ECO - Remote persistence

Table of Contents

Introduction 1

Prerequisites and goals 2

Writing the client application 3

Writing the persistence server 6

A console based server 6

An IIS hosted server 8

A Windows service based server 11

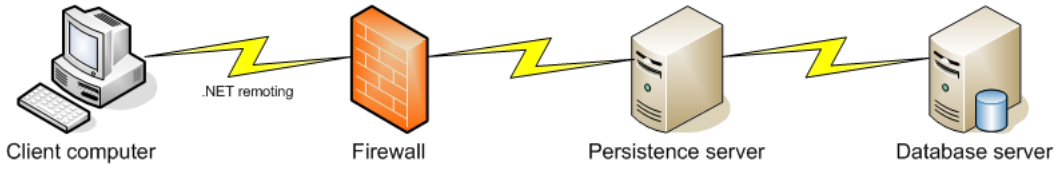
Further reading 16

Index a

1 Introduction

ECO's remote persistence mechanism enables ECO driven client applications to update a data store via an intermediate software application. This provides the following benefits:

- The connection to the persistence server is made via .NET remoting; allowing the developer to utilize numerous benefits such as encryption, authentication, different transport mechanisms (direct sockets, HTTP, etc).
- The ability to protect your database server from access via external networks such as the Internet.
- The ability to switch the data store to a different machine, or even to a different type (Oracle to SQL Server) without having to update the client software.
- A centralized process for updating the data store allows ECO to optionally track changes and then provide a list of modifications to clients that request it. This provides client applications with a way of ensuring data presented to the user is up to date without having to waste lots of network bandwidth refreshing data that has not changed.



2 Prerequisites and goals

Prerequisits

To successfully follow this document the user should be familiar with the following:

- Creating an ECO package containing business classes that compile into a DLL.
- Creating an ECO WinForm application that uses the business classes DLL.
- Using an ExpressionHandle to retrieve object instances from the EcoSpace, and displaying those objects in a DataGrid.
- Using the EcoListActions and EcoGlobalActions to configure Button controls to achieve the following
  - Add a new object to the object list in an ExpressionHandle,
  - Delete an object listed in a DataGrid,
  - Update the database.

The steps to performing these tasks will be described but not in great detail. If you are not familiar with these subjects you should consider reading the "ECO Jump Start" document.

Goals

By the end of this document you will be familiar with the steps required to:

- Configure an ECO client application to use a remote persistence server.
- Create a console application that acts as an ECO remote persistence server.
- Create an ECO remote persistence server and expose it over HTTP by hosting it in IIS.
- Create a Windows service that contains an ECO remote persistence server.

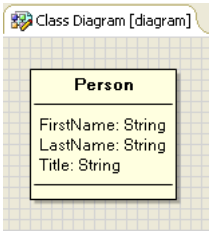
3 Writing the client application

The steps involved in writing the ECO client application are very simple. In fact the client application is merely a standard ECO application that uses a PersistenceMapperClient component instead of a database or XML based persistence mapper. The following steps describe how the client application was written in the following demo project.

```
Demos\{Language}\RemotePersistence\Client\RemotePersistenceClient
```

The first and most important step is to create the business classes so that they compile to binary that is completely separate from the client application, the business classes DLL will be used by the client application and also by the server to provide the information required to perform persistence operations.

1. Create a new "ECO package in a DLL" (C#) or "ECO package in a package" (Delphi).
2. Create one or more classes. In the example project a single class "Person" was created with three UML attributes; Title, FirstName, and LastName.
3. Save and compile the project.



Next a simple ECO WinForm application will be created to act as the client application for the remote persistence server.

4. Create a new ECO WinForm application.
5. Add the business classes DLL to the References list for the project.
6. Compile the project to create the binary files for ECO to read to obtain model information.
7. In the EcoSpace click the "Select Packages" icon.

8. Add the business classes package and remove the default package.
9. In the project manager delete the source file and EcoPkg file for the default ECO package.

Next some GUI will be created to display a list of objects from the data store.

10. Add an ExpressionHandle to the WinForm and set its RootHandle to rhRoot.
11. Set its "Expression" property to the following

```
Person.allInstances
```

12. Add a DataGrid to the WinForm, and set its DataSource to the ExpressionHandle.
13. Add a button to the form to add new instances of Person to the form.
14. Set its "EcoListAction on EcoListActions" property to "Add".
15. Set its "EcoRootHandle on EcoListActions" property to reference the ExpressionHandle.
16. Add another button to delete the Person instance selected in the DataGrid.
17. Set its EcoListAction to "Delete" and its RootHandle to the ExpressionHandle.
18. Set its "BindingContext on EcoListActions" to the DataGrid.
19. Add another button to update the database.
20. Set its "EcoAction on EcoGlobalActions" to "UpdateDatabase".

The GUI section is now complete. Running the application at this point will present you with an application in which the user may create and delete Person instances but is unable to save their changes because the "Update DB" button is disabled. This is because any ECO application with no persistence will still execute, but will operate as if all classes had been modeled as transient rather than persistent.

Next the application will be configured to use a remote persistence server.

21. Double-click the EcoSpace in the project manager to bring up the designer.
22. Add a PersistenceMapperClient to the EcoSpace, and ensure that the EcoSpace's "PersistenceMapper" property

references it.

23. In the "Url" property for the PersistenceMapperClient enter the following

```
tcp://localhost:1234/RemotePersistenceDemo
```

The demo application is now ready. Running the application will immediately result in the following error:

```
No connection could be made because the target machine actively refused it.
```

This is because there is currently no remote persistence server is running on the given URL.

4 Writing the persistence server

Because ECO remote persistence uses .NET remoting the persistence server application may be any type of .NET software project; WinForms, console, a Windows service, or a project that is accessed over the HTTP protocol via IIS.

The ECO client will serialize all persistence requests and delegate them to the persistence server. .NET communication channels may secured using authentication/encryption by hosting the service in IIS, there are also solutions to securing non-IIS hosted services too, but these are not subjects that will be covered in this document.

4.1 A console based server

This first server example will demonstrate how simple it is to create a persistence server hosted within a console application. A console based server is not really ideal as a real life solution as it would require a user to log into Windows before it could be started, however, it is the most simple type of persistence server project to develop and will clearly demonstrate the minimum steps required to satisfy the requirements of the ECO client.

1. Create a new Console application named "RemotePersistenceServerConsole".
2. Add the following DLLs to the "References" list of the project.
   - Eco.Handles
   - Eco.Interfaces
   - System.Runtime.Remoting
   - System.Runtime.Serialization.Formatters.Soap
   - The DLL generated for the business classes project
3. Add a new EcoSpace to the project.
4. Compile the application to provide ECO with the binaries required to obtain model information etc at design time.

Next the EcoSpace will be configured to use the business classes package. An instance of the EcoSpace is never created by the server, it is used only to provide model information to the EcoPersistenceMapperProvider.

5. Add a new EcoSpace to the project (File->New->Other->{Language}->New ECO Files).
6. On the EcoSpace choose the "Select Packages" icon at the bottom.

7. Ensure that the business classes package is listed in the "Selected UML Packages" list on the right of the form, and then click "OK".
8. Compile the application so that the new EcoSpace exists in the project's binary files.

Next an EcoPersistenceMapperProvider will be added to the project. This will provide the remotable persistence support to the project.

9. Add an EcoPersistenceMapperProvider to the project.
10. Set its "EcoSpaceType" property so that it references the EcoSpace just created.
11. Add a PersistenceMapperBdp to its design surface, then right-click it and select InterBase.
12. Add a BdpConnection to its design surface and configure its connection string to point to a blank database.
13. Click the "Generate Database" icon at the bottom.

Next the EcoPersistenceMapperProvider will be made remotable.

14. View the code for the EcoPersistenceMapperProvider and you will see a method that has been commented out, remove the comments so that the method is included in the project. In the Delphi language the commented out method declaration in the "interface" section must also be uncommented.

[Delphi]

```
class procedure TEcoPersistenceMapperProvider.RegisterTcpServer(Port: integer);
var
  provider: BinaryServerFormatterSinkProvider;
  props: IDictionary;
  chan: TcpChannel;
begin
  provider := BinaryServerFormatterSinkProvider.Create;
  provider.TypeFilterLevel := TypeFilterLevel.Full; // Needed for serializations
  props := Hashtable.Create;
  props['port'] := Port;
  chan := TcpChannel.Create(props, nil, provider);
  ChannelServices.RegisterChannel(chan);
  RemotingConfiguration.RegisterWellKnownServiceType(
    typeof(TEcoPersistenceMapperProvider),
    'TestServer1',
    WellKnownObjectMode.Singleton);
end;
```

[C#]

```
public static void RegisterTcpServer(int port)
{
    BinaryServerFormatterSinkProvider provider = new BinaryServerFormatterSinkProvider();
    provider.TypeFilterLevel = TypeFilterLevel.Full; // Needed for serializations
    IDictionary props = new Hashtable();
    props["port"] = port;
    TcpChannel chan = new TcpChannel(props, null, provider);
    ChannelServices.RegisterChannel(chan);
    RemotingConfiguration.RegisterWellKnownServiceType(
        typeof(EcoPersistenceMapperProvider),
        "TestServer1",
        WellKnownObjectMode.Singleton);
}
```

15. Change the text "TestServer1" to "RemotePersistenceDemo".
16. As instructed in the comments add the following to the uses/using clause of the file
    - System.Runtime.Serialization.Formatters
    - System.Runtime.Remoting, System.Runtime.Remoting.Channels
    - System.Runtime.Remoting.Channels.Tcp

Finally, the remoting service needs to be registered when the console application starts.

17. Add the following code to the "Main" method in the console project's source.

[Delphi]

```
EcoPersistenceMapperProvider.RegisterTcpServer(1234);
Console.WriteLine('Server started...');
Console.ReadLine();
```

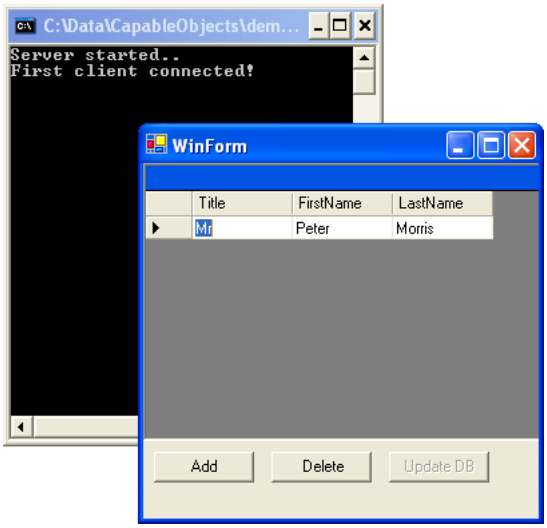
[C#]

```
EcoPersistenceMapperProvider.RegisterTcpServer(1234);
Console.WriteLine("Server started...");
Console.ReadLine();
```

Now run the console application and then the client application, your client application should now allow you to add data and update the database.

Extra credit

In the EcoPersistenceMapperProvider's constructor add a Console.WriteLine that outputs the text "First client connected!".



4.2 An IIS hosted server

The first notable benefit of hosting the remote persistence server within IIS is that the service will be available as soon as Windows has successfully started (assuming IIS is set to start automatically). Other benefits include:

- Secure communications via SLL.
- User authentication.

- If the server is a shared Internet host it is likely that this will be the only option for exposing remotable services.

To save time creating this application an ECO enabled Web Service will be created, the unwanted files will be deleted and then the project will be adapted to an IIS hosted .NET remoting service.

1. File->New->Other and then either "C# Projects" or "Delphi for .NET projects".
2. Select "ECO ASP.NET Web Service Application".
3. When prompted for a project name enter "RemotePersistenceServerWeb".
4. In the same settings window select IIS as the server.
5. Add the business classes DLL to the "References" list of the project.
6. Compile the application to create the project binaries required by the ECO design time editors.
7. Double-click the EcoSpace to open it in the designer.
8. Click the "Select Packages" icon at the bottom.

9. Remove the default package, and select the business classes package.
10. You may now delete the following unneeded files in the project manager as these are needed only for ECO web service applications.
    - Package_1 generated source file (.PAS / .CS)
    - Package_1.EcoPkg
    - EcoSpaceProvider
    - WebService1
11. In this case we do not need the RegisterTcpServer method in the EcoPersistenceMapperProvider file, if you wish then you may deleted this method which is commented out by default.
12. Open the "web.config" file by double-clicking it in the project manager, and then add the following XML directly beneath the opening <configuration> node. It is recommended that you copy/paste this text to avoid mistakes, all node names are case sensitive.

```
<system.runtime.remoting>
    <application>
        <service>
            <wellknown
                mode="Singleton"
                objectUri="RemotePersistenceDemo.rem"
                type="RemotePersistenceServerWeb.EcoPersistenceMapperProvider,
RemotePersistenceServerWeb"/>
         </service>
        <channels>
            <channel ref="http">
                <serverProviders>
                    <formatter ref="soap" typeFilterLevel="Full" />
                </serverProviders>
                <clientProviders>
                    <formatter ref="soap" />
                </clientProviders>
            </channel>
        </channels>
    </application>
</system.runtime.remoting>
```

This XML informs IIS that we have a remoting service with the following characteristics.

- Access is gained via the "objectUri" specified "RemotePersistenceDemo.rem" giving the full url http://localhost/VirtualFolderName/RemotePersistenceDemo.rem
- The service type is a Singleton, only one instance of the EcoPersistenceMapperProvider will be created.
- The class type to create is a RemotePersistenceServerWeb.EcoPersistenceMapperProvider, which is held in the file RemotePersistenceServerWeb.dll

The format of this XML is as follows

```
<wellknown
    mode="Singleton"
    objectUri="{Last part of URL}"
    type="{Name space}.{Class name}, {File name with DLL extension}"/>
```

13. Add a PersistenceMapperBdp to the EcoPersistenceMapperProvider, then right-click it and select InterBase.
14. Add a BdpConnection to the EcoPersistenceMapperProvider and set its connection string.
15. Ensure that the PersistenceMapperBdp.Connection property is set correctly, and that the EcoPersistenceMapperProvider.PersistenceMapper property is also set correctly. These should be set automatically by ECO when the relevant components are added to the design surface.
16. Generate the database if this has not already been done, i.e. if you are not using the same database created in the Console Server example.
17. Compile the application.

The server is now ready to serve requests from clients. Next the client should be updated so that it connects to the IIS hosted server rather than the console hosted server. This is as simple as changing the ClientPersistenceMapper.Url to:

```
http://localhost:80/RemotePersistenceServerWeb/RemotePersistenceDemo.rem
```

In the client demo application this change was implemented in the constructor of the client's EcoSpace using an IFDEF compiler directive. Therefore, if you are modifying the demo project, you will need to define either UseSockets or UseWebServer in the project's compiler options. Setting the URL at design time will have no affect as the value will be overwritten when the application runs.

[Delphi]

```
{$IFDEF UseSockets}
            persistenceMapperClient1.Url := 'tcp://localhost:1234/RemotePersistenceDemo';
{$ELSE}
            persistenceMapperClient1.Url :=
'http://localhost:80/RemotePersistenceServerWeb/RemotePersistenceDemo.rem';
{$ENDIF}
```

[C#]

```
#if UseSockets
            persistenceMapperClient1.Url = "tcp://localhost:1234/RemotePersistenceDemo";
#endif
#if UseWebServer
            persistenceMapperClient1.Url =
"http://localhost:80/RemotePersistenceServerWeb/RemotePersistenceDemo.rem";
#endif
```

4.3 A Windows service based server

This next example will demonstrate the steps involved in creating a Windows service manually and hosting a remote persistence server within it.

1. Create a new console application.
2. Add the following assemblies to the project's references list:
   - The business classes package DLL.
   - System.Configuration.Install
   - System.Runtime.Remoting.
   - System.ServiceProcess.
3. Rename the class "Class" to "Program".
4. Open the menu item File->New->Other, select the correct language (C# or Delphi) and then select "New ECO Files".
5. Add an ECO Space to the project.
6. Compile the project so that the ECO designers may read model information from the binaries produced.
7. Open the EcoSpace and click the "Select Packages" icon at the bottom.

7. Ensure that the business classes package is included in the the "Selected" list on the right side of the form.
8. Open the menu item File->New->Other, select the correct language (C# or Delphi) and then select "New ECO Files".
9. Add an ECO Persistence Mapper Provider to the project.
10. Set its "EcoSpaceType" property to reference the EcoSpace created earlier.
11. Add a PersistenceMapperBdp to its design surface.
12. Right-click persistenceMapperBdp1 and select "InterBase" from the context menu.
13. Add a BdpConnection to the persistence mapper's design surface, and set its "ConnectionString" property.

The basic persistence steps are now complete, next the console application will be converted into a Windows service. Windows services can be stopped and restarted, to achieve this the service will register the EcoPersistenceMapperProvider class when started, and then unregister the remoting channel when stopped.

14. Add a new class to the project named "PersistenceService".
15. Add the following items to its using/uses clause.

[Delphi]

```
using
  System.Collections,
  System.Runtime.Remoting,
  System.Runtime.Remoting.Channels,
  System.Runtime.Remoting.Channels.Tcp,
  System.Runtime.Serialization.Formatters,
  System.ServiceProcess;
```

[C#]

```
using System;
using System.Collections;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Serialization.Formatters;
using System.ServiceProcess;
```

16. Specify "ServiceBase" as the base class for the new PersistenceService class.
17. Declare a private member of the class named "RemotingChannel" of the type "TcpChannel".
18. Add a constructor, within it set the ServiceName member to "RemotePersistenceServerService".
19. Override the OnStart method and implement it as follows:

[Delphi]

```
procedure PersistenceService.OnStart(args: array of string);
var
  Provider: BinaryServerFormatterSinkProvider;
  ChannelProperties: IDictionary;
begin
  Provider := BinaryServerFormatterSinkProvider.Create;
  Provider.TypeFilterLevel := TypeFilterLevel.Full;

  ChannelProperties := Hashtable.Create() as IDictionary;
  ChannelProperties['port'] := 1234;

  //Remoting channel is a private member of the class
  RemotingChannel := TcpChannel.Create(ChannelProperties, nil, Provider);
  ChannelServices.RegisterChannel(RemotingChannel);

  //Now register the remote persistence mapper
  RemotingConfiguration.RegisterWellKnownServiceType(
    EcoPersistenceMapperProvider, "RemotePersistenceDemo",
    WellKnownObjectModel.Singleton);

  inherited OnStart(args);
end;
```

[C#]

```
protected override void OnStart(string[] args)
{
    BinaryServerFormatterSinkProvider provider = new BinaryServerFormatterSinkProvider();
    provider.TypeFilterLevel = TypeFilterLevel.Full; // Needed for serializations

    IDictionary channelProperties = new Hashtable();
    channelProperties["port"] = 1234;

    //Remoting channel is a private member of the class
    RemotingChannel = new TcpChannel(channelProperties, null, provider);
    ChannelServices.RegisterChannel(RemotingChannel);

    RemotingConfiguration.RegisterWellKnownServiceType(
        typeof(EcoPersistenceMapperProvider),
        "RemotePersistenceDemo",
        WellKnownObjectMode.Singleton);
    base.OnStart(args);
}
```

20. Override OnStop and implement it as follows:

[Delphi]

```
procedure PersistenceService.OnStop;
begin
  //Unregister the channel created in OnStart
  ChannelServices.UnregisterChannel(RemotingChannel);
end;
```

[C#]

```
protected override void OnStop()
{
    //Unregister the channel created in OnStart
    ChannelServices.UnregisterChannel(RemotingChannel);
    base.OnStop();
}
```

Now that the service is written an installer class is required by Windows so that the service may be installed and uninstalled.

21. Add a new class to the project named "PersistenceServiceInstaller".

22. Add the following items to the using/uses clause.

[Delphi]

```
uses
  System.ComponentModel,
  System.Configuration.Install,
  System.ServiceProcess;
```

[C#]

```
using System.ComponentModel;
using System.Configuration.Install;
using System.ServiceProcess;
```

23. Decorate the class with the following .NET attribute:

[Delphi]

```
type
  [RunInstaller(true)]
  PersistenceServiceInstaller = class(Installer)
```

[C#]

```
[RunInstaller(true)]
public class PersistenceServiceInstaller : Installer
```

The .NET service installer will look for this attribute on all classes that descend from the Installer class, if it is found and the

value is "True" then it will create an instance of the class. It is in the constructor of this class that the developer is expected to perform service registration.

24. Implement the constructor as follows:

[Delphi]

```
constructor PersistenceServiceInstaller.Create;
var
  MyServiceProcessInstaller: ServiceProcessInstaller;
  MyServiceInstaller: ServiceInstaller;
begin
  MyServiceProcessInstaller := ServiceProcessInstaller.Create;
  MyServiceProcessInstaller.Account := ServiceAccount.LocalService;

  MyServiceInstaller := ServiceInstaller.Create;
  MyServiceInstaller.StartType := ServiceStartMode.Automatic;
  MyServiceInstaller.ServiceName := 'RemotePersistenceService';
  MyServiceInstaller.DisplayName := 'Remote persistence server demo service';

  Installers.Add(MyServiceProcessInstaller);
  Installers.Add(MyServiceInstaller);
end;
```

[C#]

```
public PersistenceServiceInstaller()
{
    ServiceProcessInstaller serviceProcessInstaller;
    serviceProcessInstaller = new ServiceProcessInstaller();
    serviceProcessInstaller.Account = ServiceAccount.LocalService;

    ServiceInstaller serviceInstaller;
    serviceInstaller = new ServiceInstaller();
    serviceInstaller.StartType = ServiceStartMode.Automatic;
    serviceInstaller.ServiceName = "RemotePersistenceServerService";
    serviceInstaller.DisplayName = "Remote persistence server demo service";

    Installers.Add(serviceProcessInstaller);
    Installers.Add(serviceInstaller);
}
```

25. Finally change the "Program" class so that is creates an instance of the service when the application starts.

[Delphi]

```
ServiceBase.Run(PersistenceService.Create);
```

[C#]

```
public static void Main()
{
    ServiceBase.Run(new PersistenceService());
}
```

The service application is now ready to be used. To register the service use the "InstallUtil.exe" command line utility that ships with the .NET framework.

```
C:> path = C:\{Windows install folder}\Microsoft.NET\Framework\{Framework version number}
C:> InstallUtil {Project path}\Bin\RemotePersistenceServerService.exe
```

5 Further reading

This document has introduced you to the concept of remote persistence in ECO. Considering what has been achieved there has again been a very little amount of programming involved.

Index

A

A console based server 6

A Windows service based server 11

An IIS hosted server 8

F

Further reading 16

I

Introduction 1

P

Prerequisites and goals 2

W

Writing the client application 3

Writing the persistence server 6