



Building Windows Phone 7 applications with SharePoint 2010 Products and Unified Access Gateway

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

Microsoft, SharePoint , Silverlight, Visual Studio, and Windows Phone 7 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

© 2011 Microsoft Corporation. All rights reserved.

Building Windows Phone 7 applications with SharePoint 2010 Products and Unified Access Gateway

Dave Pae
Microsoft Corporation

Todd Baginski
Aptillon, Inc.

Matthew McDermott
Aptillon, Inc.

Ben Ari
Microsoft Corporation

March 2011

Applies to: Microsoft® SharePoint® Server 2010, Microsoft SharePoint Foundation 2010, Microsoft Forefront Unified Access Gateway, Windows Phone 7™

Summary:

This white paper addresses business scenarios for the development of mobile applications that use the features of SharePoint 2010 Products for collaboration while authenticating through Microsoft Forefront Unified Access Gateway (UAG). The main body of the paper introduces the concepts and code required to access SharePoint list data in a secure manner from Windows Phone 7. The Appendix details the installation and configuration of a developer instance of UAG for the purposes of testing and developing mobile applications with SharePoint 2010 Products.

Contents

Contents	4
Overview and Goals.....	6
Architecture.....	7
SharePoint 2010 Products Intranet.....	7
UAG	7
Intranet Collaboration Scenario	7
Business Value of Collaboration	8
The Application	8
Security	12
Credential Storage	12
Authentication	12
SharePoint Services.....	13
Consuming Services	13
Accessing SharePoint List Data by Using the OData Client Library	13
Activity Feed RSS.....	21
User Profile Service - Colleagues.....	24
User Profile Service - User Profile Data	27
Testing	30
Testing the Application on the Emulator.....	30
Testing the Application on a Device	30
Marketplace Considerations	30
Conclusion.....	34
Appendix – Installation and Configuration of UAG for SharePoint 2010 Products.....	35
1. Create a SharePoint Server Virtual Machine and a UAG Virtual Machine.....	35
2. Set Up Hyper-V Host Machine Virtual Networks	36
3. Set Up Hyper-V Virtual Machine Networks	37
4. Prepare the UAG Server Virtual Machine for UAG Installation	41
5. Snapshot the UAG Server Virtual Machine	42
6. Install the UAG Server	42
7. Snapshot the UAG Server Virtual Machine	48
8. Initial UAG Server Configuration and Activation.....	48
9. Create the HTTP Trunk to Publish the SharePoint Site	64
10. Create the SharePoint Application.....	74
11. Configure the SharePoint Application	84
12. Activate the Configuration	86
13. Configure and Verify SharePoint Alternate Access Mappings.....	88

14. Add Hosts File Entries to the Development Environment.....	88
15. Test the Configuration.....	89
16. Test the Newsfeed RSS	94
Resources	97
Forefront Unified Access Gateway on TechNet.....	97
Closer to the Edge Blog	97
Silverlight Web Services Team Blog.....	97
About the Authors	97
Todd Baginski, MVP	97
Matthew McDermott, MVP	97
Ben Ari	98

Overview and Goals

As organizations grow and change and seek to keep pace with technology, their employees may feel disconnected from the corporate community. Microsoft® SharePoint 2010 Products support many features that enable employee engagement and collaboration — from lists and libraries that store and manage documents and information, to centralized user profiles that enable employees to describe themselves and the role they play in the organization, to tagging and notes that facilitate the discovery of information inside and outside the firewall. Windows Phone 7™ is the latest version of mobile devices that supports Microsoft's vision of a mobile workplace. The Windows Phone is delivered with many features that support "Office in the cloud," including applications that can open and read Microsoft Office documents. The Windows Phone 7 development experience is tailored for the .NET Developer who already has a firm grasp of .NET and Microsoft Silverlight® development. This white paper seeks to bridge the gap between the Windows Phone developer and the SharePoint developer who want to create business applications that can leverage the power of SharePoint 2010 Products from the Windows Phone.

Note In this white paper, *SharePoint 2010 Products* refers to Microsoft SharePoint® Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise specified. The examples and scenarios use SharePoint Server 2010.

This paper seeks to clarify the following development scenarios:

- How do I prepare for connecting to SharePoint 2010 Products from my Windows Phone 7 applications?
- How do I connect to SharePoint through the Unified Access Gateway?
- How do I connect to, authenticate, and consume SharePoint Web services?
- How do I create and update SharePoint list items?
- How do I consume RSS feeds provided by SharePoint?

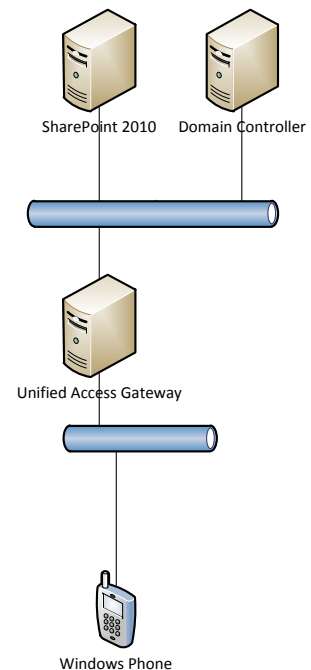
Architecture

In the scenario illustrated in this white paper, Contoso enables their employees to connect to SharePoint 2010 Products through Forefront Unified Access Gateway (UAG). UAG translates the external host name configured on the phone to the internal host name configured in SharePoint and adds a layer of security and authentication that organizations require to protect and control their corporate Web properties.

SharePoint 2010 Products Intranet

The Contoso intranet is an enterprise implementation of SharePoint 2010 Products. The site is configured by using a User Profile Service Application and Search Service Application. The mobile application will use the Web service from the User Profile Service to enable employees to view User Profile information on their mobile phones.

For more information about configuring the User Profile Service Application, see [User Profile service administration \(SharePoint Server 2010\)](http://technet.microsoft.com/en-us/library/ee721050.aspx) (<http://technet.microsoft.com/en-us/library/ee721050.aspx>).



UAG

Publishing a SharePoint site through a UAG server in a development environment involves setting up networks on the Hyper-V host machine, the server running SharePoint Server 2010, and the UAG server, in addition to installing and configuring the UAG server.

The Appendix describes how to set up these networks and install and configure a UAG server to publish a SharePoint site for Windows Phone 7 development. These steps are suitable for a development or other non-production environment. Please see the [SharePoint publishing solution guide](http://go.microsoft.com/fwlink/?LinkID=206256) (<http://go.microsoft.com/fwlink/?LinkID=206256>) on TechNet when you deploy an environment like this to a production environment.

Intranet Collaboration Scenario

Contoso has many talented people working in many different areas of production, including pharmaceuticals and electronics. The leadership at Contoso has a vision for the integration of people and software to make the workplace not only more productive but more connected, from the perspective of the employee. Contoso wants everyone to enjoy interfacing with the data to enable a happy and productive workforce. Contoso enjoys a strong starting foundation with their deployment of SharePoint 2010 Products. They see SharePoint as the central enterprise application that consolidates user data and activities. Contoso wants to reach further, offering a mobility solution for their associates. To that end, Contoso wants to develop a

mobile application that will enable users to view their activities from SharePoint, keep track of their colleagues, and update lists that are hosted on their SharePoint intranet.

Business Value of Collaboration

Although some companies may find it challenging to directly correlate business value to the collaboration features of SharePoint 2010 Products, Contoso has found a direct relationship between connecting people to each other and building a sense of community with SharePoint. The more employees feel connected to friends at work, the more likely they are to be happy in their job. This happiness and job satisfaction leads to better employee retention (and easier new employee onboarding). Better employee retention reduces the cost of hiring and improves the company's bottom line.

The Application

Through their mobile application, Contoso wants to replicate many of the features that SharePoint My Sites offer in a Web browser. The flow of the application will consist of a panoramic page with list boxes for Recently Viewed People, My Newsfeed, My Activities and My Colleagues. Anywhere a user selects another user's image, the profile of the user they select will open in a new page. Search results will be presented in a list box, and selecting a user will display the user's profile. Figure 1 illustrates these components.

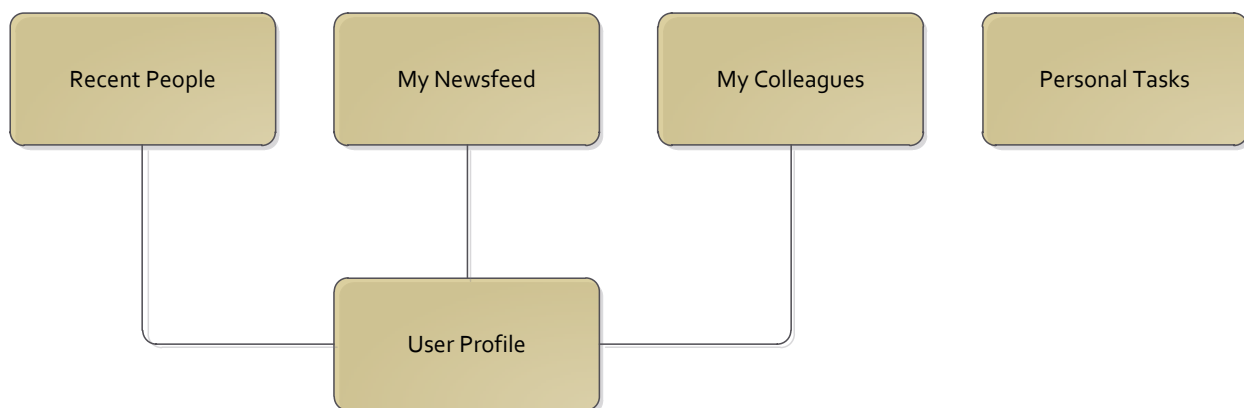


Figure 1: SharePoint social functionality Contoso wants to use in its Windows Phone 7 application.

Personal Task List

Every personal site has a task list that the employee can maintain to track personal progress on assignments. The application will connect to the personal site task list and enable the employee to create, update, and delete tasks.

My Newsfeed

The My Newsfeed (Figure 2) on the SharePoint My Site is the place for users to get the latest news from around the company. Because My Newsfeed is extensible, organizations like Contoso can add additional channels for activity events from other systems.

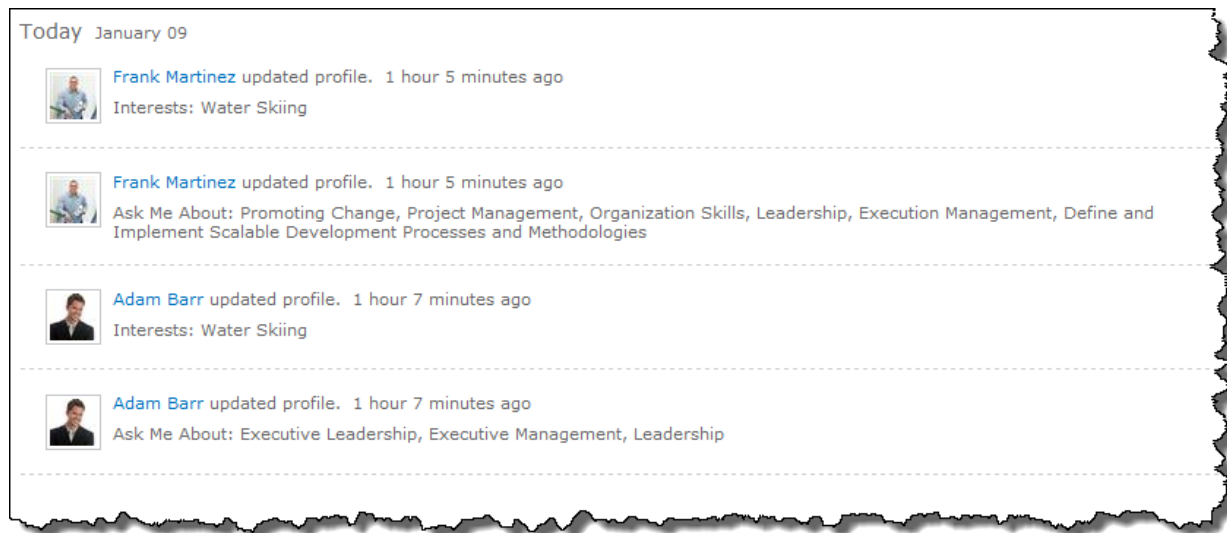


Figure 2: My Newsfeed on a SharePoint site as viewed in a Web browser.

My Colleagues

This list of colleagues maintained by a user ("the people I follow") is a valuable resource (Figure 3). Contoso wants to duplicate that list in the mobile experience.

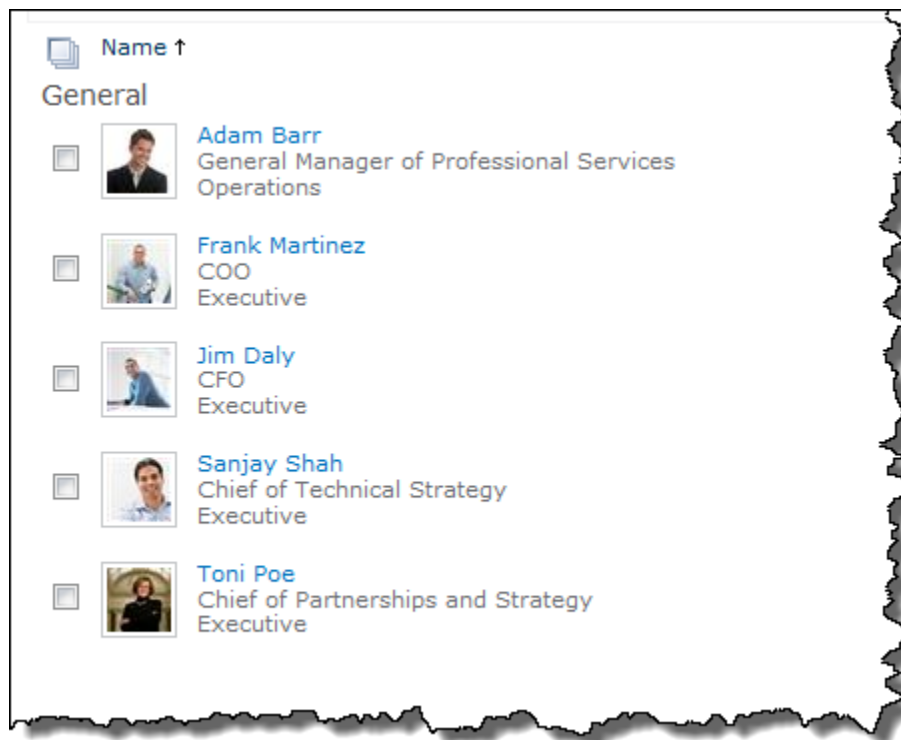


Figure 3: My Colleagues on a SharePoint site as viewed in a Web browser.

The Windows Phone application will make use of the panorama control to display the information, including recently accessed profiles, on a continuous panel. (Figure 4)



Figure 4: The panorama control that displays the holistic view of the Windows Phone 7 application.

Profile Card

Selecting any user will display details from the user's profile (Figure 5). The profile card will show information about the user in the mobile view.



A user profile card for Jim Daly, CFO, displayed on a web browser. The card has a light blue background and a torn-edge effect on the right side. It features a photo of Jim Daly, his name, title, contact information, and a bio.

 **Jim Daly**
CFO
Executive

(425) 555-0116
4252

jimd@contoso.com

▼ [More information](#)

Jim Daly serves as the Chief Financial Officer of Contoso. In this role Jim is responsible for the financial leadership of Contoso. Jim Daly joined Contoso in 1990. Jim has over 15 years of CPA experience. Jim is a graduate of William and Mary Mason School. At Contoso, there are others who share Jim's interests. Jim has started a club for other employees who like working on cars and the members just socialize after work around their shared interests.

Figure 5: Details about a user in a SharePoint site as viewed in a Web browser.



A user profile card for Jim Daly, CFO, displayed on a Windows Phone 7 application. The card has a dark background and a white border. It features a photo of Jim Daly, his name, title, contact information, and a bio.

EN

PROFILE DETAILS

Jim Daly

 **CFO**
jimd@contoso.com
Mobile: (425) 555-0120
Office: (425) 555-0911

I feel I'm really good at what I do because of my strong foundation. I am currently the Director of The Project Management Team. I enjoy promoting change. I have extensive experience defining and implementing scalable development processes and methodologies. Working in the Project Management Organization, it's really important that my clients, such as Tailspin Toys, feel confident in my ability to do my job, and relate to their needs. I got a strong foundation on how to do this from back in my school days at Slippery Elm College. Sometimes I even bond with clients over my favorite interest, horse-back riding.

Figure 6: Details about a user in a SharePoint site as viewed on the Windows Phone 7 application.

Security

Security for the Windows Phone developer should be considered both for data in transit and data at rest. Secure storage of credentials is achieved through the use of encryption and isolated storage. For more information about the security capabilities of the phone, see [Security for Windows Phone](http://go.microsoft.com/fwlink/?LinkId=215977) (<http://go.microsoft.com/fwlink/?LinkId=215977>) on MSDN.

Credential Storage

Windows Phone 7 supports the following cryptographic algorithms:

- AES
- HMACSHA1
- HMACSHA256
- Rfc2898DeriveBytes
- SHA1
- SHA256

It is imperative that the developer of any mobile application that stores credentials on the phone consider the security of the stored credentials in the event that the device is lost or compromised.

Authentication

Windows Phone 7 does not support NTLM authentication. Web service requests from the phone through UAG must create the correct authentication header by retrieving the user's credentials from encrypted storage and then attaching the credentials to the header of the `HttpWebRequest` object. Web requests should be made over HTTPS and the authentication headers passed, as in the following example.

```
upsRequest.Headers["Authorization"] = "Basic " +  
    Convert.ToBase64String(  
        Encoding.UTF8.GetBytes(AppSettings.Username + ":" + AppSettings.Password)) +  
        System.Environment.NewLine;
```

When a WCF Service Reference is used, the header may be added to the SOAP client by accessing the `OperationContextScope`, as in the following example.

```
using (OperationContextScope scope =  
    new OperationContextScope(ups.InnerChannel))  
{  
    //Create the Request Message Property  
    HttpRequestMessageProperty request = new HttpRequestMessageProperty();  
    //Create the authentication and mobile agent header  
    request.Headers[System.Net.HttpRequestHeader.Authorization] =  
        "Basic " +  
        Convert.ToBase64String(Encoding.UTF8.GetBytes(AppSettings.Username + ":" +
```

```

        AppSettings.Password)) + System.Environment.NewLine;
request.Headers[System.Net.HttpRequestHeader.UserAgent] =
    "Microsoft Office Mobile";
//Add the headers to the request
OperationContext.Current.OutgoingMessageProperties.Add(
    HttpRequestMessageProperty.Name, request);

//Call the method
ups.GetUserColleaguesAsync(account);
}

```

SharePoint Services

Consuming Services

The application will consume several SharePoint services to meet the requirements of the business; as summarized in the following table.

Requirement	Service URL	Method
Read and Update List Data	http://sharepoint/_vti_bin/listdata.svc	OData Client Library
View User Newsfeed	http://mysitehost/_layouts/activityfeed.aspx?consolidate=true	Consume the RSS Feed provided.
View User Colleagues	http://mysitehost/_vti_bin/userprofiles-service.aspx	GetUserColleagues
View User Profile	http://mysitehost/_vti_bin/userprofiles-service.aspx	GetUserProfileByName

Accessing SharePoint List Data by Using the OData Client Library

Publishing a SharePoint site collection through a UAG server enables integration between Windows Phone 7 applications and the SharePoint REST APIs. The [OData Client Library for Windows Phone 7 Series CTP](http://go.microsoft.com/fwlink/?LinkId=215984) (<http://go.microsoft.com/fwlink/?LinkId=215984>) enables Windows Phone 7 devices to consume OData feeds. The SharePoint REST APIs may be called by using the OData Client Library to provide create, read, update, and delete (CRUD) operations on SharePoint list data. This provides the ability to integrate Windows Phone 7 devices with collaboration applications built on the SharePoint platform. The following code sample demonstrates how to use the OData Client Library and the SharePoint REST APIs to perform CRUD operations on SharePoint list data.

Because you cannot add Service References for the SharePoint REST APIs to a Windows Phone 7 application, you must generate a proxy class by hand and add the class as a reference to the project in Microsoft Visual Studio®. [DataSvcUtil.exe \(http://go.microsoft.com/fwlink/?LinkID=215987\)](http://go.microsoft.com/fwlink/?LinkID=215987) is used to generate the proxy class. In this scenario, the proxy class is named **ContosoIntranetDataContext**. This class is used extensively in the code samples that follow.

The **LoadTask** method demonstrates how to load tasks from a SharePoint Tasks list. In this method, the **ContosoIntranetDataContext** class uses settings from **IsolatedStorage** to determine which SharePoint site to connect to. The **SendingRequest** event handler is assigned to the instance of the **ContosoIntranetDataContext**. This event handler is fired when the **ContosoIntranetDataContext** class invokes a request to the SharePoint REST APIs. The **requestUri** defines the REST operation, and the **BeginExecute** method submits the request. The **BeginExecute** method also registers the asynchronous callback method that fires when the query is complete.

```
private void LoadTasks()
{
    Deployment.Current.Dispatcher.BeginInvoke(() =>
    {
        if (allTasks == null)
        {
            allTasks = new ObservableCollection<Task>();
        }

        ObservableCollection<TasksItem> tasks = new
            ObservableCollection<TasksItem>();
        //Retrieve the settings from isolated storage
        SettingsModel settings =
            (IsolatedStorageSettings.ApplicationSettings["Settings"]
            as SettingsModel);
        //Set up the ODATA context to point to the SharePoint site
        context = new ContosoIntranetDataContext(
            new Uri(settings.ServerUri + "/_vti_bin/listdata.svc"));
        //Register the event handler used to authenticate to UAG
        context.SendingRequest += new EventHandler<SendingRequestEventArgs>(
            context_SendingRequest);

        //Set the URI to query the Tasks list
        //Expand is used to retrieve lookup column values
```

```

Uri requestUri = new Uri(context.BaseUri.OriginalString +
    "/Tasks()?$expand=AssignedTo,CreatedBy,ModifiedBy");

//Start the async call to query SharePoint
context.BeginExecute<TasksItem>(requestUri, QueryCallback, null);
});
}

```

The **SendingRequest** event handler is fired when the OData query is sent to SharePoint 2010 Products. The code in this event handler is significant: without this code, a UAG server cannot correctly identify a Windows Phone 7 device and interact properly to authenticate a user. The **User-Agent header** causes a UAG server to respond with an HTTP 401 instead of a 302. The **Authorization header** includes the encrypted credentials for the user accessing the OData feed. The encrypted credentials are used by the UAG server to authenticate the user.

```

private void context_SendingRequest(object sender, SendingRequestEventArgs e)
{
    e.RequestHeaders["User-Agent"] = "Microsoft Office Mobile";
    e.RequestHeaders["Authorization"] = "Basic " + Convert.ToBase64String(
        Encoding.UTF8.GetBytes(App.Credential.Name + ":" +
        App.Credential.Password)) + System.Environment.NewLine;
}

```

After the query to the server running SharePoint Server is complete, the **QueryCallback** method fires. This method parses the results returned from the query and adds them to the ObservableCollection bound to the UI elements in the phone application.

```

private void QueryCallback(IAsyncResult asyncResult)
{
    IEnumerable<TasksItem> results;
    allTaskItems = new ObservableCollection<TasksItem>();
    results = context.EndExecute<TasksItem>(asyncResult).ToList()
        as IEnumerable<TasksItem>;
    ObservableCollection<Task> returnedTasks =
        new ObservableCollection<Task>();

    foreach (TasksItem tasksItem in results)

```

```

{
    //Code omitted for brevity: Retrieve metadata about the task...

    //Create the new Task and set its properties
    Task task = new Task()
    {
        Title = tasksItem.Title,
        Priority = taskPriority,
        TaskStatus = taskStatus,
        Body = tasksItem.Description,
        Author = authorUser.Name,
        Editor = editorUser.Name,
        AssignedTo = assignedToUser.Name,
        StartDate = DateTime.Parse(tasksItem.StartDate.ToString()),
        Modified = DateTime.Parse(tasksItem.Modified.ToString()),
        Created = DateTime.Parse(tasksItem.Created.ToString()),
        DueDate = DateTime.Parse(tasksItem.DueDate.ToString()),
        UIVersion = 1,
        Last_x0020_Modified =
            DateTime.Parse(tasksItem.Modified.ToString()),
        Created_x0020_Date =
            DateTime.Parse(tasksItem.Created.ToString()),
        PercentComplete = (int)finalPercentComplete,
        ListID = tasksItem.Id
    };

    //Add each task to the ObservableCollection bound to UI elements.
    returnedTasks.Add(task);
    allTaskItems.Add(tasksItem);
}

allTasks = returnedTasks;
//UI callback methods omitted for brevity
}

```


The **SaveTask** method creates new tasks and updates existing tasks. The following code demonstrates how to use the OData Client Library to create new tasks or update existing tasks in a SharePoint task list from a Windows Phone 7 device. The **TaskItem** class represents a task in a task list. A **TaskItem** instance is created, and its properties are set to the values in the form used to create or edit tasks. The **TaskItem** instance is then compared to the tasks already loaded into the phone to see if the task already exists. This check determines whether the OData Client Library is used to create a new task in the SharePoint task list or update an existing one. The **BeginSaveChanges** method invokes the proper operation and registers the **saveChangesCallback** callback method.

```
public void SaveTask(Task task, Action<Task> callback)
{
    taskToSave = task;

    #region Create/update new task object

    saveTaskCallback = callback;
    //Retrieve the settings from isolated storage
    SettingsModel settings =
        (IsolatedStorageSettings.ApplicationSettings["Settings"]
        as SettingsModel);
    //Set up the ODATA context to point to the appropriate SharePoint site
    context = new ContosoIntranetDataContext(
        new Uri(settings.ServerUri + "/_vti_bin/listdata.svc"));
    //Register the event handler used to authenticate to UAG
    context.SendingRequest +=
        new EventHandler<SendingRequestEventArgs>(context_SendingRequest);

    TaskItem tasksItem = new TaskItem();

    tasksItem.Title = task.Title;
    tasksItem.Description = task.Body;
    tasksItem.AssignedToId = assignedToUserID;
    tasksItem.PriorityValue = task.Priority.DisplayString;
    tasksItem.StartDate = task.StartDate;
    tasksItem.DueDate = task.DueDate;

    //Set percent complete
```

```

if (task.PercentComplete > 0)
{
    tasksItem.Complete = (double)task.PercentComplete / 100;
}
else
{
    tasksItem.Complete = (double)0;
}

//Set tasks status
tasksItem.StatusValue = task.TaskStatus.DisplayString;

//If the task already exists then update it
if (allTasks.Count > 0 && DoesTaskExist(taskToSave) && task.ListID != 0)
{
    tasksItem.Id = task.ListID;
    context.AttachTo("Tasks", tasksItem, "");
    context.UpdateObject(tasksItem);
}
//If the task does not exist then create it
else
{
    context.AddToTasks(tasksItem);
}

Deployment.Current.Dispatcher.BeginInvoke(
() =>
{
    //Start the async call to SharePoint to commit the changes
    context.BeginSaveChanges(saveChangesCallBack, context);
}
);
}

```

After the query is complete, the **saveChangesCallBack** fires. This method parses the results returned from the query and updates the ObservableCollection bound to the UI elements in the phone.

```
private void saveChangesCallBack(IAsyncResult asyncResult)
{
    Deployment.Current.Dispatcher.BeginInvoke(
        () =>
        {
            //Get the data context from the response
            context = asyncResult.AsyncState as ContosoIntranetDataContext;
            //Call the endsavechanges method to commit the change
            DataServiceResponse response = context.EndSaveChanges(asyncResult);
            //If the task already exists then update it
            if (allTasks.Count > 0 && DoesTaskExist(taskToSave))
            {
                for (int i = 0; i < allTasks.Count; i++)
                {
                    if (allTasks[i].ListID == taskToSave.ListID)
                    {
                        //Update the corresponding task in the in-memory collection
                        allTasks[i] = taskToSave;
                        //Call the callback to tell the UI the save operation is complete
                        saveTaskCallback.DynamicInvoke(taskToSave);
                        break;
                    }
                }
            }
            //If the task does not exist then create it
            else
            {
                //Add the task to the in-memory collection
                allTasks.Add(taskToSave);
                //Call the callback to tell the UI the save operation is complete
                saveTaskCallback.DynamicInvoke(taskToSave);
            }
        }
    );
}
```

```
);
}
```

The **DeleteTask** method demonstrates how to use the OData Client Library to delete tasks in a SharePoint task list from a Windows Phone 7 device. This method takes the ID associated with a task selected in the Windows Phone 7 application and creates a **Task** object based on the ID. The **Task** object is used to locate the corresponding **TasksItem** in the OData Client Library context associated with the SharePoint task list. After the corresponding entry is located, it is marked for deletion. The **BeginSaveChanges** callback method invokes the SharePoint REST API and registers the **DeleteCallback** callback method.

```
public void DeleteTask(int ID, Action callback)
{
    deleteTaskCallback = callback;
    taskToDelete = GetTask(ID);

    //Retrieve the settings from isolated storage
    SettingsModel settings =
        (IsolatedStorageSettings.ApplicationSettings["Settings"]
        as SettingsModel);
    //Set up the ODATA context to point to the appropriate SharePoint site
    context = new ContosoIntranetDataContext(
        new Uri(settings.ServerUri + "/_vti_bin/listdata.svc"));
    //Register the event handler used to authenticate to UAG
    context.SendingRequest +=
        new EventHandler<SendingRequestEventArgs>(context_SendingRequest);

    foreach (TasksItem tasksItem in allTaskItems)
    {
        if (tasksItem.Id == taskToDelete.ListID)
        {
            context.MergeOption = MergeOption.OverwriteChanges;
            context.AttachTo("Tasks", tasksItem, "*");
            context.DeleteObject(tasksItem);
            //Start the async call to SharePoint to commit the delete
            context.BeginSaveChanges(DeleteCallback, context);
            break;
        }
    }
}
```

```

    }
}

```

After the query is complete, the **DeleteCallback** method fires. This method parses the results returned from the query and removes the task in the **ObservableCollection** bound to the UI elements in the phone.

```

private void DeleteCallback(IAsyncResult asyncResult)
{
    Deployment.Current.Dispatcher.BeginInvoke(
        () =>
        {
            //Get the data context from the response
            context = asyncResult.AsyncState as ContosoIntranetDataContext;
            //Call the endsavechanges method to commit the change
            DataServiceResponse response = context.EndSaveChanges(asyncResult);

            // Remove the task from the in-memory collection as well
            if (taskToDelete != null)
            {
                //Remove the task from the in-memory collection
                GetAllTasks().Remove(taskToDelete);
                //Call the callback to tell the UI the delete operation is complete
                deleteTaskCallback.DynamicInvoke();
            }
        }
    );
}

```

Activity Feed RSS

The personal newsfeed and recent activities are published from the My Site host as an RSS feed. The application only needs to request the RSS feed, download the string, and add the results to a collection object. The activities are published from the My Site host on the URL

http://<mysitehost>/_layouts/activityfeed.aspx?consolidated=true. (Ensure that you have tested the RSS feed by following the directions in the Appendix at the end of this paper before trying to access it via the code approach outlined below.)

The **LoadNewsfeed** method demonstrates how to request the consolidated activity feed for the authenticated user through UAG. A new **HttpWebRequest** is created by passing the URI of the consolidated activity feed. The necessary client headers are added to the request. Account and Password are variables accessed from encrypted isolated storage. To facilitate the use of Model View View Model pattern, the MVVM Light Toolkit is employed (for more information, see the [MVVM Light Toolkit Web site](http://go.microsoft.com/fwlink/?LinkId=216135) (<http://go.microsoft.com/fwlink/?LinkId=216135>)). In this sample, the **DispatcherHelper.CheckBeginInvokeOnUI** is called to handle the **ResponseStream** and the results are passed to **AddNewsfeedItems** to extract the result values and add them to the **NewsfeedItems** collection.

```
private void LoadNewsfeed()
{
    //My Newsfeed RSS URL
    string url =
String.Format("{0}/my/_layouts/activityfeed.aspx?consolidated=true",
AppSettings.Url);
    System.Uri authServiceUri = new Uri(url);
    HttpWebRequest client =
        WebRequest.CreateHttp(authServiceUri) as HttpWebRequest;
    //Add the necessary headers for UAG
    client.Headers["User-Agent"] = "Microsoft Office Mobile";
    client.Headers["Authorization"] = "Basic " +
        Convert.ToBase64String(Encoding.UTF8.GetBytes(AppSettings.Username + ":" +
AppSettings.Password)) +
        System.Environment.NewLine;

    client.AllowReadStreamBuffering = true;
    client.AllowAutoRedirect = true;
    // Call and handle the response.
    client.BeginGetResponse((asResult) =>
    {
        DispatcherHelper.CheckBeginInvokeOnUI (
        () =>
        {
            try
            {
                var response = client.EndGetResponse(asResult);
                StreamReader reader = new
                    StreamReader(response.GetResponseStream());
```

```

        string responseString = reader.ReadToEnd();

        AddNewsfeedItems(responseString);
    }
    catch (WebException failure)
    {
        throw failure;
    }
    });
},
null);
}

```

After the download is complete, the **AddNewsfeedItems** adds the results to the collection that is bound to the UI. The **responseString** is parsed into an **XDocument**, and the two namespaces are added. LINQ is used to query the entries collection and create **MyNewsfeedViewModel** objects that store the resulting newsfeed items and the author information. Finally, the items are added to the **MyNewsFeedItems** collection.

```

private void AddNewsfeedItems(string responseString)
{
    //Parse the XML Response
    XDocument newsfeedDoc = XDocument.Parse(responseString);
    //Add the necessary Namespaces
    XNamespace ns = "http://www.w3.org/2005/Atom";
    XNamespace af = "AF";
    //Use LINQ to extract the information into a ViewModel
    IEnumerable<MyNewsfeedViewModel> entries = from entry in
        newsfeedDoc.Descendants(XName.Get("entry", ns.NamespaceName))
        select new MyNewsfeedViewModel()
    {
        Summary = entry.Element(ns + "summary").Value,
        Published = entry.Element(ns + "published").Value,
        Author =
            (from author in entry.Descendants(XName.Get("author",
                ns.NamespaceName))
            select new PersonViewModel(NavigationService)
            {
                AccountName = author.Element(af + "AccountName").Value,

```

```

        Name = author.Element(ns + "name").Value,
        PersonalSiteUrl = author.Element(ns + "uri").Value,
        Email = author.Element(ns + "email").Value,
        PictureUrl = author.Element(af + "Picture").Value
    }).FirstOrDefault());

//Add the resulting items to the Collection bound to the UI controls
DispatcherHelper.CheckBeginInvokeOnUI(() =>
{
    foreach (MyNewsfeedViewModel e in entries.ToList())
    {
        MyNewsfeedItems.Add(e);
    }
});
}

```

User Profile Service - Colleagues

The SharePoint User Profile service provides the ability to view, create, edit, and manage user profile information in SharePoint 2010 Products. The service is the primary entry point for the application to retrieve information about user colleagues and user profiles. The following code demonstrates how to call the User Profile service from Windows Phone7 through UAG.

Create the User Profile Service Reference

Begin by adding a Service Reference to your Windows Phone 7 project. Enter the URL to the SharePoint User Profile Service end point, for example:

```
http://spwp7intranet.contoso.com/_vti_bin/userprofiles-service.asmx
```

Give the Service Reference a recognizable name, such as **UserProfileService**, and then click **OK**.

The GetUserColleagues Web Method

The **LoadColleagueData** method uses the **GetUserColleaguesAsync** method of the User Profile Service to return a collection of colleagues as **ContactData** objects. After Creating the **BasicHttpBinding** and **EndpointAddress**, a new **UserProfileServiceSoapClient** is created with the required parameters. The Service calls in Silverlight must be performed asynchronously. A new OnCompleted event handler is added for the **GetUserColleaguesCompleted** event. To add the headers required by UAG, an **OperationContextScope** is used and an **HttpRequestMessageProperty** is created to hold the two headers that are required to authenticate against UAG. The headers are added to the outgoing message, and the asynchronous call is made to **GetUserColleaguesAsync** by using the user's account as a parameter.


```

private void LoadColleagueData()
{
    string url =
String.Format("{0}/_vti_bin/userprofiles/service.asmx", AppSettings.Url);
    BasicHttpBinding binding = new BasicHttpBinding();
    EndpointAddress endpoint = new EndpointAddress(url);
    UserProfileService.UserProfileServiceSoapClient ups =
        new UserProfileServiceSoapClient(binding, endpoint);

    //Add the Event Completed Handler
    ups.GetUserColleaguesCompleted +=
        new EventHandler<GetUserColleaguesCompletedEventArgs>(
            ups_GetUserColleaguesCompleted);

    //Add the credentials
    using (OperationContextScope scope =
        new OperationContextScope(ups.InnerChannel))
    {
        //Create the Request Message Property
        HttpRequestMessageProperty request = new HttpRequestMessageProperty();
        //Create the authentication and mobile agent header
        request.Headers[System.Net.HttpRequestHeader.Authorization] =
            "Basic " +
Convert.ToBase64String(Encoding.UTF8.GetBytes(AppSettings.Account +
            ":" + AppSettings.Password)) + System.Environment.NewLine;
        request.Headers[System.Net.HttpRequestHeader.UserAgent] =
            "Microsoft Office Mobile";

        //Add the headers to the request
        OperationContext.Current.OutgoingMessageProperties.Add(
            HttpRequestMessageProperty.Name, request);

        //Call the method
        ups.GetUserColleaguesAsync(account);
    }
}

```

When the **GetUserColleaguesAsync** method returns results, the **ups_GetUserColleaguesCompleted** event is called. If there is no error, LINQ is used to create a list of **PersonViewModels** from the resulting **ContactData**. The last step is to add the **PersonViewModels** from the list to the **ColleaguesList** on the UI thread by calling **CheckBeginInvoke** and passing in our list of **PersonViewModels**.

```
private void ups_GetUserColleaguesCompleted(object sender,
    GetUserColleaguesCompletedEventArgs e)
{
    if (e.Error == null)
    {
        //Create a list of PersonViewModels
        IEnumerable<PersonViewModel> colleagues =
            from contact in e.Result
            select new PersonViewModel()
            {
                AccountName = contact.AccountName,
                UserProfileID = contact.UserProfileID.ToString(),
                Name = contact.Name,
                Title = contact.Title,
                Email = contact.Email,
                PersonalSiteUrl = contact.Url
            };

        //Load the Colleagues list on the UI thread
        DispatcherHelper.CheckBeginInvokeOnUI(() =>
        {
            foreach (PersonViewModel c in colleagues.ToList())
            {
                Colleagues.Add(c);
            }
        });
    }
    else
    {
        Debug.WriteLine("Error loading the Colleagues List: {0}",
            e.Error.Message);
    }
}
```

User Profile Service - User Profile Data

The **GetUserColleagues** Web method does not return user profiles. The method returns **ContactData** that can be passed back to the User Profile Service **GetUserProfileByName** method that returns a complete User Profile object, which can then be used to display a complete user profile. The following method is used to return the user profile details for our colleagues. Specifically, we will return the fields **AboutMe**, **WorkPhone**, **MobilePhone**, and **PictureURL**.

The **GetUserProfileProperties** method takes a **PersonViewModel** object for the person whose profile we want to retrieve. In this method, we construct a message inspector to resolve an issue that causes the PropertyData return value to fail to parse correctly. We then create a **BasicHttpMessageInspectorBinding** that takes the message inspector as a parameter, passes the binding and endpoint to the constructor for the **UserProfileServiceSoapClient**, and registers the **GetUserProfileByNameCompleted** event handler. To add the request headers that UAG requires, an **OperationContextScope** adds the **Authorization** and **UserAgent** headers to the request. Finally, we call the **GetUserProfileByNameAsync** method and pass the **Account Name** for the user and the **PersonViewModel** we want to update.

```
private void GetUserProfileProperties(PersonViewModel person)
{
    //URL for the service
    string url = String.Format("{0}/_vti_bin/userprofiles-service.asmx",
        AppSettings.ServerUrl);

    //Create the Message Inspector
    SPAsmxMessageInspector messageInspector = new SPAsmxMessageInspector();
    //Apply the Message Inspector to the Binding
    BasicHttpMessageInspectorBinding binding = new
        BasicHttpMessageInspectorBinding(messageInspector);

    EndpointAddress endpoint = new EndpointAddress(url);
    UserProfileService.UserProfileServiceSoapClient ups = new
        UserProfileServiceSoapClient(binding, endpoint);

    //Add the Event Completed Handler
    ups.GetUserProfileByNameCompleted += new
        EventHandler<GetUserProfileByNameCompletedEventArgs>
        (ups_GetUserProfileByNameCompleted);
}
```

```

using (OperationContextScope scope = new
    OperationContextScope(ups.InnerChannel))
{
    //Create the Request Message Property
    HttpRequestMessageProperty request = new HttpRequestMessageProperty();
    //Create the authentication and mobile agent header
    request.Headers[System.Net.HttpRequestHeader.Authorization] = "Basic " +
        Convert.ToBase64String(Encoding.UTF8.GetBytes(AppSettings.UserName + ":" +
            AppSettings.Password)) + System.Environment.NewLine;
    request.Headers[System.Net.HttpRequestHeader.UserAgent] =
        "Microsoft Office Mobile";
    //Add the headers to the request
    OperationContext.Current.
        OutgoingMessageProperties.Add(HttpRequestMessageProperty.Name, request);

    Debug.WriteLine("Getting User Profile for: {0}", person.AccountName);

    //Call the method
    ups.GetUserProfileByNameAsync(person.AccountName, person);
}
}

```

When **GetUserProfileByNameCompleted** is called, the return property values (if there are any) are inspected and assigned to the appropriate property on the **PersonViewModel**.

```

private void ups_GetUserProfileByNameCompleted(object sender,
    GetUserProfileByNameCompletedEventArgs e)
{
    if (e.Error == null)
    {
        Debug.WriteLine("Got the user profile for {0}",
            ((PersonViewModel)e.UserState).AccountName);
        foreach (UserProfileService.PropertyData propertyData in e.Result)
        {
            switch (propertyData.Name)
            {
                case "AboutMe":

```

```

        ((PersonViewModel) e.UserState).AboutMe = propertyData.Values.Count >
        0 ? (propertyData.Values[0].Value as string): String.Empty;
        break;

    case "WorkPhone":
        ((PersonViewModel)e.UserState).WorkPhone = propertyData.Values.Count >
        0 ? (propertyData.Values[0].Value as string) : String.Empty;
        break;

    case "CellPhone":
        ((PersonViewModel)e.UserState).MobilePhone = propertyData.Values.Count >
        0 ? (propertyData.Values[0].Value as string) : String.Empty;
        break;

    case "PictureURL":
        ((PersonViewModel)e.UserState).PictureUrl = propertyData.Values.Count >
        0 ? (propertyData.Values[0].Value as string) : String.Empty;
        break;

    }
}
else
{
    Debug.WriteLine(e.Error.Message);
}
}

```

Note: The message inspector in this example is only recommended for use in an environment in which you have applied the latest Silverlight updates. For more information, see [Workaround for accessing some ASMX services from Silverlight 4](http://go.microsoft.com/fwlink/?LinkId=216134) (http://go.microsoft.com/fwlink/?LinkId=216134) and [Silverlight and SharePoint User Profile Service GUIDs](http://go.microsoft.com/fwlink/?LinkId=216136) (http://go.microsoft.com/fwlink/?LinkId=216136).

Testing

Testing the Application on the Emulator

The application in this sample is tested on the Windows Phone 7 development emulator. The emulator uses the network connection of the host development machine. If necessary, host file entries may be added to the development machine so that the emulator can resolve the addresses of the development UAG server.

Testing the Application on a Device

Testing the application on a physical Windows Phone device requires that the phone be connected to a publicly available Wi-Fi connection that provides DNS and routing to the destination test UAG server or a test UAG server exposed to the Internet.

Marketplace Considerations

After you have developed your Windows Phone 7 application, it is important to take the proper steps to securely publish your application to the Windows Phone Marketplace. Microsoft has already put several mechanisms in place to prevent software piracy of Windows Phone 7 applications. You can read more about these measures in the [Windows Phone Marketplace Anti-Piracy Model white paper](http://go.microsoft.com/fwlink/?LinkId=216137) (<http://go.microsoft.com/fwlink/?LinkId=216137>).

At a high level, the minimum number of steps you must take to publish a Windows Phone 7 application to the Windows Phone Marketplace include creating a developer account and publishing your application to the marketplace. The [App Hub Developer Registration Walkthrough](http://go.microsoft.com/fwlink/?LinkId=216138) article (<http://go.microsoft.com/fwlink/?LinkId=216138>) describes how to register a developer account. It's important to note that a developer account may be tied to an individual or a company. Depending on your needs, you can sign up for an account that makes the most sense for you. The registration fee for both account types is the same. The [Windows Phone 7 Application Submission Walkthrough](http://go.microsoft.com/fwlink/?LinkId=216139) article (<http://go.microsoft.com/fwlink/?LinkId=216139>) describes how to submit your application for verification and publication. If your application adheres to the [Windows Phone 7 Application Certification Requirements \[PDF\]](http://go.microsoft.com/?linkid=9730558) (<http://go.microsoft.com/?linkid=9730558>), it will be published to the Windows Phone Marketplace.

At the time of publication of this white paper, there is no “private marketplace” for organizations to prevent the public distribution of their applications. See the App Hub (<http://create.msdn.com>) for updates and information about the ability of organizations to publish applications privately.

Although Microsoft has put many safeguards into place, you should still take extra precautions to safeguard your code in the event that someone obtains a copy of the .xap file representing your Windows Phone 7 application. Before you go through the steps to publish your application to the Windows Phone Marketplace, you should make sure you obfuscate your applications to protect your intellectual property. There are already several different tools available to obfuscate Windows Phone 7 applications.

Although this may not be an all-inclusive list, it should give you a good starting point to explore the options currently available.

[PreEmptive Solutions Dotfuscator Windows Phone Edition
\(http://www.preemptive.com/windowsphone7.html\)](http://www.preemptive.com/windowsphone7.html)

[RedGate SmartAssembly 6 EAP \(http://www.red-gate.com/MessageBoard/viewforum.php?f=116\)](http://www.red-gate.com/MessageBoard/viewforum.php?f=116)

[DeepSea Obfuscator \(http://www.deepseaobfuscator.com/\)](http://www.deepseaobfuscator.com/)

Proper obfuscation is an iterative process that takes time to test and verify. Be sure to build time into your project plan to obfuscate your Windows Phone 7 applications, test them, and verify that the level of obfuscation meets your needs.

You can use the Windows Phone 7 Application Deployment tool (Figure 7) to test obfuscated Windows Phone 7 applications. This tool allows you to deploy your obfuscated Windows Phone 7 applications to an actual Windows Phone 7 device that you have registered by using your developer account, or to the Windows Phone 7 emulator.

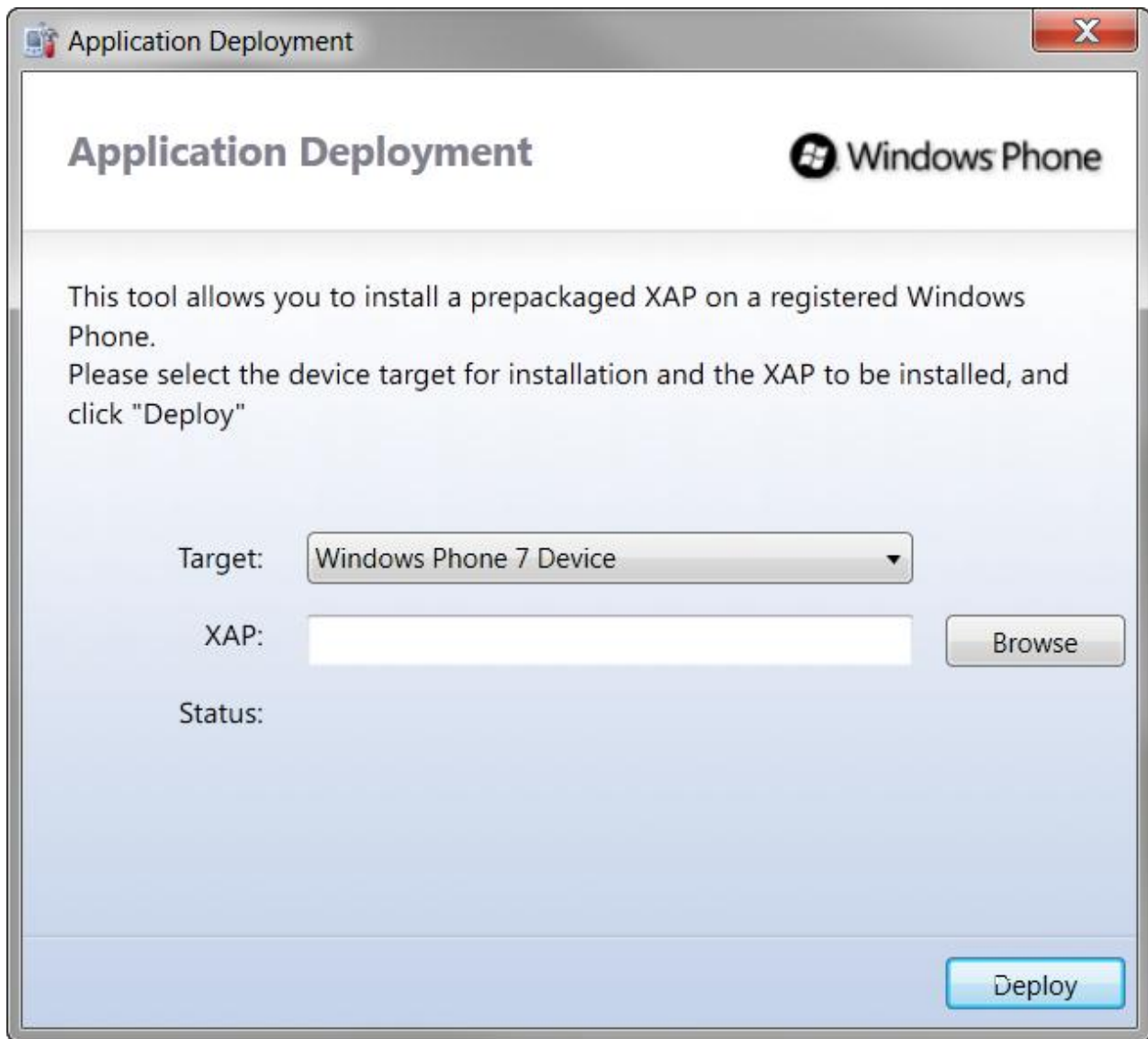


Figure 7: Windows Phone 7 Application Deployment Tool

After you deploy and test an obfuscated Windows Phone 7 application, you may find that at very high levels of obfuscation (where you are turning on every possible option to protect your code) the application will not run properly on a Windows Phone 7 device or in the Windows Phone 7 emulator. On the other hand, you may find that using a minimal level of obfuscation settings will ensure your application runs properly, but the level of obfuscation does not meet your needs.

You can take the following steps to determine what level of obfuscation is applied to your application by an obfuscation tool. First, locate the .xap file representing your Windows Phone 7 application and rename the file extension to .cab or .zip. Then, open the archive file and extract the contents. Next, use a tool like RedGate .NET Reflector to open the assembly that corresponds to your application. Opening the Resources node will allow you to see all the XAML files and save them to your hard disk. (Figure 8)

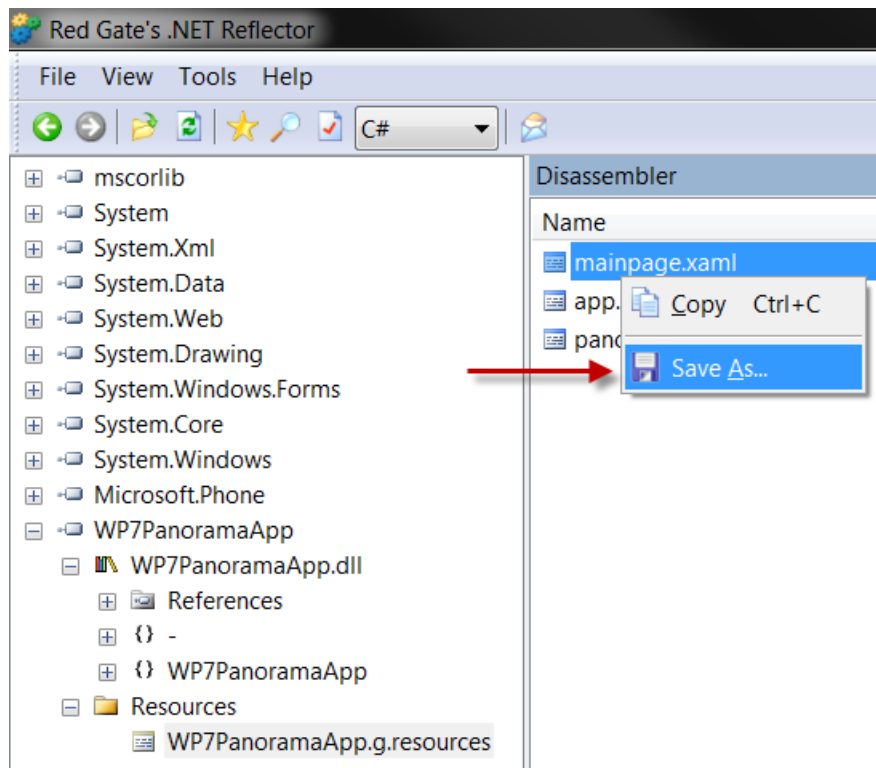


Figure 8: Saving XAML files with .NET Reflector

You can then examine the XAML files to see the level of obfuscation that has been applied to the code. You may be surprised that low levels of obfuscation do not change the contents of a XAML file at all and high levels merely remove whitespace. This really encourages the use of the Model View View Model pattern to separate presentation logic and code from data access layers.

You can also examine the code in the assemblies that your Windows Phone 7 application relies on. Figure 9 shows an obfuscated assembly.

```

namespace WP7PanoramaApp
{
    public class App : Application
    {
        // Fields
        private static MainViewModel a;
        private bool b;
        private bool c;
        [CompilerGenerated]
        private PhoneApplicationFrame d;

        // Methods
        static App();
        public App();
        private void a(object A_0, NavigationFailedEventArgs A_1);
        private void b();
        private void c(object A_0, ApplicationUnhandledExceptionEventArgs A_1);
        private void d(object A_0, NavigationEventArgs A_1);
        [DebuggerNonUserCode]
        public void InitializeComponent();
    }
}

```

Figure 9: Obfuscation applied to a Windows Phone 7 application assembly.

One key thing to remember about obfuscation is that no level of obfuscation can prevent a very determined hacker from reverse-engineering most of your source code. Obfuscation techniques are designed to protect intellectual property by providing a barrier that makes it too hard to reverse-engineer code and steal it.

Conclusion

There are many resources available to the Windows Phone 7 developer. The place to start is the [App Hub on MSDN](http://create.msdn.com), <http://create.msdn.com>. The articles, blogs, and forums assist developers from beginner to advanced development scenarios. SharePoint 2010 Products provide a wide array of capability for organizations that want to collaborate and share knowledge, expertise, and information. Enterprise applications that leverage the capabilities of both mobility and SharePoint will provide a powerful advantage to companies that can efficiently manage, maintain, and deploy solutions based on the .NET platform.

Appendix – Installation and Configuration of UAG for SharePoint 2010 Products

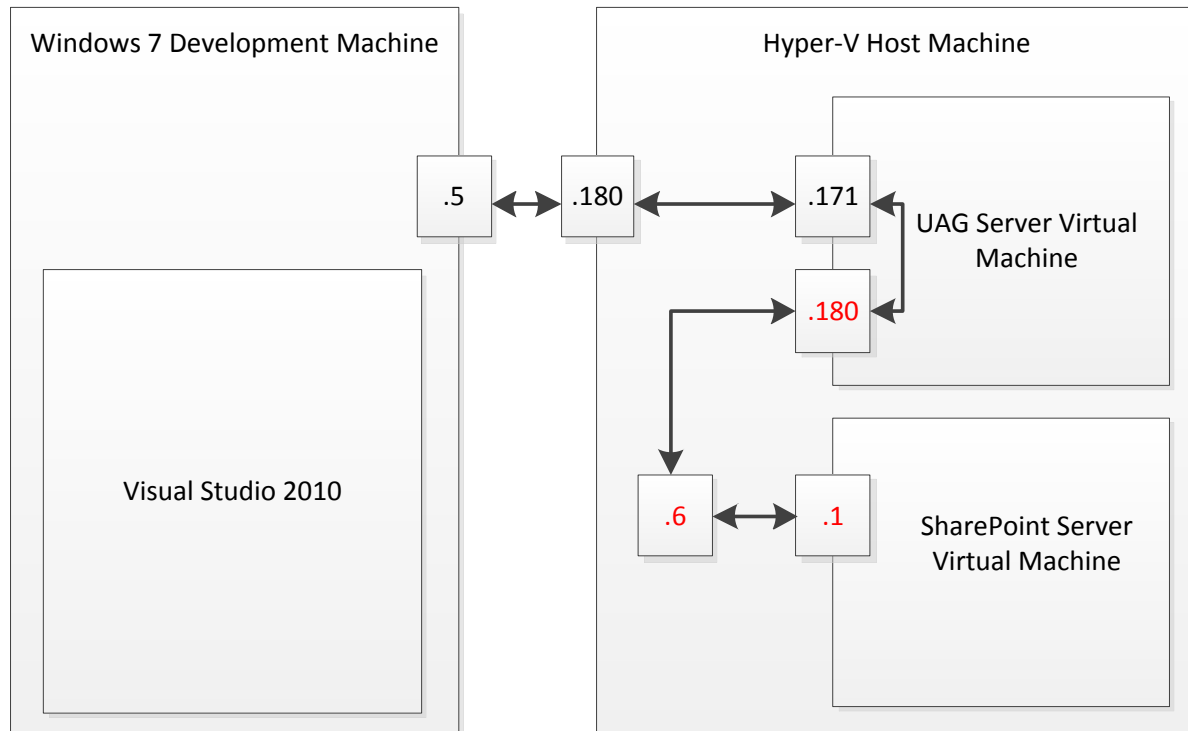
This section describes how to set up networking and install and configure a UAG server to publish a SharePoint site for Windows Phone 7 development. These steps are suitable for a development environment. For information about deploying in a production environment, see the [UAG publishing solution guide](http://go.microsoft.com/fwlink/?LinkID=206256) (<http://go.microsoft.com/fwlink/?LinkID=206256>) on TechNet. The UAG server will request at least 4 GB RAM during the installation process. For a demo environment, 2 GB is sufficient and the warning can be safely ignored.

1. Create a SharePoint Server Virtual Machine and a UAG Virtual Machine.

In this example, the [2010 Information Worker Demonstration and Evaluation Virtual Machine \(RTM\)](http://go.microsoft.com/fwlink/?LinkID=189314) (<http://go.microsoft.com/fwlink/?LinkID=189314>) is used for the SharePoint Server Virtual Machine. To create the UAG Server Virtual Machine, install Windows Server 2008 R2 in a new virtual machine, install all updates by using Windows Update, and proceed with the steps below.

Networking

The following diagram illustrates how the server running SharePoint Server, UAG server, and Windows Phone 7 development machine are connected. The documentation that follows describes how to implement this scenario.



192.168.150.x – Internal Network | 192.168.1.x – External Network

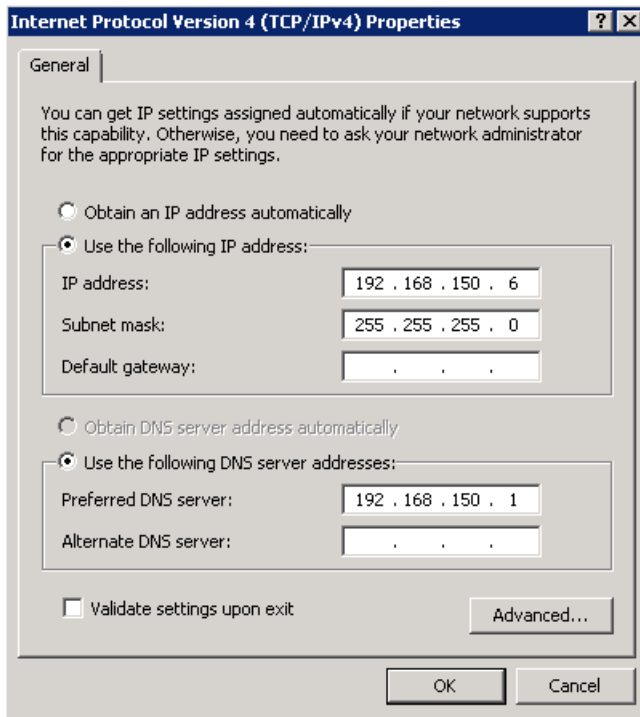
2. Set Up Hyper-V Host Machine Virtual Networks

In the Hyper-V host machine, create two virtual networks as follows. Setting up two networks simulates a DMZ environment where the UAG server talks to the Internet on one network adapter and talks to internal resources — in this case, the server running SharePoint Server — on another network adapter.

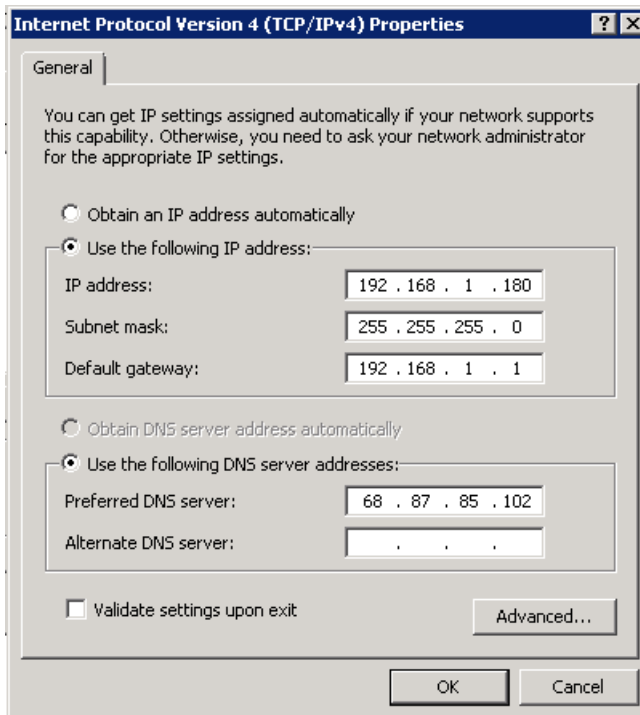
Name	Connection Type	Description
Internal	Internal Only	Connection between the UAG server and the server running SharePoint Server.
UAG External	External – Bound to network adapter on Hyper-V host machine.	Connection between the WP7 development machine and the UAG server.

Note: The IP addresses may be slightly different in your environment. The key is to have the Internal network operate on a different subnet than the external network.

Set the TCP/IP settings for the Internal network adapter on the Hyper-V host machine as follows.

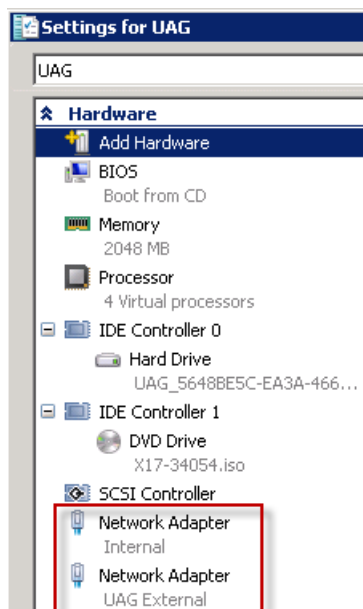


Set the TCP/IP settings for the External network adapter on the Hyper-V host machine as follows.

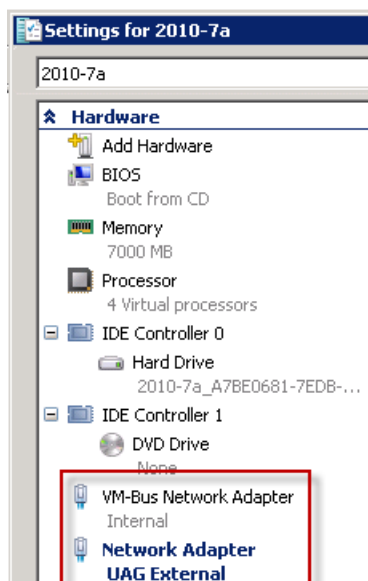


3. Set Up Hyper-V Virtual Machine Networks

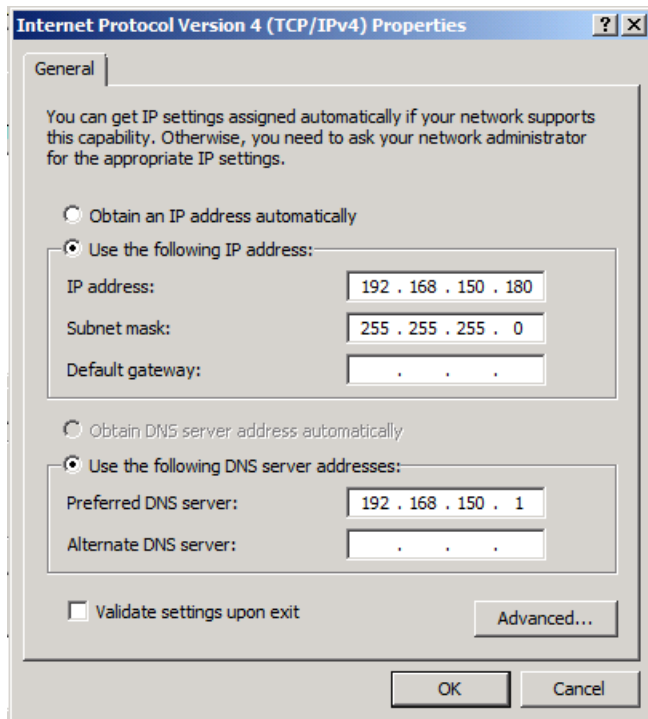
On the UAG server Virtual Machine, add another network adapter and configure the UAG server Virtual Machine to use the Internal and UAG External virtual networks in Hyper-V.



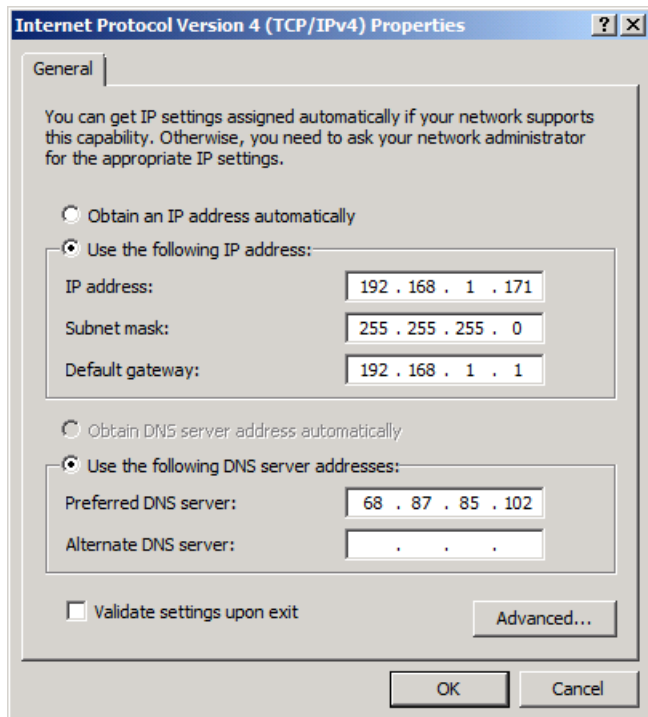
On the SharePoint Server Virtual Machine, add another network adapter and configure the SharePoint Server Virtual Machine to use the Internal and UAG External virtual networks in Hyper-V.



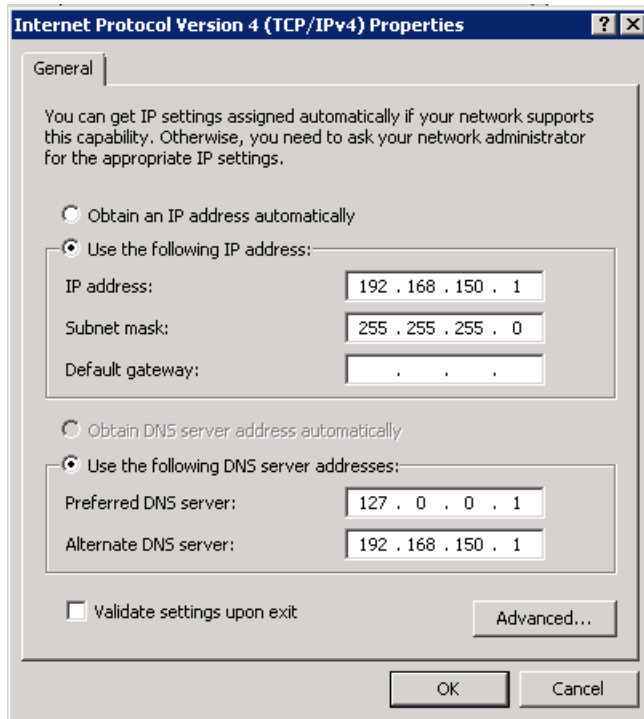
Set the TCP/IP settings for the Internal network adapter on the UAG server Virtual Machine as follows.



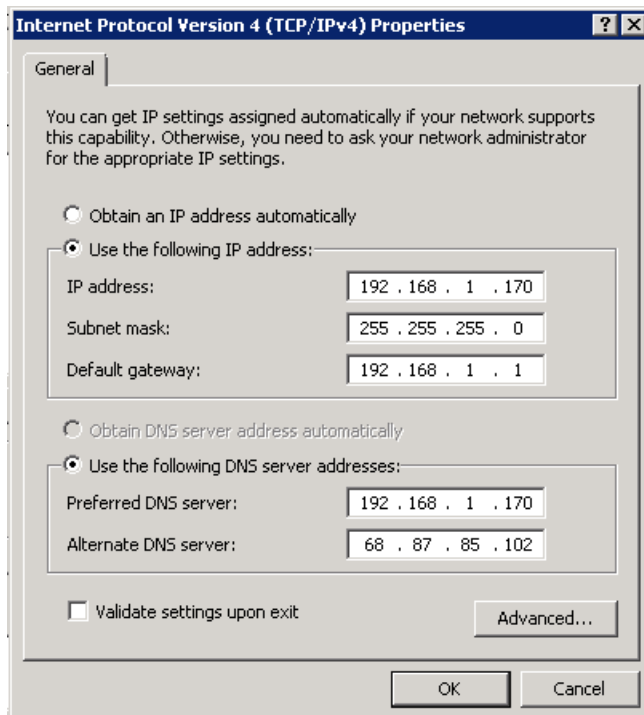
Set the TCP/IP settings for the External network adapter on the UAG server Virtual Machine as follows.



Set the TCP/IP settings for the Internal network adapter on the SharePoint Server Virtual Machine as follows.

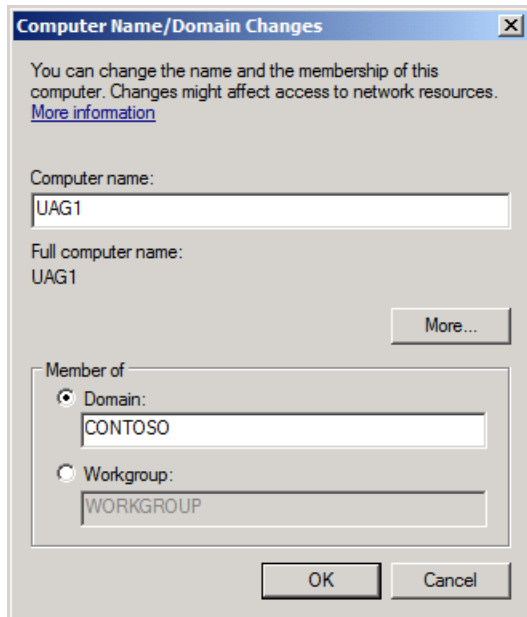


Set the TCP/IP settings for the External network adapter on the SharePoint Server Virtual Machine as follows.



4. Prepare the UAG Server Virtual Machine for UAG Installation

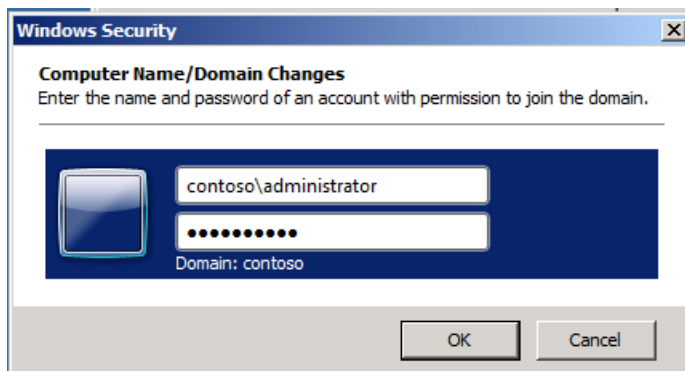
Log on to the UAG server Virtual Machine. Rename the machine **UAG1**, join it to the contoso.com domain, and reboot. Note: If you are not using the 2010 Information Worker Demonstration and Evaluation Virtual Machine (RTM) (see <http://go.microsoft.com/fwlink/?LinkID=189314>), join the UAG server to the same domain as the server running SharePoint Server. Make sure the 2010 Information Worker Demonstration and Evaluation Virtual Machine (RTM) is running.



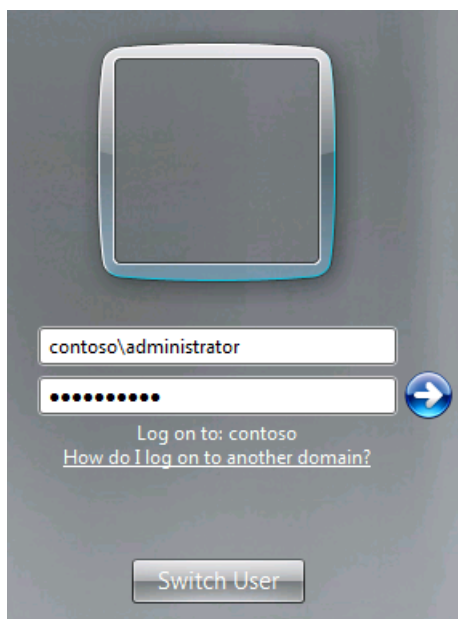
Use the following credentials to join the UAG server Virtual Machine to the contoso domain.

Username: contoso\administrator

Password: pass@word1

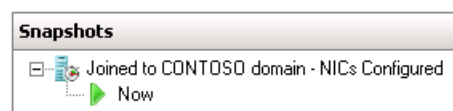


Reboot and log on by using the contoso\administrator credentials or the credentials specific to your environment.



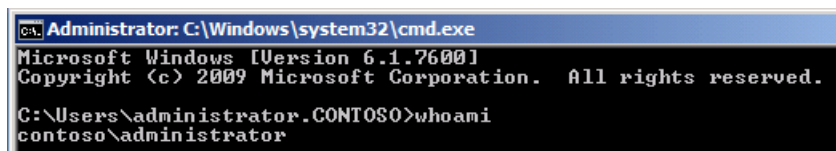
5. Snapshot the UAG Server Virtual Machine

Shut down the UAG server Virtual Machine and take a snapshot in the Hyper-V management console.

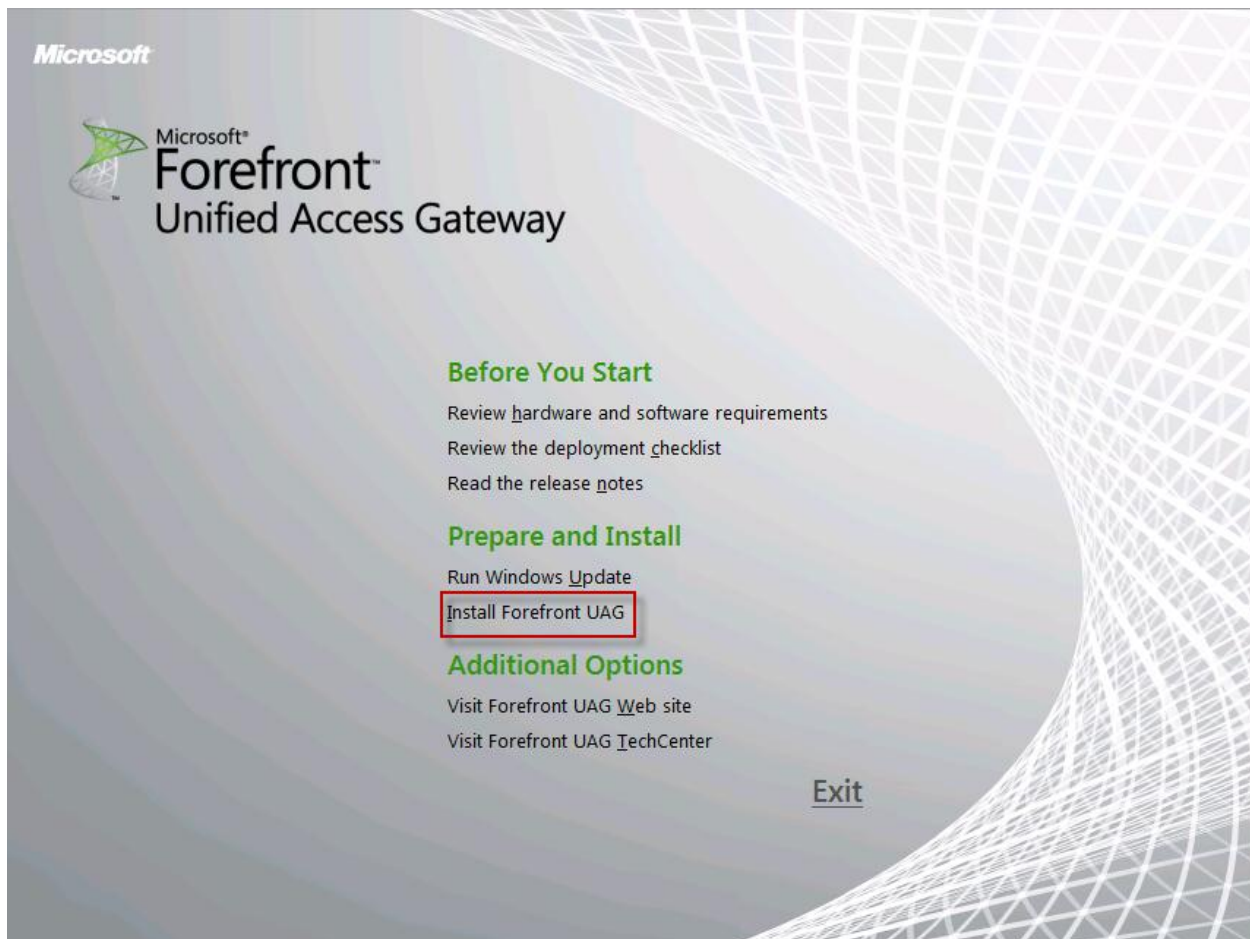


6. Install the UAG Server

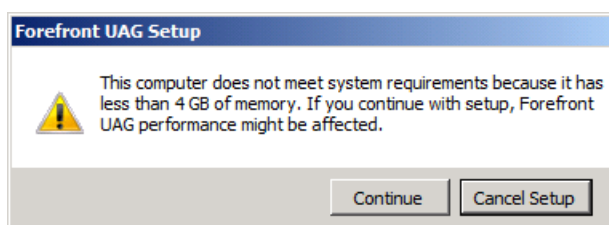
First, mount UAG installation media to the UAG server Virtual Machine. Then, start the UAG server Virtual Machine and log on by using the contoso\administrator credentials. After you have logged on, verify that you have logged on by using the contoso administrator credentials. Open a command prompt, type **whoami**, and then press ENTER to verify credentials.

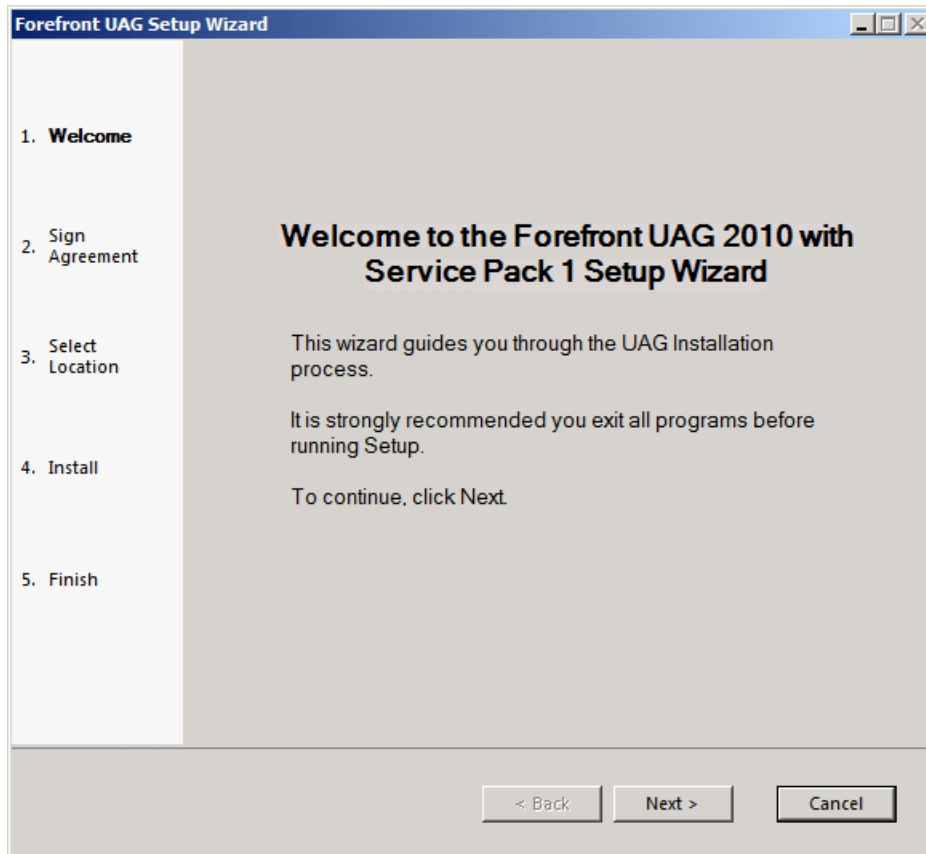


Next, start the UAG Install from installation media. See [Installing SP1 for Forefront UAG 2010](http://go.microsoft.com/fwlink/?LinkId=216130) (<http://go.microsoft.com/fwlink/?LinkId=216130>) on TechNet for additional installation information. (Note: you cannot install UAG during a remote session; you must install from the console.)

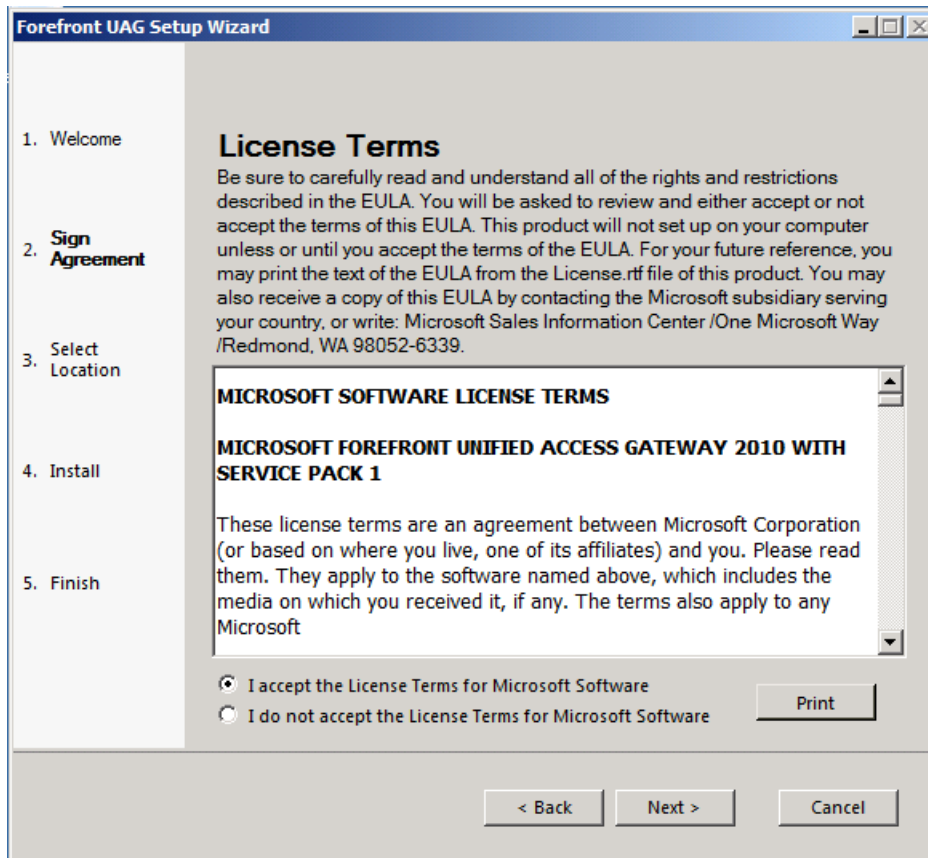


Click **Install Forefront UAG**. In a development scenario, you can run the UAG server Virtual Machine with less than 4 GB of RAM. If you have allocated less than 4 GB of RAM to the UAG server Virtual Machine, click **Continue**.

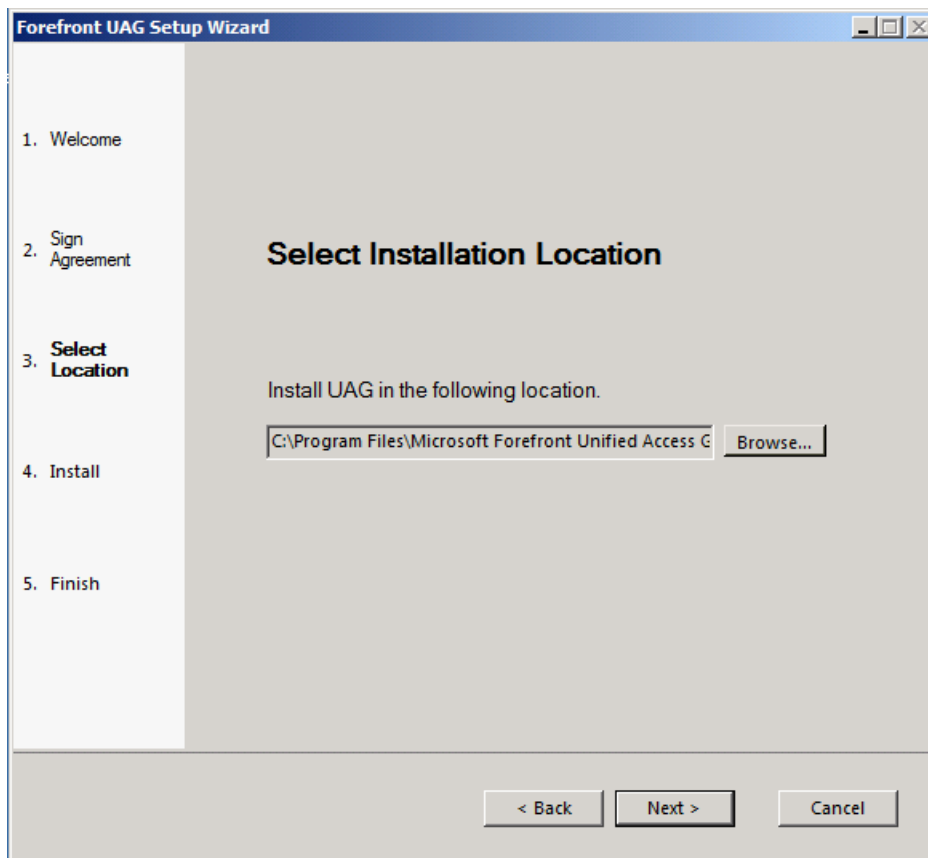




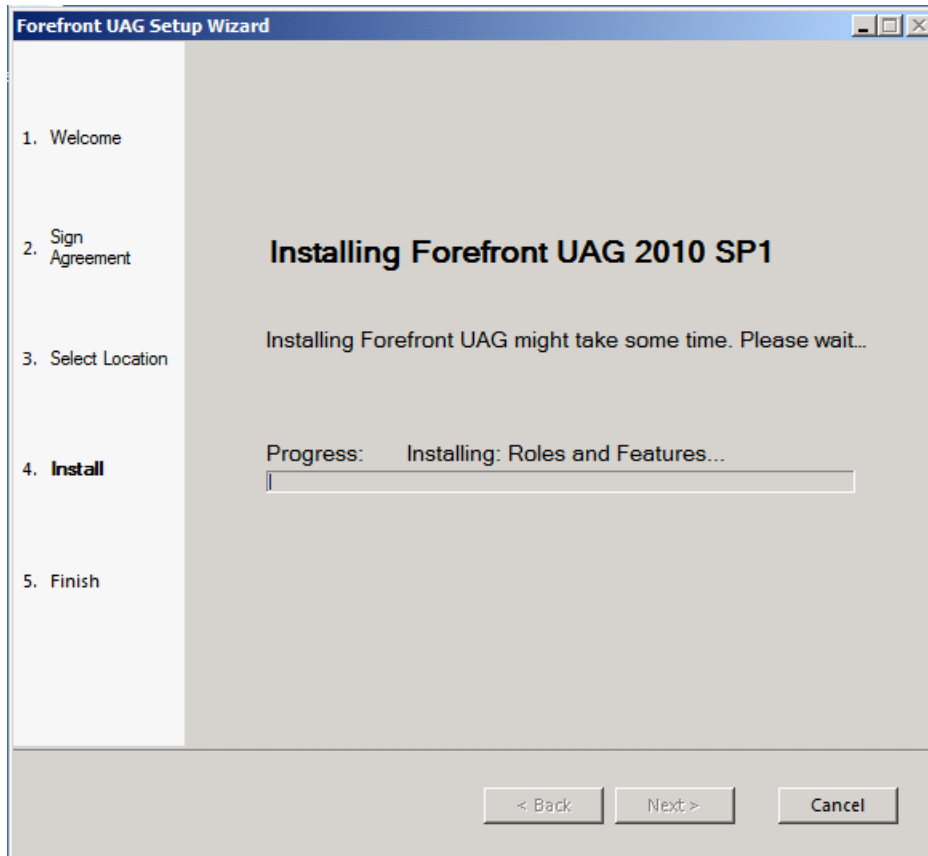
On the **Welcome** wizard page, click **Next >**.



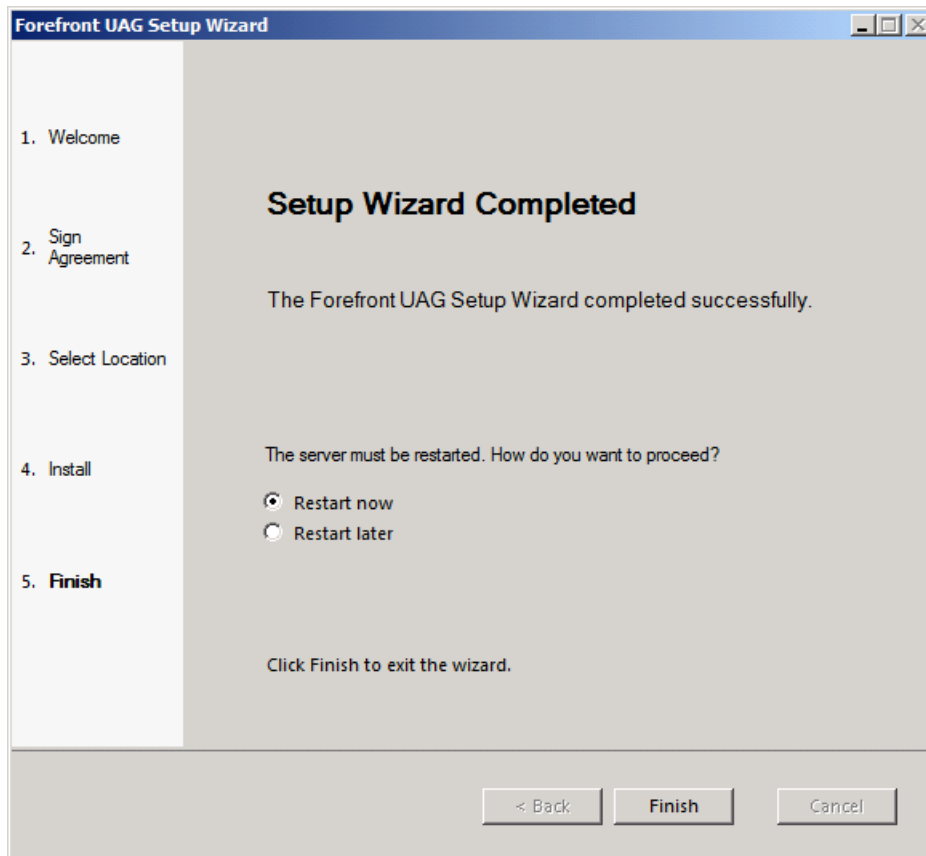
Click **I accept the Licensing Terms for Microsoft Software**, and then click **Next**.



On the **Select Installation Location** page, click **Next**.



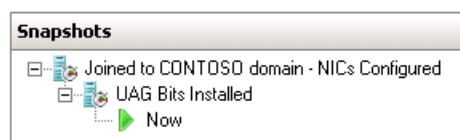
At this point, the installation begins. The installer installs all the Roles and Features that UAG depends on and configures the server for UAG.



On the **Setup Wizard Completed** page, click **Finish**.

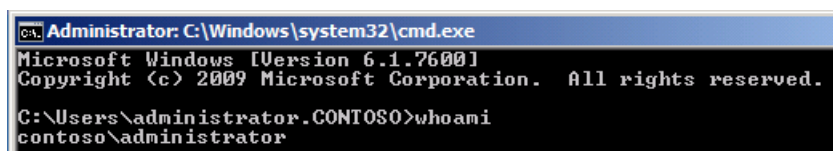
7. Snapshot the UAG Server Virtual Machine

After the UAG server Virtual Machine reboots and finishes the UAG installation, shut down the UAG server Virtual Machine and take a snapshot in the Hyper-V management console.



8. Initial UAG Server Configuration and Activation

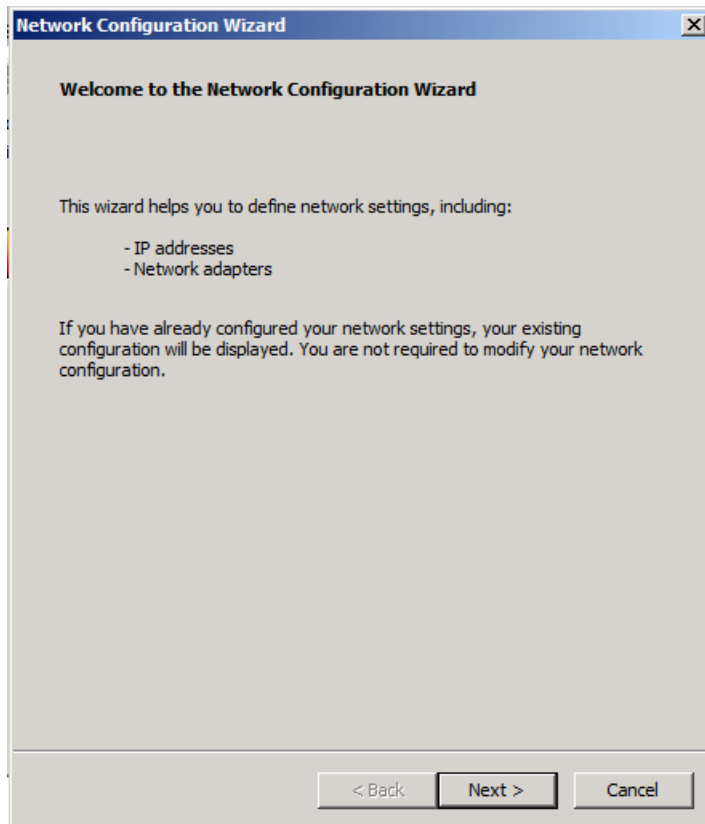
Start the UAG server Virtual Machine and log on by using the `contoso\administrator` credentials. After you have logged on, verify that you have logged on by using the `contoso` administrator credentials: Open a command prompt, type **whoami**, and then press ENTER to verify credentials.



To start the UAG configuration wizard, click **Start | All Programs | Microsoft Forefront UAG | Forefront UAG Management**.



Click **Configure Network Settings**.



Click **Next**.

Network Configuration Wizard [X]

Define Network Adapters

Specify how network adapters are connected to your network, and modify the network adapter settings.

Adapter name	Internal	External	Unassigned
External		✓	
Internal	✓		

Adapter properties:

< Back **Next >** Cancel

Choose the network adapter settings as seen in the screenshot above, and then click **Next**.

Network Configuration Wizard

Define Internal Network IP Address Range

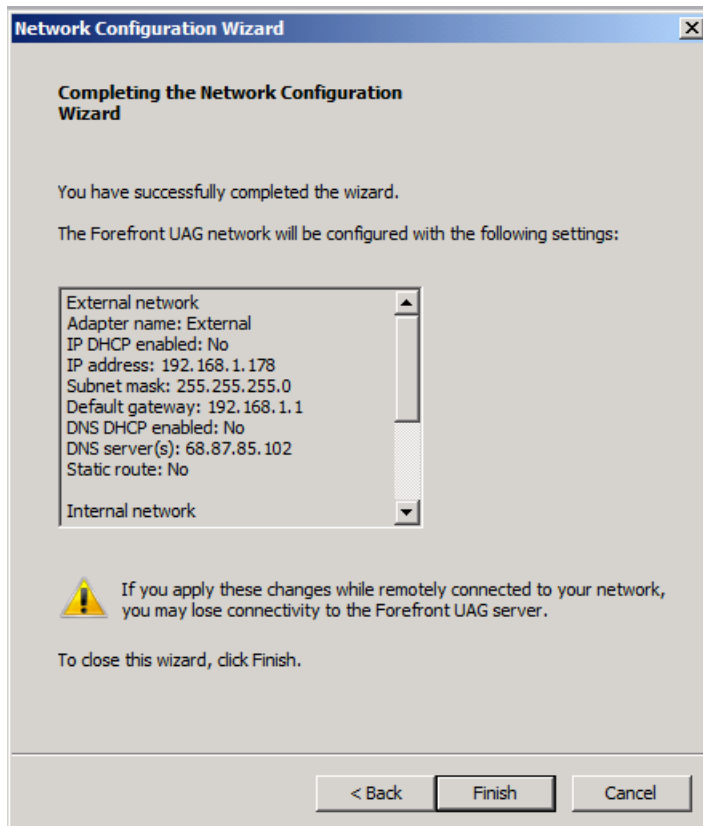
Specify internal network IP address range(s):

From IP	To IP
192.168.150.0	192.168.150.255

Add...
Edit...
Remove

< Back Next > Cancel

Click **Next**.

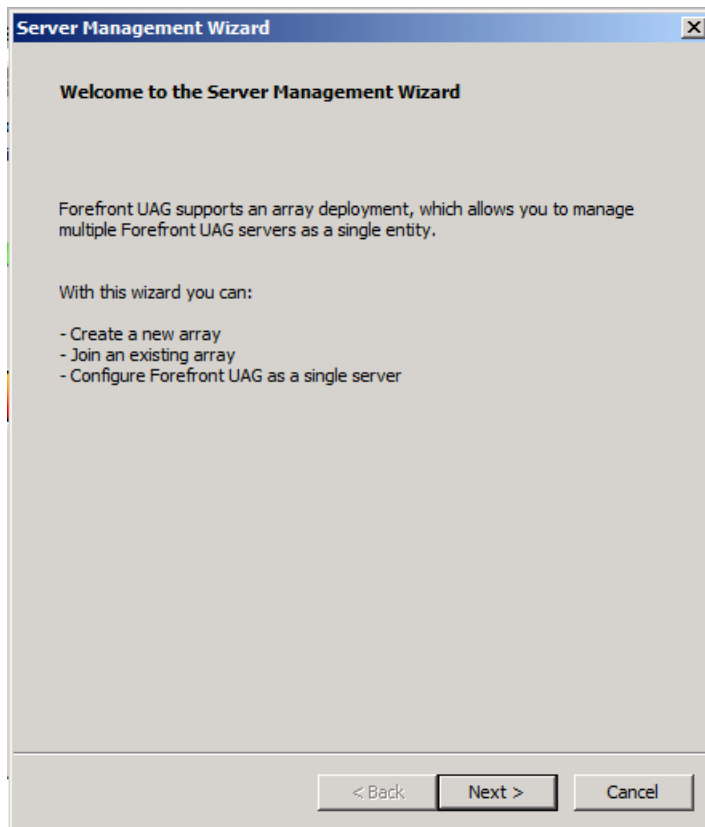


Click **Finish**.

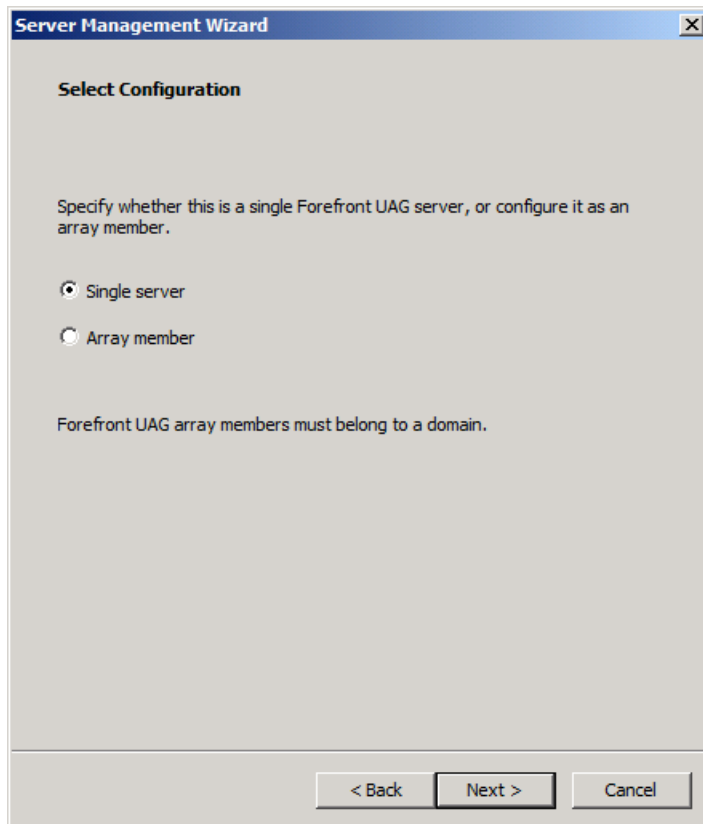


In this step, you define the UAG server as a "Single Server" configuration.

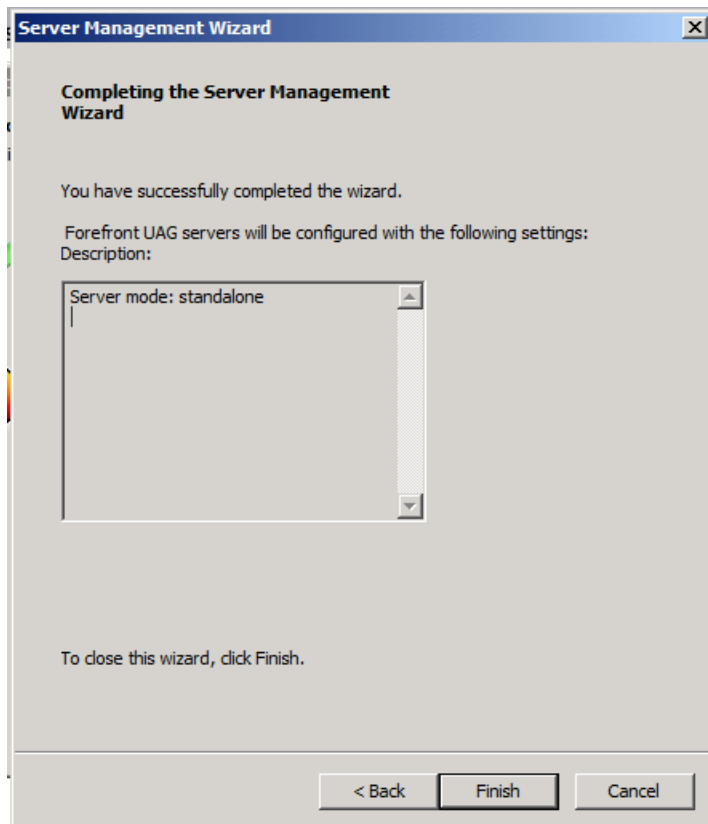
Click **Define Server Topology**.



Click **Next**.



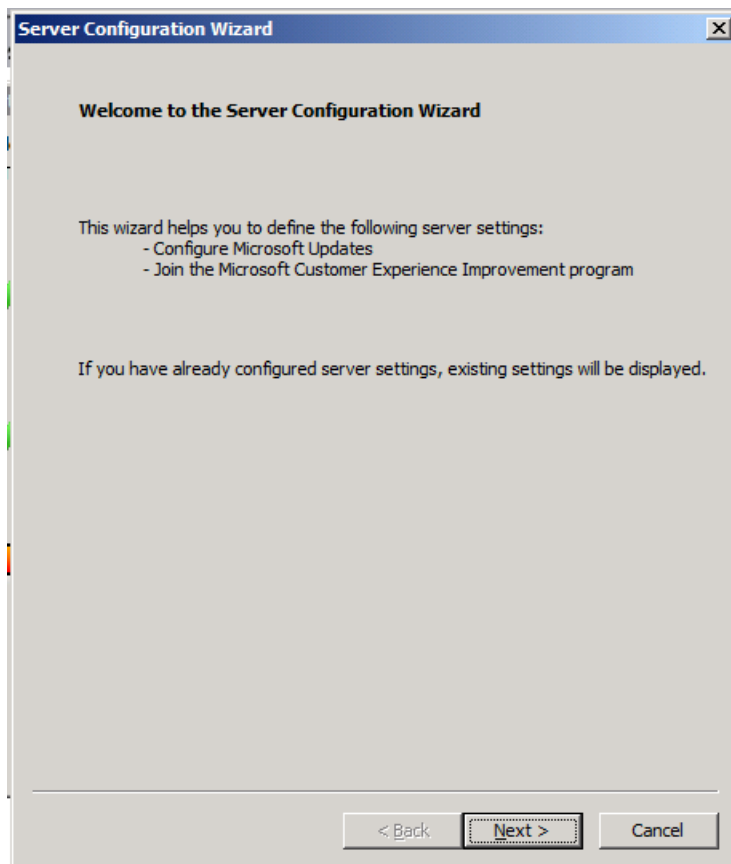
Click **Single Server**, and then click **Next**.



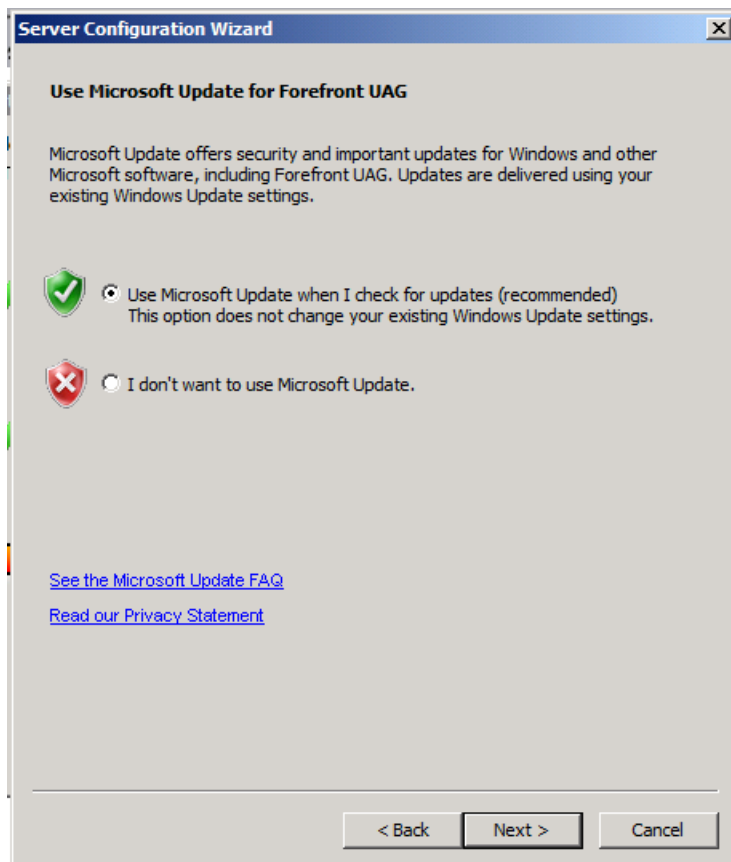
Click **Finish**.



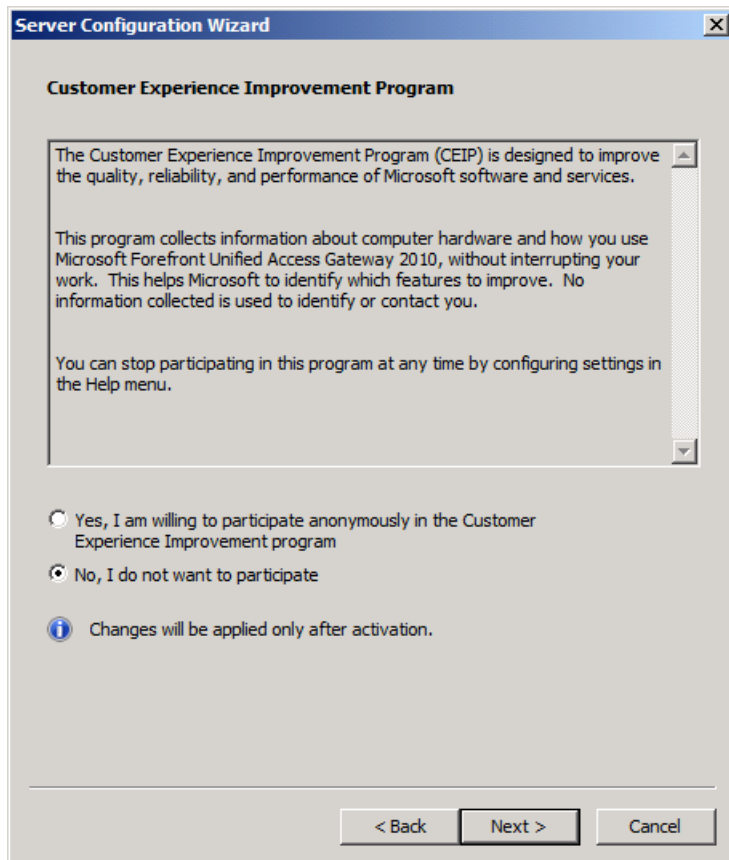
Click **Join Microsoft Update**.



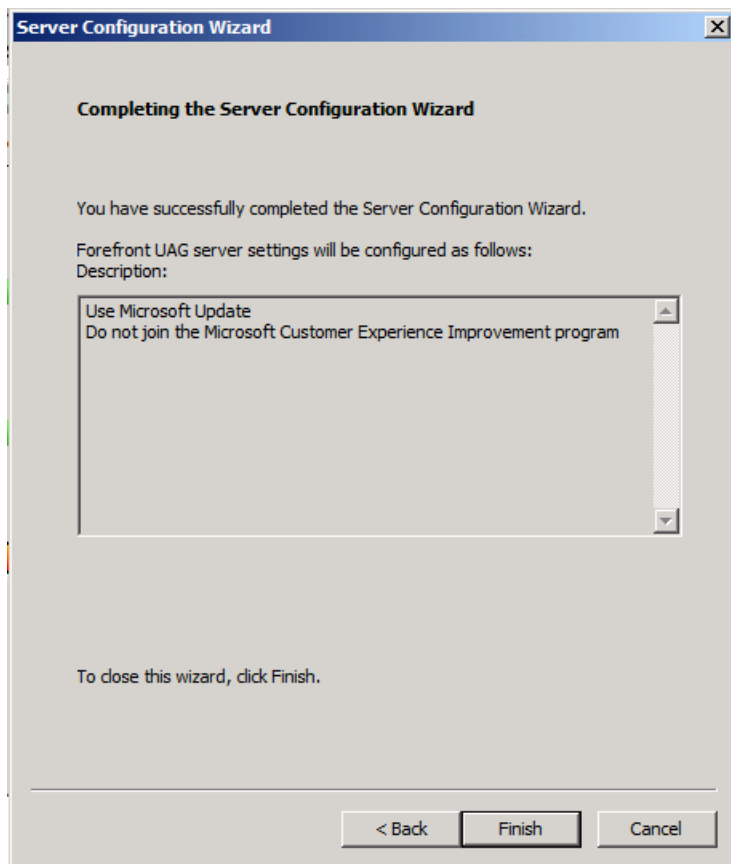
Click **Next**.



Click **Use Microsoft Update when I check for updates (recommended)**, and then click **Next**.



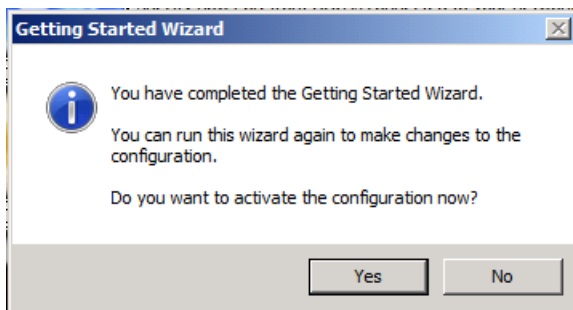
Choose **Yes** or **No**, and then click **Next**.



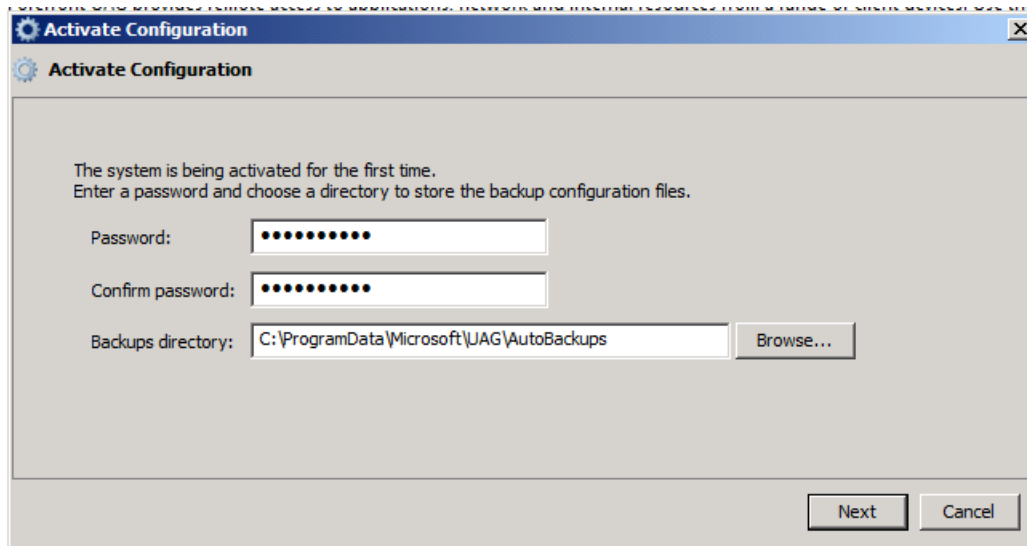
Click **Finish**.



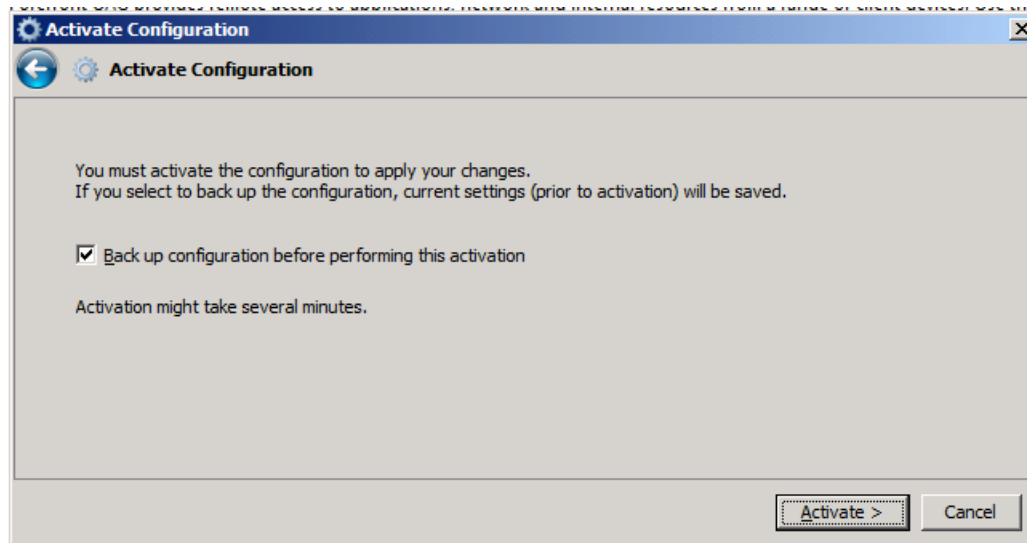
Click **Close**.



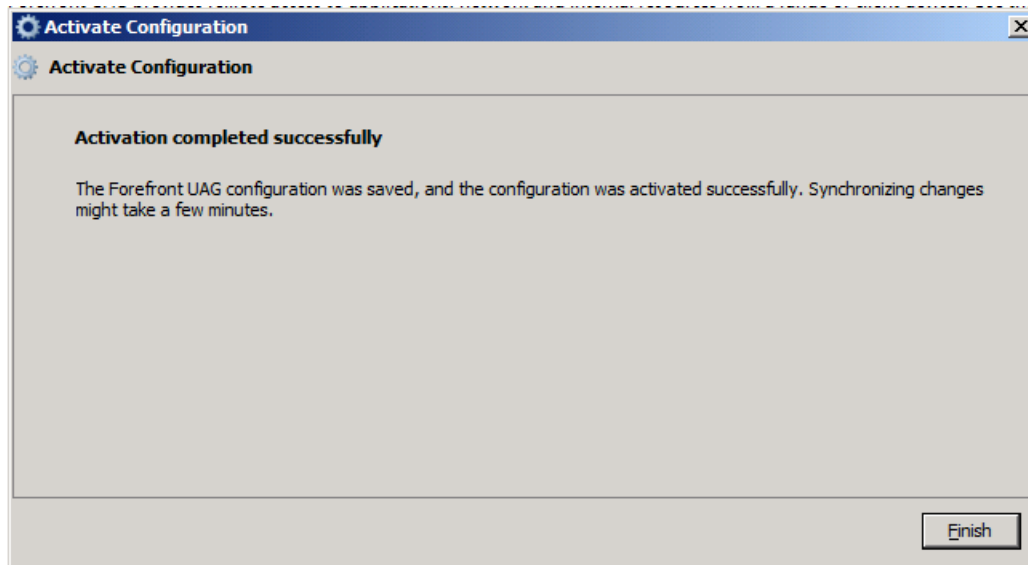
Click **Yes** to activate the configuration.



Enter a password, and then click **Next >**.



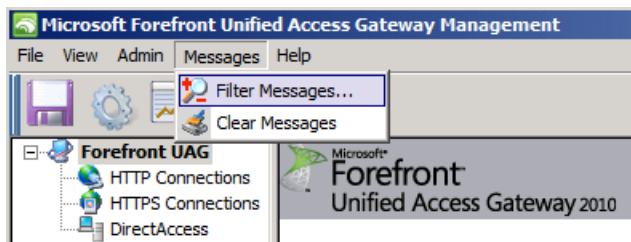
Click **Activate**.



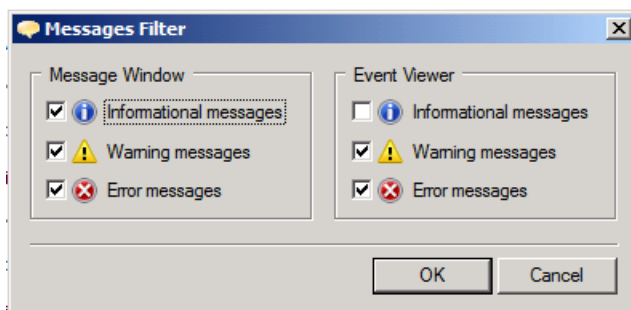
Click **Finish**.

The Activation Wizard is complete; however, UAG configuration may not be fully activated yet. To ensure it is, turn on the Informational messages in the Message Window.

On the **Messages** menu, select **Filter Messages**.



Select the **Informational messages** check box.



Click **OK**.

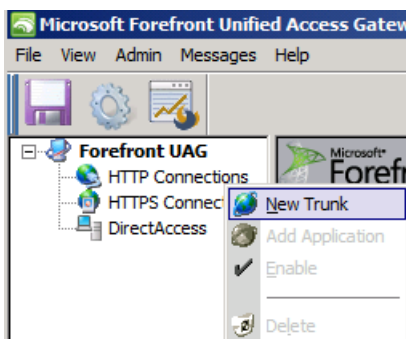
Wait until the Message Window displays the message **Activation completed successfully**.

Message Time	Message Type	Message
January 28, 2011 00:27:44	Information	TMG storage is synchronized.
January 28, 2011 00:27:44	Information	Activation completed successfully.

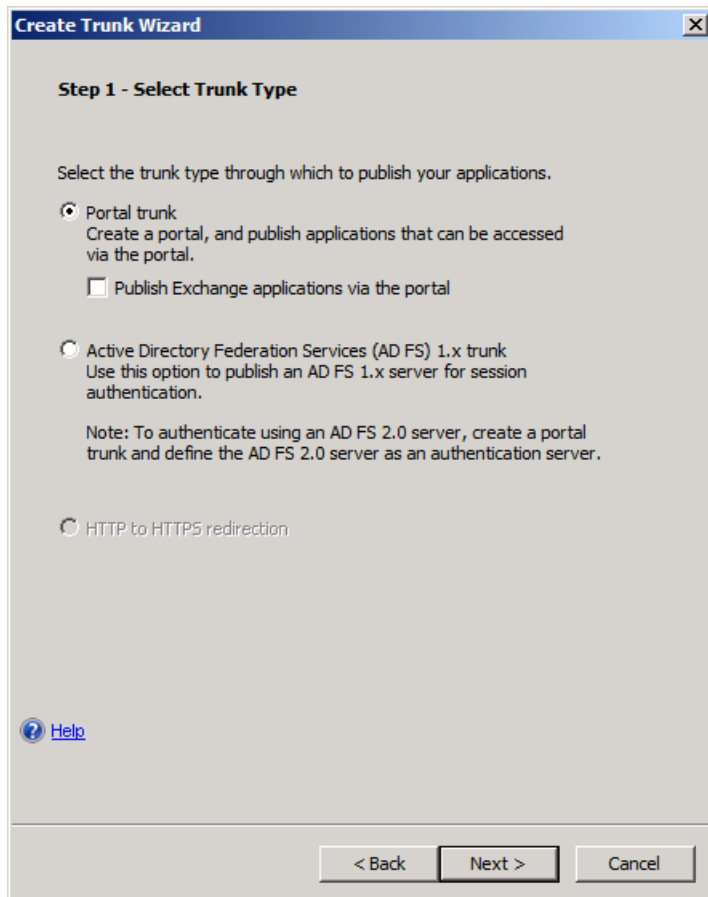
9. Create the HTTP Trunk to Publish the SharePoint Site

In this scenario, an HTTP trunk is used for development purposes. Using an HTTP trunk instead of an HTTPS trunk expedites the configuration by eliminating the steps involved with setting up security certificates to enable Secure Sockets Layer (SSL). In a development environment, it is also easier to use HTTP because it saves you from re-creating your temporary certificates every two weeks. In a testing environment and in a production environment, an HTTPS trunk is recommended to secure communications between mobile devices and the UAG server. See the [SharePoint publishing topologies](http://go.microsoft.com/fwlink/?LinkId=216131) (http://go.microsoft.com/fwlink/?LinkId=216131) on TechNet to learn about the commonly used topologies for deploying servers running SharePoint 2010 Products through UAG.

To create the HTTP trunk, right click **HTTP Connections**, and then click **New Trunk**.



Click Next.



In step 1 of the Create Trunk Wizard, click **Portal Trunk**, and then click **Next**.

Create Trunk Wizard

Step 2 - Setting the Trunk

Enter the details for your trunk.

Trunk name:

Public host name:

External Web Site

IP address:

HTTP port:

HTTPS port:

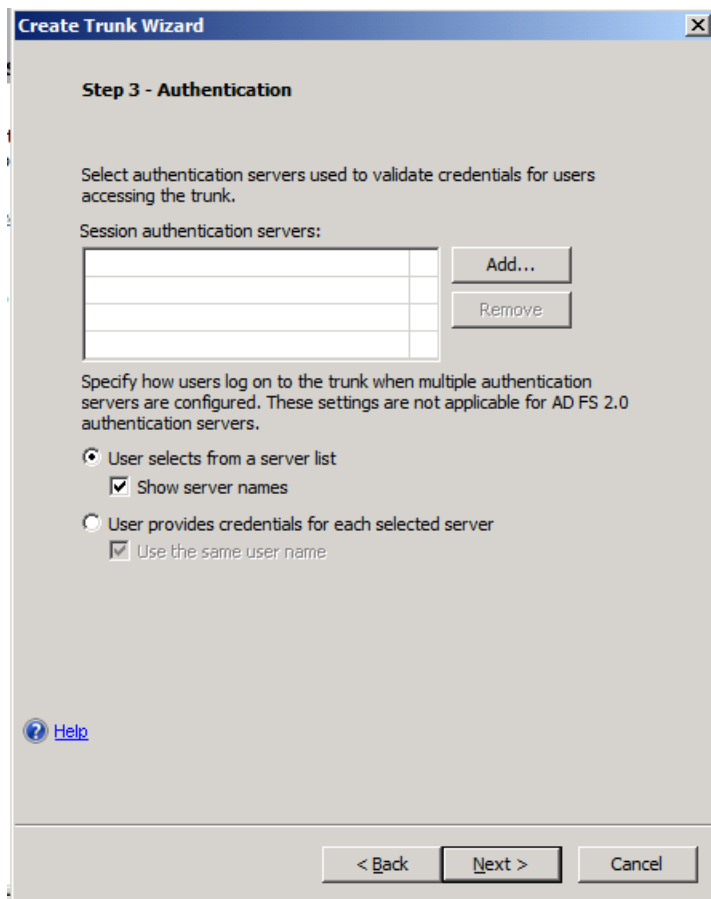
[? Help](#)

In step 2 of the Create Trunk Wizard, enter **ContosoMobile** in the **Trunk name** text box. This name is not a navigable URL; instead, it represents a friendly, identifiable name for the HTTP trunk.

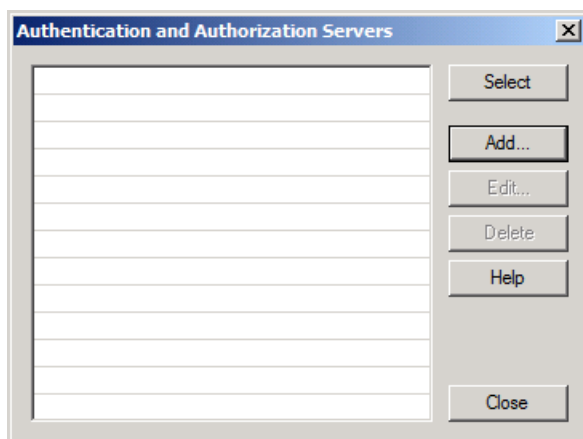
In the **Public Host Name** text box, enter **portal.contoso.com**. This value exposes an endpoint to the HTTP trunk that is used by UAG to create a single portal environment that publishes all of the applications in a single page — for example, our SharePoint site.

Select the IP address corresponding to the **external network IP** for the UAG server Virtual Machine.

Click **Next**.



In step 3 of the Create Trunk Wizard, click **Add** to open the dialog box that allows you to add the authentication server to the trunk.



Click **Add**.

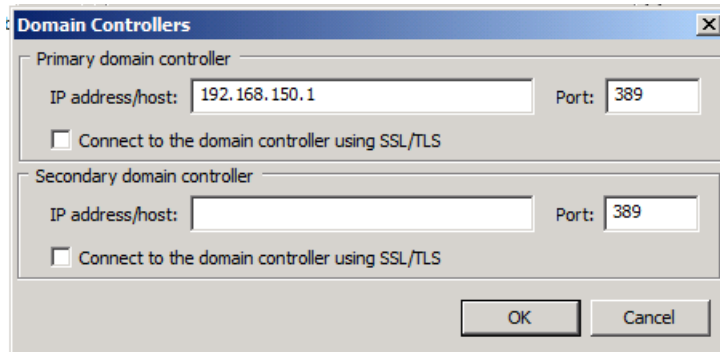
In the **Server type** drop-down list, select **Active Directory**.

In the **Server name** text box, enter **demo2010a.contoso.com**. This is the fully qualified domain name (FQDN) for the domain controller that is running on the SP2010-7a virtual machine.

In the **Connection settings** section, click **Define domain controllers**.

In the **Connection settings** section, click **Define**.

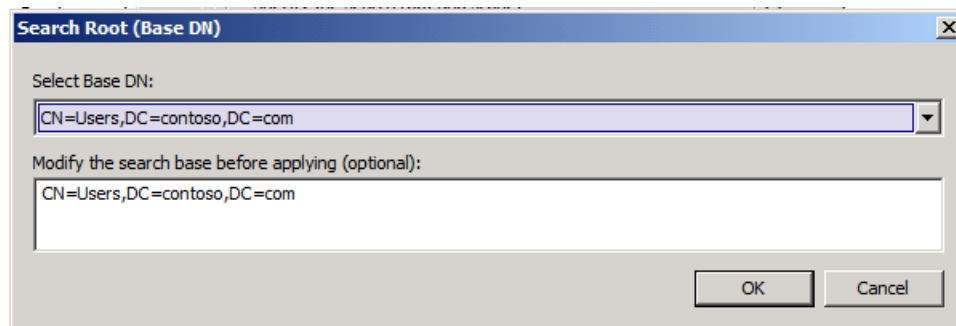
In the **Domain Controllers** dialog box, enter the internal IP address for the domain controller. In the SP2010-7a virtual machine, this IP address is **192.168.150.1**.

The 'Domain Controllers' dialog box has a title bar with a close button. It contains two sections: 'Primary domain controller' and 'Secondary domain controller'. The 'Primary' section has a text box for 'IP address/host' containing '192.168.150.1' and a 'Port' box containing '389'. Below it is a checkbox labeled 'Connect to the domain controller using SSL/TLS'. The 'Secondary' section has similar text boxes for 'IP address/host' and 'Port' (containing '389'), and another 'Connect to the domain controller using SSL/TLS' checkbox. At the bottom are 'OK' and 'Cancel' buttons.

Click **OK**.

In the **Search settings** section, click the ... button next to **Base DN**.

In the **Search Root** dialog box, in the **Select Base DN** drop-down list, select **CN=Users,DC=contoso,DC=com**.

The 'Search Root (Base DN)' dialog box has a title bar with a close button. It features a 'Select Base DN:' label above a drop-down list showing 'CN=Users,DC=contoso,DC=com'. Below this is a text box labeled 'Modify the search base before applying (optional):' containing the same DN string. At the bottom are 'OK' and 'Cancel' buttons.

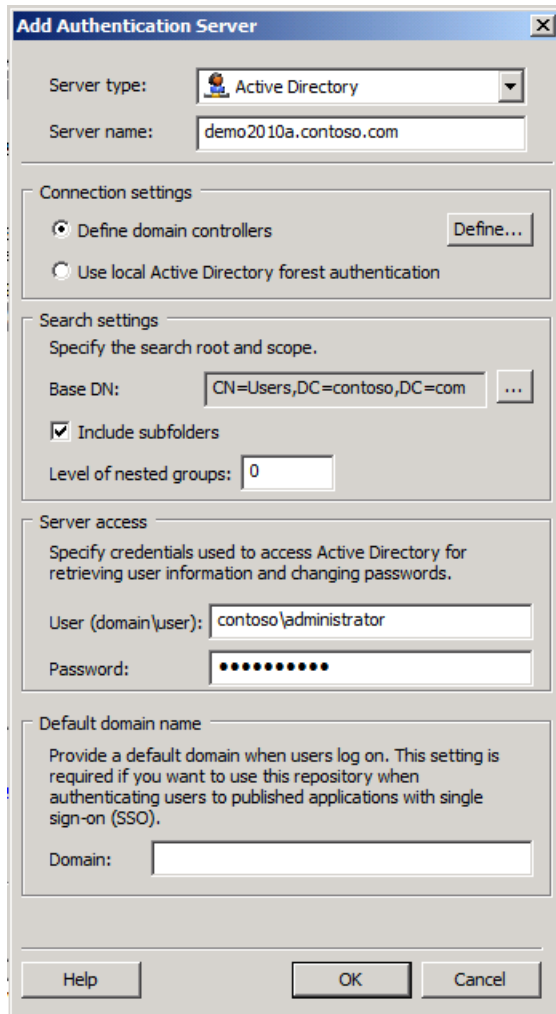
Click **OK**.

Select the **Include subfolders** check box.

Set the **Level of nested groups** equal to **0**.

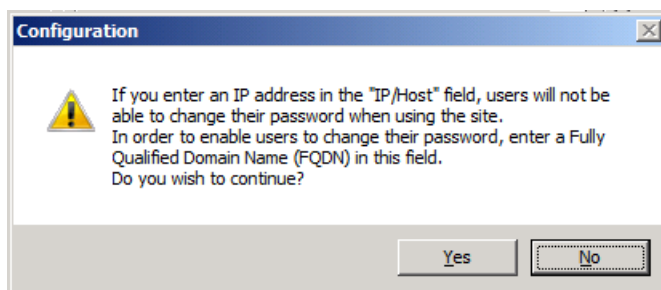
In the **Server access** section, in the **User (domain\user)** text box, enter **contoso\administrator**.

In the **password** text box, enter **pass@word1**.



The "Add Authentication Server" dialog box is shown. It has a title bar with a close button. The "Server type" dropdown is set to "Active Directory". The "Server name" text box contains "demo2010a.contoso.com". The "Connection settings" section has two radio buttons: "Define domain controllers" (selected) and "Use local Active Directory forest authentication". A "Define..." button is next to the first radio button. The "Search settings" section has a label "Specify the search root and scope." followed by a "Base DN:" text box containing "CN=Users,DC=contoso,DC=com" and a browse button "...". Below this is a checked checkbox "Include subfolders" and a "Level of nested groups:" text box with the value "0". The "Server access" section has a label "Specify credentials used to access Active Directory for retrieving user information and changing passwords." followed by a "User (domain\user):" text box containing "contoso\administrator" and a "Password:" text box with masked characters. The "Default domain name" section has a label "Provide a default domain when users log on. This setting is required if you want to use this repository when authenticating users to published applications with single sign-on (SSO)." followed by a "Domain:" text box. At the bottom are "Help", "OK", and "Cancel" buttons.

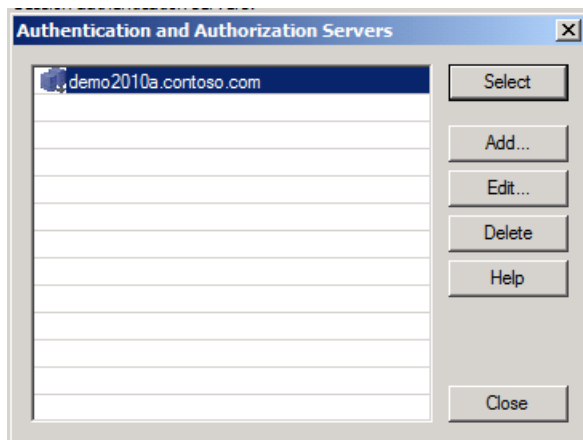
Click **OK**.



The "Configuration" dialog box is shown. It has a title bar with a close button. A yellow warning triangle icon is on the left. The text reads: "If you enter an IP address in the 'IP/Host' field, users will not be able to change their password when using the site. In order to enable users to change their password, enter a Fully Qualified Domain Name (FQDN) in this field. Do you wish to continue?". At the bottom are "Yes" and "No" buttons.

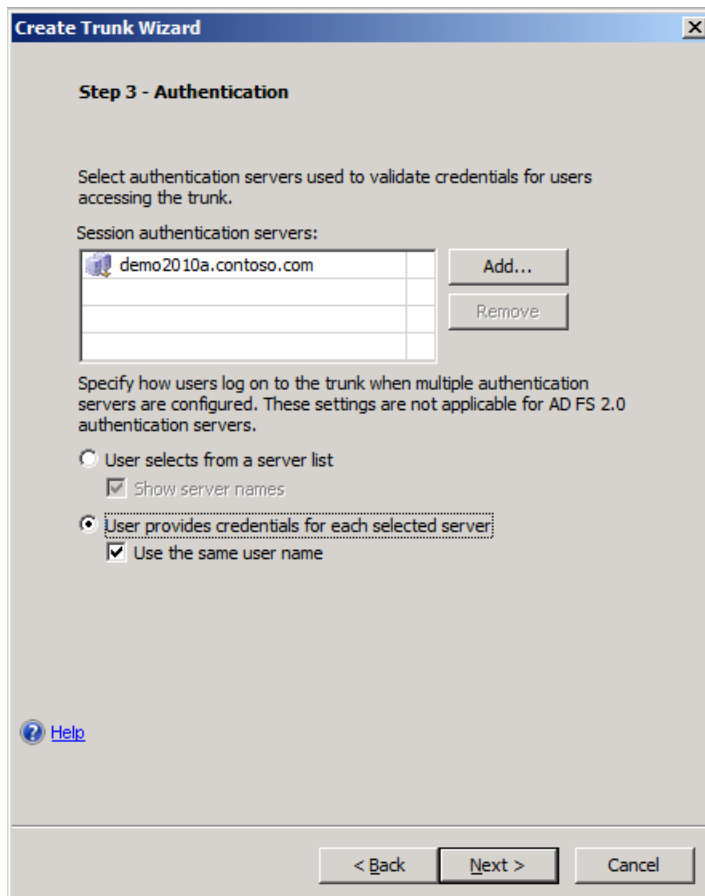
Click **Yes**.

Select the **demo2010a.contoso.com** server, and then click **Select**.

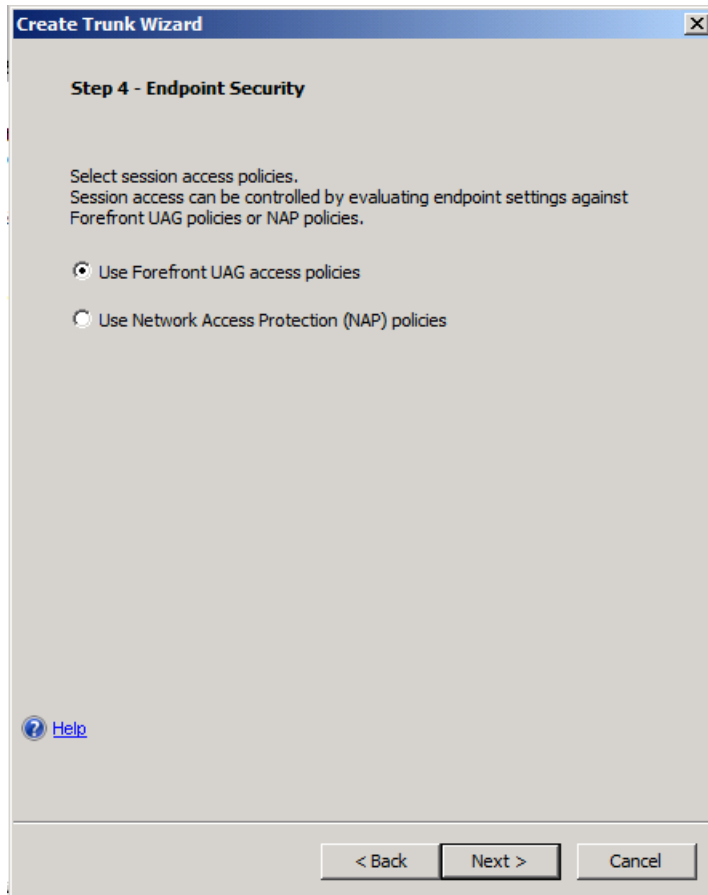


Click **User provides credentials for each selected serve**. (In a single server environment like this one, you might choose **User selects from a server list**.)

Select the **Use the same user name** check box.



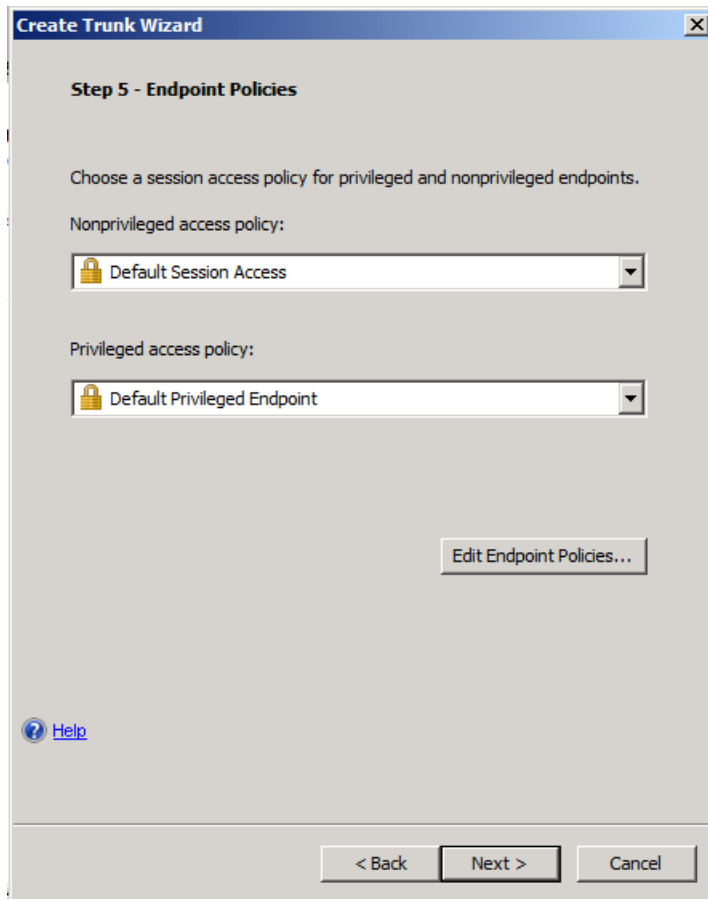
Click **Next**.



Click **Use Forefront UAG access policies**.

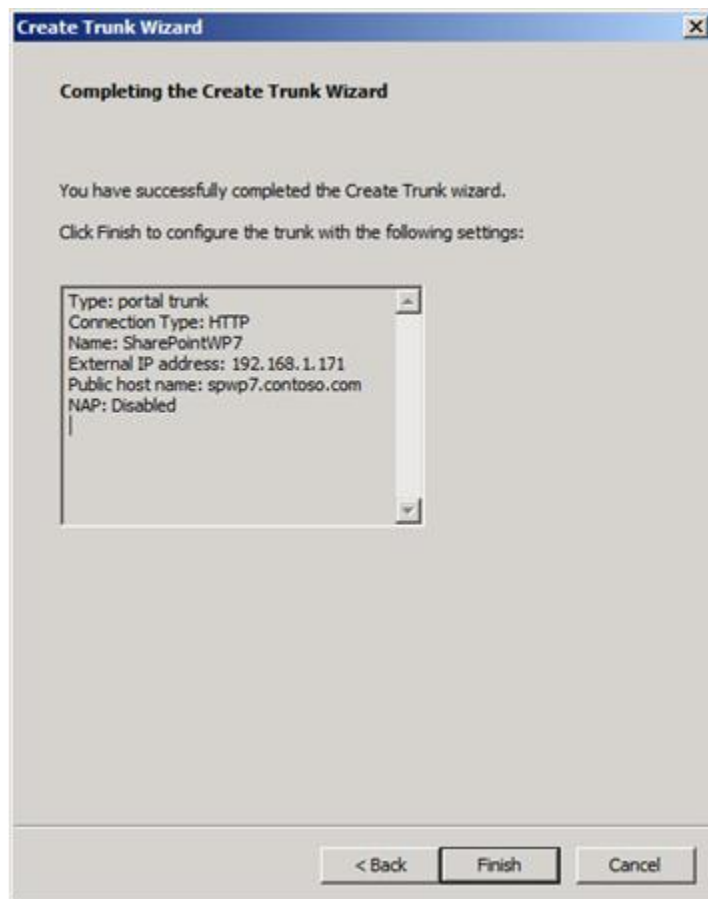
For development purposes, the default Forefront UAG access policies will suffice. In a production environment, you may adjust these policies to a configuration more specific to your environment.

Click **Next**.



For development purposes, the default Forefront endpoint policies will suffice. In a production environment, you may adjust these policies to a configuration more specific to your environment.

Click **Next**.



The final page in the Create Trunk Wizard displays a summary of the trunk configuration.

Click **Finish** to finalize the trunk configuration.

ContosoMobile

External Site Name

Specify the name that clients type in the browser to access the site.

Public host name: portal.contoso.com Port: 80

External Site Address

HTTP Port: 80

IP address: 192 . 168 . 1 . 171

Initial Internal Application

Portal home page: Portal

☒ Display home page within portal frame

Trunk Configuration

Configure trunk settings:

Configure...

Applications

Application Name	Application Type
Portal	Portal

Add...

Edit...

Remove

Limit applications to the following subnets:

Subnet Address	Subnet Mask

Add...

Edit...

Remove

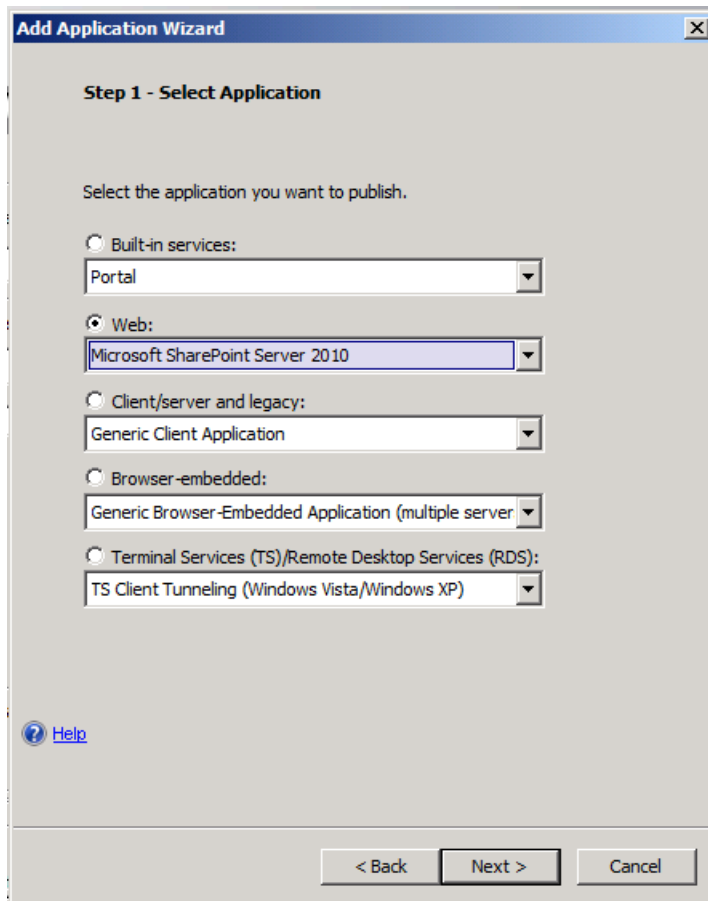
10. Create the SharePoint Application

In the Forefront UAG Management console Applications section, Click **Add**.

In the Add Application Wizard, click **Next**.

In step 2 of the Add Application Wizard, click **Web**.

In the Web drop-down list, select **Microsoft SharePoint Server 2010**.



The image shows a screenshot of the 'Add Application Wizard' window, specifically 'Step 1 - Select Application'. The window has a title bar with the text 'Add Application Wizard' and a close button. Below the title bar, the text 'Step 1 - Select Application' is displayed. The main area contains the instruction 'Select the application you want to publish.' followed by five radio button options, each with a corresponding dropdown menu. The 'Web:' option is selected. The dropdown menus show the following options: 'Portal' for Built-in services, 'Microsoft SharePoint Server 2010' for Web, 'Generic Client Application' for Client/server and legacy, 'Generic Browser-Embedded Application (multiple server)' for Browser-embedded, and 'TS Client Tunneling (Windows Vista/Windows XP)' for Terminal Services (TS)/Remote Desktop Services (RDS). At the bottom left, there is a 'Help' link with a question mark icon. At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'.

Add Application Wizard

Step 1 - Select Application

Select the application you want to publish.

☐ Built-in services:
Portal

☒ Web:
Microsoft SharePoint Server 2010

☐ Client/server and legacy:
Generic Client Application

☐ Browser-embedded:
Generic Browser-Embedded Application (multiple server)

☐ Terminal Services (TS)/Remote Desktop Services (RDS):
TS Client Tunneling (Windows Vista/Windows XP)

[Help](#)

< Back Next > Cancel

Click **Next >**.

In the Application name text box, enter **Contoso Intranet**.

This is a friendly name for the application you are publishing with the UAG server. This name will appear on the UAG Portal home page.


Add Application Wizard [X]

Step 2 - Configure Application

Enter an application name. If you are publishing a generic application, specify the application type.

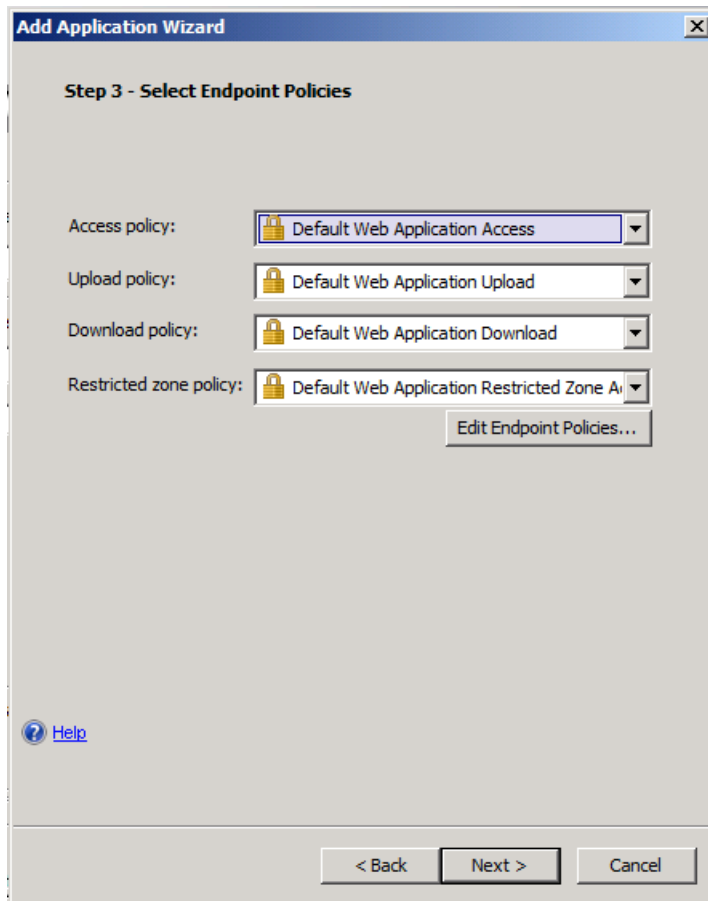
Application name:

Application type:

 [Help](#)

< Back Next > Cancel

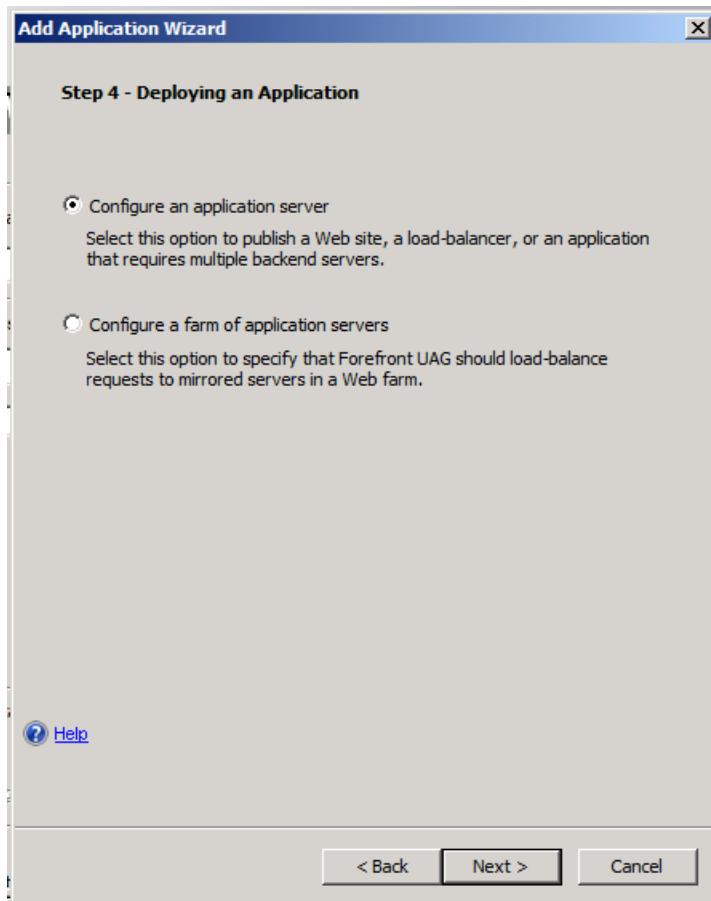
Click **Next**.



Click **Next**.

For development purposes, the default Forefront endpoint policies will suffice. In a production environment, you may adjust these policies to a configuration more specific to your environment.

Click **Configure an application server**.



Click **Next**.

Leave **Address type** as **IP/Host**.

In the **Addresses** list, enter the host name or internal IP address of the server running SharePoint Server 2010 **192.168.150.1**. Depending on your environment, you may also choose to use the FQDN for the server running SharePoint Server or the IP address for load balancing hardware.

In the **Paths** list, leave the default **"/"** entry. This indicates the published SharePoint application is published at the root level.

Click **Http port**, and enter **80** in the box.

As previously mentioned, using a HTTP port instead of an HTTPS port simplifies development setup and configuration. In a production environment, an HTTPS port should be used to encrypt the credentials passed between client machines and the UAG server.

In the **Public host name** text box, enter **spwp7intranet**. This is the alternate access mapping URL on the server running SharePoint that corresponds to the site collection you are publishing.

This value is the URL that the UAG server will use to publish the application to client machines. When accessing the intranet.contoso.com site collection through the UAG server clients, use the URL <http://spwp7intranet.contoso.com>. The UAG server then routes the request to SharePoint as <http://spwp7intranet.contoso.com> after the user is authenticated. SharePoint alternate access mapping (to be configured in later steps) will map the request to <http://intranet.contoso.com>.

Add Application Wizard

Step 5 - Web Servers

Address type: ☒ IP/Host ☐ Subnet ☐ Regular expression

Addresses:

Paths:

☒ HTTP port: ☐ HTTPS port:

Public host name: .contoso.com

☐ Replace the host header with the following

To allow access to information protected by Information Rights Management, publish the Rights Management Services server via Forefront UAG.

[Help](#)

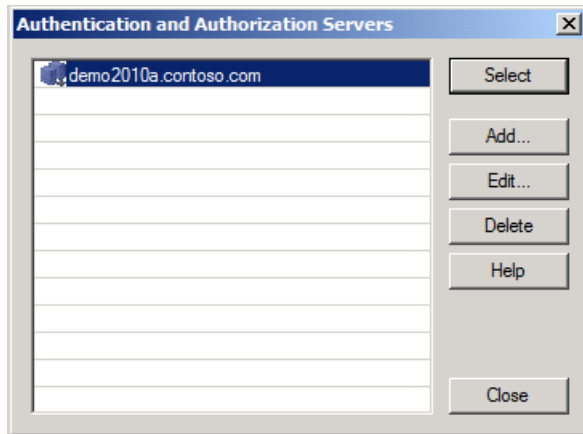
< Back Next > Cancel

Click **Next**.

Select the **Use SSO** check box.

Click **Add**.

In the **Authentication and Authorization Server** dialog box, select **demo2010a.contoso.com**.

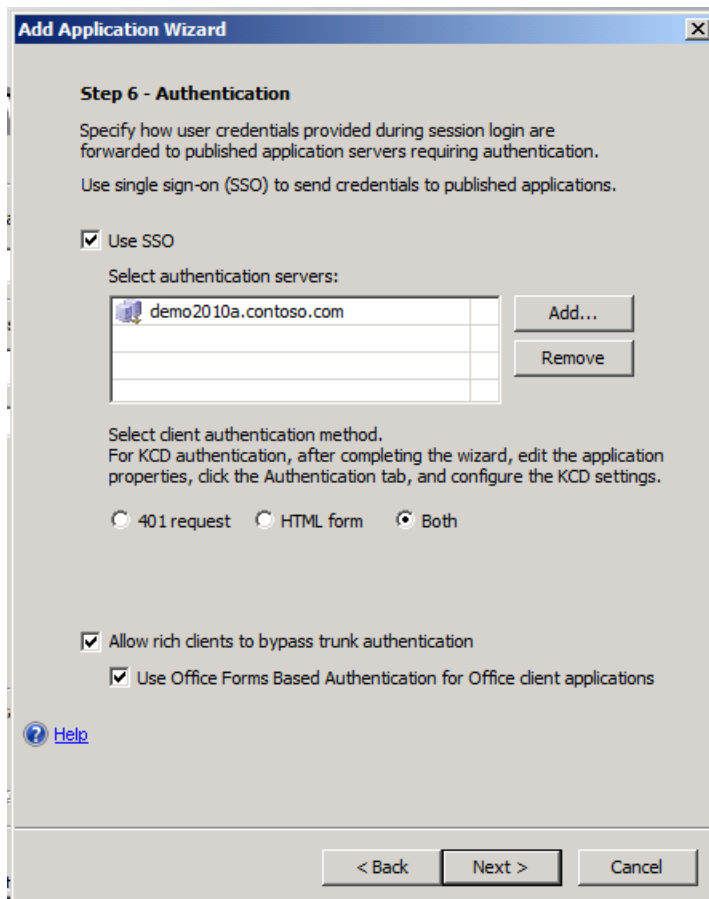


Click **Select**.

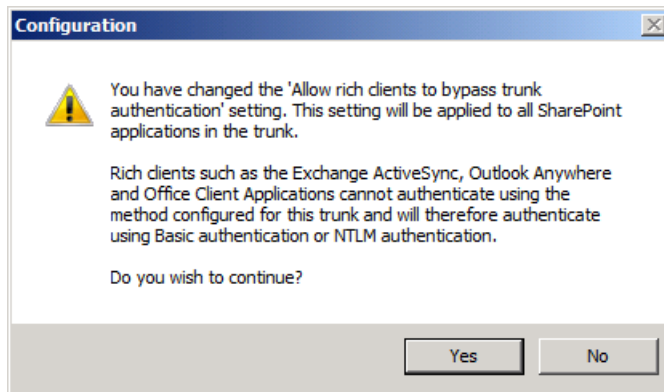
In the **Select client authentication method** section, click **Both**.

Select the **Allow rich clients to bypass trunk authentication** check box.

Select the **Use Office Forms Based Authentication for Office client applications** check box.



Click **Next**.



Click **Yes**.

In the **Portal Link** dialog box, select the **Open in a new window** check box. The portal link is optional because the Windows Phone 7 application directly accesses the SharePoint APIs, but the link is a good troubleshooting tool.

Add Application Wizard

Step 7 - Portal Link

☒ Add a portal and toolbar link

Portal name:

Folder:

Application URL:

Icon URL:

Short description:

Description:

☒ Open in a new window

[? Help](#)

Click **Next**.

Select the **Authorize all users** check box.

In a development environment, granting all users access to the published SharePoint site collection provides maximum flexibility for testing purposes. In a production environment, only grant access to the specific users and groups that are allowed to access the SharePoint site collection.

Add Application Wizard [X]


Step 8 - Authorization

Specify users and groups that can view and access the application via the portal.

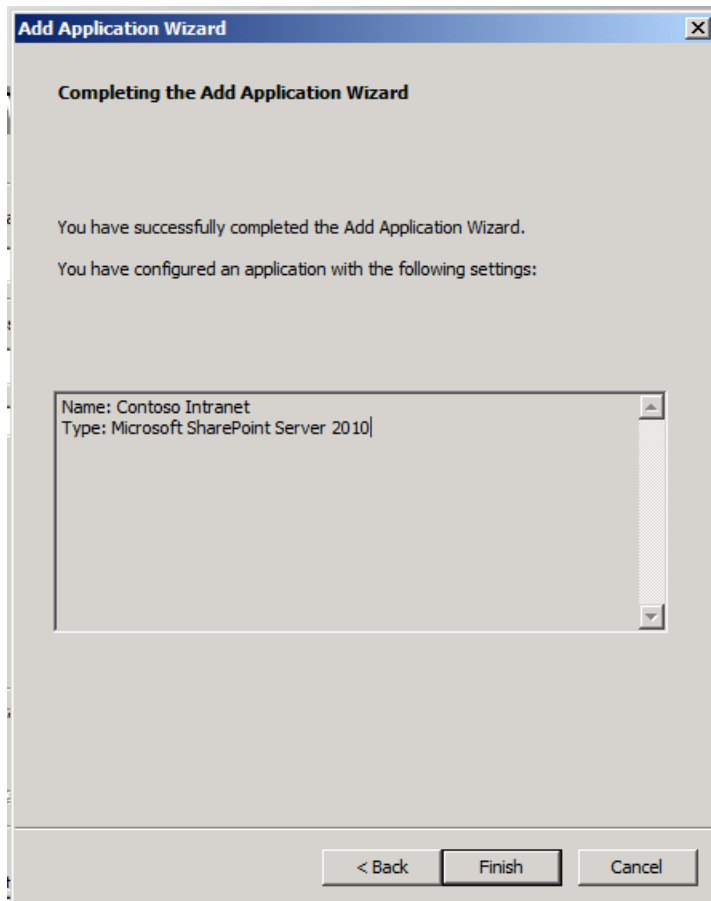
☒ Authorize all users

Users and groups:

Name	Allow	View	Deny

 [Help](#)

Click **Next**.







Click **Finish**.

11. Configure the SharePoint Application

In the Forefront UAG Management console, in the **Applications** section, select **Contoso Intranet**, and then click **Edit**.

Applications

Application Name	Application Type
 Portal	Portal
 Contoso Intranet	Microsoft SharePoint Server...

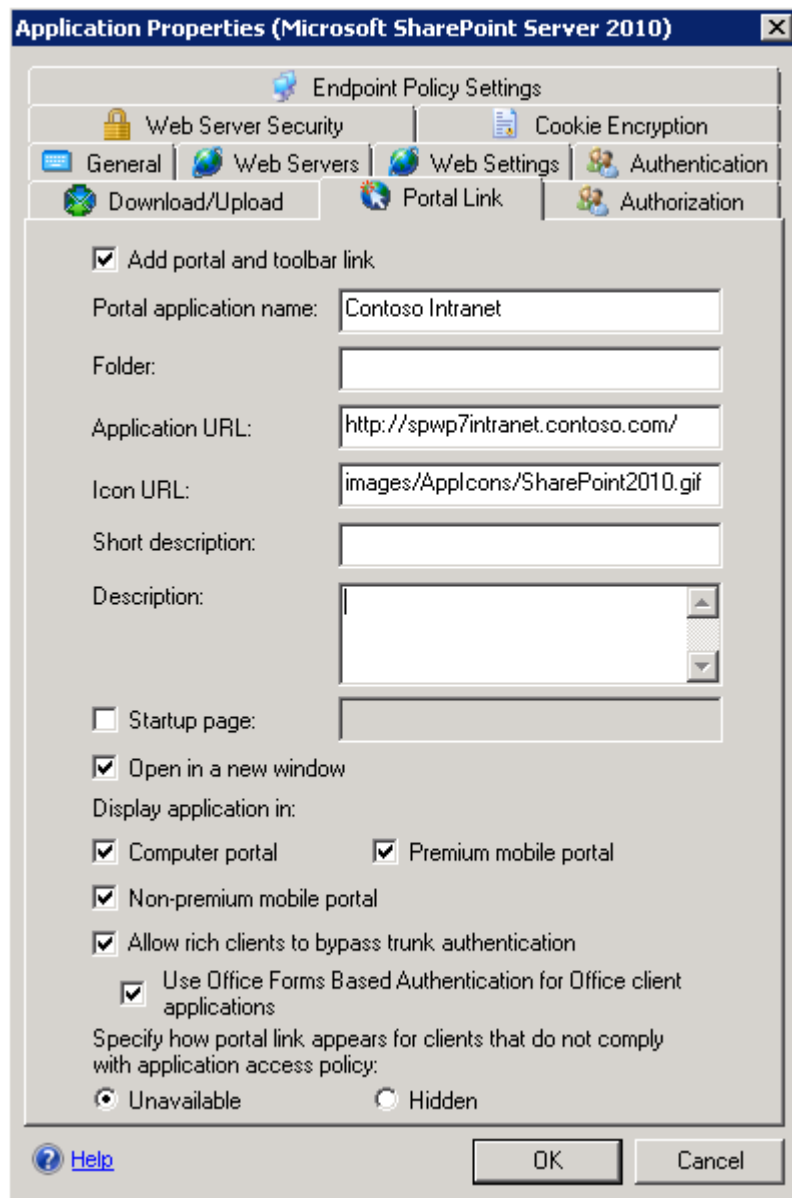
Add... Edit... Remove

Limit applications to the following subnets:

Subnet Address	Subnet Mask

Add... Edit... Remove

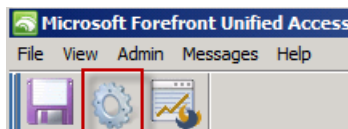
On the **Portal Link** tab, select the following check boxes: **Computer portal**, **Premium mobile portal** and **Non-premium mobile portal**.



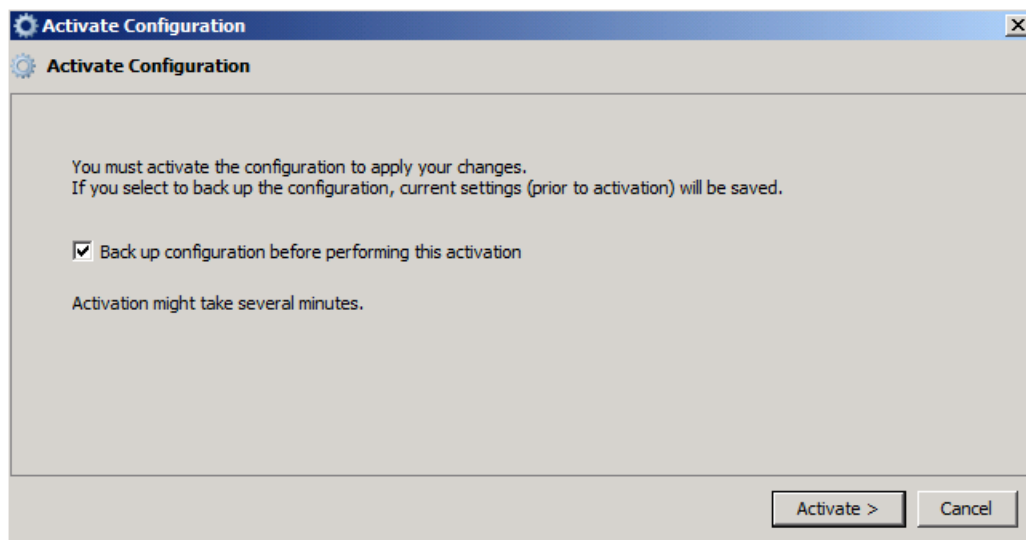
Click **OK**.

12. Activate the Configuration

In the Forefront UAG Management console, in the toolbar, click the **Activate** button.

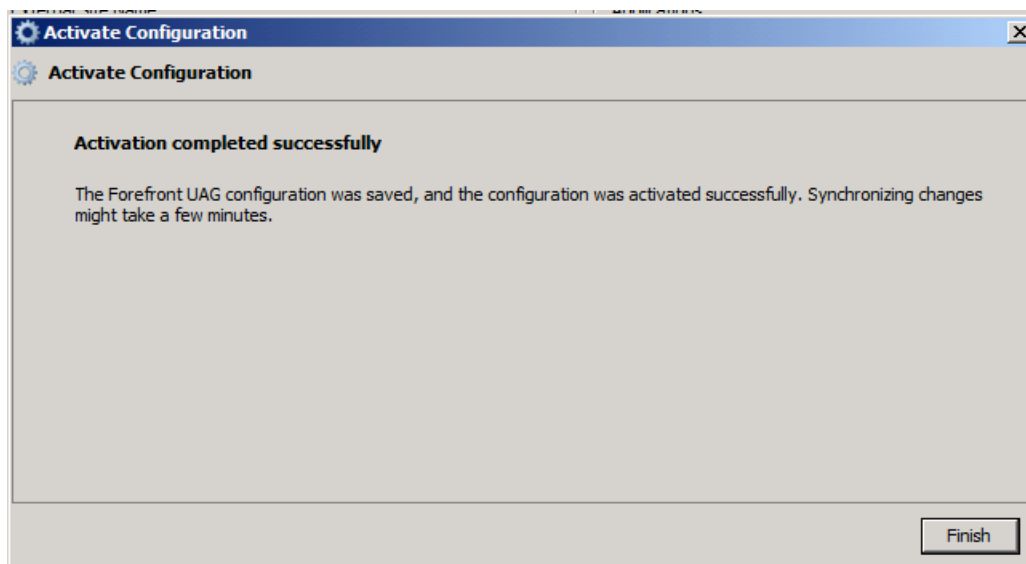


The Activate Configuration dialog box appears.





Click **Activate**.

The new configuration is activated on the UAG server.



Click **Finish**.

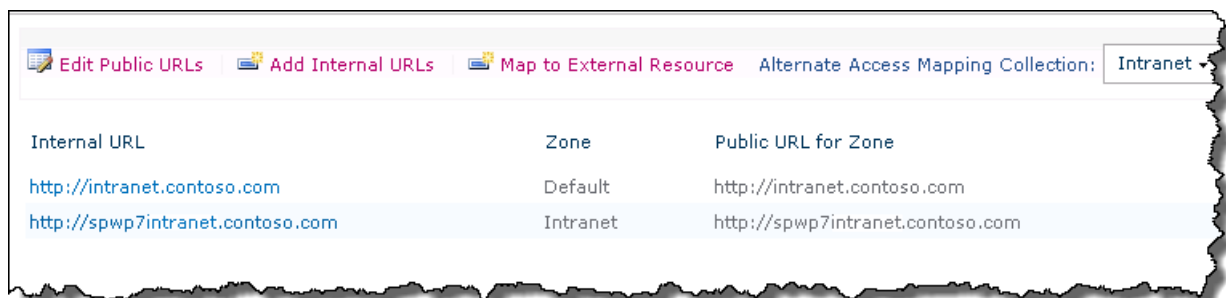
Wait until the Message Window at the bottom of the Microsoft Forefront UAG Administration Console displays the message **Activation completed successfully**. Although the Activation Configuration dialog box indicated the configuration was activated successfully, it is really not completed until you see the message in the Message Window.

	January 28, 2011 10:51:50	Information	TMG storage is synchronized.
	January 28, 2011 10:51:50	Information	Activation completed successfully.

13. Configure and Verify SharePoint Alternate Access Mappings

SharePoint must be configured to respond to requests for `http://spwp7intranet.contoso.com` and map the requests to `http://intranet.contoso.com`.

1. In SharePoint 2010 Central Administration, navigate to the **Application Management** section and choose **Web Application Management**.
2. Select **Intranet**, and choose **Extend** (Extend Web Application) from the ribbon.
3. In the **Extend Web Application to Another IIS Web Site** dialog box, enter the following settings:
 - Name: SharePoint - spwp7intranet.contoso.com – 80
 - Port: 80
 - Host Header: spwp7intranet.contoso.com
 - Zone: Intranet
4. Click **OK**.
5. Confirm the settings by navigating to **Application Management | Web Applications | Configure alternate access mappings**.
6. Choose the **Alternate Access Mapping Collection** for the Intranet and you should see the new entry for `http://spwp7intranet.com`



Internal URL	Zone	Public URL for Zone
http://intranet.contoso.com	Default	http://intranet.contoso.com
http://spwp7intranet.contoso.com	Intranet	http://spwp7intranet.contoso.com

14. Add Hosts File Entries to the Development Environment

In this scenario, hosts file entries are used to resolve the UAG server and the server running SharePoint. DNS entries could also be used, but hosts files are an acceptable workaround in a development environment. In a production environment, DNS entries should be used for name resolution.

On the machine where you are running the Windows Phone 7 emulator, add an entry to the hosts file to point to the public host name and IP address associated with the Contoso intranet portal application on the UAG server. The hosts file is located in `c:\windows\system32\drivers\etc`.

```
192.168.1.171 spwp7intranet.contoso.com
```

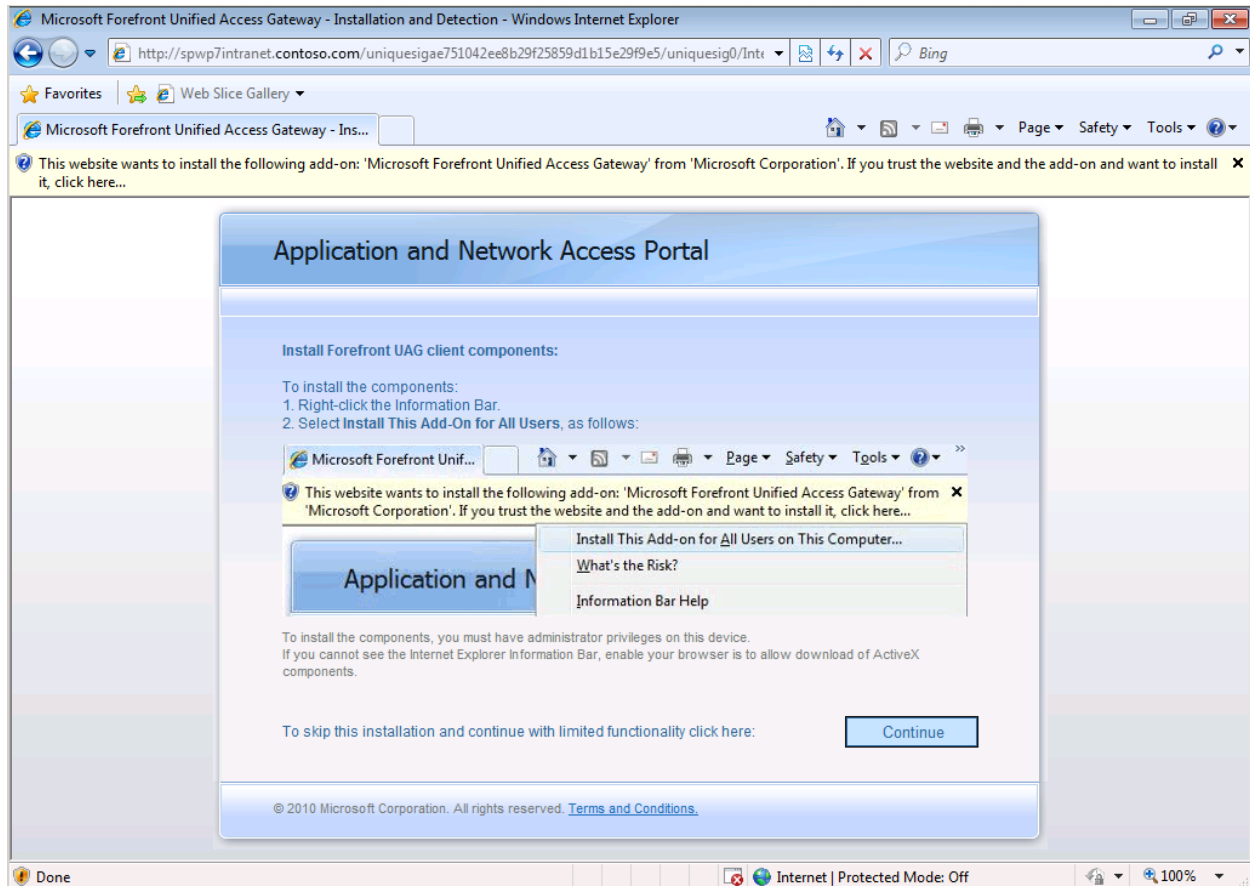
Optionally, you can add an entry to the UAG Portal Home page.

```
192.168.1.171 portal.contoso.com
```

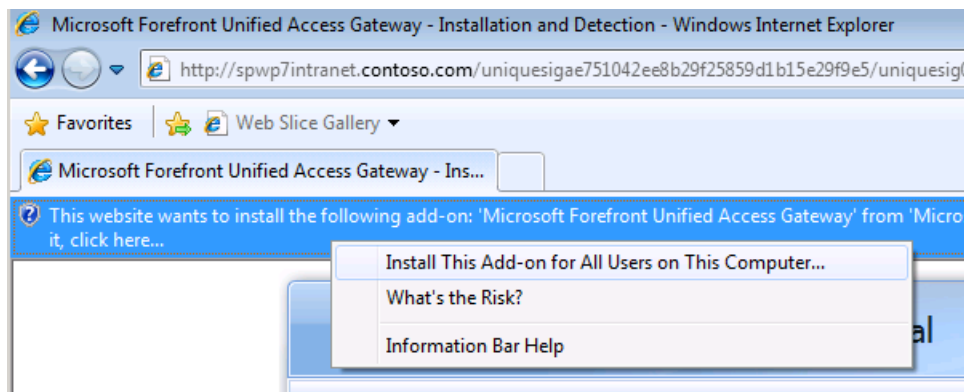

15. Test the Configuration

On the machine where you are running the Windows Phone 7 emulator, close all Web browsers to make sure the hosts file entry is recognized. Then, open Internet Explorer and navigate to <http://spwp7intranet.contoso.com>.

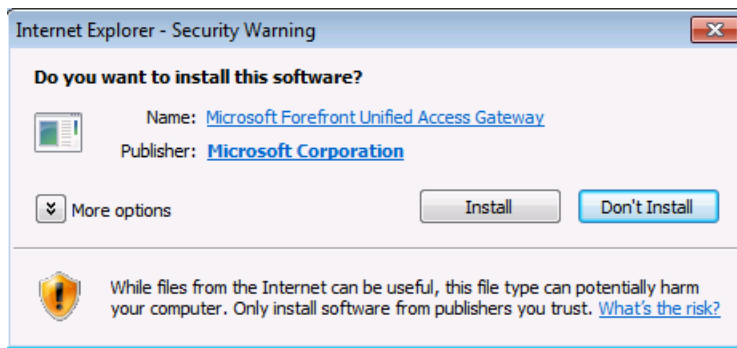
The UAG Server Application and Network Access Portal will appear.



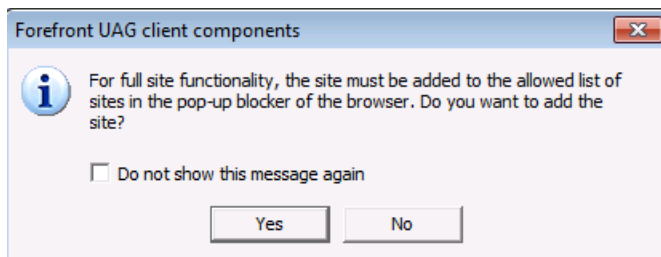
Download and install the ActiveX control.



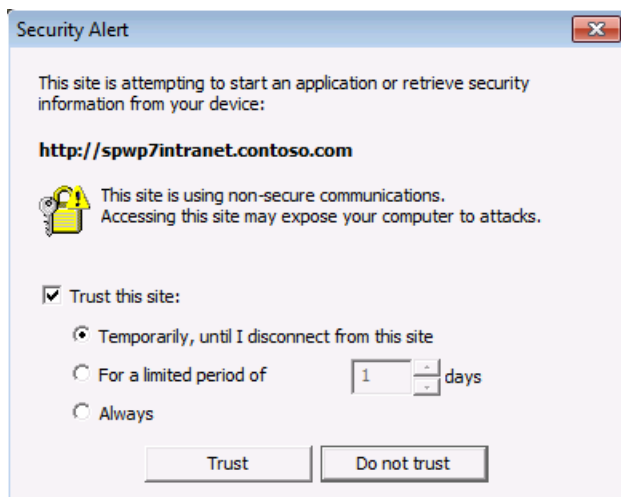
Click **Install**.



Click **Yes**.



Select the **Trust this site** check box, and then click **Trust**.



Enter credentials for a user who has access to the published SharePoint site, and then click **Log On**.

The screenshot shows a web browser window displaying the 'Application and Network Access Portal'. The page has a light blue header with the title 'Application and Network Access Portal'. Below the header, there is a 'Log On' section. This section contains three input fields: 'User name:' with the text 'contoso\tonip', 'Password:' with a masked password of ten dots, and 'Language:' with a dropdown menu set to 'English (en-US)'. A blue 'Log On' button is positioned below these fields. At the bottom of the main content area, a horizontal line separates a disclaimer: 'This site is intended for authorized users only. If you experience access problems contact the [site administrator](#).' The footer of the page contains the copyright notice '© 2010 Microsoft Corporation. All rights reserved.' and a link to 'Terms and Conditions'.

Application and Network Access Portal

Log On

User name: contoso\tonip

Password:

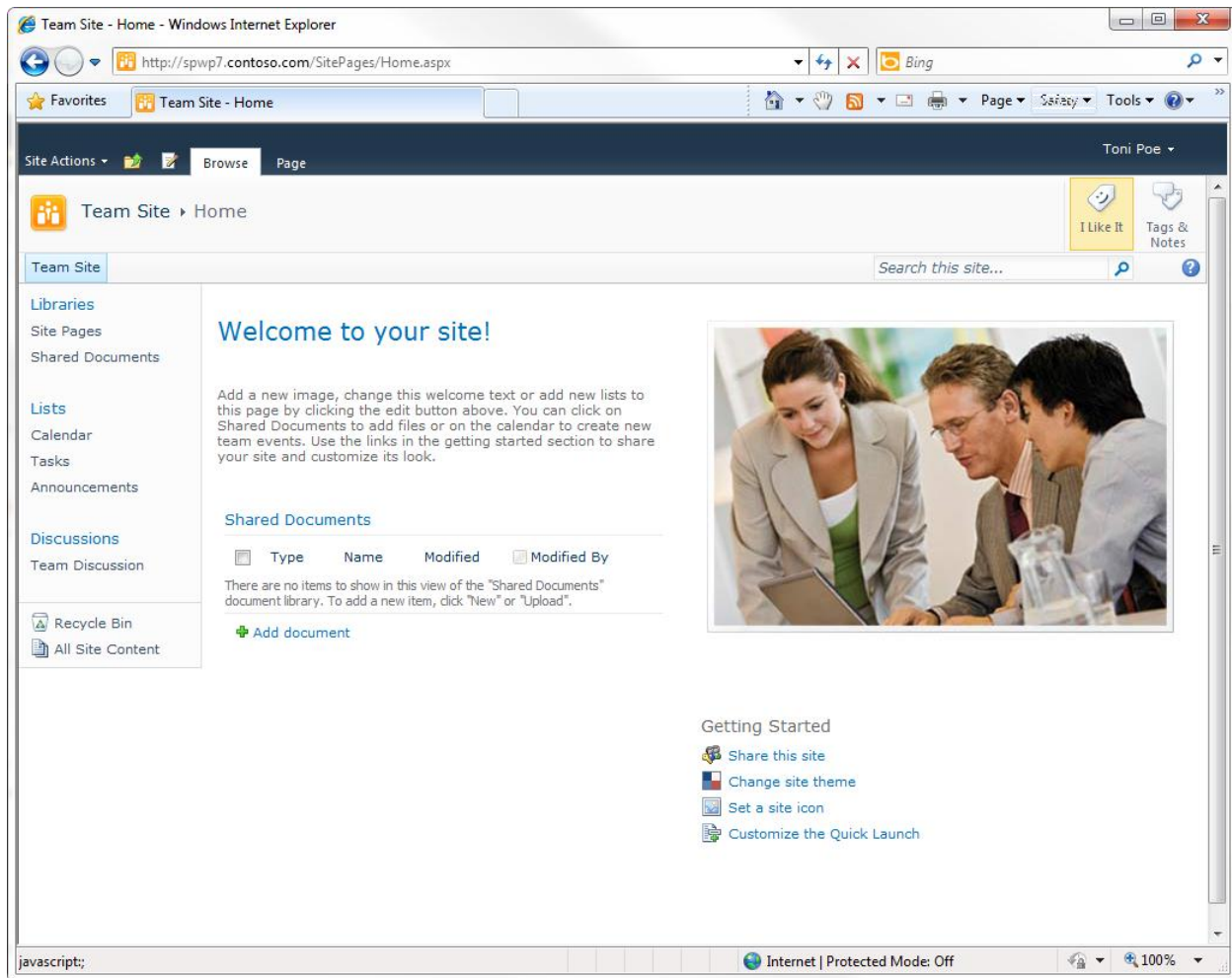
Language: English (en-US) ▼

Log On

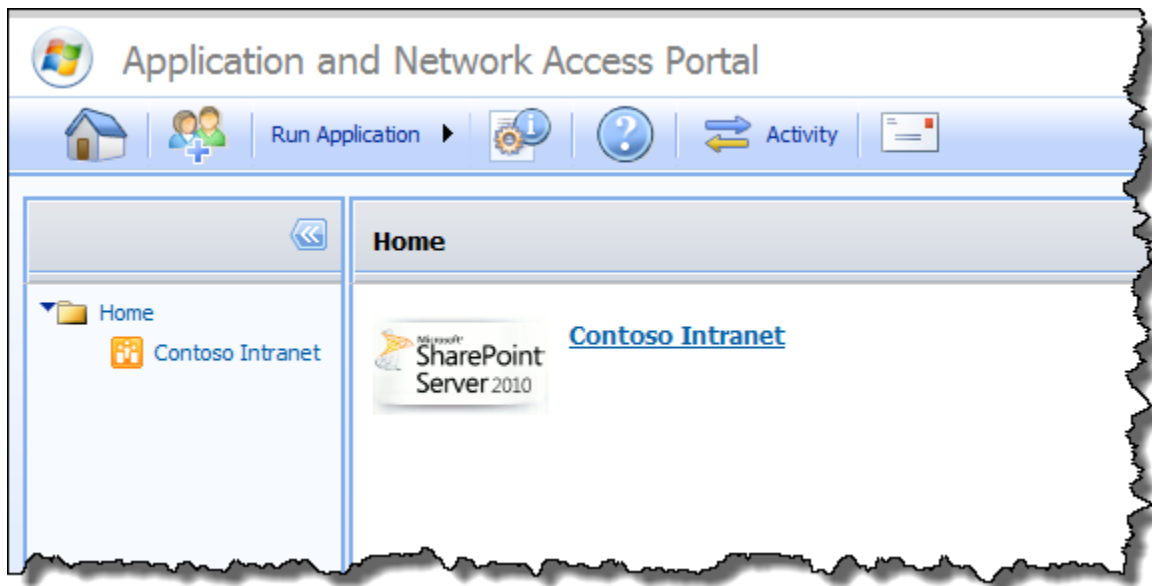
This site is intended for authorized users only.
If you experience access problems contact the [site administrator](#).

© 2010 Microsoft Corporation. All rights reserved. [Terms and Conditions](#).

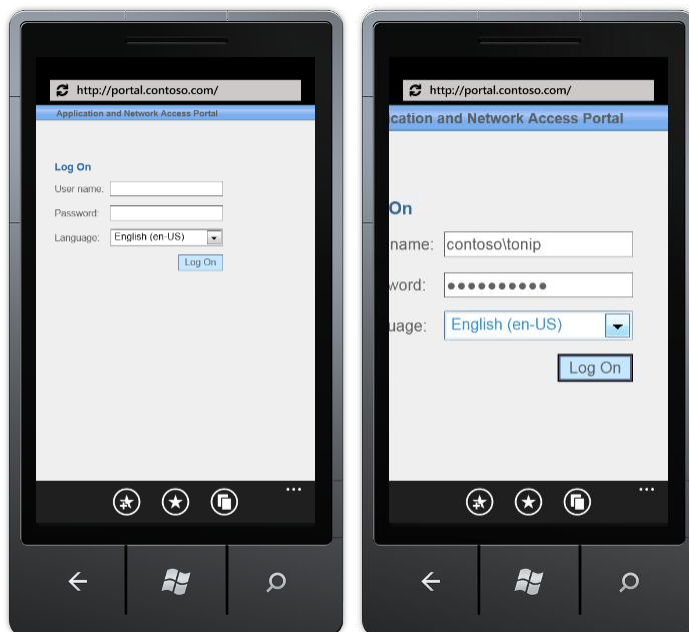
The SharePoint site is displayed in the Web browser. At this point, the SharePoint site published through UAG may now be accessed via a mobile device such as a Windows Phone 7 device.



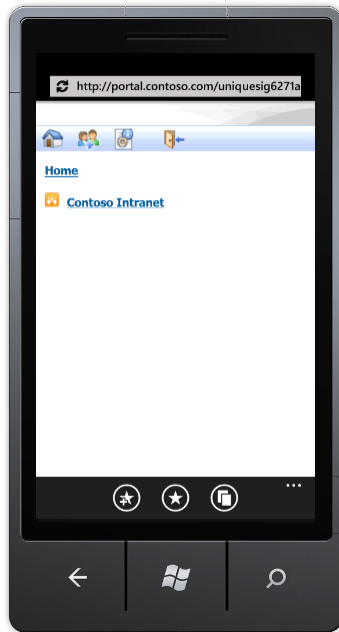
Optionally, you can test access to the UAG portal page by entering the portal URL <http://portal.contoso.com> into a browser.



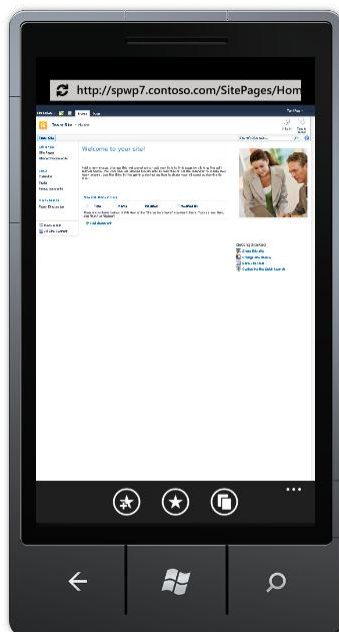
UAG also presents a Premium Mobile Portal to clients like the Windows Phone. From your emulator browser, enter the URL for portal, <http://portal.contoso.com>, and you will be redirected to the login page:



Enter the Contoso credential (for example Contoso\tonip and pass@word1) and you will be redirected to the Mobile Portal.



Select the **Contoso Intranet** link and you should see the SharePoint home page.

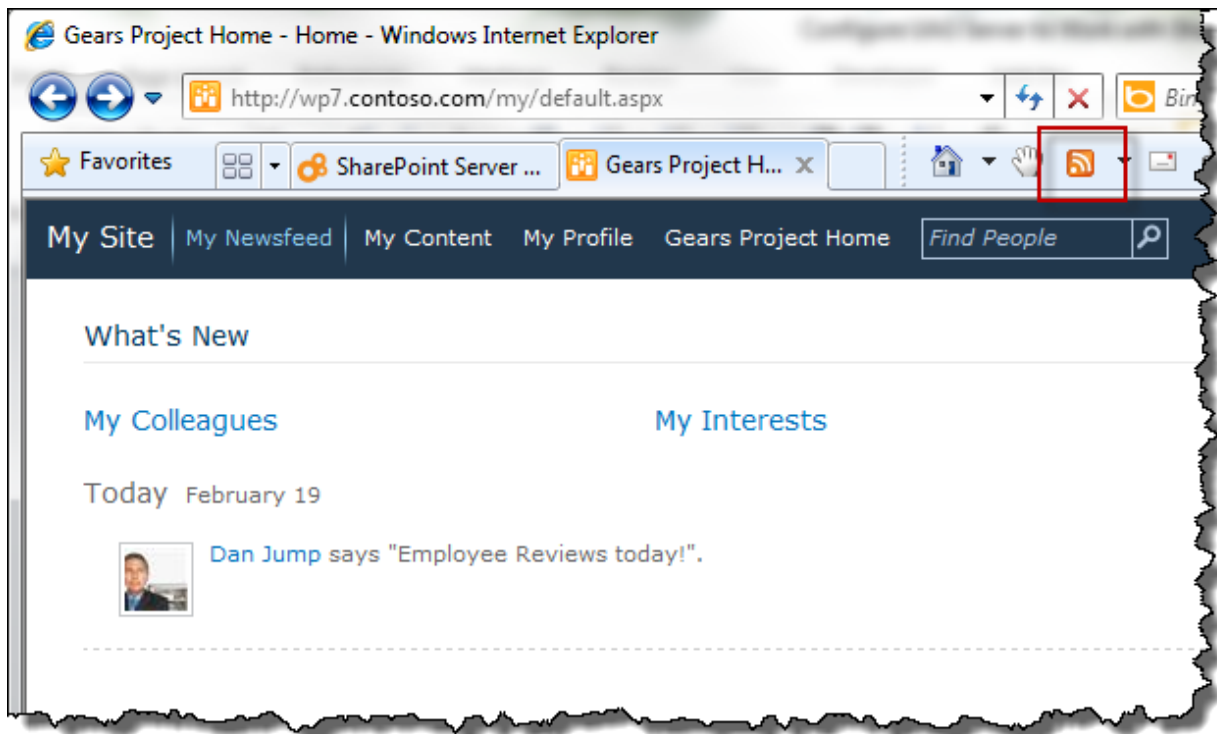


16. Test the Newsfeed RSS

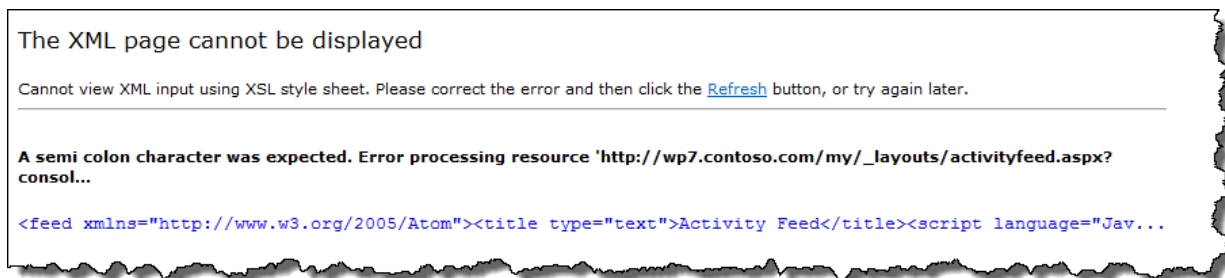
When SharePoint 2010 Products are published through UAG, some of the default pages are not parsed correctly. Of primary importance to this document is the Activityfeed.aspx page that renders the consolidated newsfeed. Test your configuration by following the directions below.

Activityfeed.aspx

Go to the My Newsfeed page, and choose to view the RSS feed.



The following error is displayed.



Resolution

The default rule for hiding the log off is being applied to the ActivityFeed.aspx page. To change this rule to prevent the application of the rule, perform the following steps:

Open the appropriate AppWrap configuration file in a text editor. The files are found in the directory %ProgramFiles%\Microsoft Forefront Unified Access Gateway\von\Conf\WizardDefaults\AppWrapTemplates.

Back up and edit the AppWrap file appropriate for the protocol of your portal:

HTTP_WhlFiltAppWrap_ForPortal for HTTP and HTTPS_WhlFiltAppWrap_ForPortal for HTTPS.

Locate the line:

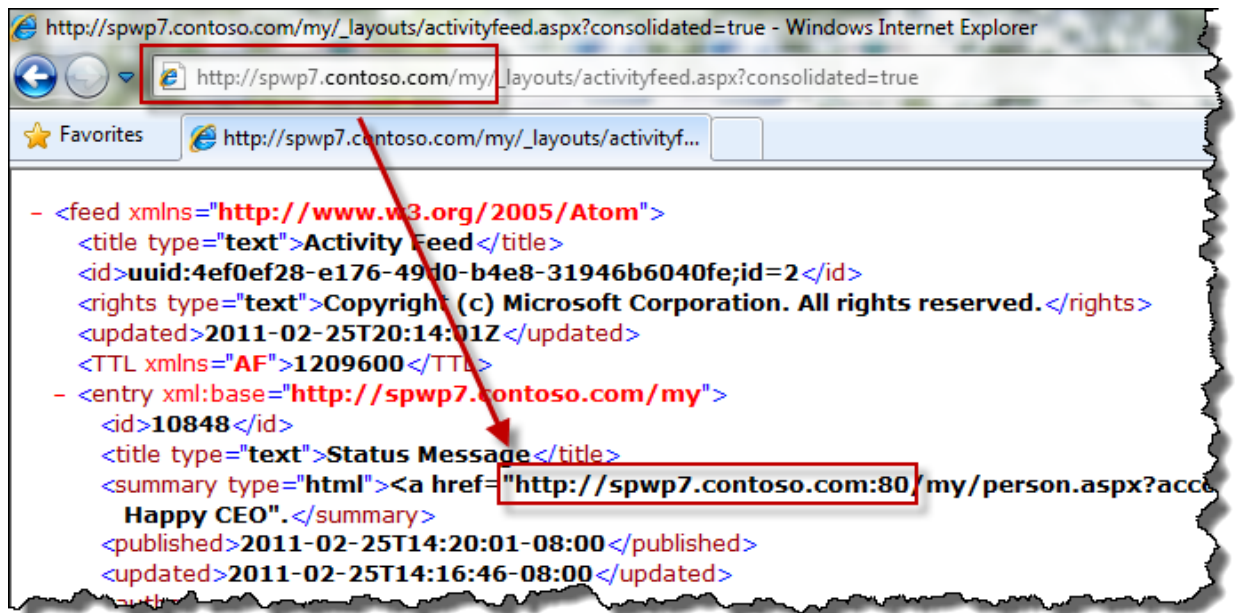
```
<!-- for sharepoint 2010 conditional appwrap hide log off -->  
<URL case_sensitive="false">.*\.aspx.*</URL>
```

Change it to read:

```
<!-- for sharepoint 2010 conditional appwrap hide log off changed to  
      exclude ActivityFeed.aspx -->  
<URL case_sensitive="false">^.*(?!(^|\\|/)(activityfeed))\.aspx.*</URL>
```

Save the file and activate the new configuration.

Return to the My Newsfeed page, and click the RSS button. The page should render correctly and the URLs should be correctly rewritten.



Resources

Forefront Unified Access Gateway on TechNet

<http://go.microsoft.com/fwlink/?LinkId=216132>

Closer to the Edge Blog

Author: Jason Jones

[Forefront UAG SP1 Endpoint Assessment Changes Impact Mobile Devices like iPads/iPhones](http://go.microsoft.com/fwlink/?LinkId=216133)

(<http://go.microsoft.com/fwlink/?LinkId=216133>)

Silverlight Web Services Team Blog

[Workaround for accessing some ASMX services from Silverlight 4](http://go.microsoft.com/fwlink/?LinkId=216134)

(<http://go.microsoft.com/fwlink/?LinkId=216134>)

About the Authors

Todd Baginski, MVP

Todd is an independent consultant and five-time Microsoft SharePoint Most Valuable Professional who uses SharePoint, Silverlight, Office, Windows Phone 7, and .NET technologies to create Web sites and custom solutions for information workers. Todd is the content author and presenter for the Business Connectivity Services (BCS) portion of the SharePoint Microsoft Certified Masters (MCM) program and a regular speaker at the TechEd, SharePoint Connections, and Microsoft SharePoint conferences. Todd contributes regular columns to SharePointPro Connections magazine and also recently served as the technical editor for Scot Hillier and Brad Stevenson's book, *Professional Business Connectivity Services in SharePoint 2010*. Todd also created the Microsoft Business Productivity Online Suite (BPOS) training materials for SharePoint Online (SPO) 2010 while BPOS 2010 was in the alpha and beta stages, and delivered the BPOS/SPO training materials at a post-TechReady 11 conference for only Microsoft employees.

Todd is a very active person who enjoys spending time with his family and skiing every chance he gets. Todd can be reached at todd@toddbaginski.com.

Matthew McDermott, MVP

Matthew McDermott, Microsoft SharePoint Server MVP, is a founding member of Aptillon, Inc. and Principal Consultant for AbleBlue in Austin, Texas. AbleBlue specializes in SharePoint integration, strategy and implementation consulting. Matthew is an author and specialist in SharePoint technologies focused on Web content management, collaboration, search and social computing. Matthew has led SharePoint implementations for Fortune 500 companies since 2002. Matthew's blog (www.ableblue.com/blog) features topics of interest to developers, IT pros, and end users alike. Matthew can be reached at matthew@ableblue.com.

Matthew's free time is spent as a canine handler for K9 Search • Austin, a volunteer K9 search team serving the FBI and Austin and San Antonio Police Departments. An accomplished cook and bartender, in his spare time Matt spends as much time with his wife as his dogs will allow.

Ben Ari

Ben Ari is a senior security engineer with Microsoft who specializes in UAG and provides support for the product for Microsoft's Premier and Professional customers worldwide. Ben is also an active journalist and blogger, and recently published the *Microsoft Forefront UAG 2010 Administrator's Handbook* (ISBN 978-1849681629). You can find Ben's blog at <http://blogs.technet.com/b/ben>.