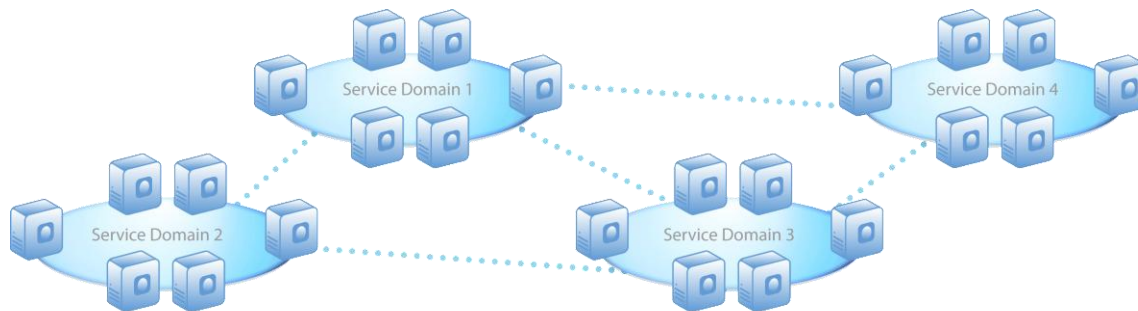


# Using the Configuration Service 5.0 Visual Studio Template

6/9/2011

© Microsoft Corporation 2011



**THIS IS NOT A PRODUCT SPECIFICATION.**

*This document and related sample code supports Windows Server® 2008 R2, Windows Azure, SQL Azure, Azure AppFabric and the Microsoft .NET Framework 4.0 as a redistributable sample application kit.*

*The information contained in this document represents the current view of Microsoft Corp. on the issues disclosed as of the date of publication. Because Microsoft must respond to changing market conditions, this document should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.*

*Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Microsoft grants you the right to reproduce this guide, in whole or in part.*

*Microsoft may have patents, patent applications, trademarks, copyrights or other intellectual property rights covering subject matter in this document, except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights or other intellectual property.*

*© 2011 Microsoft Corp. All rights reserved.*

*Microsoft, Windows Server, Windows Azure, SQL Azure, SQL Server, the Windows logo, Windows, Active Directory, Windows Vista, Visual Studio, Internet Explorer, Windows Server System, Windows NT, Windows Mobile, Windows Media, Win32, WinFX, Windows PowerShell, Hyper-V, and MSDN are trademarks of the Microsoft group of companies.*

*The names of actual companies and products mentioned herein may be the trademarks of their respective owners.*

## Contents

Introduction .....	2
Template Files and Location .....	2
Using the Template.....	2
Creating the Service Solution.....	3
Creating the Visual Studio Client Solution .....	12
Connecting the Client Application to the Service Host.....	20
Allowed Linked Services List .....	33
Additional Exercises .....	33
Adding new Monitored Databases .....	33
Showing Data for More Than Five Queries .....	36
Primary vs. Generic Services .....	37
A Client that Uses Many Connected Services .....	38
Additional Topics to Be Added Later.....	38

## Introduction

This document provides a brief overview of the Configuration Service 5.0 Visual Studio 2010 Template and Solution Wizard provided with the .NET StockTrader 5.0 sample application. This template automates the creation of a solution that implements the Configuration Service like StockTrader 5.0, creating a structured solution with the startup logic and references to the appropriate shared libraries already added. The solution created also templatizes the StockTrader 5.0 design pattern itself, including User Interface, Business Service, and Data Access layers. It includes the same DAL used by StockTrader 5.0 for high performance. The developer can either use this design pattern, or is free to implement any alternate design pattern of their choosing once the template solution is created by the Wizard. The Wizard can target any host environment for .NET 4.0, including Azure Web, Worker and VM Roles, as well as on-premise host applications including NT Services, console applications, Windows applications, and IIS-based Web applications. Once the Wizard finishes, the developer will have a working application that they can run with F5 Go from within Visual Studio, and immediately use ConfigWeb to help manage and monitor the application. The goal is to provide a quick starting place for building applications and services that implement the Configuration Service, as well as multi-tier WCF-based applications that can utilize a variety of WCF features, transports and security standards. Just start with the template, and add your own business processing logic, service and data contracts, and data access logic for your scenario.

## Template Files and Location

The StockTrader 5.0 setup should install the template in the [user]\documents\Visual Studio 2010\Templates\ProjectTemplates\ folder, and also register the Template Wizard assembly on the system. In addition, the setup creates a system environment variable 'StockTraderInstall' that points to the location of the StockTrader base setup, which contains all the Configuration Service shared libraries required by the template solutions and projects. The template is also located in \stocktrader\configuration\ConfigurationVSTemplate\Config\ConfigService5Template.zip. The wizard assembly is ConfigService.RepositoryCreate.dll, and can be found in the \stocktrader\setup\util directory. To optionally compile and look at the source for the wizard itself, you will need to install the Visual Studio 2010 SDK, free on MSDN.

## Using the Template

In this walkthrough, we will use the template to create two solutions. The first will host a service, and have no user interface. The second will be an ASP.NET Web application that will connect to and use the service.

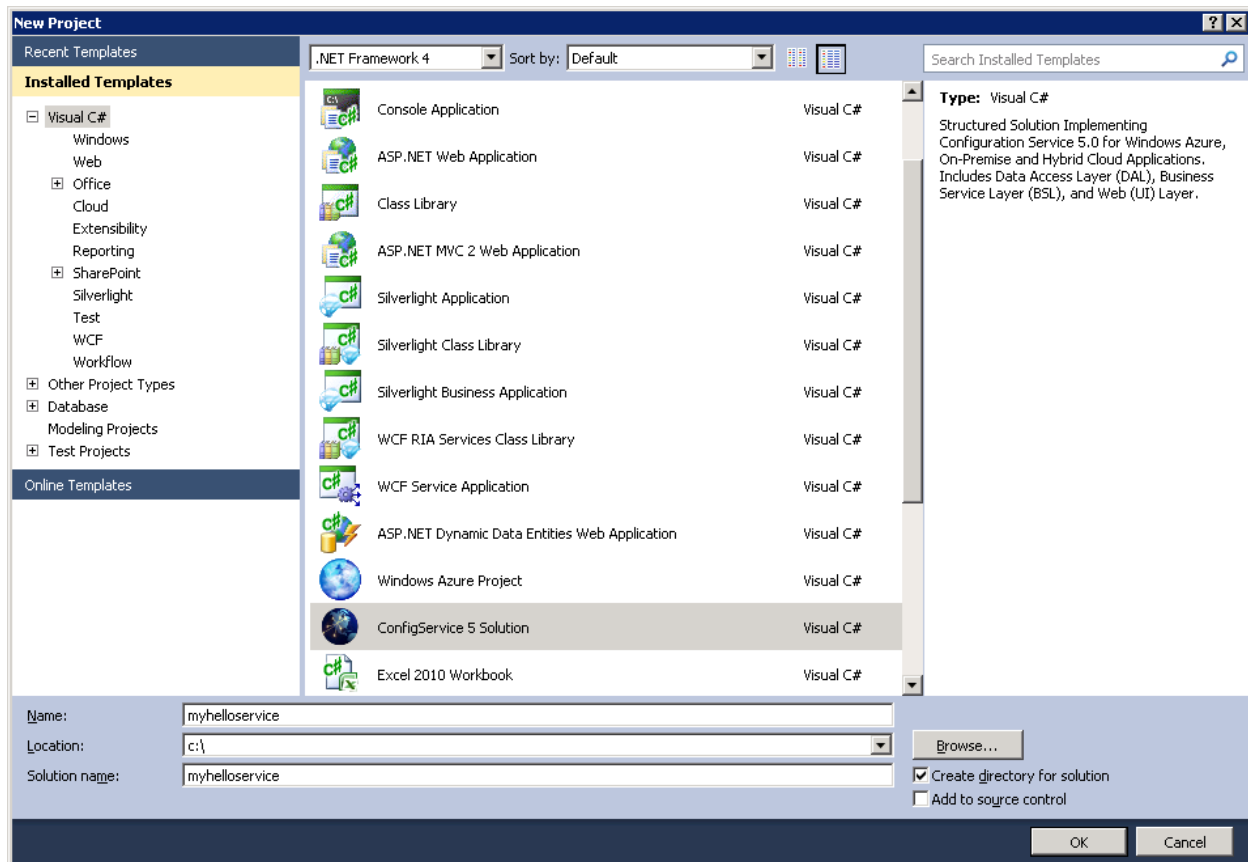
#### Note

Because each application you will create in this walkthrough (client and host) implements the Configuration Service, the clients will load-balance their requests to the service host nodes automatically for on-premise deployments via built-in Configuration Service load-balancing/failover (Azure provides the load balancing for the cloud). So you can run the clients and services on as many different servers you want. But for Web applications, the application does not 'start' until it receives a first page request. This 'start' will trigger new service hosts to notify clients it is available. On Windows Azure, new instances (nodes) will automatically eventually get client requests based on the Azure Fabric Controller load balancing. But for on-premise deployments (unless using a similar external load balancer such as F5), for IIS-hosted services, you would need to browse to the [http://\[server\]/hello/hello.svc](http://[server]/hello/hello.svc) from a browser to trigger IIS to start the application on that server and initiate notifications and the startup procedure for the Configuration Service. Until then, the application may be deployed to the server, but will be inactive. For non-IIS host applications such as NT Services, Console applications etc., startup procedures just start automatically when the host application starts.

## Creating the Service Solution

All the developer needs to do (assuming StockTrader 5.0 has been setup on their computer), is to run Visual Studio as an Administrator (always do this), and choose to create a new project. The template will show up within Visual Studio as a C# project type, called "**ConfigService 5 Solution**", as shown below.

- Run Visual Studio 'As Administrator'.
- Choose New Project.
- Click on the C# root Project Template node.
- Find the ConfigService 5 Solution template, and highlight it.



After choosing a name for your Solution and a disk location, simply click OK and the Wizard will launch.

- Type in **myhelloservice** as the project name.

Create Configuration Repository and Visual Studio Solution

Microsoft .NET Configuration Service 5.0 Solution Wizard Version 5.0.0

Create Databases and Solution

**Service Details**

☒ Create Exception Logging DB

☒ Include Hosted Service

☐ Include Service Client

☒ Include Data Access Layer (DAL)

Host Name Identifier: My First Service

Cluster Name: My First Service Nodes

Business Service Implementation Namespace: Sample.HelloServiceImplementation

Business Service Contract Namespace: Sample.HelloServiceContract

Business Service Data Contract Namespace: Sample.HelloServiceDataContract

Configuration Service NameSpace: Sample.HelloServiceHostConfigurationImplementation

Settings Class NameSpace: Sample.HelloServiceSettings

**Hosting Details**

Note: For IIS-hosted services you MUST enter the correct application VDir + the .svc filename as the virtual path below, and the correct IIS listen port!

Host Environment:

**On-Premise**

☒ Internet Information Services 7.x ☐ Internet Information Services 6.x

☐ Windows Application ☐ Windows NT Service ☐ Console Application

**Cloud**

☐ Windows Azure Web Role ☐ Windows Azure Worker Role

NA:

Azure DNS Base Address: Ex: myhelloservice.cloudapp.net

Node Svc Virtual Path: Hello/node.svc

Node Service Port: 80

Config Svc Virtual Path: Hello/config.svc

Config Service Port: 80

Note this wizard will create a virtual directory at the base path as specified above.

Config/Primary Service Https Port: 443

**Database Details**

Database Server: dotnetdns.com

SQL Login Mode: ☒ SQL Authentication

Admin Login ID: sa

Password:

Name for New Configuration Database: HelloServiceRepository

Create Cancel

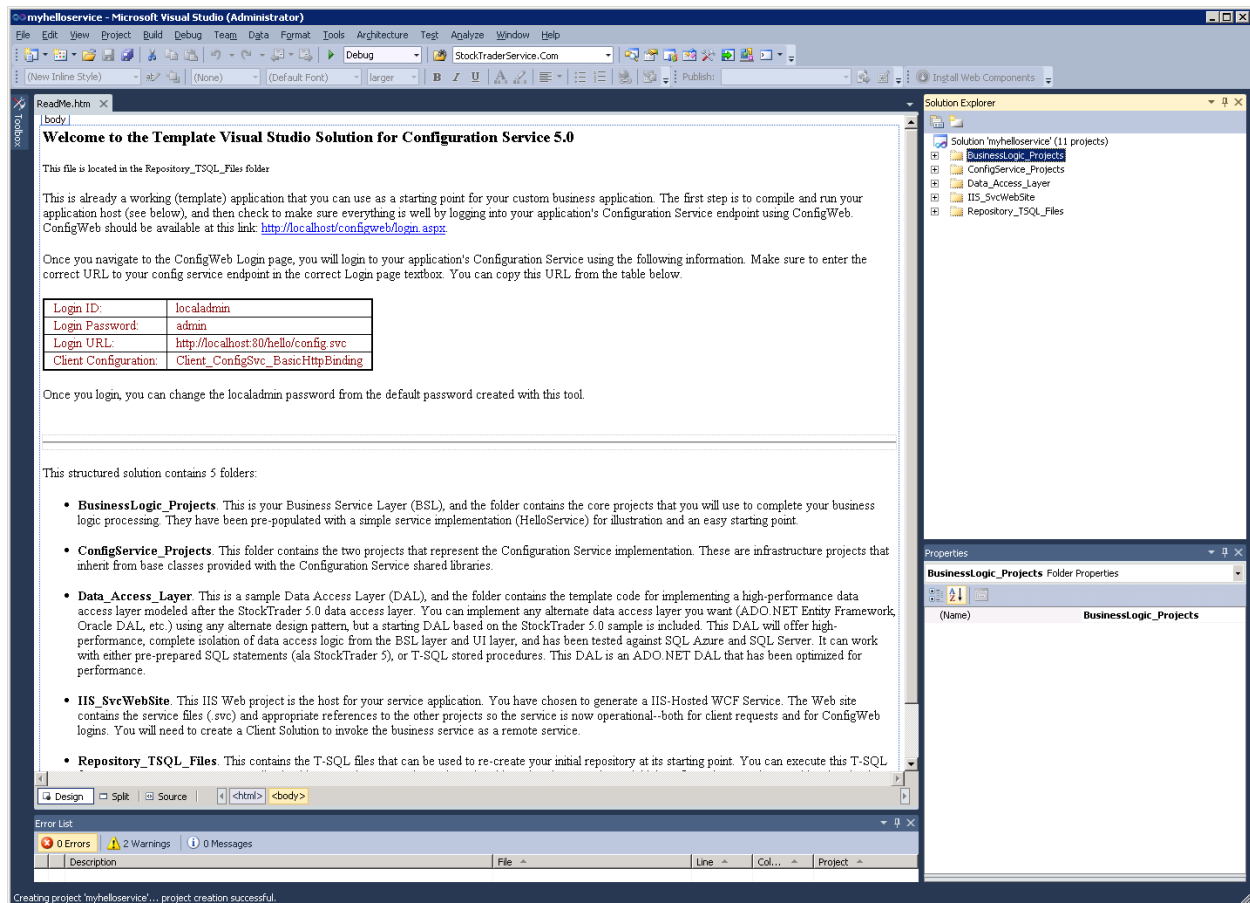
Note that almost all fields have been filled in and can be left completely alone, which is recommended for the first time you use it. You should choose which type of **host** you are creating, choosing from the available On-Premise and Azure host types. Note for an Azure VM Role; you would treat the application as an On-Premise application, using any on-premise host types listed. So, for the first time through, you should choose the host type, and leave the other fields as defaults; and provide your database login credentials, as shown above. Note we have changed one setting above, and to follow this guide you should do the same:

- **Deselect Include Service Client.** In this solution, we will be creating a remote service that is not a client to other services.
- Leave the host type as Internet Information Services 7.x.
- Leave all other defaults as they are for this walkthrough.
- Fill in the name of your database server (this can be SQLExpress or a remote SQL Server).
- Fill in administrator credentials to the database, for example your 'sa' account.
- Click **Create**.

### Note

For both IIS and Azure Web Roles, you can create a virtual directory automatically by typing in the virtual directory as part of the virtual path to the node service and configuration service endpoints above (the path must be the same but this is validated). For example, the default above will create a virtual application/vdir as `http://localhost/hello/*.*`. You can also do this on Windows Azure Web Roles. If no virtual path prefaces the `.svc` file, the site will be the root of the host (`http://localhost/*.*`).

After clicking **Create**, the Wizard will first create your Configuration Service repository, the database that will store WCF information and configuration settings much like an `<appSettings>` section from `web.config` or `app.config` files (except settings are kept in sync across scale-out nodes by the Configuration Service). After the database is created, an appropriate structured solution will be built, and you will have a readme file (Html) within this Visual Studio solution open for reading. Switch to HTML design view to read the readme file.



In general, you should be able to simply hit F5-Go to build and launch your program right away, as sample logic is already included. But you do need to compile the application before you can login to the application's configuration service.

- Compile the solution.



- The IIS virtual directory has already been created, so you do not need to do anything with IIS. The Web application is configured to use the full IIS (vs. development IIS) server on your PC.

**Note**

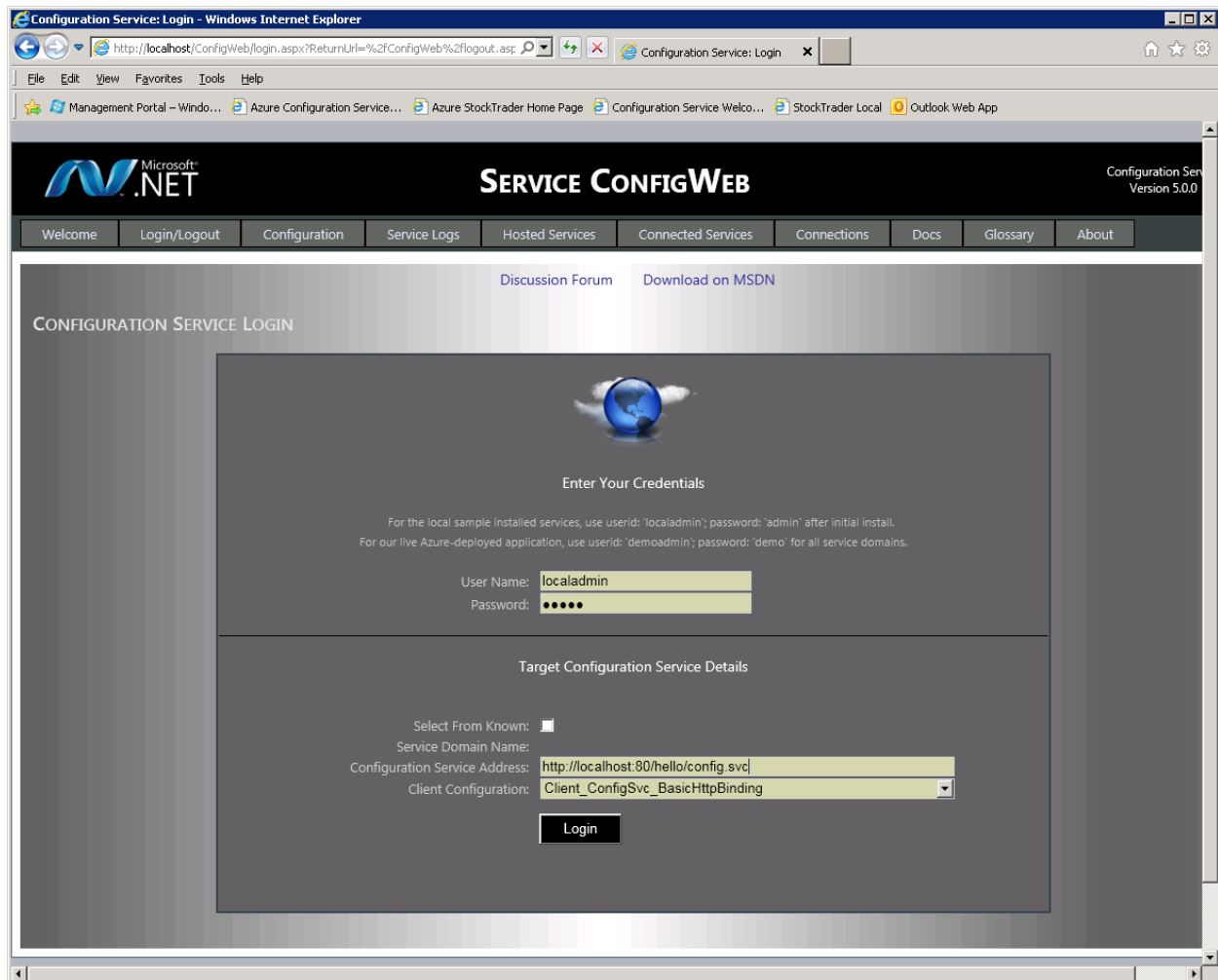
You can run the application with F5 GO within Visual Studio 2010. Visual Studio will launch a browser (if a Web application), and debug the solution against the full IIS server, and not use the development server. This is important when running multiple nodes, since the services should start on predictable endpoints (and ports). But for a service application with no client web pages, as with this solution, the only thing you can browse to are the service endpoints such as <http://localhost:80/hello/hellosvc.svc>. This will show the WCF HTTP Welcome/Helper page if the service is operating/starting correctly.

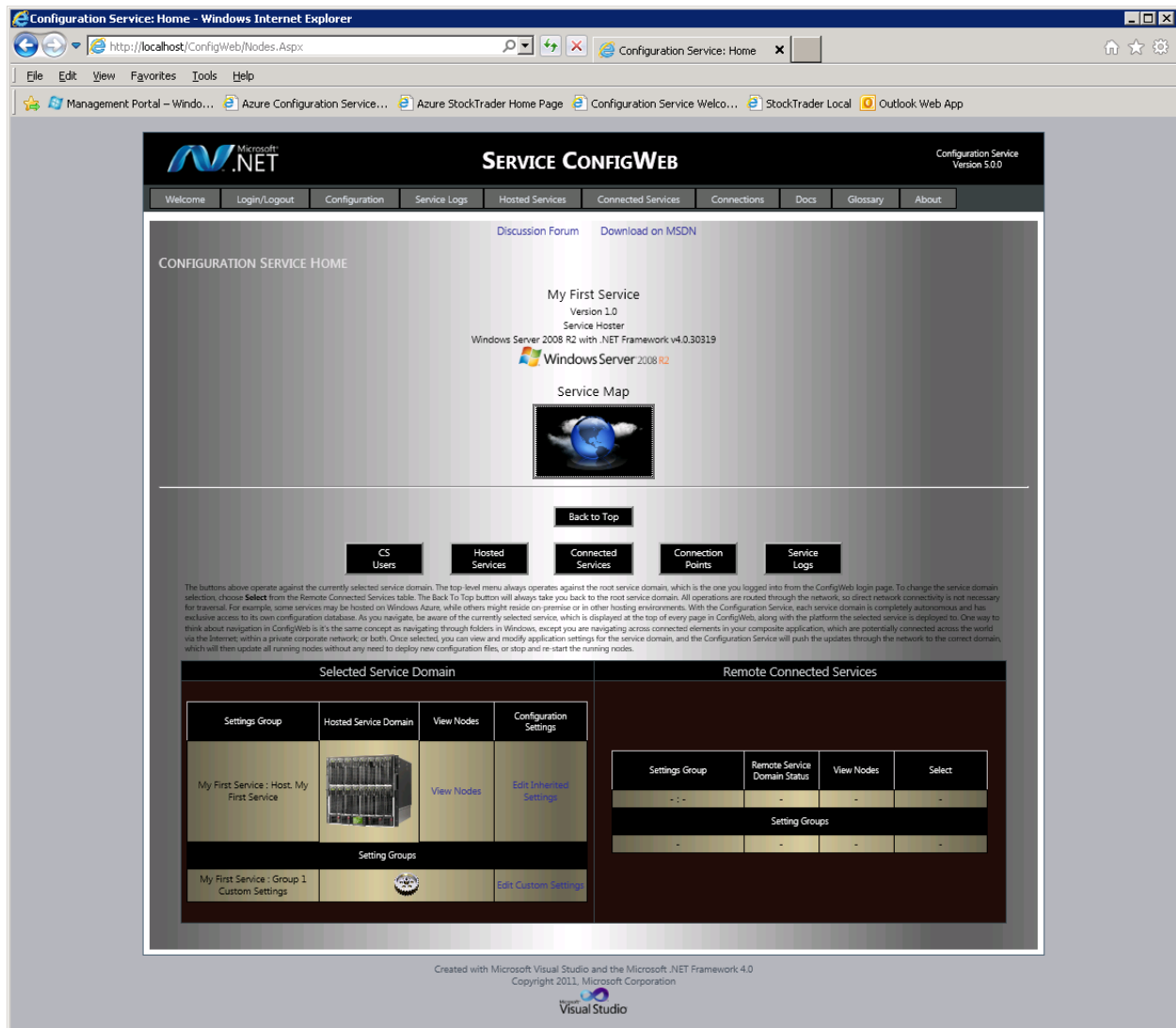
Once running, you can immediately use ConfigWeb to login to the host application's configuration service, based on the login information in the readme file. This is the best way to first make sure the application is working properly.

- Navigate to <http://localhost/configweb>
- In the login page, deslect 'Select from Known.'
- You can now enter the login address shown above in the ReadMe within VS:  
<http://localhost:80/hello/config.svc>
- Use localadmin/admin as the userid and password. You can change the password from this default in ConfigWeb at any time with the Users page<sup>1</sup>.
- Click the **Login** button.

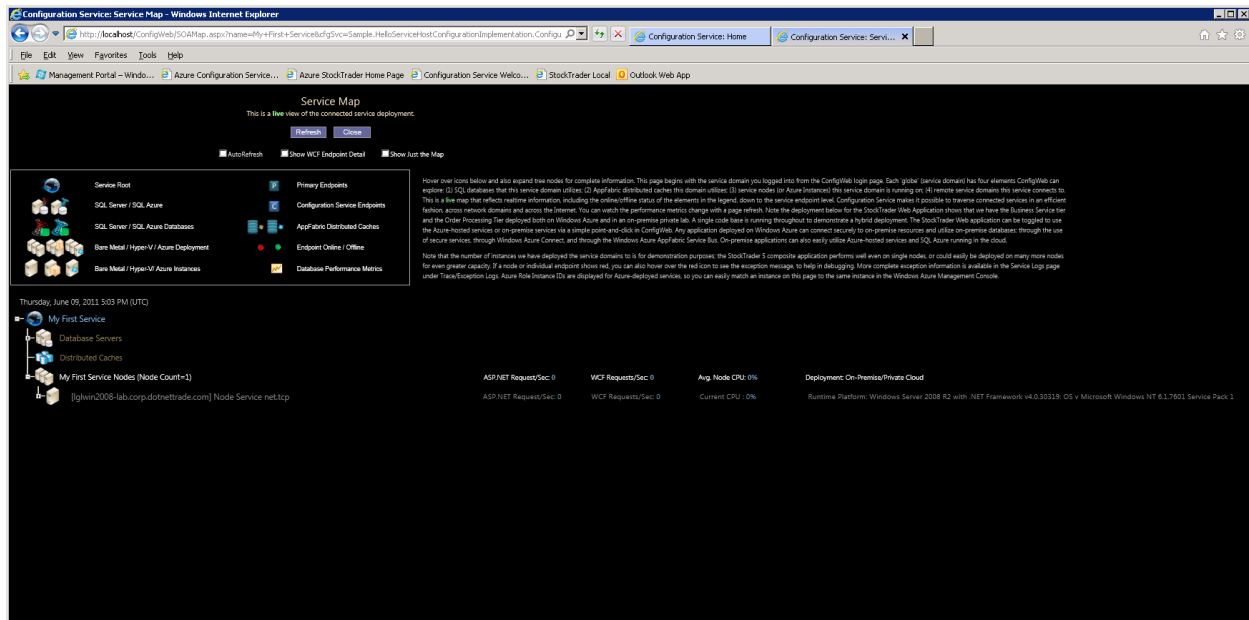
---

<sup>1</sup> If you do so, you will not be able to use Linked Services to navigate between client and host later in this walkthrough, unless you also ensure the client has matching admin credentials (login/password) in its Users table as well. See the later section on Linked Services.





- From this home page, bring up the Service Map view by clicking the Service Map button.



You can explore this page. Hover over icons for more information, and expand tree nodes. If everything is ok, you will not see any red or yellow icons in this view. This is a live view of the physical application deployment, with online status checking done by the Configuration Service on each page request. Base performance metrics are also shown. If any icon shows red, you can hover over it to see an exception message. Note that based on implementing the Configuration Service, you can view exceptions using ConfigWeb in the Service Logs page. So for complete logging and exception information, let's close this page and navigate to the Service Logs page.

- Click the Close button on the Service Map page.
- Choose Service Logs from the Config Service home page, it's in the top menu.
- From Service Logs, you need to click on the button **Trace and Error Logs**.



The Configuration Service can capture exception and trace messages in the logging database. The logging database is specified in a connection string within `web.config/app.config`. You can also change the setting “Detailed Logging” in ConfigWeb (inherited settings) to turn on logging of informational trace messages, and not just warnings and errors.

#### Note

A **WCF Service Fault Behavior** has been added already to the service the template creates. This behavior causes automatic logging of exceptions if they occur in the context of a service operation. However, you may want to write your own trace and exception messages to this log, perhaps outside of service operations (such as Web pages, etc). To do so, you would simply use a line of code as shown:

```
ConfigUtility.writeConfigConsoleMessage("\nError! Exception is: " +  
error.ToString(), EventLogEntryType.Error, true, new Settings());
```

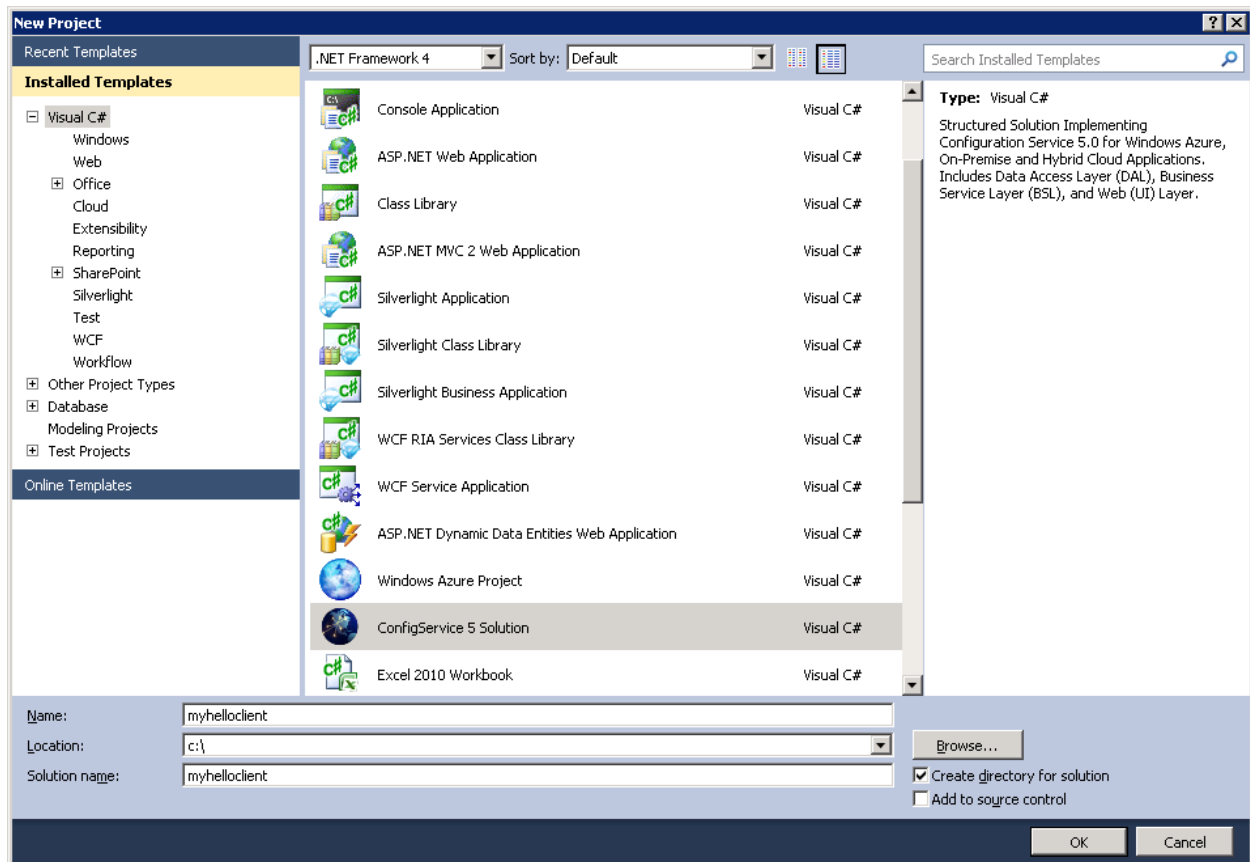
You will need references and using statements for ConfigService.ServiceConfigurationUtility.dll, and the Settings project that the solution wizard has already created in your solution (and reference already added to the business service implementation project). You can specify `EventLogEntryType.Error`, `EventLogEntryType.Warning`, or `EventLogEntryType.Information`.

Errors and Warnings are always written to the Database log, and also the node's Application Event log. Informational messages are only written if the 'Detailed Logging' configuration setting is set to true. It is false by default. This is an inherited setting you can set in ConfigWeb.

## Creating the Visual Studio Client Solution

Now we will create a second solution that will be the client ASP.NET application. The template will generate test pages to run the sample service operations. The template also creates pages that invoke the its own BSL layer operations within the Web application, to show how a standalone application can still be logically partitioned, even though it will not necessarily make remote service calls (e.g. not physically partitioned).

- Start another instance of Visual Studio 2010. Remember to run it 'As Administrator'.
- Choose the ConfigService 5 Solution template again.
- Name the solution **myhellocient**.



Create Configuration Repository and Visual Studio Solution

Microsoft .NET Configuration Service 5.0 Solution Wizard Version 5.0.0

Create Databases and Solution

Service Details

☒ Create Exception Logging DB

☐ Include Hosted Service

☒ Include Service Client

☒ Include Data Access Layer (DAL)

Host Name Identifier: My First Service Client

Cluster Name: My First Service Client Nodes

Business Service Implementation Namespace: Sample.HelloServiceImplementation

Business Service Contract Namespace: Sample.HelloServiceContract

Business Service Data Contract Namespace: Sample.HelloServiceDataContract



Configuration Service NameSpace: Sample.HelloServiceClientConfigurationImplementation

Settings Class NameSpace: Sample.HelloClientSettings

Hosting Details

Note: For IIS-hosted services you MUST enter the correct application VDir + the .svc filename as the virtual path below, and the correct IIS listen port!

Host Environment:

On-Premise  Cloud 

☒ Internet Information Services 7.x ☐ Internet Information Services 6.x ☐ Windows Azure Web Role

☐ Windows Application ☐ Windows NT Service ☐ Console Application ☐ Windows Azure Worker Role

NA:

Azure DNS Base Address: Ex: myhelloservice.cloudapp.net

Node Svc Virtual Path: HelloClient/node.svc Node Service Port: 80

Config Svc Virtual Path: HelloClient/config.svc Config Service Port: 80

Note this wizard will create a virtual directory at the base path as specified above. Config/Primary Service Https Port: 443

Database Details

Database Server: dotnetdns SQL Login Mode: ☒ SQL Authentication

Admin Login ID: sa Password:

Name for New Configuration Database: HelloClientRepository

Create Cancel

- In the Wizard, this time keep **Include Service Client** checked, but uncheck **Include Hosted Service**, as shown above.
- Leave the default host type as IIS 7.x.
- Leave all other fields as their defaults.
- Fill in your database details.
- Click **Create**.

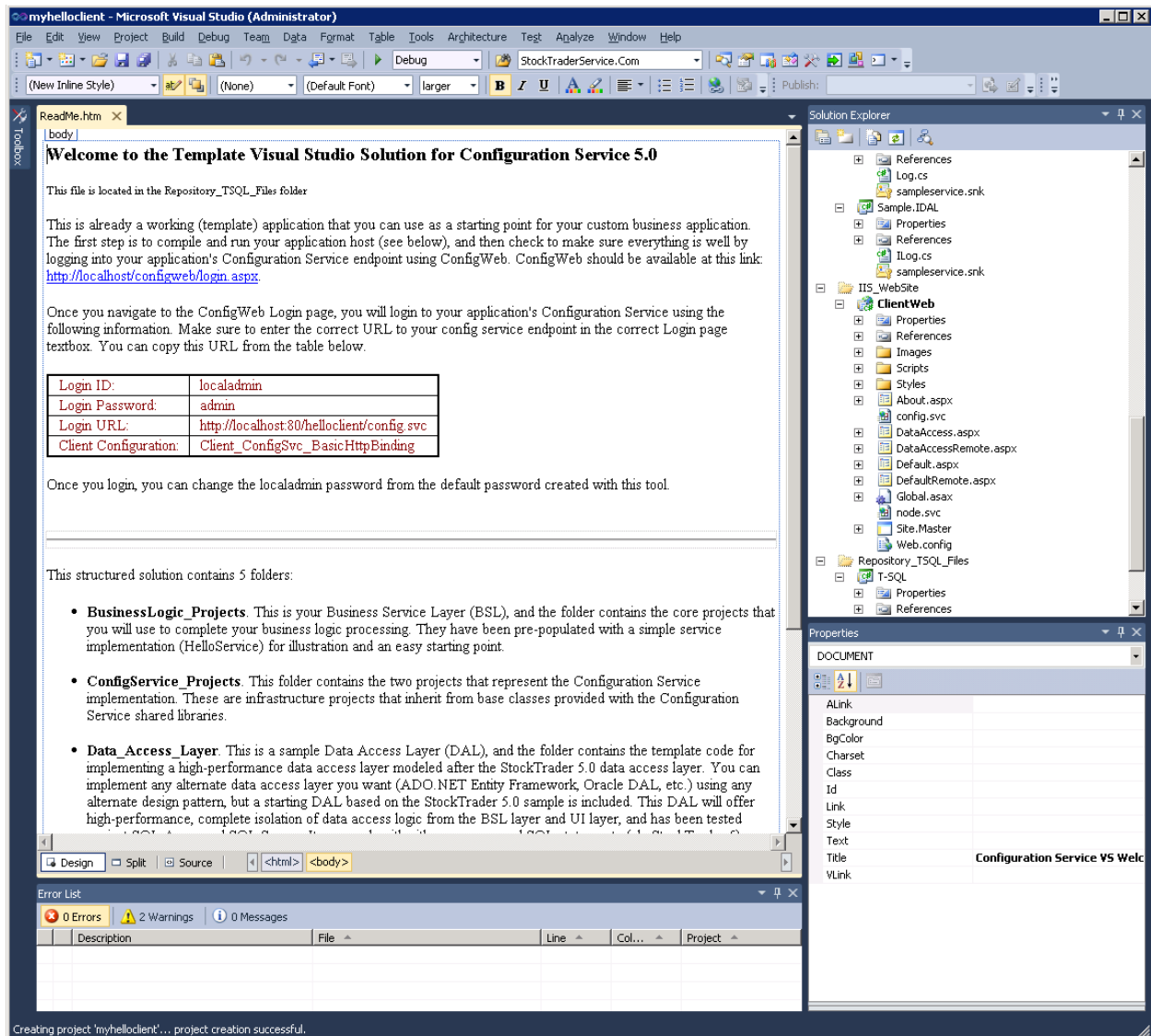
#### Note

This can be a completely different database server than the other solution, if you want. Services are autonomous and maintain their own independent configuration repositories. This allows complete location independence between clients and services, potentially across the Internet as with Windows Azure services. You will notice the wizard is creating a second, **different** database (configuration repository) on the specified server, this time named HelloClientRepository.

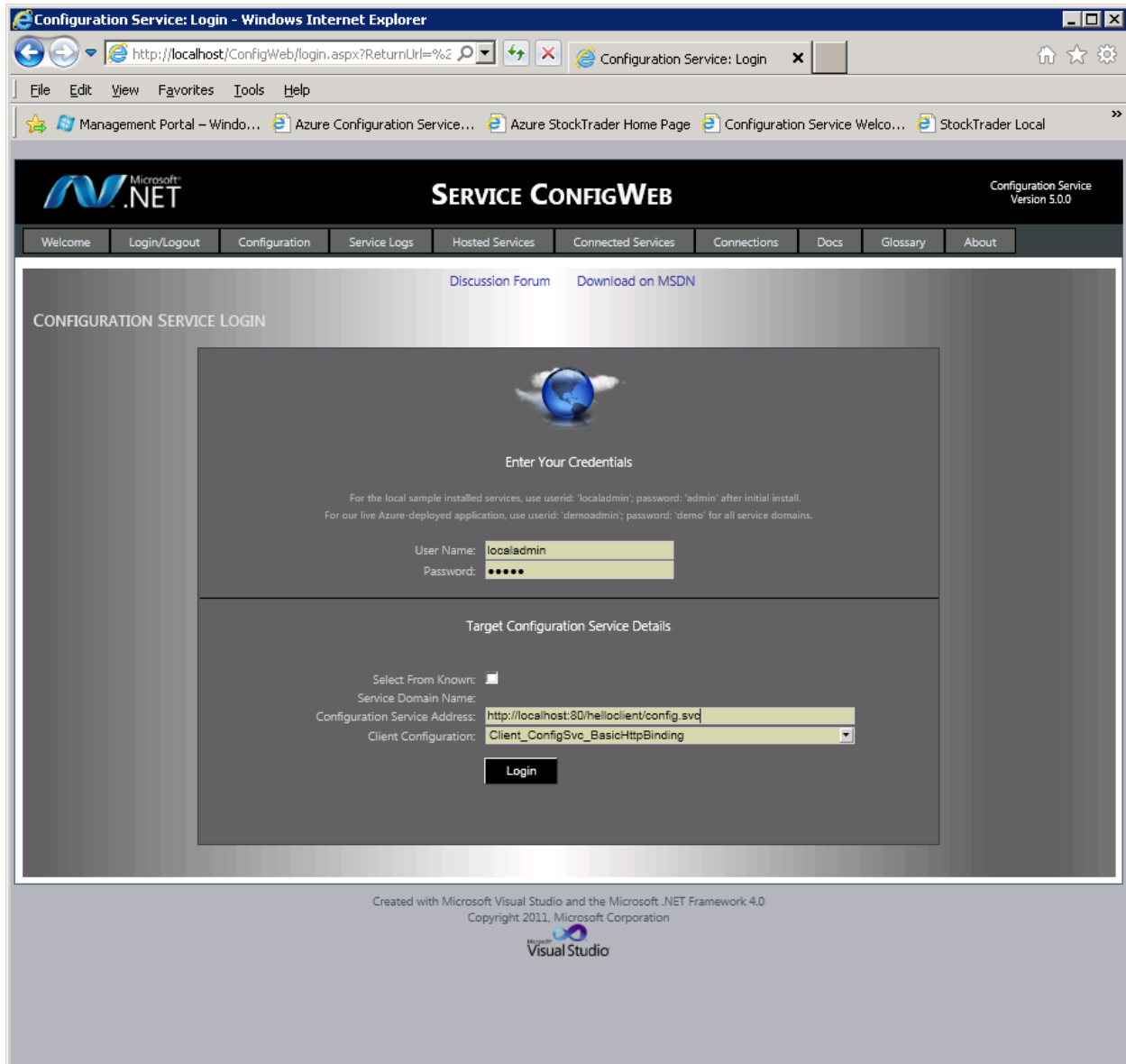


**Note:**

You might get an error creating the database with a message "Cannot Drop Logging-Data because it is currently in use." The Wizard always creates the same logging database, in this case Logging-Data. And your first service application is using it. Remember the logging database can be changed for a solution after it has been created simply by changing the connection string in Web.config/app.config to point to a different database. But it is often convenient, especially when developing/debugging, to have services share the same logging database. At any rate, if the logging database is in use, restart SQL Express/SQL Server or stop the IIS HelloService application so the wizard can complete. Also note the HelloService application, via its configuration service, is activated and already persisting request counts to the request logging table in the Logging-Data database. By default, requests are persisted every 60 seconds by each running node in a cluster. You might want to change this to a longer interval. As long as an application using the Logging-Data database is active, it will be re-establishing connections every 60 seconds to the database, even if you restart the database to drop existing connections.



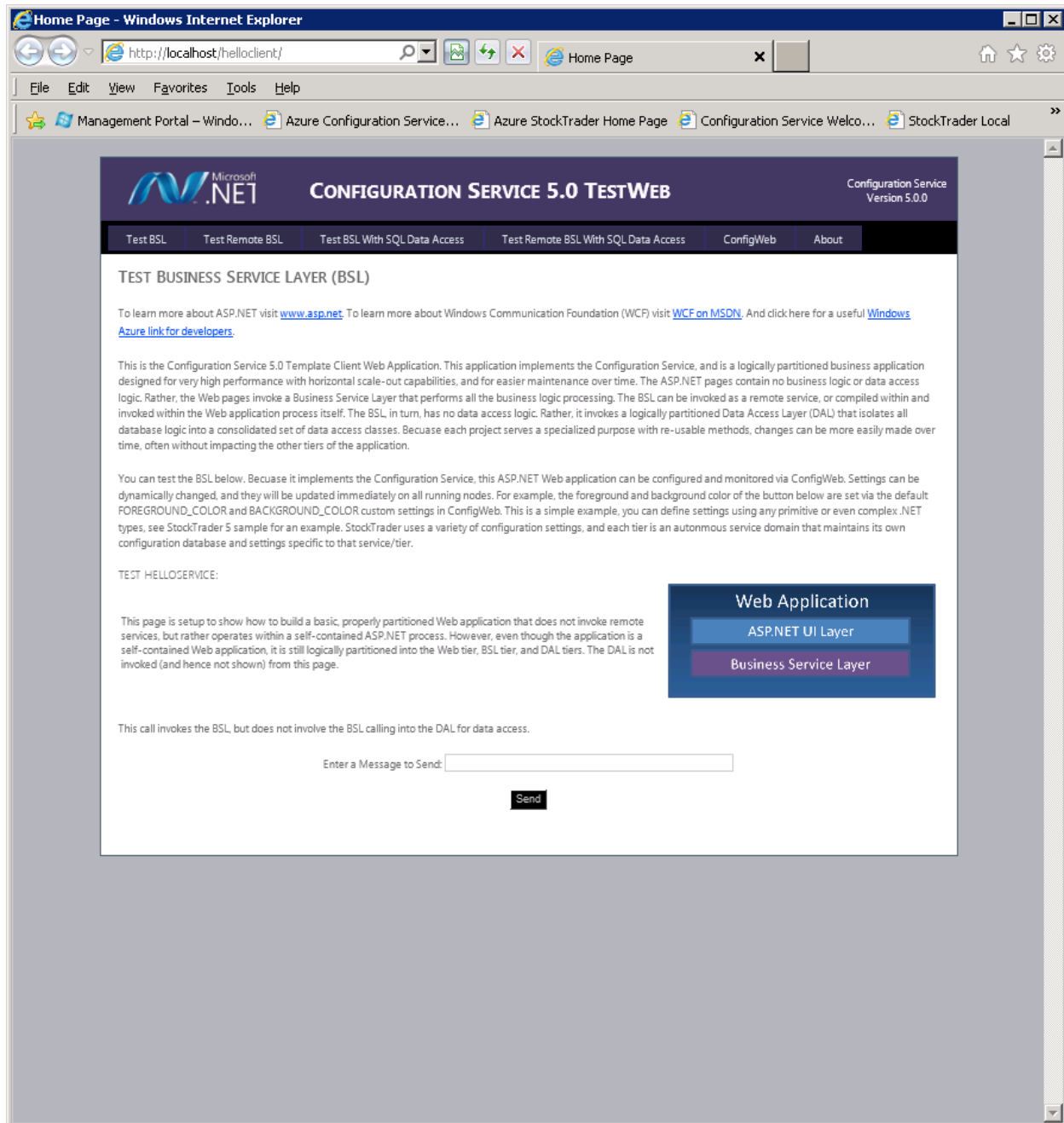
- Build the solution.
- Now, navigate to ConfigWeb in a browser.
- You can even navigate via the Visual Studio browser, if you want. Simply view the source of the welcome.html page in Visual Studio's editor, and ctrl-click the url to ConfigWeb within VS.
- Login as below using `http://localhost/helloclient/config.svc`.



- Remember to build the solution before attempting login.



- You can independently look at Service Map, and Service Logs. Note this client application is sharing the same Logging-Database as the first application created.
- This application has UI pages (template pages). So go back to Visual Studio, and hit F5 to build and launch the Web site in debug mode.



The client solution right now is a standalone Web application. Even though it is not hosting any service endpoints, it does have the Business Service implementation logic embedded in the solution as separate projects. This is because some developers might not be interested in building multi-tier remote services; but just ASP.NET web applications that use the Config Service and the logical partitioning with the BSL and DAL layers as the StockTrader application is structured. In this case, you can invoke the BSL methods simply as methods within the Web application, and the BSL will in turn invoke its logical DAL tier to talk to the remote database. So, immediately in the page above, you can test the local BSL operations. You are not yet ready to invoke remote services to the first Service solution, however.

- Use the template web site to invoke local BSL/DAL operations. There are two pages in the menu that do so, including the home page above.
- As template code, the DAL layer and BSL layer have sample logic that happen to read/write to from the Logging-Database, as an example (the same database used for service logs). You would of course change the BSL and DAL layers to be your logic, against your database(s).

**TEST BSL WITH DATABASE ACCESS LAYER**

This page is designed to call into the BSL, which will actually be performing the data access by in turn invoking the DAL. Results are loaded into model classes (the BSLDataContract project), and passed in turn from the DAL, to the BSL, and back to this page for display.

You should always logically partition the UI, BSL and DAL layers for enterprise-class business applications. Otherwise you will have database logic and business logic scattered throughout UI/pages, etc.; making most changes and ongoing application maintenance much more difficult over time. Note that the database operations are performed against the LOGGING-DATABASE created from the VS Wizard, either on SQL Azure in the cloud or an on-premise edition of SQL Server. For the transaction test, enter a message of "acid" to trigger an exception in the BSL after all the records have been inserted but before commit.

**Web Application**

- ASP.NET UI Layer
- Business Service Layer
- Data Access Layer

**Remote Enterprise RDBMS**  
SQL Server/SQL Azure

Choose a service operation to test:

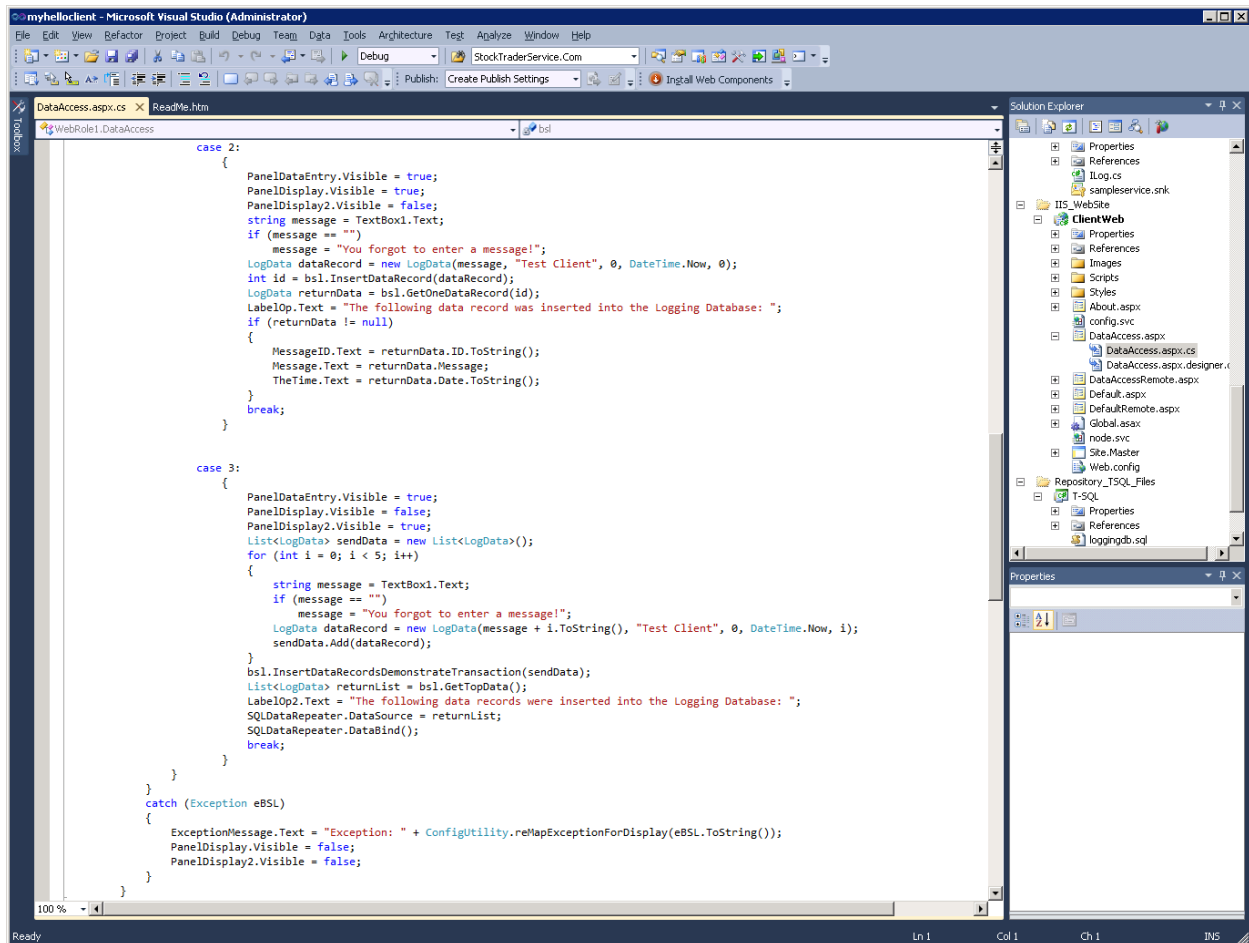
- ☒ GetTopData
- ☐ GetOneDataRecord
- ☐ InsertDataRecord
- ☐ InsertDataRecordsDemonstrateTransaction

GetTopData() returned the following data records:

ID	Severity	Source	Message	Date
4	1	MyFirstServiceClient	Web Application Global Application_Start: Web App New Node Starting. Hello!	6/9/2011 10:52:23 AM
3	0	MyFirstServiceClient	Done with Service Host Startup Procedures!	6/9/2011 10:52:23 AM
2	0	MyFirstServiceClient	Initializing Connection Pool WCF Client Channels.	6/9/2011 10:52:23 AM
1	0	MyFirstServiceClient	Successfully Loaded All Settings from configuration database...	6/9/2011 10:52:23 AM

Note the picture on each page describes the deployment topology of your application.

- Take some time to open the source files for the ASPX.cs code-behind files in the Client solution, and read the comments and look at the code.



## Connecting the Client Application to the Service Host

We are now ready to use ConfigWeb to establish the service linkage between the client application and remote service.

### Note

The template reproduces (independent copies of) each project in each solution created. However, really there are two projects that should be shared between the two solutions, both of which are schema for the remote service. These are the ServiceContract and ServiceDataContract projects. You can re-arrange the solutions to share these projects, if you wish. Just fixup references and namespaces as required, after deleting this from the Client Solution and adding in the ones from the Service Solution on disk. This way, as you change these method names, add methods, change data contracts, etc., you only need to do so in the shared projects—once.

- Goto the ConfigWeb home page for the Client application:



We will now add a Connected Service Definition to the client application, which will be a connected service definition to the remote service host application.

- Make sure you are logged into the **Client** application's Configuration Service (not the Host application). Simply look at the top of the page for the service name.
- Click on the **Connected Services** button.

### Note

Read the text on the home page about navigation in ConfigWeb. The top-level menu for the ConfigWeb site always operates against the root service domain you logged into. The buttons, however, are all context sensitive. As you later navigate across services using Linked Services, the buttons allow you to perform all the operations from ConfigWeb from the same login, without having to login to ConfigWeb independently for every service in your composite application. The top of every page shows which service is selected, at all times, if you get confused later. After a few times around the block, it should become clear.

The screenshot shows a web browser window titled "Configuration Service: Connected Services - Windows Internet Explorer". The address bar shows the URL "http://localhost/ConfigWeb/CSServices.aspx?name=My First". The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar shows various icons for navigation and settings. The main content area displays the "SERVICE CONFIGWEB" interface, which includes a navigation menu with links like Welcome, Login/Logout, Configuration, Service Logs, Hosted Services, Connected Services, Connections, Docs, Glossary, and About. The "Connected Services" section shows a list of services, with "My First Service Client" selected. This client is identified as "Version 1.0" and "Service Hosted", running on "Windows Server 2008 R2 with .NET Framework v4.0.30319". Below this information, there is a "Back to Top" button and an "Add Service" button. A table at the bottom lists the details of the connected services, with columns for Virtual Host ID, Service Name, Service Type, Service Contract, Client Configuration, and Edit/Delete. The footer of the page mentions "Created with Microsoft Visual Studio and the Microsoft .NET Framework 4.0" and "Copyright 2011, Microsoft Corporation".

Configuration Service: Connected Services - Windows Internet Explorer

http://localhost/ConfigWeb/CSServices.aspx?name=My First

Configuration Service: Conn...

File Edit View Favorites Tools Help

Management Portal - Windo... Azure Configuration Service... Azure StockTrader Home Page Configuration Service Welco... StockTrader Local Outlook Web App

Microsoft .NET

SERVICE CONFIGWEB

Configuration Service Version 5.0.0

Welcome Login/Logout Configuration Service Logs Hosted Services Connected Services Connections Docs Glossary About

Discussion Forum Download on MSDN

CONNECTED SERVICES

My First Service Client

Version 1.0

Service Hosted

Windows Server 2008 R2 with .NET Framework v4.0.30319

Windows Server 2008 R2

Back to Top

Add Service

This page shows definitions established in client applications that describe service endpoints this client connects to. Services that implement the Configuration Service can connect to other services that also implement the Configuration Service, or any service via WCF SOAP, WS-\* or REST capabilities. These might include services running on other platforms such as Java or PHP. Likewise, from a service hosting standpoint, any client you allow can connect to your service, no matter whether it implements the Configuration Service or not, and no matter what platform it runs on.

Virtual Host ID	Service Name	Service Type	Service Contract	Client Configuration	Edit/Delete
-----------------	--------------	--------------	------------------	----------------------	-------------

Return to Home Page

Created with Microsoft Visual Studio and the Microsoft .NET Framework 4.0

Copyright 2011, Microsoft Corporation

Microsoft Visual Studio



- Click **Add Service**.

Configuration Service: Connected Service Definitions - Windows Internet Explorer

http://localhost/ConfigWeb/UpdateConnectedService.aspx?

File Edit View Favorites Tools Help

Management Portal - Windo... Azure Configuration Service... Azure StockTrader Home Page Configuration Service Welco... StockTrader Local Outlook Web App

**SERVICE CONFIGWEB** Configuration Service Version 5.0.0

Welcome Login/Logout Configuration Service Logs Hosted Services Connected Services Connections Docs Glossary About

Discussion Forum Download on MSDN

**MANAGE CONNECTED SERVICE DEFINITIONS**

My First Service Client  
Version 1.0  
Service Host  
Windows Server 2008 R2 with .NET Framework v4.0.30319  
Windows Server 2008 R2

Back to Top

**Instructions**

This page enables you to establish connections to remote services your application will utilize. A primary service implements the Configuration Service; a generic service does not. For example, a generic service might be one written in Java, PHP or .NET. When establishing connections to **primary** services, you will simply connect to the configuration service of the remote host, and a list of available services (contracts) will be returned. You can also establish connection to **generic** services simply by providing the endpoint to the service, and filling out the requested details. A service implementing the Configuration Service can always be treated as a generic service, since the Configuration Service imposes no requirement on the language clients are developed in, or whether they also implement the Configuration Service.

**Connected Service Type**

☒ Primary Connected Service  
☐ Generic Connected Service

**Remote Configuration Service Details - Get Connected!**

Address to Configuration Service:

Client Configuration Name:

Selected Binding Type:

User Id:  (see information below left)

Password:

The credentials above are used by the Configuration Service to send/receive notifications between hosts and subscribed clients. Hosts can be configured to automatically return these credentials for any client attempting to subscribe, in which case nothing needs to be entered before connecting, the credentials will be returned. If the remote host is not configured to automatically provide these credentials, you will need to enter valid credentials to the remote host above, with the userid supplied having at least 'Connected Service Rights' as defined by the remote host in its users database.

Get Remote Services!

[Return to Connected Services List](#)

Created with Microsoft Visual Studio and the Microsoft .NET Framework 4.0  
Copyright 2011, Microsoft Corporation

To create the Connected Service definition, ConfigWeb uses this page to make a request to the remote service host application created in the first part of this walkthrough. The request is to the remote service's configuration service, which will send back information about its available services.

- Type in the address to the remote host: **http://localhost/hello/config.svc**.

#### Note

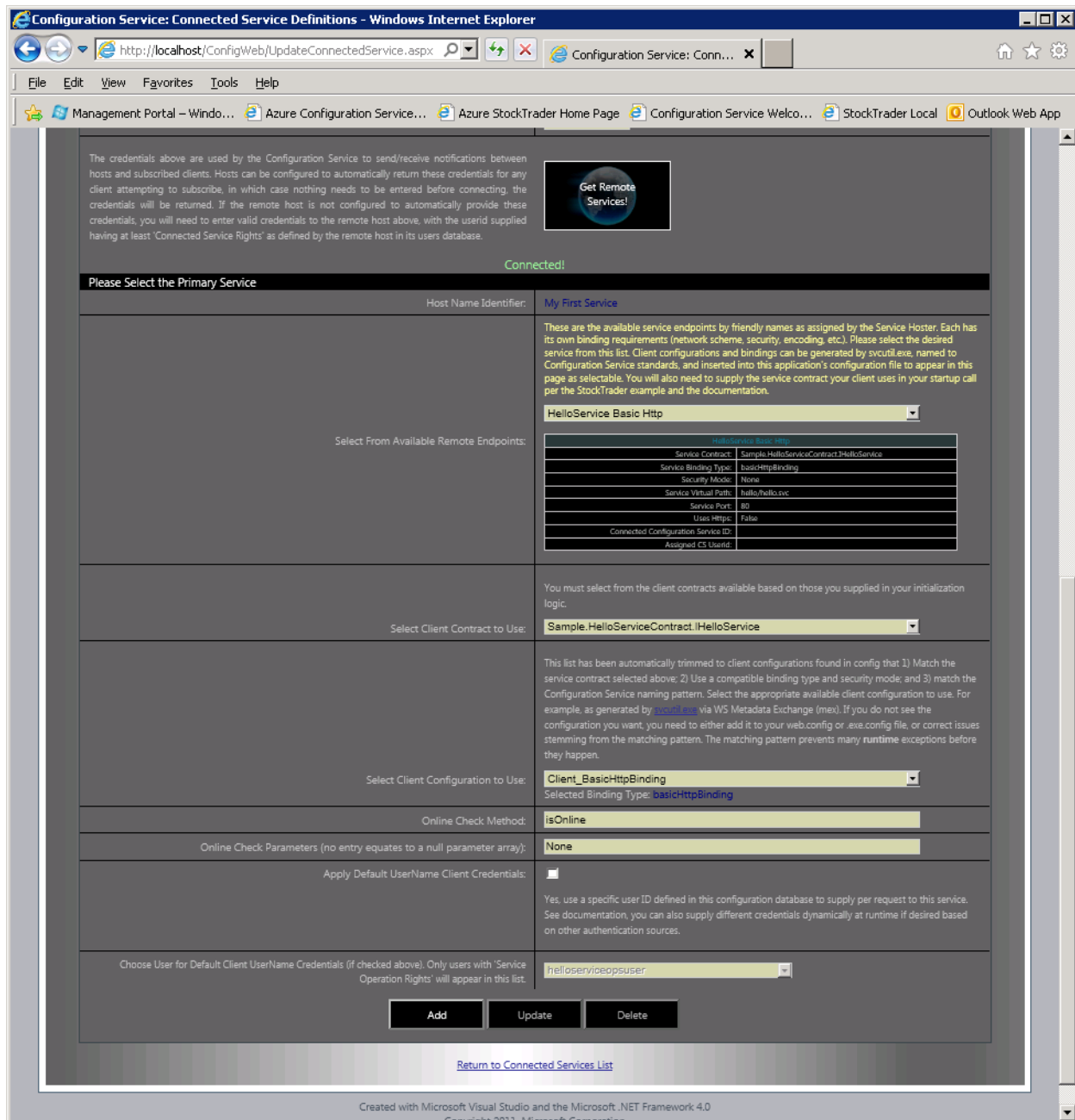
ConfigWeb is just a user interface that only communicates (via WCF directly with the root service you logged into). So ConfigWeb is actually not going to make this request to the Hello Service host; rather it is going to tell the **client** application (currently being configured) to make this request. So, in fact, you might be running ConfigWeb anywhere (from a hotel in another country, for example), and ConfigWeb itself does not even need connectivity to the ServiceHost application, as long as the Service Client application has connectivity to the remote service it is going to use (the endpoint above), and ConfigWeb can login (has connectivity to) the Client application. It may take some getting used to, but remember the concept of linked services. This concept is what allows cross service domain navigation, with each connected service that implements the Configuration Service also serving as a configuration gateway, routing Configuration operations through a (potentially complex) network of connected services. See the section on allowing linked services later; as you have to explicitly enable this for client domains to specific service domains (it's a configuration setting called Allowed Link Service List). Also remember every configuration operation is authenticated independently, by each connected service, against its own **Users** store. So only valid administrator credentials the ConfigWeb login will allow configuration of connected services down the network tree, if those credentials are also valid for that remote connected service domain.

- You do not need to enter any credentials to get the available services from the remote host. Leave these blank, as they are returned automatically as a 'connected service' user.

#### Note

By default, the template has set the Configuration Setting 'Auto Return Connected Service UserId/Password To New Clients' as true. This means any client can make a request to subscribe to the remote hosted services from the service host domain. It does not mean any client can actually use the services, since security on the business service operations is completely independent (as with StockTrader 5.0). However, subscribing creates a low-privilege credential for the client to send and receive notifications to the remote host. To completely lock down your service domain, you will set the setting 'Auto Return Connected Service UserId/Password To New Clients' to false in ConfigWeb, and then you must supply the credentials (with the correct password) for a user account with exactly Connected Service rights to the remote service domain or you will not be able to subscribe and create the definition.

- Click **Get Remote Services!**



You will now see the services that the remote service hosts. You can select one of two services (one is on an http binding, the other a net.tcp binding).

- Keep the default selections to the http-based service. Keep other default as well but note them and the text explanation.
- Click **Add**.

We have now created essentially a subscription to this service host. Client Definitions are in fact based on specific endpoints, so we only have a definition to the http-based endpoint, not the net.tcp endpoint at this point.

- So, you can repeat this process per below to create a second Connected Service definition to the other endpoint (net.tcp) the Service domain hosts. You should do this now, so you only have to use the Add Connection Point page once. When you create your second definition, the list will be automatically pared down to just the endpoint you have not subscribed to yet.
- Click on the top-level menu item **Connected Services**.
- Click the **Add Service** button again.
- Again supply the login URL to the remote service's configuration service endpoint:  
**http://localhost/hello/config.svc.**
- Click **Get Remote Services!**.

Configuration Service: Connected Service Definitions - Windows Internet Explorer

http://localhost/ConfigWeb/UpdateConnectedService.aspx

File Edit View Favorites Tools Help

Management Portal - Windo... Azure Configuration Service... Azure StockTrader Home Page Configuration Service Welco... StockTrader Local Outlook Web App

Selected Binding Type: **basicHttpBinding**

User Id: **csUserMyFirstSe** (see information below left)

Password: **\*\*\*\*\***

The credentials above are used by the Configuration Service to send/receive notifications between hosts and subscribed clients. Hosts can be configured to automatically return these credentials for any client attempting to subscribe, in which case nothing needs to be entered before connecting, the credentials will be returned. If the remote host is not configured to automatically provide these credentials, you will need to enter valid credentials to the remote host above, with the userid supplied having at least 'Connected Service Rights' as defined by the remote host in its users database.

**Get Remote Services!**

**Please Select the Primary Service**

Host Name Identifier: **My First Service**

Select From Available Remote Endpoints:

HelloService netTop	
Service Contract:	Sample.HelloServiceContract.IHelloService
Service Binding Type:	customTcpBinding
Security Mode:	unknown
Service Virtual Path:	hello/hello.svc
Service Port:	808
Uses Https:	False
Connected Configuration Service ID:	
Assigned CS Userid:	

Select Client Contract to Use: **Sample.HelloServiceContract.IHelloService**

Select Client Configuration to Use: **Client\_TcpBinding**

Online Check Method: **IsOnline**

Online Check Parameters (no entry equates to a null parameter array): **None**

Apply Default UserName Client Credentials: ☐

Choose User for Default Client UserName Credentials (if checked above). Only users with 'Service Operation Rights' will appear in this list. **hellouseropsuser**

**Add Update Delete**

[Return to Connected Services List](#)

Created with Microsoft Visual Studio and the Microsoft .NET Framework 4.0  
Copyright 2011, Microsoft Corporation

Visual Studio

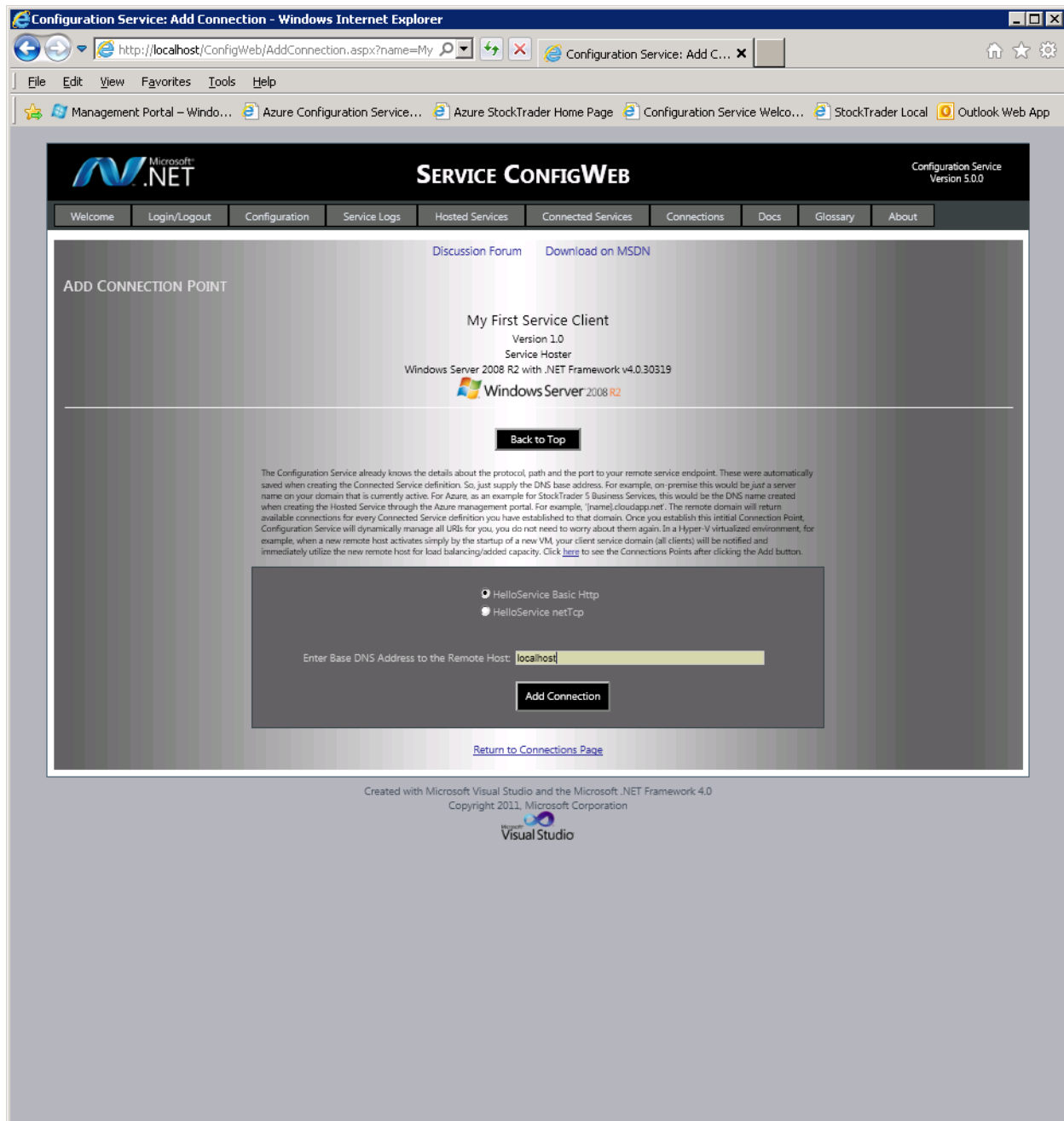
- The service binding this time is a custom tcp binding. Because of this, there are several client configurations to select from, and the default selection will NOT BE CORRECT.
- Change the client definition to Client\_TcpBinding as shown above.
- Click **Add**.

Now we just need to add an initial URI (Connection Point) to the remote service domain, telling the Client Application where, exactly the remote service domain lives.

- Click on the Connections top-level menu item.



- Click Add Connection.
- You will see you automatically have the two services (endpoints) you can choose. Since you have definitions to both already established, you will actually have to only select one (either one) and do this step once.



- Enter **localhost** as the base DNS address.
- Click Add Connection button.
- Next, after the connection has been successfully added, click on the **Connections** top-level menu item again.

Configuration Service: Connection Points - Windows Internet Explorer

http://localhost/ConfigWeb/ConnectionPoint.aspx

File Edit View Favorites Tools Help

Management Portal - Windo... Azure Configuration Service... Azure StockTrader Home Page Configuration Service Welco... StockTrader Local Outlook Web App

Microsoft .NET SERVICE CONFIGWEB Configuration Service Version 5.0.0

Welcome Login/Logout Configuration Service Logs Hosted Services Connected Services Connections Docs Glossary About

Discussion Forum Download on MSDN

### SERVICE CONNECTION POINTS

My First Service Client  
Version 1.0  
Service Host  
Windows Server 2008 R2 with .NET Framework v4.0.30319

Windows Server 2008 R2

Back to Top

View Services View Clients

Add Connection

This page shows the actual WCF connections between the client service domain and the remote service domain for any business service endpoints, as maintained by the Configuration Service. If the remote service domain is running behind any type of NAT/external load balancer, as is the case with Windows Azure service domains, then you will see just one connection for this service domain (with a corresponding icon), just as the client service domain sees - even though in reality there may be many nodes servicing this endpoint behind the NAT load balancer. On private cloud deployments, unless you are also using a NAT to translate addresses, you will see every node as a distinct connection point, as clients communicate directly to each load-balanced node, with load balancing and request-level failover provided by the Configuration Service itself with no NAT required. While the Node Map and Service Map pages display all nodes even if behind a NAT, this page is important in that it shows the view the client service domain is actually operating against. As such, you can see the detail on the WCF client being utilized, including the endpoint address, the name of the client definition as defined in configuration, the binding type, binding configuration name, and security mode the client is utilizing.

If a connection point below shows red, hover over it to see the actual exception the client is receiving. If a connection below shows as a grey-colored server, a timeout was exceeded (also indicating an issue).

#### Connections To Host: My First Service

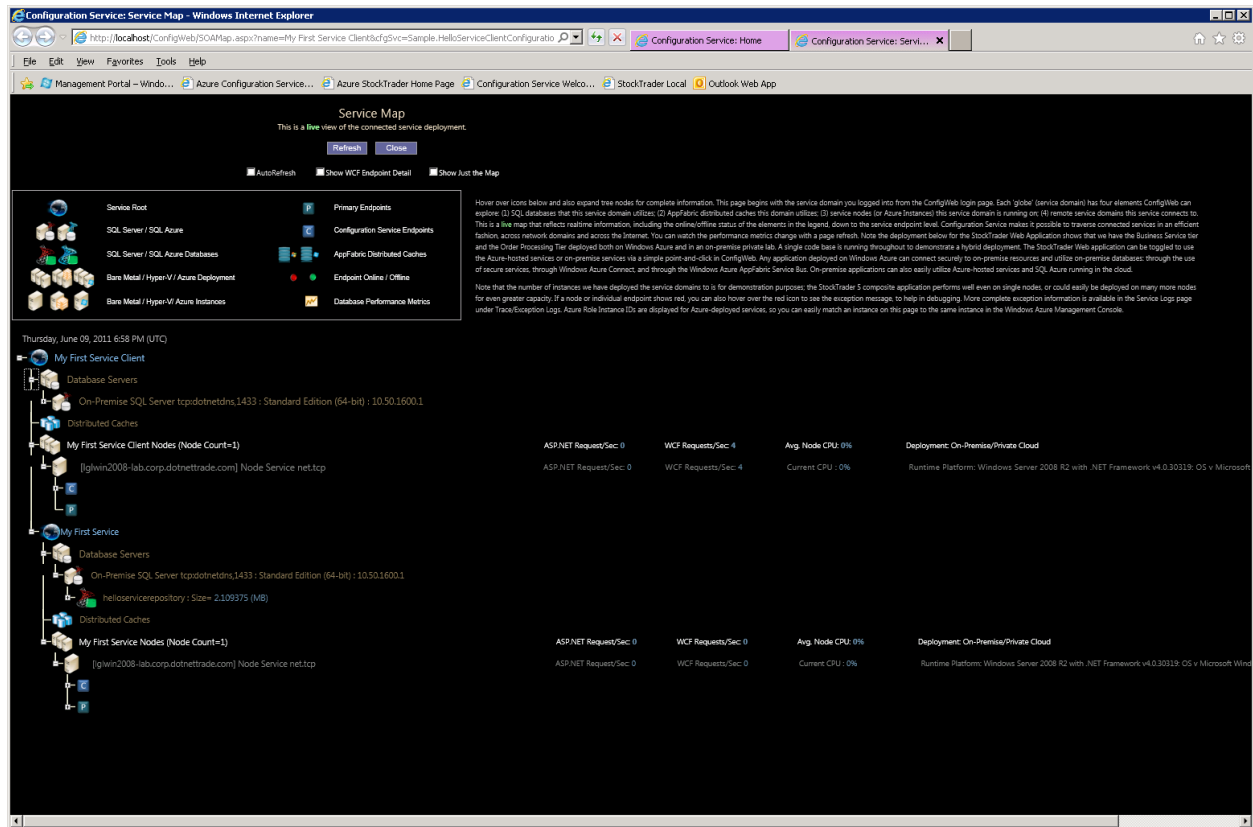
View/Configure This Service Host's Connection Points

Purge All Connections to This Service Host

Remote Address	Client Details	Service Status	Remove
http://glwin2008-lab.corp.dotnettrade.com/hello/hello.svc	<b>HelloService Basic Http</b> Client Configuration: Client_BasicHttpBinding Service Contract: Sample.HelloServiceContract.IHelloService Binding Configuration: Client_BasicHttpBinding Binding Type: basicHttpBinding Security Mode: None Endpoint Behavior: None Assigned		<a href="#">Delete</a>
net.tcp://glwin2008-lab.corp.dotnettrade.com/hello/hello.svc	<b>HelloService netTcp</b> Client Configuration: Client_TcpBinding Service Contract: Sample.HelloServiceContract.IHelloService Binding Configuration: Client_TcpBinding Binding Type: netTcpBinding Security Mode: None Endpoint Behavior: None Assigned		<a href="#">Delete</a>

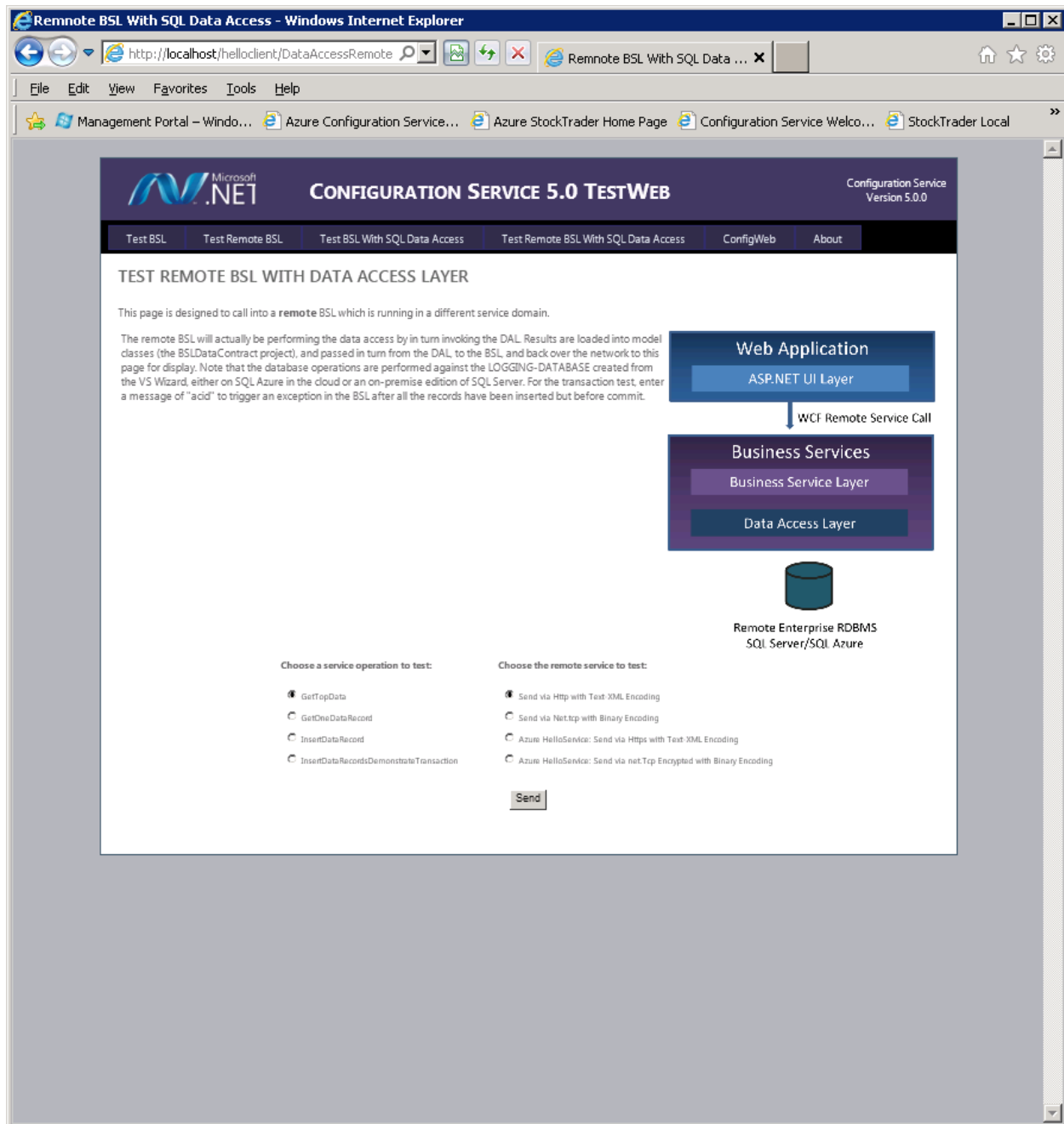
- If either shows red when you see the page above, hover over the server icon for the exception message. Always check Service Logs for more complete exception information.
- Now, navigate back to the ConfigWeb home page by clicking **Back to Top**.
- Bring up the Service Map page.





Now you can see the remote connected service(s) and the network tree in the live map view, as above.

- You can now go back to the Client application Web site pages. Go to the Test Remote BSL with SQL Data Access Page.



You can activate the various methods and explore the code behind pages. Try attaching a remote Visual Studio debug session to the appropriate W3P worker process application pool using the Service Host solution, and set a breakpoint on the Business Service implementation class (in one of the methods), and you will see it hit when the client makes the remote call.

## Allowed Linked Services List

If you change the **Host Name Identifier** (first textbox in the Wizard) for a solution from the default the Wizard fills in, it is important to note that to be able to navigate across service domains in ConfigWeb two conditions have to be met:

1. There must be a valid userid/password setup as an administrator in **both** the client and the remote connected service via the ConfigWeb Users management page. These need to be identical.
2. You must explicitly enable linkage via editing (via ConfigWeb) the **Allowed Linked Service List** inherited setting of the client(s). Here you will add the Host Name Identifier of the remote service, which means you will allow, when clicking Select in ConfigWeb, for the client to pass your login credentials to the remote service for authentication against its own registered Users table.

In this way you can create configuration gateways between elements in a composite application, even if they are hosted in geographically disperse regions, or in hybrid cloud environments. If connecting to services over the Internet, you always want to make sure the Configuration Service endpoint is only hosted and active on **[https/ssl](#)** (or an encrypted net.tcp endpoint). In this way, communication is secured. The Wizard only creates secure/encrypted endpoints for Windows Azure host types, so this is already done when selecting Azure host types in the wizard. In addition, every service operation is authenticated independently by the remote domain to ensure the credentials have the proper rights to perform the operation.

You can review the *Configuration Service 5.0 Technical Guide* for more details on the Configuration Service and some of the more advanced features and scenarios. You should also visit the forum at <http://msdn.microsoft.com/stocktrader> to ask any questions or offer feedback. The Configuration Service is itself a sample application (although geared at providing re-usable libraries); and illustrates use of WCF, .NET and the Azure SDKs. The entire source for the Configuration Service is provided with the StockTrader 5 sample application. It is supported as other samples are, with an online support forum; but treated as “customer code” from a Microsoft enterprise support standpoint.

## Additional Exercises

### Adding new Monitored Databases

You can use ConfigWeb to add a new monitored database in the Service Map page. To do this for the Logging-Database, we would select Edit for inherited settings in the home page of ConfigWeb.

Configuration Service: Configuration Keys - Windows Internet Explorer

http://localhost/ConfigWeb/Config.aspx?name=My First Service Client&cfgSvc=Sample

File Edit View Favorites Tools Help

Management Portal - Windo... Azure Configuration Service... Azure StockTrader Home Page Configuration Service Welco... StockTrader Local Outlook Web App

Windows Server 2008 R2

Back to Top

Inherited settings are simply application global settings inherited from a base Configuration Service settings class. Custom settings are application-specific settings the developer has defined for the specific application. These can be any settings you might otherwise have defined in the <appSettings> section of a .NET Web.config or App.config file. While stored in the central configuration database, these settings are cached locally in memory, and used just like they came from a config file (although no code is necessary to populate them into local variables). The in memory cache means access is as fast as accessing a local static variable. But, Configuration Service keeps all nodes in sync when configuration updates are made on live systems - without having to deploy new application configuration files across nodes, and stop/re-start running nodes.

Basic Detailed Advanced

Define Keys

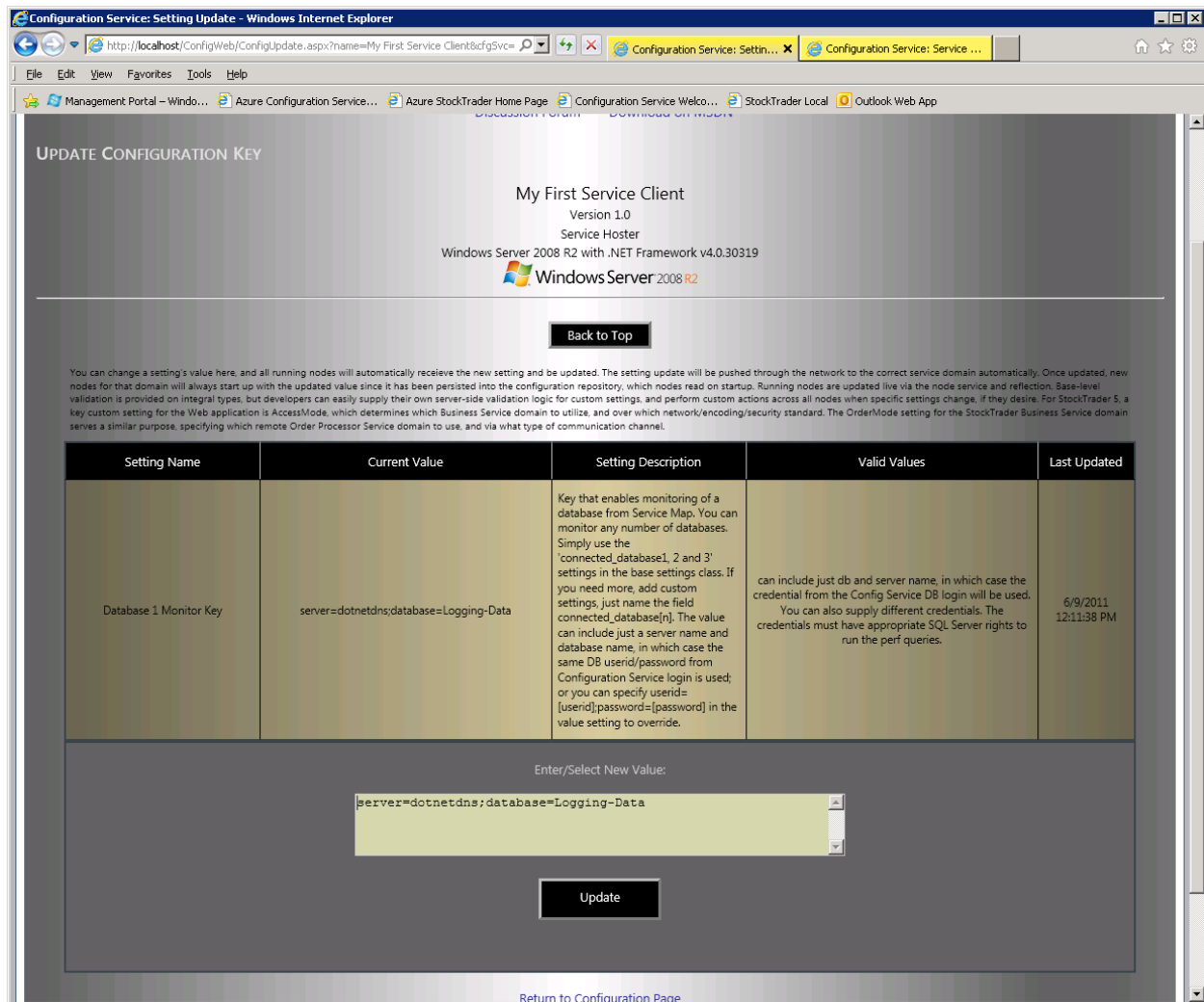
Target Settings Group: Host:My First Service Client  
Configuration Service Inherited Settings

Setting Name	Current Value	Description	Change Value
HostNameIdentifier	My First Service Client	The name of the service domain. This is a very important setting, and should be a unique name within a set of connected services. Once set using the repository create tool, it should not be changed in ConfigWeb. It is stamped on other records within the configuration database and the logic is not yet implemented to ripple this change through these records with the appropriate update logic. That will need to be a 5.1 enhancement.	Read Only
Configuration Service Name Identifier	Sample.HelloServiceClientConfigurationImplementation.ConfigurationService	The name identifier of the configuration service for this host.	Read Only
Allowed Linked Service List	My First Service:My First Service Web Role:My First Service Worker Role:My First Service Client	You will need to enter a list of Host Name Identifiers (separated by /) here to explicitly enable navigation to remote services via ConfigWeb. While authentication is performed by each service for every configuration service call, it is still necessary to list allowable navigation links here, telling a service it is ok to attempt to navigate with the login credentials supplied on ConfigWeb login. The list (or single name) is the Host Name Identifier of the remote host you want to enable for navigation. Obviously, that host must be also setup with the correct administrator login credential.	Change Value
Log to Database	true	Whether to log exceptions/trace to database. If true, you can view the logs from ConfigWeb. You need to setup the logging database, one table, on SQL Server/SQL Azure. You can have many services write to the same log db, if desired.	Change Value
NodeActiveServiceID	2	This selects the active Node Endpoint for nodes to communicate with each other for cluster-wide operations	Change Value
Top Query Number in SQAMap	5	To change the number of queries returned in the Service Map page.	Change Value
Database 1 Monitor Key	none	Key that enables monitoring of a database from Service Map. You can monitor any number of databases. Simply use the 'connected_database1, 2 and 3' settings in the base settings class. If you need more, add custom settings, just name the field 'connected_database{n}'. The value can include just a server name and database name, in which case the same DB userid/password from Configuration Service login is used, or you can specify userid= [userid];password=[password] in the value setting to override.	Change Value
Database 2 Monitor Key	none	Entry to enable DB monitoring of business database from Service Map view.	Change Value
Database 3 Monitor Key	none	Entry to enable DB monitoring of business database from Service Map view.	Change Value
Connected Cache 1	none	Connected Cache Name	Change Value
		For Windows Azure AppFabric: Authorization token string as obtained from Azure AppFabric Management Portal. For Windows Server AppFabric, enter	

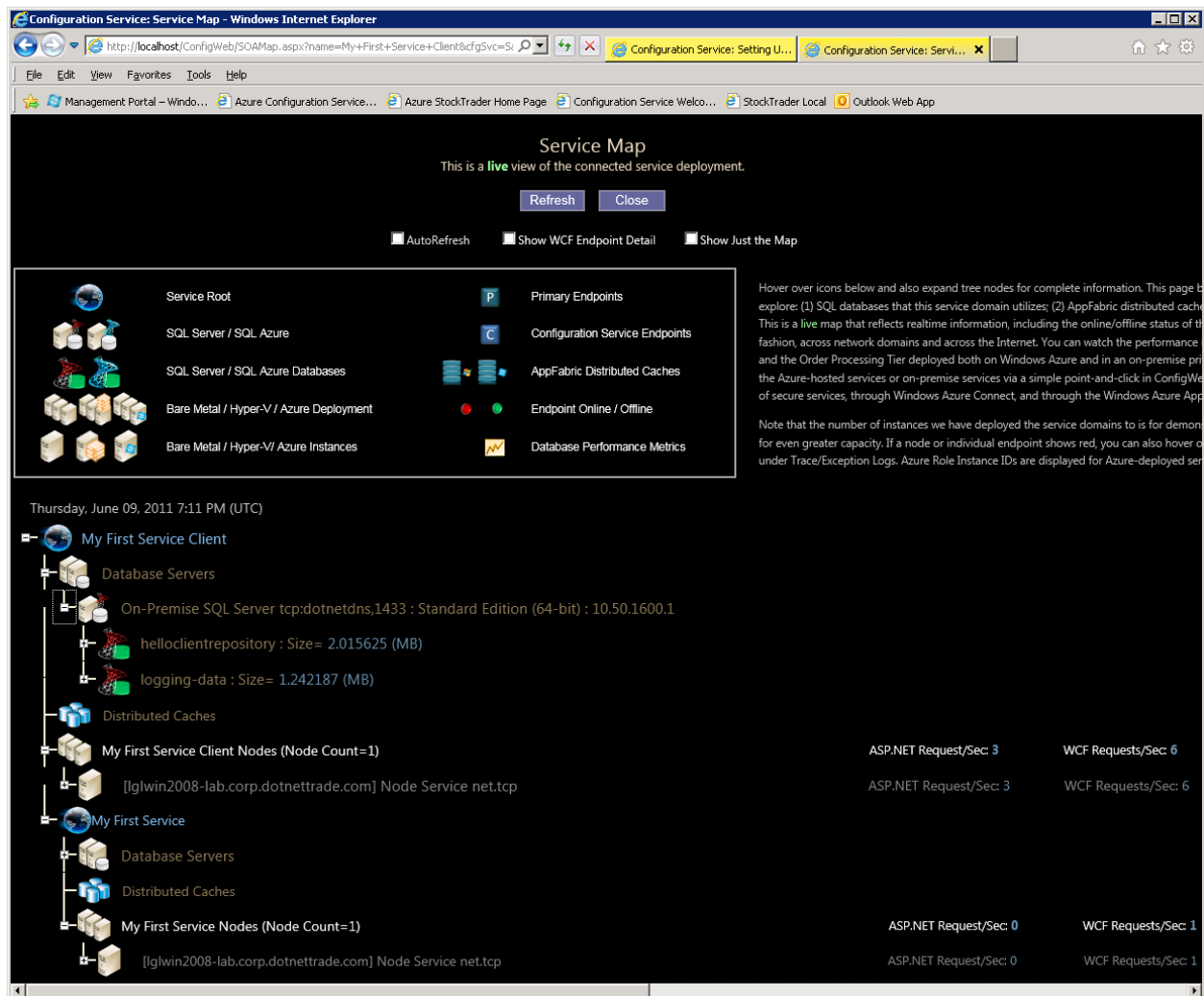
- Choose Change Value for the **Database 1 Monitor Key**

#### Note

There are three database monitor keys provided as inherited settings. You can always add more, just follow the naming convention. You add new settings by using the Define Keys button: for each setting you need to create a static variable to hold the setting in the Settings class. Do not alter the base class, just use the Settings class generated via the Wizard, which inherits from the base. You can choose to group new settings for display in either group (inherited or custom), and create new group names.



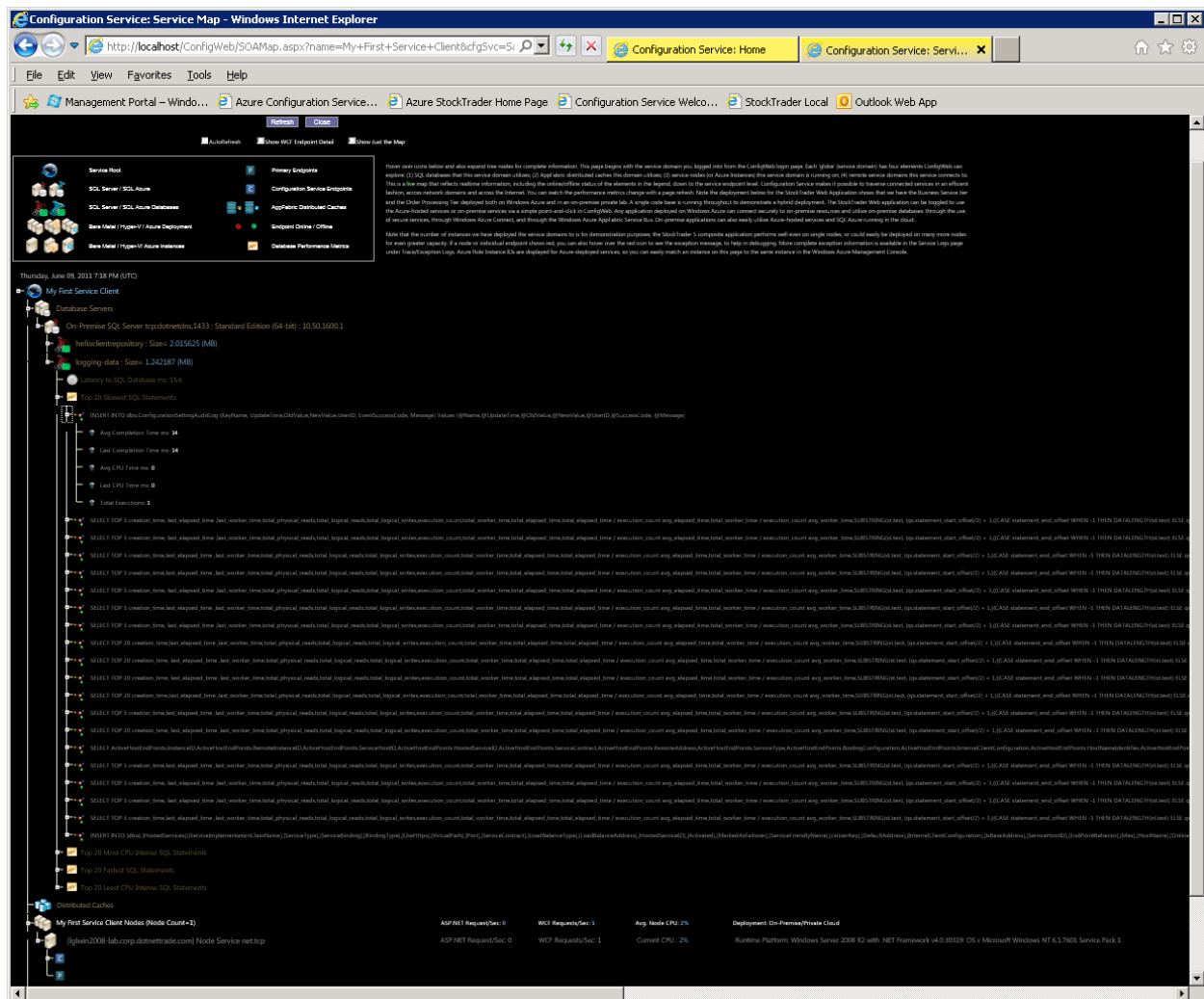
- Enter the database name and server name as above and click Update. Make sure you type 'server=[servername];database=Logging-Data'. (Your server name will be '.\SqlExpress' for SQL Express).
- By default, if no userid/password is provided, the ConfigAdmin repository credentials are used.
- Go to the Service Map view now.



- You can now see the logging-data database in the view.

## Showing Data for More Than Five Queries

To show performance metric for more queries, simply change the inherited setting for 'Top Query Number in SOAMap'.



## Primary vs. Generic Services

The term Primary Service is used to denote a service that implements the Configuration Service, and the term Generic Service denotes one that does not (.NET, Java, PHP or any service implementing the WS standards for interoperable services). There are a few important notes about the Configuration Service around this topic.

First, all services created via the ConfigWeb/Configuration Service are just standard WCF services—they can be called by any type of client such as Java, PHP, etc. as long as you choose an interoperable binding such as basicHttpBinding or the ws2007HttpBinding. If you choose a .NET-only binding such as net.tcp, .NET clients not implementing the Configuration Service can of course still use your service since it's just a standard .NET WCF service.

Second, you can host services that are not managed/created via the Configuration Service even when implementing the Configuration Service: you can put any logic into your startup host logic you want to

create your own ServiceHosts and WCF endpoints, outside of the Configuration Service and not stored and managed in the repository.

Third, you can define Generic Connected Service definitions to hosts that do not implement the Configuration Service (or even hosts that do). In this case, the Configuration Service will still manage the WCF channels and Client class. You can define a virtual host name to logically group Generic Connection Points, to load balance across any type of host running in a cluster, for example WebSphere endpoints within a WebSphere cluster. The client class will automatically still provide load balancing and failover across the endpoints, but you need to manually enter the cluster endpoints in ConfigWeb as individual, multiple Connection Points to different addresses (servers in the Websphere, Tomcat, or other type of application server cluster). To do so, you simply create one Connected Service Definition and create a virtual host name (after choosing Generic as the Connected Service type). Then, create the definition in the Add Connected Service page. Next, use Add Connection Point to add any number of unique endpoints to the servers in the cluster. They will show up in Service Map, and you can optionally supply an online check method (any simple method the service exposes) to do online checking of the host for failover and display in Service Map. You can investigate this by using the template Client application to create a **Generic Service** definition to the Hello Service—as if its host did not actually implement the Configuration Service.

## A Client that Uses Many Connected Services

Configuration Service treats a composite application including remote connected services as a tree structure. So, there is no limit to the number of different Connected Service Definitions to completely different host domains you can create from a single client application. The different hosts (and their nodes) will be displayed in the home page in the Remote Services table; and in the Service Map page as just other nodes in the tree. You can explore this using StockTrader and the on-premise .NET StockTrader Web application. If you also deploy your Azure Business Services (and Order Processor Service) to Windows Azure, you can create a Connected Service Definition to both the on-premise and Azure business services. They will be treated as completely distinct hosts by the Configuration Service, since they are in different service domains. The key to making this work is that services in different domains should be given unique Host Name Identifiers when created. The Host Name Identifier is used to locate/track and manage the connections to remote services, and target configuration operations to the correct connected service in the network tree if using Linked Services.

## Additional Topics to Be Added Later

This completes this walkthrough. We have not explored all of ConfigWeb and the Configuration Service yet, however. The key other tasks you may well want to accomplish are to change the hosted service endpoints to use your bindings, and your custom security settings. You would use ConfigWeb Hosted Services to do so, including adding, de-activating , updating or deleting endpoints. These are topics that we will explore soon in an updated document (and maybe video). For now see the *Configuration Service*



*5.0 Technical Guide* and the *.NET StockTrader 5.0 Installation and Configuration* guide. You can also use the StockTrader Discussion Forum for information and to ask questions.