signature (SAS). There are a number of ways to create a shared access signature, please see here for the details:

<http://msdn.microsoft.com/en-us/library/ee395415.aspx>

Once one is constructed, you then pass in the full URL with the SAS for the Page Blob to be mounted as shown here:

CloudDrive drive = new CloudDrive(**SAS-URI**, null);

drive.Mount(200000, DriveMountOptions.None);

When using a shared access signature, it is not required to keep the process that called mount around, but the drive will only be able to stay mounted for as long as the SAS is valid. To maintain a mount longer than an hour when using a SAS as credentials, you would need to create the SAS using a signed-identifier as described in the above link.

## Creating and Mounting a Drive Example

The following gives a short code example to create and mount a drive, putting all of the parts together.

 // Create a storage account object with the account and key

 StorageCredentialsAccountAndKey credentials =

 new StorageCredentialsAccountAndKey("myaccount", storageAccountKey);

  try

  {

 CloudDrive.InitializeCache(localCache.RootPath,

 localCache.MaximumSizeInMegabytes);

   CloudDrive drive = new CloudDrive(BlobURI, credentials);

 drive.Create(sizeOfDrive);

      string driveLetter = string.Empty;

      driveLetter = drive.Mount(cacheSize, DriveMountOptions.None);

  }

  catch (CloudDriveException e)

  {

Console.WriteLine(e.StackTrace);

Throw;

  }

# How the Windows Azure Drive Works

The Windows Azure operating system has an OS driver that serves two primary functions. The first is to provide the client library operations described above for mounting and unmounting a drive. The second is performing I/O for all of the NTFS operations to the mounted drive.

The following describes at a high level how the drive works during execution:

* **Mount** – When mounting the drive, the OS Driver attempts to acquire an exclusive access lease on the Page Blob. If it can, then it will mount the drive to a free drive letter. If cache space is specified to be used, then local disk space will be reserved for the mounted drive to use as an on-disk local cache.
* **Write**  – All non-buffered/flushed NTFS writes are sent to the OS driver, where the write is made durable in the Page Blob in the Windows Azure Blob service before sending success back to the application. This ensures that if application gets a success back for the write, and the drive is unmounted or the application crashes, the application knows that the write was persisted to the Page Blob. When doing the write, if the data pages are already in the local on-disk cache they will also be updated in the cache. If the data pages are not in the cache, then they will not be allocated into the cache. We only allocate on read miss into the local on-disk cache.
* **Read** – All reads to the drive go through the OS Driver, which will read the data either from the local cache or the Page Blob in the Windows Azure Blob service. If the cache is being used, the driver will first look in the local disk cache, and if found there the results are returned to the application without talking to the blob service. If the data is not found in the local on-disk cache, then it will be retrieved from the blob service and returned to the application as well as being placed in the local on-disk cache.
* **Lease Renewal** – While a drive is mounted and in use for an application instance, under the covers the OS driver continuously renews the lease on the Page Blob that is mounted as the drive, using the credentials passed in when mounting the drive. The lease has to be maintained in order for the mounted drive to be used.
* **Unmount** – When a drive is unmounted, the lease on the Page Blob is released (freed) and the drive letter is unallocated. Releasing the lease allows another VM/application to quickly mount the Page Blob as a drive if it chooses.

# Usage Patterns and Best Practices

We now go through common usage patterns and best practices when using Windows Azure Drive.

## Application Drive Allocation, Failover, and Upgrade

A key aspect for using Windows Azure Drive is how to deal with drive allocation and failover, and application upgrade.

### Drive Allocation

Drive allocation refers to the fact that a formatted Page Blob can only be mounted by a single VM at a time for read/write access. This often means that the Leader or Manager in the service will allocate each drive used to a given application instance. We refer to an application instance as a single instance of part of the application (often called a role) running on a given VM instance for the service. It can be the responsibility of this Manager to assign drives to the given role instances, and potentially even ping the role instances to make sure they are still running and hosting the data for the drives allocated to them.

Alternatively, one could have a service where each application instance lists the service’s Page Blobs that exist for its account, and then use some method for assigning the drives on a per application instance basis (e.g., each role instance could go through all of the drives in the list trying to mount each one until a role instance finds a free one to serve).

There are many ways to perform drive allocation, and it is application dependent. No matter what the approach used, once the drives have been allocated, the next key concept the service has to handle is failovers when using the drives and maintaining availability of the application’s service.

### Failover

There are two common failover scenarios to deal with – application instance failover and application node migration. The first is the typical type of failover, and the second is very uncommon, but can occur.

Application instance failover refers to the situation when the application using the mounted drive crashes, typically due to an application bug. When this occurs, the Windows Azure Fabric will quickly restart the application instance on the same VM instance. In this situation, the VM instance will most likely have the drive still mounted and still have the lease on the Page Blob. Therefore, it is best practice for when the application instance restarts to try to remount and use the same Page Blob drive. When this is done, the common result will be that the mount will be successful, and the same drive letter would be returned. This allows the application to leverage the warmed up local on-disk cache that has already been populated on that VM instance.

An uncommon failure scenario is when an application instance has to be migrated to a different VM instance on failover. It is uncommon, but it can occur, so applications should decide how to best deal with this scenario for the availability of their service. In this situation, the application instance will be brought up on a different VM instance by the Windows Azure Fabric. When this occurs, it could actually be several minutes until the application instance comes back up on the new VM instance. This means the service has the following choices for how to deal with this uncommon failover. Since it is uncommon, the service may choose to just wait until the application instance comes back up, mount the drive there and continue using the drive from that point on. For higher availability, the service may instead have a Manager (as described above) that monitors when the application instance is no longer available. It could then assign the drive to an already running application instance to reduce the amount of time that drive is offline. In this scenario, when reassigning the drive to be mounted by a different application instance, the drive (Page Blob) may still be mounted at the prior VM instance A. Therefore, when mounting the drive at a new VM instance B, the new application instance should pass in the “Force” option when performing the mount. This option forces the Page Blob lease to be released, if the prior VM instance A still has it. This will result in the existing lease on the blob (if there is one by the prior VM instance A) to not be renewed, which will result in that prior VM instance A’s mount lease to expire. Then the new “Force” mount request from VM instance B will acquire the Page Blob lease, mount the drive, and allow access to it. Please note that the VM instance A will only be notified that it has lost the lease once it has done a flushed/unbuffered IO, which will then fail. If possible, we recommend that VM instance A unmount the drive when it knows it has lost its lease.

### Upgrade

Windows Azure provides applications two types of upgrades – VIP Swap and Rolling Upgrade. Both can be used with Windows Azure Drive, but when using VIP swap this means that all of the drives have to be moved to new VMs under the new VIP, and the already populated local on-disk cache will be lost when moving the drives. Therefore, when using Windows Azure Drive, an application typically uses rolling upgrades for the application instances with mounted drives.

Rolling upgrade breaks the application’s instances up into N upgrade domains. For example, if the service has 20 instances of a worker role, and 5 upgrade domains, then it breaks the 20 worker roles into 5 upgrade domains (UD0, UD1, UD2, UD3, and UD4) with each domain consisting of 4 worker roles. Then when performing the upgrade, each domain is upgraded separately before going onto the next upgrade domain. When the application roles in the upgrade domain are upgraded, the new binaries will be placed on each application instance’s VM, and then the application instance will be restarted running the new version of the code. Then as described above, when the application instance restarts it should remount the drive and continue where it left off.

## Snapshotting a Drive and Using Snapshots

A quick and efficient way to perform a backup of your drives is to use the Snapshot Drive command provided with the Windows Azure Drive client library. This is usually used to create a snapshot of a drive to allow execution to continue from that snapshot at some point in the future. This could be used to roll back execution to that prior version or to potentially share the state of the drive with some other application instances.

In order to get a consistent view of the data for your application, when using snapshot, it is recommended that your application have the notion of having a write barrier to the drive, and then performing a flush before calling snapshot. For example, if using a database on a mounted drive, one could have a write barrier that locks all of the tables in the database, followed by a flush of all pending database transactions, and then do a snapshot. Then once the snapshot has finished, unlock the tables to allow execution to continue. Similar types of write barriers can potentially be put into your application. If not, then one can use the Windows Azure Drive Snapshot functionality as is, and it may provide a consistent enough view of the data for your application to continue from a snapshot.

Snapshots also provide a way of sharing drive data among your service’s application instances. After you have made a successful snapshot, the snapshot drive can be copied to another Page Blob name for another application instance to use as a read/writeable drive. In addition, we allow snapshots themselves to be mounted as read only drives, and a single snapshot can be mounted as a read-only drive by many VMs at the same time.

## NTFS Semantics and ChkDsk

When using a Windows Azure Drive, standard NTFS semantics apply. You can think of the durable drive as an NTFS disk, which means NTFS by default will buffer updates in memory before they are written to the underlying drive. Just like with a standard NTFS drive, only flushed writes or non-buffered writes are pushed to the drive where they are written and then made persistent. NTFS naturally flushes the writes after its buffers fill up, or the application can force a flush by issuing an NTFS flush command. In addition, certain applications such as databases typically have options to run in a “non-buffered” mode. In this situation, all writes are non-buffered, so when the database gets a success back from a write, it knows that the write has made it to the drive.

One issue to keep in mind is that if the drive is not cleanly unmounted, then NTFS may not be in a completely consistent state when the drive is mounted later. An option is provided when mounting a drive, to run ChkDsk on the drive. The option is off by default.

## Using Windows Azure Drive within a Geo-Location

When creating your compute and storage accounts in the Windows Azure Developer portal you can choose which geo-location to host your accounts. For example, within the US you can choose “US North” or “US South”.

When using Windows Azure Drive you want to make sure in the Windows Azure Developer Portal that your storage account used to store the Page Blob and the compute account used to run the application are located in the same geo-location (e.g., “US North”). This is important because it allows the best performance out of a drive by decreasing the latency between the VM instance using the drive and the blob service where the Page Blob is stored.

## Known Issues

It has been reported, that when a mounted drive loses its lease, subsequent calls to mount will succeed even though the drive is not available. Therefore, as best practice, we recommend after mounting a drive, applications should open a file with FILE\_WRITE\_THROUGH on the drive and write to it. If the write succeeds, then the mount was successful. If not, then the application should unmount the drive and repeat the mount.

## Troubleshooting

My application isn’t working, what should I do now?

* Have you called InitializeCache? InitializeCache is required, even if you aren’t going to use the cache. All CloudDrive APIs will fail if you have not properly called IntializeCache.
* Are you running the devFabric? The devFabric simulates CloudDrives by using “subst”. You cannot mount a drive stored in the cloud from within the devfabric, instead you will get a simulated drive.
* Did you create the container for your drive? CloudDrive.Create will fail if you have not created the container.
* A CloudDrive needs to be created as a fixed VHD. If you create a dynamic VHD, and upload that, it will fail to mount.

# Summary

Windows Azure Drive provide Windows Azure applications the ability to use a durable drive to store and access their data using standard Windows NTFS APIs.

The following summarizes some high level points to remember when using a Windows Azure Drive:

* A Drive is a Page Blob formatted as a Fixed NTFS VHD between 16 MB and 1TB.
* A VM can dynamically mount up to 16 drives.
* A Page Blob can only be mounted by only one VM at a time for read/write access.
* The storage account used to store the Page Blob and the compute account used to run the application should be allocated in the same geo-location in the Windows Azure Developer Portal in order to get the best performance out of the drive.
* Drives can be uploaded or downloaded via the Windows Azure Blob interface. The formatted Fixed NTFS VHDs can be uploaded as a Page Blob using the blob interface and then mounted as a drive. Similarly, a formatted Page Blob can be downloaded through the Page Blob interface and then the downloaded drive can be used remotely as a VHD.
* The OS driver provided by Windows Azure is only available within the Windows Azure VM instances in the Windows Azure cloud. No driver is distributed by Microsoft at this time to allow you to mount drives remotely. Even so, you can download a drive using use the Windows Azure Blob interfaces and mount the downloaded VHD drive where it is downloaded for use, but any updates to that downloaded version will not be sent to the Windows Azure Blob service.
* Drives can be backed up and shared via Snapshot and Copy.