Impersonating the Client

.NET Framework 4.5

The Impersonation sample demonstrates how to impersonate the caller application at the service so that the service can access system resources on behalf of the caller.

This sample is based on the Self-Host sample. The service and client configuration files are the same as that of the Self-Host sample.

Note:
The setup procedure and build instructions for this sample are located at the end of this topic.

The service code has been modified such that the `Add` method on the service impersonates the caller using the OperationBehaviorAttribute as shown in the following sample code.

```
[OperationBehavior(Impersonation = Imperson
public double Add(double n1, double n2)
{
    double result = n1 + n2;
    Console.WriteLine("Received Add({0},{1}
    Console.WriteLine("Return: {0}", result
    DisplayIdentityInformation();
    return result;
}
```

As a result, the security context of the executing thread is switched to impersonate the caller before entering the `Add` method and reverted on exiting the method.

The `DisplayIdentityInformation` method shown in

the following sample code is a utility function that displays the caller's identity.

```
static void DisplayIdentityInformation()
{
    Console.WriteLine("\t\tThread Identity
         WindowsIdentity.GetCurrent().Name)
    Console.WriteLine("\t\tThread Identity
         WindowsIdentity.GetCurrent().Imper
    Console.WriteLine("\t\thToken
         WindowsIdentity.GetCurrent().Token
    return;
}
```

The `Subtract` method on the service impersonates the caller using imperative calls as shown in the following sample code.

```
public double Subtract(double n1, double n2
{
    double result = n1 - n2;
    Console.WriteLine("Received Subtract({0
    Console.WriteLine("Return: {0}", result
Console.WriteLine("Before impersonating");
DisplayIdentityInformation();

    if (ServiceSecurityContext.Current.Wind
        ServiceSecurityContext.Current.Wind
    {
        // Impersonate.
        using (ServiceSecurityContext.Curre
        {
            // Make a system call in the ca
            // on the system resource are e
            Console.WriteLine("Impersonatin
            DisplayIdentityInformation();
        }
    }
    else
    {
        Console.WriteLine("ImpersonationLev
    }

Console.WriteLine("After reverting");
DisplayIdentityInformation();
    return result;
}
```

Note that in this case the caller is not impersonated for the

entire call but is only impersonated for a portion of the call. In general, impersonating for the smallest scope is preferable to impersonating for the entire operation.

The other methods do not impersonate the caller.

The client code has been modified to set the impersonation level to Impersonation. The client specifies the impersonation level to be used by the service, by using the TokenImpersonationLevel enumeration. The enumeration supports the following values: None, Anonymous, Identification, Impersonation and Delegation. To perform an access check when accessing a system resource on the local machine that is protected using Windows ACLs, the impersonation level must be set to Impersonation, as shown in the following sample code.

```
// Create a client with given client endpoi
CalculatorClient client = new CalculatorCli

client.ClientCredentials.Windows.AllowedImp
```

When you run the sample, the operation requests and responses are displayed in both the service and client console windows. Press ENTER in each console window to shut down the service and client.

Note:

The service must either run under an administrative account or the account it runs under must be granted rights to register the http://localhost:8000/ServiceModelSamples URI with the HTTP layer. Such rights can be granted by setting up a Namespace Reservation using the Httpcfg.exe tool.

Note:

On computers running Windows Server 2003, impersonation is supported only if the Host.exe application has the Impersonation privilege. (By default, only administrators have this permission.) To add this privilege to an account the service is running as, go to **Administrative Tools**, open **Local Security Policy**, open **Local Policies**, click **User Rights Assignment**, and select **Impersonate a Client after Authentication** and double-click **Properties** to add a user or group.

To set up, build, and run the sample

1. Ensure that you have performed the One-Time Setup

Procedure for the Windows Communication Foundation Samples.

2. To build the C# or Visual Basic .NET edition of the solution, follow the instructions in Building the Windows Communication Foundation Samples.

3. To run the sample in a single- or cross-machine configuration, follow the instructions in Running the Windows Communication Foundation Samples.

4. To demonstrate that the service impersonates the caller, run the client under a different account than the one the service is running under. To do so, at the command prompt, type:

```
runas /user:<machine-name>\<user-name
```

You are then prompted for a password. Enter the password for the account you previously specified.

5. When you run the client, note the identity before and after running it with different credentials.