December 19, 2014

Version 1.0

Azure Data Center Migration Solution

User Guide

www.persistentsys.com

Contents

[1. Introduction 4](#_Toc409714746)

[1.1 Solution architecture 4](#_Toc409714747)

[1.1.1 Metadata export 5](#_Toc409714748)

[1.1.2 Deployment import 6](#_Toc409714749)

[1.1.3 Migration 7](#_Toc409714750)

[1.2 When to use this solution 8](#_Toc409714751)

[1.3 Limitations of this version 8](#_Toc409714752)

[1.4 Use of the Microsoft Azure Management Libraries 9](#_Toc409714753)

[1.5 How this solution was tested 9](#_Toc409714754)

[2. Easing the challenge of migration: The Data Center Migration Solution 10](#_Toc409714755)

[2.1 Atomicity… 10](#_Toc409714756)

[2.2 Resumption 11](#_Toc409714757)

[2.3 Compatibility 12](#_Toc409714758)

[2.4 Consistency 13](#_Toc409714759)

[2.5 Performance 14](#_Toc409714760)

[2.6 Extensibility 16](#_Toc409714761)

[3. Getting started 17](#_Toc409714762)

[3.1 System requirements 17](#_Toc409714763)

[3.2 Download publish settings file 17](#_Toc409714764)

[3.2.1 Using Azure Portal 17](#_Toc409714765)

[3.2.2 Using Microsoft Azure PowerShell 18](#_Toc409714766)

[3.3 Set up Azure management certificate 20](#_Toc409714767)

[3.4 Configuration parameters 20](#_Toc409714768)

[3.4.1 Export operation 20](#_Toc409714769)

[3.4.2 Import operation 22](#_Toc409714770)

[3.4.3 Migrate operation 25](#_Toc409714771)

[4. Running the Azure Data Center Migration Solution 27](#_Toc409714772)

[4.1 Input parameters 27](#_Toc409714773)

[4.1.1 Using the Console application 27](#_Toc409714774)

[4.1.2 Using PowerShell cmdlets 29](#_Toc409714775)

[4.2 Logging configurations 30](#_Toc409714776)

[4.2.1 Configuration settings 30](#_Toc409714777)

[4.2.2 Log database setup 32](#_Toc409714778)

[5. Completing a migration 34](#_Toc409714779)

[5.1 After a successful migration 34](#_Toc409714780)

[5.2 If the import fails 34](#_Toc409714781)

[6. Troubleshooting 35](#_Toc409714782)

[7. Use cases 37](#_Toc409714783)

[7.1 Reserved IP configuration 37](#_Toc409714784)

[7.2 Regional VNet 37](#_Toc409714785)

[7.3 Change VM size 38](#_Toc409714786)

[7.3.1 Change the size within the same range 38](#_Toc409714787)

[7.3.2 Change the size from range 1 to range 2 39](#_Toc409714788)

[7.4 Delete resources manually 39](#_Toc409714789)

[7.5 Choose resources to migrate 41](#_Toc409714790)

[8. Using the source code 42](#_Toc409714791)

[8.1 System requirements 42](#_Toc409714792)

[8.2 Installer project dependencies 42](#_Toc409714793)

[9. Feedback 43](#_Toc409714794)

[Appendices 44](#_Toc409714795)

[A Implementation details 44](#_Toc409714796)

[A.1 .NET components of the solution 44](#_Toc409714797)

[A.2 PowerShell cmdlets 45](#_Toc409714798)

[A.3 Resume import operation 45](#_Toc409714799)

[B Resource naming 46](#_Toc409714800)

[B.1 Resource naming conventions 46](#_Toc409714801)

[B.2 Resource name mapper XML 46](#_Toc409714802)

[C Performance results 47](#_Toc409714803)

List of tables

[Table 1. Performance results for SharePoint farm with high-availability environment 14](#_Toc406762869)

[Table 2. Number of faults and retries 15](#_Toc406762870)

[Table 3. Settings for exporting information about resources 20](#_Toc406762871)

[Table 4. Settings for importing information about resources 22](#_Toc406762872)

[Table 5. Settings for migration 25](#_Toc406762873)

[Table 6. Details of Azure.DataCenterMigration cmdlets 29](#_Toc406762874)

[Table 7. Troubleshooting exceptions 35](#_Toc406762875)

[Table 8. Major components of the Azure Data Center Migration Solution 44](#_Toc406762876)

[Table 9. PowerShell cmdlets for export, import, and migrate 46](#_Toc406762877)

[Table 10. Azure Data Center Migration Solution conventions for converting names during migration 47](#_Toc406762878)

[Table 11. Performance chart of Data Center Migration Solution for various environments 48](#_Toc406762879)

1. Introduction

To ease migrations, Persistent Systems, along with the Microsoft Customer Advisory Team, has developed the open source Azure Data Center Migration Solution for automatically copying an entire infrastructure as a service (IaaS) deployment on Azure from one location to another.

With the Azure Data Center Migration Solution version 1.0, you can automate the migration of Microsoft Azure resources:

* From one subscription to another subscription in the same data center (region).
* From one subscription to another subscription in different data centers.
* In the same subscription with different data centers.
* In the same subscription with the same data center.

You can migrate all of the following resources in the source data center:

* Affinity groups
* Networks
* Cloud services
* Storage accounts
* Virtual machines (VMs)

As an open source solution, the Azure Data Center Migration Solution is customizable.

* 1. Solution architecture

The Data Center Migration Solution supports three operations: **metadata export**, **deployment import**, and **migration**. It runs from an on-premises computer or on a VM in the cloud.

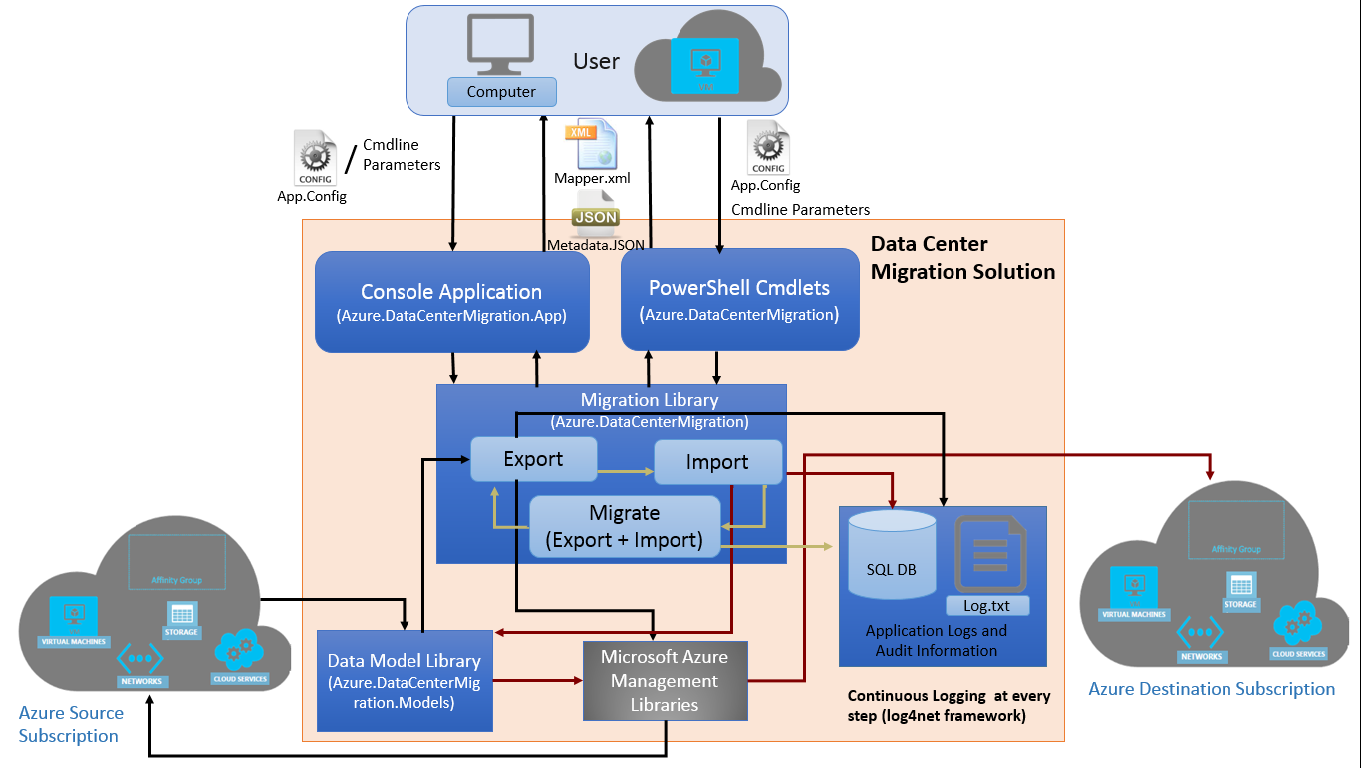


Figure 1. Data Center Migration Solution overview

The solution logs every step in a text file using the Apache log4net framework. Optionally, you can choose to log to a SQL Database. You can modify the solution’s configuration file to add new log appenders.

### Metadata export

The solution generates a JSON file containing a hierarchically structured list of information (metadata) about an Azure subscription data center. The metadata describes all the resources that are required for IaaS migration:

* Storage accounts, affinity groups, VNets, subnets
* Cloud services, VMs, availability sets, VM properties

Optionally, the solution also generates an XML file that can be used to customize names of resources in the destination.

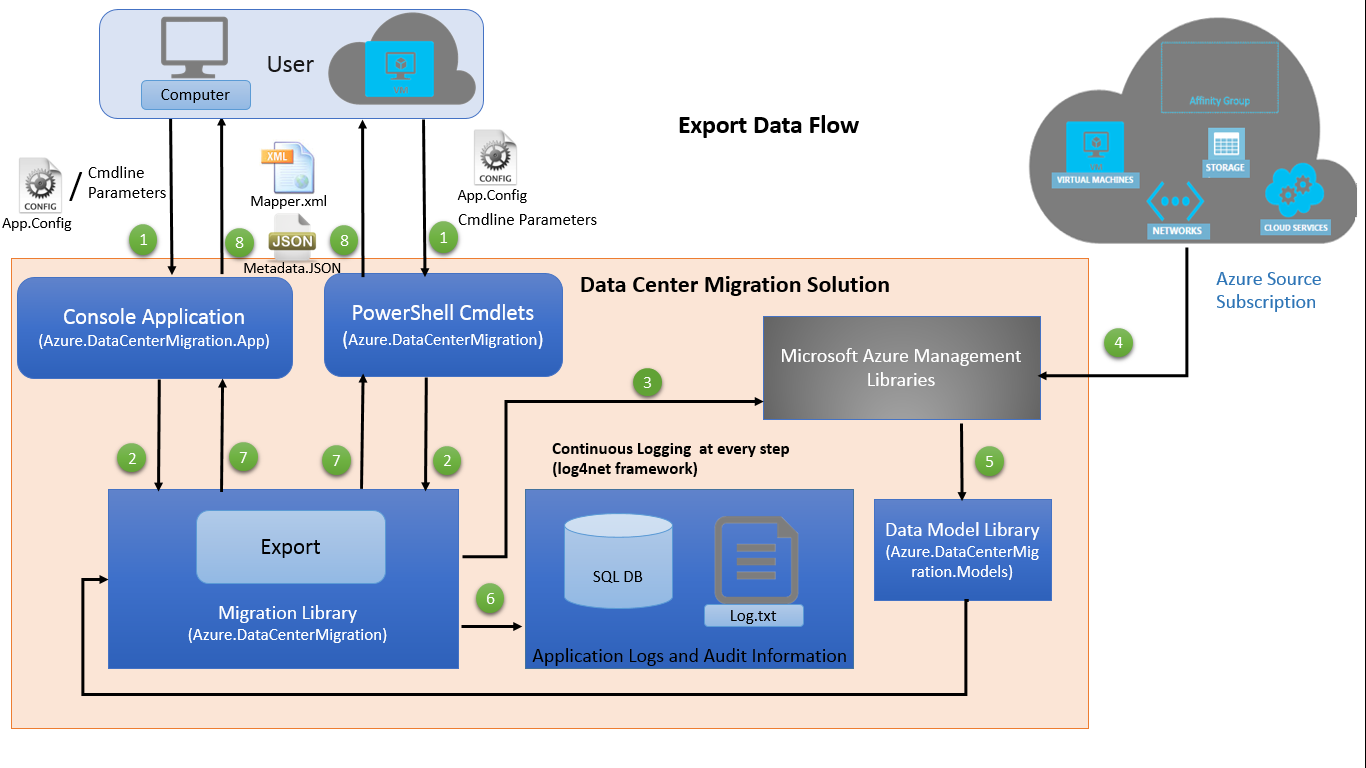


Figure 2. Export data flow

The **Export** option of the Data Center Migration Solution exports information of resources, such as affinity groups, virtual machines, storage accounts, cloud services, and virtual networks, from a data center of an Azure subscription.

The **export** operation works as follows:

1. You provide input parameters using the configuration file or using command line parameters for the console application or command line parameters for the PowerShell cmdlet.
2. Depending on your choice, either the console app or the PowerShell cmdlet calls the Migration Library function to export the metadata.
3. The Migration Library requests details about the resources from Azure using the Microsoft Azure Management Libraries.
4. Azure returns the details about the resources.
5. The Data Model Library stores the received data in memory.
6. The solution continuously logs every step.
7. The Migration Library serializes the data model to a JSON file.
8. Depending on step 1, either the console app or the PowerShell cmdlet returns a JSON file containing the information about the resources to you. If you enabled the generation of mapper XML, you also receive the source-to-destination resource name mapper XML file.

### Deployment import

The resources from the exported metadata are deployed onto the destination subscription. This operation creates a new deployment from the exported JSON file in any of the following destinations:

* Same data center, different subscription
* Different data center, same subscription
* Different data center, different subscription
* Same data center, same subscription (useful for test or rename scenarios)

You can specify the names of the resources to use in the resource name mapper XML file or specify a prefix to use. You can also customize the metadata file to support advanced scenarios that are not supported directly by the solution.

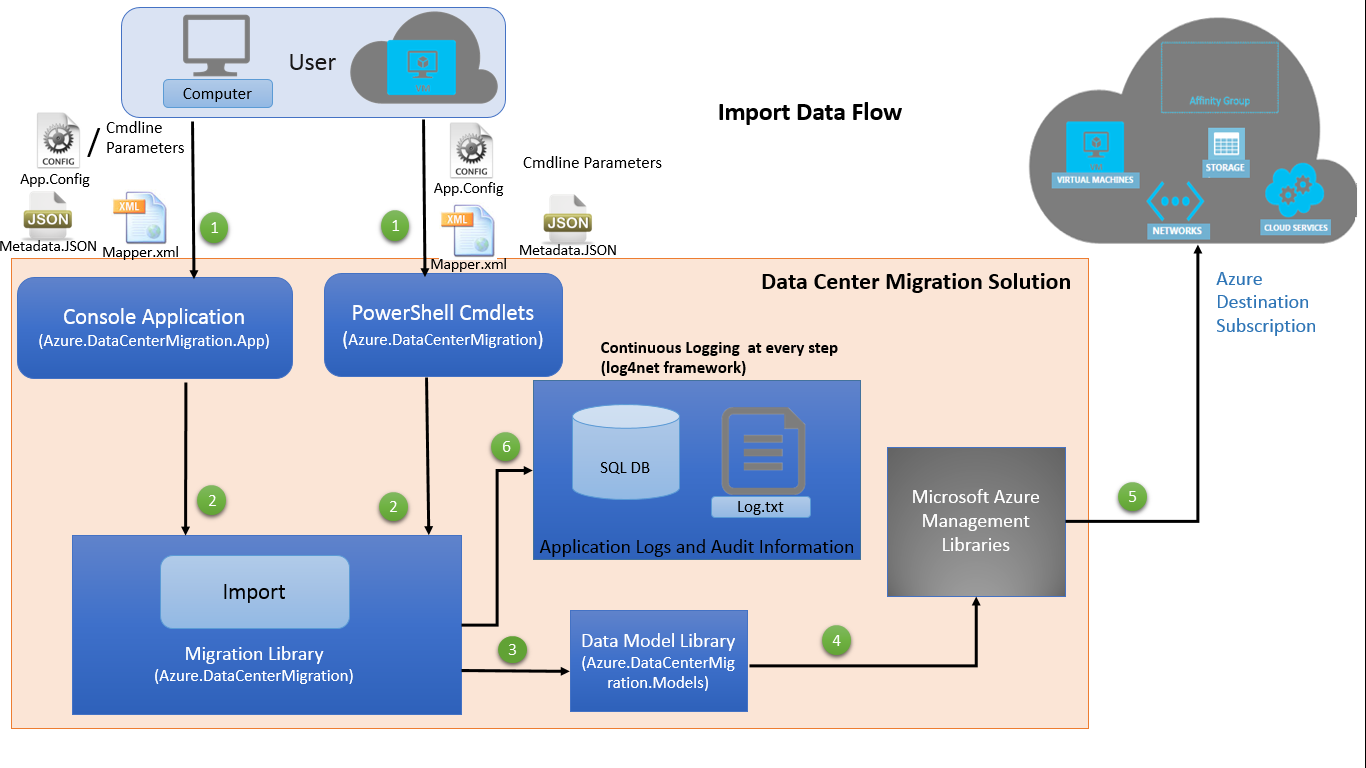


Figure 3. Import data flow

**NOTE:** Resources created or deleted after the exported metadata file was generated are not imported in the ongoing run.

During import, the Data Center Migration Solution deploys resources—affinity groups, VMs, storage accounts, cloud services, VNets—to a destination Azure subscription. To import the resources, it uses the JSON file containing metadata that was created during the **export** operation and, optionally, the XML file containing the map of source-to-destination names.

The **import** operation works as follows:

1. You provide the JSON file, source-destination name mapper XML file (optional), and the configuration file / command line parameters for the console application or the command line parameters for the PowerShell cmdlet as an input to the solution.
2. Depending on step 1, either the console app or the PowerShell cmdlet passes the metadata to the Migration Library.
3. The Migration Library deserializes the JSON-formatted metadata into data models using the Data Model Library.
4. The information about the resources is passed to the Microsoft Azure Management Libraries.
5. The Azure Management Libraries deploy the resources on the destination Azure subscription.
6. The solution continuously logs every step.

### Migration

You can use the **Migrate** option of the solution to execute export and import in a single run. The solution exports the Azure subscription resources, including affinity groups, virtual machines, storage accounts, cloud services, and VNets, into a metadata file and deploys them on the destination subscription. This option is useful when you don’t need any customization in the JSON file or in the naming of resources.

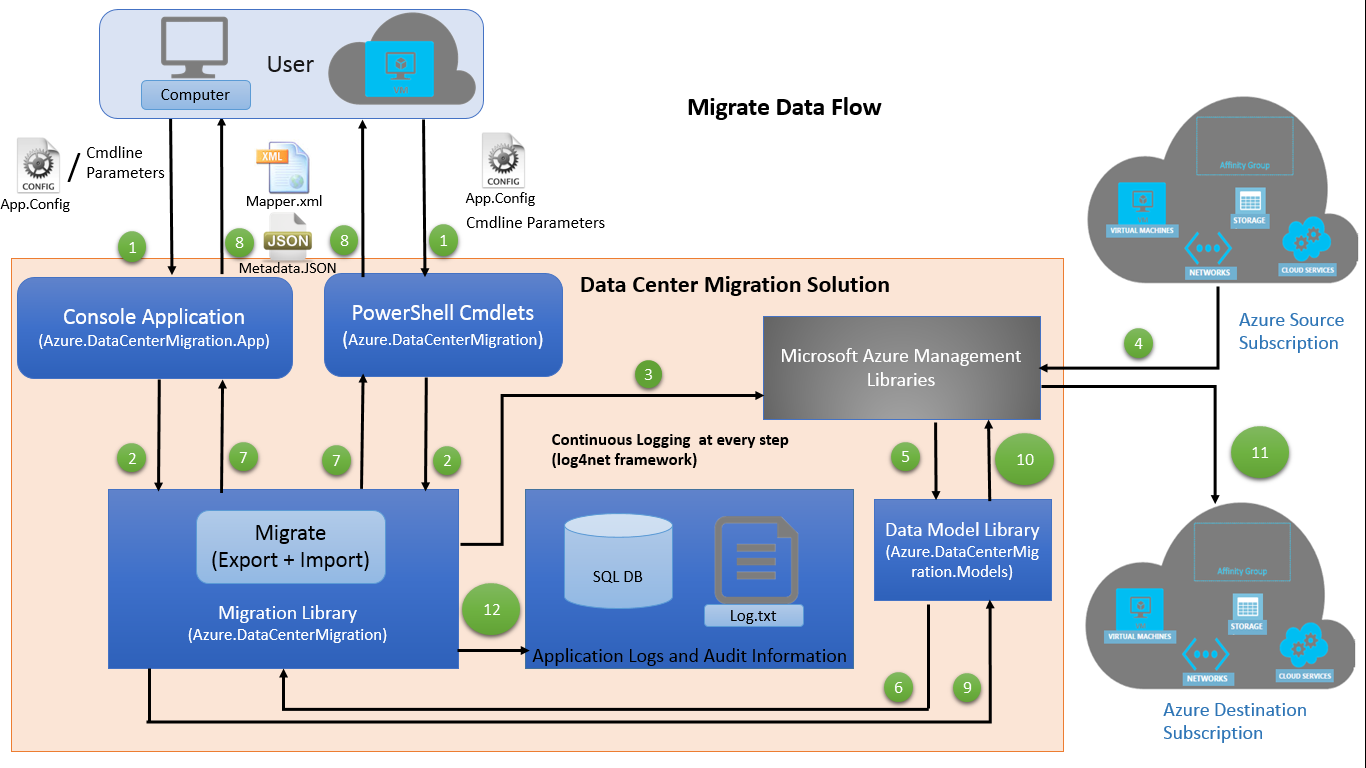


Figure 4. Migrate (export and import) data flow

**NOTE:** Resources created or deleted during or after the **export** operation are not migrated in the ongoing run.

The **migrate** operation works as follows:

1. You provide the input parameters using either the configuration file or command line for the console application or the command line parameters for the PowerShell cmdlet.
2. Depending on step 1, either the console app or the PowerShell cmdlet calls the Migration Library function to export the metadata.
3. The Migration Library requests details about the resources in the source Azure subscription using the Microsoft Azure Management Libraries.
4. Azure returns the details about the resources.
5. The Data Model Library stores the received data in memory.
6. The Data Model Library passes the data to the Migration Library for serialization.
7. The Migration Library serializes the data model to a JSON file.
8. Depending on step 1, the console app or the PowerShell cmdlet returns the JSON file containing information about the resources to you. If you enabled the generation of mapper XML, you also receive the source-destination name mapper XML file.
9. The Migration Library deserializes the JSON formatted metadata into data models using the Data Model Library.
10. The information about the resources is passed to the Microsoft Azure Management Libraries to deploy the resources on the destination Azure subscription.
11. The Microsoft Azure Management Libraries deploy the resources on the destination Azure subscription.
12. The solution continuously logs every step.
    1. When to use this solution

You can use the Data Center Migration Solution to help automate tasks associated with the following common IaaS deployment migrations:

* Deploying to multiple data centers. You may have started with a single data center deployment, but now you need to duplicate the setup. Maybe you need a second or third data center to support high availability or to provide disaster recovery. The solution helps you quickly discover, lift, and shift an entire deployment from one data center to another.
* Managing IaaS maintenance plans. Sometimes an IaaS-based system can’t use availability sets when, for example, it predates their use or uses stand-alone VMs. Yet, if system applications must remain available during IaaS maintenance windows, you need a way to shift your entire deployment to a second data center before that maintenance window closes and, afterward, to shift the deployment back to the primary data center.
* Moving between subscriptions. You can use the solution to transition deployments between subscriptions. For example, your organization may need to move from one account to another, switch from an MSDN subscription to a production subscription, or move a development or test deployment to a production environment.
* Repatriating to different data centers. As more Azure data centers come online around the world, you might gain a network latency advantage by moving your deployment to a location closer to home or to the largest customer base.
  1. Limitations of this version

The Data Center Migration Solution is not appropriate for every migration scenario. Note the following known limitations:

* The solution creates the cloud services of a specified data center *excluding* the staging and production deployment of web roles and worker roles. It considers only production deployments of virtual machines.
* The solution creates all storage accounts of the source subscription but excludes the VM-independent blobs. You must copy these blobs manually, if required.
* All cloud services are created in parallel, but all VMs of the same service are created sequentially. Azure does not support the creation of multiple VMs simultaneously in a single service. The following error appears:

Microsoft Azure is currently performing an operation   
on this hosted service that requires exclusive access.

VMs of different services are created in parallel.

* You cannot migrate a specific set of resources from a source subscription to a destination subscription. To do this, you must revise the JSON file.
* Migration occurs offline, so more downtime is needed during asset migration compared to solutions that use an agent-based approach to replicate volume and file writes between source and target locations—for example, Microsoft Migration Accelerator and Vision Solutions Double-Take MOVE.
* In case of failure, the rollback functionality cleans up the resources created by the solution. If you disable rollback, the solution reports an error and stops, and any required subsequent clean-up must be done manually.
  1. Use of the Microsoft Azure Management Libraries

To communicate with Azure, the solution uses Microsoft Azure Management Libraries, which are updated frequently. The solution currently uses the following versions of the NuGet packages:

* Microsoft Azure Compute Management Library – 6.1.2
* Microsoft Azure Infrastructure Management Library – 2.2.2
* Microsoft Azure Network Management Library – 4.3.0
* Microsoft Azure Storage Management Library – 3.2.0
* Microsoft Azure Common Library – 1.4.1
* Microsoft Azure Common Library Dependencies – 1.1.1

Any properties of deployed resources that are added or removed in future releases of these libraries will not be migrated.

**NOTE:** If you upgrade these libraries and fix any compile-time errors that may occur due to addition or removal of properties, the changes will be reflected in the exported metadata file. However, you will need to add the new properties to the **create** parameters of resources so that they are imported to the destination resources.

* 1. How this solution was tested

In test and production scenarios using data centers in Public Azure, the Data Center Migration Solution worked successfully. Among the scenarios tested:

* Migration of small (10 VMs, one VNet) to medium (100+ VMs, up to 10 VNets) IaaS deployments
* Migration between subscriptions in the same data center, between subscriptions in near data centers (same continent), and between subscriptions in data centers far apart (different continents)
* Migrations with variations in IaaS features: regional versus affinity-bound VNets, pure IaaS deployments versus mixed IaaS-platform as a service (PaaS) deployments, stand-alone VMs, VMs in subnets, single and multiple storage accounts, with or without availability sets, and more
* Compatibility with recently released features in IaaS, such as reserved IP, internal load balancer, and static internal IP
* Migrations where the exported JSON undergoes a number of name, configuration, binding, or other changes that make the destination deployment different from the source deployment
* Migration of a high-availability SharePoint farm

For more information about performance during these tests, see [Appendix C, “Performance results](#_Appendix_II:_Performance).”

1. Easing the challenge of migration: The Data Center Migration Solution

Migration projects typically start simply with a custom-written PowerShell script and a functional focus. Then, feature creep begins: error handling, operation logging, customizability, configuration, performance, and more, until the initial design is stressed beyond its limits. Teams spend a lot of time and effort debugging, troubleshooting, and fine-tuning performance. Unknown limitations may arise during the migration, causing stability and accuracy to suffer. Afterward, the custom solution usually isn’t generic enough to be reused for the next migration unless additional engineering resources are available.

The Data Center Migration Solution helps ease the challenge of migrating IaaS deployments between subscriptions or data centers. The following sections detail how the solution handles key migration considerations.

* 1. Atomicity…

Migrations are combinations of a large number of individual operations on resources, such as **stop**, **start**, **read**, **copy**, **move**, **create**, **check**, **write**, **wait**, and **cancel**. One or more of these operations can fail, possibly multiple times and to varying degrees. However, migrations should either wholly succeed or cleanly fail. Implementing this all-or-nothing semantic across a large set of operations requires great care.

In general, migrations may encounter two kinds of faults:

* **Permanent faults.** Faults which don’t go away by themselves and which must be corrected explicitly. For example, a destination location may have an existing resource of the same name as one in the source location that needs to be migrated. Ideally, permanent faults are spotted before migrations begin.
* **Transient faults.** Faults which go away after one or more retries but which don’t occur deterministically (for example, network errors or calls to APIs temporarily failing).

Handling these failures requires a combination of validations, retries, and rollbacks. The Data Center Migration Solution uses early validations, retries, and compensations to implement a limited level of atomicity. First, the solution attempts to find incompatibilities that will result in an unsuccessful migration, early in the migration lifecycle, by performing validation checks. For example, it checks that:

* The names of the resources, such as cloud services and storage accounts which will be created in the destination location, are unique and do not conflict with already existing resources.
* The data center specified for migration is allowed in the destination subscription.
* The certificate used for authentication of the source/destination subscription has the private key.

In addition, if there are validation errors, the migration does not begin. When known transient errors are encountered in any operation after the migration has begun, the migration solution attempts the operation again. For example, affinity group creation is retried if a task cancellation error occurs, as shown in the following figure. This can happen sometimes when a large number of affinity groups are being created.

|  |
| --- |
|  |

Figure 5. Task cancellation error during affinity group retry

Another example of a transient error is during blob copies across storage accounts. The Data Center Migration Solution uses an exponential retry policy in which the retry parameters are configurable. All executed operations are maintained in a permanent log. When the retry limit is exceeded or when unrecoverable faults are encountered, the solution uses the operation log to compensate for actions already done.

For example, if Azure is unable to upgrade the deployment due to internal constraints, such as an unavailable VM size in the cloud service deployment, the solution retries the operation according to the retry policy and logs the exception in detail. If the fault persists even after the defined number of retries, the solution gracefully stops the migration process. The following is an example of a permanent fault:

|  |
| --- |
|  |

Figure 6. Example error message caused by a permanent fault

* 1. Resumption

A migration run might encounter unrecoverable errors or may terminate if the host environment crashes. If this happens, the run must be resumed where it left off so that the remaining operations can be completed or so a clean rollback can be effected. Automated resumption avoids manual cleanup operations and saves time—when an execution fails, you don’t have to execute every step from the beginning.

When execution restarts, previously started migration actions, such as asynchronous blob copies, may be in progress, in addition to operations that have completed. To resume effectively, the migration solution must maintain a progress record of the result of execution of each operation—from the time the migration started. Maintaining a record of operations and their corresponding results in detail also helps in analyzing or debugging events later.

The Data Center Migration Solution maintains a hierarchical completion status of operations that corresponds with the resource hierarchy. The solution executes **copy**, **transfer**, **create**, and other operations based on a JSON file that represents a hierarchy of resources in a source deployment to be migrated.

While executing these operations, the solution maintains records of the completion result of each operation against each resource in a copy of the JSON file. If the solution stops for some reason and then resumes, it can read the completion status for each resource in the JSON file and execute only operations that have not yet completed. Resumption of import works only if **rollback on failure** is turned off.

Following is the snippet of JSON file, in which only affinity group and storage account details are visible:

****

Figure 7. JSON code snippet

Every resource in the JSON file contains the **IsImported** field. In addition, an **IsImported** field at the data center level indicates whether the import process is complete. By default, the value is **false**, which means that resources have not been migrated yet. During the migration process, the value of this field changes only if a resource is successfully imported to the destination subscription.

When the solution resumes, it imports only those resources where **IsImported** is **false**.

* 1. Compatibility

Compatibility issues can halt a migration. The Data Center Migration Solution includes an XML-based name-mapping file that you can use to map the names of resources discovered in the migration source to those in the destination. Alternately, you can specify a prefix to add to destination resource names. These features are designed to allow flexibility in resolving naming incompatibilities.

In general, incompatibility issues can arise in the following cases:

* **Globally unique names.** Certain names in Azure must be globally unique. For example, the name of an existing cloud service in IaaS cannot repeat anywhere else.
* **Differences in mechanisms used to create resources.** Resources in Azure can be created using the Azure Management API, Azure Rest API, and PowerShell cmdlets, or manually via the Azure Portal. Currently, these tools differ somewhat in the way they enforce name constraints and, in very rare cases, certain features supported by the portal may not be supported by the API or vice versa. So any migration faces a choice between enforcing the stronger or weaker set of constraints. The Data Center Migration Solution enforces the weaker set of constraints so that resources get migrated and the solution can be reused.
* **Limitations in the destination subscription of the migration.** The destination subscription may differ from the source (for example, it may allow fewer cores, making it impossible to migrate all resources). Or certain services might be disabled in the destination data center, so the migration attempt fails.

The Data Center Migration Solution checks names and detects any conflicts before migration starts so that problems can be fixed. To help ensure a successful migration, the solution enforces only a minimal set of constraints on names—those supported by both Azure Management API and Azure Portal. Before migration starts, the solution also checks for potential limitations in the destination subscription. For example, it verifies the total number of cores and whether sufficient space is available for affinity groups, storage accounts, cloud services, virtual networks, local networks, and DNS servers. In this way, incompatibilities are detected early.

In future versions, we plan to enable the Data Center Migration Solution to detect even more potential incompatibilities.

* 1. Consistency

When a migration is in progress, you need to weigh risk against availability. Should resources remain available (*warm migration*), which is risker, or should they be shut down (*cold migration*), which may take more time and disrupt users? The Data Center Migration Solution is designed to help reduce risks, when possible.

Warm migrations are the riskier scenario, in which resources, such as VMs, remain running so that you can continue to read information and data about the resources. For example, you can read VM properties or take blob snapshots while the VM is running. The source deployment remains usable during migration. Even if the migration takes hours or days to complete, users don’t perceive downtime. However, changes in the data or metadata in the source deployment may occur after a snapshot is taken but before the migration is finished. These in-flight changes are not propagated to the destination, causing inconsistencies in the data or metadata and even loss of data.

In addition, higher level, logical inconsistencies can occur with software that runs on VMs. For example, relational database management systems (RDBMSs) that run on VMs may issue an error, such as “database not recoverable,” when using point-in-time disk snapshots. Perhaps a transaction was only partially written to disk when the snapshot was taken.

For these reasons, the Data Center Migration Solution takes a safety-first approach and shuts down resources before migrating them. This approach results in downtime during migration, but the migration is shielded from in-flight changes. The export process is very fast (seconds to minutes) compared to actual migration, so the effect of configuration changes during capture is small. The solution shuts down VMs in the source deployment prior to the import process. The **copy** and **move** operations are based on the exported information, so the metadata stays stable.

It takes time to copy data disks across subscriptions. During this time, VMs could be restarted. The solution does not validate if VMs are started inadvertently after they are shut down, so there is a risk of data changes occurring while data disks are being copied.

**NOTE:** The Data Center Migration Solution does not address the situation in which resources are added or deleted while a migration is in progress. After the solution shuts down VMs, you must ensure that no changes are inadvertently caused. We recommend shutting down the VMs manually so that ongoing operations are not unexpectedly terminated.

* 1. Performance

The migration speed of an IaaS deployment from one location, such as a subscription or one data center to another, mainly depends on:

* The number and size of data disks.
* Time to create VMs at the destination location.
* The number of faults and retries.

To boost performance, the Data Center Migration Solution:

* Starts blob copies as early as possible. First, all VMs are shut down in parallel, and then all blob copies are started in parallel.
* Schedules all blob copies in parallel using the [asynchronous blob copy API](http://blogs.msdn.com/b/windowsazurestorage/archive/2012/06/12/introducing-asynchronous-cross-account-copy-blob.aspxhttp:/blogs.msdn.com/b/windowsazurestorage/archive/2012/06/12/introducing-asynchronous-cross-account-copy-blob.aspx).
* Continuously checks the status of copies for cancellations or failures.
* Implements a retry mechanism for transient faults. You can specify the number of retries.

When it comes to migration speed, the single biggest factors are the number and size of data disks on the source deployment. The total time taken to copy data disks from the source to destination is, on average, more than 80 percent of the overall migration time. Data disks are essentially blobs, so the asynchronous blob copy API provides an efficient, non-blocking way to transfer data disks. For a table of approximate copy times for data disks and different location combinations, see [Appendix C, “Performance results](#_Appendix_II:_Performance).”

For example, during a test run of a high-availability SharePoint farm with deployed sites, it took approximately six hours to copy the blobs from the eastern U.S. source to a northern European destination.

Table 1. Performance results for SharePoint farm with high-availability environment

|  |  |  |  |
| --- | --- | --- | --- |
| **Azure environment details** | **Time for export** | **Time for blob copy** | **Time for import** |
| Affinity group = 1  Cloud services in affinity groups = 3  Storage accounts in affinity groups = 1  VMs in affinity groups = 9  Number of VNets = 1  VMs in VNet = 9  Blobs = 13  Number of A1 VMs = 3  Number of A2 VMs = 4  Number of A6 VMs = 2 | 15 s | 6 h 3 m | 6 h 11 m |

The second-biggest factor in overall migration speed is the time it takes to create VMs on the destination data center. VM creation activities in Azure must be serialized, because only one VM can be created at a time in each cloud service on a subscription in Azure. In addition, a VM becomes fully functional only after data disks have been attached to it, so the VMs can’t be created at the destination until the data disks have been copied. The Azure engineering team is working to improve performance and parallelism in future releases.

The third significant component affecting migration speed is the time spent on handling faults. This is a variable component that adds an unpredictable amount of time to the entire migration. The operations that take longer to complete are the ones that more likely require retries. Sometimes, operations can throw an exception indicating a cancellation due to timeouts, but the operations actually succeed.

Table 2. Number of faults and retries

|  |  |
| --- | --- |
| **Operations requiring retries** | |
| To retrieve the data centers that are allowed in the subscription | client.Locations.List() |
| To export: | |
| Affinity groups | client.AffinityGroups.List() |
| Storage accounts | client.StorageAccounts.List() |
| Cloud services | client.HostedServices.List() |
| To import: | |
| Affinity groups | client.AffinityGroups.Create(<AffinityGroupCreateParameters>) |
| Storage accounts | computeManagementClient.StorageAccounts.Create( StorageAccountCreateParameters) |
| VNets | vnetClient.Networks.SetConfiguration(<NetworkSetConfigurationParameters>) |
| Cloud services | computeManagementClient.HostedServices.Create(<HostedServiceCreateParameters>) |
| VMs | computeClient.VirtualMachines.CreateDeployment(<serviceName>,<VirtualMachineCreateDeploymentParameters>) |
| To shut down virtual machines before blob copy | client.VirtualMachines.Shutdown(serviceName,deploymentName,VMName, <VirtualMachineShutdownParameters>) |
| To copy the Azure blobs | destBlob.StartCopyFromBlob(new Uri(srcBlobSasUri), null, null, requestOptions, null) |
| To delete or roll back: | |
| Affinity groups | client.AffinityGroups.Delete(affinityGroupName) |
| Storage accounts | client.StorageAccounts.Delete(storageAccountName) |
| VNets | vnetClient.Networks.SetConfiguration(<NetworkSetConfigurationParameters>) |
| Cloud services | client.HostedServices.DeleteAll(serviceName) |

* 1. Extensibility

To support a variety of migration scenarios, the Data Center Migration Solution provides customization options. For example, you can use a custom interface to run your migration, use a specific authentication or authorization module, execute specific actions before or after migration, or integrate with other tools. The following options help you extend the Data Center Migration Solution:

* With a core set of simple APIs, you can export, import, and migrate. These APIs are publicly exposed through a class library which can be called from any .NET-based application. For example, you can create a front-end application with a custom user interface and authentication and authorization mechanisms.
* Using the command line interface through PowerShell, you can create a script-based solution and easily call the commands.
* You can take advantage of the exported source subscription, which is available in the form of a JSON file, to automate certain tasks.
* For greater customization, the source code of the solution is also available.

1. Getting started
   1. System requirements

Before you run the Data Center Migration Solution, verify the following system requirements:

* Windows 7 SP1 or higher, or Windows Server 2008 R2 SP1 or higher

**NOTE:** The solution has been tested on 64-bit machines.

* Microsoft .NET Framework 4.5 or higher
* Windows PowerShell 3.0 and above

The solution runs from an on-premises computer or on a VM in the cloud.

* 1. Download publish settings file

A publish settings file contains information about your subscription and management certificate which can be used to authenticate Microsoft Azure Service Management API requests. The solution uses publish settings files to access the source and destination subscriptions during import/export/migrate.

To get the publish settings file, you can use either Azure Portal or Microsoft Azure PowerShell as follows.

### Using Azure Portal

1. To get the publish settings profile, go to [windows.azure.com/download/publishprofile.aspx](https://windows.azure.com/download/publishprofile.aspx).

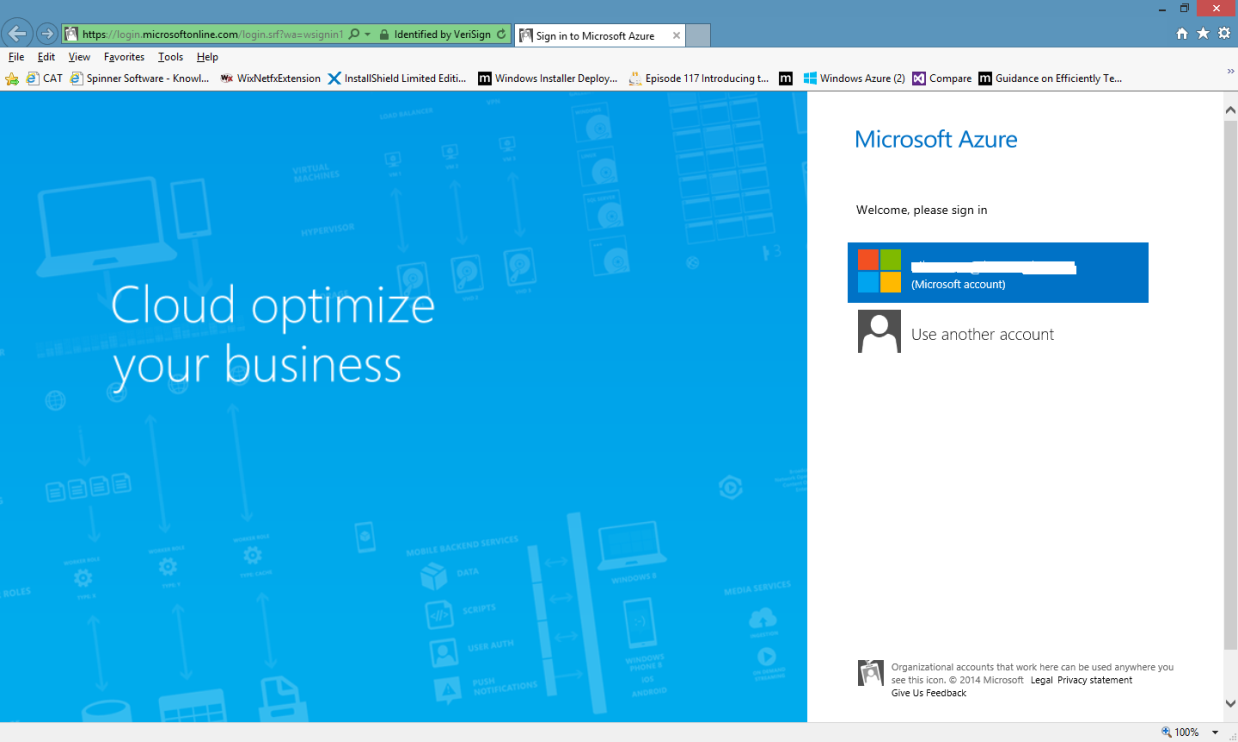


Figure 8. Getting the publish settings profile

1. Using your Azure subscription credentials, sign in to Microsoft Azure.

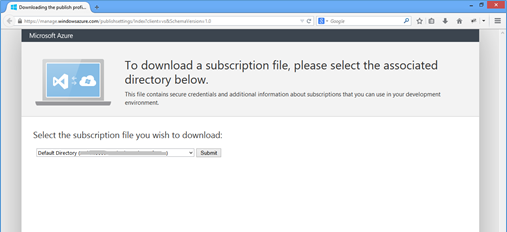


Figure 9. Downloading publish settings file

1. Select the subscription for which the Azure subscription file should be downloaded.
2. Provide file name with **.publishsettings** extension, and save the file.

### Using Microsoft Azure PowerShell

To download the publish settings files using Azure PowerShell:

1. On the command line, type:

**PS C:\> Get-AzurePublishSettingsFile**

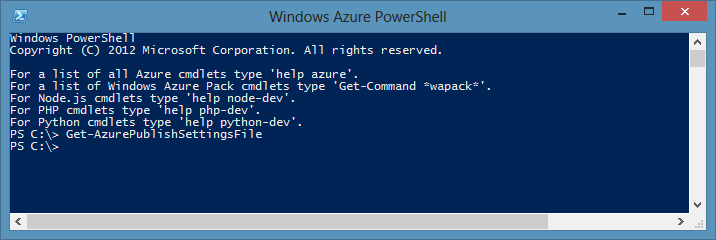


Figure 10. Using PowerShell to get the publish settings file

You are then redirected to a Microsoft page, as shown:

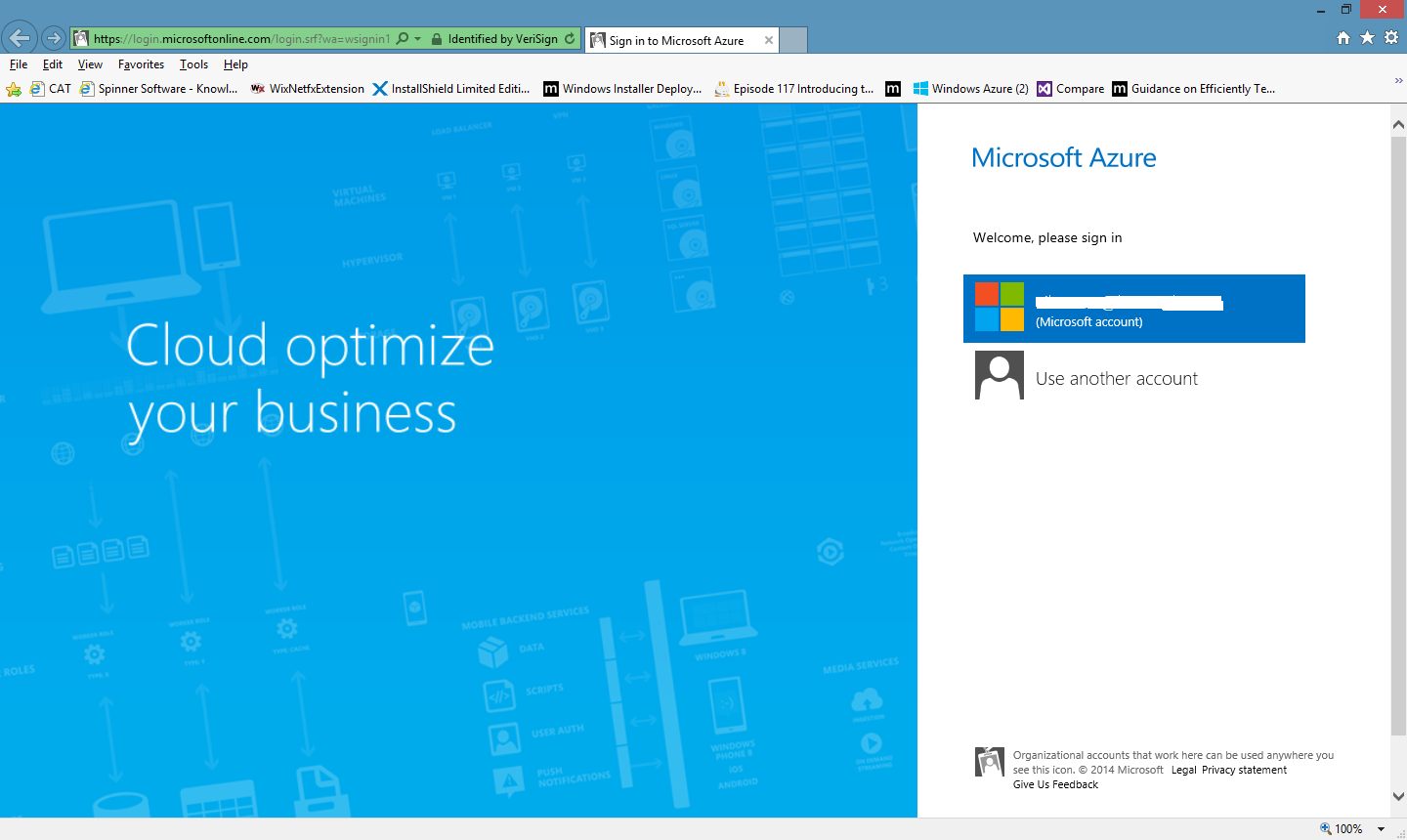


Figure 11. Microsoft Azure sign-in page

1. Using your Azure subscription user name and password, sign in to Microsoft Azure.

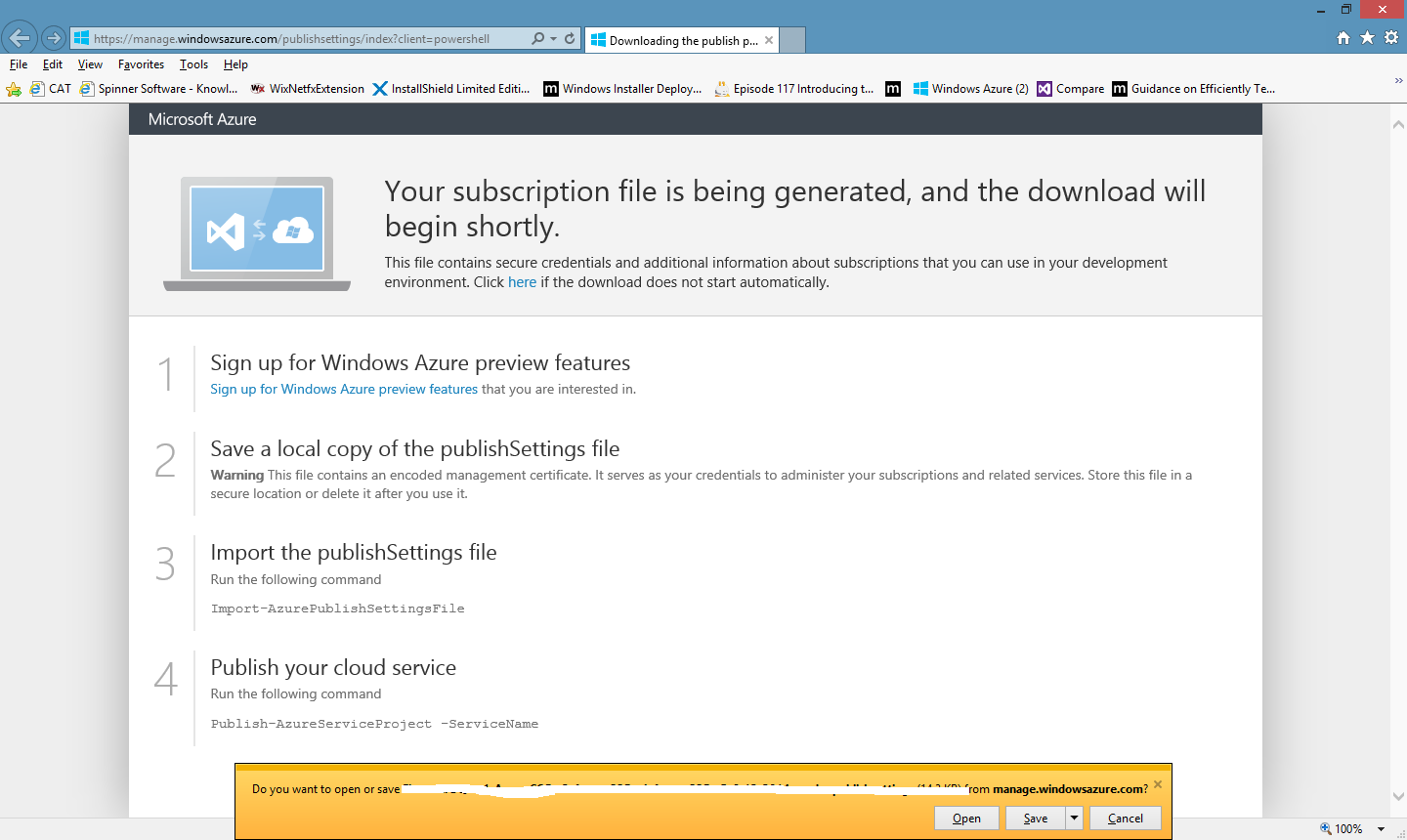


Figure 12. Downloading publish settings file

1. Save the file, and use **.publishsettings** as the file extension.
   1. Set up Azure management certificate

To use a subscription management certificate instead of a publish settings file:

1. Upload the .cer file, with public key as a management certificate for the subscription. To upload a management certificate to Azure, in the Management Portal, go to the Settings page and then select **MANAGEMENT CERTIFICATES**.
2. Install the certificate, with the private key set to **Personal store** (either **Current User** or **Machine**).
   1. Configuration parameters

Whether you update the configuration file or use command line parameters to run the solution, you must specify the configuration parameters, as shown in the following sections.

### Export operation

When you want to export information about resources (affinity groups, VNets, storage accounts, cloud services, and VMs) in a subscription data center to a metadata file in JSON format, specify these settings.

Table 3. Settings for exporting information about resources

|  |  |  |
| --- | --- | --- |
| **Key** | **Description** | **Is Mandatory** |
| **SourceSubscriptionID** | ID of the source subscription from the publish settings file mentioned in **SourcePublishSettingsFilePath**. | √ |
| **SourcePublishSettingsFilePath** | Path of the publish settings file for the source subscription.  To download this file, see Section 3.2, “Download publish settings file.” | √ |
| **SourceCertificateThumbprint** | Certificate thumbprint for the source subscription.  To set up the certificate, see Section 3.3, “Set up Azure management certificate.”  **NOTE:** This parameter is required if **SourcePublishSettingsFilePath** is not provided.  If both **SourcePublishSettingsFilePath** and **SourceCertificateThumbprint** are provided, the publish settings file gets preference. |  |
| **SourceDCName** | Name of the data center from which you want to export the resources. | √ |
| **ExportMetadataFolderPath** | Folder path to save exported metadata file.  This file contains information about all the resources in the source subscription for the specified data center.  The metadata JSON file name has the format:  **[Source DC Name]-[Date Time of execution in the format MM-dd-yyyy-HH-mm].json**  Example: **East Asia-07-18-2014-07-24.json** | √ |
| Operation | Possible values:  **Export** Choose this as the operation to be performed.  **Import**  **Migrate** | √ |
| **QuietMode** | Possible values:  **True** Progress messages are not displayed on the console.  **False** Default. Messages are displayed on the console. |  |
| **GenerateMapperXml** | Possible values:  **True** Resource Name Mapper XML file is generated.  **False** Default. Resource Name Mapper XML file is not generated. |  |
| **DestinationPrefixName** | Prefix to be appended to resource name in the destination subscription to maintain uniqueness.  Possible values:  **String** containing 1–5 characters.  Recommendation: Use fewer than five characters.  **NOTE:** This parameter is required if **GenerateMapperXml** is set to **true**.  For details about the solution’s naming conventions, see Appendix B.1, “Resource naming conventions.” |  |
| **RetryCount** | Number of retries to attempt in case of exception. |  |
| **MinBackoff** | Minimum backoff in seconds.  Parameter for exponential backoff retry strategy. |  |
| **MaxBackoff** | Maximum backoff in seconds.  Parameter for exponential backoff retry strategy. |  |
| **DeltaBackoff** | Delta backoff in seconds.  Parameter for exponential backoff retry strategy. |  |

### Import operation

Specify these settings when you want to import the previously exported resource into the specified destination subscription.

Table 4. Settings for importing information about resources

|  |  |  |
| --- | --- | --- |
| **Key** | **Description** | **Is Mandatory** |
| **SourceSubscriptionID** | ID of the source subscription from the publish settings file mentioned in **SourcePublishSettingsFilePath**. | √ |
| **SourcePublishSettingsFilePath** | Path of the publish settings file for the source subscription.  To download this file, see Section 3.2, “Download publish settings file.” | √ |
| **SourceCertificateThumbprint** | Certificate thumbprint for the source subscription.  To set up the certificate, see Section 3.3, “Set up Azure management certificate.”  **NOTE:** This parameter is required if **SourcePublishSettingsFilePath** is not provided.  If both **SourcePublishSettingsFilePath** and **SourceCertificateThumbprint** are provided, the publish settings gets preference. |  |
| **DestinationSubscriptionID** | ID of the destination subscription from the publish settings file mentioned in **DestinationPublishSettingsFilePath**. | √ |
| **DestinationDCName** | Name of the data center to which the resources are imported. | √ |
| **ImportMetadataFilePath** | Path of the metadata file with details about the source subscription.  The export operation creates the file in the folder specified in **ExportMetadataFolderPath**. | √ |
| **MapperXmlFilePath** | Path of the resource name mapper XML file.  When **GenerateMapperXml** is set to **true**, the export operation creates this file in the folder specified in **ExportMetadataFolderPath**. |  |
| **DestinationPublishSettingsFilePath** | Path of the publish settings file for the destination subscription.  To download this file, see Section 3.2, “Download publish settings file.” | √ |
| **DestinationCertificateThumbprint** | Certificate thumbprint for the destination subscription.  To set up the certificate, see Section 3.3, “Set up Azure management certificate.”  **NOTE:** This parameter is required if **DestinationPublishSettingsFilePath** is not provided.  If both **DestinationPublishSettingsFilePath** and **DestinationCertificateThumbprint** are provided, the publish settings gets preference. |  |
| **DestinationPrefixName** | Prefix to be appended to resource name in the destination subscription to maintain uniqueness.  For details about the solution’s naming conventions, see Appendix B.1, “Resource naming conventions.”  Possible values:  **String** containing 1–5 characters. Recommendation: Use fewer than five characters.  **NOTE:** This parameter is optional if **MapperXmlFilePath** is provided.  If both **MapperXmlFilePath** and **DestinationPrefixName** are provided, preference goes to the mapper XML file. |  |
| **ResumeImport** | Possible values:  **True** In case of import failure during import, the process continues and uses the value provided in **ImportMetadataFilePath** as the status file of the last import.  **False** Default. If false, the import status is updated in a new JSON file with the following name format: **{ImportMetadataFilePath}\_ImportStatus.json**  For details, see Appendix A.3, “Resume import operation.” |  |
| **Operation** | Possible values:  **Export**  **Import** Choose this as the operation to be performed.  **Migrate** | √ |
| **RollBackOnFailure** | Possible values:  **True** Roll back all deployment, if import failure occurs.  **False** Default. In case of failure, you can delete the resources manually. For details, see Section 7.4, “Delete resources manually.” |  |
| **QuietMode** | Possible values:  **True** Progress messages are not displayed on the console. Confirmation prompt for shutting down VMs is also not shown.  **False** Default. Messages and prompts appear on the console. |  |
| **RetryCount** | Number of retries in case of an exception. |  |
| **MinBackoff** | Minimum backoff in seconds.  Parameter for exponential backoff retry strategy. |  |
| **MaxBackoff** | Maximum backoff in seconds.  Parameter for exponential backoff retry strategy. |  |
| **DeltaBackoff** | Delta backoff in seconds.  Parameter for exponential backoff retry strategy. |  |

### Migrate operation

Specify these settings when you want to export the resources from a given source subscription for specified data center and to import the resources into specified destination subscription.

Table 5. Settings for migration

|  |  |  |
| --- | --- | --- |
| **Key** | **Description** | **Is Mandatory** |
| **SourceSubscriptionID** | ID of the source subscription from the publish settings file mentioned in **SourcePublishSettingsFilePath**. | √ |
| **SourcePublishSettingsFilePath** | Path of the publish settings file for the source subscription.  To download this file, see Section 3.2, “Download publish settings file.” | √ |
| **SourceCertificateThumbprint** | Certificate thumbprint for the source subscription.  To set up the certificate, see Section 3.3, “Set up Azure management certificate.”  **NOTE:** This parameter is required if **SourcePublishSettingsFilePath** is not provided.  If both **SourcePublishSettingsFilePath** and **SourceCertificateThumbprint** are provided, the publish settings file gets preference. |  |
| **SourceDCName** | Name of the data center from which you want to migrate the resources. | √ |
| **ExportMetadataFolderPath** | Folder path to save exported metadata file.  This file contains information about all the resources in the source subscription for the specified data center.  The metadata JSON file name has the format:  **[Source DC Name]-[Date Time of execution in the format MM-dd-yyyy-HH-mm].json**  Example: **East Asia-07-18-2014-07-24.json** | √ |
| **DestinationSubscriptionID** | ID of the destination subscription from the publish settings file mentioned in **DestinationPublishSettingsFilePath**. | √ |
| **DestinationDCName** | Name of the data center to which you want to migrate the resources. | √ |
| **DestinationPublishSettingsFilePath** | Path of the publish settings file for the destination subscription.  To download this file, see Section 3.2, “Download publish settings file.” | √ |
| **DestinationCertificateThumbprint** | Certificate thumbprint for the destination subscription.  To set up the certificate, see Section 3.3, “Set up Azure management certificate.”  **NOTE:** This parameter is required if **DestinationPublishSettingsFilePath** is not provided.  If both **DestinationPublishSettingsFilePath** and **DestinationCertificateThumbprint** are provided, the publish settings file gets preference. |  |
| **DestinationPrefixName** | Prefix to be appended to resource name in the destination subscription to maintain uniqueness.  For details about the solution’s naming conventions, see Appendix B.1, “Resource naming conventions.”  Possible values:  **String** containing 1–5 characters.  Recommendation: Use fewer than five characters. | √ |
| **Operation** | Possible values:  **Export**  **Import**  **Migrate** Choose this as the operation to be performed. | √ |
| **RollBackOnFailure** | Possible values:  **True** Roll back all deployment, if import failure occurs.  **False** Default. In case of failure, you can delete the resources manually. For details, see Section 7.4, “Delete resources manually.” |  |
| **QuietMode** | Possible values:  **True** Progress messages are not displayed on the console. Confirmation prompt for shutting down VMs is also not shown.  **False** Default. Messages and prompts are displayed on the console. |  |
| **RetryCount** | Number of retries in case of exception. |  |
| **MinBackoff** | Minimum backoff in seconds.  Parameter for exponential backoff retry strategy. |  |
| **MaxBackoff** | Maximum backoff in seconds.  Parameter for exponential backoff retry strategy. |  |
| **DeltaBackoff** | Delta backoff in seconds.  Parameter for exponential backoff retry strategy. |  |

1. Running the Azure Data Center Migration Solution

The Data Center Migration Solution is configurable via input parameters in command line and configuration files.

**NOTE:** The Data Center Migration Solution does not address the situation in which resources are added or deleted while a migration is in progress. After the solution shuts down VMs, you must ensure that no changes are caused inadvertently. We recommend shutting down the VMs manually so that ongoing operations are not terminated unexpectedly.

* 1. Input parameters

### Using the Console application

* + - 1. Updating the configuration file

To update the configuration file:

1. Edit the **Azure.DataCenterMigration.App.exe.config** file from the application’s installation location (for example, C:\Program Files\Azure Data Center Migration 1.0\bin).
2. Change the parameters in the **<appSettings>** section.

|  |
| --- |
| <appSettings>  <!-- Required for: Export/Import/Migrate -->  <add key="SourceSubscriptionID" value="########-########-########-########"/>  <!-- Required for: Export/Migrate -->  <add key="SourceDCName" value="Southeast Asia"/>  <!-- Required for: Export/Migrate -->  <add key="ExportMetadataFolderPath" value="D:\DC migration\"/>  <!-- Required for: Export/Import/Migrate -->  <add key="SourcePublishSettingsFilePath" value="D:\DC migration\publishsettingsfile.publishsettings"/>  <!--Required for: Export/Import/Migrate if SourcePublishSettingsFilePath is not provided-->  <add key="SourceCertificateThumbprint" value="########################" />  <!-- Required for: Export - False : Resource Name Mapper XML file will not be generated.  True : Will create Resource Name Mapper XML file -->  <add key="GenerateMapperXml" value="true" />    <!-- Required for: Import/Migrate -->  <add key="DestinationSubscriptionID" value="########-########-########-########"/>  <!-- Required for: Import/Migrate -->  <add key="DestinationDCName" value="Southeast Asia"/>  <!-- Required for: Import -->  <add key="ImportMetadataFilePath" value="D:\DC migration \Southeast Asia-07-03-2014-08-14.json"/>  <!-- Required for: Import/Migrate -->  <add key="DestinationPublishSettingsFilePath" value="D:\DC migration\publishsettingsfile.publishsettings"/>  <!-- Required for: Import -->  <add key="MapperXmlFilePath" value=" D:\DC migration \Southeast Asia-07-03-2014-08-14.XML" />  <!-- Required for: Import/Migrate if DestinationPublishSettingsFilePath is not -->  <add key="DestinationCertificateThumbprint" value="########################" />  <!-- Required for: Import/Migrate -->  <add key="DestinationPrefixName" value="dcmig"/>  <!-- Required for: Import -  False: Will make a copy of Import Metadata File and update import status in the copy.  True : Will update import status in Import Metadata File -->  <add key="ResumeImport" value="false"/>  <!-- Required for: Import/Migrate -  True if roll back all deployment if failure occurs-->  <add key="RollBackOnFailure" value="False"/>  <!--Common Parameters-->  <add key="QuietMode" value="False"/>  <add key="Operation" value="Import"/>  <!--Parameters for exponential backoff retry strategy, required for: Import/Migrate - -->  <!-- No. of times to retry in case of exception -->  <add key="RetryCount" value="3" />  <!--Minimum backoff in seconds -->  <add key="MinBackoff" value="3" />  <!--Maximum backoff in seconds -->  <add key="MaxBackoff" value="90" />  <!--Delta Backoff in seconds -->  <add key="DeltaBackoff" value="90" />  </appSettings> |

1. After updating the configuration file, launch the executable **Azure.DataCenterMigration.App.exe** from the installation location (for example, C:\Program Files\Azure Data Center Migration 1.0\bin).
   * + 1. Using command line parameters

To run the Azure Data Center Migration Solution using command line parameters:

1. Open a command prompt window.
2. Go to the installation location (for example, C:\Program Files\Azure Data Center Migration 1.0\bin).
3. To pass the parameters to the Azure.DataCenterMigration.App.exe file, refer to the following sample commands. Check the details about the parameters in Section 3.4, “[Configuration parameters](#ConfigurationParameterDetails).”

**NOTE:** To specify the directory path in the command line parameters, use **\\** (do not use **\**). For example:

Azure.DataCenterMigration.App.exe -Operation "Export" -SourceSubscriptionID "########-########-########-########" -SourceDCName "East Asia" -ExportMetadataFolderPath "D:\\DataCenterMigration" -PublishSettingsFilePath "D:\\PublishSettings.PublishSettings" -QuietMode "True"

|  |
| --- |
| **Example1 (Export)** : Azure.DataCenterMigration.App.exe -Operation "Export" -SourceSubscriptionID "########-########-########-########" -SourceDCName "East Asia" -ExportMetadataFolderPath "D:\\DataCenterMigration" -PublishSettingsFilePath "D:\\PublishSettings.PublishSettings" -QuietMode "True" – GenerateMapperXml "True"  **Example2 (Import)**: Azure.DataCenterMigration.App.exe -Operation "Import" -SourceSubscriptionID "########-########-########-######## " –DestinationSubscriptionID "########-########-########-######## " -DestinationDCName "West US" –SourcePublishSettingsFilePath "D:\\PublishSettings.PublishSettings" -DestinationPublishSettingsFilePath "D:\\PublishSettings.PublishSettings" -ImportMetadataFilePath "D:\\DataCenterMigration\\mydata.json" -DestinationPrefixName "dc" - QuietMode "True" -RollBackOnFailure "True" -ResumeImport "True"  **Example3 (Migrate)** : Azure.DataCenterMigration.App.exe -Operation "Migrate" -SourceSubscriptionID "########-########-########-######## " -DestinationSubscriptionID "########-########-########-########" -SourceDCName "East Asia" -DestinationDCName "West US" -SourcePublishSettingsFilePath "D:\\PublishSettings.PublishSettings" -DestinationPublishSettingsFilePath "D:\\PublishSettings.PublishSettings" -ExportMetadataFolderPath "D:\\DataCenterMigration" -DestinationPrefixName "dc" - QuietMode "True" -RollBackOnFailure "True" |

### Using PowerShell cmdlets

To run the solution using PowerShell cmdlets, set up and run it as follows.

* + - 1. Setup

1. With administrator rights, open a PowerShell window.
2. In the installation location, go to the PowerShell folder (for example, C:\Program Files\Azure Data Center Migration 1.0\PowerShellCmdlets) and copy the Azure.DataCenterMigration folder.
3. With administrator rights, in the PowerShell window, type **$Env:PSModulePath**. The PSModulePath environment variable stores the paths to the locations of the modules that are installed on disk. PowerShell uses this variable to locate modules when you do not specify a full path to a module. The paths in this variable are searched in the order in which they appear.
4. Paste the **Azure.DataCenterMigration** folder to one of the module folders specified by the **$Env:PSModulePath** variable.
   * + 1. Run cmdlets
5. Open a PowerShell window. At the prompt, use **Get-Module Azure.DataCenterMigration** to check if the module is imported. If it is not available, use **Import-Module Azure.DataCenterMigration** to import the module.

**NOTE:** The module is automatically imported with Windows PowerShell 3.0 and above.

1. To display all the cmdlets of the Azure.DataCenterMigration module, use **Get-Command –Module Azure.DataCenterMigration**.

Table 6. Details of Azure.DataCenterMigration cmdlets

|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Help command** |
| **Export-AzureSubscriptionMetadata** | Exports the information about source subscription and stores the metadata in the following format in the specified **ExportMetadataFolderPath**:  **SourceDataCenterName-MM-DD-YYYY-hh-mm.json** | **Get-Help Export-AzureSubscriptionMetadata**  Example:  **Get-Help Export-AzureSubscriptionMetadata -** |
| **Import- AzureSubscriptionMetadata** | Reads exported metadata JSON file and deploys all the source resources into destination subscription. | **Get-Help Import-AzureSubscriptionMetadata**  Example:  **Get-Help Import-AzureSubscriptionMetadata -Examples** |
| **Migrate-AzureSubscription** | Combines export and import functionality. Exports information about source subscription and stores the metadata in the following format in the specified **ExportMetadataFolderPath**:  **SourceDataCenterName-MM-DD-YYYY-hh-mm.json**  It also reads the exported metadata JSON file and deploys all the source resources into destination subscription. | **Get-Help Migrate-AzureSubscription**  Example:  **Get-Help Migrate-AzureSubscription –Examples** |

**NOTE:** Log files for the PowerShell cmdlets are generated in the **C:\Windows\System32\WindowsPowerShell** folder, by default. To change this folder path, specify a different value in the configuration file **Azure.DataCenterMigration.App.exe.config** where you copied the PowerShell module (for example, C:\Windows\System32\WindowsPowerShell\v1.0\Modules\Azure.DataCenterMigration\bin\ Azure.DataCenterMigration.App.exe.config).

|  |
| --- |
| <appender name="LogFileAppender" type="log4net.Appender.FileAppender">       <file type="log4net.Util.PatternString"           value="="..\Logs\DCMigrationLog -%date{MM-dd-yyyy-HH-mm}.txt" />       <lockingModel type="log4net.Appender.FileAppender+MinimalLock" />       <appendToFile value="true" />       <layout type="log4net.Layout.PatternLayout">         <param name="ConversionPattern" value="%date &#9; %-5level &#9; %logger &#9; %property{ResourceType} &#9; %property{ResourceName} &#9; %message%newline"/>       </layout>     </appender> |

* 1. Logging configurations

### Configuration settings

The Data Center Migration Solution uses [log4net](https://www.nuget.org/packages/log4net/)[[1]](#footnote-1) for logging, using two kinds of log appenders:

* A text file with the name **DCMigrationLog - [DateTimeOfExecution].txt**
* Optionally to a SQL Server database (can be chosen during installation or configured after installation)

**NOTE:** This section assumes that you enabled SQL Database Logging during installation. If you haven’t, and you want to enable it after installation, please refer to Section 4.2.2, “Log database setup.”

To change the configuration settings for logging:

1. To customize logging for the console application, open the **Azure.DataCenterMigration.App.exe.config** file from the installation location (for example, C:\Program Files\Azure Data Center Migration 1.0\bin).
2. To customize logging for the PowerShell cmdlets, open the **Azure.DataCenterMigration.App.exe.config** file from the PowerShell modules folder (for example, C:\Windows\System32\WindowsPowerShell\v1.0\Modules\Azure.DataCenterMigration\bin\ Azure.DataCenterMigration.App.exe.config).
3. Edit the highlighted sections as follows:
4. To change the log folder path or file name of the text file logging, edit the value attribute of the **<file>** tag for the **LogFileAppender** (highlighted in the code sample that follows). The **%date{MM-dd-yyyy-HH-mm}** parameter is replaced by the date and time of execution.
5. To disable text file logging, comment out the **appender-ref** tag for **LogFileAppender** highlighted in the configuration file that follows.
6. To change the connection string used for logging into the database, modify the value attribute of the **<connectionString>** tag for the **AdoNetAppender** (highlighted in the code sample that follows).
7. If you want to disable database logging, comment out the **appender-ref** tag for **AdoNetAppender** highlighted in the configuration file that follows.

|  |
| --- |
| <log4net>  <root>  <level value="All"/>  <!--send all logs to appenders-->  <appender-ref ref="AdoNetAppender"/>  <appender-ref ref="LogFileAppender"/>  </root>  <appender name="AdoNetAppender" type="log4net.Appender.AdoNetAppender">  <bufferSize value="1"/>  <connectionType value="System.Data.SqlClient.SqlConnection,  System.Data, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>  <connectionString value="data source=[data source here];initial catalog=DataCenterDB; user id=[user id here];password=[password here];" />  <commandText value="INSERT INTO DataCenterLogs ([Date],[Level],[Logger],[Message],[ResourceType],[ResourceName]) VALUES (@log\_date, @log\_level,@logger, @message, @resource\_type,@resource\_name)"/>  <parameter>  <parameterName value="@log\_date"/>  <dbType value="DateTime"/>  <layout type="log4net.Layout.RawTimeStampLayout"/>  </parameter>  <parameter>  <parameterName value="@log\_level"/>  <dbType value="String"/>  <size value="50"/>  <layout type="log4net.Layout.PatternLayout">  <conversionPattern value="%level"/>  </layout>  </parameter>  <parameter>  <parameterName value="@logger"/>  <dbType value="String"/>  <size value="255"/>  <layout type="log4net.Layout.PatternLayout">  <conversionPattern value="%logger"/>  </layout>  </parameter>  <parameter>  <parameterName value="@message"/>  <dbType value="String"/>  <size value="4000"/>  <layout type="log4net.Layout.PatternLayout">  <conversionPattern value="%message"/>  </layout>  </parameter>  <parameter>  <parameterName value="@resource\_type"/>  <dbType value="String"/>  <size value="4000"/>  <layout type="log4net.Layout.PatternLayout">  <conversionPattern value="%property{ResourceType} "/>  </layout>  </parameter>  <parameter>  <parameterName value="@resource\_name"/>  <dbType value="String"/>  <size value="4000"/>  <layout type="log4net.Layout.PatternLayout">  <conversionPattern value="%property{ResourceName}"/>  </layout>  </parameter>  </appender>  <appender name="LogFileAppender" type="log4net.Appender.FileAppender">  <file type="log4net.Util.PatternString"  value="..\Logs\DCMigrationLog -%date{MM-dd-yyyy-HH-mm}.txt" />  <lockingModel type="log4net.Appender.FileAppender+MinimalLock" />  <appendToFile value="true" />  <layout type="log4net.Layout.PatternLayout">  <param name="ConversionPattern" value="%date &#9; %-5level &#9; %logger &#9; %property{ResourceType} &#9; %property{ResourceName} &#9; %message%newline"/>  </layout>  </appender>  </log4net> |

### Log database setup

If you chose to enable SQL Database Logging during installation, the installer sets up the necessary database schema for you.

If you chose not to enable SQL Database Logging during installation, and you want to enable it after installation, step through the instructions that follow, to set up the database schema and configuration.

1. To customize logging for the console application, open the **Azure.DataCenterMigration.App.exe.config** file from the installation location (for example, C:\Program Files\Azure Data Center Migration 1.0\bin).
2. To customize logging for the PowerShell cmdlets, open the **Azure.DataCenterMigration.App.exe.config** file from the PowerShell modules folder (for example, C:\Windows\System32\WindowsPowerShell\v1.0\Modules\Azure.DataCenterMigration\bin\ Azure.DataCenterMigration.App.exe.config).
3. Update the following connection string in the **Azure.DataCenterMigration.DatabaseLogSetup.exe.config**:

<add name="DbConnection" connectionString="data source=[data source here];initial catalog=DataCenterDB; user id=[user id here];password=[password here];" providerName="System.Data.SqlClient" />

1. To create the database schema, run the executable **Azure.DataCenterMigration.DatabaseLogSetup.exe** from the installation location (for example, C:\Program Files\Azure Data Center Migration 1.0\bin).
2. Update the connection string for the **AdoNetAppender** in **Azure.DataCenterMigration.App.exe.config** with the connection string. (For details, refer to Section 4.2.1, “Configuration settings.”)

<connectionString value="data source=[data source here];initial catalog=DataCenterDB; user id=[user id here];password=[password here];" />

1. Add an **appender-ref** entry to the **AdoNetAppender** as shown in the following section. (For details, refer to Section 4.2.1, “Configuration settings.”)

<appender-ref ref=”AdoNetAppender”/>

1. Completing a migration
   1. After a successful migration

Do the following:

1. Start the VMs (in the source subscription) that were shut down (stopped) during migration process.

**NOTE:** VMs in any state consume computing resources unless stopped (deallocated).

1. Verify that all resources—affinity groups, VMs, storage accounts (blobs related to VMs), and cloud services— migrated successfully, as follows:
   * Check the resource name details in Appendix B.1, “Resource naming conventions.”
   * To verify imported resources, open the **{ExportedMetadataFile}\_ImportedStatus.json** file. In the metadata file, the **IsImported** flag of each resource indicates whether it was imported in the destination.
   * Verify that all the resources are migrated successfully in the destination subscription.

**NOTE:** To communicate with Azure, the solution uses Microsoft Azure Management Libraries, which are updated frequently. Any properties of deployed resources that are added or removed in future releases of these libraries will not be migrated, so you will need to migrate these properties manually. For more information about the versions used, please refer to section 1.4, “Use of the Microsoft Azure Management Libraries.”

* 1. If the import fails

If the import fails for any reason:

1. Check the log file for the reason of failure.
2. If **RollBackOnFailure** was set to **true** during the run, the solution would have cleaned up resources that were created during the run.
3. If **RollBackOnFailure** was set to **false**

* You can use the solution’s resumption feature to resume the operation by setting **ImportMetadataFilePath** to **{ExportedMetadataFile}\_ImportedStatus.json** and **ResumeImport** to **true**.
* Alternately, you can refer to section 7.4, “Delete resources manually” to clean up resources and start over.

1. Troubleshooting

Table 7. Troubleshooting exceptions

|  |  |
| --- | --- |
| **Exception** | **Troubleshoot** |
| Insufficient Space - Cloud Services | Increase the quota for “Cloud Services” in destination subscription. |
| Insufficient Space - Cores | Increase the quota for “Cores” in destination subscription. |
| Insufficient Space – DNS Servers | Increase the quota for “DNS Servers” in destination subscription. |
| Insufficient Space – Local Network | Increase the quota for “Local Networks” in destination subscription. |
| Insufficient Space – Storage Account | Increase the quota for “Storage Account” in destination subscription. |
| Insufficient Space – Virtual Network | Increase the quota for “Virtual Network” in destination subscription. |
| Invalid Affinity Group Name | Check the affinity group name.  The name must start with a letter or a number and can contain only letters, numbers, and dashes. |
| Affinity Group Name already exists | Affinity Group with same name already exists in the destination subscription. Please change the name. |
| Invalid Destination Prefix Name | Destination prefix name must contain 1–5 characters. |
| Invalid Parameter Value {Parameter name} | Parameter value for specified parameter name is invalid. |
| Cloud Service Name already exists | Cloud service with same name already exists in the destination subscription. Please change the name. |
| Source Data Disk does not exist | Check whether source data disk exists. |
| Source OS Disk does not exist | Check whether source OS disk exists. |
| Invalid Storage Account | Check the storage account name. The name can contain only lowercase letters and numbers. |
| Storage Account Name already exists | Storage account with same name already exists in the destination subscription. Please change the name. |
| Invalid Service Name | Check the cloud service name.  The name can contain only letters, numbers, and hyphens. The first and last character in the field must be a letter or number. Trademarks, reserved words, and offensive words are not allowed. |
| VM already exists | VM with same name already exists in the cloud service at destination subscription. Please change the name of VM. |
| Invalid VM Name | Check the VM name.  The name can contain only letters, numbers, and hyphens. The name must start with a letter and must end with a letter or a number. |
| Generated new resource name contains duplicate Affinity Group Name(s). | New affinity group name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| Generated new resource name contains duplicate DNS Name(s). | New DNS name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| Generated new resource name contains duplicate Local Network Name(s). | New local network name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| Generated new resource name contains duplicate Cloud Service Name(s). | New cloud service name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| Generated new resource name contains duplicate Storage Account Name(s). | New storage account name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| Generated new resource name contains duplicate VM Name(s). | New VM name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| Generated new resource name contains duplicate Virtual Network Name(s). | New virtual network name with prefix is generating duplicate names. Reducing the prefix name characters might solve this issue. |
| The Reserved IP “ReservedIPNameHere” does not exist. | Reserved IP with specified name does not exist in the destination subscription. Please create Reserved IP in the destination subscription. For details, see Section 7.1, “[Reserved IP configuration](#ReservedIPConfiguration).” |
| ReservedIPName “ReservedIPNameHere” is already assigned to service “ServiceNameHere” in destination subscription. | Reserved IP with specified name is already assigned to the service in the destination subscription. Please create another Reserved IP in the destination subscription. For details, see Section 7.1, “[Reserved IP configuration](#ReservedIPConfiguration).” |

1. Use cases
   1. Reserved IP configuration

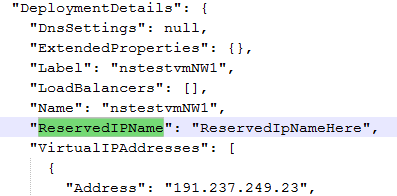
If a reserved IP is assigned to a VM in the source subscription, you can set a reserved IP in the destination subscription during import. The reserved IP should already be created in the destination subscription.

To create the reserved IP, you can use:

* Windows PowerShell (see <http://msdn.microsoft.com/en-us/library/azure/dn690120.aspx>).
* Service Management API (see <http://msdn.microsoft.com/en-us/library/azure/dn722420.aspx>).

After you create the reserved IP, modify the JSON file obtained during an export in the folder specified in [**ExportMetadataFolderPath**](#ExportMetadataFolderPath). For details, see Section 3.4.1, “Export operation.”

Set the value of the **ReservedIPName** property in the **DeploymentDetails** section of specific cloud service to the newly created Reserved IP, as follows:



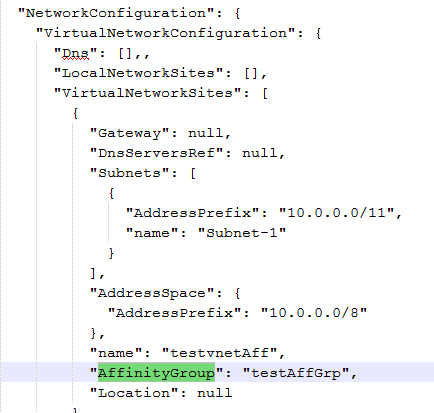
You can use this modified JSON file as input during the **import** operation. For details, see Section 3.4.2, “Import operation.”

**WARNING:** If the exported JSON file contains **ReservedIPName** for the deployment, but you do not manually create it in the destination subscription, the migration fails.

* 1. Regional VNet

To migrate VMs from a narrow VNet to a regional VNet, modify the JSON file obtained during an [export](#_Export_Resources) in the folder specified in [**ExportMetadataFolderPath**](#ExportMetadataFolderPath).

Modify the following properties of **VirtualNetworkSites** in the **NetworkConfiguration** section in the JSON file.



* **AffinityGroup.** Set **AffinityGroup** to **null**.
* **Location.** Set **location** to destination location as specified in [**DestinationDCName**](#DestinationDCName).



* 1. Change VM size

To change the VM size, modify the JSON file obtained during an export in the folder specified in [**ExportMetadataFolderPath**](#ExportMetadataFolderPath). For details, see Section 3.4.1, “Export operation.”

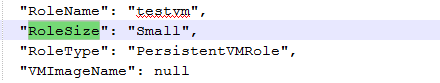
You can specify the size in two sections:

* Range 1: A0 to A7
* Range 2: A8 and A9

### Change the size within the same range

To change the size within the same range, you must modify in the JSON file.

Change the **RoleSize** property of the VM in the **DeploymentDetails** section, as follows:



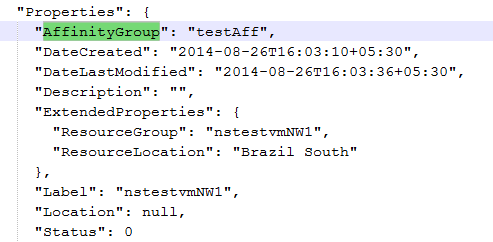
### Change the size from [range 1](#Range1) to [range 2](#Range2)

To change the VM size from range 1 to range 2, modify the **RoleSize** property of the VM in the **DeploymentDetails**.

Since A8 and A9 sizes are only allowed in a [regional VNet](#RegionalVNet), modify the JSON, as described earlier in this section.

If the cloud service of the VM is in the affinity group, modify the following properties in **CloudServiceDetails** Properties section:

* **AffinityGroup.** Set AffinityGroup to **null**.
* **Location.** Set **location** to destination location, as specified in [**DestinationDCName**](#DestinationDCName).



* 1. Delete resources manually

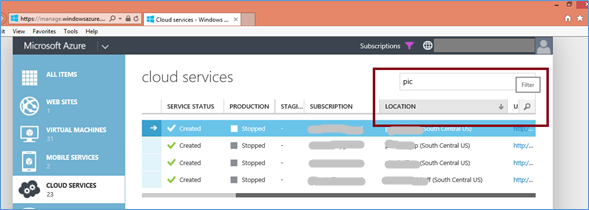
If a failure occurs and the **RollBackOnFailure** property is disabled, you can delete the imported resources manually using the Azure Portal.

To delete resources manually:

1. Open the resource name mapper XML file of the run. This file contains names of resources that are created in the destination. You need to delete all the resources that may have been created.

**NOTE:** The Import Status file (**{ExportedMetadataFile}\_ImportedStatus.json**) contains information on which resources were successfully created. However, there may be some partially created resources. These must be deleted, too.

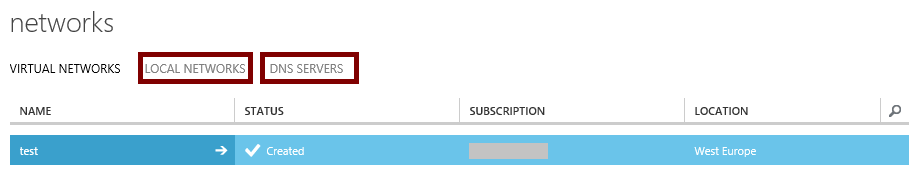
1. Using your subscription credentials, sign in to Azure Portal (<https://manage.windowsazure.com>).
2. Select the destination subscription.
3. In the left pane, navigate to **CLOUD SERVICES**. Find a cloud service specified in the resource name mapper file, as shown.



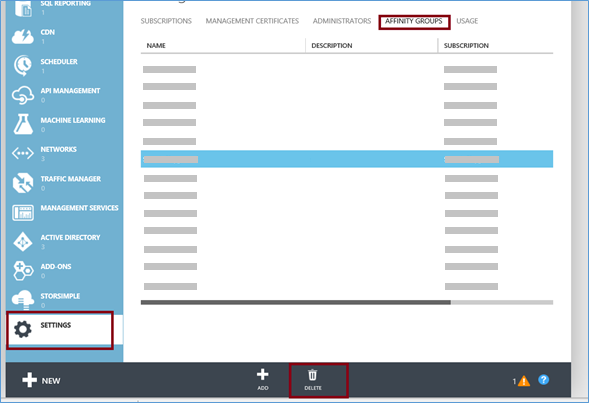
1. From the list, select the cloud service and choose **Delete**. Select **Delete the cloud service and its deployments**.
2. For all cloud services in the resource name mapper XML, repeat steps 4 and 5.

7. In the left pane, select **STORAGE**. Find a storage account specified in the resource name mapper XML.

1. Select the storage account name, and then, in the lower panel, select **Delete**.
2. For all storage accounts in the resource name mapper XML, repeat steps 7 and 8.
3. In the left panel, select **NETWORKS**. Find a network specified in the resource name mapper XML.
4. Select the network, and choose **Delete**.
5. On the Networks page, from the **LOCAL NETWORKS** and **DNS SERVERS** tabs, delete the local networks and DNS servers.



1. To delete affinity groups, in the left panel, select **Settings** and then select the **AFFINITY GROUPS** tab. Delete the affinity groups that are in the resource name mapper XML.



* 1. Choose resources to migrate

At this time, the solution does not provide a way to choose which resources to export or import. A workaround to achieve this is to:

- Export the source data center to the metadata JSON file

- Modify it to remove resources that should not be copied to the destination

- Use the edited file during the Import process

The metadata file contains all VM dependent resources like VMs/affinity groups/cloud services/storage accounts/virtual networks. If your source data center contains extra resources, you can choose not to import them by removing them from the exported JSON file. While editing the JSON file, you need to be careful that you don’t miss the dependent resources of the VMs you want to migrate. E.g. If the virtual machine is present within affinity group or in a virtual network, all the dependent resources i.e. cloud service in which the VM is located, storage account where the blobs are stored, affinity group and virtual network if applicable should be present in the metadata file.

1. Using the source code

The solution is open sourced and is licensed under the Apache License, Version 2.0.

The source code is available at <https://github.com/persistentsystems/adcms>.

* 1. System requirements

The solution runs from an on-premises computer or on a VM in the cloud.

Before you can build the Data Center Migration Solution, verify the system requirements:

* Windows 7 SP1 or higher, or Windows Server 2008 R2 SP1 or higher

**NOTE:** The solution has been tested on 64-bit machines.

* Microsoft .NET Framework 4.5 or higher
* Visual Studio 2013 Professional/Premium/Ultimate, or Visual Studio 2012 Professional/Premium/Ultimate

**NOTE:** Visual Studio paid editions are required for the WiX project to create the installer. The dev code is compatible with Visual Studio Express editions.

* Windows PowerShell 3.0 and above
* Sandcastle Help File Builder (run SandcastleInstaller.exe and install the Sandcastle Help File Builder and SHFB Visual Studio Package)
* WiX Toolset 3.9 or WiX Toolset 3.8

**NOTE:** The solution has been tested on versions 3.8 and 3.9.

* 1. Installer project dependencies

Make sure that Documents and SourceCode folders are present at the same folder level. They are used by the Installer project.

1. Feedback

You may report issues or feature requests at <https://github.com/persistentsystems/adcms/issues>.

For questions or feedback about the solution, please contact [adcms@persistent.com](mailto:adcms@persistent.com).

# Appendices

1. Implementation details

The Data Center Migration Solution is implemented in .NET Framework 4.5 and uses the Microsoft Azure Management Libraries to communicate with Azure. It can run as a stand-alone executable or via Power-Shell cmdlets and is extensible via the .NET API.

## A.1 .NET components of the solution

Table 8. Major components of the Azure Data Center Migration Solution

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Azure.DataCenterMigration.App** | A console application which calls the public APIs of **Azure.DataCenterMigration** library as per the requirement.  You can pass the input parameters required to run the solution from the configuration file or from command line parameters. |
| **Azure.DataCenterMigration** | A class library containing the business logic of the solution. It has following public APIs which can be consumed by the customer:   1. void ExportSubscriptionMetadata(IDictionary<string, string> parameters)   Where **parameters** are a collection of key value pairs required for export functionality.  This API exports information about a source subscription and stores the metadata into a JSON file.   1. void ImportSubscriptionMetadata(IDictionary<string, string> parameters)   Where p**arameters** are a collection of key value pairs required for import functionality.  This API reads the exported metadata JSON file and deploys a copy of the source resources into the destination subscription.   1. void MigrateSubscription(IDictionary<string, string> parameters)   Where **parameters** are a collection of key value pairs required for migrate functionality.  This API exports information about the source subscription and deploys a copy of the source resources into the destination subscription. |
| **Azure.DataCenterMigration.Models** | This class library contains the data models required by the solution to store the information about the resources. |

## A.2 PowerShell cmdlets

Data Center Migration Solution provides the **Azure.DataCenterMigration** PowerShell module, which contains the following PowerShell cmdlets to provide **export**, **import**, and **migrate** functionality.

Table 9. PowerShell cmdlets for export, import, and migrate

|  |  |
| --- | --- |
| Cmdlet | Description |
| **Export-AzureSubscriptionMetadata <parameters>** | This cmdlet exports information about the source subscription and stores the metadata in a JSON file. |
| **Import-AzureSubscriptionMetadata <parameters>** | This cmdlet reads exported metadata JSON file and deploys a copy of the source resources into destination subscription. |
| **Migrate-AzureSubscription <parameters>** | This cmdlet exports information about source subscription and deploys a copy of the source resources into the destination subscription. |

## A.3 Resume import operation

During **import** or **migrate** operations, the status of imported resources is updated in **{ImportMetadataFilePath}\_ImportStatus.json**. After a resource is successfully imported, the **IsImported** flag for the resource is **true** in the JSON file.

In case of failure, you can resume the **import** operation by setting the [**ResumeImport**](#ResumeImport) parameter to **true** and the [**ImportMetadataFilePath**](#ImportMetadataFilePath) to the **{ImportMetadataFilePath}\_ImportStatus.json** file.

**NOTE:** Make sure that the [**RollBackOnFailure**](#RollBackOnFailure) flag is set to **false** in all runs.

1. Resource naming

## B.1 Resource naming conventions

Table 10. Azure Data Center Migration Solution conventions for converting names during migration

|  |  |
| --- | --- |
| Resource type | Destination resource name |
| Affinity group | [**DestinationPrefixName**](#DestinationPrefixName) + source affinity group name.  Get first 63 characters of the above string. |
| VNet | [**DestinationPrefixName**](#DestinationPrefixName) + source virtual network name.  Get first 63 characters of the above string. |
| Local network | [**DestinationPrefixName**](#DestinationPrefixName) + source local network name.  Get first 63 characters of the above string. |
| DNS servers | [**DestinationPrefixName**](#DestinationPrefixName) + source DNS server name.  Get first 63 characters of the above string. |
| Storage account | [**DestinationPrefixName**](#DestinationPrefixName) + source storage account name.  Get first 24 characters of the above string. |
| Cloud service | [**DestinationPrefixName**](#DestinationPrefixName) + source cloud service name.  Get first 63 characters of the above string. |
| VM | The VMs are not renamed automatically by the solution. The solution keeps the same names as source VM. |

To change the resource names in the destination subscription, modify the resource name mapper XML.

## B.2 Resource name mapper XML

The resource name mapper XML file contains resource names in the source and respective names in the destination. During export, if [**GenerateMapperXml**](#GenerateMapperXml)parameter is set to **true**, the mapper file is generated with the following file name:

**[Source DC Name]-[Date Time of execution in the format MM-dd-yyyy-HH-mm].XML**

During export, if [**GenerateMapperXml**](#GenerateMapperXml)parameter is set to **false**, the mapper file is generated during import for future reference.

Mapper XML file has following properties:

* **SourceName.** Name of resource in source subscription
* **DestinationName.** Name of the resource in the destination subscription (**DestinationPrefixName** + **SourceName**)

To change the resource name in the destination subscription, modify the **DestinationName** property in the XML file.

1. Performance results

The Data Center Migration Solution was tested for performance under varying conditions. The results are summarized in the following table.

Table 11. Performance chart of Data Center Migration Solution for various environments

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Title** | **Azure environment details** | **Source data center** | **Target data center** | **Time for export** | **Time for blob copy** | **Time for import** |
| Migration for 10 VMs (different data centers) | Affinity groups = 200 Cloud services in location = 5 Cloud services in affinity groups = 5 Storage accounts in location = 5 Storage accounts in affinity groups = 5 VMs in affinity groups = 5 VMs in location = 5 Blobs = 20  Size of VMs = A1 | Central United States | Western Europe | 17 s | 3 h  2 m  19 s | 3 h  25 m |
| Migration for 10 VMs (same data center) | Affinity groups = 200  Cloud services in location = 5  Cloud services in affinity groups = 5 Storage accounts in location = 5 Storage accounts in affinity groups = 5 VMs in affinity groups = 5 VMs in location = 5 Blobs = 20  Size of VMs = A1 | Central United States | Central United States | 18 s | 11 s | 17 m |
| Migration for 20 VMs (different data centers) | Affinity groups = 200  Cloud services in location = 10 Cloud services in affinity groups =10 Storage accounts in location =10 Storage accounts in affinity groups = 10 VMs in affinity groups = 10  VMs in location = 10 Blobs = 40  Size of VMs = A1 | Central United States | Western Europe | 25 s | 6 h  22 m  15 s | 6 h  48 m |
| Migration for 20 VMs (same data center) | Affinity groups = 200 Cloud services in location = 10 Cloud services in affinity groups =10 Storage accounts in location =10 Storage accounts in affinity groups = 10 VMs in affinity groups = 10 VMs in location = 10 Blobs = 40  Size of VMs = A1 | Central United States | Central United States | 23 s | 14 s | 18 m |
| Migration for 50 VMs (different data centers) | Affinity groups = 200 Cloud services in location = 20 Cloud services in affinity groups =20 Storage accounts in location =10 Storage accounts in affinity groups = 10 VMs in affinity groups = 25 VMs in location = 25 Blobs = 100  Size of VMs = A1 | Central United States | Western Europe | 31 s | 15 h  41 m  8 s | 16 h  23 s |
| Migration for 50 VMs (same data center) | Affinity groups = 200 Cloud services in location = 10 Cloud services in affinity groups =20 Storage accounts in location =20 Storage accounts in affinity groups = 10 VMs in affinity groups = 25 VMs in location = 25 Blobs = 100  Size of VMs = A1 | Central United States | Central United States | 30 s | 21 s | 23 m |
| Migration for 100 VMs (different data centers) | Affinity groups = 200 Cloud services in location = 75 Cloud services in affinity groups =75 Storage accounts in location =10 Storage accounts in affinity groups = 10 VMs in affinity groups = 50 VMs in location = 50 Number of VNets = 2 VMs in VNet 2 Blobs = 204  Size of VMs = A1 | East Asia | Western United States | 1 m 45 s | 1 d  8 h  37 m | 34 h  15 m |
| Migration for 100 VMs (same data center) | Affinity groups = 200 Cloud services in location = 75 Cloud services in affinity groups =75 Storage accounts in location =10 Storage accounts in affinity groups = 10 VMs in affinity groups = 50 VMs in location = 50 Number of VNets = 2 VMs in VNet = 2 Blobs = 204  Size of VMs = A1 | Central United States | Central United States | 2 m 20 s | 52 s | 53 m |

1. <https://www.nuget.org/packages/log4net/> [↑](#footnote-ref-1)