Contents

[Windows Server AppFabric Web Farm Guide 3](#_Toc265237735)

[Installing and Configuring AppFabric Hosting and Management on a Web Farm 3](#_Toc265237736)

[Step 1: Configure Domain Security 4](#_Toc265237737)

[AppFabric Security Model 4](#_Toc265237738)

[AppFabric Domain Security 5](#_Toc265237739)

[Step 2: Set Up the Initial AppFabric Base Configuration 6](#_Toc265237740)

[Prepare and Install AppFabric Hosting and Management on the Base Configuration 7](#_Toc265237741)

[Prepare and Initialize SQL Server 7](#_Toc265237742)

[Configure AppFabric Hosting and Management on the Base Configuration 7](#_Toc265237743)

[Create and Initialize Databases Using the Windows PowerShell Cmdlets 8](#_Toc265237744)

[Create and Initialize Databases Using the AppFabric Configuration Wizard 9](#_Toc265237745)

[Step 3: Install AppFabric on the Web Farm 10](#_Toc265237746)

[Configure CAS Policy 11](#_Toc265237747)

[Install AppFabric 12](#_Toc265237748)

[Step 4: Configure AppFabric Hosting and Management on the Web Farm 12](#_Toc265237749)

[AppFabric Configuration Wizard 12](#_Toc265237750)

[Remote AppFabric Configuration 13](#_Toc265237751)

[Scripting AppFabric Configuration 14](#_Toc265237752)

[Step 5: Transfer AppFabric Applications to the Web Farm 15](#_Toc265237753)

[Application Transfer Using MSDEPLOY in IIS 15](#_Toc265237754)

[Export an Application Using IIS 16](#_Toc265237755)

[Import an Application Using IIS 16](#_Toc265237756)

[Application Transfer Using MSDEPLOY Scripting 16](#_Toc265237757)

[Changing AppFabric Configuration on a Web Farm 17](#_Toc265237758)

[Configuration Change Mechanisms 17](#_Toc265237759)

[Configuration Change Recommendations 18](#_Toc265237760)

[Remove an AppFabric Server from a Web Farm 18](#_Toc265237761)

[Leave the Web Farm 19](#_Toc265237762)

[Remove AppFabric Hosting and Management Support from Web Applications 19](#_Toc265237763)

[Windows Server AppFabric Caching 19](#_Toc265237764)

[Using Caching on an AppFabric Web Farm 21](#_Toc265237765)

[Web Farm Caching Configuration 22](#_Toc265237766)

[Application Pool Identity 22](#_Toc265237767)

[ASP.NET Session State Provider 23](#_Toc265237768)

[AppFabric Scripting Samples 24](#_Toc265237769)

[Sample: Windows PowerShell Script to Configure Throttling on Multiple Computers 24](#_Toc265237770)

[Sample: Scripted Configuration of AppFabric 24](#_Toc265237771)

[Sample: Synchronize All Computers in a Web Farm to a Model Computer 25](#_Toc265237772)

[Sample: Import and Export an Application in Windows Server AppFabric 25](#_Toc265237773)

[Summary 25](#_Toc265237774)

# Windows Server AppFabric Web Farm Guide

Windows Server AppFabric is an extension to Internet Information Services (IIS) 7.0 using the Windows Process Activation service (WAS). Windows Server AppFabric hosting and management delivers improved hosting and management of .NET Framework 4 Windows Communication Foundation (WCF) and Windows Workflow (WF) services. AppFabric hosting and management provides support for improved management, deployment, and simplified configuration of these services. Windows Server AppFabric caching offers a high-speed in-memory cache to improve scalability of .NET Framework and ASP.NET applications. For more information about the hosting, management, and caching capabilities of Windows Server AppFabric, see [Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=192511).

Within a small AppFabric installation or within a development environment, having a single AppFabric server is sufficient. In this configuration the SQL Server instance may even be on the same server as AppFabric, and users’ processing needs can be met. However, in a larger production environment, a Web farm using multiple AppFabric servers is recommended to handle increased user load. Load balancing distributes incoming requests so that the processing of incoming messages is spread across multiple servers. With a Web farm of AppFabric servers, you can implement load-balancing solutions using a hardware or software element on a common front-end dispatcher. For example, you could use the software option of Network Load Balancing Services (NLBS) that comes with Windows Server 2008 Enterprise Edition. For a technical overview of NLBS, see [Network Load Balancing Technical Overview](http://go.microsoft.com/fwlink/?LinkId=192510). For more information about step-by-step configuration of NLBS, see [Network Load Balancing Step-by-Step Guide](http://go.microsoft.com/fwlink/?LinkId=192512).

Web farms not only increase performance by reducing the load on each server, but also increase availability. If one server in the Web farm is disabled (for example, taken offline for maintenance), other servers in the Web farm take over, so users are never aware of the offline server. Servers within a Web farm share a pool of resources and minimize chances for performance bottlenecks due to resource contention. For more information about high performance, high availability, and load balancing in Windows Server 2008, see [High Availability](http://go.microsoft.com/fwlink/?LinkId=192513).

The focus of this document is to assist you in properly installing and configuring Windows Server AppFabric on an existing IIS Web farm. It assumes that you have already configured a farm of IIS Web servers, have a load balancing cluster of servers in place, and are preparing to install and configure Windows Server AppFabric hosting and management on each of the servers. Let’s start by first summarizing the process of installing and configuring AppFabric hosting and management on your Web farm.

## Installing and Configuring AppFabric Hosting and Management on a Web Farm

There is a process to ensure that your Web farm installation operates correctly and efficiently. In this document, we will go through the following steps to enable you to properly install and configure Windows Server AppFabric hosting and management services on a Web farm of servers that have IIS deployed on them:

1. Configure domain security to support the AppFabric hosting and management Web farm configuration.

2. Configure a base AppFabric hosting and management installation. This consists of one AppFabric hosting and management computer and a separate SQL Server computer that hosts the persistence and monitoring databases.

3. Install AppFabric hosting and management on all Web farm servers.

4. Configure AppFabric hosting and management on all Web farm servers.

5. Duplicate existing applications from the base AppFabric hosting and management installation on all the Web farm servers.

After AppFabric hosting and management is installed and configured correctly you may need to perform maintenance due to changes. Here are a few common examples of changes you may need to make:

 Make a configuration change (such as modifying the WCF throttling level) to one of the AppFabric servers. You will then need to replicate that change to the remaining AppFabric servers in the Web farm cluster.

 Remove one AppFabric server from the Web farm cluster. Then ensure that the Web farm and the displaced server still function properly.

 Add Windows Server AppFabric caching to an upgraded existing ASP.NET application.

We will discuss important points surrounding these types of changes to assist you in understanding from a higher level what you will need to do when changes occur on your AppFabric Web farm installation.

### Step 1: Configure Domain Security

Before we discuss the initial configuration of AppFabric on a Web farm we need to discuss a key central concept that is critical to its proper configuration – security. When using more than one AppFabric server in a Web farm, it is a best practice to shift security from the local AS\_Administrators and AS\_Observers Windows security groups created during installation on a single computer to using their domain counterparts across multiple computers. Domain security accounts and groups must be properly configured before you can successfully configure AppFabric on Web farm servers. For more information about AppFabric Windows security groups, see [Windows Security](http://go.microsoft.com/fwlink/?LinkId=192514).

#### AppFabric Security Model

The three AppFabric conceptual security roles help us understand the AppFabric security model. These roles are purely conceptual so you will not find entities with these names on any of the Web farm servers. Rather, these conceptual roles manifest themselves in correspondingly mapped physical Windows security groups and SQL Server database roles based upon access and functionality requirements. You determine the membership in these conceptual roles as you architect a Web farm security solution.

 Administrators – This role allows complete control over configuration, the use of applications and services, and access to monitoring and persistence data. The identities of the Event Collection Service (ECS) and the Workflow Management Service (WMS) are part of this conceptual group. You would typically map security identities in this category to the local AS\_Administrators Windows security group if they were not on a domain.

 Observers – This role allows viewing and enumerating of monitoring and persistence data, and the use of services and applications. You would typically map security identities in this category to the local AS\_Observers Windows security group if they were not on a domain.

 Users - This role is used by IIS at run time to assign security identities to its application pools hosting AppFabric services and applications. This gives the services contained in the applications access to persistence and monitoring databases and system services. You will typically map security identities used for the IIS application pools hosting your AppFabric services to this conceptual category.

For more information about the AppFabric security model, see [Security and Protection](http://go.microsoft.com/fwlink/?LinkId=194476).

#### AppFabric Domain Security

Now that we understand the AppFabric security model, let’s examine how to design security to support AppFabric on a Web farm. The local AS\_Administrators and AS\_Observers groups created during AppFabric setup on a single server should not be used to secure a Web farm using multiple AppFabric servers – use domain accounts instead. Be aware that the AppFabric installation and configuration programs will not create any domain accounts for you, so you must create them manually by using Active Directory. You will then specify them during the AppFabric configuration process.

For the sake of consistency within this document we will use a hypothetical and generic domain named MyDomain. Within that domain we will create AppFabric-specific and arbitrarily named domain groups of MyAppFabricDomainAdministrators, MyAppFabricDomainObservers, and MyAppFabricDomainUsers. The MyAppFabricDomainAdministrator user account is a part of the MyAppFabricDomainAdministrators group, while the MyAppFabricDomainUser user account is a part of the MyAppFabricDomainUsers group.

Here are steps to properly configure domain security before configuring Windows Server AppFabric hosting and management on multiple AppFabric server computers within that domain:

1. Create domain Windows security groups representing each of the AppFabric conceptual roles (Administrators, Observers, and Users). Grant the users assigned to these groups the appropriate privileges associated with each AppFabric conceptual role, but at the domain scope level. For more information about managing groups in Active Directory Domain Services, see [Managing Groups](http://go.microsoft.com/fwlink/?LinkId=192515).

2. After you create these domain Windows security groups, create and add domain user accounts to them based upon membership in the AppFabric conceptual roles. For example, you may create and add the MyDomain\MyAppFabricDomainAdministrator user account to the MyDomain\MyAppFabricDomainAdministrators group. For more information about managing users in Active Directory Domain Services, see [Managing Users](http://go.microsoft.com/fwlink/?LinkId=192516).

3. The service identities under which the Event Collection Service and Workflow Management Service will run on the various servers in the Web farm should be in the MyDomain\MyAppFabricDomainAdministrators group. Typically this will include the MyDomain\ MyAppFabricDomainAdministrator account. The “Log on as a service” privilege must be granted to the users in this group and enforced in the domain. This right allows a security principal to log on as a service. Any service that runs under a separate user account must be assigned the right. For information about how to add the “Log on as a service” right to an account, see [Add the Log on as a service right to an account](http://go.microsoft.com/fwlink/?LinkId=192517).

A best practice for Web farm security is to pass either a built-in account or a domain account when required to pass a Windows principal to an AppFabric cmdlet. For instance, during configuration of the initial AppFabric server in a Web farm you may run the Initialize-ASMonitoringSqlDatabase and Initialize-ASPersistenceSqlDatabase cmdlets. When doing so, do not pass in a local account but rather pass in domain accounts.

### Step 2: Set Up the Initial AppFabric Base Configuration

We will begin with an initial and typical configuration commonly found in small single-server Windows Server AppFabric hosting and management installations. An initial “base server” will function as the AppFabric hosting and management configuration model for all other servers in the Web farm. From the base server comes the IIS and AppFabric hosting and management base configuration that is duplicated on all other servers in the Web farm. In this configuration all the .NET Framework 4 WCF/WF services are managed on one AppFabric hosting and management server. The persistence and monitoring databases exist on a separate server running SQL Server. Multiple clients access the .NET Framework 4 WCF/WF services through the single AppFabric hosting and management Web server.



#### Prepare and Install AppFabric Hosting and Management on the Base Configuration

Windows Server AppFabric hosting and management needs to be installed on the single “base server.” Before installation, ensure that your system contains all the prerequisites needed for a successful setup process. This includes installing the .NET Framework version 4, critical Windows updates, and Windows PowerShell version 2.0. For more information about preparing your computer for installation, see [Preparing Your Computer for Installation](http://go.microsoft.com/fwlink/?LinkId=192549).

We will not cover all the details of the installation and configuration process in this document. Rather, we will discuss only specific areas relating to AppFabric on a Web farm. For more information about the installation and configuration processes, see [Installing Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194477) and [Installing and Configuring Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194478).

After the AppFabric hosting and management installation is complete, SQL Server must be configured properly before you can configure AppFabric hosting and management on the base server.

#### Prepare and Initialize SQL Server

Configuring SQL Server on an AppFabric Web farm requires some initial preparation to ensure that the configuration works properly. If SQL Server has not yet been initialized and configured with respect to AppFabric, you will need to do so before running the AppFabric configuration program. How you prepare SQL Server depends upon your database permissions in the SQL Server database.

Note

SQL Server should be installed on its own dedicated server computer to avoid competing for system resources with AppFabric and IIS. The computer running SQL Server may be clustered using SQL Server clustering to improve availability. For more information about SQL Server clustering, see [Getting Started with SQL Server 2008 R2 Failover Clustering](http://go.microsoft.com/fwlink/?LinkId=192550).

If you are not in the sysdamin role on the SQL Server computer, and will not be given complete administrative control over the SQL Server installation, do the following before you proceed:

 Request the database administrator (DBA) to create empty AppFabric monitoring and persistence databases. Record the name of the server and databases because you will need them later when performing AppFabric configuration.

 Ensure that the DBA adds you to the dbcreator role for each database. This allows you to initialize the database but does not give you full access as you would have in a sysadmin role.

#### Configure AppFabric Hosting and Management on the Base Configuration

Now that Windows Server AppFabric hosting and management has been installed and SQL Server has been prepared, you can configure AppFabric hosting and management. When configuring AppFabric hosting and management on a single base server, you can use either the AppFabric Windows PowerShell cmdlets or the Windows Server AppFabric Configuration Wizard. However, when installing and configuring AppFabric hosting and management on multiple servers across a Web farm, it would be very time consuming (and error prone) to have to run the configuration wizard on each individual computer. Creating script files using the AppFabric Windows PowerShell cmdlets would be a more efficient way to configure AppFabric. However, for the sake of completeness, and to show the link between the AppFabric user interface and cmdlet functionality, we will look at both methods.

##### Create and Initialize Databases Using the Windows PowerShell Cmdlets

Let’s first examine initializing the database by using AppFabric Windows PowerShell cmdlets. The two cmdlets we will use are Initialize-ASMonitoringSqlDatabase and Initialize-ASPersistenceSqlDatabase.

 Initialize-ASMonitoringSqlDatabase cmdlet. Running the Initialize-ASMonitoringSqlDatabase cmdlet will create the monitoring database if it does not already exist. After the database exists, the cmdlet will assign role memberships for the three domain accounts specified in the Admins, Readers, and Writers parameters. Here is an example of running the Initialize-ASMonitoringSqlDatabase cmdlet:

Initialize-ASMonitoringSqlDatabase

–Server “SQLServerComputer”

–Database “monitoringDB”

–Admins “MyDomain\MyAppFabricDomainAdministrators”

–Readers “MyDomain\MyAppFabricDomainObservers”

–Writers “MyDomain\MyAppFabricDomainUsers”

Running this cmdlet will create the monitoringDB monitoring database on the SQLServerComputer database server if it does not exist. It will also create the following role memberships for the domain accounts in the monitoring database.

|  |  |
| --- | --- |
| Domain group | monitoringDB role membership |
| MyDomain\MyAppFabricDomainAdministrators | ASMonitoringDBAdmin, ASMonitoringDBReader, and ASMonitoringDBWriter roles |
| MyDomain\MyAppFabricDomainObservers | ASMonitoringDBReader role |
| MyDomain\MyAppFabricDomainUsers | ASMonitoringDBWriter role |

 Initialize-ASPersistenceSqlDatabase cmdlet. Running the Initialize-ASPersistenceSqlDatabase cmdlet will create the persistence database if it does not already exist. After the database exists, the cmdlet will assign role memberships for the three domain accounts specified in the Admins, Readers, and Users parameters. Here is an example of running the Initialize-ASPersistenceSqlDatabase cmdlet:

Initialize-ASPersistenceSqlDatabase

-Server “SQLServerComputer”

-Database “persistenceDB”

-Admins “MyDomain\MyAppFabricDomainAdministrators”

-Users “MyDomain\MyAppFabricDomainUsers”

-Readers “MyDomain\MyAppFabricDomainObservers”

Running this cmdlet will create the persistenceDB persistence database on the SQLServerComputer database server if it does not exist. It will also create the following role memberships for the domain accounts in the persistence database.

|  |  |
| --- | --- |
| Domain group | persistenceDB role membership |
| MyDomain\MyAppFabricDomainAdministrators | Microsoft.ApplicationServer.DurableInstancing.WorkflowAdministrators, Microsoft.ApplicationServer.DurableInstancing.WorkflowManagementServerUsers,System.Activities.DurableInstancing.InstanceStoreObservers, andSystem.Activities.DurableInstancing.InstanceStoreUsers System.Activities.DurableInstancing.WorkflowActivationUsers roles |
| MyDomain\MyAppFabricDomainObservers | System.Activities.DurableInstancing.InstanceStoreObservers role |
| MyDomain\MyAppFabricDomainUsers | System.Activities.DurableInstancing.InstanceStoreUsers role |

In these examples, persistenceDB and monitoringDB are arbitrarily chosen names of the respective persistence and monitoring databases in SQL Server. When configuring AppFabric hosting and management on a Web farm, you will need to use these names later.

For more information about how to create and initialize a database using AppFabric cmdlets, see [Create and Initialize a Database Using Windows Server AppFabric Cmdlets](http://go.microsoft.com/fwlink/?LinkID=194479). For more information about the syntax of these cmdlets, see [Initialize-ASMonitoringSqlDatabase](http://go.microsoft.com/fwlink/?LinkId=194480) and [Initialize-ASPersistenceSqlDatabase](http://go.microsoft.com/fwlink/?LinkId=194481).

##### Create and Initialize Databases Using the AppFabric Configuration Wizard

Alternatively if you decide not to use Windows PowerShell cmdlets and scripts, you can run the Windows Server AppFabric Configuration Wizard. This provides a user interface to create and initialize the persistence and monitoring databases and to configure database security. Below is one dialog box from the wizard for configuring the monitoring database. There is a similar dialog box for the persistence database. For both databases, you specify the same database and domain group parameters as you would in the respective database initialization cmdlets described previously.

For more information about the installation and configuration processes, see [Installing Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194477) and [Installing and Configuring Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkID=194478).



### Step 3: Install AppFabric on the Web Farm

In this step, you will take an existing Web farm on a load balanced cluster and install Windows Server AppFabric hosting and management. For load balancing to work effectively on an AppFabric Web farm, the configuration on all the AppFabric servers in the cluster must appear identical to clients. Each cluster server should contain the same .NET Framework 4 WCF/WF services and inherit the same configuration settings. All servers share the same SQL Server monitoring and persistence databases. This means that all display the same configuration and data within the AppFabric user interface. Because all have the same applications, configuration, and databases, all computers will display the same monitoring metrics in their AppFabric Dashboard and related Tracked Events, Persisted WF Instances, and Tracked WF Instances pages.



At a high level, AppFabric now needs to replicate its configuration and program code from the model “base server” out to many servers on the farm. A typical installation decision is to run the AppFabric installation program from a common network share. If you will be doing this, however, be aware of the configured Code Access Security (CAS) policy.

#### Configure CAS Policy

By default, CAS is disabled, so this may not be an issue on your network. But if CAS settings are enabled, and if they prohibit the execution of code from intranet shares, AppFabric setup will fail. The reason is that the installation will not be able to execute any code, including the code that displays error messages. For that reason, platform validation logic cannot be used to enforce the proper CAS settings.

The solution for this issue is to configure CAS policy so the installation program executes in full trust from intranet shares, as shown in the following CAS examples:

 The following command adds the Windows Server AppFabric installation assembly to the full trust list for the computer policy:

caspol -computer -addfulltrust WindowsServerAppFabricSetup.exe

 The following command adds a child code group that gives the share \\netserver\netshare local intranet permissions:

caspol -computer -addgroup 1. -url \\netserver\netshare\\* LocalIntranet

If you need more information about configuring CAS, see [.NET Framework Configuration Tool (MSCORCFG.MSC)](http://go.microsoft.com/fwlink/?LinkId=181958) and [Code Access Security Policy Tool (CASPOL.EXE)](http://go.microsoft.com/fwlink/?LinkId=181960).

#### Install AppFabric

After the CAS policy has been configured correctly you can proceed to install Windows Server AppFabric hosting and management on each of the new AppFabric servers individually. You may have valid reasons to run the AppFabric installation program through its user interface on each individual server. However, most likely on a Web farm you will want to run an automated installation from a command line. This ensures that all servers on the Web farm have the same configuration after installation. With an automated installation, an administrator does not have to provide manual input during the installation process. However, if you run an automated installation on a Web farm, control will return to the command line while the installation is still running. To ensure that the installation does not return control until it has completed, invoke an automated installation from an Administrative command prompt by using the start /w command of cmd.exe. For more information about automated installation of AppFabric, see [Automated Installation](http://go.microsoft.com/fwlink/?LinkId=194482).

### Step 4: Configure AppFabric Hosting and Management on the Web Farm

After the Windows Server AppFabric hosting and management installation is complete the next step is to configure the AppFabric hosting and management servers. This can be done by using the Windows Server AppFabric Configuration Wizard, through Windows PowerShell scripting, or by remote administration.

#### AppFabric Configuration Wizard

You can invoke the Windows Server AppFabric Configuration Wizard by selecting Configure AppFabric from the Windows Server AppFabric group from the Start menu.

To set monitoring configuration using IIS

|  |
| --- |
| 1. Set the AppFabric Event Collection Service Account to MyDomain\MyDomainAdministrator. A message shows that the Log on as a service permission has been granted.2. Click Configure for Monitoring Provider.  Select Register AppFabric monitoring store in root Web.config. Do not select Initialize Monitoring Store because the monitoring database already exists in an initialized state. This results in all the entries in the Security Configuration section being disabled because domain accounts have already been specified for the Administrators, Readers, and Writer roles.3. For Connection String, specify the server and monitoring database, and then click OK. |

To set persistence configuration using IIS

|  |
| --- |
| 1. Set the AppFabric Workflow Management Service Account to MyDomain\MyDomainAdministrator. A message shows that the Log on as a service permission has been granted.2. Click Configure for Persistence Provider.  Select Register AppFabric persistence store in root Web.config. Do not select Initialize Persistence Store because the persistence database already exists in an initialized state. This results in all the entries in the Security Configuration section being disabled because domain accounts have already been specified for the Admins, Readers, and Writers roles.3. For Connection String, specify the server and persistence database, and then click OK. |

For more information about how to configure AppFabric using the Windows Server AppFabric Configuration Wizard, see [Configure Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194483).

Be aware that the AppFabric Configuration Wizard may fail on a domain controller if you use non-domain accounts. Per-service, non-domain accounts for the Event Collection Service and Workflow Management Service cannot be added to the domain administrators group or the observers groups on a domain controller. A Windows Service Security ID (SID) is longer than those supported by Active Directory. When you attempt to initialize a store using the SID from a non-domain account, this can result in an error that a Windows security principal is invalid. To resolve this issue, configure AppFabric using domain accounts, and associate the Active Directory Service role with the Application Server role as follows:

1. Run the Event Collection Service and Workflow Management Service under domain accounts, rather than per-service accounts.

2. Add the domain accounts used for the ECS and WMS services to the domain administrators group you created (for instance, the MyDomain\MyAppFabricDomainAdministrator account in the MyDomain\MyAppFabricDomainAdministrators group). This account will be used to connect to the associated monitoring or persistence database.

3. Add the Event Collection Service account to the Performance Log Users group manually. This enables the Event Collection Service to read and write to the Event Tracing for Windows (ETW) session.

#### Remote AppFabric Configuration

You can remotely configure and administer another IIS (AppFabric) server on a Web farm by using Internet Information Services (IIS) Manager as follows:

1. Click the Start Page in the Connections pane on the originating server.

2. Select Connect to a server, enter Server name, and then provide credentials to connect.

3. In the center pane, under the Management section, click the Management Service icon to bring up the correspondingly named dialog box.

4. Select Enable Remote Connections.

5. Ensure that the WMSVC (IIS Web Management Service) is started on both servers.

While connected remotely, you can configure the server as a remote administrator by selecting Configure within Manage WCF and WF Services. However, you do not have access to the Deploy option for importing or exporting a deployment package. For more information about remotely administering Windows Server AppFabric, see the “Remote Administration” section of the [Securing Administration](http://go.microsoft.com/fwlink/?LinkId=194484) topic, and [Remote Administration for IIS Manager](http://go.microsoft.com/fwlink/?LinkId=194485).

#### Scripting AppFabric Configuration

Manual configuration operations done through IIS are time consuming and prone to human error. So typically as a Web farm administrator you will rely on scripting to manage your Windows Server AppFabric Web farm configuration. Scripts expedite your duties and create consistency in the results of change operations across large groups of AppFabric Web farm servers. Using scripts will also save you time in diagnosing problems should they arise.

You can create a Windows PowerShell configuration script so that at the end of its processing AppFabric hosting and management is configured properly on the added Web farm server. This script can contain core Windows PowerShell commands as well as calls to cmdlets specific to AppFabric. AppFabric cmdlets enable you to accomplish almost any task that you can do through the IIS AppFabric UI by using Windows PowerShell cmdlets instead. Cmdlets can be combined effectively with core Windows PowerShell commands and other utilities (such as the IIS APPCMD.EXE utility) to script the configuration of AppFabric hosting and management on a Web farm. Your particular configuration scripts will contain commands and parameters specific to your Web farm installation. However, configuration scripts at this point in the process most likely will minimally include the following functionality:

 Manage security and assign group membership to the local Administrators group using ADSI (Active Directory Services Interface): [Active Directory Service Interfaces](http://go.microsoft.com/fwlink/?LinkId=194486)).

 Add or update connection string settings in the specified .NET Framework configuration file (using the IIS appcmd utility) for monitoring and persistence.

 Configure service identities for ECS and WMS (using SetServiceCredential).

 Create default behaviors and configure persistence and monitoring settings (using Set-ASAppMonitoring and Set-ASAppSqlServicePersistence).

For a script example of configuring AppFabric hosting and management on a single server that does these operations, see [Scripted Configuration of AppFabric](http://go.microsoft.com/fwlink/?LinkId=194487). This serves as a useful starting point for writing customized installation scripts particular to your Web farm and domain. You can call your modified version of this script multiple times to configure each AppFabric hosting and management server on the Web farm. For information about the IIS configuration and query utility, see [Appcmd.exe](http://go.microsoft.com/fwlink/?LinkId=192553).

Here are some general recommendations to increase the odds that your configuration scripts will function correctly across multiple AppFabric Web farm servers:

 Configure Windows PowerShell to remotely execute cmdlets on remote computers using Windows Remote Management (WinRM) 2.0. This requires the Windows Remote Management services (WinRM service) to listen to WSMAN requests over HTTPS from the Windows PowerShell console at the remote location. For more information about Windows Remoting, see [About Windows Remote Management](http://go.microsoft.com/fwlink/?LinkId=182112).

 The remote Windows PowerShell case requires Windows PowerShell V2 RTM to be installed and running on both client and remote server computers. For more information, see [Setting Up Windows PowerShell Remoting](http://go.microsoft.com/fwlink/?LinkId=192554).

 All computers that will be modified during the synchronization must allow the World Wide Web Services (HTTP) through their firewall. Ensure that your database is prepared for remote access. If you are using SQL Server, see [Configuring the Windows Firewall to Allow SQL Server Access](http://go.microsoft.com/fwlink/?linkid=94001).

 All computers in the Web farm must be part of a domain, and the account that you use to synchronize the farm must have administrative privileges on all computers.

 If you are using the Web Deployment Tool remotely, its Remote Agent Service must be installed and running on all computers. By default AppFabric installs the Web Deployment Tool, but its Remote Agent Service is not part of the default installation. Install the Remote Agent Service by using Programs and Features in Control Panel. To learn more about the installation and configuration of the Web Deployment Tool, see [Installing the Web Deployment Tool](http://go.microsoft.com/fwlink/?LinkId=194489).

Here are some links to give you additional information about Windows PowerShell scripting:

 [Using Windows PowerShell Cmdlets in Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194490)

 [Windows PowerShell for Windows Server AppFabric Reference](http://go.microsoft.com/fwlink/?LinkId=194491)

 [Windows PowerShell](http://go.microsoft.com/fwlink/?LinkId=194492)

 [AppFabric Hosting PowerShell Cmdlets](http://go.microsoft.com/fwlink/?LinkId=194493)

 [AppFabric Caching PowerShell Cmdlets](http://go.microsoft.com/fwlink/?LinkId=194494)

### Step 5: Transfer AppFabric Applications to the Web Farm

Installing and configuring the Windows Server AppFabric hosting and management system files is only part of the process of setting up the AppFabric Web farm. To properly utilize load balancing, you will want to identically duplicate the Web applications and services configured in IIS on the initial AppFabric hosting and management server. These must transfer across all other AppFabric servers you are replicating as a part of the Web farm cluster. While this task may sound daunting, it is actually very straightforward through AppFabric.

#### Application Transfer Using MSDEPLOY in IIS

Windows Server AppFabric leverages the existing IIS capabilities for importing or exporting applications by using the Web Deployment Tool (Web Deploy). This technology moves deployment entities in and out of the Web server through IIS. The command-line version of this tool is msdeploy.exe, and also allows seamless deployment of applications from IIS Manager using Windows PowerShell scripts. The msdeploy.exe tool allows you to control with fine-grained specificity what the deployment operations entail, as well as to execute troubleshooting operations when deployment does not go as planned. For more information about how to use the Web Deployment Tool, see [Web Deployment Tool](http://go.microsoft.com/fwlink/?LinkId=151481) and [Web Deploy](http://go.microsoft.com/fwlink/?LinkId=192556).

Deployment entities are imported and exported in AppFabric by using actions in IIS Manager that are built on Web Deploy. You can import or export entities for an individual application, all the applications under a Web site, or all the Web sites under one server. For more information about how to export and import applications, see [Import and Export an Application in Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194495). For more information about using the Web Deployment Tool to synchronize IIS 7.0 Web sites between source and destination computers, see [Synchronize IIS 7](http://go.microsoft.com/fwlink/?LinkId=192557).

##### Export an Application Using IIS

Because we are building an AppFabric Web farm, we will want to choose which application packages to export to the other servers. Exporting a package once for an application allows any of the other Web farm servers to host the same service configured identically by importing that package.

On the base server, in the Connections pane within IIS Manager, select a scope level from which you want to delineate the scope of export. Select the appropriate Export Server Package/Export Application command. As you go through the wizard select whatever you want to export within the final output package. This creates an application package (.zip file) containing configuration data, including registry settings, Web content, and SQL Server database information and scripts. All of these are necessary to successfully import this package onto another AppFabric server and re-create the configuration necessary for it to work correctly.

##### Import an Application Using IIS

While the export process needs be run only once, you will execute the import process once for each AppFabric server on the Web farm. To maintain consistency across servers, ensure that when you begin the import process on the new server you select the same scope in the Connections pane in IIS Manager that you exported the package from on the base server. Click Import Server or Site Package to begin the import process. After this process completes, the new server will be configured the same as the base server with respect to the IIS/AppFabric applications and settings.

After the import is complete, change the account identity of the application pools that will be used to run AppFabric managed applications on the new server. Configure this value to use the account MyDomain\MyAppFabricDomainUser. This account is a member of the MyDomain\MyAppFabricDomainUsers group, which has access to the persistence and monitoring SQL Server databases. This access comes through membership in specific roles by its parent domain group, MyDomain\MyAppFabricDomainUsers.

#### Application Transfer Using MSDEPLOY Scripting

You can script msdeploy.exe to transfer applications from the base server to other servers on the Web farm. For information about the syntax of the Web Deployment Tool (msdeploy.exe), see [Web Deployment Tool](http://go.microsoft.com/fwlink/?LinkId=192558).

With a minor modification to their syntax, Web Deploy commands can be run from a Windows PowerShell prompt. To do this, change the colon character (:) after the verb, source, and dest arguments of the Web Deploy command to an equal sign (=). In the following example, compare the Web Deploy command with its Windows PowerShell version. Here is a Web Deploy command:

MSDEPLOY -verb:sync -source:metakey=/lm/w3svc/1 -dest:metakey=/lm/w3svc/2 -verbose

Here is the same Web Deploy command as a Windows PowerShell operation:

.\MSDEPLOY.exe -verb=sync -source=metakey=/lm/w3svc/1 -dest=metakey=/lm/w3svc/2 –verbose

Note that although we can use Windows PowerShell at this point in the process, it is not required. We will really see the power of AppFabric Windows PowerShell commands in the next section on configuring a Web farm.

## Changing AppFabric Configuration on a Web Farm

After you configure the initial Windows Server AppFabric hosting and management Web farm, it’s only a matter of time before you will need to make changes to the configuration on all of the AppFabric hosting and management servers. Here are some examples of changes that could occur:

 At the service level, an administrator may need to change a WCF throttling setting. Changes made at the service level result in the individual application’s Web.config file being changed.

 At the server level, a connection string used by all applications on that server may need to be updated to point to another monitoring database server. Changes made at the computer level result in the root Web.config file being modified.

For more information about configuration change management, see [Configuration Process in Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=194496).

### Configuration Change Mechanisms

Applying multiple changes to a Web farm requires that all changes be applied to all computers in exactly the same way. If all the changes that are needed are scripted, this is an easy task. Otherwise it presents the problem of how to record and reproduce all the changes on all computers in the Web farm. Another solution is to apply all the changes to a single computer and then replicate the configuration across the Web farm. Here are some choices to ensure that the configuration changes from one server are mirrored on the others in the Web farm:

 Develop a Windows Powershell script containing the names of all the computers in the cluster. Within the code you can loop and invoke the appropriate Windows PowerShell cmdlet on each computer in the Web farm to change this setting through script code.

 Use the XCOPY utility to copy the modified Web.config file from one base server to each of the servers in the Web farm.

 Use msdeploy.exe to export the changes from the appropriate scope level on the base server. This content is created during the export process within an installation package (.zip file). This is similar to what you do when you are first adding another AppFabric hosting and management cluster computer to the Web farm. You can then import the installation package from each server in the cluster to get updated configuration settings. For more information about using the Web Deployment Tool to synchronize IIS 7.0 Web sites between source and destination computers, see [Synchronize IIS 7](http://go.microsoft.com/fwlink/?LinkId=192557). For a sample showing how to use msdeploy.exe to synchronize all computers in a Web farm to a model computer, see [Synchronize All Computers in a Web Farm to a Model Computer](http://go.microsoft.com/fwlink/?LinkId=192559).

Note

In this document we do not discuss using IIS shared configuration for configuration change management. That paradigm uses a single configuration file on a shared common file store holding the configuration information for all the computers in one central location. When global changes are made that affect multiple sites and applications, these changes will cause all the app domains to recycle at one time. Often this is an undesirable side effect that temporarily removes the entire Web farm cluster from servicing incoming requests.

### Configuration Change Recommendations

When you are performing configuration updates to AppFabric hosting and management servers in a Web farm we recommend that you update only a percentage of the computers at any one time. Temporarily remove a portion of the servers from the load-balanced AppFabric hosting and management cluster, update them, and then put them back into the cluster so the other servers can be updated.

A consistent way of making changes is to run the same script over multiple servers. In the case of deployment, for example, this requires putting all the msdeploy.exe commands in one script file (say scriptmsdeploy.cmd), and then calling that script file from another script file (scriptDriver.cmd). The second file iterates a number of times through a loop to invoke scriptsmsdeploy.cmd once for each server on the Web farm. To determine the looping value you can hardcode the name of each server into a third file (appFabricServers.txt), and then read each server name from that static file from within scriptDriver.cmd as follows:

for /f %%i in ('type appFabricServers.txt’) do

(start scriptDriver.cmd %%i %1)

A slightly more complex way to make consistent changes is to dynamically obtain a list of AppFabric servers on the Web farm by querying SQL Server. When managing a large farm of many AppFabric servers, hardcoding the names of the servers in a text file may not be the best solution. You can extract a list of AppFabric servers on a Web farm directly from the application monitoring data and use that information to run Windows PowerShell commands on those servers. For an example of how to extract the Web farm computers and run Windows PowerShell cmdlets on all the active computers on a Web farm, see [Windows PowerShell Script to Configure Throttling on Multiple Computers](http://go.microsoft.com/fwlink/?LinkId=192560).

## Remove an AppFabric Server from a Web Farm

You can remove an AppFabric server from a Web farm physically. After a Web server is removed from the Web farm cluster, you can also uninstall AppFabric hosting and management. You will want to ensure that Web applications that previously used AppFabric hosting and management features continue to work properly. There are two different ways to remove AppFabric hosting and management functionality from clients, so let’s look at each individually.

### Leave the Web Farm

From a client visibility standpoint, to physically remove an AppFabric server from a Web farm is to remove it logically from the Web farm cluster. At that point the server is no longer a part of the load-balancing scheme of the cluster. However it is still directly accessible to clients that are aware of its actual IP address. If this scenario is needed, clients just need to be made aware to use the actual IP address of the Web server and not the virtual IP address of the Web farm cluster.

### Remove AppFabric Hosting and Management Support from Web Applications

After removing an AppFabric hosting and management server physically from a Web farm, you may still want to keep it as a functional Web server on your domain, but remove all traces of AppFabric from the server itself. You can uninstall AppFabric through the AppFabric setup wizard or from Application Server Extensions for .NET 4 in Control Panel.

Realize that a Web server, site, or application initially configured to use AppFabric-specific features can become non-functional after you uninstall Windows Server AppFabric hosting and management. You may need to clean your application configuration after AppFabric hosting and management is uninstalled. This can involve locating and cleaning the applicationHost.config file and the Web.config files for each scope. You can use the AppFabric Uninstall Cleanup Utility that is available on the AppFabric download center for an automated solution. For more information about properly uninstalling AppFabric from a server, see [Remove Features and Uninstall](http://go.microsoft.com/fwlink/?LinkId=194497), [Clean Application Configuration After Uninstalling AppFabric](http://go.microsoft.com/fwlink/?LinkId=185913), and [Windows Server AppFabric Add or Remove Features Wizard UI](http://go.microsoft.com/fwlink/?LinkId=194498).

## Windows Server AppFabric Caching

If you decide to use Windows Server caching within your AppFabric architecture on a Web farm, you will need to take certain steps to ensure that it works correctly in that environment.

The features of AppFabric caching provide a high-performance distributed cache for .NET Framework 4 WCF/WF services. For ASP.NET applications that store information in session, AppFabric caching can be used by installing the AppFabric caching service and modifying the appropriate configuration files (to be discussed shortly). Alternatively, any .NET Framework 4 WCF/WF service can use the AppFabric caching APIs to store and retrieve items from a cache. This includes .NET Framework 4 WCF/WF services that are configured to use AppFabric hosting and management.

One or more servers configured to use AppFabric caching make up a cache cluster. A cluster can be as simple as one server, named a cache host, running the AppFabric caching Windows service. But it typically consists of many cache hosts that work together to form a single cache cluster. While a physical named cache can be spread across servers, a client application is presented with a unified cache system. All operations on the cache are virtualized to a single point of reference through the cache cluster. While the content of the cache is volatile and stored in memory on the cache host servers, after a cache is created its configured definition is persisted in a SQL Server database, an XML file, or a custom configuration store and persists across server restart operations. In the following diagram the named cache “Inventory” spans the memory of three computers in the cache cluster, but is abstracted as a single point of reference to the consumer of the cache.



Within a Web farm of AppFabric hosting and management servers, we recommend that any servers running the AppFabric caching service exist separately outside of a load-balanced cluster of AppFabric hosting and management servers. If a named cache exists, and Web and client applications have been granted access to the cache cluster, they can use it. An advantage of using a named cache from an AppFabric Web farm is that all .NET Framework 4 WCF/WF services that access a named cache entity access a global value whose single value is accessible to all clients of that named cache.

The following diagram shows a load-balanced AppFabric Web farm cluster that uses the AppFabric caching server to store ASP.NET session state. This caching cluster is a separate logical cluster from the AppFabric hosting and management Web farm cluster. In the diagram, all computers in the AppFabric caching server cluster access either the SQL Server configuration store, or the Windows Server XML cache configuration store – but not both. (Not shown here is the fact that it is also possible to use a custom provider to maintain the configuration store in a custom location.) These persistent storage mechanisms are used only for configuration management. The actual caches and associated data are stored in memory on the AppFabric caching servers across the cache cluster.



It is not a requirement for cache client applications taking advantage of AppFabric caching to be configured to use Windows Server AppFabric hosting and management. For instance, an ASP.NET Web application can use the AppFabric caching session state provider to store session state through the Web.config file. But cache clients can also use AppFabric hosting and management if it makes sense. For instance, a cache client could be a .NET Framework 4 WCF/WF service hosted and managed by AppFabric on the Web farm that uses the cache client APIs to access a cache on the cache cluster. Web farm administrators need to understand their role in each of these situations for the cache to work correctly.

For more information about Windows Server AppFabric caching, see [Windows Server AppFabric Caching Features](http://go.microsoft.com/fwlink/?LinkId=194499).

## Using Caching on an AppFabric Web Farm

Let’s first look at the option of using Windows Server AppFabric caching to develop an application incorporating the caching APIs in client code. For example, say a WCF banking service is configured to use AppFabric hosting and management on a Web farm. That same service could achieve performance benefits by caching some of its operational data on an AppFabric cache cluster. The cache settings can be set programmatically or set in the configuration file. In both cases, the administrator needs to know the name of the cache to perform the following steps before the cache client can access the cache:

1. In Windows PowerShell, use the Use-CacheCluster cmdlet to set the context to the target cache cluster. Because this action writes to a central cache configuration store (XML file or SQL database, for example) the change only needs to be made on one cache server in the cluster to locate a cache server at run time.

2. Create any necessary named caches with the New-Cache command. In this case, the administrator must obtain the name of the cache used in the code from the person developing the client using the AppFabric caching APIs.

3. Grant access to the cache client's Windows account with the Grant-CacheAllowedClientAccount command.

4. Start the cache cluster with the Start-CacheCluster command.

For information, see [Preparing the Cache Client Development Environment](http://go.microsoft.com/fwlink/?LinkId=194500).

Another option is for an ASP.NET Web application hosted in a Web farm to use the AppFabric caching session state provider to store session state for its pages. Here none of the caching APIs are used directly, and all access to the cache is managed through entries in the Web.config file. Here are the tasks that an administrator carries out to add AppFabric caching to an application:

1. Create a named cache for use by the application by using the New-Cache command. As noted earlier, because this action writes to a central cache configuration store (XML file or SQL database, for example) the change only needs to be made on one cache server in the cluster to locate a cache server at run time. An administrator must obtain the cache name from the ASP.NET pages, possibly a configuration file, or the developer of the application.

2. Ensure that the Web.config file for the ASP.NET application points to the named cache, and is using the AppFabric caching session state provider. It’s possible that the developer of the ASP.NET application has already done this.

3. Ensure that the identity of the application pool hosting the ASP.NET application has access to the cached cluster. Run the Grant-CacheAllowedClientAccount cmdlet to accomplish this.

For more information about caching configuration file entries, see the following section on “ASP.NET Session State Provider,” and the topic “Configuring an ASP.NET Session State Provider” at [Configuring an ASP.NET Session State Provider (Windows Server AppFabric Caching)](http://go.microsoft.com/fwlink/?LinkId=194501).

### Web Farm Caching Configuration

Now that we have briefly summarized how to configure the cache, let’s see how it specifically applies to an AppFabric hosting and management Web farm.

#### Application Pool Identity

When using caching from an AppFabric server on a Web farm it is critical that proper access is granted. If an IIS application running a .NET Framework 4 WCF/WF service is configured to use AppFabric hosting and management and is using the cache cluster, the identity of its application pool must have access to the cache cluster. It does not matter whether the application pool is hosting an ASP.NET application and is not using AppFabric hosting and management, or hosting a .NET Framework 4 WCF/WF service configured to use AppFabric hosting and management. Both of these run in an IIS application pool, so permission for the identity of that pool must be given to access the cache cluster.

If AppFabric applications are installed to run using the DefaultAppPool, then by default they are configured to run using the NETWORKSERVICE account. This built-in account has minimum privileges on the local computer and acts as the computer on the network using the Domain\computer$ account.

Here are the steps to take to ensure access to the cache server from all IIS/AppFabric hosting and management servers on a Web farm:

1. On one of the cache cluster computers, click Start, click Windows Server AppFabric, and then select Caching Administration Windows PowerShell. Accessing Windows PowerShell this way (instead of through Administrative Tools) automatically runs the Use-CacheCluster cmdlet so that all subsequent commands apply to that cluster. You do not need to run this as Administrator unless you intend to stop or start the AppFabric caching service through cmdlets.

2. Run the Get-CacheAllowedClientAccounts cmdlet to display the accounts that can connect as clients to the cache cluster. Note that any computer accounts are in the form of DOMAIN\COMPUTERNAME$. The appended “$” means the account identity of the computer and is not the same as the computer name.

3. The identity of an application pool accessing the cache cluster must be granted access by using Grant-CacheAllowedClientAccount. If using the NETWORKSERVICE account, you will need to add the computer account of the IIS Web farm server to the list of allowed clients to access the cache cluster. If you are running an ASP.NET application or a .NET Framework 4 WCF/WF service managed by AppFabric in an application pool other than the DefaultAppPool, that identity will need to be granted cache access as a client.

#### ASP.NET Session State Provider

This section applies when using the AppFabric caching session state provider from an ASP.NET page. As a Web farm administrator, you will need to ensure the configuration files on each Web server for applications using AppFabric caching properly point to all the cache cluster servers. This is done by ensuring that there is a <host> entry for each cache host server in the <dataCacheClient><hosts> section in the Web.config file. Caching changes to a configuration file do not require recompilation of any .NET Framework code.

In the following configuration the administrator has added entries for the three servers in the AppFabric caching cluster. This also increases the availability of the caching servers.

<dataCacheClient>

 <hosts>

 <host name="CacheServer1" cachePort="22233"/>

 <host name="CacheServer2" cachePort="22233"/>

 <host name="CacheServer3" cachePort="22233"/>

 </hosts>

</dataCacheClient>

The named cache (“NamedCache1” here) must be specified in the Web.config file for the AppFabric Cache session state provider.

<sessionState mode="Custom" customProvider="AppFabricCacheSessionStoreProvider">

 <providers>

 <!-- specify the named cache for session data -->

 <add

 name="AppFabricCacheSessionStoreProvider"

 type="Microsoft.ApplicationServer.Caching.DataCacheSessionStoreProvider"

 cacheName="NamedCache1"

 />

 </providers>

</sessionState>

For more information about caching configuration file entries, see [Configuring an ASP.NET Session State Provider (Windows Server AppFabric Caching)](http://go.microsoft.com/fwlink/?LinkId=194501).

## AppFabric Scripting Samples

The Windows Server AppFabric documentation contains some samples of particular interest to administrators managing a Web farm of AppFabric servers. We will summarize the purpose of each sample and provide links to the actual documentation. We highly recommend that you use the provided links to download the samples and work through the script files to truly understand the concepts. The samples package for AppFabric can be downloaded from [Windows Server AppFabric Samples](http://go.microsoft.com/fwlink/?LinkId=169336). Although these samples will work with any application, we recommend that you use the Common Windows Server AppFabric Sample Application, which was created for use with AppFabric samples.

### Sample: Windows PowerShell Script to Configure Throttling on Multiple Computers

This sample demonstrates the required steps to use a Windows PowerShell script to configure service throttling defaults for multiple computers that are part of a farm. This sample can easily be modified to change any other AppFabric-accessible setting. It also shows you how to extract farm computer information and how to run Windows PowerShell cmdlets on all the active computers in the farm. For the complete documentation of this sample, see [Windows PowerShell Script to Configure Throttling on Multiple Computers](http://go.microsoft.com/fwlink/?LinkId=192560).

### Sample: Scripted Configuration of AppFabric

This sample walks you through the series of actions necessary to configure Windows Server AppFabric to use domain accounts in conjunction with integrated security authentication to connect to a remote database. For the complete documentation of this sample, see [Scripted Configuration of AppFabric](http://go.microsoft.com/fwlink/?LinkId=192561).

### Sample: Synchronize All Computers in a Web Farm to a Model Computer

This sample demonstrates how to synchronize both the application content and the configuration on all computers in a Web farm running Internet Information Services (IIS) to a single model computer. It can be easily modified to synchronize only parts of the IIS content and configuration, for example, to synchronize only a single application. It also shows how to use the Web Deployment Tool (Web Deploy) to replicate the content and configuration of a single server to all the computers in the Web farm. For the complete documentation of this sample, see [Synchronize All Computers in a Web Farm to a Model Computer](http://go.microsoft.com/fwlink/?LinkId=192562).

### Sample: Import and Export an Application in Windows Server AppFabric

This sample demonstrates using import and export of AppFabric applications. AppFabric leverages the existing IIS capabilities for importing or exporting applications by using msdeploy.exe (also called the Web Deployment Tool, or Web Deploy) technology for moving entities in and out of the Web server. The msdeploy.exe tool allows seamless deployment of applications from IIS Manager, Visual Studio 2010, and by using Windows PowerShell scripts. For the complete documentation of this sample, see [Import and Export an Application in Windows Server AppFabric](http://go.microsoft.com/fwlink/?LinkId=192563).

## Summary

Because Windows Server AppFabric is based on IIS, it maps well into the concept of a Web farm. Conceptually AppFabric security must be configured properly at a domain level before configuring AppFabric on a Web farm. The Web farm can be configured using either software or hardware to balance the load using the different servers in the cluster. Permission to the persistence and monitoring databases in SQL Server involves Windows domain accounts and groups that map to SQL Server role membership.

The process of duplicating an AppFabric installation on all servers in a Web farm includes exporting and importing application packages through the use of msdeploy.exe. When changes are made to one AppFabric server they must be duplicated to all other servers in the Web farm cluster. This can be done by using Windows PowerShell scripts, XCOPY, or export/import using msdeploy.exe. Uniform configuration ensures that the client requests are always handled the same way regardless of which computer in the load-balanced cluster receives the actual request for work. When removing a Windows Server AppFabric server from a client’s visibility, care must be taken for existing Web applications previously configured to use Windows Server AppFabric hosting and management to ensure they can still function properly.

In addition to Windows Server AppFabric hosting and management, a high-performance caching solution exists with AppFabric caching. This can be used by ASP.NET applications to more effectively store session state. Its APIs can also be used by directly from .NET Framework 4 WCF/WF service code. On a Web farm, there are certain configuration and security tasks an administrator must accomplish to ensure that the caching solution works properly.

Administrators develop and maintain administrative script files to ensure their duties are done efficiently and consistently. With the AppFabric Windows PowerShell cmdlets almost everything you can do in the AppFabric user interface can be done by using AppFabric hosting and management and Caching cmdlets. There are also some tasks, such as initializing a monitoring or persistence SQL Server database, that can be done only through AppFabric cmdlets. We discussed some general guidelines, as well as four specific samples you can modify to quickly get your AppFabric server configuration synchronized with other servers in the Web farm.