Running Excel Calculations on the

Windows Azure Cloud with HPC Services for Excel

[Prerequisites 2](#_Toc360439692)

[Windows HPC Server 2012 SP1 2](#_Toc360439693)

[Desktop Configuration 2](#_Toc360439694)

[Windows Azure VM Nodes 2](#_Toc360439695)

[Account Information 3](#_Toc360439696)

[Using Hpcpack and Hpcsync to Stage Files to the Cloud 5](#_Toc360439697)

[Examples 5](#_Toc360439698)

[Example 1: Using a “Static” Workbook on the Azure Cloud 5](#_Toc360439699)

[Installing the Spreadsheet 6](#_Toc360439700)

[Configuring the Example 7](#_Toc360439701)

[Running the Example 8](#_Toc360439702)

[Example 2: Using “Dynamic” Spreadsheets on the Azure Cloud 9](#_Toc360439703)

[Building the SyncWorkbooks Package 10](#_Toc360439704)

[The SyncWorkbooks.exe binary will be dropped to SyncWorkbooks\bin\debug and the Visual Studio project has a post-build event to create the SyncWorkbooks.zip package in that directory. 10](#_Toc360439705)

[Configuring the Azure Nodes 10](#_Toc360439706)

[Configuring the Example Spreadsheet 13](#_Toc360439707)

[Running the Workbook Calculation 14](#_Toc360439708)

[Example 3: Using Custom SOA Services with Excel and Windows Azure 15](#_Toc360439709)

[Installing the Packages 16](#_Toc360439710)

[Running the Sample Code 18](#_Toc360439711)

[Using Windows Azure Connect 19](#_Toc360439712)

# Prerequisites

## Windows HPC Server 2012 SP1

At a minimum, to run HPC calculations on Azure compute nodes you will need an HPC Scheduler installed and configured within your network. Installing HPC Server is outside of the scope of this document. See the Windows HPC Server 2012 SP1 installation instructions for more information.

## Desktop Configuration

On your desktop (where you want to run Excel calculations), you will need Excel 2010 installed, either the 32-bit or the 64-bit version; and the HPC client utilities. The HPC client utilities are libraries which allow your desktop to communicate with the HPC server. You can install the HPC client utilities either from the standard HPC server install package, or using a stand-alone client library installer available at

<http://www.microsoft.com/download/en/details.aspx?id=17017>

The client libraries also include Excel components used to run calculations directly from Excel VBA.

## Windows Azure VM Nodes

You will also need Azure VM nodes configured and deployed, with Excel installed on the VM image. Step-by-step instructions for configuring and deploying Azure VM nodes are available at

<http://technet.microsoft.com/en-us/library/hh184325(WS.10).aspx>

Pay special attention to Step 4 in this guide; make sure to install the Windows Azure Integration components and the Windows HPC Server components (this is not the standard HPC installation; these are specific components for Azure VMs). In the step “Install and configure your applications”, install and activate Excel 2010 on your VM image.

After installing Excel on your VM image, you should use Windows Update one more time to ensure you have the latest Excel patches installed (but after that, make sure Windows Update is disabled).

For Excel calculations, we are not using MPI so you can skip the steps related to MPI: installing the mpi\_x64.msi package and configuring the firewall for MPI. If you want to use the same Azure VM nodes for MPI applications, then include these steps.

Windows Azure Connect is a service which provides direct connections among Azure VM instances and your local network, using IPSec. If you can use Windows Azure Connect, it can make the process of running Excel calculations much easier (because you can use the filesystem to stage spreadsheet files for calculation). In this document we will discuss both situations: when you have Windows Azure Connect available, and when you cannot use or cannot install Windows Azure Connect.

To use Windows Azure Connect, before finalizing the VM image make sure to follow the additional steps on this page:

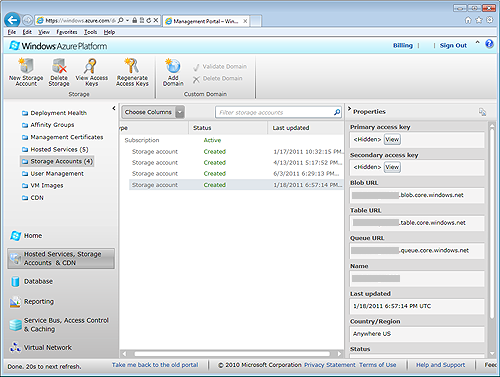
<http://msdn.microsoft.com/en-us/library/gg697584.aspx>

Using Windows Azure Connect requires configuring both the VM image and local endpoints (machines in your network), and may require some changes to your network firewall.

## Account Information

At various points in the steps below you will need information on your cloud storage account and some information from your HPC server installation.

### Azure Storage Account Name and Account Key

To use Azure storage as a staging area for Excel files, you will need the name of a storage account and the associated storage key. To find these, open the Windows Azure console in a web browser (<http://windows.azure.com>). If necessary, sign in to your account.

< Storage account name

< Storage account keys

At the bottom left, click **Hosted Services, Storage Accounts and CDN**. On the left-hand side, click **Storage Accounts**. The center panel will show a list of your storage accounts. Click the account you would like to use. It should be highlighted once you’ve selected it.

On the right-hand side of the screen, find the section marked **Name**. This is the storage account name you will need later. At the top of the right-hand side are the storage account keys. Click the **View** button in either section. This will open a dialog box showing both the Primary and the Secondary key. The buttons next to the keys will copy them to the clipboard – this helps avoid copy and paste errors. For these test services, we should use the Secondary key. Click the button next to the secondary key (at the bottom-right) to copy the key to the clipboard. Then paste the key into a text editor so you can use it later.

### HPC Server Node Template and Job Template

When uploading packages to the cloud, you will need the name of the node template used to create and manage your Azure compute nodes.

To find the name of the node template, open the HPC Cluster Manager application (from the Windows Start menu, **Programs** > **Microsoft HPC Pack 2012** > **HPC Cluster Manager**). At the bottom left, click **Node Management**. In the list of nodes, one of the columns is “Node Template”. Locate one of your Azure compute nodes in the node list. The node template name shown for this node is the name you need.

The default node template name for Azure nodes is “Default AzureNode Template”. We will use that template name in the steps that follow. If you use a different name for your node template, be sure to change it when you run the examples.

You should also create a Job Template. Job templates are a feature of Windows HPC Server which allow you to select specific resources and set parameters when running cluster or cloud calculations. In the examples below, we use a job template to ensure that the jobs we run will use only Azure compute nodes and not cluster (local, on-premises) nodes. Some of this code is specific to cloud nodes, so we want to make sure that it does not run on local compute nodes.

To create a job template, in the HPC Cluster Manager click **Configuration** at the bottom left. In the list at the top-left, select **Job Templates**. On the right-hand side, click **New…** In the dialog box, enter a name for this job template. For simplicity, we suggest using the name “AzureTemplate” (without quotes). We will use the name “AzureTemplate” as the job template in the examples below, so if you use a different name be sure to change it when running the examples.

Click **Next** through the next three pages, accepting the default options, until you come to the page **Limit Node Groups**. On this page, click the radio button **Allow only the selected node groups**. In the list, check **AzureNodes** and leave the others un-checked. Then click **Next** and then click **Finish**. You should now see an entry in the list of Job Templates named “AzureTemplate”.

### Windows Azure SDK

To build the included sample code, the Windows Azure SDK and Visual Studio are required. These can be downloaded from the following locations:

* Visual Studio: http://msdn.microsoft.com/en-us/vstudio/bb984878
* Windows Azure SDK: http://www.microsoft.com/windowsazure/sdk/

### CloudBerry

Although not required, CloudBerry is a useful tool for working with Azure storage. It provides an interface similar to Windows Explorer for browsing and navigating cloud storage accounts. For more information, see:

<http://cloudberrylab.com/?page=explorer-azure>

# Using Hpcpack and Hpcsync to Stage Files to the Cloud

**Hpcpack** is a tool provided with Windows HPC Server 2012 SP1 which packages files for staging on Azure cloud VM nodes. You can use it to package HPC services or regular files. A “package” is just a zip file with some additional metadata used for installation. The hpcpack tool is used both to create packages and to upload those packages to Azure storage.

When you start Azure VM nodes in the HPC Cluster Manager, any software which has been previously staged – including SOA services and file packages – will be automatically installed on the nodes. This is done when the nodes are started, using a tool called **hpcsync**. Any package which has been created and uploaded using hpcpack will be automatically installed on startup.

If the nodes are already running, and you create or modify a package using hpcpack, you can run hpcsync manually to ensure that the latest package version is installed on your cloud nodes. Hpcpack runs on the nodes themselves, so to run it manually we use the **clusrun** tool, part of the HPC server installation.

In each of the examples below, we will use hpcpack and hpcsync to move files to the Azure cloud and configure the Azure VM nodes. For more information on these tools, see

<http://technet.microsoft.com/en-us/library/gg481764(WS.10).aspx>

<http://technet.microsoft.com/en-us/library/gg481752(WS.10).aspx>

# Examples

In the sections that follow, we provide sample code and discuss three basic models for using HPC Services for Excel with Windows Azure VM compute nodes. These are just examples, but we will try to describe common use cases and situations in which you might want to use one model or another.

# ExcelService Configuration

Before run the excel service, we should deploy excel service on AzureNode first.

1. Package the dependency files.

Run the following command:

> hpcpack create Microsoft.Hpc.Excel.ExcelService.zip "%SYSTEMROOT%\Microsoft.NET\assembly\GAC\_MSIL\Microsoft.Hpc.Excel\v4.0\_2.0.0.0\_\_31bf3856ad364e35\Microsoft.Hpc.Excel.dll","%SYSTEMROOT%\Microsoft.NET\assembly\GAC\_MSIL\Microsoft.Hpc.Excel.Internal\v4.0\_4.0.0.0\_\_31bf3856ad364e35\Microsoft.Hpc.Excel.Internal.dll","%CCP\_HOME%Bin\Microsoft.Hpc.Excel.tlb","%CCP\_HOME%Bin\Microsoft.Hpc.Excel.ExcelService.dll","%CCP\_HOME%ServiceRegistration\Microsoft.Hpc.Excel.ExcelService\_1.0.config","%CCP\_HOME%Bin\CosmosLogging.dll","%CCP\_HOME%Bin\Microsoft.Hpc.Trace.dll"

1. Upload the file to Azure Storage

> hpcpack upload Microsoft.Hpc.Excel.ExcelService.zip /scheduler:HEADNODE /nodetemplate:"Default AzureNode Template"

1. Sync the file to AzureNode

> clusrun /scheduler:HEADNODE /template:AzureTemplate hpcsync

# Example 1: Using a “Static” Workbook on the Azure Cloud

The first example presents the simplest method for running Excel workbook calculations on the Azure cloud. This method is designed to use a fixed (“static”) spreadsheet, which is stored on the Azure cloud. Once the spreadsheet is in place, we can run calculations using that spreadsheet directly from Excel VBA.

The primary benefit of this model is that it requires no extra software (just Excel and the HPC client utilities). It is flexible enough to run just about any Excel calculation, and it is especially useful for VBA developers or Excel users.

The main limitation of this model is that it requires Administrator permissions to move the spreadsheet to the cloud. That means that regular users won’t be able to install new or modified spreadsheets themselves.

Another benefit of this approach is that in many cases it will be the fastest way to use Excel and Azure. Because the spreadsheet is already present on the cloud nodes, the calculation can start almost immediately – it does not have to copy the spreadsheet before it starts running.

## Installing the Spreadsheet

The first step in this example is installing the spreadsheet on the Azure VM nodes. Make sure you have Azure VM nodes configured (see the above section “Prerequisites”). They should be started and in the “online” state in HPC Cluster Manager.

The spreadsheet we will use is called “ConvertiblePricing\_AzureCloud\_Static.xlsb”. In the install kit, it is in the directory “Example1”. This spreadsheet uses the HPC / Excel VBA macro framework for cluster and cloud calculations.

To install this spreadsheet on the Azure cloud, we need to create a package and then upload that package to Azure storage. You can do this from either your desktop or the HPC Head Node, but you will need cluster Administrator permissions.

### Create the package

Open a command window or a PowerShell window as an Administrator (PowerShell is available in the Windows Start menu, under **Programs** > **Microsoft HPC Pack 2012** > **HPC PowerShell**).

In the shell, navigate to the directory “Example1” in the install kit. To create the package, run the following command (this command is a single line; don’t insert a newline):

> hpcpack create ConvertiblePricing\_AzureCloud\_Static.zip ConvertiblePricing\_AzureCloud\_Static.xlsb

The package will be named “ConvertiblePricing\_AzureCloud\_Static.zip”, and it contains a single file (the spreadsheet). We want the name of the zip file to match the name of the spreadsheet, because the HPC/Excel calculation library uses this to locate the spreadsheet. If you are building a package with your own spreadsheet, make sure that the name of the spreadsheet matches the name of the package. The only difference is the extension (packages should always have a “.zip” extension).

### Upload the package

Next, upload the package to Windows Azure. For this step you will need (1) the name of your cluster scheduler; and (2) the name of the node template you used to install Azure VM nodes. See the above section ”HPC Server Node Template and Job Template” if you don’t know the name of your node template.

In the shell, run the following command. Substitute the name of your cluster head node for the HEADNODE value. If you are not using the default node template name, change “Default AzureNode Template” to match your node template. If there are spaces in the template name, it must be enclosed in quotes. (This command is a single line, don’t enter a newline).

> hpcpack upload ConvertiblePricing\_AzureCloud\_Static.zip /scheduler:HEADNODE /nodetemplate:"Default AzureNode Template"

If you are running the command on the head node, you can omit the “/scheduler:HEADNODE” parameter.

### Synchronize Azure Nodes

When you start your Azure VM nodes from the HPC manager, any software that has previously been uploaded using “hpcpack upload” will be automatically installed. However if the nodes are already running, there is one more step: to synchronize the nodes so the latest software is installed.

We can synchronize nodes using the **hpcsync** tool. Hpcsync runs on the Azure nodes, so we run it using the **clusrun** application.

Clusrun can run commands on any or all cluster compute nodes. In this case, we only want to run the command on Azure nodes. We do that using a job template. In the above section “HPC Server Node Template and Job Template” we described creating a job template specifically for Azure nodes.

You can run this command without a job template (on all nodes); if you have any cluster nodes, the command will fail, but it will still run on the Azure nodes.

Run the following command. Change the value of HEADNODE to match your cluster head node. If you did not use the job template name “AzureTemplate” we suggested above, then change that value in the command to match your job template. If the template name includes spaces, you must enclose it in quotes.

> clusrun /scheduler:HEADNODE /template:AzureTemplate hpcsync

When the command completes, you will see a lot of status information about deploying packages. The important line is at the end: it will tell you if the command succeeded or failed on the Azure nodes.

*In the beta release of Windows HPC Server 2012 SP1, this command may fail the first time. If you see an error “unable to create user account”, try running the command again and see if it succeeds the second time.*

The package containing the spreadsheet will now be installed on the Azure nodes. Any user who has permissions to run calculations on the HPC cluster can now use this spreadsheet on the cloud nodes.

## Configuring the Example

To run the example, open the spreadsheet file on your desktop. We are using the spreadsheet “ConvertiblePricing\_AzureCloud\_Static.xlsb” from the directory “Example1” in the install kit.

The spreadsheet uses VBA macros. If you see a warning about active content, click “Enable this Content” to enable VBA.

Before running on the cloud, we need to make one change to the spreadsheet. The spreadsheet VBA code connects to the cluster using the name of the cluster head node. Open the VBA editor (From Excel, press Alt+F11). In the VBA editor, in the tree view on the top of the left-hand side, double-click the module named “HPCControlMacros” to open it.

In this file, locate the line

Private Const HPC\_ClusterScheduler = "HEADNODE"

Change the value of HEADNODE to match your cluster head node. This should be the machine name. If you are in a different subdomain, use a qualified name (e.g. mydomain.myheadnode).

The file also specifies the name of the Job Template, which we used above when installing the spreadsheet on the cloud. The file uses the name we suggested, “AzureTemplate”. If you used a different job template name, then find the line

Private Const Azure\_jobTemplate = "AzureTemplate"

and change the value “AzureTemplate” to match your job template name. Remember that this is the job template, not the node template.

## Running the Example

Close the VBA window so you are back at the spreadsheet. The example spreadsheet includes code to run both on the desktop and on the Azure cloud. Run on the desktop first, to see how it works.

Click the button “Calculate on Desktop”. You will see the table of values cleared, and then it will fill in one cell at a time. The spreadsheet uses VBA code to run calculations, and fills in individual cells with the calculation results.

Once the calculation is complete, try running on the cloud. Click the “Calculate on Cloud” button. If you see a dialog with an error message, verify that your settings are correct in the VBA code (from the last section).

This time you should see the table fill in several cells at a time, depending on how many cloud nodes you have running. In this model, the spreadsheet sends a request to the HPC head node to run a single calculation for each cell. Those requests are passed to the Azure cloud nodes. The Azure cloud nodes load the spreadsheet (from the package we created and uploaded in the last section) and run calculations, returning results to the spreadsheet on your desktop.

The more nodes you have available, the faster the calculation will run, because it can run multiple calculations in parallel. This spreadsheet is somewhat simple, so there is a limit on how fast it will run. But the same pattern can be used with almost any spreadsheet.

How does it identify the spreadsheet to use on the cloud nodes? If you go back to the VBA editor (Alt+F11 in Excel), in the module “HPCControlMacros”, look for the subroutine “CalculateWorkbook”.

This routine uses the HPC/Excel calculation macros (the structure of these macros is outside of the scope of this document; look at the examples provided with Windows HPC Server). To calculate on the cloud, we initialize the client library with the line

hpcexcelclient.OpenSession HPC\_ClusterScheduler, "ConvertiblePricing\_AzureCloud\_Static.xlsb", 1, 128, SessionUnitType\_Core, Azure\_jobTemplate

In this line, we specify the name of the spreadsheet. When you created the package for Azure, in the previous section, the name of the package was the same as the name of the spreadsheet. When the HPC/Excel calculation service runs on cluster compute nodes, it will use the name to try and identify a package containing the spreadsheet.

# Example 2: Using “Dynamic” Spreadsheets on the Azure Cloud

In the first example, we used the HPC/Excel calculation service to connect to Azure compute nodes and to run Excel calculations from VBA. The standard Excel service and the VBA client libraries fully support calculation on Windows Azure. However, in that example we noted a limitation – the spreadsheet must already be installed on the cloud nodes, and it must be installed by an Administrator.

This is a limitation if you are working with a spreadsheet that changes frequently, or if you need to change various parameters within the spreadsheet every time you run it.

If you are running HPC/Excel calculations on a cluster (on local, on-premises nodes), it is possible for non-Administrators to deploy a spreadsheet to cluster nodes for calculation. Typically this is done using a network share directory on the filesystem. You can copy your spreadsheet to the network share, and the compute nodes can read it from there. This is easy to automate using VBA, so you have a complete end-to-end solution entirely within Excel.

When you use Azure nodes, on the other hand, there is no shared filesystem, so this won’t work[[1]](#footnote-1). We need to find another way to transfer spreadsheets to the Azure cloud compute nodes.

We can use hpcpack to package spreadsheets and upload packages to cloud storage. This will work for regular users, and it’s possible to automate this from Excel VBA. But we can’t use hpcsync to download the packages to the cloud nodes, because that requires Administrator permissions.

The solution is to use a separate application to download spreadsheets to the cloud nodes just before the workbook is calculated. This application must be installed on the Azure compute nodes, but once it has been installed it will work for regular users.

The application “SyncWorkbooks” provided with this example will download spreadsheet packages from Azure storage, based on the package name, and unpack them into a directory on cloud compute nodes. We can ensure that this runs just before the spreadsheet is calculated by using a *node preparation* task. Node preparation (“node prep”) tasks are a feature of Windows HPC Server 2012 SP1. Node prep tasks are executed on any node used for an HPC job, and they are typically used to prepare or preload resources. In this case, we will use the node prep task to download spreadsheets for calculation.

There is one additional requirement: we need a directory on the Azure nodes to use as a working directory, where the spreadsheet will be downloaded and run. We also need to ensure that regular users have read/write access to this directory.

It is possible to create a working directory manually on the compute nodes, using clusrun (and we will do this in the next section, to get started). However any directory created this way will disappear when nodes are stopped and re-started, or if the nodes are redeployed. We can automate this process using a *startup script*. Startup scripts are another feature of Windows HPC Server 2012 SP1 which can execute configuration scripts when the node is started.

This approach is more flexible than the approach used in the first example, because it allows users to upload their models to the cloud with any recent changes. However the drawback of this approach is that the file must be copied to Azure storage every time it is run. This can add latency to the calculation, particularly if the spreadsheet file is very large.

## Building the SyncWorkbooks Package

The workbook sync application is a simple C# program which downloads spreadsheet packages from cloud storage and copies them to a local directory on the Azure nodes. The source code for this application is included in the “SyncWorkbooks” directory in Example2 of the install kit. To build this application, open SyncWorkbooks.sln in Visual Studio and build the project.

## The SyncWorkbooks.exe binary will be dropped to SyncWorkbooks\bin\debug and the Visual Studio project has a post-build event to create the SyncWorkbooks.zip package in that directory.

## Configuring the Azure Nodes

The steps required to configure the Azure nodes are (1) installing the workbook sync application; (2) setting up the node prep task; and (3) creating the working directory.

### Installing the Workbook Sync Application

Open a terminal window or a PowerShell window as an Administrator (PowerShell is available from the Windows Start menu, under **Programs** > **Microsoft HPC Pack 2012** > **HPC PowerShell**). In the shell, navigate to the directory Example2 \SyncWorkbooks\bin\debug” in the install kit.

Run the following command. Change the values for HEADNODE to match your cluster scheduler. If you are not using the default node template name, change the value to match your node template. If the template name contains spaces, it must be enclosed in quotes. (This command is a single line, don’t enter a newline).

> hpcpack upload SyncWorkbooks.zip /scheduler:HEADNODE /nodetemplate:"Default AzureNode Template" /relativePath:SyncWorkbooks

In this command, we are uploading the package to the Azure nodes using hpcpack. The “relativePath” parameter is used to set a fixed path to the application, so we can call it from the node prep task.

Once the package has been uploaded, synchronize running nodes with hpcsync:

> clusrun /scheduler:HEADNODE /template:AzureTemplate hpcsync

### Configuring the Node Prep Task

The next step is to set up a node prep task which will call the workbook sync application whenever you calculate Excel spreadsheets on Azure compute nodes. For this step you will need information about your cloud storage account: specifically you will need the name of a storage account and the account key. See the above section “Account Information” for help on locating these values.

The Excel calculation service is an HPC SOA service. Node prep tasks can be configured in the service configuration file for any SOA service. In this step we will edit the service configuration file to add a node prep task, as well as a *node release task* which is run when the service completes. The node release task is not required, but we can use it to clean up temporary files on the Azure nodes.

Edit the service configuration file for the Excel calculation service. This file is located in the Service Registration directory on your HPC head node. By default, the head node shares this directory on the network when HPC is installed. Open a Windows Explorer window and browse to your head node using the network name (e.g. \\HEADNODE). Open the “HpcServiceRegistration” directory.

The configuration file for the standard HPC/Excel service is “Microsoft.Hpc.Excel.ExcelService\_1.0.config”. Back up this file before you edit it just in case. Then open the file in a text editor or XML editor. Scroll down to find the section

<microsoft.Hpc.Session.ServiceRegistration>

<service assembly="%CCP\_HOME%Bin\Microsoft.Hpc.Excel.ExcelService.dll"

contract="Microsoft.Hpc.Excel.IExcelService"

type="Microsoft.Hpc.Excel.ExcelService"

includeExceptionDetailInFaults="true"

maxConcurrentCalls="1"

serviceInitializationTimeout="60000">

</service>

</microsoft.Hpc.Session.ServiceRegistration>

Change this section so it reads as follows (you can copy and paste over the old configuration section). The changes are highlighted in yellow. The lines “prepareNodeCommandLine” and “releaseNodeCommandLine” are the node prep and node release tasks. Each of these should be entered as a single line (there is no line break). We are also adding some environment variables for configuration.

<microsoft.Hpc.Session.ServiceRegistration>

<service assembly="%CCP\_HOME%Bin\Microsoft.Hpc.Excel.ExcelService.dll"

contract="Microsoft.Hpc.Excel.IExcelService"

type="Microsoft.Hpc.Excel.ExcelService"

includeExceptionDetailInFaults="true"

maxConcurrentCalls="1"

serviceInitializationTimeout="60000"

prepareNodeCommandLine="powershell if(Test-Path env:CCP\_ONAZURE){ %CCP\_PACKAGE\_ROOT%\SyncWorkbooks\SyncWorkbooks.exe install }"

releaseNodeCommandLine="powershell if(Test-Path env:CCP\_ONAZURE){ %CCP\_PACKAGE\_ROOT%\SyncWorkbooks\SyncWorkbooks.exe cleanup }"

>

<environmentVariables>

<add name="WORKING\_DIRECTORY" value="C:\Resources\WorkbookTemp"/>

<add name="CLOUD\_ACCOUNT" value="storageAccountName"/>

<add name="CLOUD\_KEY" value="storageAccountKey"/>

</environmentVariables>

</service>

</microsoft.Hpc.Session.ServiceRegistration>

After you have made this change, update the values of the environment variables to match your cloud storage account information. Change the value “storageAccountName” to the name of your cloud storage account. Change the value “storageAccountKey” to your account key (this is the long base-64 string).

Finally, save the configuration file.

### Creating the Working Directory on Azure Compute Nodes

As described above, we want to use a startup script to create a working directory so that it will be available on new Azure nodes or when Azure nodes are re-deployed. If your nodes are already running, you can use the same commands to create a directory with clusrun.

The startup script is included in the install kit in the “Example2” directory. Upload the package to Azure storage using the following command in your shell:

> hpcpack upload StartupScript.bat.zip /scheduler:HEADNODE /nodetemplate:"Default AzureNode Template"

Once again change the value of HEADNODE to your head node, and if you are using a different node template update that value as well. The name of the package (StaruptScript.bat.zip) is important because this is how HPC identifies the package when it starts the node – we use the same name in the node template editor in the next step.

To set the startup script for the node template, open the HPC Cluster Manager (from the Windows Start menu, **Programs** > **Microsoft HPC Pack 2012** > **HPC Cluster Manager**). At the bottom-left click **Configuration**. On the left-hand side, click **Node Templates**. Right-click the node template you are using (if you accepted the default name, this will be “Default AzureNode Template”) and click **Edit**.

In the node template editor, click the **Startup Script** tab and set the name of the startup script to “StartupScript.bat” (no quotes). This should match the name of the package you uploaded earlier, without the .zip extension. Click **Save** to close the template editor.

The startup script is just two lines: creating a directory and setting permissions. If you have an existing startup script, you can add these commands to your script.

If you have nodes already running, you can run these commands manually rather than restarting the nodes. In the HPC Cluster Manager application, click **Node Management** at the bottom-left. Select all your Azure nodes (shift-click to select multiple). Then right-click the selection and click **Run Command**.

In the command line window, enter

mkdir C:\Resources\WorkbookTemp

to create a directory. Wait for the command to complete; then run the command

cacls c:\resources\workbookTemp /e /g Everyone:f

to set access rights to the directory.

## Configuring the Example Spreadsheet

To configure the example, you will need the name of your head node; and the Azure cloud storage account name and account key you used in the previous section, when configuring the node prep task.

Open the spreadsheet “ConvertiblePricing\_AzureCloud\_Dynamic.xlsb” from the “Example2” directory in the install kit. This spreadsheet will look identical to the spreadsheet from the first example, but it includes additional VBA code to support automatically deploying the file to the Azure cloud.

The spreadsheet uses VBA macros. If you see a warning about active content, click “Enable this Content” to enable VBA.

Before running on the cloud, we need to make some configuration changes in the spreadsheet. Open the VBA editor (From Excel, press Alt+F11). In the VBA editor, in the tree view on the top of the left-hand side, double-click the module named “HPCControlMacros” to open it.

If you have the spreadsheet from the first example still open, there may be multiple files in the VBA editor. Make sure you are looking at the “HPCControlMacros” module in the file “ConvertiblePricing\_AzureCloud\_Dynamic.xlsb”.

In this file, locate the line

Private Const HPC\_ClusterScheduler = "HEADNODE"

Change the value of HEADNODE to match your cluster head node. This should be the machine name. If you are in a different subdomain, use a qualified name (e.g. mydomain.myheadnode).

Next find the line

Private Const Azure\_cloudStorageAccount = "storageAccountName"

and change the value “storageAccountName” to your Azure storage account name (enclosed in quotes). Find the line

Private Const Azure\_cloudStorageAccountKey = "storageAccountKey"

and change “storageAccountKey” to your account key (the long base-64 string).

The file also specifies the name of the Job Template, which we used above when installing the spreadsheet on the cloud. The file uses the name we suggested, “AzureTemplate”. If you used a different job template name, then find the line

Private Const Azure\_jobTemplate = "AzureTemplate"

and change the value “AzureTemplate” to match your job template name. Remember that this is the job template, not the node template.

Close the VBA editor and switch back to Excel.

## Running the Workbook Calculation

This spreadsheet runs in the same way as the spreadsheet from the first example. To run locally, click the button “Calculate on Desktop”. To run on the cloud, click the button “Calculate on Cloud”.

When you calculate the spreadsheet on the cloud, it uses a VBA routine to package and upload the spreadsheet. The spreadsheet is stored in Azure storage and when the calculation runs, the node prep task – which you configured in the previous section – calls the “SyncWorkbooks” program which downloads the file to the individual Azure node.

Compared to the first example, this spreadsheet will take a little longer to start the calculation. That additional time includes (1) the time to package the spreadsheet; (2) the time to upload the spreadsheet; and (3) on the Azure compute nodes, the time to download and unpack the file. Once the model starts calculating, the performance should be roughly identical to that of the first model.

The VBA code used to package and upload the spreadsheet is in the VBA function “StageWorkbook”. Open the VBA editor and scroll down to this function. Essentially, this runs the following steps:

1. The function saves a temporary copy of the spreadsheet to the local TEMP directory. We save a copy in order to ensure that any recent changes are preserved.
2. Next the function uses hpcpack to create a package for Windows Azure. This is the same function call that you used to create a spreadsheet package in the first example. The package contains the single spreadsheet, but it could also contain dependent files if necessary.
3. Next the function uses hpcpack to upload the package to Azure storage. Here we use the storage account name and storage account key to specify where the files should be stored. The account name and account key used here must match the values set in the node prep task (from the last section). We are also setting a relative path (via the relativePath parameter) so that there is a static path to the model – we use this to tell Excel where to load the file.

At the end of the function, it returns a path to the spreadsheet that can be used to load it on the Azure compute nodes (after the file has been downloaded by the node prep task). That path is used to initialize the HPC/Excel library.

Other than this method (and the required parameters), there is no difference between this spreadsheet and the spreadsheet from the first example; so it should be easy to “drop in” this code if you want to support regular users running Excel models on the Azure cloud.

# Example 3: Using Custom SOA Services with Excel and Windows Azure

The default HPC/Excel service provided with Windows HPC Server 2012 SP1 supports running calculations directly from Excel VBA, as in the previous examples. If you have more specific requirements, or if you want to build a service which uses Excel to perform background calculations, you can construct your own service using the basic HPC/Excel libraries. In this example, we build a custom SOA service which runs on Azure cloud nodes and which uses Excel to perform a specific calculation. We also build a custom client application, which runs on the desktop and uses the cloud service.

The files in the “Example3” directory include (1) a spreadsheet which is used to run calculations; (2) a SOA service which runs calculations using that spreadsheet; and (3) a client application which calls the service, passes input parameters, and receives calculation results.

The spreadsheet used here is a simple asset pricing worksheet: it calculates option prices using the familiar Black-Scholes model. Our aim is to demonstrate how you can use a spreadsheet as the back-end calculation for an HPC/SOA service running on the cloud. While this is a simple model, the same framework can support arbitrarily complex calculations, using just about any Excel spreadsheet (including spreadsheets which use VBA).

The SOA service which will be installed is designed to load the workbook; set parameter values in the spreadsheet; and then retrieve calculated results. The source code for the service is in the “Sources” directory in the install kit.

## Installing the Packages

For this example, we will install an HPC SOA service on the Azure cloud; and then install spreadsheets to be used with the service, in a separate package.

### Building the SOA Service and Client

To build these programs, open ExcelAzureSamples.sln, located in the “Example3” directory in the install kit, with Visual Studio and build both projects.

The StaticWorkbookService binaries will be dropped to StaticWorkbookService\bin\debug and the Visual Studio project has a post-build event to create the StaticWorkbookService.zip package in that directory. Likewise, the StaticWorkbookClient.exe will be dropped to StaticWorkbookClient\bin\debug.

### Installing the SOA service

The first step in installing the SOA service is packaging it for distribution to the cloud nodes. We use hpcpack to create the package, then use hpcpack again to upload it to the cloud. The Visual Studio post-build event takes care of creating the package, so the package just needs to be uploaded to Azure storage.

Open a command window or a PowerShell window as an Administrator (PowerShell is available in the Windows Start menu, under **Programs** > **Microsoft HPC Pack 2012** > **HPC PowerShell**). In the shell, navigate to the “Example3\StaticWorkbookService\bin\debug” directory in the install kit.

In the shell, run the following command. Substitute the name of your cluster head node for the HEADNODE value. If you are not using the default node template name, change “Default AzureNode Template” to match your node template. If there are spaces in the template name, it must be enclosed in quotes. (This command is a single line, don’t insert a newline).

> hpcpack upload StaticWorkbookService.zip /scheduler:HEADNODE /nodetemplate:"Default AzureNode Template"

The code will now be automatically deployed on your Azure nodes when they are started. If your Azure VM nodes are already started, there is one more step to make sure that each node has the latest code installed: using hpcsync to synchronize all nodes.

If your nodes are started, first make sure they are in the “online” state. Then run the following command. Change the value of HEADNODE to match your cluster head node. If you did not use the job template name “AzureTemplate” we suggested above, then change that value in the command to match your job template. If the template name includes spaces, you must enclose it in quotes.

> clusrun /scheduler:HEADNODE /template:AzureTemplate hpcsync

The next step is to configure the cluster scheduler (the head node). To configure the head node, copy the service configuration file to the scheduler’s service registration directory. If you have configured SOA services for a local cluster, this step is the same; except that with cloud services, we do not need to specify the path to the service library (DLL).

Copy the file “StaticWorkbookService.config” from the “Example3” directory in the install kit to the Service Registration directory on your cluster scheduler. In a default installation, this directory is shared, so you should be able to find it by opening a Windows Explorer window and navigating to your scheduler (using the UNC path \\HEADNODE). There will be a directory called “HPCServiceRegistration”. Copy the config file into that directory.

If the directory is not shared on the network, copy the file to the head node and then into the service registration directory; in a normal installation this is (on the cluster head node)

C:\Program Files\Microsoft HPC Pack 2012\ServiceRegistration

### Packaging and Uploading Spreadsheet Files

Packaging spreadsheet files for the cluster is the same as packaging the SOA service. In the install kit, the example spreadsheets are in the directory “Example3\SpreadsheetFiles”. To package these files, in your shell run the command

> hpcpack create spreadsheets.zip "SpreadsheetFiles"

This will create the package. The next step is to upload the package to cloud storage.

Uploading spreadsheets (or any other files) to the cloud is the same as uploading the SOA service, except that we want to specify a relative path. When installing a SOA service, the cloud nodes use a default path that contains some versioning and timestamp information. For static files, we want to use a path that we can specify later in the client code. To do that, use the option /relativePath with hpcpack. We will install the spreadsheets in the relative directory “workbooks”.

In your shell, run the following command. Change the value of HEADNODE to match your head node. If you are not using the default node template name, change the value to match your node template. This is a single line, don’t enter a newline.

> hpcpack upload spreadsheets.zip /scheduler:HEADNODE /nodetemplate:"Default AzureNode Template" /relativePath:workbooks

As in the last step, if your Azure VM nodes are already running, you will need to synchronize them to ensure that the latest code is downloaded. Use hpcsync:

> clusrun /scheduler:HEADNODE /template:AzureTemplate hpcsync

The SOA service and the spreadsheets are now installed on the Azure VM nodes. If you like, you can connect directly to an Azure node and look at the file structure. In the HPC Cluster Manager application, right-click one of the Azure nodes and select “Remote Desktop”. You can also connect to a running Azure VM node from the Windows Azure console (in your web browser).

The package files will be installed in the drive labeled “Resources” (the specific drive letter will be different depending on the size of the Azure node), in the “Application” directory. The SOA service we uploaded first will be in a directory named “StaticWorkbookService”, and will contain versioning information. The spreadsheets will be in the directory “Workbooks”, as specified when the package was uploaded.

You can re-package, upload and sync the spreadsheets any time you need. If your Azure nodes are not running, follow the steps for packaging and uploading the spreadsheets. If your Azure nodes are running, you must also run the hpcsync step to deploy updated packages to the running nodes.

## Running the Sample Code

As we described earlier, the spreadsheet contains a simple asset pricing model. The SOA service installed in the last step is designed to load the spreadsheet on the Azure compute nodes, and use the model to perform calculations. We can now use a client program to connect to that service, send arbitrary input values, and receive the calculation results.

The client program provided is a shell application which calls the service. When you run the shell application, it will request pricing for a variety of options; the service will respond with calculated prices. The source code for the client application is provided in the “Sources” directory in the install kit.

To run the shell application, open a terminal window or a PowerShell window and navigate to the directory “Example3\StaticWorkbookClient\bin\debug” in the install kit. You will need the name of your cluster scheduler (the head node), and the name of the Job Template you created earlier. There are some additional parameters you can change if necessary, either in the code or with command-line arguments.

Run the following command, changing HEADNODE to the name of your head node and changing the value of the job template parameter (if necessary). If your job template name includes spaces, it must be enclosed in quotes.

> StaticWorkbookClient.exe -headnode HEADNODE -jobTemplate AzureTemplate

The application will first connect to your cluster scheduler, creating a session to run the service. If you open the HPC Cluster Manager application, you will see a job created for the “StaticWorkbookService”.

Once the session has started, the client application will submit a set of parameters for calculation – by default, the program sends 240 requests. On the Azure VM nodes, the SOA service library will open Excel, and for each request, one of the nodes will run the calculation and return the resulting price back to the client.

Once again, this is a simple model but it should illustrate how you can build calculation services using Excel workbooks, and how you can build client applications to call those services. You can use almost any Excel spreadsheet as the back-end calculation service, and the client code can be integrated in any Windows application, including shell applications, visual / GUI applications, or even plug-ins for other programs.

# Using Windows Azure Connect

Windows Azure Connect provides virtual network connectivity between Azure VM nodes and your local network. If you can install and use Windows Azure Connect, it’s possible to join your Azure VM nodes to your local domain – so that they will be able to access network files and network shares on your domain.

When you run an HPC job, including a SOA service for Excel, it runs with your username and your account permissions. When running on the cloud, if Windows Azure Connect is enabled and the cloud VM is connected to your domain, the service will be able to locate files on share directories within your domain.

In this model, the Azure VM compute nodes operate as if they were local compute nodes within your HPC cluster. Therefore it’s possible to run Excel calculations the same way they run on a local cluster.

For more information on using Windows Azure Connect with your domain, see

<http://www.microsoft.com/windowsazure/virtualnetwork/>

<http://msdn.microsoft.com/en-us/library/gg454720.aspx>

1. Actually it is possible to use the filesystem, if you are using Windows Azure Connect – see the last section for more information on using Connect. This section applies if you are not using Azure Connect, or if there’s no shared directory accessible to both the compute nodes and the client desktop. [↑](#footnote-ref-1)