A
Active Directory
ASP.NET
Assemblies

B
Base Data Types

C
Collections
Component Model
Configuration
Cryptography

D
Data and ADO.NET
Debugging
Diagnostics

E
Exceptions

F
G
Globalization and NLS

H
I
Input and Output
Interoperability

J
K
L
Languages

M
Messaging

N
Networking

O
P
Q
R
Reflection
Regular Expressions
Remoting
Resources

S
Security
Serialization
Services
Setup

T

Welcome to the Common Tasks QuickStart

The Common Task QuickStart is designed to help you quickly find the answers to your most frequent programming questions. Unlike other QuickStarts that are meant to be read from start to end, this section may be used as a topical reference. If there are topics not addressed here that you'd like to see in future releases, please drop us an email at qswish@microsoft.com.

How Do I...

Active Directory:

- Access Active Directory?
- Modify Active Directory?

ASP.NET:

- Get Started with ASP.NET?
- Work with server controls?
- Find available server controls?
- Write an ASP.NET page?
- Apply styles to controls?
- Perform form validation?
- Write a user control?
- Work with databinding?
- Work with the DataGrid control?
- Work with the DataList control?
- Write and use a business object?
- Write my own server control?
- Cache my pages or data?
- Configure an ASP.NET application?
- Deploy an ASP.NET application?
- Secure an ASP.NET application?
- Check Windows Identity?
- Apply forms-based authentication?
- Grant or deny permission to a user or role?
- Localize an ASP.NET application?
- Instrument an ASP.NET Framework application (Tracing)?
- Debug an ASP.NET application?
- Write optimized ASP.NET applications?
- Migrate my ASP applications to ASP.NET?
- Interoperate with a COM component?
- Use MTS Transactions?
- Include a rotating advertisement on a page?
- Include a calendar on a page?
- Upload a File to an ASP.NET application?
- Handle errors from an ASP.NET application?

Assemblies:

- Create an assembly with a strong name?
- Create a delay signed shared assembly?
- Work with the global assembly cache?
- Change the default search path?
- Use version policy?
- Create a publisher policy assembly?

Messaging:

- Send a message to a message queue?
- Receive a message from a message queue?
- Pass a complex type through a message queue?
- Receive asynchronously?
- Use message formatters?
- Control a message queue?

Networking:

- Make a GET request?
- Make a GET request with SSL?
- Make a GET request with NTLM?
- Make a GET request through a proxy?
- Make a GET request asynchronously?
- Make a GET request from ASP.NET?
- Make an async GET request from ASP.NET?
- Handle WebExceptions?
- Make a POST request?
- Create a DateTime client with TCPClient?
- Create a DateTime server with TCPListener?
- Create a Chat client program with UDPClient?

Reflection:

- Get the types in an assembly?
- List all the members of a type?
- Invoke methods?

Regular Expressions:

- Use Regular Expressions to match a pattern?
- Get all matches for a pattern?
- Use Regular Expressions to make replacements?
- Find common Regular Expression patterns?

Remoting:

- Get an overview of remoting?
- How remoting works?
- Use the remoting samples?
- Create a remote server?
- Create a client of a remote server?
- Create a remote object as a singleton?
- Pass an object to a server by reference?

Timers

U
V
W
Web Services
Windows Forms

X
XML Data
XML Serialization

Y
Z

Get URL for this page

Base Data Types:

- Sort an array?
- Format basic types?
- Parse datatypes?
- Compare references?
- Compare object values?
- Use mathematical functions?

Collections:

- Iterate over a collection?
- Use a Hashtable collection?
- Choose a collection to use?
- Implement a collection?
- Clone a collection?

Component Model:

- Use the CodeDOM to Emit Source Code?

Configuration:

- Write Client Configuration Files?
- Retrieve Configuration Settings?
- Write a Custom Config Handler?

Cryptography:

- Encrypt and decrypt a file?
- Generate and compare a hash value?

Data and ADO.NET:

- Get an overview of ADO.NET?
- Execute a command?
- Get out parameters from a stored procedure?
- Populate a DataSet from a Database?
- Update a database from a DataSet?
- Read and write XML?
- Use database transactions?
- Handle errors?
- Work with hierarchical or relational data?
- Work with typed data?
- Filter data?
- Use connection pooling?
- Generate XML with data from SQL Server?
- Load a native ADO recordset into a DataSet?
- Retrieve data from a database Using OLE DB?
- Retrieve data from a SQLServer database?
- Migrate from ADO to ADO.NET?

Debugging:

- Debug an ASP.NET application?

Diagnostics:

- Write to an event log?
- Read from an event log?
- Monitor an event log?
- Write to a performance counter?
- Read from a performance counter?
- Count rate of change?
- Work with tracing?

- Pass an object to a server by value?
- Compile a client against an interface?
- Make an asynchronous call to a remote object?

Resources:

- Create resources?
- Use resources?
- Read and write resources?
- Make your own ResourceReader?

Security:

- Control access to my shared component?
- Check Windows Identity in a client application?
- Request the permissions my code needs?
- Script Security Policy Changes?
- Secure an ASP.NET application?
- Check Windows Identity in a server application?
- Apply forms-based authentication?
- Authorize users or roles in ASP.NET?

Serialization:

- Serialize an object graph?
- Use ISerializable?

Services:

- Write a simple Service application?

Setup:

- Create system resources during installation?
- Execute custom code during installation?

Timers:

- Get timer events without message pump?

XML Web services:

- Work with a basic Web Service?
- Use various datatypes with XML Web services?
- Work with XML Web services and DataSets?
- Match text patterns as a Web Service?

Windows Forms:

- Create a default error handler?
- Create an Owner-drawn ListBox?
- Manipulate a control from a background thread?

XML Data:

- Work with XML data and the XML classes?
- Read XML from a file?
- Read XML from a stream?
- Read XML from a URL?
- Write XML to a file?
- Apply validation when reading XML?
- Use multiple XmlReaders?
- Load and use the XmlDocument (W3C

- Use trace switches and listeners?
- Instrument a small application with tracing?
- Start and stop a process?
- Get information about processes?

Exceptions:

- Catch an exception?
- Throw an exception?

Globalization and NLS:

- Use encodings?
- Set your current culture and region?
- Handle multiple encodings?
- Display multiple currencies?

Input and Output:

- Read a text file?
- List a directory?
- Create a log file?
- Read and write large files?
- Read and write binary data?
- Watch file system changes?

Interoperability:

- Get started using Interoperability?
- Build a .NET Server Callable from COM clients?
- Pass simple data from .NET code to VB6 code?
- Pass an array from .NET code to VB6 code?
- Build a .NET client that uses a COM server?
- Call COM methods from .NET?
- Sink unmanaged events from .NET?
- Marshal strings between managed/unmanaged code?
- Call a function exported from an unmanaged DLL?

Languages:

- Declare a variable?
- Issue a statement?
- Comment my code?
- Declare and use simple property?
- Use an indexed property?
- Declare an indexed property?
- Declare and use an enumeration?
- Enumerate a Collection/Array?
- Declare and use a method?
- Define custom attributes?
- Declare and use an array?
- Statically initialize a variable?
- Write an if-statement?
- Write a case-statement?
- Write a for-loop?
- Write a while-loop?
- Handle Exceptions?
- Concatenate a string?
- Declare an Event?
- Declare an Event Handler delegate?
- Add an Event Handler delegate to an Event?
- Cast a variable to a type?
- Convert a variable to type?
- Declare a class or interface?
- Inherit from a base class?
- Implement an interface?

DOM)?
- Save XML with the XmlDocument (W3C DOM)?
- Handle Events from XmlDocument?
- Use an XmlNodeReader?
- Load and use the XmlDataDocument?
- Read XML from a XmlDataDocument?
- Navigate XML with the XmlNavigator?
- Query XML with an XPath expression?
- Apply an XSL transformation to XML?
- Use an XPathExpression class?
- Use XML and the DataSet class?
- Create DataSet mappings from an XSD schema?
- Load a DataSet with XML?
- Infer DataSet mappings from XML?
- Save DataSet mappings to an XSD schema file?
- Save a DataSet as XML?
- Read and write schema?
- Use the Xml Schema Object Model?
- Use XML Schema Collection?
- XmlNamespaceManager and NameTable?

XML Serialization:

- Create an XSD schema from a class?
- Create a class from an XSD schema?
- Read and Write objects into XML?

- Write a class with a Main method?
- Write a standard module?

Copyright 2001 Microsoft Corporation. All rights reserved.

Active Directory
Access Active Directory
Modify Active Directory

Back to Index

Get URL for this page

How Do I...Access Active Directory?

The Active Directory Service Interface (ADSI) accesses the capabilities of directory services from different network providers in a distributed computing environment to present a single set of directory service interfaces for managing network resources. Administrators and developers can use ADSI services to enumerate and manage the resources in a directory service, regardless of which network environment contains the resource. Use Active Directory to perform common administrative tasks, such as locating resources throughout the distributed computing environment.

This sample illustrates how to locate resources in the Active Directory. It is a small console application that can be run from a command prompt. The application takes one command line argument. The first argument has to be a valid path to an Active Directory entry.

Try running the following command, substituting the path for a valid Active Directory path for your particular network:

```
> ADRead.exe "LDAP://DC=Microsoft,DC=COM"
```

In its simplest form, reading from the Active Directory involves:

1. Creating a new **DirectoryEntry**:

```
Dim objDirEnt As DirectoryEntry = New DirectoryEntry(path)
```

VB

2. Reading the information from the Active Directory and printing it to the screen:

```
Dim tab As String = "    "
Dim key As String
Dim objCollection As Object

For Each Key In objDirEnt.Properties.PropertyNames
    Console.WriteLine(tab & key & " = ")
    For Each objCollection In objDirEnt.Properties(key)
            Console.WriteLine(tab & tab & CStr(objCollection))
    Next
Next
```

VB

Example

VB ADRead.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Active Directory
Access Active Directory
Modify Active Directory

Back to Index

Get URL for this page

How Do I...Modify Active Directory?

The Active Directory Service Interface (ADSI) accesses the capabilities of directory services from different network providers in a distributed computing environment to present a single set of directory service interfaces for managing network resources. Administrators and developers can use ADSI services to enumerate and manage the resources in a directory service, regardless of which network environment contains the resource. Use Active Directory to perform common administrative tasks, such as adding new users and managing printers throughout the distributed computing environment.

This sample illustrates how to change a value of a property of an Active Directory entry. It is a small console application that can be run from a command prompt. The application takes three command line arguments. The first argument must be a valid path to an active directory entry. The second argument is a property name of the entry. The third argument is the new value of the entry.

Try running the following command substituting the path and property name for a valid Active Directory path for your particular network:

```
> ADWrite.exe "LDAP://DC=Microsoft,DC=COM" "name" "Microsoft"
```

In its simplest form, writing to the Active Directory involves:

1. Creating a new **DirectoryEntry**:

```
Dim adPath As String = ...
Dim objDirEnt As DirectoryEntry = New DirectoryEntry(adPath)
```

VB

2. Setting the **DirectoryEntry** object's properties:

```
Dim propertyName As String = ...
Dim newValue As String = ...

objDirEnt.Properties(propertyName)(0) = newValue
```

VB

3. Committing the changes to the Active Directory:

```
objDirEnt.CommitChanges()
```

VB

Example

VB ADWrite.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Create an assembly with a strong name?

Assemblies can be assigned a cryptographic signature, called a strong name, which provides name uniqueness for the assembly and prevents someone from taking over the name of your assembly (name spoofing). If you are deploying an assembly that will be shared among many applications on the same machine, it must have a strong name. Even if you only use the assembly within your application, using a strong name ensures that the correct version of the assembly gets loaded

The first step in building an assembly with a strong name is to obtain a cryptographic key pair. The .NET Framework SDK includes a Strong Name tool (Sn.exe) that can be used to generate a key pair. The key pair that is generated by the Strong Name tool can be kept in a file or you can store it in your local machine's Crytographic Service Provider (CSP). The following command uses the Strong Name tool to generate a new key pair and store it in a file called TestKey.snk:

```
sn -k Testkey.snk
```

Once you have obtained the key pair, you need to add the proper custom attribute to your source in order for the compiler to emit the assembly with a strong name. Choosing the correct attribute depends on whether the key pair used for the signing is contained in a file or in a key container within the CSP. For keys stored in a file, use System.Reflection.AssemblyKeyFileAttribute. For keys stored in the CSP use System.Reflection.AssemblyKeyNameAttribute.

The following example uses **AssemblyKeyFileAttribute** to specify the name of the file containing the key pair. In Visual Basic, the assembly level attributes must be the first statements in the file.

```
Imports System
Imports System.Reflection

<assembly:AssemblyKeyFileAttribute("TestKey.snk")>
```

VB

You can also choose to delay sign your assembly. This means that the space for the signature is reserved in the PE file, but the signing itself is not done until later. This approach is typically used if you do not have access to the private key you need to generate the strong name. For more details, see How do I... Use Delay Signing.

The makefile included with this sample uses the Strong Name Tool to generate a key pair, then builds a strong-named assembly with C# or Visual Basic. If you look at the assembly with MSIL Disassembler (Ildasm.exe) after the build is complete, you can see the public key as part of the assembly's identity in the manifest. The signature itself is stored elsewhere in the PE file. You do not need to use the Strong Name tools to generate a new key-pair for each build. Since the public key is part of the assembly's name, you can generate one key-pair and never change it for the lifetime of the assembly.

Example

VB Strong Names

[Run Sample] | [View Source]

For more information regarding strong names, see the topic "Working with Strongly-Named Assemblies" in the core SDK documentation.

LateBreaking Samples:

- C++

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Create a delay signed shared assembly?

Giving an assembly a strong name requires two cryptographic keys: a public key and a private key. This key pair is passed to the compiler at build time, as described in How Do I... Create an assembly with a Strong Name?, to create the strong name.

However, the person building the assembly does not always have access to the private key required for strong naming. This is most common in corporations that have a central signing entity closely guards private keys. Only a few select people have access to these keys. Also, the process of assigning a strong name cannot be done after building because the public key is part of the assembly's identity and must be supplied at build time so that clients of the assembly can compile against the full assembly identity.

The .NET Framework offers delay signing, which effectively splits the process of assigning the strong name into two steps:

- 1. At build time, the public key is given to the compiler so it can be recorded in the **PublicKey** field in the assembly manifest. Also, space is reserved in the file for the signature, although the actual signature is not generated at this time.
- 2. At a later time, the the actual signature is generated and stored in the file. Signature generation is done with the -R switch to the Strong Named tool (Sn.exe).

When you include the System.Reflection.AssemblyDelaySignAttribute in your source code, it indicates to the compiler that the assembly needs to be created with delay signing. You also need to include the public key, using **AssemblyKeyFileAttribute**. Typically, the signing entity will use the SN -k to generate a key pair and store it in a file. Next, it pulls the public key out of the file using SN -p. The public key can then be given out, with the private key still secret.

```
sn -k Testkey.snk
sn -p Testkey.snk TestPublicKey.snk
```

The following example uses **AssemblyKeyFileAttribute** and **AssemblyDelaySignAttribute** to create a delay signed assembly. In Visual Basic, the assembly level attributes must be the first statements in the file.

```
Imports System
Imports System.Reflection

<assembly:AssemblyKeyFileAttribute("TestPublicKey.snk")>
<assembly:AssemblyDelaySignAttribute(true)>
```

VB

Since the assembly in the example does not have a valid signature, the signature validation performed by the common language runtime will fail when you try to install the assembly into the global assembly cache or load it from an application directory. However, the Strong Name tool can be used to disable signature verification of a particular assembly by using the -Vr option:

```
sn -Vr DelaySign.dll
```

A valid signature must be generated before the assembly is shipped to customers using **sn -R**. This is typically done by the company signing entity. You must supply the full key pair to create a valid signature.

```
sn -R DelaySign.dll Testkey.snk
```

The makefile included with this sample performs the following steps:

- 1. Creates a key-pair using **sn -K**.
- 2. Separates the public key from the private key and stores the public key in its own file.
- 3. Creates a delay signed assembly in either Visual Basic or C#.
- 4. Uses the Strong Name tool to request that signature verification be skipped for the assembly just generated.
- 5. Generates a valid signature using the Strong Name tool. This typically happens just before you ship the assembly.

Example

VB Delay Signing

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

For more information regarding delay signing, see the topic "Delayed Signing an Assembly" in the core SDK documentation.

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Work with the global assembly cache?

The global assembly cache is a machine-wide store used to hold assemblies that are intended to be shared by several applications on the machine. The .NET Framework provides two tools for working with the cache. One is a Windows shell extension that allows you to work with the cache using a Graphical User Interface (GUI). The other is a command line tool, called the Global Assembly tool (Gacutil.exe), that is typically used in build and test scripts. The command line tool is called the Global Assembly Cache tool (Gacutil.exe).

All assemblies in the global cache must have strong names.

Viewing the Contents of the Assembly Cache

Navigate to %winnt%\assembly using the Windows Explorer to activate the shell extension. The default view shows the contents of the assembly cache. You can also view the contents of the cache with the /l option of the Global Assembly Cache tool:

```
gacutil /l
```

Installing Assemblies

To install an assembly using the shell extension simply drag and drop the file containing the assembly's manifest into the global assembly cache directory. To install an assembly using the Global Assembly Cache tool, use the /i option:

```
gacutil /i math.dll
```

Uninstalling Assemblies

To delete an assembly with the shell extension, right click on it and select Delete. The /u option of the Global Assembly Cache tool can also be used:

```
gacutil /u math,ver=1.0.0.0
```

The version of Windows Installer (1.5) that will ship in the Visual Studio.NET timeframe has native support for the assembly cache. When creating an Installer package with the Visual Studio Deployment tool or another setup tool, you can specify which assemblies you'd like installed in the cache. Using the Windows Installer to work with the assembly cache gives you the standard Installer benefits like install-on-demand, advertisement, publishing, and so on.

Assemblies
- Use strong names
- Use delay signing
- Work with the global cache
- Change the search path
- Use assembly version policy
- Create publisher policy

Back to Index

Get URL for this page

How Do I...Change the search path for private assemblies?

Private assemblies are assemblies that are visible to only one application. The .NET Framework enables developers to build applications that are isolated from changes made to the system by other applications. Private assemblies must be deployed in the directory structure of the containing application and are found during runtime through a process called probing, which is simply a mapping from an assembly's identity to a file on disk that contains the manifest.

By default, probing for private assemblies is done in the application base (the root directory of the application) and the subdirectories that follow naming conventions based on assembly name and culture. You can customize this behavior by specifying a privatePath in your application's configuration file. The privatePath is a semi-colon delimited list of directories in which the common language runtime will search for private assemblies. These directory names are relative to the application base - they cannot point outside the application base. The directories on privatePath will be searched after the application base itself. The following configuration file adds a bin directory to the assembly search path:

```
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

            <probing privatePath="bin"/>

        </assemblyBinding>
    </runtime>
</configuration>
```

Example

C# Config Example

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Use Assembly Version Policy?

A primary goal of the deployment system in the .NET Framework is to eliminate conflicts between applications caused by shared components and shared states (or DLL conflicts). A key solution to this problem is a robust versioning system. The .NET Framework records information about an application's dependencies in the assembly manifest. This dependency information includes a version number that is used at runtime to load the proper version of a dependency.

By default, the common language runtime will load the version of a dependency that is specified in the manifest. This is preferred in the majority of scenarios. However, there are cases where running an application with a different version of a dependency can be useful. In order to accomplish this, version policies can be included in an application's configuration file. For example, the following XML code fragment redirects references to version 5.0.0.0 of a shared assembly called "caclR" up to version 6.0.0.0 of that assembly:

```
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v

            <dependentAssembly>
                <assemblyIdentity name="calcR"
                                  publicKeyToken="a1690a5ea44ba
                                  culture=""/>

                <bindingRedirect oldVersion="5.0.0.0"
                                 newVersion="6.0.0.0"/>

            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

In addition to policies specified at the application level, the .NET Framework also provides two other policy levels: publisher and administrator. A Publisher policy statement describes the compatibility of an assembly issued by the publisher of that shared assembly. See How Do I ... Create Publisher Policy for details. Administrator level policy is a machine version policy that affects all applications. Administrator policy is created using the same XML syntax as application level policy. The administrator policy file is called machine.config, and resides in the common language runtime install directory. The three policy levels are evaluated in the following order:

- Application Policy
- Publisher Policy
- Administrator Policy

Example

C# Math.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Create a Publisher Policy Assembly?

A Publisher policy statement describes the compatibility of an assembly issued by the publisher of that shared assembly. See How Do I...Use Version Policy" to see how a publisher's policy statement fits into the overall .NET Framework version policy scheme. Publisher policy is commonly used in service pack scenarios. For example, a publisher may produce a number of small releases that enhances certain features for a particular customer. For maintenance reasons, the publisher may wish to collect all of these fixes into a single service pack release and have all exsiting customers upgrade to the new service pack.

A publisher policy statement is an XML configuration file wrapped as a separate assembly. There are three reasons that publisher policies are shipped as assemblies. The first is to ensure that the policy statement comes from the author of the assembly that the policy is making the compatibility statement about. This is accomplished by requiring that the policy assembly has a strong name generated with the same key-pair as the original assembly. The second reason is ease of deployment. Publishers or administrators can ship policy statements using the assembly deployment mechansims provided by the .NET Framework, including the Windows Installer and code download. Using the Windows Installer is particularly convenient because all policy assemblies must be installed in the global assembly cache. Finally, assemblies ship policy statements so that they can be versioned. This allows a publisher to ship a subsequent statement if a previous policy is found not to work in some scenarios. In effect, this allows a publisher to change his mind about the compatibility of his assembly independent of when it was shipped. The flexibility enabled by decoupling the compatibility statements from the code makes it much easier to fix broken applications in the .NET Framework. If multiple versions of a given policy assembly are found in the assembly cache, the .NET Framework will use the policy assembly with the highest version number.

In general, there are two steps required to create a publisher policy assembly:

- 1. Create the XML file containing the compatibility statement. You will have to use an XML editor to create this file.
- 2. Use the Assembly Generation tool (Al.exe) to create an assembly containing the XML file.

The format of the xml file, along with rules about how the elements relate, is described in detail in the .NET Framework SDK Guide. Here is an example file:

```
<configuration>
    <runtime>
        <assemblyBinding>

            <dependentAssembly>
                <assemblyIdentity name="myasm"
                                  publicKeyToken="e9b4c4996039ede8"
                                  culture="en-us"/>

                <bindingRedirect oldVersion="1.0.0.0-1.0.9.9"
                                 newVersion="2.0.0.0"/>

                <codeBase version="2.0.0.0"
                          href="http://www.foo.com/bar.dll"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

After the xml file is created, use the Assembly Generation tool to create a policy assembly. The following switches to the Assembly Generation tool are required:

1. **/link:** links the xml file into the assembly.
2. **/out:** gives the resulting policy assembly a name. Policy assemblies are found in the global assembly cache by naming convention. Therefore, their names must be:

   policy.<major number>.<minor number>.<main assembly name>

   For example, **policy.2.0.myasm**

3. **/keyfile:** The key pair used to give the assembly a strong name (or at least the public key if delay signing is used). As described earlier, this key pair must be the same key pair used to sign the assembly to which this policy statement applies.
4. **/version:** The version number of the policy assembly.

The following example shows a command line that uses the Assembly Generation tool:

```
Al /link:publisher.cfg /out:policy.2.0.myasm /keyfile:myasm.snk /version:2.0.0.0
```

Example

Publisher.cfg

[Run Sample] | [View Source]

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Create a delay signed shared assembly?

Giving an assembly a strong name requires two cryptographic keys: a public key and a private key. This key pair is passed to the compiler at build time, as described in How Do I... Create an assembly with a Strong Name?, to create the strong name.

However, the person building the assembly does not always have access to the private key required for strong naming. This is most common in corporations that have a central signing entity closely guards private keys. Only a few select people have access to these keys. Also, the process of assigning a strong name cannot be done after building because the public key is part of the assembly's identity and must be supplied at build time so that clients of the assembly can compile against the full assembly identity.

The .NET Framework offers delay signing, which effectively splits the process of assigning the strong name into two steps:

- 1. At build time, the public key is given to the compiler so it can be recorded in the **PublicKey** field in the assembly manifest. Also, space is reserved in the file for the signature, although the actual signature is not generated at this time.
- 2. At a later time, the the actual signature is generated and stored in the file. Signature generation is done with the -R switch to the Strong Named tool (Sn.exe).

When you include the System.Reflection.AssemblyDelaySignAttribute in your source code, it indicates to the compiler that the assembly needs to be created with delay signing. You also need to include the public key, using **AssemblyKeyFileAttribute**. Typically, the signing entity will use the SN -k to generate a key pair and store it in a file. Next, it pulls the public key out of the file using SN -p. The public key can then be given out, with the private key still secret.

```
sn -k Testkey.snk
sn -p Testkey.snk TestPublicKey.snk
```

The following example uses **AssemblyKeyFileAttribute** and **AssemblyDelaySignAttribute** to create a delay signed assembly. In Visual Basic, the assembly level attributes must be the first statements in the file.

```
Imports System
Imports System.Reflection

<assembly:AssemblyKeyFileAttribute("TestPublicKey.snk")>
<assembly:AssemblyDelaySignAttribute(true)>
```

VB

Since the assembly in the example does not have a valid signature, the signature validation performed by the common language runtime will fail when you try to install the assembly into the global assembly cache or load it from an application directory. However, the Strong Name tool can be used to disable signature verification of a particular assembly by using the -Vr option:

```
sn -Vr DelaySign.dll
```

A valid signature must be generated before the assembly is shipped to customers using **sn -R**. This is typically done by the company signing entity. You must supply the full key pair to create a valid signature.

```
sn -R DelaySign.dll Testkey.snk
```

The makefile included with this sample performs the following steps:

- 1. Creates a key-pair using **sn -K**.
- 2. Separates the public key from the private key and stores the public key in its own file.
- 3. Creates a delay signed assembly in either Visual Basic or C#.
- 4. Uses the Strong Name tool to request that signature verification be skipped for the assembly just generated.
- 5. Generates a valid signature using the Strong Name tool. This typically happens just before you ship the assembly.

Example

VB Delay Signing

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

For more information regarding delay signing, see the topic "Delayed Signing an Assembly" in the core SDK documentation.

Base Data Types

Sort an array
Format a string
Parse datatypes
Compare references
Compare object values
Use mathematical functions

Back to Index

Get URL for this page

How Do I...Sort an array?

This sample illustrates sorting the elements in an array. Using the **Array.Sort** method, you can sort any array whose elements implement the **IComparable** interface. You can also sort arrays of your own types as long as they implement **IComparable**. If a type does not implement **IComparable** (or if you want to customize the sort order) you can still use **Array.Sort** if you provide a Comparer, such as the **MyTypeNameComparer** shown in this sample.

VB Sort.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Sorting is a standard basic function of any group of objects or data. In the following examples, sorting is performed specifically on an array. Most data types can be sorted when they are stored in an array, as long as they support the **IComparable** interface. The easiest way to sort an array is to call the **Sort** method, passing in the array you want to sort. The following code example illustrates populating an array with names, and then calling the **Sort** method to organize those names in alphabetical, ascending order.

```
Dim stringArray() As String ={"Michael", "mary", "Robert", _
                "Abigail", "lorne", "Therese", "terrence", "Samantha"}

Array.Sort(stringArray)
```

VB

If you want to sort your array in descending order, call the **Sort** method, and then call the **Reverse** method, passing in the array you want to sort in each case. Note that calling **Reverse** reverses the existing entries in the array. To get a descending array, you need to sort the array before reversing it.

```
Dim intArray() As Integer = {43, 12, 1, 17, 23, 36, 11}

Array.Sort(intArray)
Array.Reverse(intArray)
```

VB

You might want to sort more than predefined data types. What if you have defined your own class and want to sort an array of members of that class? To do this, implement the **IComparable** interface on that class, and then define the rule for comparing an instance of that class to other objects. Generally, you can do this by selecting a variable inside that class that can be used to determine the way your class should be sorted (for example, you might want to sort by 'LastName' for the class 'Employee').

The following code example defines **MyType**, which has two pieces of information: name and age. In this case, the sort mechanism is based on age. You need to include a **CompareTo** method, and define the rules for comparison in that method. When one object is compared to another, the return value from this method is used to determine whether the object being compared is greater than or less than the current object (a negative value indicates the comparing object is less than the current object).

```
'  imports statements...
Imports System
Imports System.Collections

Class MyType: Implements IComparable

        '  Property, method, and field declarations...
        '  included are an Age, and a Name property

        Public Function CompareTo (ByVal o as Object) As Integer _
                                Implements IComparable.CompareTo
                If Not (TypeOf o Is MyType) Then
                        throw new ArgumentException ("o must be of type 'MyType'")
                End If

                Dim v As MyType = CType(o, MyType)
                CompareTo = _age - v._age
```

```
        End Function
End Class
```

VB

To sort members of this class, populate an array with entries and call the **Array.Sort** method. Because you implemented the **IComparable** interface, **Sort** knows to use the **CompareTo** method of the class to determine how to sort the array.

```
Dim myTypeArray() As MyType = New MyType() { New MyType("Max",23), _
                New MyType("Lisa",47), New MyType("Andrea",35), _
                New MyType("Mel",29), New MyType("Andrew",32), New MyType("Ahmed",35) }

Array.Sort(myTypeArray)
```

VB

The **Sort** method can also accept a class defining the comparison mechanism to apply to that array, if you want to override the default sorting that will be provided. You can make your own comparison mechanism by defining a class that implements the **IComparer** interface. By defining a comparer in this manner, you can specify different sorting techniques.

The following code example defines a comparer that allows you to compare the names, rather than ages, inside the **MyType** class. This means that you won't be able to apply the comparer to other classes, but you can define different modes of sorting this specific object.

```
class MyTypeNameComparer: Implements IComparer

        Public Function CompareTo(ByVal x As Object, ByVal y As Object) As Integer _
                                Implements IComparer.Compare

                If Not (TypeOf x is MyType And TypeOf y is MyType) Then
                                Throw New ArgumentException _
                                ("The objects to compare must be of type 'MyType'")
                End If

                Dim tempObj1 As String = CType(x, MyType).Name
                Dim tempObj2 As String = CType(y, MyType).Name

                CompareTo = tempObj1.ToString().CompareTo (tempObj2.ToString ())
        End Function
End Class
```

VB

To use the comparer, you have to pass **Array.Sort** the comparer as the second parameter. After it is sorted, you can use a foreach (For Each in VB) statement to iterate through the array, and print out the elements to the Console, formatting the information for easier reading.

```
Array.Sort(myTypeArray, new MyTypeNameComparer())

Dim t As MyType

For Each t in myTypeArray
        Console.WriteLine("Name = {0, -7}, Age = {1}", t.Name, t.Age)
Next t
```

VB

Note that the **Sort** method also allows you to sort only a specified section of an array. See the .NET Framework SDK for more information on **Sort**, **Reverse**, and other **Array** functions.

Summary

1. To sort a set of like elements (specifically, an array) in ascending order, use the **Sort** method on an array. The basis of the sort used will be the datatype of the elements in the array (datatype of the array).
2. To sort the elements of an array in descending order, use the **Sort** method, followed by the **Reverse** method.
3. To sort your own arrays of objects, implement the **IComparable** interface on that object. You have to define a **CompareTo** method inside your object in order to specify how this object should be

compared.

4. To define a specific way to compare objects, create a class that implements the **IComparer** interface. Inside this class, you must include a **Compare** method that accepts two objects to compare. It helps to know how you want to compare those objects, and which objects should be allowed to use the interface you define.

Base Data Types

Sort an array
Format a string
Parse datatypes
Compare references
Compare object values
Use mathematical functions

Back to Index

Get URL for this page

How Do I...Format basic types?

This sample illustrates how to format base data types for display. Traditionally, formatting the user can see, such as brackets and the dash in a phone number, has not been viewed as the kind of information you want to store in a permanent datastore. The formatting provided in these demonstrations enables you to take unformatted information and make it easier for the user to read.

The following section includes examples of formatting dates, integral and floating-point numbers. You can also find out how to customize formatting for your own types of data, and adding your own formatting codes. To support formatting, your object should implement the **IFormattable** interface.

VB StringFormat.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

When using the **Console** object, you can request it to write out a line of information to the standard output stream (unless you have previously requested a different output location). Using the **Write** or **WriteLine** methods, you can give **Console** a string with parameters you want to be printed, referenced inside the string. This is a simple way of formatting, where you are requesting nothing to be done to the information you have provided, except embed it in the string at the specified place.

```
Dim strFirst As String = "John"
Dim strLast As String = "Doe"

'  the output from this line displays 'My name is John Doe'
Console.WriteLine("My name is {0} {1}.", strFirst, strLast)
```

VB

Suppose you have some tabular information you want to make sure is lined up from one row to the next. You could use tabs to achieve this, but tabs are a predefined size. Instead, inside your embedded instruction in the string you passed to **WriteLine**, include another tag specifying the size of the parameter you are passing. If the actual size (or length) of your parameter is shorter than this size, it will be formatted with space to fill out the area leftover. A negative sign is on the specified size; this indicates that the text for that parameter should be left-aligned. Therefore if you want it right-aligned, omit the negative sign.

In the following example, two formattable parameters are defined. The first should take a minimum of four columns, and the second, a minimum of 20 columns. Note that if the strings you are formatting exceed this minimum, your results will not look as good.

```
Dim names() as String = {"Mary-Beth", "Aunt Alma", "Sue", _
                "My Really Long Name", "Matt"}

Dim intCount as Integer

For intCount = 0 to 4 Step 1

        '  the | in this example are included simply to show you
        '  the boundaries of the formatting
        Console.WriteLine ("| {0,-4}{1,-20} |", k, names(k))
Next intCount
```

VB

Of course, you don't only want to be able to format string information. The ability to format dates can save you a lot of time and effort. Imagine having to figure out how to change a short date format such as 01/06/1996 into a full string naming the day and the month. Fortunately, formatting can do all of this for you.

Dates support the **IFormattable** interface. This interface provides access to one method: the **Format** method. To use the **Format method**, you have to know the format code you want to use (such as "d" for the short date format). The format code you specify determines what will happen when using the **Format** method. A short sub-selection is displayed here; see the .NET Framework SDK for information on each of the format specifiers.

```
Dim d As DateTime = DateTime.Now

Console.WriteLine ("{0,-2} {1}", "d", d.Format ("d", null) )  '  short date
Console.WriteLine ("{0,-2} {1}", "D", d.Format ("D", null) )  '  long date
Console.WriteLine ("{0,-2} {1}", "G", d.Format ("G", null) )  '  short date and time
Console.WriteLine ("{0,-2} {1}", "m", d.Format ("m", null) )  '  Month and day only
Console.WriteLine ("{0,-2} {1}", "r", d.Format ("r", null) )  '  GMT time
Console.WriteLine ("{0,-2} {1}", "T", d.Format ("T", null) )  '  time only (hh:mm:ss)
Console.WriteLine ("{0,-2} {1}", "u", d.Format ("u", null) )  '  UTC date and time
```

VB

You can also use 'picture' formats for dates to control what gets displayed if you don't want to use the predefined formats above. A picture format is a string of letters instructing a particular display. For example, 'ddd' instructs the formatter to display the day segment of your **DateTime** object as a three letter selection. 'MMMM' indicates to display the full word for the month. You can then put these picture formats together in your own way, as in the following sample.

```
Dim d As DateTime = DateTime.Now

'  example output: Sat
Console.WriteLine ("{0}", "ddd", d.Format ("ddd", null) )

'  example output: October 17, 2000
Console.WriteLine ("{0}", d.Format ("MMMM dd, yyyy"", null) )
```

VB

In addition, you may need to format numeric information. In these situations, you can use predefined or picture formats (just like you can for dates). Use a hash (#) to denote a number, and if you want to use a particular number of spaces, simply add that many hashes. When using these formats, make sure you only use numeric information, as in the following example (for more information on the formatting codes, see the .NET Framework SDK.

```
Dim dblNum As double = 12.9625
Dim intNum As Integer = 73
Dim dblPercent As double = 0.25

' predefined formats
Console.WriteLine ("{0,-8} {1}", "c", dblNum.Format ("c", null) )  ' currency
Console.WriteLine ("{0,-8} {1}", "d", intNum.Format ("d", null) )  ' Decimal Format
Console.WriteLine ("{0,-8} {1}", "e", dblNum.Format ("e", null) )  ' Engineering (scientific)
Console.WriteLine ("{0,-8} {1}", "f", dblNum.Format ("f", null) )  ' Fixed Point format
Console.WriteLine ("{0,-8} {1}", "g", dblNum.Format ("g", null) )  ' General Format
Console.WriteLine ("{0,-8} {1}", "n", dblNum.Format ("n", null) )  ' add comma separators
Console.WriteLine ("{0,-8} {1}", "x", intNum.Format ("x", null) )  ' Hexadecimal

' pictures formats
Console.WriteLine ("{0,-8} {1}", "#", intNum.Format ("#", null) )        '  the number
Console.WriteLine ("{0,-8} {1}", "###", intNum.Format ("###", null) )    '  take three places
Console.WriteLine ("{0,-8} {1}", "#.00", intNum.Format ("#.00", null) )  '  decimal holder on the end
Console.WriteLine ("{0,-8} {1}", "D4", intNum.Format ("D4", null) )      '  leading zeroes
Console.WriteLine ("{0,-8} {1}", "%#", dblPercent.Format ("%#", null) )  '  percentage
```

VB

You can also use formatting to get both name and value information about the entries in an enumeration (enum). An enumeration defines names that programmers can use to identify constant values (the constant integer values need not even have a specific number: the system can work it out for you). This allows you to make code that is more readable. For example, is it easier to say 'Car.Color = Green' or 'Car.Color = 67'? The former is far easier to read and understand. Enumerations have their own fields and methods that allow you to retrieve the values and names in a particular enum; however, you can achieve the same objective using formatting. In the following example, an enum called **Color** is defined, and then use formatting to print the values, and names inside that enum.

```
'  an enum to be used for Color
Public Enum Color

        Red
        Blue
        Green = 27      '  if need be, you can specify a value
                        '  (27 is arbitrary). Otherwise, the system will
                        '  define the value. Red is given 0, Blue 1, etc.
End Enum

'  now, to format the names and values.
Console.WriteLine ("Name: {0}, Value: {1}", Color.Red.Format(), Color.Red.Format ("D", null))
Console.WriteLine ("Name: {0}, Value: {1}", Color.Blue.Format(), Color.Blue.Format ("D", null))
Console.WriteLine ("Name: {0}, Value: {1}", Color.Green.Format(), Color.Green.Format ("D", null))
```

VB

Now that you know about predefined and picture formats, you can try a few things out. You may be asking the question: what if you want to format information for a class you have made, in your own way? You can do this by making a class that implements the **IFormattable** interface, and make sure you include a **Format** method in this class that accepts a string, and an **IServiceObjectProvider**. The string that gets passed in is the format code, so you will have to make sure that people know the valid codes to pass your object when calling this function. For most demonstrations, the IserviceObjectProvider can be null. Below is a class, **MyType**, which defines its own formatting. **MyType** accepts an integer and allows you to format that integer as a binary (b), octal (o), decimal (d), or hexadecimal (h) number.

```
Public class MyType : Implements IFormattable

        Private _value As Integer ' to store the value internally

        Public Sub New(value As Integer)
                MyBase.New()
                _value = value
```

```
        End Sub

        '  This is the formatting method called by String.Format.
        '  In it, look for the "b" format  and then
        '  fall through to Int32's format for anything else you don't know about.
        Function Format(ByVal formatString As String, _
                                ByVal sop As System.IServiceObjectProvider) As String

                If (formatString.Equals ("b")) Then
                        Format = Convert.ToString (_value, 2)
                Else If (formatString.Equals ("o")) Then
                        Format = Convert.ToString (_value, 8)
                Else If (formatString.Equals ("d")) Then
                        Format = Convert.ToString (_value, 10)
                Else If (formatString.Equals ("h")) Then
                        Format = Convert.ToString (_value, 16)
                Else
                        Format = _value.Format (format, sop)
                End If
        End Function
End Class
```

VB

Having defined your own object, and formatting, you can then use the format method of that object as in this sample.

```
Dim t As MyType = new MyType (43)

Console.WriteLine ("{0}", t.Format ("b", null) )  ' binary
Console.WriteLine ("{0}", t.Format ("o", null) )  ' octal
Console.WriteLine ("{0}", t.Format ("d", null) )  ' decimal
Console.WriteLine ("{0}", t.Format ("h", null) )  ' number
```

VB

You can also define your own Formatter if you need to. This allows you to define formatting rules that can be used across many classes, whereas the previous example applies formatting rules to a specific class (namely, that in which it was defined). The following example defines the general rule for creating a formatter. The critical steps are to implement the **ICustomFormatter** interface, and provide a **Format** function that defines your formatting rule.

```
Public Class BaseFormatter : Implements IServiceObjectProvider, ICustomFormatter

'  ...

        Public Function Format (formatString as String, arg As Object, _
                sop As IServiceObjectProvider)  as string implements ICustomFormatter.Format

                '  verification code here....

                formatString = formatString.Trim (new char () {"B"})
                Dim b As Integer = Convert.ToInt32 (formatString)
                Format = Convert.ToString ((int)arg, b)
        End Function
End Class
```

VB

Using the custom formatter is slightly different, however. To do this, call the format function on string, and pass the new formatter in the appropriate position (the final parameter). In this example, the 'B8' instructs the custom formatter to format the number passed in as an octal number.

```
Console.WriteLine (String.Format _
                ("{0} in the custom B8 format is {1:B8}", _
                new object () {100,100}, new BaseFormatter () ))
```

VB

Summary

1. Formatting provides you with a quick and easy way to format information you want to display to the user. It works well with raw information straight from databases, which you want to put into a more human-readable format, for display purposes.
2. You can use the standard formatting provided with **Console.Write** and **Console.WriteLine** methods to include parameters to pass to these methods to format. This standard formatting can be extended to provide tabular formatting, as well as other formats.
3. Dates can be formatted in a wide range of predefined styles, and you can also use 'picture' formatting, to define your own stylized formats.
4. Numbers can be formatted in much the same way as dates, but instead using a hash (#) to denote numeric formatting
5. If required, you can include your own formatting on a class, by implementing the **IFormattable** interface on that class, and defining a **Format** method within it.
6. If you want to provide custom formatting and make it available to many different classes, you can create a class that

implements **ICustomFormatter**.

Base Data Types
Sort an array
Format a string
Parse datatypes
Compare references
Compare object values
Use mathematical functions

Back to Index

Get URL for this page

How Do I...Parse datatypes?

Moving information from one data type to another is essential when dealing with data. Parsing information from a string into another data type is available with almost all the nonstring data types you will work with, through the use of the **Parse** method. You can also convert data into Strings using the **ToString** method. This demonstration includes examples of methods inside the **Convert** class, to convert nonstring data types to and from each other.

VB Parse.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

First, you need to convert information from some arbitrary base data type into a string. The **Object** class (the class from which all other objects derive) has a **ToString** method that is automatically inherited by all other objects. Converting an object to a string may be less useful (if you don't override it, expect to receive the name of the class), so the derived class must override the method, and make it more useful. In its simplest form, the **ToString** method needs to be applied to an instance of a particular object, with no parameters, as in the following sample.

```
Dim intExample As Integer = 76
Dim strExample As String = intExample.ToString()

'  example one, having put the value in a string
Console.WriteLine("The string value is " & strExample)

'  example two, converting the integer to a string on the fly
Console.WriteLine("The string value is " & intExample.ToString())

'  example three, not converting the integer to a string at all???
Console.WriteLine("The string value is " & intExample)
```

VB

In the previous example, there are three lines being printed to the standard out stream. Which ones do you think will work? Believe it or not, all three lines will result in the same output to the output stream. The first one is hopefully pretty clear; having converted the integer to a string, you can concatenate that string to another string. The second one is fairly intuitive as well, if you remember that the **ToString** method on intExample returns a string, and therefore concatenates this return value onto the string being printed.

But what about the third example? This may look like it should not work, but in fact, there are some helpful routines that will convert nonstring data types into strings for you automatically (this is known as 'implicit conversion'). Don't rely on implicit conversion, however; until you learn where implicit conversion does and does not occur, you should use either of the first two examples.

To convert from a String to an integer, a Boolean, or some other non-String data type, use the non-instantiated **Parse** method, passing in the String you want to convert. The following lines are examples of using parse. To print the String you just parsed out to the screen, it needs to be converted back to a String by the **WriteLine** method.

```
Dim strExample1 As String = "true"
Dim strExample2 As String = "00.987"
Dim strExample3 As String = "127"
Dim strExample4 As String = "42"

Console.WriteLine( "Parsing {0} = {1}", strExample1, Boolean.Parse(strExample1) ) '  to boolean
Console.WriteLine( "Parsing {0} = {1}", strExample2, Double.Parse(strExample2) )  '  to double
Console.WriteLine( "Parsing {0} = {1}", strExample3, Int32.Parse( strExample3) )  '  to Int
Console.WriteLine( "Parsing {0} = {1}", strExample4, Byte.Parse(strExample4) )    '  to byte

Dim strExample5 As String = "three"

' catching any exceptions...
Try
     Console.WriteLine("The integer value is " & Int32.Parse(strExample5))
catch e As Exception
     Console.WriteLine("The value '{0}' cannot be converted to a integer value.", strExample5)
End Try
```

VB

Now that you can convert information to and from Strings, the next issue is how to convert information from other data types. Fortunately, there is a **Convert** class designed specifically for this. The **Convert** class has a number of static methods, beginning with 'To' and ending with the target data type of the Object you want to convert; for example, **ToInt32** or **ToByte**. If successful, the returned variable from the method you use is an object of the target data type. Not all conversions will be possible when using the **Convert** method, so use a try...catch block if you think you may get uncertain results. The following examples demonstrate some of the 'To' methods in the **Convert** object.

```
Dim intExample as Integer = 76

'  the methods inside convert...
Console.WriteLine("Convert.ToString, result = {0}", Convert.ToString(intExample))   '  displays 76
Console.WriteLine("Convert.ToBoolean, result = {0}", Convert.ToBoolean(intExample)) '  displays True
Console.WriteLine("Convert.ToByte, result = {0}", Convert.ToByte(intExample))       '  displays 76
Console.WriteLine("Convert.ToChar, result =  {0}", Convert.ToChar(intExample))      '  displays L
Console.WriteLine("Convert.ToDouble, result = {0}", Convert.ToDouble(intExample))   '  displays 76
```

VB

If you want to convert dates into and out of strings (perhaps for formatting), you can use either the **Parse** method on the **DateTime** data type, or you can use a string's **ToDateTime** method. Both will achieve the same objective. Because dates can often be entered incorrectly by your users, don't forget to include appropriate exception handling to make sure that any conversion errors are caught. If you want to provide more control over parsing a date, see the **ParseExact** method.

Also note in the following sample that the first line forces this parse to work, because dates for the U.S. culture are used. If you want to try this code in your own culture, omit the first line, and then pick dates that make sense for your culture.

```
System.Threading.Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US")

Dim strDate As String = "03/17/77"

'  Simple DateTime parsing
Console.WriteLine("Parsing {0}, using the Parse method.", strDate)

Dim d As DateTime = DateTime.Parse(strDate)
Console.WriteLine("The date parsed as {0}",d)

'  DateTime parsing using ParseExact
Dim strDate2 As String = "Thursday, March 17, 1977"
Console.WriteLine("Parsing {0}, using the ParseExact method.", strDate2)

Dim d2 As DateTime = DateTime.ParseExact(strDate2,"D", DBNull)
Console.WriteLine("The date parsed as {0}",d2)
```

VB

One great feature of parsing is that it allows you to remove unnecessary or mistyped information. For example, let's imagine that you have asked users to enter their area code as a set of three digits. People often just go ahead and type whatever they want, so users could put brackets and extra spaces on the front of the area code. Parsing can remove that information for you, rather than asking the user go back and retype the number, or perform some checking on the number yourself. In the following example, the enumeration **NumberStyles** is used to identify which style should be allowed from the user. The Bitwise '|' (BitOr in VB) is used as an 'or', indicating that the user may enter this string with parentheses, leading spaces, or trailing spaces, and the number is still acceptable. Note that a space in the middle of the number will not meet these requirements.

```
Dim toBeParsed As String = "   (555)   "

Console.WriteLine("Parsing the string {0}, with whitespace and parentheses", toBeParsed)

Dim j As Integer = Int32.Parse(toBeParsed, _
                NumberStyles.AllowParentheses BitOr NumberStyles.AllowLeadingWhite BitOr _
                NumberStyles.AllowTrailingWhite)

'  note that the output may well be negative, because of the parentheses
Console.WriteLine("The string parsed to {0}.", j)
```

VB

Summary

String and data type conversion and manipulation is essential when dealing with information on a daily basis. Because most screen inputs are automatically interpreted as textual or string information, it is often necessary to convert this information into other data types before you can use it appropriately. When dealing with conversions between base data types, keep the following items in mind:

1. Use the **ToString** method to convert nonstring data types into strings.
2. Avoid implicit conversions where possible.
3. To convert string information into other data types, use the **Parse** method on the target datatype of your string conversion (for example, Boolean.Parse).
4. To convert nonstring information from one data type to another, use any of the provided methods inside the **Convert** class, designed for this purpose. These methods begin with the word 'To' followed by the name of the target data type.
5. Parsing can provide richer functionality when combined with the **NumberStyles** enumeration, to help parse incorrectly entered information into numerical data.

Base Data Types
Sort an array
Format a string
Parse datatypes
Compare references
Compare object values
Use mathematical functions

Back to Index

Get URL for this page

How Do I...Compare object references?

This topic illustrates the differences between comparing base data types using the operator == (= or 'Is' in VB) and using the **Equals** method. The exception case, **Strings**, is also demonstrated and discussed. The difference between reference and object equality is also covered, to help you determine where to use == (= or 'Is'), or **Equals**.

VB Equals.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Comparing objects and base data types for equality is a common task. However, it is essential to know whether you are comparing the actual values of the items or the items themselves. To clarify the situation, the following code example demonstrates creating two **CultureInfo** objects, and then compares those objects using the **Equals** method and the == (= or 'Is' in VB) operator.

```
Imports System
Imports System.Globalization ' needed when using CultureInfo

Class test

    Public Shared Sub Main()

        Dim cu As CultureInfo = new CultureInfo("en-NZ")
        Dim ci As CultureInfo = new CultureInfo("en-NZ")

        Console.WriteLine( cu Is ci );        ' results in True, or False being printed to output?
        Console.WriteLine( cu.Equals( ci ) ); ' results in True, or False being printed to output?
    End Sub
End Class
```

VB

You might expect both cases in the previous demonstration to return the value **True**. However, if you ran this code you would actually see the words **False** and **True**. This can take an unsuspecting programmer by surprise, but there is a valid reason for this result. Recall that when you create a new instance of an object, you actually only have a reference to that object. The object is made in the background somewhere, but in the above example when cu or ci is used, it is in reference to **CultureInfo** objects. Think of it as a 'name' for the objects. For example, you could make a car called 'MyCar'. To change MyCar's color, you would need to do more than change the color of the word MyCar. Instead, you need to go to the actual car and repaint it.

Now imagine two people buy cars of the same kind and name them MyCar and YourCar. Even if they are the same model, color, and come with the same extras, they are not the exact same car. In the same way, two **Object** references do not necessarily refer to the same object, and the above **CultureInfo** sample is analogous to buying two different cars. They are the same kind of object, but they are not the same actual objects.

To ask the computer to see if two different object references refer to the same object, use == (= or 'Is' in VB). This is why this scenario in the above sample resulted in **False**. In contrast, when you use the **Equals** method, you are asking if the objects have the same information in them, regardless of the actual objects they refer to.

The following code example demonstrates creating an object that is actually made with the same reference of another object. This means the second object refers to the one instance of the object. To do this, assign the second object to the first object when a new instance of it is created.

```
Dim firstVal CultureInfo = new CultureInfo("en-NZ")   ' new object
Dim secondVal CultureInfo = new CultureInfo("en-US")  ' new object
Dim thirdVal CultureInfo = new CultureInfo("en-NZ")   ' new object

' same reference as the FIRST object! fourthVal
' and firstVal both refer to the same object
Dim fourthVal As CultureInfo = firstVal

Console.WriteLine( firstVal Is secondVal )        ' using == ... true/false? False
Console.WriteLine( firstVal Is thirdVal )         ' using == ... true/false? False
Console.WriteLine( firstVal Is fourthVal )        ' using == ... true/false? True

Console.WriteLine( firstVal.Equals (secondVal) )  ' using Equals ... true/false? False
Console.WriteLine( firstVal.Equals (thirdVal) )   ' using Equals ... true/false? True
Console.WriteLine( firstVal.Equals (fourthVal) )  ' using Equals ... true/false? True

' Note that Equals and Is did NOT result in the same output for all samples
```

VB

It is important to understand that all object references work the same way, including an actual **Object** object. However, all base

data types do not work in the same way. Imagine that there are two integers int1 and int2, which both have the value 1. Is it helpful to think of these as separate objects? Maybe in some scenarios, but in general, the only significant piece of information these integers have is their value. Therefore, when you ask if int1 == int 2 (int1 = int2 in VB), you actually want this to evaluate to true in this case. So for base data types, == and Equals are the same thing. The following code example demonstrates this for Booleans.

```
Dim firstVal As Boolean = false ' new base datatype instance
Dim secondVal As Boolean = true ' new base datatype instance
Dim thirdVal As Boolean = false ' new base datatype instance

' copies the value of the first instance to this instance
' does NOT refer to the same object
Dim fourthVal As Boolean = firstVal

' NOTE: Do not use the Is keyword to compare value types in Visual Basic,
'       but instead use the = operator, as in this example. Only use Is when you
'       are dealing with non base-datatype object references
Console.WriteLine( firstVal = secondVal )         ' using == ... true/false? False
Console.WriteLine( firstVal = thirdVal )          ' using == ... true/false? TRUE     <==
Console.WriteLine( firstVal = fourthVal )         ' using == ... true/false? True

Console.WriteLine( firstVal.Equals (secondVal) )  ' using Equals ... true/false? False
Console.WriteLine( firstVal.Equals (thirdVal) )   ' using Equals ... true/false? True
Console.WriteLine( firstVal.Equals (fourthVal) )  ' using Equals ... true/false? True

' Note that Equals and == resulted in the same results, for each comparison
```

VB

String objects are inherently 'immutable'. Although they act like objects in many ways, you can think of them as base data types. For example, say you make a string, MyName, and then make a new string, YourName, which you could set = to MyName. If you then changed YourName so it was different, MyName would not change. This is demonstrated in the following example. Base data types operate in the same way.

```
Car myCar = new Car("Red")        ' My car, which is red!
Car yourCar = myCar               ' Your car, referring to the SAME car...
yourCar.Color = "Pink"            ' Change the color of your car...
Console.WriteLine( myCar.Color )  ' what color is MyCar now?

String myName = "Kayleen"         ' Make my name
String yourName = myName          ' Make your name, which appears to reference the same string?
yourName = "Mary"                 ' Change your name...
Console.WriteLine( myName )       ' What will my name be now?
```

VB

Summary

Using the **Equals** method on objects and primitive data types always compares the values inside the items being compared. If you want to compare whether an object's reference (that is, the actual object that it refers to) is the same as another object, use the == (= or 'Is' in VB) operator. Note that Strings are the 'in-between' case, and because of their immutable (unchangeable) nature, they are treated in the same way as base data types.

Base Data Types
Sort an array
Format a string
Parse datatypes
Compare references
Compare object values
Use mathematical functions

Back to Index

Get URL for this page

How Do I...Compare the value of two objects?

This sample illustrates how to compare any two values by using the **IComparable** interface that all the base types implement. The **Max** method shown below returns the maximum of any two values that implement **IComparable**. The **CompareTo** method on **IComparable** has the following return values: < 0 if the value is less than the current instance, == 0 (= 0 in VB) if the value is equal to the current instance, or > 0 if value is greater than the current instance. A custom implementation of **IComparable** is also shown.

VB Compare.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

To compare two objects to each other, they should implement the **IComparable** interface. This interface has a **CompareTo** method that allows you to compare one object to another and determine if one object is less than, greater than, or equal to another object. For the moment, assume you are dealing with string or numeric data, where the rules are already known.

The base data types all support the **IComparable** interface, which means they can all be compared to one another. The following code example demonstrates a fairly simple routine called Max, which accepts two objects, both of which support **IComparable**. Using the **CompareTo** method of those objects, you will do some standard comparisons to determine which one is the greater of the two (which is then returned out of the method). While what you pass into this routine are the objects you want to compare, inside the routine you only deal with the **IComparable** nature of those objects, which is why the data type of the return value and the parameters is **IComparable**, not **Object**. If you try to use this particular routine with objects that do not support **IComparable**, you will get a compile error.

```
Public Shared Function Max (val1 As IComparable , _
                        val2 As IComparable) As IComparable

        If (val1.CompareTo(val2) > 0)
                Return val1  '  val1 > val2
        End If

        If (val1.CompareTo(val2) < 0)
                Return val2  '  val1 < val2
        End If

        If (val1.CompareTo(val2) = 0)
                Return val1  '  val1 = val2, return val1
        End If
End Function
```

VB

To use the above example, you can pass into the Max routine any two objects that support **IComparable**. You can flow through the Max routine yourself to determine which object will be passed back as the 'max'. Remember that the nature of how these objects are compared is defined within the objects themselves.

```
Console.WriteLine ( Max (5, 10) )             '  comparing integers
Console.WriteLine ( Max (4.3, 2.5) )          '  comparing doubles
Console.WriteLine ( Max ("Mathew", "Mary") )  '  comparing strings
```

VB

To compare your own objects, you need to implement the **IComparable** interface on your object. Then, you need to create a **CompareTo** method inside your object, which defines the rules for comparing your object to other objects. Often, you will want to compare your object only to other objects of the same type. For instance, how do you compare a car object to a tree object? You might suggest comparing the weights of the objects, but you could also compare the shape or color of the objects. This is what you need to decide when defining the **CompareTo** method, and unless you are certain that comparisons to other types are acceptable, you should inform the user that such comparisons are not supported, or are undefined.

In the next sample, you make an object that implements the **IComparable** interface, and also defines a **CompareTo** method for comparing itself to other objects. Note that you can only compare this object to its own data type, and not other data types. If the comparable object is not **MyType**, an **ArgumentException** is thrown.

```
Public Class MyType : Implements IComparable

        Private _value As Integer

        Public Sub New(value As Integer)
                MyBase.New()
                _value = value
        End Sub

        Public Function CompareTo (o As Object) As Integer _
                                Implements IComparable.CompareTo
                If (Not (TypeOf(o) is MyType))
                        Throw New ArgumentException ("o must be of type 'MyType'")
                End If

                Dim v As MyType = CType (o, MyType)

                Return _value - v._value
        End Function

        Overrides Public Function  ToString () As String

                Return String.Format ("MyType({0})", _value)
        End Function
End Class
```

VB

Now that you have defined the object, you can use the above Max routine for the base data types. The following sample compares two objects of **MyType**. It's a good idea to flow through the logic, and see how the **IComparable** interface is used within the **Max** method, to compare one of the objects to the other. Note that inside the **Max** method, you invoke **CompareTo** on the **MyType** object.

```
Dim t1 As MyType = new MyType (12)
Dim t2 As MyType = new MyType (17)

Console.WriteLine ( Max (t1, t2) )  '  comparing MyTypes
```

VB

Summary

All the base data types can be compared because they support the **IComparable** interface. To compare objects, use the **CompareTo** method on one of the objects, passing the object you want to compare as a parameter. The rules for the comparison are defined by the **CompareTo** method for the object in question. You can create your own objects that may be compared to each other by implementing the **IComparable** interface on the object, and defining a **CompareTo** method within it.

Base Data Types
Sort an array
Format a string
Parse datatypes
Compare references
Compare object values
Use mathematical functions

Back to Index

Get URL for this page

How Do I...Use mathematical functions?

This example demonstrates using the methods and constants built into the **Math** class. The methods in the **Math** class provide you with most of the standard functions you will want to use in daily computations, and are all provided as non-instantiated methods, perfect for quickly being able to perform the function you require. See the .NET Framework SDK for a full listing of all the functions you can access using the **Math** class. At the end of this page is an example allowing you to invoke all the methods on the **Math** class, which uses the **Reflection** namespace to call the methods dynamically.

VB Math.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The **Math** class contains a number of standard functions for assisting you with mathematical operations. Because you don't want to actually have a 'Math' object lying around in memory, all the tools you are going to use through **Math** are provided as non-instantiated functionality. All the methods are intuitive and easy to use.

The following example demonstrates using the **Max** and **Min** functions. These functions allow you to determine which is the larger or smaller of two numbers, and returns the identified number back out of the called function.

```
Dim i As Integer = -10

Dim d1 As double = 39.25
Dim d2 As double = -27.84

Console.WriteLine( Math.Max(i, d1) )   ' returns 39.25
Console.WriteLine( Math.Max(d2, i) )   ' returns -10
Console.WriteLine( Math.Max(d1, d2) )  ' returns 39.25

Console.WriteLine( Math.Min(i, d1) )   ' returns -10
Console.WriteLine( Math.Min(d2, i) )   ' returns -27.84
Console.WriteLine( Math.Min(d1, d2) )  ' returns -27.84
```

VB

You may think the previous example goes a little overboard, since a demonstration of just two numbers would give you the idea. The purpose is to point out that the data types, and the order of those data types, is different in each example. In the first call to the **Max** function, an integer and a double are passed, in that order, and then you call the same function, with a double, and then an integer. The **Math** class provides many overloaded methods, so you can pass many different data types in different orders, and still get the functionality you require, as demonstrated above.

The **Ceiling** method rounds a number up to the first whole number larger than the number you pass, and the **Floor** method rounds a number down to the first whole number smaller. The **Abs** method returns the absolute value of the number you pass (defined as the non-negative distance from zero), and the **Sqrt** method returns the square root of a number. Note that rather than throwing an exception, if you pass the **Sqrt** method a negative number then the result is 'NaN', signifying that the result is 'Not a Number'.

```
'  refer to the previous sample for the variable declarations being used...

'  NB: Floor and Ceiling are fairly meaningless with integer values,
'  since by definition, they are already whole numbers!
Console.WriteLine( Math.Floor(i) )      ' returns -10
```

```
Console.WriteLine( Math.Floor(d1) )     ' returns 39
Console.WriteLine( Math.Floor(d2) )     ' returns -28

Console.WriteLine( Math.Ceiling(i) )    ' returns -10
Console.WriteLine( Math.Ceiling(d1)) )  ' returns 40
Console.WriteLine( Math.Ceiling(d2)) )  ' returns -27

Console.WriteLine( Math.Abs(i) )        ' returns 10
Console.WriteLine( Math.Abs(d1) )       ' returns 39.25
Console.WriteLine( Math.Abs(d2) )       ' returns 27.84

Console.WriteLine( Math.Sqrt(d1) )      ' returns 6.26
Console.WriteLine( Math.Sqrt(i) )       ' returns NaN
```

VB

Using the **Math** class is fairly straightforward. You can perform many other functions using other methods, including sin, cos, and tan functions, and logarithmic functions. There are two constant values defined in the **Math** class: PI and E. PI is the constant ratio between a circle and its diameter, and is provided for quick and fairly accurate computations where it is required. E is the natural exponent (generally written as a lowercase e) also used in many mathematical operations. The following code example demonstrates how to invoke each constant.

```
Console.WriteLine(" e equals {0}", Math.E )
Console.WriteLine(" PI equals {0}", Math.PI )
```

VB

This final example allows you to invoke all the static methods on the **Math** class. It uses members in the **Reflection** namespace to determine the methods available dynamically, and builds a list of all the methods you can select to invoke. For details on the code in this sample, see the How Do I samples listed under the Reflection topic.

VB MathRefl.aspx

[Run Sample] | [View Source]

The following example demonstrates how to sort datatypes using the Array.Sort method. Different arrays of datatypes are created, and the arrays are displayed unsorted, sorted (ascending), sorted (descending where displayed), and in the case of the MyType class, sorted by Age, and by Name. Being able to sort MyType in different ways demonstrates how to define your own comparers for sorting objects.

```
The unsorted int array:
{43, 12, 1, 17, 23, 36, 11}

The sorted int array:
{1, 11, 12, 17, 23, 36, 43}

The sorted int array, in descending order (after a Sort and Reverse):
{43, 36, 23, 17, 12, 11, 1}
-----------------------------------

The unsorted double array:
{1.5, 4.3, 2.5, 6.9, 2.01, 0.04}

The sorted double array:
{0.04, 1.5, 2.01, 2.5, 4.3, 6.9}
-----------------------------------

The unsorted string array:
{Michael, mary, Robert, Abigail, lorne, Therese, terrence, Samantha}

The sorted string array:
{Abigail, lorne, mary, Michael, Robert, Samantha, terrence, Therese}

The sorted string array, in descending order:
{Therese, terrence, Samantha, Robert, Michael, mary, lorne, Abigail}
-----------------------------------

The unsorted MyType array:
Name = Max    , Age = 23
Name = Lisa   , Age = 47
Name = Andrea , Age = 35
Name = Mel    , Age = 29
Name = Andrew , Age = 32
Name = Ahmed  , Age = 35

The MyType array, sorted by age (default sort):
Name = Max    , Age = 23
Name = Mel    , Age = 29
Name = Andrew , Age = 32
Name = Ahmed  , Age = 35
Name = Andrea , Age = 35
Name = Lisa   , Age = 47
```

```
The MyType array, sorted by Name (separately defined comparer):
Name = Ahmed  , Age = 35
Name = Andrea , Age = 35
Name = Andrew , Age = 32
Name = Lisa   , Age = 47
Name = Max    , Age = 23
Name = Mel    , Age = 29
```

Collections

Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I...Iterate over a collection?

This topic illustrates how to create and use an **ArrayList**. An important part of this discussion is how to use the foreach (For Each in VB) command to loop through the list. An **ArrayList** is a member of the **Collection** namespace, and can accept any type of object as its elements. This topic also covers the methods and properties of the **ArrayList** object.

VB List.aspx

[Run Sample] | [View Source]

Collections are very similar to arrays. On the whole, collections can be viewed as specialized arrays, with a few more methods and properties to help you out. In this demonstration, you will work with one particular collection, **ArrayList**.

ArrayList can take all kinds of objects, but when working with collections, it is beneficial to try to use only one data type. The first piece of sample code you are going to look at is creating the list. This is an easy, one line piece of code in which you use the constructor for **ArrayList** to make a new instance of this collection. Unlike an array, there is no need to figure out how many elements the list is going to take: the **ArrayList** can do all that for you as you add elements to it.

```
'  declare the list, with a default capacity of 16
Dim al as New ArrayList()
```

VB

When you make an **ArrayList**, it has no elements in it, but it has the ability to accept up to 16 elements. If you try to add a 17th element, the **ArrayList** helps you out by doubling its capacity. This can be useful to know because there is a certain performance hit associated with increasing the collection in this way. If you know how many elements your collection will need, you can set the capacity in advance, saving on that performance issue (however small).

To prove that the default capacity of **ArrayList** is greater than 16, you need to ask **ArrayList** what its capacity is, and how many elements it has. You can do this by accessing properties of the same name on the list.

```
'  display the initial count, and capacity. 0 and 16
Console.WriteLine("The initial Count of our list is {0} " & _
                  "and it's Capacity is {1}", al.Count, al.Capacity)
```

VB

Try this out yourself to make sure you agree. You can set the capacity property, as long as you don't go below the number of elements currently in the list, or below zero. Count is read-only, since it reflects the actual number of elements in the list, which is determined by how many you add.

If you want to add items to the list, use the **Add** method, which takes any object as an argument and places that object at the end of the list. If you want to place an object at a particular place in the list, use the **Insert** method, specifying the index to place the new object at, and the object to insert.

Not only can you add elements to your list, you can also take elements away from it. To do this, use the **Remove** method, specifying the object to remove (you should use an object reference to do this), or the **RemoveAt** method, passing the index of the element to remove. The following code example demonstrates adding and removing objects.

```
'  declare an Integer object, to add to the list
dim intTemp As Integer = 12

' add elements to the list
al.Add("Apple")
al.Add(intTemp)
al.Add( New MyType() )  '  a specific object we made ourselves

Console.WriteLine("Removing the integer object...")
'  remove a specific element, using the Remove method
al.Remove(intTemp)

Console.WriteLine("Removing the MyType object...")
'  remove a specific object, using RemoveAt
al.RemoveAt(2)  '  remove the MyType. Remember, we just removed intTemp,
                '  which reduced all element indexes by 1.

Console.WriteLine("Inserting new object...")
'  add an object at a specific location
al.Insert(0, "Grapes")

' display the count, and capacity
Console.WriteLine("After modifying our list, Count is {0} " & _
                "and Capacity is {1}", al.Count, al.Capacity)
```

VB

Notice that the only objects left in the list after running the above code are strings. This is going to help with the foreach statement. After making the **ArrayList**, you can iterate through that list by making a temporary object that can point at each element of the list in turn. The data type of the temporary 'pointing' object can help you out. Because you only have Strings, you can make it a **String** object. If you are uncertain, use a generic Object. After you have printed out the elements in the list, use the **Clear** method to remove all the elements in the list. The following code example demonstrates these concepts.

```
'  note that because we know only Strings are left in our list,
'  we can use a String object in our For Each statement. If you
'  are at all uncertain, use For Each objItem in al
Dim strItem As String

for Each strItem in al
        Console.WriteLine(strItem)
Next strItem

'  remove all elements from the list
al.Clear()
```

VB

Remember that an **ArrayList** has many other useful methods and properties that can help you manipulate your list. Use the .NET Framework SDK to determine how these elements work.

Collections

Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I...Iterate over a collection?

This topic illustrates how to create and use an **ArrayList**. An important part of this discussion is how to use the foreach (For Each in VB) command to loop through the list. An **ArrayList** is a member of the **Collection** namespace, and can accept any type of object as its elements. This topic also covers the methods and properties of the **ArrayList** object.

VB List.aspx

[Run Sample] | [View Source]

Collections are very similar to arrays. On the whole, collections can be viewed as specialized arrays, with a few more methods and properties to help you out. In this demonstration, you will work with one particular collection, **ArrayList**.

ArrayList can take all kinds of objects, but when working with collections, it is beneficial to try to use only one data type. The first piece of sample code you are going to look at is creating the list. This is an easy, one line piece of code in which you use the constructor for **ArrayList** to make a new instance of this collection. Unlike an array, there is no need to figure out how many elements the list is going to take: the **ArrayList** can do all that for you as you add elements to it.

```
'  declare the list, with a default capacity of 16
Dim al as New ArrayList()
```

VB

When you make an **ArrayList**, it has no elements in it, but it has the ability to accept up to 16 elements. If you try to add a 17th element, the **ArrayList** helps you out by doubling its capacity. This can be useful to know because there is a certain performance hit associated with increasing the collection in this way. If you know how many elements your collection will need, you can set the capacity in advance, saving on that performance issue (however small).

To prove that the default capacity of **ArrayList** is greater than 16, you need to ask **ArrayList** what its capacity is, and how many elements it has. You can do this by accessing properties of the same name on the list.

```
'  display the initial count, and capacity. 0 and 16
Console.WriteLine("The initial Count of our list is {0} " & _
                  "and it's Capacity is {1}", al.Count, al.Capacity)
```

VB

Try this out yourself to make sure you agree. You can set the capacity property, as long as you don't go below the number of elements currently in the list, or below zero. Count is read-only, since it reflects the actual number of elements in the list, which is determined by how many you add.

If you want to add items to the list, use the **Add** method, which takes any object as an argument and places that object at the end of the list. If you want to place an object at a particular place in the list, use the **Insert** method, specifying the index to place the new object at, and the object to insert.

Not only can you add elements to your list, you can also take elements away from it. To do this, use the **Remove** method, specifying the object to remove (you should use an object reference to do this), or the **RemoveAt** method, passing the index of the element to remove. The following code example demonstrates adding and removing objects.

```
'  declare an Integer object, to add to the list
dim intTemp As Integer = 12

' add elements to the list
al.Add("Apple")
al.Add(intTemp)
al.Add( New MyType() )  '  a specific object we made ourselves

Console.WriteLine("Removing the integer object...")
'  remove a specific element, using the Remove method
al.Remove(intTemp)

Console.WriteLine("Removing the MyType object...")
'  remove a specific object, using RemoveAt
al.RemoveAt(2)  '  remove the MyType. Remember, we just removed intTemp,
                '  which reduced all element indexes by 1.

Console.WriteLine("Inserting new object...")
'  add an object at a specific location
al.Insert(0, "Grapes")

' display the count, and capacity
Console.WriteLine("After modifying our list, Count is {0} " & _
                "and Capacity is {1}", al.Count, al.Capacity)
```

VB

Notice that the only objects left in the list after running the above code are strings. This is going to help with the foreach statement. After making the **ArrayList**, you can iterate through that list by making a temporary object that can point at each element of the list in turn. The data type of the temporary 'pointing' object can help you out. Because you only have Strings, you can make it a **String** object. If you are uncertain, use a generic Object. After you have printed out the elements in the list, use the **Clear** method to remove all the elements in the list. The following code example demonstrates these concepts.

```
'  note that because we know only Strings are left in our list,
'  we can use a String object in our For Each statement. If you
'  are at all uncertain, use For Each objItem in al
Dim strItem As String

for Each strItem in al
        Console.WriteLine(strItem)
Next strItem

'  remove all elements from the list
al.Clear()
```

VB

Remember that an **ArrayList** has many other useful methods and properties that can help you manipulate your list. Use the .NET Framework SDK to determine how these elements work.

Collections

Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I....Create and Use a Hashtable?

This example illustrates how to create and use a hash table. A hash table is a collection that consists of key-value combinations, organized into 'buckets' for fast searching. You can search a hash table either by the keys, or by the associated values; however, if fast searching is your objective, searching by key will generally be faster.

VB HashTable.aspx

[Run Sample] | [View Source]

As mentioned in the introduction, a hash table is a type of collection. When you populate a hash table, you give it a key to add to the table, and a value that goes with the key. In the following example, employee id numbers are used for the key, and then the employee names, to give you an idea of the kind of combinations you can use. Note that the key does not need to be numeric: any object is acceptable.

Why is a hash table useful? When one of these key-value pairs is added to the hash table, the hash table looks at the key and gives it a hash code, a number that identifies the key (each key you add gets a hash code). These hash codes are then placed in 'buckets' to help organize the entries in the table. This can help later when you try to find something in the table.

Imagine for a second that these buckets aren't used. If you wanted to find something in an array, you would have to go through every element of the array, looking for the element. You have no idea where it is, so there is no fast way to search through the entries. By contrast, a hash table can search for a particular key faster, because once you give it a key it knows which bucket to look in. This means it only has to look through a subset of all the elements.

It is important to remember that this faster searching mechanism is only available when searching by keys. Searching by a value is available, but is potentially slower than key searching. Therefore, where possible you should perform key searches.

To add elements to a hash table, call the **Add** method, passing in the key-value pairs you want to add.

```
'  Here we'll create a hashtable of employee numbers and employee names
Dim table As Hashtable = New Hashtable()

'  now, we will add elements to the hashtable, as key-value pairs
table.Add(5123, "Jay")
table.Add(1829, "Tom")
table.Add(2882, "Matt")
```

VB

Now that the hash table is populated, you can search for elements in it. You are going to search both by key and value. Note that in the actual example, you will accept a value or key to find from the user, so the implementation is slightly different, but the concepts are the same.

```
'  SEARCHING BY VALUE. Use the method ContainsValue
Dim valueToFind as String = "Jay"
```

```
If table.ContainsValue(valueToFind) Then
        Console.WriteLine("Found {0} in the list.", valueToFind )
End If

'  SEARCHING BY KEY. Use the method Contains
Dim keyToFind As Integer = 5123

If table.Contains(keyToFind) Then
        Console.WriteLine("Found {0} in the list.", keyToFind)
End If
```

VB

Of course, you can do more than simply searching. You can remove entries from the hash table by using the **Remove** method, which you pass a key identifying the element you want to remove. You can also enumerate over the hash table using the Keys collection or the Values collection inside your instance of the hash table.

```
'  using the Remove method to delete a particular item...
table.Remove(5123);

Dim o As Object

'  enumerating over the collection with a foreach statement
'  Note the alternate of iterating through by value is demonstrated
'  For Each o In table.Values
For Each o In table.Keys
        Console.WriteLine(o.ToString())
Next
```

VB

Those are the basics of using a hash table. It is important to know how to iterate through your hash table, once you have it populated with entries. This important topic is covered in the section How Do I...Iterate over a collection? The following code shows you how to use a foreach (For Each in VB) statement to move through our list, and print each of the key-value combinations to the screen. To do this, you also need to reference the **Key** and **Value** properties of each element. The following sample demonstrates iterating through the collection.

```
'  if you are wondering where DictionaryEntry comes from, it is
'  the generic object type of each element in the hashtable
Dim d As DictionaryEntry

For Each d in table
        '  use the Key and Value properties. Note chr(9) is a tab
        Console.WriteLine ("{0}" & chr(9) & "{1}", d.Key, d.Value)
Next d
```

VB

Collections

Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I....Choose a collection to use?

This example discusses the storage procedures and capabilities of three different collections: Queue, SortedList, and Hashtable. The primary purpose of this demonstration is to help developers understand the functions of the different collection types, so that they can choose one that will meet their needs.

VB Collections.aspx

[Run Sample] | [View Source]

At first, you may find it difficult to determine which collection best suits your needs. With experience, you will become familiar with each of the different collections you can use, and even customize your own by extending ICollection or IList. The three collections discussed here are: Queue, SortList, and Hashtable. For more detailed information on Hashtable, see How Do I...Use a hashtable?.

A Queue is an unsorted collection, which is meant to be used on a 'first-in, first-out' basis.

- Do use a Queue when you need entries to be stored in the same order they were entered
- Do use a Queue when you want to be able to easily add elements to the end of the Queue. The **Enqueue** method provides this functionality
- Do use a Queue if you want to easily be able to read and remove the top item from the Queue. This is the capability provided by the **Dequeue** method. If you don't want to remove the element but still what to see what it is, use the **Peek** method
- Don't use a Queue if you need fast searching capabilities. Queue searching is not fast
- Don't use a Queue if you need to remove a particular element specified by you or the user
- Don't use a Queue if you need to be able to reorder or sort the elements.

A SortedList is exactly that: a collection of sorted items. A SortedList consists of key-value pairs, and the sort is organized by the key, not the values.

- Do use a SortedList if you want to sort your items, without having to do much work
- Do use a SortedList if you want to control how the elements are sorted (how they are compared)
- Do use a SortedList if you can use fast searching capabilities of the keys of the SortedList
- Do use a SortedList if you want to be able to remove specific elements (use the **Remove** or **RemoveAt** methods)
- Don't use a SortedList if you need to insert an element in a particular location. It will just be sorted anyway
- Don't use a SortedList if you want to store multiple keys with the same value. All keys inside a SortedList must be unique
- Consider using a different collection (such as Hashtable) if fast searching (particularly by Value) is your highest priority.

A hash table is a set of elements also requiring key-value pairs. It sorts the key-value pairs using an internal algorithm, which is not known to the user. The key-value pairs are stored in 'buckets' which makes searching for values very fast.

- Use a hash table if searching is a priority. Hash tables provide very quick search mechanisms when searching by key, and fairly good searching when searching by

value
- Use a hash table if you want to be able to remove specific elements (use the **Remove** method)
- Use a hash table when the order the elements is stored is irrelevant to you
- Don't use a hash table if you need the elements in some specific order. You cannot rely on how a Hashtable will sort its elements
- Don't use a hash table if you need to insert an element in a particular location
- Don't use a hash table if you want to store multiple keys with the same value. All keys inside a hash table must be unique

The next step is to look at declaring each collection, and populating each with the same set of information. Remember that the Hashtable and the SortedList both need key-value combinations, but in this example, you are going to use the same entries for both the key and the value. That is, you will make collections of last names, and use the same last name for the key, and the value of a particular pair (for example, one pair will be "Smith"-"Smith").

```
// don't forget to include those 'Imports' statements
Imports System
Imports System.Collections

'  ... class and method declarations...

'  the declaration for an array of information
Dim names() As String = New String() {"Simpson","Bristow","Walls", _
                "Gilligan","Davidson","Laurence", _
                "Jackson","Singh","O'Connor","Moynagh"}

'  the declarations for each of the three collections
Dim q As Queue q = new Queue()
Dim sl As SortedList = New SortedList()
Dim ht As Hashtable = New Hashtable()

Dim s As String

'  we use a foreach statement to populate our collections
For Each s In names

        '  to add an element to the queue, use Enqueue
        '  A queue only receives the value you want to add
        q.Enqueue(s)

        '  Use Add of SortedList/Hashtable to add elements
        '  Note that you must add key-value pairs
        sl.Add(s, s)
        ht.Add(s, s)
Next s
```

VB

You can also remove items from collections, although different methods must be used depending on the collection in question. Many collections support the **Remove** method to remove a specified element. Both Hashtable and SortedList support this method, to remove the element you specify. **RemoveAt** is also supported by many collections; however, out of the three collections mentioned in this discussion, only SortedList supports the **RemoveAt** method. **RemoveAt** requires you to supply an index of the element to remove, which doesn't really make sense in the case of a Queue or Hashtable (which is why they don't support it). A Queue is a special case since removing elements in a Queue can be performed only on the top element off the Queue. Use the **Dequeue** method to achieve this, as in the following sample.

```
' to remove an element with a specific key, use the Remove method
' where the concept of a key is invalid, specify the object to remove
sl.Remove(keyValue); '  removes the arbitrary keyvalue
ht.Remove(keyValue); '  removes the arbitrary keyvalue
```

```
' to remove an element at a specific index, use the RemoveAt method
' This does not make sense for collections such as a Hashtable/Queue
sl.RemoveAt(indexValue); ' removes the specified index

' A Queue does not have the concept of an arbitrary remove
' Use Dequeue to read, AND remove the top element off the Queue
q.Dequeue(); ' remove the top value
```

VB

After you create the collection, you can look at the information inside those collections by using a simple iteration statement to print the values out. It is difficult to show here what the results will be, so it requires you to test the output. Note that the Queue will not change the order of the items, the SortedList will sort them alphabetically, and the Hashtable will change the order, but not necessarily to anything that you or I could understand: it uses its own sorting mechanism.

```
Dim s As String

Console.WriteLine("The Entries In The Queue")
For Each s In q
        Console.WriteLine(s)
Next s

'  note that we have to parse the Values of the SortedList, rather than
'  the actual SortedList itself
Console.WriteLine("The Entries In The SortedList")
For Each s In sl.Values
        Console.WriteLine(s)
Next s

'  note that we have to parse the Values of the Hashtable, rather than
'  the actual Hashtable itself
Console.WriteLine("The Entries In The SortedList")
For Each s In ht.Values
        Console.WriteLine(s)
Next s
```

VB

So far, you have looked at the code that demonstrates how each of the collections is organized, and how to retrieve their values. However, you also need to consider how to search for values inside each of the different collections. The following code example shows you the commands needed to find a value inside each collection. The methods to search by key and by value are displayed for both SortedList and Hashtable. Recall that hash tables are very good for searching, and sorted lists are fairly good, especially when searching by key. Queues also provide a searching capability, but this is not really a Queue's purpose, and the search mechanism is slower.

```
'  use these lines of code to retrieve the time before the search...
Dim startTime as Int64
startTime = Environment.TickCount

'  use these lines of code to retrieve the time, after the search.
'  You can subtract this value from the above, to get a duration.
Dim endTime as Int64
endTime = Environment.TickCount

'  use Contains to search a Queue for an entry,
'  or a Hashtable/SortedList for a KEY
'  Contains returns a boolean, indicating if value/key was found
Dim foundValue As Boolean
Dim valToFind as Int64
foundValue = q.Contains(valToFind)
```

```
foundValue = sl.Contains(valToFind)
foundValue = ht.Contains(valToFind)

'  use ContainsValue to search a SortedList/Hashtable for a VALUE
foundValue = sl.ContainsValue(valToFind)
foundValue = ht.ContainsValue(valToFind)
```

VB

Summary

Within the **Collections** namespace, there are many collections that providing different functionality. It is a good idea to familiarize yourself with the different collections available, so that you know which collection to use in each situation you encounter. For information on more specialized collections, see the **Collections.Specialized** namespace.

Collections
Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I....Implement a collection?

This example demonstrates how to create a class that is derived from the **ICollection** interface. There are a number of interfaces in the **collection** namespace, each with their own set of properties and methods. Knowing the basics of how to implement these collections will allow you to build strongly typed collections that return specific data types, rather than generic objects, as in this example.

VB ICollection.aspx

[Run Sample] | [View Source]

The first step in implementing **ICollection** is to do just that: Implement the interface in the declaration for the class. At that point, you have made a commitment to support the interface, which means you have to provide implementations for all the methods or properties the interface expects you to use. You don't necessarily have to do anything in all the methods, but note that users who are aware you have implemented a particular interface will certainly expect to see the functionality, so at the least, let them know if you don't actually do anything in certain methods. In this sample code, you don't do anything in **CopyTo**, but you do let the user know, by throwing an exception. The rest of the methods displayed here are methods you must implement to support **ICollection**.

```
Imports System.Collections

Class EmployeeCollection : Implements ICollection

        '  The methods we have to support...
        Public Sub CopyTo(arr As Array, index as Integer) Implements ICollection.CopyTo
                '  you would have to make it VERY clear in documentation that users cannot
                '  use this functionality. Ultimately, you need to support all implemented methods
                Throw New Exception("This method is not supported in this implementation")
        End Sub

        Public ReadOnly Property Count As Integer Implements ICollection.Count
                Get
                        return _size
                End Get
        End Property

        '  these two properties are required to support synchronization
        '  only the signatures are included here...
        '  Note that they are read-only, so you only have to define a Get procedure
        public ReadOnly Property SyncRoot As Object Implements ICollection.SyncRoot
        public ReadOnly Property IsSynchronized As Boolean Implements ICollection.IsSynchronized

        '  this method is what allows users to parse the collection using foreach
        Public Function GetEnumerator() As IEnumerator Implements ICollection.GetEnumerator
                return New EmployeeEnumerator(Me)
        End Function
End Class
```

VB

Now that you have the basis for the new collection, you have to determine how to design the internal mechanisms of the collection itself. A good idea is to use an array for the collection members that you expose through methods. In this example, you are going to use an **Employee** array, as well as variables to keep track of the current number of elements the user has put in the array (size, or count), and the actual size of the array (known as capacity). You should also decide what constructors you want to support, if any. This scenario supports the zero-argument constructor, which sets the initial capacity of the collection to 16, although it doesn't have any elements yet.

```
'  the internal deployment of the collection.
'  Note that you have strongly typed it here to an Employee
Private _array() As Employee

Private _size As Integer
Private _capacity As Integer '  Last valid element in the collection

'  the only constructor supported in this example.
'  Often you would provide many constructors
Public Sub New()
        MyBase.New()

        Redim _array(16)
        _capacity = 0
        _size = 0
End Sub
```

VB

Now that you know the variable used to store elements, the next step is to make the class that the first sample referenced in the **GetEnumerator** method. This class is going to implement **IEnumerator**, which has to support two methods and a property: **Reset**, **Current**, and **MoveNext**. There are fairly standard steps for what these methods should do, and in this sample, you are only going to look at **MoveNext** (the signatures for **Reset** and **Current** will be included, but not the code). The **MoveNext** method is invoked each time a user loops around a foreach (For Each) statement. In the **MoveNext** method, you keep a pointer to the current element of the collection, and move the pointer to the next element when called. **Current** will return the element when it is requested.

Note that in this example, you keep an internal representation of the collection, _ec, as well as the index of the current element. Also note that all these methods are invoked automatically: the user does need to know about them, they only need to know how to use foreach.

```
' note that you implement IEnumerator
Class EmployeeEnumerator : Implements IEnumerator

        Private _ec As EmployeeCollection '  a copy of our collection
        Private _index As Integer         '  track the index of the current element
        Private currentElement As Object  '  the current element, returned by Current

        '  our implementation of MoveNext
        Public Function MoveNext() As Boolean Implements IEnumerator.MoveNext
                If _index < 0 Then '  check to see if you can take this action. False indicates
                                   '  that the MoveNext did not work
                        currentElement = _ec._array

                        return false
                End If

                currentElement = _ec.GetElement(_index)  '  moved so reset current element
                _index = _index + 1                      '  increase our index appropriately

                ' once at the end of our collection, ensure the user cannot MoveNext
                If _index = _ec._size
                        _index = -1
                End If

                return true
        End Function

        '  these are the signatures for the other methods you need to use. See the
        '  major sample for the code we placed in these methods
        Public Sub Reset() Implements IEnumerator.Reset ' simply resets the enumerator
                                '
        '  Do exception checking, and if all is OK, return the current element
        Public ReadOnly Property Current() As Object Implements IEnumerator.Current
End Class
```

VB

The next step is to determine what functionality you want the collection to provide. This is exposed through the methods you want to add to your collection. Be sure to consider the other interfaces you could implement (such as **IList**), because they might provide you with functionality that more closely matches your needs. **ICollection** is selected here purely for demonstration purposes.

The following sample code demonstrates the **GetEmployee** method and the **Clear** method. See the major example for the other methods included (**AddEmployee** and **SetName**).

```
'  routine to return a employee, specified by id
Public Function GetEmployee(id As Integer) As Employee
        If _size = 0 Then
                Throw New InvalidOperationException( _
                                "You cannot get an employee when the collection is empty.")
        End If

        Dim e As Employee

        Fpr Each e In _array            '  loop through the array, searching for the employee
                If id = e.ID Then

                        return e  '  return the element found!
                End If
        Next e

        return Nothing '  if the employee wasn't found, return null. Note that your user
                '  documentation would explain that this is what you do in this scenario
End Function

public Sub Clear()    ' clear the array
        If _capacity > 0 Then
                Array.Clear(_array, 0, _size)  ' simply clear the array
```

```
            Else
                    Array.Clear(_array, 0, _array.Length)
                    Array.Clear(_array, 0, _capacity)
            End If

            _capacity = 0  ' reset the capacity, and size of the array
            _size = 0
    End Sub
```

VB

How do users now use the collection? Because enumerator support (through **GetEnumerator**) is included, using a foreach (For Each) statement is supported. Users need to know what the collection can and can't do, but otherwise, they need to call the methods as expected. The following code example shows how a user would use the collection.

```
    '  declare our collection, using the default constructor provided
    '  its a good idea to include many overloaded constructors, for all the items that can be set
    Dim ec As EmployeeCollection = New EmployeeCollection()

    Console.WriteLine("Adding elements to the EmployeeCollection")
    ec.AddEmployee( 1234, "John" )
    ec.AddEmployee( 5678, "Lisa" )
    ec.AddEmployee( 3283, "Max" )
    ec.AddEmployee( 1928, "Regina" )
    ec.AddEmployee( 4019, "Clarence" )

    Dim e As Employee

    For Each e In ec          '  iterate over the collection using foreach
            Console.WriteLine(ec.GetEmployee(e.ID))

    Console.WriteLine("Retrieving the employee with id 4019...")
    Console.WriteLine(ec.GetEmployee(4019).ToString())

    Console.WriteLine("Clearing the collection...")
    ec.Clear()

    Console.WriteLine("Setting the name of existing employee with id 5678...")
    ec.SetName(5678, "Katherine")
```

VB

Collections
Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I....Clone a collection?

This example demonstrates using cloning on an **ArrayList**. Cloning performs a 'shallow' copy of the **ArrayList**, meaning that any object references in the original list are copied by the target list rather than taking actual copies of the objects themselves. If the cloned list is modified, so is the original, when the list contains object references. For more information, see How Do I...Compare object references?.

VB Clone.aspx

[Run Sample] | [View Source]

An **ArrayList** is a generic collection of objects. When you make an ArrayList, you can place value or reference type objects in the collection. The examples included here are Employees (consisting of a Name, and an ID) which are reference types, and Strings which are immutable. The first step is to look at the code necessary to construct an **ArrayList** and put elements into it.

```
' don't forget your Imports statements, at the top of your code...
Imports System
Imports System.Collections

' ... class and method declarations...

Dim al As ArrayList = New ArrayList()

ArrayList.Add( new Employee(7182, "Laurie Davies") )  ' add an employee object to the list
ArrayList.Add( "Test String 1" )                      ' add a string to the list
```

VB

Now that the list is populated, you will take a 'clone' of it. A clone is often referred to as a 'shallow copy' of an original. This means that any nonreferential data types (such as strings or integers) are actually copied from one list to the other. In contrast, if the original list contains object references, then the references are copied, so that the cloned list contains the same references to the objects in the first list. This is a far faster way of copying one list to another than actually copying all the elements yourself. The following sample shows you how to clone a list. When you modify the object references in the copy, the original will also change, but when you modify the nonreferential items, the original remains unchanged.

```
' this line creates a new list, cloned from the list you made in the previous example
Dim nl As ArrayList = CType(al.Clone(), ArrayList)

' set the Name property of the Employee object in our copied list, to another value
CType(nl(0), Employee).Name = "Mary Smith"

' set the String element in our copied list to another value
newList(1) = "Modified String 1"

Dim o As Object

' print the values of the ORIGINAL list. The employee NAME has changed,
' while the String has remained the same
For Each o In newList
        Console.WriteLine(o.ToString())
Next o
```

VB

Now that you have identified what happens when an **ArrayList** is cloned (note that other collections also support Clone, as does the **Array** object), you may wonder how to make an actual copy of the entire collection. Most copies that you can normally do are all similar to the Clone process, since this is the fastest process to copy a collection, and supports most instances you are going to want to program. However, you may also come across situations where you want to do a 'deep copy' of a list. The following code example demonstrates how to make the new list from the old list.

```
' declare your new list
Dim nl As ArrayList = New ArrayList()

Dim i As Integer

' loop through the entries in the original ArrayList
For i = 0 to al.Count - 1
        If (al(i)).GetType() = GetTYpe(Employee) Then
```

```
                ' add the copied object reference to the new list, as a New Object
                nl.Add(New Employee(CType(al(i), Employee).ID, CType(al(i), Employee).Name))

        Else
                ' non-referential entries can be added as they are...
                nl.Add(al(i))
        End If
Next i

' use the same code as in the previous sample to modify elements in your new list,
' and see if the original list is affected. In this scenario, the original will not be changed
' by modifying EITHER types of elements
```

VB

The concept of shallow and deep copying is useful to be aware of, and is ultimately a reflection of object reference equality. The default cloning is to simply copy object references, rather than remake all the objects in a collection, a potentially lengthy process. For more information on cloning, see **ICloneable** interface.

The following example demonstrates iterating over an ArrayList. Elements are added to, and removed from the list through the use of the methods and properties in the ArrayList. To iterate over the list, use a For Each statement, which can identify each object in the list in turn.

```
The initial Count of our list is 0 and it's Capacity is 16

After populating our list, it's Count is 8 and it's Capacity is 16

datatypes of elements in the list:
---------------------------------
System.String
System.Int32
System.String
System.String
System.String
_ASP.list_aspx+MyType
System.String
System.String
---------------------------------

Removing the integer object...
Removing the MyType object...
Inserting new object...
After modifying our list, it's Count is 7 and it's Capacity is 16

The entire list now contains the values:
---------------------------------------
Grapes
Apple
Orange
Banana
Pear
Strawberry
Cherry
---------------------------------------

After removing all elements from our list, it's Count is 0 and it's Capacity is 16
```

Component Model
Use the CodeDOM

Back to Index

Get URL for this page

How Do I...Use the CodeDOM to emit source code?

What is the CodeDOM

The CodeDOM is an API that gives you the ability to create a programming structure of namespaces, objects, and programming constructs by adding items to different collections. All CodeDOM programs start out with the creation of a **ICodeGenerator** for the target language. The following code shows the creation of the **ICodeGenerator** for C#.

```
Dim cdp As CSharpCodeProvider = New CSharpCodeProvider()
Dim cg As ICodeGenerator = cdp.CreateGenerator()
```

VB

The **ICodeGenerator** is used to generate the actual code once a CodeDOM tree is built. To begin building your tree, you need to start with a **CodeNamespace** object. This object will contain any types and import directives. With the **CodeNamespace** object you can begin the tree. For the following sample, several **CodeNamespaceImport** objects are used to generate import directives for commonly used namespaces. The sample also needs to generate a class construct so by using a **CodeTypeDeclaration** and setting the **IsClass** property to true. The following code demonstrates these concepts.

```
Dim cnamespace As CodeNamespace = New CodeNamespace("Microsoft.Com2Sdk.TypedCollectio
 ...
cnamespace.Imports.Add (New CodeNamespaceImport ("System") )
 ...
Dim co As CodeTypeDeclaration = New CodeTypeDeclaration (typeName +"List")
co.IsClass = true;
cnamespace.Types.Add (co)
```

VB

At the end of the previous example, the class type definition is added to the **Types** collection of the **CodeNamespace** object. This is how a CodeDOM tree is built. Starting with a **CodeNamespace** object and working down through classes to class members. Class members in turn contain different types of code statements and expressions. Once the entire tree is built, the entire source tree can be written with a call to **GenerateCodeFromNamespace**, a method on **ICodeGenerator**. The method is being passed the **CodeNamespace** object and a **TextWriter**.

```
baseCompiler.GenerateCodeFromNamespace (cnamespace, w, Nothing)
```

VB

The following sample demonstrates the techniques described above by generating the code for a strongly typed List. It also contains additional information and code for generating code statements and expressions that were left out of the above documentation for brevity.

Example

VB ListBuilder.exe

[Run Sample] | [View Source]

Regular Expressions
Write Client Configuration
Retrieve Configuration
Custom Config Handler

Back to Index

Get URL for this page

How Do I... Write client configuration files?

In addition to supporting access to configuration settings from within an ASP.NET Web Application Server, the ASP.NET configuration system also enables client-side exe applications to obtain access to configuration data at runtime. The API you do this through - **System.Web.Configuration.ConfigurationSettings** - is the same as that for Web scenarios. There are, however, four differences:

1. The root configuration file for client side exe configuration access -- Machine.config -- is the same as that for Web scenarios. However, the application-specific configuration data is not stored in files named Web.Config, but rather in XML text files whose name matches the host exe that launched the client process (note that the file format is identical to Web.Config).

   For example, a client exe "MySample.exe" would store its application-specific configuration data within the MySample.exe.config file stored in the same directory as the launching exe. Note that because the configuration file name matches the name of the client exe, it is possible to store multiple exe applications (along with their appropriate configuration data) within the same file system directory.

2. Client-side exes can only store configuration data in one of two files: Machine.config and the application-specific configuration data file (for example, MySample.exe.config). The ASP.NET configuration system will not examine sub-directories beneath the .exe for configuration data. Nor will it look at any parent directory configuration.

   For example, here is the configuration resolution path mentioned for the above MySample.exe:

```
C:\Program Files\Microsoft .NET Framework 2.0\Machine.config        <= Base Configuration Settings for Machine
D:\MyApplication\MySample.exe.config                 <= Overriden by Application Config Settings
```

3. The location tag directive is not enabled or used for client exe scenarios. This is only available for use in Web application scenarios.

4. Changes to the originating configuration file are not automatically picked up by the ASP.NET configuration system. Instead, a client exe must stop and re-launch the exe application in order for changes to the configuration files to take effect.

For more information on the configuration file format, see the ASP.NET Configuration File Format pages.

For information on retrieving configuration information, refer to the ASP.NET section on Retrieving Configuration Information

---

Regular Expressions
Write Client Configuration
Retrieve Configuration
Custom Config Handler

Back to Index

Get URL for this page

How Do I...Write a Custom Configuration Section Handler?

When configurations become too complex for the simple section handlers provided by the **System.Configuration** namespace, a custom section handler should be created. All that is needed for a custom section handler is for a class to support the **IConfigurationSectionHandler** interface. A basic class definition for the handler might look like the following.

```
Public Class MyConfigurationSectionHandler : Implements IConfigurationSectionHandler
 ..
End Class
```

VB

The **IConfigurationSectionHandler** interface has only one method. The **Create** method is called on the section handler whenever a configuration section is found that is registered to the handler. The **Create** method accepts three parameters. The first parameter is an **Object** that contains parent configuration settings in the instance that inherited configuration settings exist. The second parameter is a **ConfigurationContext** that contains any contextual information such as the current virtual path. This information is provided for convenience, and doesn't have to be used. The final parameter is an **XmlNode** containing the raw XML configuration. The **DictionarySectionHandler** returns a hash table, but a handler can return any possible type after parsing data. A sample implementation of **Create** might consist of the following code sample.

```
Public Function Create(Parent As Object, ConfigContext As ConfigurationContext, Section As XmlNode) As Object
    Dim Hash As New Hashtable()
    ...
    Return Hash
End Class
```

VB

When trying to manipulate and investigate the parent properties, keep the following guidelines in mind: the **parent** object should never be modified in the **Create** method; any modifications, no matter how small, should be made on a clone; the **parent** object may also be null; if the section handler is being called for the first time, there won't be any inheritable settings available. The following code example demonstrates proper use of the **parent** object.

```
Dim Hash As Hashtable

If Parent = Nothing
    Hash = New Hashtable()
Else
    Hash = CType(CType(parent, Hashtable).Clone(), Hashtable)
End If
```

VB

Cryptography
Encrypt and decrypt a file
Compare a hash value

Back to Index

Get URL for this page

How Do I...Encrypt and Decrypt a file?

The **CryptoStream** class in the **System.Security.Cryptography** namespace is used to easily define cryptographic transforms on any data stream. The constructor is defined as the following: **CryptoStream** (**Stream** argument, **ICryptoTransform** transform, **CryptoStreamMode** mode).

Stream argument - Defines the stream on which the cryptographic transform is to be performed. Any stream that derives from **System.IO.Stream** can be plugged in here. For example, pass in an instance of **System.IO.FileStream** to perform a cryptographic transform on a file. Because **CryptoStream** derives from **Stream**, it is possible to use **CryptoStream** to define cryptographic transforms on other cryptographic streams. This makes it possible to chain objects that implement **CryptoStream** together, for example encrypting a file and computing the hash for the encryption in a single operation.

ICrypto Transformtransform - Defines the cryptographic transform that is to be performed on the stream. Because every class that derives from **HashAlgorithm** implements the **ICryptoTransform** interface, an instance of a hash algorithm can be passed in here to take the hash of a stream. All symmetric encryption or decryption algorithms that derive from the **SymmetricAlgorithm** class have **CreateEncryptor()** and **CreateDecryptor()** functions that return an instance of an **ICryptoTransform** implementation. To define a **TripleDES** encryption on a given stream, call the **CreateEncryptor()** function on an instance of a **TripleDES** implementation and pass the result into the **CryptoStream** constructor. Generally, any class that implements **ICryptoTransform** can be passed into the **CryptoStream** constructor.

CryptoStreamMode mode - Defines whether you are reading from or writing to the stream. To write to a **CryptoStream** you must pass **CryptoStreamMode.Write** into the **CryptoStream** constructor. To read from the stream, **CryptoStreamMode.Read** must be passed into the constructor.

The **CryptoStream** class contains the standard stream member functions to either read a byte array from the stream or write a byte array to the stream. The **CryptoStream** class handles the buffering internally when reading from or writing to the stream. Application code needs only to provide the byte buffer and call the appropriate read or write method on the stream.

The following sample code shows the creation of a **CryptoStream** to encrypt a file using the **DES** algorithm. First, the **FileStream** that will contain the encrypted file is created. Then, an instance of a **DES** implementation is created. If an instance of a symmetric or asymmetric algorithm is created without explicit constructor arguments, a random key (or public or private key pair) is generated and default properties are set that cover most encryption or decryption scenarios. A **DES** encryptor object is created on the **DES** instance. Next, a **CryptoStream** is created by passing the **FileStream** instance and the **DES** encryptor into the **CryptoStream** constructor; the stream is set to write mode. Finally, we write a byte array of plain text to the stream and close the stream. The result is a file named "EncryptedFile.txt" which contains the **DES** encryption of **bytearrayinput**.

```
Dim fs As New FileStream("EncryptedFile.txt", FileMode.Create, FileAccess.Write)
Dim des As New DESCryptoServiceProvider()
Dim desencrypt As ICryptoTransform = des.CreateEncryptor()
Dim cryptostream As New CryptoStream(fs, desencrypt, CryptoStreamMode.Write)
cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length)
cryptostream.Close()
```

VB

Example

VB fileencrypt.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Cryptography
Encrypt and decrypt a file
Compare a hash value

Back to Index

Get URL for this page

How Do I...Generate and compare a hash value?

It is easy to generate and compare hash values using the cryptographic resources contained in the **System.Security.Cryptography** namespace. Because all hash functions take input of type **Byte[]**, it might be necessary to convert the source into a byte array before it is hashed.

To generate a hash value, create an instance of a hash algorithm and call **ComputeHash()** on it. The following code creates an instance of the **MD5** hash algorithm and calls **ComputeHash()** on it. The hash function returns a byte array containing the hash value of the byte array **Input**

```
Dim HashVal As Byte() = New MD5CryptoServiceProvider().ComputeHash(Input)
```

VB

Note that **ComputeHash()** is the final operation performed on an instance of a hash object. If another hash value needs to be generated, a new instance of a hash algorithm must be created.

System.Security.Cryptography contains implementations of **MD5**, **SHA1**, **SHA256**, **SHA384**, and **SHA512**. The following code computes a **SHA1** hash value.

```
Dim HashVal As Byte() = New SHA1CryptoServiceProvider().ComputeHash(Input)
```

VB

To compare hash values, perform an element-by-element comparison of the hash value byte arrays.

Example

VB hash.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLEDB/SQL .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections**. For connection to and managing transactions against a database.
- **Commands**. For issuing SQL commands against a database.
- **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
- **DataSets**. For storing, remoting and programming against flat data, XML data and relational data.
- **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.

Note When dealing with connections to a database, there are two different options: ADO.NET SQL Data Provider (System.Data.SqlClient) and ADO.NET OLE DB Data Provider (System.Data.OleDb). In these samples we will use the ADO.NET SQL Data Provider. These are written to talk directly to Microsoft SQL Server. The ADO.NET OLE DB Data Provider is used to talk to any OLE DB provider (as it uses OLEDB underneath).

Connections

Connections are used to 'talk to' databases, and are respresented by provider-specific classes such as **SQLConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

The example below shows how to create a connection object. Connections can be opened explicitly by calling the **Open** method on the connection, or will be opened implicitly when using a **DataAdapter**.

VB AdoOverview1.aspx

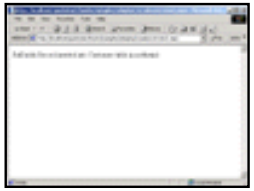
[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Commands

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SQLCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

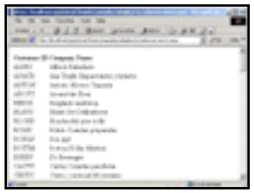


VB AdoOverview2.aspx

[Run Sample] | [View Source]

DataReaders

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset.For example, you might use the **DataReader** to show the results of a search list in a web page.



VB AdoOverview3.aspx

[Run Sample] | [View Source]

DataSets and DataAdapters

DataSets
The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

DataAdapters (OLEDB/SQL)
The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

```
' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (e.g. Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
Dim myConnection As SqlConnection = New
SqlConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
```

```
Dim mySqlDataAdapter As SqlDataAdapter = New SqlDataAdapter("select * from customers", myConnection)

mySqlDataAdapter.InsertCommand.CommandText = "sp_InsertCustomer"
mySqlDataAdapter.InsertCommand.CommandType = CommandType.StoredProcedure
mySqlDataAdapter.DeleteCommand.CommandText = "sp_DeleteCustomer"
mySqlDataAdapter.DeleteCommand.CommandType = CommandType.StoredProcedure
mySqlDataAdapter.UpdateCommand.CommandText = "sp_UpdateCustomers"
mySqlDataAdapter.UpdateCommand.CommandType = CommandType.StoredProcedure
```

VB

```
mySqlDataAdapter.Update(myDataSet)
```

VB

The records are appropriately mapped to the given commands accordingly.

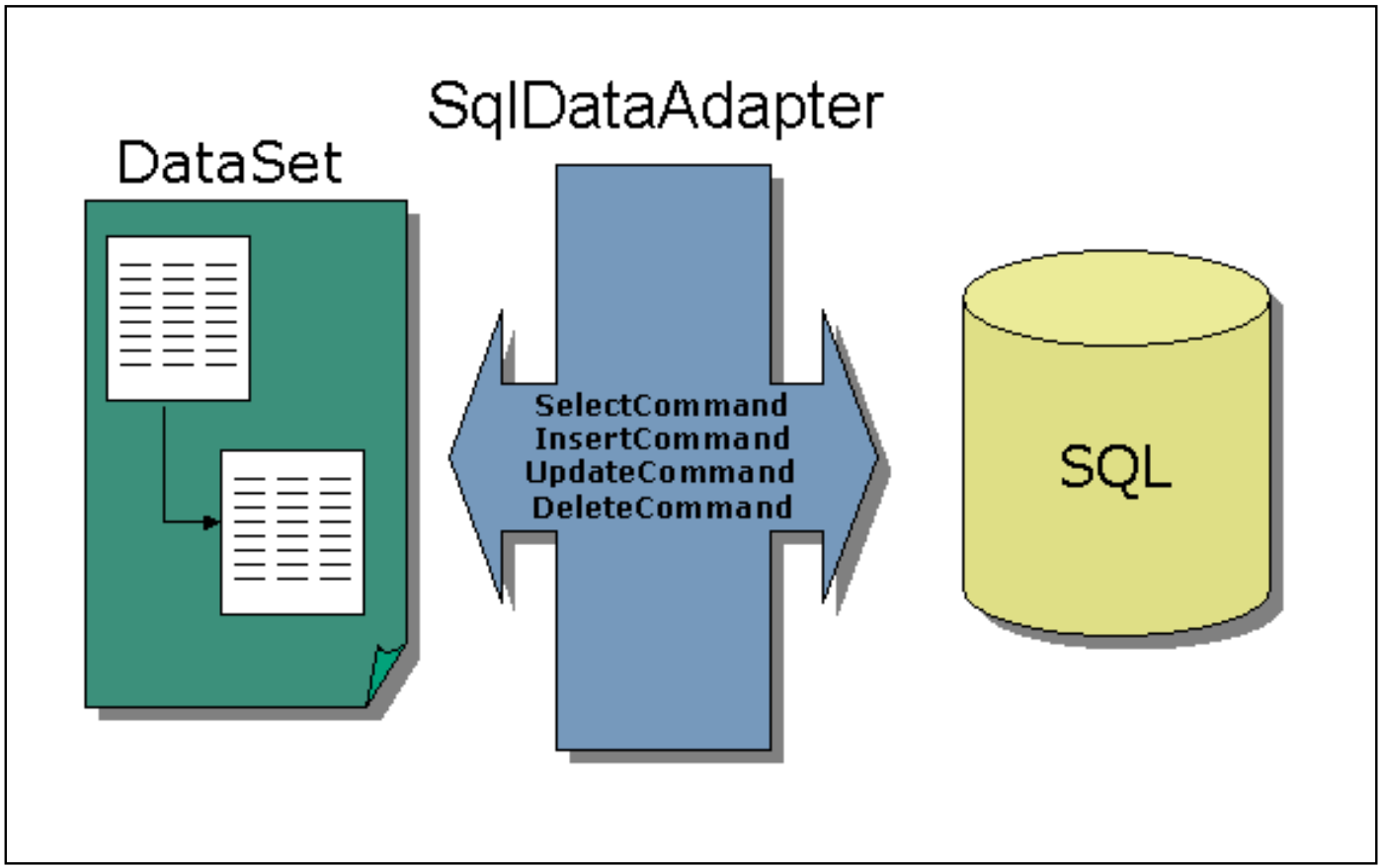
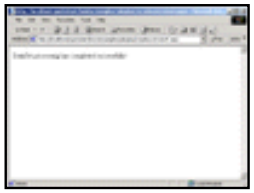


Figure: DataAdapters and DataSets

The sample below illustrates loading a **DataAdapter** via a SELECT statement. Then it updates, deletes and adds some records within the **DataSet**. Finally, it returns those updates to the source database via the **DataAdapter**. The constructed DeleteCommand, InsertCommand and UpdateCommand are shown in the page. It also illustrates using multiple **DataAdapter** objects to load multiple tables (Customers and Orders) into the **DataSet**.



VB AdoOverview4.aspx

[Run Sample] | [View Source]

Section Summary

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships.

Copyright 2001 Microsoft Corporation. All rights reserved.

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLEDB/SQL .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections**. For connection to and managing transactions against a database.
- **Commands**. For issuing SQL commands against a database.
- **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
- **DataSets**. For storing, remoting and programming against flat data, XML data and relational data.
- **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.

Note When dealing with connections to a database, there are two different options: ADO.NET SQL Data Provider (System.Data.SqlClient) and ADO.NET OLE DB Data Provider (System.Data.OleDb). In these samples we will use the ADO.NET SQL Data Provider. These are written to talk directly to Microsoft SQL Server. The ADO.NET OLE DB Data Provider is used to talk to any OLE DB provider (as it uses OLEDB underneath).

Connections

Connections are used to 'talk to' databases, and are respresented by provider-specific classes such as **SQLConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

The example below shows how to create a connection object. Connections can be opened explicitly by calling the **Open** method on the connection, or will be opened implicitly when using a **DataAdapter**.

VB AdoOverview1.aspx

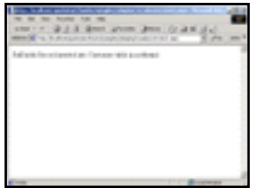
LateBreaking Samples:

- C++

[Run Sample] | [View Source]

Commands

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SQLCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

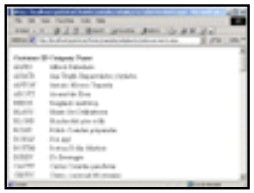


VB AdoOverview2.aspx

[Run Sample] | [View Source]

DataReaders

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset.For example, you might use the **DataReader** to show the results of a search list in a web page.



VB AdoOverview3.aspx

[Run Sample] | [View Source]

DataSets and DataAdapters

DataSets
The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

DataAdapters (OLEDB/SQL)
The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

```
' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (e.g. Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
Dim myConnection As SqlConnection = New
SqlConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
```

```
Dim mySqlDataAdapter As SqlDataAdapter = New SqlDataAdapter("select * from customers", myConnection)

mySqlDataAdapter.InsertCommand.CommandText = "sp_InsertCustomer"
mySqlDataAdapter.InsertCommand.CommandType = CommandType.StoredProcedure
mySqlDataAdapter.DeleteCommand.CommandText = "sp_DeleteCustomer"
mySqlDataAdapter.DeleteCommand.CommandType = CommandType.StoredProcedure
mySqlDataAdapter.UpdateCommand.CommandText = "sp_UpdateCustomers"
mySqlDataAdapter.UpdateCommand.CommandType = CommandType.StoredProcedure
```

VB

```
mySqlDataAdapter.Update(myDataSet)
```

VB

The records are appropriately mapped to the given commands accordingly.

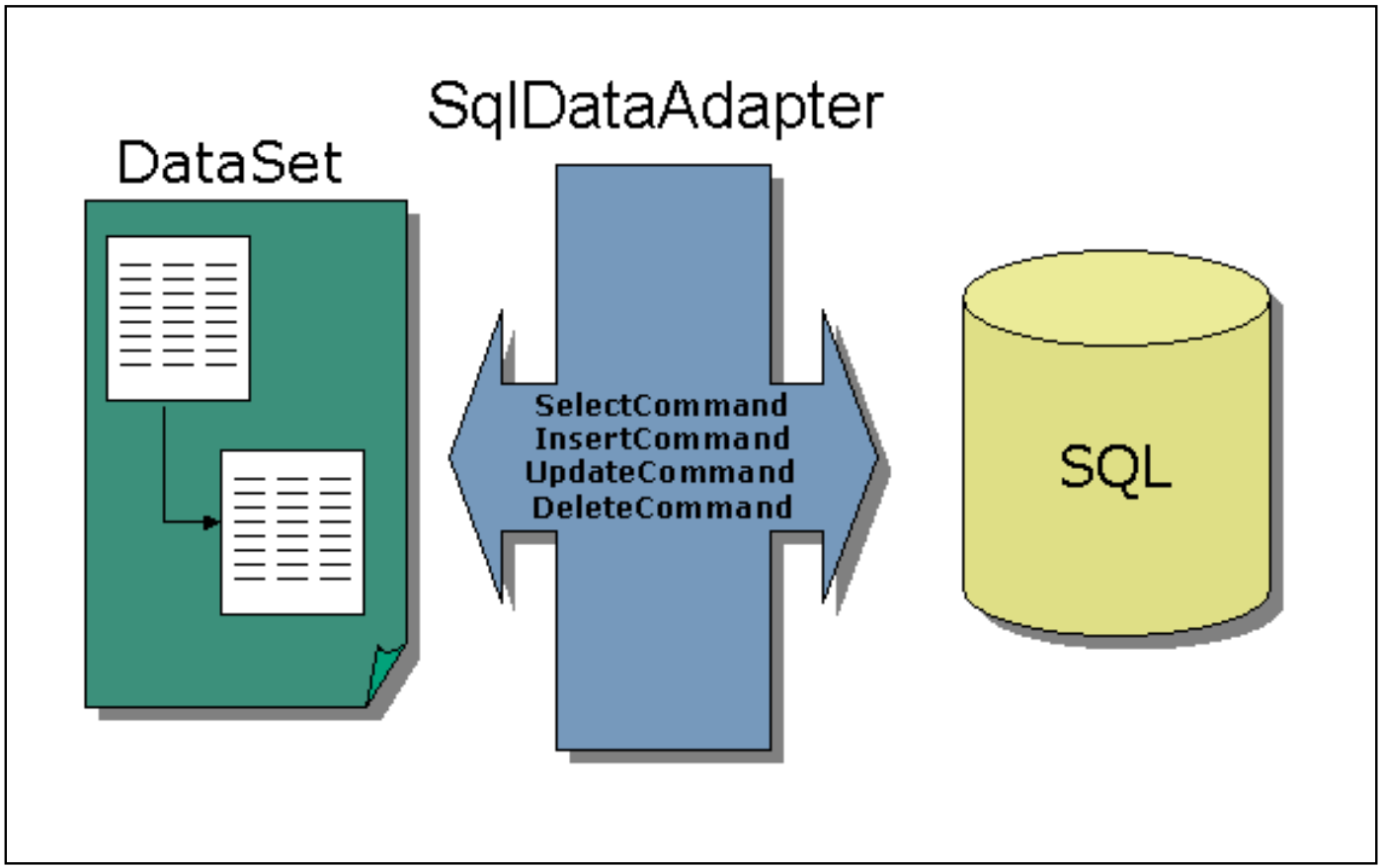
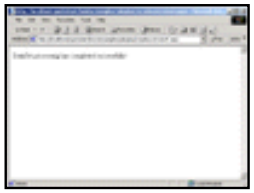


Figure: DataAdapters and DataSets

The sample below illustrates loading a **DataAdapter** via a SELECT statement. Then it updates, deletes and adds some records within the **DataSet**. Finally, it returns those updates to the source database via the **DataAdapter**. The constructed DeleteCommand, InsertCommand and UpdateCommand are shown in the page. It also illustrates using multiple **DataAdapter** objects to load multiple tables (Customers and Orders) into the **DataSet**.



VB AdoOverview4.aspx

[Run Sample] | [View Source]

Section Summary

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships.

Copyright 2001 Microsoft Corporation. All rights reserved.

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Execute a Command

Commands are issued against databases to take actions against data stores. For example, you could execute a command that inserts or deletes data. For more information on moving data into and out of databases, see "Update a Database from a DataSet".
Commands include any command that can be issued against a database, and in the case of the **OleDbCommand**, can be data store specific. For example, you could issue a stored procedure call for a command, or perhaps a command to "set quoted_identifier on". Whatever the command may be, the **OleDbCommand** or **SqlCommand** can be used to get the command to your back-end data store.

With ADO classic, you could issue commands through the **Command** object, **Connection** object, or **Recordset** object. In ADO.NET, only the **Command** object executes commands.

To issue a command against a database, the **Command** object must have two basic things: a **Connection** and **CommandText**, both of which can be set in the constructor. To execute the command, the **Connection** has to be open and not in fetching state:

```
Dim InsertCmdString As String
InsertCmdString = "Insert into Region (RegionID, RegionDescription) VALUES (100, 'Description')"
Dim mySqlCommand As SqlCommand = New SqlCommand(InsertCmdString, myConnection)
```

VB

This topic covers executing non-result generating commands. To execute the command that does not return results, call the **ExecuteNonQuery** method.

```
mySqlCommand.ExecuteNonQuery()
```

VB

The **OleDbCommand** and **SqlCommand** objects have parameter collections that behave like the parameter collections from ADO classic. You can pass your parameters inline:

```
mySqlCommand.CommandText = "myStoredProc 'CustId'"
```

VB

Or by using the **Parameters** collection:

```
workParam = mySqlCommand.Parameters.Add("@CustomerID", SQLDataType.NChar, 5)
workParam.Value = "NewID"
```

VB

For more information about using parameters, see Getting Out Parameters from a Stored Procedure.

The following sample shows how to execute an insert command against a SQL database using the **SqlCommand**.

VB executingacommand.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Get Out Parameters from a Stored Procedure

Some stored procedures return values through parameters. When a parameter in a SQL statement or stored procedure is declared as "out", the value of the parameter is returned back to the caller; the value is stored in a parameter in the Parameters collection on the **OleDbCommand** or **SqlCommand** objects.

Unlike the sample below, if the connection and command name are not set, you can still establish the parameters, but you have to create the collection of parameters and expected types.

```
workParam = myCommand.Parameters.Add("@CustomerID", SQLDataType.NChar, 5)
workParam.Value = "CUSTID"
```

VB

The following sample shows how to use a stored procedure that returns output parameters. The command is executed to create a stored procedure in the Northwind database.

VB OutParamsWithACommand.aspx

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Populate a DataSet from a Database

Getting data from a database is easy, and working with data is easier than before. If you want the results from a database as a forward only, read-only stream of data, you can execute a command and retrieve the results using the **DataReader**. For examples on how to use a **DataReader**, see Retrieve Data Using OLE DB and Retrieve Data from SQL Server. For more interactive operations such as binding to, navigating through, or remoting the results of a database query, you can place the results in a DataSet as shown in this example.

The most important concept to remember is that the DataSet is a data structure separate and distinct from a data store. Although you get data from a database in this example, it doesn't matter where the data comes from; the DataSet will always present a consistent programming model. It is a simple collection of data with relational database characteristics. There is no Load, Open, or Execute on a DataSet because it doesn't know where it gets its data from. This section describes how to use a SqlDataAdapter to get data from a database into a DataSet.

You can add data to the DataSet using the **Add** method (see Update a Database from a DataSet), or directly push data in with the intrinsic XML methods (see Read and Write XML Data). SqlDataAdapter can be used to retrieve data from a database but can also push data back into the database. Here, the focus is on getting data out.

The first step is to construct a SqlDataAdapter. This is done in the same way a SqlCommand is constructed.

```
Dim SelectCmdString As String = "select * from customers"
Dim mySqlDataAdapter As SqlDataAdapter = New SqlDataAdapter(SelectCmdString, myConnection)
'...or this can also be done as follows:

Dim mySelectCommand as SqlCommand = New SqlCommand("select * from customers", myConnection)
Dim mySqlDataAdapter as SqlDataAdapter = new SqlDataAdapter(mySelectCommand)
```

VB

The SqlDataAdapter and SqlCommand are very similar, except for the **Fill** and **Update** methods. The **Fill** method populates a DataSet. The **Update** method takes changes from a DataSet and pushes them back into the database. This is accomplished by four commands specified on the DataAdapter. These commands are: SelectCommand, UpdateCommand, InsertCommand, and DeleteCommand. You can explicitly set these commands to control the statements used at run time to resolve changes, including the use of stored procedures. For ad hoc scenarios, a **CommandBuilder** object can generate these at run time based on a select statement (see Update a Database from a DataSet). However, this run-time generation requires an extra round trip to the server to gather required metadata, so explicitly providing the insert, update, and delete commands at design time will result in better run-time performance.

NOTE: Visual Studio adds great value to establishing typed SqlDataAdapters and DataSets, and eventually creates Stored Procedures for you. Explore this feature by using the **ComponentDesigner** and **Database** objects.

Once your SqlDataAdapter is established you can pass it a DataSet to populate:

```
myDataSet = New DataSet()
mySqlDataAdapter.Fill(myDataSet,"Customers")
```

VB

The DataSet now holds the results of your query. In fact, the DataSet can hold results from multiple queries and even relate them (see Work with Relational Data). Because it holds multiple results, the DataSet contains a collection of tables. Notice that the **Fill** method has "Customers" as the second argument. This is the name of the table to fill in the DataSet. If that table does not exist, it is created for you.

Because the data is stored in a collection of rows in the table, you can easily use a **foreach** statement to iterate through the rows:

```
Dim myDataRow as DataRow
For Each myDataRow In myDataSet.Tables("Customers").Rows
  Console.WriteLine(myDataRow("CustomerId").ToString())
Next
```

VB

In fact, you can use **foreach** over the columns as well. The following example demonstrates placing together all the code in this document. To learn how to update this data back into the database see Update a Database from a DataSet.

VB GettingData.aspx

LateBreaking Samples:

- C++

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Update a Database from a DataSet

This topic illustrates how to update data in a database using a DataSet. It is important to remember that you can also insert, update, and delete data in a database directly using a **SqlCommand**. Understanding the concepts covered in Populate a DataSet from a Database will help you understand the current topic.

Some of the topics covered in Populate a DataSet from a Database include retrieving data from a database and into a DataSet, and how the DataSet is separate and distinct from the database. Once the **DataSet** is loaded, you can modify the data, and the DataSet will track the changes.

The **DataSet** can be considered an in-memory cache of data retrieved from a database. The **DataSet** consists of a collection of tables, relationships, and constraints. In this example we will show how to use the **Add** method on the DataTable to add new data to a DataSet. The **Add** method takes either an array of the expected data columns, or a DataRow.

```
' Create a new Connection and SqlDataAdapter
Dim myConnection as SqlConnection
Dim mySqlDataAdapter as SqlDataAdapter

' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (for example, Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
myConnection = new SqlConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
mySqlDataAdapter = new SqlDataAdapter("Select * from Customers", myConnection)

Dim myDataSet as DataSet = new DataSet()
Dim myDataRow As DataRow

' Create command builder. This line automatically generates the update commands for you, so you don't
' have to provide or create your own.
Dim myDataRowsCommandBuilder as SqlCommandBuilder = new SqlCommandBuilder(mySqlDataAdapter)

' Set the MissingSchemaAction property to AddWithKey because Fill will not cause primary
' key & unique key information to be retrieved unless AddWithKey is specified.
mySqlDataAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey

mySqlDataAdapter.Fill(myDataSet, "Customers")

myDataRow = myDataSet.Tables("Customers").NewRow()
myDataRow("CustomerId") = "NewID"
myDataRow("ContactName") = "New Name"
myDataRow("CompanyName") = "New Company Name"

myDataSet.Tables("Customers").Rows.Add(myDataRow)
```

VB

Note that the DataTable must return a DataRow through the **NewRow** method. The method returns a DataRow object with the appropriate schema of the DataTable. The new DataRow is independent of the table until it is added to the RowsCollection.

You can change data in a DataRow by accessing the DataRow. You can use the index of the row in the RowsCollection accessed through the **Rows** property:

```
myDataSet.Tables("Customers").Rows(0)("ContactName")="Peach"
```

VB

You can also access a specific row by the Primary Key value:

```
Dim myDataRow1 As DataRow = myDataSet.Tables("Customers").Rows.Find("ALFKI")
myDataRow1("ContactName")="Peach"
```

VB

where "ALFKI" is the value of the Primary Key "CustomerID" in the "Customers" table. When using the SqlDataAdapter, the Key is established from the database. You can also set the Key if you are not using the database through the **PrimaryKey** property.

Use the **Delete** method to remove the Row. Note that a logical deletion occurs in the DataSet, which only results in a hard deletion once the DataSet is updated to the database. Similarly you can use **RejectChanges** on the DataSet, in which case the Row is restored.

```
myDataSet.Tables("Customers").Rows(0).Delete()
```

VB

The original and new values are maintained in the row. The **RowChanging** event allows you to access both original and new values to decide whether you want the edit to proceed. Because we maintain original and new values, we can establish scenarios such as optimistic locking and key changes.

Before submitting the update back to the database, you need to setup the **InsertCommand**, **UpdateCommand**, and **DeleteCommand** to reconcile the changes to the database. For limited scenarios you can use the **SqlCommandBuilder** to automatically generate those for you, as is shown in the following example:

```
Dim myDataRowsCommandBuilder as SqlCommandBuilder = new SqlCommandBuilder(mySqlDataAdapter)
```

VB

To submit the data from the DataSet into the database, use the **Update** method on the SqlDataAdapter.

```
mySqlDataAdapter.Update(myDataSet, "Customers")
```

VB

The following example demonstrates how to get data from a database using a SqlDataAdapter, modify the data within the DataSet, and then submit the data back to the database through the SqlDataAdapter.

VB UpdatingData.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Read and Write XML

ADO.NET and DataSet can read and write XML schema and XML. For more information, see How do I...Use XML and the DataSet?

Schemas, or the tables, columns, constraints and so forth, of a DataSet can be defined in several ways. One method is to create them using properties and methods (**PrimaryKey**, **Table.Add**, **Column.Add**, and so on). This establishes a schema within your DataSet that can be used as a container to hold data. Another way is to use the SqlDataAdapter or OleDbDataAdapter. When you use these commands, if the schema does not exist in a DataSet, it is created for you.

XML is an intrinsic element of ADO.NET. Several methods that support XML have been implemented. The **GetXmlSchema** method allows you to get the schema of a DataSet using an XSD schema. For example, if you load a DataSet from the database, you could get an XSD schema from it:

```
Console.WriteLine(myDataSet.GetXmlSchema())
```

VB

This returns an XSD compliant XML schema for the schema in your DataSet.

XML representation is also retrieved through the **GetXml** method.

```
Console.WriteLine(myDataSet.GetXml())
```

VB

The sample below loads data from a database, and then outputs the XSD Schema and XML data.

VB ReadAndWriteXML.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Use Database Transactions

Database transactions are used to control data commitment to databases. For example, in standard account procedures, it is necessary to debit one account and credit another at the same time. Since computers break down on occasion (power outages, network outages, and so on) there is the potential for one record to be updated or added, but not the other. To avoid these situations, transactions are used. Transactions in ADO.NET are, just as in ADO, handled at the database level: your database must support transactions.

There are three basic commands for transactions: **BeginTransaction**, **Commit**, and **Rollback**. **BeginTransaction** marks the beginning of a transaction. Anything that happens between the **BeginTransaction** and the next command (either **Rollback** or **Commit**) is considered part of the transaction. The following code example demonstrates using transactions.

```
    ' Warning: Embedded usernames/passwords in database connection strings are inherently
    ' insecure, and should only be used on platforms (for example, Win9x) that do not support Integrated
    ' Security.  See the ASP.NET QuickStart for examples of Integrated Security.
    Dim myConnection as SQLConnection = new
SQLConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
    Dim myCommand as SQLCommand = new SqlCommand()
    Dim myTrans as SqlTransaction

    ' Open the connection.
    myConnection.Open()

    ' Assign the connection property.
    myCommand.Connection  = myConnection

    ' Begin the transaction.
    myTrans = myConnection.BeginTransaction()

    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    try
      ' Restore database to its original condition so sample will work correctly.
      myCommand.CommandText = "DELETE FROM Region WHERE (RegionID = 100) OR (RegionID = 101)"
      myCommand.ExecuteNonQuery()

      ' Insert the first record.
      myCommand.CommandText = "Insert into Region (RegionID, RegionDescription) VALUES (100, 'MidWestern')"
      myCommand.ExecuteNonQuery()

      ' Insert the second record.
      myCommand.CommandText = "Insert into Region (RegionID, RegionDescription) VALUES (101, 'MidEastern')"
      myCommand.ExecuteNonQuery()

      myTrans.Commit()
      Console.Write("Both Records are written to the database!")
    catch e as Exception
      myTrans.Rollback()
      Console.WriteLine(e.ToString())
      Console.Write("Neither record is written to the database!")
    finally
      myConnection.Close()
```

```
        end try
```

VB

The example shows that if either insert fails, both are rolled back to their original states. If each succeeds, then the transaction is committed.

As in classic ADO, you can control transactions through the **connection** object. In fact, when you use the **OleDbConnection**, it uses the same underlying OLE DB transaction model. Thus, if you were able to commit transactions against your database with ADO classic, you can still commit them with ADO.NET. The following code example demonstrates using the **SqlConnection** and **SqlCommand** to insert two records into "Region" table. If either fails, then the changes are rolled back.

The **DataSet** object also has a commit model (AcceptChanges,RejectChanges) but it does not affect the database. The commit model is for the cache of data in the DataSet alone. To submit the Data from the DataSet into the database, use the **Update** method on the **SqlDataAdapter**.

VB DBTransactionsWithACommand.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Handle Errors

In addition to Try/Catch and exceptions, the new ADO.NET data architecture allows you to add error messages to each row of Data in a DataSet. **SqlDataAdapters** attach error messages to Rows if updates or other actions fail. Furthermore, you can filter for rows in error to present them to the user, or pass them to error handling functions.

Errors persist with the DataSet even when being transferred using XML or Web Services. You can use the **RowError** property to set the Error message of a DataRow in a DataSet.

```
myDataSet.Tables("Customers").Rows(0).RowError = "An error was added"
myDataSet.Tables("Customers").Rows(1).RowError = "This is another error message"
```

VB

Now you can walk the error in a DataTable with the **GetErrors()** method. You can also test for errors using **HasErrors**.

```
if myDataSet.Tables("Customers").HasErrors then
  Dim ErrDataRows as DataRow()
  ErrDataRows = myDataSet.Tables("Customers").GetErrors()
  Console.WriteLine("DataTable " + myDataSet.Tables("Customers").TableName + " has " +
ErrDataRows.Length.ToString() + " Error(s)!")

  Dim i as integer
  for i = 0 to ErrDataRows.Length -1
    Console.WriteLine("Row Error for row " + ErrDataRows(i)("CustomerID").ToString() + " --  Error Msg="  +
ErrDataRows(i).RowError)
  next
else
  Console.WriteLine("=================")
  Console.WriteLine("DataTable " + myDataSet.Tables("Customers").TableName + " Has no errors")
end if
```

VB

The following example loads a DataSet, sets some errors, and then shows the Errors in the Rows.

VB HandleErrors.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET
Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Work with Relational Data

A DataSet can contain either related or unrelated data, and can have either two unrelated tables or two related tables. You can think of a DataSet as a document of data. In fact, an XML data document is like this, except it is based on a hierarchical paradigm. Because data is often stored in relational databases, the DataSet can handle both hierarchical relationships and key/foreign key relationships. Relationships can also have different types of enforcement. By default, deletions and updates are cascaded: if you delete a Customer row, the related Orders rows are also deleted; if you Update the key of a Customer row, the associated foreign key values in the Orders table is also updated.

A DataSet contains a Relations collection. You can add a relationship to this collection using the column or columns (in a multicolumn key) of the related tables. The following code example creates a relation between **Customers** data and **Orders** data, and names the relation CustOrders.

```
myDataSet.Relations.Add("CustOrders",myDataSet.Tables("Customers").Columns("CustomerID"), _
                        myDataSet.Tables("Orders").Columns("CustomerID"))
```

VB

The following code example adds a relation between the CustomerID Key on the Customers table and the CustomerID foreign key on the Orders table in the DataSet. Now that a relationship exists, you can iterate through the data.

```
Dim myDataRow1 as DataRow
for each myDataRow1 in myDataSet.Tables("Customers").Rows
  Console.WriteLine("Customer: " + myDataRow1("ContactName").ToString())

  ' Iterate over orders data
  Dim myDataRow2 as DataRow
  for each myDataRow2 in myDataRow1.GetChildRows(myDataSet.Relations("CustOrders"))
    Console.WriteLine("Order #" + myDataRow2("OrderID").ToString())
  next

  Console.WriteLine()
next
```

VB

The following sample iterates over the data and outputs it to a Web page hierarchically.

VB RelationalData.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Work with Typed Data

ADO classic code provides late-bound access to values within its recordset through weakly typed variables. ADO.NET code enables you to access the data held in the DataSet through a "strongly typed" metaphor. This means you can access tables and columns that are part of the DataSet with user-friendly names and strongly typed variables. Therefore, rather than writing the ADO classic code shown below:

```
AdoRecordset.Find("CustomerID = 'ALFKI'")
AdoRecordSet.Fields("FirstName").Value  = "Bob"
```

VB

with a strongly typed DataSet, you write the following ADO.NET code:

```
CustomerDataSet.Customers("ALFKI").CustomerName = "Bob"
```

VB

Additionally, the strongly typed DataSet provides access to row values as the correct, strongly typed value. With ADO, variant types are used when assigning and retrieving data. If the value that you assigned was of the wrong type, ADO would give you a run-time error. With ADO.NET, if the value is of type integer and you attempt to assign a string, you will get a compile-time error.

Given an XML schema that complies with the XSD standard, you can generate a strongly typed DataSet using the XSD.exe tool provided with the .NET Framework SDK. The syntax for generating a DataSet using this tool is:

```
xsd.exe /d /l:CS {XSDSchemaFileName.xsd}
```

The /d directive directs the tool to generate a DataSet; the /l: directive specifies the language to use -- either "C#" or "VB."

The following example uses a strongly typed DataSet called "myCustDS" to load a list of customers from the Northwind database. Once the data is loaded using the **Fill** method, the code loops through each customer in the "Customers" table using the typed **myDataRow** object. Note the direct access to the "CustomerID" field, as opposed to accessing it through the **Fields** collection.

Before running the following sample, you must run "nmake /a" in the "..\typeddata\DllSrc" directory from a console window to create the TypedData.dll file.

VB TypedData.aspx

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Filter Data

There are many ways to filter data. One way is to filter data at the database command level, using a where clause on your query. A second way is to filter the data after it is in the DataSet. This topic covers filtering in the DataSet.

Once data is in a DataSet you can use methods on the DataSet to get at subsets of that data.

Filtering with the Select method

Consider a DataSet with Customers and Orders tables. To filter on Data for Customers with the FirstName of Kelly, you can use the **Select** method, which returns an array of rows.

```
myRowArray = dsCustomers.Select("ContactName like 'Kelly%'")
```

VB

Notice that the statements are ANSI-SQL like. They are essentially a subset of ANSI-SQL, with the following difference: because the DataSet can hold multiple related tables, the filter can also filter over these related tables. In the following example, use the **Child** keyword to filter the orders and the customers.

```
myRowArray = dsCustomers.Select("ContactName like 'Kelly%' AND [child].OrderDate = '7/26/68'")
```

VB

These functions return an array of rows. You can iterate through the array using the **foreach** statement.

Filtering and Sorting with DataViews

With the **DataView** object, you can place multiple filters on a DataSet, set up databinding to them, and so forth. A **DataView** can be filtered using the same query language rules as in the **Select**, but the filter is dynamic. Therefore, if a row is added to the data, and it qualifies with the filter, it is visible in the view. Views can be sorted and filtered. Filters include data filters and version filters (Current, New, Modified, Deleted). To set up a **DataView**, construct it by handing it a DataTable within a DataSet:

```
Dim myDataView As DataView = New DataView(myDataSet.Tables("Customers"))
```

VB

You can set up a filter with the **RowFilter** property using the same query filter language as in the example for the **Select** method above. To set up a sort, use a comma-delimited list of column sorts, followed by ASC (default and optional) or DESC for ascending and descending order.

```
' Sort the view based on the FirstName column
myDataView.Sort = "CustomerID"

' Filter the dataview to only show customers with the CustomerID of ALFKI
myDataView.RowFilter = "CustomerID='ALFKI'"
```

VB

You can also filter based on the state of the rows (deleted, new, and so forth). Here is an example of a Filter on Deleted rows:

```
myDataView.RowStateFilter = DataViewRowState.Deleted
```

VB

The following sample demonstrates filling a DataSet, and then filtering it with a **DataView**. Note that a **DataView** can easily be data bound. See the databinding topics in Windows Forms or ASP.NET.

VB FilterDataTable.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET
Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Connection Pooling

This sample illustrates how to construct a pool of connections to a datasource. You will want to do this to deploy high-performance applications. In this example the pool is established in the connection string and managed automatically by the **SqlConnection**.

VB ConnectionPooling.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

In this example we specify the pooling characteristics in the connection string when the **SqlConnection** is constructed, as shown in the following code sample. Please note: Pooling is implicit, you automatically get it unless you disable it. Therefore, "true" is the default for the pooling keyword (pooling=true).

```
Dim connString as String

' Specification in the connection string:
' Please note: Pooling is implicit, you automatically get it unless you disable it.
'              Therefore, "true" is the default for the pooling keyword (pooling=true).
' Connection Reset:    False
' Connection Lifetime: 5
' Enlist:              true
' Min Pool Size:       1
' Max Pool Size:       50
' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (e.g. Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
connString = "server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind;" & _
             "connection reset=false;" & _
             "connection lifetime=5;" & _
             "enlist=true;" & _
             "min pool size=1;" & _
             "max pool size=50"

Dim myConnection1 as SqlConnection = new SqlConnection(connString)
Dim myConnection2 as SqlConnection = new SqlConnection(connString)
Dim myConnection3 as SqlConnection = new SqlConnection(connString)
```

VB

Now we have code to use several of the connections in our pool. First, open two connections and return both of the connections to the pool. Then, open three connections from the pool and return all three connections to the pool.

```
' Open two connections. One is from the pool (see min pool size), the other is created.
Console.WriteLine ("Open two connections.")
myConnection1.Open()
myConnection2.Open()

' Now there are two connections in the pool that matches the connection string.
' Return the both connections to the pool.
Console.WriteLine ("Return both of the connections to the pool.")
myConnection1.Close()
myConnection2.Close()

' Get a connection out of the pool.
Console.WriteLine ("Open a connection from the pool.")
myConnection1.Open()

' Get a second connection out of the pool.
Console.WriteLine ("Open a second connection from the pool.")
myConnection2.Open()

' Open a third connection.
Console.WriteLine ("Open a third connection.")
myConnection3.Open()

' Return the all connections to the pool.
Console.WriteLine ("Return all three connections to the pool.")
myConnection1.Close()
myConnection2.Close()
myConnection3.Close()
```

VB

The model for pooled connections is the similar to non-pooled connections. However, it is especially important to call Close when you finish with a pooled connection to release it back into the pool.

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Use Distributed Transactions

This sample illustrates how to use distributed transactions, which are available automatically from the Microsoft .NET Framework component services. Microsoft Transaction Server (MTS), Microsoft .NET Framework SDK, and the common language runtime support the same automatic, distributed transaction model. To use this feature, complete the following steps to provide for automatic distributed transactions to your data source.

1. Use **System.EnterpriseServices**.
2. Create a **ServicedComponent** to provide automatic transaction support.
3. Add some features to the **ServicedComponent**.
4. Write code to execute (distributed) transactions on the **ServicedComponent**.

One major advantage of the automatic distributed transaction programming model (Windows 2000 Component Services) is that data stores (the Resource Managers) are automatically enlisted in transactions. When the transaction group is to be completed, call **SetComplete** or if it is not to be completed call **SetAbort**. These distributed transactions can span multiple databases. ADO.NET providers (ADO Interop Provider and SQL Server Provider) automatically enlist in Component Services transactions that initiated using **ComponentServices**.

Any instance of a .NET Framework class can participate in an automatic transaction, as long as you prepare the class to do so. Each resource accessed by a class instance or object enlists in the transaction. For example, if an object uses ADO.NET to post money on an account in a database, the resource manager for the database determines if it should execute in a transaction and, if so, it enlists the database in the transaction automatically.

VB DistribTransaction.aspx

[Run Sample] | [View Source]

First, make sure you are using **System.EnterpriseServices**.

```
Imports System.EnterpriseServices
```

VB

Next, create a derived class of **ServicedComponent** for ComServices to provide automatic transaction support.

A serviced component is a .NET Framework component that derives from the **System.ServicedComponent** class and is automatically serviced by Microsoft Windows 2000 Component Services. A serviced component is typically hosted in a Microsoft .NET Framework library as an in-process server or as an out-of-process server that is called by an unmanaged client.

To prepare a class for using the transaction model, the class must be marked to participate in a transaction. Once the class is marked with a **Transaction** attribute it will automatically execute within the scope of a transaction. You can control an object's transactional behavior by setting a transaction attribute value on the page, webmethod, or class. The attribute value, in turn, determines the transactional behavior of the object that is created.

In the case of class **myServicedComponent** 'TransactionOption.Required' indicates that the object requires a transaction. It runs in the scope of an existing transaction, if one exists. If not, it will start one.

```
public class TransactionAttribute<(TransactionOption.Required)> myServicedComponent : Inherits ServicedComponent
  ...
end class
```

VB

The next step is to add some features to the derived class of the **ServicedComponent**.

You can use **System.EnterpriseServices.ContextUtil**, which exposes the **SetComplete** or **SetAbort** members, to explicitly commit or abort a transaction. **SetComplete** indicates that your object votes to commit its work; **SetAbort** indicates that your object encountered a problem and votes to abort the ongoing transaction. A transaction is neither committed nor aborted until it reaches the end of the page. Further, a single abort vote from any object participating in the transaction causes the entire transaction to fail.

```
...
private m_rConnection as SqlConnection

public sub myServicedComponent()
end sub

public function SqlConn as SqlConnection
  return m_rConnection
end function

public function State as ConnectionState
  return m_rConnection.State
end function

public sub myServicedComponentConnect(ConnectionString as string)
  m_rConnection = new SqlConnection(ConnectionString)
  m_rConnection.Open()
end sub

public sub myServicedComponentClose()
  if not (m_rConnection is nothing) then m_rConnection.Close()
end sub

public sub myServicedComponentDispose()
  m_rConnection.Dispose()
end sub

public function myServicedComponentCreateCommand() as SqlCommand
  return new SqlCommand(Nothing, m_rConnection)
end function

public sub myServicedComponentSetComplete()  ' to commit changes
  Console.WriteLine("Committing connection's transaction.")
  ContextUtil.SetComplete()
  Console.WriteLine("SetComplete")
end sub
```

```
    public sub myServicedComponentSetAbort()     ' to rollback the distributed transaction
      Console.WriteLine("Aborting connection's transaction.")
      ContextUtil.SetAbort()
      Console.WriteLine("SetAbort")
    end sub
    ...
```

VB

Finally, you will write code to execute (distributed) transactions on the **ServicedComponent**. In this code, you create an instance of the new derived class of the **ServicedComponent** and use it to connect to the database provider. Then several SQL commands are executed as distributed transactions using the **myServicedComponent**. The transactions are then finalized using the **SetComplete** method of the Microsoft .NET Framework Component Services.

```
    Dim mySqlCommand as SqlCommand
    Dim mySC as myServicedComponent = new myServicedComponent()

    ' Warning: Embedded usernames/passwords in database connection strings are inherently
    ' insecure, and should only be used on platforms (e.g. Win9x) that do not support Integrated
    ' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
    mySC.myServicedComponentConnect("server=(local)\\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
    Console.WriteLine("connection state is: " & mySC.State)
    mySqlCommand = new SqlCommand(Nothing, mySC.SqlConn)

    try
      ' Clean up the database to restore to its original state.
      mySqlCommand.CommandText = "if exists (select * from sysobjects where name = 'TestTable' and type = 'U') drop
table TestTable"
      mySqlCommand.ExecuteNonQuery()

      ' Create table "TestTable".
      Console.WriteLine("Creating table TestTable.")
      mySqlCommand.CommandText = "create table TestTable (c1 int identity, c2 int)"
      mySqlCommand.ExecuteNonQuery()

      ' Create unique index ss on TestTable.
      Console.WriteLine("Creating unique index ss on TestTable.")
      mySqlCommand.CommandText = "create unique index ss on TestTable(c1)"
      mySqlCommand.ExecuteNonQuery()

      ' Insert into TestTable values.
      Console.WriteLine("Inserting into TestTable values.")
      mySqlCommand.CommandText = "insert into TestTable values (11)"
      mySqlCommand.ExecuteNonQuery()

      ' Commit the transaction.
      mySC.myServicedComponentSetComplete()
    catch e as Exception
      Console.Write(e.ToString())

      ' Rollback the transaction.
      mySC.myServicedComponentSetAbort()
    finally
      mySC.myServicedComponentClose()
      Console.WriteLine("connection state is: " & mySC.State)
```

```
        end try
```

VB

A related issue is key creation and strong naming of keys. We can use Strong Name tool "sn.exe" to create a key file. For example, the following command creates a new, random key pair and stores it in keyPair.snk.

```
    sn -k keyPair.snk
```

Here is how to use the key pair in the registration of the assembly.

```
        <assembly:System.Reflection.AssemblyKeyFile("keyPair.snk")>
```

VB

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Read XML from SQL Server

This sample illustrates how to read data in the form of XML from SQL Server using two different techniques. The first uses the **ExecuteXmlReader** method of **SqlCommand** to get an **XmlTextReader**, which is then populated into a DataSet using the **ReadXml** method of the **DataSet** class. The second uses the **SqlDataAdapter** to extract the data content and adapt it to the XML structure in the DataSet. The **SqlDataAdapter** class encapsulates a set of data commands and a database connection that is used to fill the DataSet and potentially update the data source.

VB xmlfromsqlsrv.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

This sample begins by opening a connection to the SQL Server Northwind database. Then a SqlCommand is created that will select all rows from the Customers table of the Northwind database. In this command the FOR XML clause is used to indicate you are requesting the SQL Server return the results of the query as an XML document. The XMLDATA flag on the FOR XML clause specifies that an XML-Data schema should be returned. The schema is prepended to the document as an inline schema.

```
' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (for example, Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
Dim sConnection as String = "server=(local)\\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind"
Dim mySqlConnection as SqlConnection = new SqlConnection(sConnection)
Dim mySqlCommand as SqlCommand = new SqlCommand("select * from customers FOR XML AUTO, XMLDATA", mySqlConnection)
mySqlCommand.CommandTimeout = 15
...
mySqlConnection.Open()
```

VB

In the first section of the sample a DataSet called myDataSet1 is created. Then, you add an **XmlTextReader** and the **XmlReadMode.Fragment** flag to read the XmlDocument returned from SQL Server into myDataSet1 using the **ReadXml** method. This method infers and creates a schema from the data stream.

In the second section of the sample a DataSet called myDataSet2 is created and a SqlDataAdapter is created to provide a SQL command to access and retrieve the data from the SQL server database. Then the **Fill** method is called on the SqlDataAdapter to load the data into myDataSet2.

The final step is to write out both datasets to validate that the data was the same and review the XML format.

```
' Now create the DataSet and fill it with xml data.
Dim myDataSet1 as DataSet = new DataSet()
myDataSet1.ReadXml(mySqlCommand.ExecuteXmlReader(), XmlReadMode.Fragment)
```

```
' Modify to match the other dataset
myDataSet1.DataSetName = "NewDataSet"


' Get the same data through the provider.
Dim mySqlDataAdapter as SqlDataAdapter = new SqlDataAdapter("select * from customers", sConnection)
Dim myDataSet2 as DataSet = new DataSet()
mySqlDataAdapter.Fill(myDataSet2)

' Write data to files: data1.xml and data2.xml.
myDataSet1.WriteXml("data1.xml")
myDataSet2.WriteXml("data2.xml")
Console.WriteLine ("Data has been written to the output files: data1.xml and data2.xml")
```

VB

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET
Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Reading data into a Dataset

This sample illustrates using ADO recordsets with managed Datasets. To provide access to ADO Recordset and Record objects from ADO.NET, the OLE DB .NET Data Provider overloads the Fill method of the OleDbDataAdapter to accept an ADO Recordset object or an ADO Record object. Filling a DataSet with the contents of an ADO object is a one-way operation. That is, data can be imported from the ADO Recordset or Record into the DataSet, but any updates to the data must be handled explicitly by either ADO.NET or ADO.

The Fill operation will append rows to the existing DataTables in the DataSet if no primary key information is available. If primary key information is available for the DataTable, then the Fill operation will update rows with matching primary keys.

In this sample the same basic procedures that were used with old ADO Recordset method of getting data are followed. First a connection to the database is created and then a SQL command is executed on the connection or the Recordset.

VB adorstodataset.aspx

[Run Sample] | [View Source]

To consume a COM component that returns an ADO Recordset or Record object using .NET COM Interop services and ADO.NET, it is required to first import the type library information for the COM component and ADO using TlbImp.exe. For example, an existing COM component with a ProgId of ADOComponent.DataClass is compiled into ADOComponent.dll. It has methods that return objects of type ADODB.Recordset. To consume this object from .NET, both ADOComponent.dll, and msado15.dll would have to be imported, which contain the ADODB.Recordset and ADODB.Record objects. The following commands would need to be issued to import the COM type libraries to .NET:

TlbImp "C:\Program Files\Common Files\System\Ado\msado15.dll" /out:ADODB.dll TlbImp ADOComponent.dll /out:ADOCOM.dll

The resulting .NET libraries, ADODB.dll and ADOCOM.dll, would then be passed as library references when compiling a .NET program as shown below.

```
vbc MyVB.vb /r:system.dll /r:system.data.dll /r:system.xml.dll /r:ADODB.dll /r:ADOCOM.dll
```

VB

Summary

1. The Fill method on the OleDbDataAdapter provides access to ADO Recordset and Record objects.
2. Filling a DataSet with the contents of an ADO object is a one-way operation.
3. ADO.NET provides access to ADO via the .NET COM Interop services.

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET
Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Retrieve Data using OLE DB

This sample illustrates how to read data from an OLE DB provider using the **OleDbDataReader** class. This class provides a way of reading a forward-only stream of data records from a data source. If you want to work with SQLServer 7.0 or higher, you should refer to the topic Retrieve Data from SQLServer.

The **OleDbDataReader** is created by calling the ExecuteReader method of the OleDbCommand, not through direct use of the constructor. While the **OleDbDataReader** is in use, the associated **OleDbConnection** is busy serving the **OleDbDataReader**. While in this state, no other operations can be performed on the **OleDbConnection** other than closing it. This is the case until the Close method of the **OleDbDataReader** is called.

VB adodtreader.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

OleDbDataReader provides a means of reading a forward-only stream of data records from an OleDb data source. For more interactive operations, such as scrolling, filtering, navigating, remoting, etc., use the DataSet.

The example creates an OleDbConnection to the Northwind database using the SQL Server .NET Data Provider. The OleDbCommand selecting items from the employee table is then executed using the OleDbCommand ExecuteReader method. The results of this command are passed to the **OleDbDataReader**.

```
' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (e.g. Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
Dim myDataReader as OleDbDataReader
Dim myOleDbConnection as OleDbConnection
Dim myOleDbCommand as OleDbCommand

myOleDbConnection = new OleDbConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;
    database=northwind;provider=sqloledb")
myOleDbCommand = new OleDbCommand("SELECT EmployeeID, LastName,
    FirstName, Title, ReportsTo FROM Employees", myOleDbConnection)
...
myOleDbConnection.Open()
myDataReader = myOleDbCommand.ExecuteReader()
```

VB

The example reads through the data using the **OleDbDataReader** Read method and writing the data elements out to the console.

```
do while (myDataReader.Read())
    ...
    if (myDataReader.IsDBNull(4)) then
      Console.Write("N/A" + Chr(10))
    else
      Console.Write(myDataReader.GetInt32(4).ToString() + Chr(10))
    end if
loop
```

VB

Finally, the example closes the **OleDbDataReader**, then the **OleDbConnection**.

```
' Always call Close when done reading.
myDataReader.Close()

' Close the connection when done with it.
myOleDbConnection.Close()
```

VB

Summary

1. An **OleDbDataReader** is for reading a forward-only stream of data records from a data source fast.
2. Remember to close the **OleDbDataReader** and then the **OleDbConnection**.
3. Remember if the **OleDbDataReader** is in use, the associated **OleDbConnection** is busy serving the **OleDbDataReader** and while in this state, no other operations can be performed on the **OleDbConnection** other than closing it.

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Retrieve Data from SQL Server

This sample illustrates how to read data from SQL Server using the **SqlDataReader** class. This class provides a way of reading a forward-only stream of data records from a data source. If you want to work with databases that have OLE DB interfaces or versions of SQL Server prior to SQL Server 7.0, see Retrieve Data Using OLE DB.

The **SqlDataReader** is created by calling the **ExecuteReader** method of the **SqlCommand**, not through direct use of the constructor. While the **SqlDataReader** is in use, the associated **SqlConnection** is busy serving the **SqlDataReader**. While in this state, no other operations can be performed on the **SqlConnection** other than closing it. This is the case until the **Close** method of the **SqlDataReader** is called.

VB sqldtreader.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

SqlDataReader provides a means of reading a forward-only stream of data records from a SQL Server data source. For more interactive operations such as scrolling, filtering, navigating, and remoting, use the DataSet.

The example creates a SqlConnection to the Northwind database. The **SqlCommand** selecting items from the employee table is then executed using the **SqlCommand ExecuteReader** method. The results of this command are passed to the **SqlDataReader**.

```
    Dim myDataReader as SqlDataReader
    Dim mySqlConnection as SqlConnection
    Dim mySqlCommand as SqlCommand

    ' Warning: Embedded usernames/passwords in database connection strings are inherently
    ' insecure, and should only be used on platforms (for example, Win9x) that do not support Integrated
    ' Security.  See the ASP.NET QuickStart for examples of Integrated Security.
    mySqlConnection = new SqlConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
    mySqlCommand = new SqlCommand("SELECT EmployeeID, LastName, FirstName, Title, ReportsTo FROM Employees",
mySqlConnection)
    ...
    mySqlConnection.Open()
    myDataReader = mySqlCommand.ExecuteReader(CommandBehavior.CloseConnection)
```

VB
The example reads through the data using the **SqlDataReader Read** method and writing the data elements out to the console.

```
    do while (myDataReader.Read())
      Console.Write(myDataReader.GetInt32(0).ToString() + Chr(9))
```

```
  Console.Write(myDataReader.GetString(2) + " " + myDataReader.GetString(1) + Chr(9))
  Console.Write(myDataReader.GetString(3) + Chr(9))
  if (myDataReader.IsDBNull(4)) then
    Console.Write("N/A" + Chr(10))
  else
    Console.Write(myDataReader.GetInt32(4).ToString() + Chr(10))
  end if
loop
```

VB

Finally, the example closes the **SqlDataReader**, then the **SqlConnection**.

```
' Always call Close when done reading.
myDataReader.Close()

' Close the connection when done with it.
mySqlConnection.Close()
```

VB

Summary

1. A **SqlDataReader** is for reading a forward-only stream of data records from SQL Server.
2. Remember to close the **SqlDataReader** and then the **SqlConnection**.
3. There can be only one **SqlDataReader** open at a time on the **SqlConnection**. If the **SqlDataReader** is in use, the associated **SqlConnection** is busy serving the **SqlDataReader** and while in this state, no other operations can be performed on the **SqlConnection** other than closing it.

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Convert an ADO Application to ADO.NET

This is an example of an ADO application that has been ported to .NET without the original graphical interface. It also shows the usage of a forward-only read-only fast DataReader. It shows how to use a **DataView** class to take a table from a DataSet and operate on it in a fashion similar to the old ADO Recordset model. Remember an ADO Recordset only holds data for one table, but an ADO.NET **DataSet** can hold multiple tables and therefore is very flexible.

The original ADO sample is the employee sample in SQL Server 2000. The original sample uses an ADO Recordset to manage the resulting data returned from the SQL query. The new sample shows how to use the SqlDataAdapter to fill a DataSet in a model similar to the ADO Recordset. Also, the original sample uses a dialog window (.cpp) or a form (.frm) to display the output of the employee table of the Northwind database. However, this .NET example only uses the Console window to output the non-image type data in the sample.

The main topics covered in this sample are:

- Connection to the database
- Usage of a light-weight read-only, forward-only reader
- Execution of the SQL query and resulting ADO Recordset or ADO.NET **DataSet**
- Accessing individual records in the ADO Recordset or ADO.NET **DataSet**

VB employees.aspx

[Run Sample] | [View Source]

Typically in an ADO application there is a connection to the database and execution of the SQL query and resulting ADO Recordset.

In the original code used in the Visual Basic version of Employee, a connection is opened to the SQL Server database 'Northwind' using an ADO connection object and a connection string "server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind;provider=sqloledb". A Recordset is then returned using the ADO **Recordset.Open** method with a SQL 'SELECT' query.

Then, the **FillDataFields** function is called to retrieve individual record values.

```
' Open the database.
' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (for example, Win9x) that do not support Integrated
' Security.  See the ASP.NET QuickStart for examples of Integrated Security.
cn.Open("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind;provider=sqloledb")

' Open the Recordset.
Set rs = New ADODB.Recordset
rs.Open "select * from Employees", cn, adOpenKeyset, adLockPessimistic
```

```
' Move to the first record and display the data.
rs.MoveFirst
FillDataFields
```

In the .NET implementation the steps are very similar. A connection is opened to the SQL Server database 'Northwind' using a **SqlConnection** object and a connection string "server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind". A Reader is then used to cycle through the data returned on execution of the query. This reader read-only, forward-only DataReader is more performant than its native ADO counterpart. A **DataSet** is then populated using **SqlDataAdapter** with a SQL 'SELECT' query and the **SqlDataAdapter Fill** method.

```
    Dim mySqlConnection as SqlConnection = new
SqlConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
    Dim mySqlDataAdapter as SqlDataAdapter = new SqlDataAdapter("select * from employees", mySqlConnection)
    Dim myDataSet as DataSet = new DataSet()
    mySqlDataAdapter.Fill(myDataSet,"Employees")
```

VB

Then in an ADO application there is some function that moves among the Recordset getting data from each record and field combination. This would typically be accomplished using calls to the Recordset **MoveFirst**, **MoveLast**, **MovePrevious**, and **MoveNext** methods. For Example:

```
If rs.EOF = False Then
    If rs.BOF = True Then
         rs.MoveFirst
    End If
    rs.MoveNext
End If
If rs.EOF = False Then
    FillDataFields
End If
```

Then data would be extracted from each current record in a Recordset using individual field accessors. For Example:

```
For Each fld In Flds
    FieldSize = fld.ActualSize
    If FieldSize > 0 Then
        Select Case fld.Name
            Case "EmployeeID"
                txtEID.Text = Str(fld.Value)
            Case "LastName"
                txtLastName.Text = fld.Value
            Case "FirstName"
                txtFirstName.Text = fld.Value
            Case "Title"
                txtTitle.Text = fld.Value
            ...
        End Select
```

```
        End If
    Next
```

In this example, the DataSet Table "Employee" is specifically assigned to a **DataView** and the resulting **DataView** iterated over to extract the data values. By using a **DataView** you can turn any table in a **DataSet** into an object that functions similarly to the old ADO Recordset.

```
    ' Create a new dataview instance on the Employees table that was just created
    Dim myDataView as DataView = new DataView(myDataSet.Tables("Employees"))

    ' Sort the view based on the first column name.
    myDataView.Sort = "EmployeeID"

    Dim iReportsTo as integer
    Dim i as integer

    for i = 0 to myDataView.Count -1
      Console.Write(Chr(10) & "************************ Employee number " & (i+1).ToString() + "
************************" & Chr(10))
      Console.Write("EmployeeID:" & Chr(9) & myDataView(i)("EmployeeID").ToString() + Chr(10) & _
                    "FirstName:" & Chr(9) & myDataView(i)("FirstName").ToString() + Chr(10) & _
                    "LastName:" & Chr(9) & myDataView(i)("LastName").ToString() + Chr(10) & _
                    "Title:" & Chr(9) & Chr(9) & myDataView(i)("Title").ToString() + Chr(10) & _
                    "TitleOfCourtesy:" & myDataView(i)("TitleOfCourtesy").ToString() + Chr(10) & _
      ...
    next
```

VB

Copyright 2001 Microsoft Corporation. All rights reserved.

Opened Connection to server=(local)\NetSDK;uid=QSUser;database=northwind
Closed Connection. It is important to close connections explicitly.

New Record inserted into Customers table in northwind.

Customer ID	Company Name
ABC	ABC Company
ALFKI	Alfreds Futterkiste
ANATR	Ana Trujillo Emparedados y helados
ANTON	Antonio Moreno Taquería
AROUT	Around the Horn
BERGS	Berglunds snabbköp
BLAUS	Blauer See Delikatessen
BLONP	Blondesddsl père et fils
BOLID	Bólido Comidas preparadas
BONAP	Bon app'
BOTTM	Bottom-Dollar Markets
BSBEV	B's Beverages
CACTU	Cactus Comidas para llevar
CENTC	Centro comercial Moctezuma
CHOPS	Chop-suey Chinese
COMMI	Comércio Mineiro
CONSH	Consolidated Holdings
DRACD	Drachenblut Delikatessen
DUMON	Du monde entier
EASTC	Eastern Connection
ERNSH	Ernst Handel
FAMIA	Familia Arquibaldo
FISSA	FISSA Fabrica Inter. Salchichas S.A.
FOLIG	Folies gourmandes
FOLKO	Folk och fä HB
FRANK	Frankenversand
FRANR	France restauration
FRANS	Franchi S.p.A.
FURIB	Furia Bacalhau e Frutos do Mar
GALED	Galería del gastrónomo
GODOS	Godos Cocina Típica
GOURL	Gourmet Lanchonetes
GREAL	Great Lakes Food Market
GROSR	GROSELLA-Restaurante
HANAR	Hanari Carnes
HILAA	HILARION-Abastos
HUNGC	Hungry Coyote Import Store
HUNGO	Hungry Owl All-Night Grocers
ISLAT	Island Trading
KOENE	Königlich Essen
LACOR	La corne d'abondance
LAMAI	La maison d'Asie
LAUGB	Laughing Bacchus Wine Cellars
LAZYK	Lazy K Kountry Store
LEHMS	Lehmanns Marktstand

LETSS Let's Stop N Shop
LILAS LILA-Supermercado
LINOD LINO-Delicateses
LONEP Lonesome Pine Restaurant
MAGAA Magazzini Alimentari Riuniti
MAISD Maison Dewey
MEREP Mère Paillarde
MORGK Morgenstern Gesundkost
NewID New Company Name
NORTS North/South
OCEAN Océano Atlántico Ltda.
OLDWO Old World Delicatessen
OTTIK Ottilies Käseladen
PARIS Paris spécialités
PERIC Pericles Comidas clásicas
PICCO Piccolo und mehr
PRINI Princesa Isabel Vinhos
QUEDE Que Delícia
QUEEN Queen Cozinha
QUICK QUICK-Stop
RANCH Rancho grande
RATTC Rattlesnake Canyon Grocery
REGGC Reggiani Caseifici
RICAR Ricardo Adocicados
RICSU Richter Supermarkt
ROMEY Romero y tomillo
SANTG Santé Gourmet
SAVEA Save-a-lot Markets
SEVES Seven Seas Imports
SIMOB Simons bistro
SPECD Spécialités du monde
SPLIR Split Rail Beer & Ale
SUPRD Suprêmes délices
THEBI The Big Cheese
THECR The Cracker Box
TOMSP Toms Spezialitäten
TORTU Tortuga Restaurante
TRADH Tradição Hipermercados
TRAIH Trail's Head Gourmet Provisioners
VAFFE Vaffeljernet
VICTE Victuailles en stock
VINET Vins et alcools Chevalier
WANDK Die Wandernde Kuh
WARTH Wartian Herkku
WELLI Wellington Importadora
WHITC White Clover Markets

WILMK Wilman Kala
WOLZA Wolski Zajazd

Updating DataSet succeeded!

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Write to an event log?

Event logging provides a standard, centralized way for your applications to record important software and hardware events. Windows supplies a standard user interface for viewing the logs (the Event Viewer). Using the common language runtime's **EventLog** component, you can easily connect to existing event logs on both local and remote computers, and write entries to these logs. You can also read entries from existing logs and create your own custom event logs.

This sample illustrates how to write to an event log. It's a small console application that can be run from a command prompt. The application takes three command line arguments. The first argument is the log to which the application will write. The second is the message to be written. The third is the name of the source of the event log entry (message).

Try running the sample as follows:

```
> LogWrite.exe MyLog "Hello World!" "LogWrite Sample"
```

Now, open the event log viewer and observe that there is a new log added to the list of logs on your machine. The log's name is "MyLog"and it contains an entry written by "LogWrite Sample" (source). Double-click on the entry. It will greet you with a "Hello World!" message.

In its simplest form, writing to an event log involves:

1. Registering an event source (if not already registered):

```
If (Not EventLog.SourceExists(source)) Then
    EventLog.CreateEventSource(source, log)
End If
```

VB

2. Creating a new instance of an **EventLog** component and setting its **Source** property:

```
Dim aLog As New EventLog
aLog.Source = source
```

VB

3. Calling the **WriteEntry** method:

```
Dim message As String = "some message"
aLog.WriteEntry(message, EventLogEntryType.Information)
```

VB

Now, you may want to try to modify the sample so it writes error log entries instead of informational entries. Just change the EventLogEntryType.Information to EventLogEntryType.Error, rebuild the sample and run it again.

Example

VB LogWrite.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Write to an event log?

Event logging provides a standard, centralized way for your applications to record important software and hardware events. Windows supplies a standard user interface for viewing the logs (the Event Viewer). Using the common language runtime's **EventLog** component, you can easily connect to existing event logs on both local and remote computers, and write entries to these logs. You can also read entries from existing logs and create your own custom event logs.

This sample illustrates how to write to an event log. It's a small console application that can be run from a command prompt. The application takes three command line arguments. The first argument is the log to which the application will write. The second is the message to be written. The third is the name of the source of the event log entry (message).

Try running the sample as follows:

```
> LogWrite.exe MyLog "Hello World!" "LogWrite Sample"
```

Now, open the event log viewer and observe that there is a new log added to the list of logs on your machine. The log's name is "MyLog"and it contains an entry written by "LogWrite Sample" (source). Double-click on the entry. It will greet you with a "Hello World!" message.

In its simplest form, writing to an event log involves:

1. Registering an event source (if not already registered):

```
If (Not EventLog.SourceExists(source)) Then
    EventLog.CreateEventSource(source, log)
End If
```

VB

2. Creating a new instance of an **EventLog** component and setting its **Source** property:

```
Dim aLog As New EventLog
aLog.Source = source
```

VB

3. Calling the **WriteEntry** method:

```
Dim message As String = "some message"
aLog.WriteEntry(message, EventLogEntryType.Information)
```

VB

Now, you may want to try to modify the sample so it writes error log entries instead of informational entries. Just change the EventLogEntryType.Information to EventLogEntryType.Error, rebuild the sample and run it again.

Example

VB LogWrite.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Read from an event log?

Event logging provides a standard, centralized way for you to have your applications record important software and hardware events. Windows supplies a standard user interface for viewing the logs (Event Log). Using the Microsoft .NET Framework's **EventLog** component, you can easily connect to existing event logs on both local and remote computers, and read entries from these logs programmatically.

This sample illustrates how to enumerate entries in an event log. It is a small console application that can be run from a command prompt. The application takes two command line arguments. The first argument is the name of the log that you want to enumerate. The second argument is optional and allows you to specify the machine name of the server on which the log resides.

Try running the sample as follows:

```
> LogInfo.exe Application
```

The sample will list all the entries from the application log. You can try the same with the System, Security, and any other logs residing on your machine.

In its simplest form, writing to an event log involves:

1. Creating a new instance of an **EventLog** component and pointing it to an appropriate event log:

```
Dim log, machine As String
...
Dim aLog As New EventLog
aLog.Log = log
aLog.MachineName = machine
```

VB

2. Enumerating through the **Entries** collection of the **EventLog** instance:

```
For Each entry In alog.Entries
    Console.WriteLine("    Entry: " + entry.Message)
Next entry
```

VB

Example

VB LogInfo.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Diagnostics
Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Monitor an event log?

Event logging provides a standard, centralized way for you to have your applications record important software and hardware events. Windows supplies a standard user interface for viewing the logs (Event Log). Using the common language runtime's EventLog component, you can easily connect to existing event logs on both local and remote computers, and receive event notifications when a new entry is written to the log.

This sample illustrates how to monitor an event log for new entries. It is a small console application that can be run from a command prompt. The application takes two command line arguments. The first argument is the name of the log that you want to monitor. The second argument, which is optional, allows you to specify the machine name of the server on which the log resides.

Try running the sample as follows:

```
> LogMonitor.exe Application
```

Now run the LogWrite.exe sample and write a new entry to the application log. You will see that the **LogMonitor** is being notified about the new entry being written.

In its simplest form, monitoring an event log involves:

1. Creating a new instance of an **EventLog** component and pointing it to a appropriate event log:

```
Dim log, machine As String
...
Dim aLog As New EventLog
aLog.Log = log
aLog.MachineName = machine
```

VB

2. Adding an event handler:

```
AddHandler aLog.EntryWritten, AddressOf OnEntryWritten
```

VB

3. Handling the event notification in your event handler:

```
Sub OnEntryWritten(ByVal source As Object, ByVal e As EventLogEvent)
    Console.WriteLine("Written: " + e.Entry.Message)
End Sub
```

VB

Example

VB LogMonitor.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Write to a performance counter?

Windows performance counters allow your applications and components to publish, capture, and analyze the performance data that applications, services, and drivers provide. You can use this information to determine system bottlenecks, and fine-tune system and application performance. For example, you might use a performance counter to track the number of orders processed per second or the number of users currently connected to the system. Using the common language runtime's **PerformanceCounter** component, you can easily create your own custom counters and publish performance data relevant to your application, such as those mentioned above.

This sample illustrates publishing (writing) simple performance information to a custom performance counter. It's a small console application that can be run from a command prompt. The application takes three command line arguments. The first is a performance object name (category). The second argument is the counter name. The third argument is the counter instance name.

For example, if you want to publish data about the total number of international orders for milk, try running the sample as follows:

```
> PCWrite.exe "Orders" "Milk Orders" "International"
```

Now, wait for the application to display Started and run the PerfMon.exe. In PerfMon, click on the add toolbar button. A dialog will open. Select the Orders performance object, Milk Orders counter, and International instance. Click Add, close the dialog, and observe that you can use the PCWrite sample to change the published value by pressing + or -.

In its simplest form, writing to a custom performance counter involves:

1. Instantiating a PerformanceCounter component and pointing it to an appropriate performance counter:

```
Dim objectName As String = ...
Dim counterName As String = ...
Dim instanceName As String = ...

Dim counter As PerformanceCounter
counter = New PerformanceCounter(objectName, counterName, instanceName)
```

VB

2. Setting the RawValue property of the counter:

```
counter.RawValue = 50
```

VB

Example

VB PCWrite.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Diagnostics
Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Read from a performance counter?

Windows performance counters enable your applications and components to publish, capture (read), and analyze the performance data that applications, services, and drivers provide. You can use this information to determine system bottlenecks and fine-tune system and application performance. For example, you can use a performance counter to track the number of orders processed per second or a system's processor utilization. Using the common language runtime's **PerformanceCounter** component, you can easily read performance data relevant to your application, such as those mentioned above.

This sample illustrates how to read simple performance information from a performance counter. It is a small console application that can be run from a command prompt. The application takes three command line arguments. The first is a performance object name (category). The second argument is the counter name. The third argument is the counter instance name.

For example, in order to see the processor utilization on your machine run the sample with the following command line arguments:

```
> PCRead.exe "Processor" "% Processor Time" "_Total"
```

You will see the processor utilization data updated every half-second.

In its simplest form, reading a performance counter involves:

1. Creating a new instance of a **PerformanceCounter** component and pointing it to an appropriate performance counter:

```
Dim objectName As String = ...
Dim counterName As String = ...
Dim instanceName As String = ...

Dim counter As PerformanceCounter
counter = New PerformanceCounter(objectName, counterName, instanceName)
```

VB

2. Reading the **NextValue** property of the counter:

```
counter.NextValue()
```

VB

Remember that you have to read **NextValue** more than once to get a relevant data.

Example

VB PCRead.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Count rate of change?

Windows performance counters enable your applications and components to publish, capture, and analyze the performance data that applications, services, and drivers provide. You can use this information to determine system bottlenecks and fine-tune system and application performance. For example, you can use a performance counter to track the number of orders processes per second or the number of users currently connected to the system. Using the common language runtime's **PerformanceCounter** component, you can easily create your own custom counters and publish performance data relevant to your application, such as those mentioned above.

This sample illustrates how to publish the number of orders processed per second using a custom performance counter. It's a small console application you can run from the command prompt.

```
> PCDemo.exe
```

Now, wait for the application to display "Started" and run the PerfMon.exe. In PerfMon, choose the Add toolbar button. A dialog will open. Select the ACounterDemo performance object, CountPerSecond counter, and _Total instance. Choose Add, close the dialog, and notice that you can use the PCDemo sample to change the published value by pressing + or -. When the application starts, it simulates the processing of two new orders per second. The + and - keys can be used to double the number or divide it in two.

In its simplest form, writing to a custom performance counter that counts number of items per second involves:

1. Creating a counter of the **RateOfCountsPerSecond32** type:

```
If (Not PerformanceCounter.CategoryExists(objectName)) Then
    Dim ccd As CounterCreationData = new CounterCreationData()
    ccd.CounterName = counterName
    ccd.CounterType = PerformanceCounterType.RateOfCountsPerSecond32
    Dim ccds(1) As CounterCreationData
    ccds(0) = ccd

    PerformanceCounterCategory.Create(objectName,"Sample Object",ccds)
End If
```

VB

2. Creating a new instance of a **PerformanceCounter** component and pointing it to an appropriate performance counter:

```
Dim objectName As String = ...
Dim counterName As String = ...
Dim instanceName As String = ...

Dim counter As PerformanceCounter
counter = New PerformanceCounter(objectName, counterName, instanceName)
```

VB

3. Setting the **RawValue** property of the counter:

```
counter.IncrementBy(1)
```

VB

Example

VB PCDemo.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++
- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I... Work with tracing?

Trace instrumentation allows developers and administrators to monitor the health of applications running in real-life settings (as opposed to running in a debugger). Sometimes using a debugger can hide bugs and obscure some performance and threading problems. Tracing is a very important monitoring and debugging tool for distributed, multitier applications. Such applications often contain problems that can only be observed when the application is under a heavy load and the inherent randomness of a real-life environment.

This sample illustrates the basics of using the **Trace** and **Debug** classes. Run it and take a look at the source code. You will see how easy it is to add simple tracing to your applications.

You have to compile your instrumented applications with trace or debug compiler directives enabled. Otherwise, all calls to **Trace** or **Debug** (respectively) are ignored during compilation.

Before running the application, open a command line window and run the Debug Monitor utility (DbMon.exe). You will be able to see any debug and trace messages coming from any application running on your system.

Note: DBMon.exe is available from the Windows Platform SDK on the Microsoft Developer Network website.

Example

VB Trace.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Use tracing switches and listeners?

Trace instrumentation allows developers and administrators to monitor the health of applications running in real-life settings (as opposed to running in a debugger). Sometimes using a debugger can hide bugs and obscure some performance and threading problems. Tracing is a very important monitoring and debugging tool for distributed, multitier applications. Such applications often contain problems that you can only observe when the application is under a heavy load, and in the inherent randomness of a real-life environment.

This sample illustrates the basics of using the trace and Boolean switches. A switch provides an efficient mechanism for enabling and disabling code paths at runtime using external settings. After running it, take a look at the source code. You will see how easy it is to add simple tracing to your applications. You can then go to the config file and modify the settings of the switches and then run the sample again. You will possibly get a completely different output.

Set `value` property of the Boolean switch to one of the following: 0 (false) or 1 (true)

Set `value` property of the trace switch to one of the following: 0 (off), 1 (error), 2 (warning), 3 (info), or 4 (verbose):

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
        <system.diagnostics>
                <switches>
                        <add name="ABooleanSwitch" value="1" />
                        <add name="ATraceSwitch" value="4" />
                </switches>
            <trace autoflush="false" indentsize="4" />
    </system.diagnostics>
</configuration>
```

For more information about switches and registry, please refer to the SDK documentation.

You have to compile your instrumented applications with trace or debug compiler directives enabled. Otherwise, all calls to **Trace** or **Debug** (respectively) are ignored during compilation.

Before running the application, open a command line window and run the Debug Monitor utility (DBMon.exe). You will be able to see all debug and trace messages coming from any application running on your system.

Note: DBMon.exe is available from the Windows Platform SDK on the Microsoft Developer Network website.

Example

VB Switches.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Instrument a small application with tracing?

Trace instrumentation enables developers and administrators to monitor applications running in real-life settings (as opposed to running in a debugger). Sometimes using a debugger can hide bugs and obscure some performance and threading problems. Tracing is a very important monitoring and debugging tool for distributed, multitier applications. Such applications often contain problems that you can only observe when the application is under a heavy load, and in the inherent randomness of a real-life environment.

This sample illustrates the basics of instrumenting applications with tracing. After running it, take a look at the source code. You will see how easy it is to add simple instrumentation to your applications.

You have to compile your instrumented applications with trace or debug compiler directives enabled. Otherwise, all calls to **Trace** or **Debug** (respectively) are ignored during compilation.

Before running the application, open a command line window and run the Debug Monitor utility (DbMon.exe). You will be able to see any debug and trace messages coming from any application running on your system.

Note: DBMon.exe is available from the Windows Platform SDK on the Microsoft Developer Network website.

Example

VB TraceDemo.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Start and stop a process?

You can use the **Process** component to accomplish most process management tasks quickly and easily. You can use this component to work with processes on either local or remote computers. On a local computer, you can, for example, start and stop a process.

This sample illustrates how to start and stop an executable. It is a small console application that can be run from a command prompt. The application takes one command line argument. The argument is a full path to the executable that you are starting.

Note: This sample will only work on processes which have a window, and wait for input. Using this sample on a process which does not meet these criteria will cause an exception to be thrown.

For example, if you want to start and then stop Notepad, run the sample with the following command line argument:

```
> ProcessDemo.exe c:\winnt\notepad.exe
```

You will see Notepad started and then closed after approximately 1 second.

In its simplest form, starting a process involves:

1. Creating a new instance of a **Process** component and initializing the path to an executable:

   ```
   Dim process As New Process
   process.StartInfo.FileName = executableFilename
   ```

   VB

2. Calling the **Start** method on the component instance:

   ```
   process.Start()
   ```

   VB

A call to **WaitForInputIdle** will make sure the application actually starts and is ready for use.

Example

VB ProcessDemo.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Diagnostics

Write to an event log
Read from an event log
Monitor an event log
Write to a performance counter
Read from a performance counter
Count rate of change
Work with tracing
Use trace switches and listeners
Instrument an application
Start and stop a process
Get process information

Back to Index

Get URL for this page

How Do I...Get information about processes?

Please note that this sample might fail on Windows 95 or Windows 98 machines.

You can use the **Process** component to accomplish most process management tasks quickly and easily. You can use this component to work with processes on either local or remote computers. For example, you can query a process for specific types of information, including the names of the modules it is loaded into, the time it was started, the number of handles it is consuming, and the set of its threads.

This sample illustrates how to get a simple information about a process. It is a small console application that you can run from a command prompt. Run the sample without any command line arguments and it will print its full usage instructions.

For example, if you want to enumerate all processes on your machine, run:

```
> PrInfo.exe e
```

If you want to show the most common information about a process, run the sample with the "i" option with a process id. Remember to substitute a real process id for "125".

```
> PrInfo.exe i 125
```

In its simplest form, getting information about a process involves:

1. Creating a new instance of a **Process** component and pointing it to a process running on your system:

```
Dim process As Process = Process.GetProcessById(processid)
```

VB

2. Reading the desired property of the **Process** component:

```
Console.WriteLine("Priority: " & process.PriorityClass.Format())
```

VB

Example

VB PrInfo.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Exceptions
Catch an exception
Throw an exception

Back to Index

Get URL for this page

How Do I...Catch an Exception?

This sample illustrates how to catch an exception. A **try...catch...finally** block is used in this sample. Any code that might throw an exception is placed inside of the **try** block. If an exception is thrown, the **catch** block is entered and the program can perform the appropriate operation to recover or alert the user. The code in the **finally** block is always executed and can perform cleanup after an exception has occurred.

Catch.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Catching and dealing with exceptions is a standard part of any object-oriented program. A **try...catch** block consists of a statement that you are going to try to do something that you know might cause an exception to occur. If an exception occurs, control is immediately passed to your **catch** block, which can deal with the exception. If an exception does not occur, then your code will execute normally. As an option, you can add a **finally** block after your **catch** block, which contain code segments which will execute, whether any exceptions are raised or not.

The following code fragment demonstrates a **try...catch...finally** statement. The key points to note are the structure and the flow of control, depending on when an error may occur.

```
' This code shows how to catch an exception
Try
        Console.WriteLine("We're going to divide 10 by 0 and see what happens...")
        Console.WriteLine()

        Dim i as Integer = 10
        Dim j as Integer = 0
        Dim k As Integer k = i/j ' error on this line. Control will jump to the catch block...

Catch e As Exception  ' perform code which deals with the exception, or informs the user what occurred...
        Console.WriteLine("The following error occurred:")
        Console.WriteLine(e.ToString())  ' print the error message to the user...

Finally   ' this section will be performed, regardless of the above processing
        Console.WriteLine()
        Console.WriteLine("This statement is always printed")
End Try
```

VB

You can use the **exception** object in a number of ways. An exception has a number of properties that can help you identify help, source, or even stack information about an exception. This can be very useful for tracking down the original cause of the exception, or providing a better explanation of its source. If you do not think you are going to actually use the exception for any information, do not actually catch an object, even though you may still have a catch block. The following sample demonstrates the properties of an **exception** object and declares a **catch** block without an **exception** object.

```
' using properties of an exception object
Try
        '  ... Actions which may cause an exception go here

Catch e As Exception
        Console.WriteLine( e.Source )      ' the name of the application or object that caused the error
        Console.WriteLine( e.Message )     ' an alternate to using the ToString method, as above
        Console.WriteLine( e.StackTrace )  ' a string containing the stack trace for this exception
        Console.WriteLine( e.HelpLink )    ' a reference to a file which contains help information
End Try

' the following segment defines a try...catch block. Note that
' we do not actually define a usable Exception object, although we must include the Exception
' keyword
Try
    ' ... Actions

Catch                ' no 'e As Exception' required, if you're not going to use the exception object
    ' ... Actions
End Try
```

VB

Why are these blocks important? Try making the above application without the **try...catch** statements. Clearly, this demonstration will always raise an exception. However, imagine you have a situation where an exception might occur. If you forget to include a **try...catch** block, then your application can crash on your users.

You can take the above examples one step further. We have dealt with the particular case of a non-specific exception. However, you might know in advance what kind of exception is going to occur. When you do, you can catch the anticipated exception, and do process it

accordingly. You can then catch all other exceptions, and deal with them as well. The next sample demonstrates this.

```
' we are going to attempt to do some file processing. This can cause a number of potential exceptions
' Note that you can use the specific exceptions to get extra information about that exception,
' but this example is focused on showing control execution for those exceptions
Try
        File.Create("c:/temp/test.txt")  ' can fail for a number of reasons...

Catch ioe As IOException ' this error may occur if the temp directory does not exist...
        Console.WriteLine("An IO error occurred. It may be that the c:\temp folder does not exist")

Catch se As SecurityException ' you don't have the rights to make this action...
        Console.WriteLine("You don't have the security permissions to take this action!")

Catch e As Exception    ' catch all other exceptions...
        Console.WriteLine( e.ToString() )  ' simply print the standard exception information
End Try
```

VB

Summary

Catching exceptions is an essential part of object-oriented programming. Using **try...catch** statements is a simple way to make sure that your users are not confronted with unexpected crashes. You can get helpful information from the **exception** object about the source and nature of the exception if needed, or even catch specific exceptions if you want to customize the kind of information you want to give the user for a given exception.

Exceptions

Catch an exception

Throw an exception

Back to Index

Get URL for this page

How Do I...Throw an Exception?

This sample illustrates how to throw an exception. A **try..catch..finally** block is used in to illustrate to use and catch the exception you threw. It is important to remember that when you throw an exception, you need to document the exception you throw, so that users are aware they need to include a **try...catch** block. If an exception is thrown, actions can be taken to recover, or alert the user.

VB Throw.aspx

[Run Sample] | [View Source]

Imagine that you are developing a class that creates and reads files. You have a **Create** method in your class, which accepts a string from the user that you use to create your file. Unfortunately, someone using your method may pass it a file name like "*\?/<". You could give your user a return code, indicating that the attempt to create a file failed, but that does not explain why it failed. A good solution is to throw an exception. This provides an opportunity to give the user detailed information about the problem.

Throwing an exception involves identifying actions which can result in unexpected or undesired behavior, preventing it from happening, and letting the person who is requesting that behavior know about it. They should always know in advance which exceptions might be thrown when using your code, so that they can catch them appropriately. In this first sample, we demonstrate a simple method, which accepts a decimal value, and determines if that value is within a specified range. If it is not, then we throw an exception to indicate that we cannot process that integer for the method in this sample.

```
Public Shared Sub ResetBurgerPrice(d As Decimal)

    ' it should be noted that these exceptions do not utilize the runtime's resource
    ' system which would make the text of the exception easily localizable.
    ' For examples of using resources, see the Resources section of Quickstart.

    If d < 2 Then

        ' let anyone using this method know that charging too little for an item is no good...
        ' the type of the exception thrown is an out of range exception
        throw new ArgumentOutOfRangeException("NewPrice", "Disallowed: Charging less than $2.00 will mean we lose
money!")

    ElseIf d > 5 Then

        ' let anyone using this method know that charging too much for an item is no good either!
        ' the type of the exception thrown is an out of range exception
        Throw New ArgumentOutOfRangeException("NewPrice", "Disallowed: Charging more than $5.00 will scare people
away!")
```

```
    Else

        ' reset the internal burger price to that specified. e.g.
        ' _itemPrice = d
        ' update a database if necessary, etc.
        ' ...
    Endif
End Sub
```

VB

Before looking at how someone uses our method from outside, we are going to add example above. You may be in a situation where you want to make your own exceptions. This is not a difficult process. You must provide the basic constructors that will throw most exceptions, and inherit your exception from the **exception** class. The following example demonstrates an intentional exception that is designed specifically to be thrown when we try to change the price to 2.99.

```
' making your own exception...
public Class MyMadeUpException
                Inherits System.Exception
    Overloads Sub New ()
        MyBase.New
    End Sub

    Overloads Sub New(message As String)
        MyBase.New(message)
    End Sub

    Overloads Sub New(message As String, inner As Exception)
        MyBase.New(message, inner)
    End Sub
End Class
```

VB

Having created our new exception, we could add the following clause to the previous example of throwing an exception, to invoke our exception in particular.

```
If d < 2 Then

        ' ...

Else If d > 5 Then

        ' ...

Else If  d = 2.99 Then
```

```
        ' let anyone using this method know that we don't like 2.99!
        ' the type of the exception thrown is a 'my made up' exception
        Throw New MyMadeUpException("Disallowed: If you're going to make it $2.99, make it $3.00!")
Else

        ' ...
End If
```

VB

From the previous section, How Do I...Catch an Exception?, we can develop the following code, which will exhibit the type of behavior that someone could expect when using our method. Note that we use a loop to try out all of our exceptions, as well as those situations where the value entered as the burger value is valid.

```
' This code shows how to catch an exception
Dim d() As Decimal = {1.02, 2.34, 2.99, 3.76, 4.99, 5.99}
Dim intCount as Integer

' loop through the array, attempting each
For intCount = 0 to d.Length - 1

    Try

        Console.WriteLine()
        Console.WriteLine("{0}. Trying to call the method with an argument of '{1}'...", intCount + 1, d(intCount))
        ResetBurgerPrice(d(intCount))

        ' if you get to this line, then everything worked for that value!
        Console.WriteLine("The value {0} was successfully assigned as the new burger price.", d(intCount))
        Console.WriteLine()

    Catch aoe As ArgumentOutOfRangeException ' catch the out of range exceptions...

        Console.WriteLine("The following Out Of Range Exception occurred:")
        Console.WriteLine(aoe.Message)

    Catch mue As MyMadeUpException ' catch the made up exception. Unnecessary to do this seperately, but a good
demonstration

        Console.WriteLine("The following Made Up Exception occurred:")
        Console.WriteLine(mue.Message)

    Finally

        Console.WriteLine()
        Console.WriteLine("Move to the Next Value...")
    End Try

Next intCount
```

```
Console.WriteLine("Finished processing the Decimal Array.")
```

VB

Summary

When a method or class you are developing exhibits behavior that can cause you to end abnormally, it is a good idea to feed that error back to the person using your class by throwing the method up the calling stack.

The following example demonstrates using try...catch blocks. To show that the exception does not occur until a specified loop, we try acceptable actions first. Note that in this segment, the 'finally' block is always performed. We also print out information for this particular exception, including the Message (if you don't want the whole exception information) and the StackTrace (if you just want to know where the exception occurred).

```
We're going to divide 10 by 0 and see what happens...
Note that results are floored to the nearest integer.

Attempting to divide 10 by 10...  Success!!! Result is 1
Attempting to divide 10 by 9...  Success!!! Result is 1
Attempting to divide 10 by 8...  Success!!! Result is 1
Attempting to divide 10 by 7...  Success!!! Result is 1
Attempting to divide 10 by 6...  Success!!! Result is 2
Attempting to divide 10 by 5...  Success!!! Result is 2
Attempting to divide 10 by 4...  Success!!! Result is 2
Attempting to divide 10 by 3...  Success!!! Result is 3
Attempting to divide 10 by 2...  Success!!! Result is 5
Attempting to divide 10 by 1...  Success!!! Result is 10
Attempting to divide 10 by 0...

The following exception occured:
Exception  : System.OverflowException: Exception of type System.OverflowException was
thrown.
   at _ASP.ExceptionTest.Go()
Message    : Exception of type System.OverflowException was thrown.
StackTrace :    at _ASP.ExceptionTest.Go()

Now we can continue...

-------------------------------------------------
Now let's handle a specific exception.

We are going to try to open a non-existent file...
Trying to open file 'FileDoesn'tExist.File'...

We are sorry, we could not find that file in the specified directory.

-------------------------------------------------
Finished!
```

Globalization
Use Encodings
Set Culture and Region
Handle Multiple Encodings
Display Multiple Currencies

Back to Index

Get URL for this page

How Do I...Use different text encodings?

This sample illustrates how to use multiple encoding formats available from the **System.Text** namespace. A **StreamWriter** object is created and passed a **Stream** object along with an **Encoding** object from the **System.Text** namespace. Once created, any output method calls on the **StreamWriter** object will be written in the specified encoding.

```
Dim Fs As FileStream = New FileStream("text.txt", FileMode.OpenOrCre
 ...
Dim T As StreamWriter = New StreamWriter (Fs, Encoding.UTF8)
T.Write("This is in UTF8")
```

VB

A **StreamReader** object is used to read text in a given encoding and is constructed in the same manner as a **StreamWriter** object.

```
Dim Fs As FileStream = New FileStream("text.txt", FileMode.Open)
 ...
Dim T As StreamReader = New StreamReader(Fs, Encoding.UTF8)
Dim S As String = T.ReadLine()
```

VB

The following example illustrates creating **StreamWriter** objects of different encodings and using each object to write to a text file.

Example

VB Encoding.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Globalization
Use Encodings
Set Culture and Region
Handle Multiple Encodings
Display Multiple Currencies

Back to Index

Get URL for this page

How Do I...Use the CultureInfo and RegionInfo classes?

The **CultureInfo** and **RegionInfo** classes are exported from the **System.Globalization** namespace. **CultureInfo** contains a culture's **DisplayName**, **Calendar**, and various official abbreviations.

```
Dim C As CultureInfo = New CultureInfo("en-us")
Console.WriteLine ("The CultureInfo is set to: {0}", C.DisplayName)
Console.WriteLine ("The parent culture is: {0}", C.Parent.DisplayName)
Console.WriteLine ("The three leter ISO language name is: {0}", C.ThreeLetterISOLanguageNa
Console.WriteLine ("The default calendar for this culture is: {0}\n\n", C.Calendar.ToStrin
```

VB

RegionInfo contains information for a given region including **DisplayName**, currency information, and official abbreviations. **RegionInfo** also contains a static property to retreive the **CurrentRegion**.

```
Dim R As RegionInfo = New RegionInfo("us")
Console.WriteLine ("The name of this region is: {0}", R.Name)
Console.WriteLine ("The currency symbol for the region is: {0}", R.CurrencySymbol)
Console.WriteLine ("Is this region metric : {0}", R.IsMetric)
```

VB

Example

VB CultureRegion.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Globalization

Use Encodings

Set Culture and Region

Handle Multiple Encodings

Display Multiple Currencies

Back to Index

Get URL for this page

How Do I...Handle Multiple Encodings in one ASP.NET page?

This sample illustrates how to use **ContentEncoding** property provided in the **HttpWebResponse** class and how to create a proper **Encoding** object from the property value. This enables your application to handle multiple encodings on the same page. The **ContentEncoding** property has the value of the character set in the **ContentType** HTTP header sent from ASP.NET. The **GetEncoding()** method of the **System.Text.Encoding** class accepts the encoding name string for creating the object.

```
Dim result As HttpWebResponse = req.GetResponse()
...
Dim encode As Encoding = Encoding.GetEncoding( result.CharacterSet )
```

VB

Example

VB MultipleEncodings.aspx

[Run Sample] | [View Source]

Globalization
Use Encodings
Set Culture and Region
Handle Multiple Encodings
Display Multiple Currencies

Back to Index

Get URL for this page

How Do I...Display Multiple Currencies (the Euro and local currency)?

This sample illustrates how to use multiple currencies such as the Euro and German DM (Deutsch mark) in your application.

The **System.Globalization.NumberFormatInfo** class contains formatting information such as the currency symbol, decimal digit, separator, and so on. In many European countries, two separate currencies are commonly used: the Euro and the local currency. A static property **NumberFormatInfo.CurrentInfo** contains current local currency info, and the Currency symbol that is associated with the Euro. To represent the local currency and the Euro, you must first create a copy of the NumberFormatInfo instance using **NumberFormatInfo.Clone()**, then replace the **CurrencySymbol** property with the local currency symbol.

```
Dim LocalFormat As NumberFormatInfo = CType(NumberFormatInfo.CurrentInfo.Clone(), NumberFormatInfo)
LocalFormat.CurrencySymbol = "DM" ' Replace currency symbol with DM symbol.
```

VB

Example

VB MultiCurrency.aspx

[Run Sample] | [View Source]

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Read a text file?

This sample retrieves information from a text file, and uses an **ArrayList** to display that information for the user.

Opening and reading files for read access is an important part of IO functionality, even if you do not need to write to the file in question. In this example, we open a file for reading. This is useful for reading text files, but inappropriate for binary files. In this example, we use one of many methods available for opening the file. Although many data structures are available for storing information retrieved from the file, an **ArrayList** is the easiest to use. A more detailed analysis of the code follows the example.

VB Anagrams.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

To open a file, you need to manipulate a number of objects found in the **IO** namespace. The first class of interest is the **File** class, which defines the general methods and rules for dealing with files. The **File** class has a number of different methods for opening files, but for this specific scenario, the most appropriate is **OpenText**. This is because this particular method returns a **StreamReader** object, which is ideal for reading lines of information from a standard text file. The following code demonstrates using the **OpenText** method, passing it the name of the file you intend to open. You can include a fully-qualified path when specifying the file.

```
Dim din As StreamReader = File.OpenText ("words.txt")
```

VB

Here are some other ways in which you can open files. Each of these alternative samples could replace the previous sample, although with this code, you have to do a bit more work to translate the return value back to a **StreamReader** object. This is because each of these methods returns a **Stream** object, which can be used to construct a new instance of **StreamReader**. While you can use a **Stream** to read information from a file, a **StreamReader** lessens your workload by providing you with a richer set of methods that you can use to get information from the file that you have opened.

```
'  Use OpenRead which opens a file in read-only mode
Dim din As StreamReader = New StreamReader( File.OpenRead("words.txt") )

'  Use Open, which accepts the FileMode to open the file in.
'  Here we request FileMode.Open, but an alternative would have been Create, or OpenOrCreate
Dim din As StreamReader = New StreamReader( File.Open("words.txt", FileMode.Open) )
```

VB

Once you have opened your file, you will need to get the information from it. As mentioned earlier, the **StreamReader** is good for this because it provides you with the **ReadLine** method which enables you to get one line at a time. Of course, you need to know how the information in your file is stored to interpret it correctly. In this sample, the input file consists of one-word lines of information, making it easier to interpret.

You can place the information you retrieve into a number of datastores. In this example, you could just place each line in a string, but that is not typically what would happen. An array is a possible solution, but there is no way to tell in advance how many lines it is going to read. It would have to keep resizing, or remaking, the array (which would require extra work). The ideal solution in this scenario is to choose an **ArrayList**, a collection on which we can simply use the **Add** method to add more elements. This saves a lot of effort.

The following sample reads each new line from our opened file and places each new piece of information into an **ArrayList**. You can detect the end of the file by checking to see if the last thing that was read was null. If it was, then it has reached the end of the file (note that an empty line will not equal null).

```
Dim str As String
Dim al As ArrayList = New ArrayList() ' make our temporary storage object

' loop through all the rows, stopping when we reach the end of file
Do
        str = din.ReadLine()

        If str <> Nothing Then
                al.Add(str)           ' add each element to our ArrayList
        End If
```

```
    Loop Until str = Nothing
```

VB

Now that you have all your information from the file, you can manipulate it at will. In this example, you can choose to simply print out the information for the user. To do this, you can use a **for each** statement to go through all the elements in the **ArrayList**, printing each item to the **Console** object.

```
    Dim s As String

    For Each s in al
            Console.WriteLine (s)
    Next s
```

VB

Summary

You can read information from a text file in a number of ways. One of the easiest is to use the **File** object's **OpenText** method, in combination with a **StreamReader** object to to retrieve one line of information at a time. It is recommended that you store the information you are retrieving in a temporary store such as an array, or an **ArrayList**.

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Enumerate directories and their contents?

This sample illustrates how to use the **DirectoryInfo** class to create a directory listing. This is a great way to quickly list files for users or reports, or to find basic information about a specific directory. When used in combination with **FileInfo**, you can get all the information you need about files and directories in a specific directory.

VB Directory.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The ability to look over files and subdirectories for a given directory is essential for many programming tasks. This QuickStart shows you how to use the **FileInfo** and **DirectoryInfo** classes. Why not the **File** and **Directory** classes? This QuickStart intends to display information such as the size, and creation date for a given file. In this case, it is easier to use **FileInfo** and **DirectoryInfo** objects. The **File** class focuses on the use of non-instantiated (static in C#, or shared in Visual Basic) methods, while the **FileInfo** class offers methods based on an instance of a **FileInfo** object. **Directory** and **DirectoryInfo** are similar.

The first task explains how create an instance of the **DirectoryInfo** object (based on the current directory), then find the fully-qualified path of that directory. Next, you can make a table of all the files and directories inside our current directory, so it will print the full path of the current directory at the top of that table. We use the **GetFullPath** method of the Path class to determine the fully-qualified path of our directory.

```
' don't forget your Imports statements at the top of your code...
Imports System
Imports System.IO
' ...

'  make an object which represents our current directory. The dot (".") represents this directory
Dim dir As DirectoryInfo = new DirectoryInfo(".")

'  make the header for our table, letting the user know the path we are in...
Console.WriteLine("Following is a listing for directory: {0}", Path.GetFullPath(dir.ToString()))
```

VB

The objects inside the directory can be files or directories. You can iterate through the directory twice, looking for files first, and directories next. An alternate solution is to use the **FileSystemInfo** object, which can represent a **FileInfo** or a **DirectoryInfo** object. This means you only have to iterate through the collection once. Then, call the **GetFileSystemInfos** method of the object created in the code above, which returns an array of **FileSystemInfo** objects. You can iterate through that array with the **foreach** (**For Each** in Visual Basic), as the following code demonstrates.

```
Dim fsi As FileSystemInfo

'  loop through our array of FileSysteminfo objects
For Each fsi In dir.GetFileSystemInfos()

        '  ... the code in the following samples goes here.

Next fsi
```

VB

So now you can see what kind of object each element of the **FileSystemInfo** array is, and process it accordingly. First test the instance of the **FileSystemObject** to see if it is a file. If it is, create a new instance of the **FileInfo** object to represent it. This enables you to call the methods particular to that kind of object. If you do not do this, you could not call properties such as **Length**, which does not exist in **FileSystemInfo**. In contrast, **CreationTime** is a property of the **FileSystemInfo** object, which is why you can deal with it before you enter your **If** statement, regardless of whether you have a file or a directory.

Notice that you are shortening the name you display to the user, primarily to save screen space. Because there can be file access or other related exceptions, execute this code in a **Try** block, to make sure it does not end abnormally. Put this code together with the code in the previous examples, and you have your program (once inside a method, of course). Note that this only demonstrates how to process a file.

```
Try
        Dim creationTime As DateTime = fsi.CreationTime
        Dim subLength As Integer = 25

        If (fsi is FileInfo) Then '  check to see if the current object is a file...

                Dim f As FileInfo = CType(fsi, FileInfo)
```

```
                    '  this if statement simply ensures that we do not shorten the
                    '  name of the file too much!
                    if f.Name.Length < subLength Then
                           subLength = f.Name.Length
                    End If

                    Dim name As string = f.Name.Substring(0, subLength)
                    Dim size As long = f.Length

                    '  format the output to the screen
                    Console.WriteLine("{0, -25} {1,-12:N0} {2, -12} {3,-20:g}", _
                                  name, (size + " KB").PadLeft(12), "File", creationTime)

            Else   '  it must be a directory
                    '  ...
            End If
    Catch                 '  ignore!!! errors such as 'file being used' and so on...
    End Try
```

VB

Summary

You can working with files and directories using **FileInfo** and **DirectoryInfo**, although you can also use **File** and **Directory** as alternatives. By working with the Info-based objects, you need an instance of the class, but you can easily obtain specific information about a file or directory (such as size, or creation date).

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Create a log file?

This sample illustrates how to create a basic log file using the **IO** classes. This demonstration introduces many useful concepts such as writing to a file, and opening a file, even if the file does not yet exist. Since the information written to the file needs to be shown to the user, this example also demonstrates reading a file.

VB Logfile.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Reading and writing information to and from a file is a common task, which you will no doubt come across when programming. Many programs regularly deal with reading and writing (Microsoft Excel, Microsoft Word, or Microsoft Powerpoint are a few examples). This example shows how to create log file that is a text file (as opposed to a binary file), so you can open and read the file.

The example shows you how to create a **FileStream** object. A **FileStream** holds a reference to a file that you can manipulate. There are a few things that the **FileStream** needs to be given when making a reference to our file. First, it needs the name of the file. Next, you have to designate to the **FileStream** the mode in which you want to refer to our file. We could ask it to only open an existing file, but that means if the file does not exist, it will cause a problem. Therefore, we are going to tell the **FileStream** to open the existing file if it is there, or create a new file if it is not. Finally, you have to designate to the **FileStream** the kind of access you want to the file (read, or write). This example will demonstrat reading and writing.

```
' make a new FileStream object, ready for read and write steps.
Dim fs As FileStream = New FileStream("log.txt", FileMode.OpenOrCreate, FileAccess.ReadWrite)
```

VB

Now that you have a reference to your file, the next step is to write some log information into it. To do this, you need to create a **StreamWriter**, an object that can write a string into a **Stream** object. If you are familiar with the **FileStream** object, you might be wondering why we do not use its **Write** method. Why make the **StreamWriter** object at all? Because a **FileStream** can only write an array of bytes to a file, and many times (as in this example) you might want to write other kinds of data, such as strings. A **StreamWriter** can write strings to our file, while our FileStream cannot. This is going to make your job a lot easier.

The following segment demonstrates how to create a new **StreamWriter**. Notice that the **StreamWriter** is made with a reference to the stream. Having made the stream, you can use the **Seek** method to get to the end of the file (if you do not do this, and the file already exists, you would be putting your new information in the wrong place), then use the **Write** method to write your strings into the file. Note that **Write** itself does not guarantee that the strings get into the file, therefore to force the information through the **FileStream**, invoke the **Flush** method on your **Writer**.

```
Dim w As StreamWriter = New StreamWriter(fs) '  create a stream writer
w.BaseStream.Seek(0, SeekOrigin.End)         '  set the file pointer to the end of file

w.Write("Log Entry : {0}", vbcrlf)
w.WriteLine("{0} {1}{2}", DateTime.Now.ToLongTimeString(), DateTime.Now.ToLongDateString(), vbcrlf)

w.WriteLine("--Log Entry goes here--")

w.Flush()                                    '  update underlying file
```

VB

At this point, you have finished updating your log file. Normally, you would shut the file so anyone can open it and see the log entries. To read the file, you have to make a **StreamReader**. Again, the **FileStream** does have a **Read method**, although creating a **StreamReader** makes this easier. Having made the **StreamReader**, you need to point it to some place in your file, although this time you are going to move to the beginning of the file and move from there.

In order to determine when you have reached the end of the file, you have to 'peek' at the file from the current position of the **StreamReader** current position. If you have reached the end of the file (signified by a value less than zero), it does not keep reading the file since there is no information left. However, if there is still some information to read, you can instruct your reader to get the next line of information, using its **ReadLine** method. Once finished, you can close the **Stream**, the **Reader**, and the **Writer**.

```
Dim r As StreamReader = New StreamReader(fs) '  create a Char reader

r.BaseStream.Seek(0, SeekOrigin.Begin)       '  set the file pointer to the beginning

While r.Peek() > -1                          '  while not at the end of the file
        output.Append(r.ReadLine())          '  get the next line of information from the file
        output.Append( chr(13) )             '  a newline
End While

w.Close()                                    '  close the writer and underlying file
```

```
r.Close()                                    '  close the read
```

VB

The result is a readable file. This functionality is ideal for server administrators as well as those who maintain websites. You can track performance information, the number of hits for a site, as well as numerous other logging tasks that do not require a database.

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Read and write large files?

This sample demonstrates how to write a large file and read it back in. This sample also shows how to time such operations. It accepts a file size (measured in kilobytes) from the user, indicating the size of the file to create. The larger the specified size, the longer the process will take to both write and read the temporary file created. When working with large files, it is a good idea to give consideration to mechanisms to divert a user's attention away from the processing, such as progress indicators, or splash screens.

VB LargeReadWrite.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

This sample uses objects found in the **IO** namespace, but now you will write a large amount to a file. This concept builds on the issues discussed in How Do I...Create a log file?. However, this time you will use Binary readers and writers. This sample will also demonstrate how to measure the length of the process.

The first thing you have to do is determine the size of the file to create. This example asks the user to specify a file size, making sure that they are warned in advance about specifying numbers which are too large. The information comes from the user through a text field on our aspx page. For more details on creating the ASP.NET aspects of our example application, please review the topics listed under Welcome to the ASP.Net QuickStart Tutorial.

Once you have the file size from the user, the next step is to create a file of that size. You could create a file that specifies the size, but you generally do not know in advance how big you are going to make your file (the situation we have here is fictitious, and just for demonstration). Therefore, this example only writes one byte at a time to the file. As you can imagine, this could potentially be a very long process.

There are a few steps required to make the file. First, you need to take the size specifed by the user, and convert it to kilobytes, which is the measurement the example asked the user to specify, so you multiply their number by 1024. Now that you have the correct total, you need to create a reference to the temporary file. This example uses a **FileStream** object. When you make your reference, you need to specify a file name (make sure it is a test file because you do not want to accidently replace a necessary file), and the **FileMode** in which to open the file. Since you are not concerned if the file already exists (just in case you forgot to remove it previously), this example specifies how to open the file in **OpenOrCreate FileMode**. This means that it will open the file if it exists, or create it if it does not. Having made the reference, you can ensure the file is empty by setting the length of the **Stream** to zero, as in the following sample.

```
' a variable to hold the size of the file as specified by the user (fileSize)
Dim origTestSize As double = fileSize

' a variable to hold the actual size of the file we want to make
Dim testSize As double = origTestSize * 1024

Console.WriteLine ("Running test with size {0} KB", origTestSize.ToString())
Console.WriteLine ("This may take a while...")

FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate)
fs.SetLength(0)
```

VB

Now that you have your file ready to populate, you can begin the writing process. In this example, you will make a binary file, so you are going to create a **BinaryWriter** object. You need to make sure that the writer is pointing to the beginning of our file, so use the **Seek** method to move to the front of the file, and then loop through the file, writing the specified number of bytes to the writer. The nature of the information does not matter, since the process is important here, so we are just writing a series of ones. Once finished, you force the changes back into the file by flushing the stream. Note that at the point you write the first byte to the file, you get the current **TickCount** from the **Environment** you are operating in. You will use this to determine when you began the writing process. You can also take a count once the write is over, to determine the total duration for the write process.

```
Dim w As BinaryWriter = New BinaryWriter(fs)      ' create a BinaryWriter

w.BaseStream.Seek(0,SeekOrigin.Begin)             ' seek the beginning of the file
Dim beginWrite As Integer = Environment.TickCount ' get the start time

Dim intCount As Integer

For intCount = 0 To testSize - 1                  ' loop through the file, writing the bytes

        w.Write( CType(1, byte))                  ' do the write
```

```
        intCount = intCount + 1
Next intCount

fs.Flush()

Dim endWrite As Integer = Environment.TickCount   ' get the end time
' ... feed the information back to the user ...
```

VB

Having performed the Write, it is important to let the user know the results, so you will notice that the complete sample feeds back the information to the console. The next important step is to get the information out of the file, to see how long it takes us to read the file. In a fashion similar to writing, create a **BinaryReader**, once again setting the current postiion to the beginning of the file using the **Seek** method. Again take a **TickCount** to determine the start time, and loop through the file, reading each subsequent byte until the end of the file. Once again, flush the information from the **Stream**, and take a count of the ticks at the end of the process. Once you give the information back to the user, it's important to close the **FileStream**, and because you are making a demonstration file, use the **Delete** method of the **File** class to remove the file.

```
Dim r As BinaryReader = New BinaryReader(fs)     ' create a BinaryReader

w.BaseStream.Seek(0,SeekOrigin.Begin)            ' set the file pointer to the beginning
Dim beginRead As Integer = Environment.TickCount ' get the start time

For intCount = 0 To testSize - 1                 ' loop through, reading the bytes

        dummyInt = r.ReadByte()                  ' do the read
        intCount = intCount + 1
Next intCount

fs.Flush()

Dim endRead As Integer = Environment.TickCount   ' determine the end time
' ... feed the information back to the user ...

fs.Close()                                       ' close the file
System.IO.File.Delete("data.bin")                ' delete the file
```

VB

Creating and reading a large file is not much different from creating files of other types. It is good to be aware of the specific methods and properties associated with different input and output techniques, and the classes available to assist you with your **IO** tasks, such as Binary Readers and Writers. Make sure that you keep in mind your current position when dealing with large files.

Input and Output

Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Read and write binary data?

This sample illustrates basic binary file input and output using the **BinaryReader**, **BinaryWriter** and **FileStream** classes. A similar topic is presented under the heading How Do I...Create a log file?. Reading and writing binary information enables you to create and use files that are not accessible through other input and output methods. This example also demonstrates writing non-string data, and demonstrates the capabilities of binary I/O.

VB ReadWrite.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Although files on your computer can be stored in different types and files, one of the more common formats for files is a binary format. This short introduction to creating binary files uses the base classes **BinaryReader** and **BinaryWriter** to get information from, and put information into a file. Each of these classes wraps a stream of information, so before going further, you need create a stream which you can use to write your information to and from. Because you will be creating files, you will use a **FileStream** to expose a specific file, which in this case, you can modify if it already exists, or create if it does not exist. Once you have your **FileStream**, you can use it to construct your **BinaryReader** and **BinaryWriter**, as in the following example.

```
' Make a new FileStream object, exposing our data file.
' If the file exists, open it, and if it doesn't, then Create it.
Dim fs As FileStream = New FileStream("data.bin", FileMode.OpenOrCreate)

' create the reader and writer, based on our file stream
Dim w As BinaryWriter = New BinaryWriter(fs)
Dim r As BinaryReader = New BinaryReader(fs)
```

VB

To use the **BinaryReader**, a number of different **Read** methods are available to interpret the binary information being read from your chosen stream. This example deals with strings of information, so you will be using the **ReadString** method, but the **BinaryReader** exposes other readers such as **ReadBoolean**, or **ReadChar** for intepreting other data types. When reading information from your stream, use the **PeekChar** method to determine when you have reached the end of your stream (in this case, the end of the file). **PeekChar** returns a negative value when you have reached the end of the stream, as in the following sample.

```
' make an appropriate receptacle for the information being read in...
' a StringBuilder is a good choice
Dim output as StringBuilder = New StringBuilder()

' set the file pointer to the beginning of our file...
r.BaseStream.Seek(0, SeekOrigin.Begin)

' continue to perform this loop while not at the end of our file...
Do While r.PeekChar() > -1

        output.Append( r.ReadString() )  ' use ReadString to read from the file
Loop
```

VB

Once you have read in the information, you can do whatever you need with it. At some point however, you are probably going to want to write information back out to the file, so we are going to need our **BinaryWriter**. In this example, you are going to append information to the end of the file using the **Seek** method, so make sure the pointer to your file is at the end of the file before beginning our write. There are a couple of choices when writing information with the **BinaryWriter**. You can use the **Write** method to

write any standard form of information to the stream your writer wraps, since the **Write** method has plenty of overloads for all the kinds of information you can write. In this situation, you can also use the **WriteString** method to write a length-prefixed string to the stream. This example uses the **Write** method. Note that the **Write** method does accept a string.

```
' set the file pointer to the end of our file...
w.BaseStream.Seek(0, SeekOrigin.End)

w.Write("Putting a new set of entries into the binary file..." & chr(13))

Dim i As Integer

For i = 0 To 5 Step 1       ' some arbitrary information to write to the file

        w.Write( i.ToString() )
Next i
```

VB

Having demonstrated the basics of reading and writing binary information, it is always a good idea to open the file you have created with an an application such as Microsoft Word or notepad, just to see what the file looks like. You will notice that the file is not human-readable, since the information gets transformed by the writing process to a form which is easier for the computer to use.

Input and Output

Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Watch file system changes?

Use the **FileSystemWatcher** component to monitor a file system and react when changes to it occur. This makes it possible for you to quickly and easily launch business processes when certain files or directories are created, modified, or deleted. For example, suppose you and a group of coworkers are collaborating on a document that is stored on a shared directory on your server. Using the **FileSystemWatcher** component, you can easily program your application to watch for changes to the shared directory. When a change is detected, the component can run processing that notifies each of the group members through email.

This sample illustrates how to use a **FileSystemWatcher** to watch for any changes, renaming, creation, and deletion of any files in the specified directory. The application takes the directory that you wish to watch as the only argument.

Try running the sample as follows:

```
> Watcher.exe c:\
```

Now, open the c:\ directory and try creating, modifying and deleting a file in the directory. Observe the sample application printing an appropriate message to the console window.

In its simplest form, using a FileSystemWatcher involves:

1. Creating a new instance of the component and specifing the directory to watch:

```
Dim watcher As FileSystemWatcher = New FileSystemWatcher()
watcher.Path= "c:\Foo"
```

VB

2. Adding event handler[s]:

```
AddHandler watcher.Changed, AdressOf OnChanged
AddHandler watcher.Created, AdressOf OnChanged
```

VB

3. Specifying what the application should do when a change occurs:

```
Public Sub OnChanged(source As Object, e As FileSystemEventArgs)
  Console.Write("File: {0} {1}", e.FullPath, e.ChangeType)
End Sub
```

VB

4. Enabling the component:

```
watcher.EnableRasingEvents = True
```

VB

Example

VB Watcher.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

The following example opens a text file for reading, and copies the information in the text file to an ArrayList. The ArrayList is then printed to the screen.

```
Reading data and inserting into an ArrayList...

Printing out the ArrayList.
---------------------------
This
is
a
test
file
with
some
some
repeated
words
that
the
program
will
insert
into
an
ArrayList
```

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I... Get Started Using Interoperability?

The common language runtime's COM interoperability services make it possible to use existing COM objects from within mananged applications and expose mananged objects to exisitng COM applications.

Note: Managed code is code that targets the common language runtime.

In addition, the runtime's platform invoke mechanism makes it possible to call functions exported from unmanaged DLLs. One of the QuickStarts in this area is devoted to demonstrating COM interoperability and platform invoke.

COM Interoperability

COM interoperability provides support for producing and consuming COM objects from mananged code.

Objects that are intended to be consumed by COM applications must be designed with interoperability in mind right from the start. Before a type can be consumed by a COM application, the type must meet the following requirements.

- The managed type (class, interface, struct or enum) must be public.
- It is strongly recommended that the class explicitly implement an interface instead of exposing class interface, which contains all of the members of the class and all the members of its base classes.
- The members of the type that are being accessed from COM must be public, non-static instance memebers. Private, protected, internal, and static members are not accessible.
- If you need to create the type from COM, it must have a public, default (parameterless) constructor.

See the How Do I...Build a .NET Server Callable from COM clients? topic for details on the build process.

COM interoperability provides mechanisms that make it possible to use COM objects from managed code as well. This is the mechanism that runtime clients use to access the services provided by COM objects. Before any managed client can reference a specific COM type, the type must be described in metadata.

See the How Do I...Build a .NET Client That Uses a COM Server? topic for details on obtaining an assembly containing metadata for the COM types and the build process.

Platform Invoke

The platform invoke service makes it possible to call functions that are exported from an unmanaged DLL. Using platform invoke is similar to calling the **LoadLibrary** and **GetProcAddress** Win32 API functions, which call into an exported funtion. It is also similar to using a **Declare** statement in Visual Basic.

Each exported function being called must have a managed method

defintion. The method definition can be provided as part of any class and is attributed with the **DllImport** attribute in C# or with **Declare** statement in VB.NET to indicate that the method is implemented in unmananged code.

See the How Do I...Call a Function Exported From an Unmanaged Library? topic for details on the platform invoke syntax.

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Build a .NET Server Callable from COM clients?

This section explains how to build and install managed code that will be used from COM applications. The steps involved in the build process are as follows:

1. Write and compile the managed code.
2. Generate a type library for the assembly using the tlbexp.exe utility.
3. Install and register the assembly.
4. Write and compile the COM code that references types in the assembly.

Writing and Compiling the Managed Code

Before setting out to build an assembly that will be used from COM, it is important to understand the limitations of the common language runtime's interoperability services. Refer to the Get Started with Interoperability for specific details on exactly what those limitations are.

If your managed assembly is designed to be shared among several applications, be sure that the assembly has a shared name so that it can be installed in the global assembly cache (see How Do I...Create an Assembly with a Shared Name?). If your assembly does not have have a shared name, it can only be used by a single application.

Once the managed code is written, the compilation process is the same as it would be for any other piece of managed code.

Generating a Type Library

Most unmanaged application development tools require a type library before you can make references to external types. A type library can be generated from an assembly using the tlbexp.exe, which produces a .tlb file that can then be referenced from your unmanaged development tool.

```
tlbexp TestServer.dll
```

For example, with Visual Basic 6.0, you can reference the .tlb file from the Project/References dialog. In Visual C++ 6.0, you can use the #import statement to import the type defintions from the type library directly into C++. Once the reference to the type library is added to the project, the types defined within that library can be referenced from unmanaged code.

Installing and Registering the Assembly

In order to actually create managed types from unmanaged code, the assembly needs to be installed in the global assembly cache (GAC) and registered for use from COM.

You can install an assembly in the global assembly cache using gacutil.exe utility. Assemblies can be uninstalled using the /u option.

```
gacutil /i TestServer.dll
```

If the assembly is to be used privately by a single application, it can be copied directly to the application directory (rather than installing the assembly in the GAC). Private assemblies do not require shared names, but they must be registered for use from COM.

You can register an assembly using regasm.exe utility. Assemblies can be uninstalled using the /u option.

```
regasm TestServer.dll
```

Writing and Compiling the Unmanaged Code

Once the assembly is registered and properly installed, the types defined within the assembly can be used from COM as though they were normal COM types. For example, new objects can be created by calling **CoCreateInstance** API or by calling **CreateObject** or **GetObject** from Visual Basic or script languages. See the remaining samples in this section for specific coding details.

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Pass Simple Data from .NET Code to VB6 code?

This example demonstrates how to use .NET objects from a Visual Basic 6.0 application. The same technique can be used to create the object from any COM application, including those built with Visual C++ 6.0, VBScript, or JScript.

In this example, a very simple class called Test implements the **ITest** interface. This interface has a single method that returns the current time and can be easily used from COM. It is possible to avoid an explicit definition of interface and use class interface, which contains all of the members of the class and all the members of its base classes. This technique is available by applying the DefaultInterface.AutoDual attribute but is strongly discouraged because it seriously limits your ability to version your classes.

```
Namespace TestServer
        Public Interface ITest
                Function GetTime() As Date
        End Interface

        Public Class Test : Implements ITest
                Function GetTime() As Date Implements ITest.GetTime
                        Return DateTime.Now
                End Function
        End Class
End Namespace
```

VB

After compiling the managed code to create the TestServer assembly, the assembly must be installed in the global assembly cache (GAC) and registered for use from COM. Use gacutil.exe to install the **TestServer** assembly in the GAC and add the registry entries needed to make the types wihtin the assembly createable from COM.

```
gacutil /i TestServer.dll
```

In order to use the types defined within the **TestServer** assembly from Visual Basic 6.0, it helps to have a type library that describes the types contained in the assembly. Use tlbexp.exe to produce a type library from any managed assembly.

```
tlbexp TestServer.dll
```

The type library produced by tlbexp.exe can then be added to the project through the Project/Reference dialog in Visual Basic 6.0. Once the reference to the type library is added to the project, the types defined in the assembly can be referenced directly within the Visual Basic code.

```
Dim dotNETServer As TestServer.ITest
Set dotNETServer = New TestServer.Test

Debug.Print ".NET server returned: " + FormatDateTime(dotNETServer.GetTime, vbGeneralDate)
```

In order to run the application, the assembly is typically installed in the global assembly cache as described above. During development, it is easier to simply copy the assembly to the application directory. Assemblies can only be located if they reside in the application directory or in the global assembly cache.

Note: If you are trying to run the application from within the Visual Basic 6.0 development environment, the assembly must be in the same directory as vb6.exe because vb6.exe is the process hosting the assembly.

VB6 Test Client

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Pass an Array From .NET Code to VB6 Code?

This example demonstrates how to use a .NET object from a Visual Basic 6.0 application. The same technique can be used to create the object from any COM application including those built with Visual C++ 6.0, VBScript or JScript.

In this example, a simple class called Test implements **ITest** interface. This interface has a single method that returns an array of strings and can be easily used from COM. The array can be enumerated as any other Visual Basic array. It is possible to avoid explicit definition of interface and use class interface, which contains all of the members of the class and all the members of its base classes. This technique is available by applying the DefaultInterface.AutoDual attribute, but is strongly discouraged because it limits your ability to version your classes.

```
Namespace TestServer
        Public Interface ITest
                Function GetArray() As String()
        End Interface

        Public Class Test : Implements ITest
                Function GetArray() As String() Implements ITest.GetArray
                        Dim stringArray() As String = {"one", "two", "three"}
                        Return stringArray
                End Function
        End Class
End Namespace
```

VB

After compiling the managed code to create an assembly, the assembly must be installed in the global assembly cache (GAC) and registered for use from COM. Use gacutil.exe to install the assembly in the GAC and add the registry entries needed to enable the types within the assembly to be created from COM.

```
gacutil /i TestServer.dll
```

In order to use the type from Visual Basic 6.0, it helps to have a type library that describes the types contained in the assembly. Use tlbexp.exe to produce a type library from any managed assembly. Once a type library is available, it can be added the project through the Project/Reference dialog in Visual Basic 6.0.

```
tlbexp TestServer.dll
```

Once the reference to the type library is added to the project, the types defined in the assembly can be referenced directly with the Visual Basic 6.0 code.

```
Dim dotNETServer As TestServer.ITest
Set dotNETServer = New TestServer.Test

Dim str As Variant

Debug.Print ".NET server returned: "
For Each str In dotNETServer.GetArray
        Debug.Print str
Next
```

In order to run the application, the assembly is typically installed in the global assembly cache as described above. During development it is easier to simply copy the assembly to the application directory. Assemblies can only be located if they reside in the application directory or in the Global Assembly Cache.

Note: If you are trying to run the application from within the Visual Basic 6.0 development environment, the assembly must be in the same directory as vb6.exe because vb6.exe is really the process hosting the assembly.

VB6 Test Client

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Build a .NET Client That Uses a COM Server?

This section explains how to build managed code that uses COM. The steps involved in the build process are as follows:

1. Obtain an assembly containing definitions of the COM types to be used.
2. Install the assembly in the global assembly cache. (optional)
3. Reference the assembly containing the COM type definitions.
4. Reference the COM types.

Obtain an Assembly Containing Definitions of the COM Types to be Used

Before any managed application can reference a specific type, the type must be described in metadata. For managed types, this is easy because the compiler that produces the managed code also produces the necessary metadata. Getting metadata for existing COM types is a little more tricky. There are several ways to do this.

1. Obtain a signed assembly containing the metadata from the producer of the library. The provider of the COM types should be the first source of metadata for any COM library. This allows the vendor to sign the assembly with the correct publisher key.

2. If a signed assembly is not available from the publisher, consider using the tlbimp.exe (Type Library Importer) utility to produce an assembly yourself. You can produce an assembly from most COM type libraries. If the assembly is to be shared, the assembly must be signed with a publisher's key. The tlbimp.exe can produce signed assemblies using the /keyfile option.

```
tlbimp SHDocVw.dll /keyfile:MyKey.snk /out:ExplorerLib.dll
```

3. As a last resort, the metadata can be created directly from source code. In order to do this, the types are defined in a language like C# or Visual Basic.NET and marked with the **ComImportAttribute**. The attribute is used to by the runtime to differentiate these types from other managed types. There are several other attributes that can be used to customize how a managed type is exposed to COM.

```
[ComImport] interface IMyComInterface {...}
```

Install the Assembly in the Global Assembly Cache

If you want your assembly containing definitions of the COM types to be shared among several applications, it must be installed in the global assembly cache (GAC). Use gacutil.exe to install an assembly in the GAC.

```
gacutil /i ExplorerLib.dll
```

Reference the Assembly Containing the Type Definitions.

With Visual Basic.NET or with C#, you can reference the assembly using the compiler /r switch or you can add reference to the project directly from Visual Studio.NET development tool.

```
vbc TestClient.vb /r:ExplorerLib.dll
```

VB

Reference the COM Types

Once the reference to the type library is added to the project, the types defined within that library can be referenced from managed code. Refer to the How Do I...Call COM methods from .NET? for an example on how to do that.

Note: See the article on http://support.microsoft.com/support/misc/kblookup.asp?id=Q264957 when using VB6 components from managed code.

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Call COM Methods from .NET?

This example demonstrates how to use COM object from a Visual Basic.NET or C# application.

In order to use the types defined within a COM library from managed code, you must obtain an assembly containing definitions of the COM types. Refer to the How Do I...Build a .NET Client That Uses a COM Server? for specific details.

With Visual Basic.NET or with C#, you can reference the assembly using compiler /r switch or you can add reference to the project directly from Visual Studio.NET development tool.

```
Namespace TestClient
        Public Class Test
                Public Shared Sub Main()
                        Dim explorer As ExplorerLib.InternetExplorer
                        Dim webBrowser As ExplorerLib.IWebBrowserApp

                        explorer = New ExplorerLib.InternetExplorer
                        webBrowser = explorer

                        webBrowser.Visible = True
                        webBrowser.GoHome
                        ...
                End Sub
        End Class
End Namespace
```

VB

The following example uses the Internet Explorer object methods to display an Internet Explorer window and to navigate to the home page. In order to do that, an assembly containing definitions of the Internet Explorer types is created from SHDocVw.dll and saved into ExplorerLib.dll, which then can be referenced from code.

TestClient.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Sink Unmanaged Events from .NET?

This example demonstrates how to sink unmanaged events from .NET code.

In order to use the types defined within a COM library from managed code, you have to obtain an assembly containing definitions of the COM types. Refer to the How Do I...Build a .NET Client That Uses a COM Server? for specific details.

With Visual Basic.NET or with C#, you can reference the assembly using compiler /r switch or you can add reference to your project directly from the Visual Studio.NET development tool.

Once you have a reference, you can create new instances of the event handlers and add them to existing event handlers in unmanaged code. For specific syntax see examples below.

```
Public Shared Sub Main()
        Dim explorer As ExplorerLib.InternetExplorer
        Dim webBrowser As IWebBrowserApp

        explorer = New ExplorerLib.InternetExplorer
        webBrowser = explorer

        'Title Change event
        AddHandler explorer.TitleChange, AddressOf OnTitleChange


        webBrowser.Visible = True
        webBrowser.GoHome
        ...
End Sub

Public Shared Sub OnTitleChange(txt As String)
        Console.WriteLine("Title changes to {0}", txt)
End Sub
```

VB

The following example uses the Internet Explorer object methods and events to open an Internet Explorer window, catch all TitleChange events and show them in console window. To do this, an assembly containing definitions of the Internet Explorer types and events is created from SHDocVw.dll and saved into ExplorerLib.dll, which then can be referenced from the code.

TestClient.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Marshal Strings Between Managed and Unmanaged Code?

This example demonstrates how to marshal strings from .NET client to COM server.

In order to use the types and methods defined within a COM library from managed code, you have to obtain an assembly containing definitions of the COM types. Refer to the How Do I...Build a .NET Client That Uses a COM Server? for specific details.

This assembly contains all of the information about how parameters should be marshaled. You can call COM methods from managed code using managed data types (in this case **String** type) and they will be marshaled as COM server expects: as BSTR, as LPSTR or as LPWSTR. There is no need for you to know details, all necessary attributes are generated by tlbimp.exe tool automatically.

For example, the COM methods below will be represented as these managed methods in an assembly containing definitions of the COM types. You can call them with the managed **String** type as they are managed methods and the marshaler marshal's them to unmanaged methods properly.

COM methods:

```
HRESULT PassBStr([in] BSTR s);
HRESULT PassLPStr([in] LPSTR s);
HRESULT PassLPWStr([in] LPWSTR s);
```

Managed methods representation:

```
public void PassBStr([MarshalAs(UnmanagedType.BStr)] String s);
public void PassLPStr([MarshalAs(UnmanagedType.LPStr)] String s);
public void PassLPWStr([MarshalAs(UnmanagedType.LPWStr)] String s);
```

With Visual Basic.NET or with C#, you can reference the assembly file using the compiler /r switch or you can add reference to the project directly from Visual Studio.NET development tool.

```
Public Class Test
        Public Shared Sub Main()
                Dim strServer As StringTest2Lib.ITestString
                strServer = New StringTest2Lib.TestString

                Dim param1 As String = "String Sample 1"
                Dim param2 As String = "String Sample 2"
                Dim param3 As String = "String Sample 3"

                Console.WriteLine("Calling PassBSTR with s = {0}", param1)
                strServer.PassBStr(param1)
                Console.WriteLine("After call s = {0}", param1)

                Console.WriteLine("Calling PassLPStr with s = {0}", param2)
                strServer.PassLPStr(param2)
                Console.WriteLine("After call s = {0}", param2)

                Console.WriteLine("Calling PassLPWStr with s = {0}", param3)
                strServer.PassLPWStr(param3)
                Console.WriteLine("After call s = {0}", param3)
                ...
```

VB

The following example uses one COM server, which has the methods for string manipulation. Each method accepts one string parameter, prints it to a console window, and then changes its first letter. The only difference is how the string parameter is declared.

StringClient.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Introduction
Interop Overview

Using .NET from COM
Build a .NET COM Server
Pass simple data to VB6
Pass an array to VB6

Using COM from .NET
Build a .NET COM Client
Call COM methods
Sink unmanaged events
Marshal strings

Platform Invoke
Call a DLL Export

Back to Index

Get URL for this page

How Do I...Call a Function Exported From an Unmanaged Library?

This example demonstrates how to call a function exported from an unmanaged library from managed code. The **MessageBox** routine in User32.dll and **GetSystemTime** routine in Kernel32.dll are called. The same technique can be used to call any exported function in any other library.

In order to call **MessageBox** and **GetSystemTime**, you need the metadata that describes each method and its arguments. The necessary metadata is provided by defining a managed class with one or more static methods. The class and the static methods can have any name you choose. You might want to create several classes that contain related functions or simply group all the functions you need into a single class.

In C# the **DllImport** attribute is used to identify the name of the actual DLL that contains the exported function. The method must be defined as static and external in order to apply the attribute. In VB.NET the **Declare** statement serves the same purpose.

Many unmanaged DLL functions expect you to pass structures, as a parameter to the function. When using platform invoke to pass a structure, you must provide additional information to format the type. In managed code, a formatted type is a structure or class that is annotated with the **StructLayoutAttribute** to ensure predictable layout information to its members.

Once the class and methods are defined, the method can be called like any other method.

```
imports System.Runtime.InteropServices

Public Structure _
<StructLayout(LayoutKind.Sequential)> SystemTime
        Public wYear As Short
        ...
End Structure

Public Class Win32
        Declare Auto Sub GetSystemTime _
        Lib "Kernel32.dll"(ByRef sysTime As SystemTime)
        ...
End Class

Public Class TestPInvoke
        Public Shared Sub Main()
                Dim sysTime As New SystemTime()
                Win32.GetSystemTime(sysTime)
                ...
        End Sub
End Class
```

VB

If the name of the method is different than the actual export name in the DLL, the **DllImport.EntryPoint** property can be used to map the function to the correct entry point if you are using C#. In Visual Basic you can use **Alias** keyword for this purpose.

```
using System.Runtime.InteropServices;

Public Class Win32
        ...
        Declare Auto Function MsgBox Lib "User32.dll" Alias "MessageBox" _
        (hWnd As Integer, txt As String, caption As String, Typ As Integer) As Integer
End Class

Public Class TestPInvoke
        Public Shared Sub Main()
                ...
                Dim dt As String
                dt = ...
                Win32.MsgBox(0, dt, "Platform Invoke Sample", 0)
        End Sub
End Class
```

VB

Also, you can specify the entry by ordinal rather than by name.

```
Declare Function MsgBox Lib "User32.dll" Alias "#452" _
(hWnd As Integer, txt As String, caption As String, Typ As Integer) As Integer
```

VB

PInvokeSimple.exe

LateBreaking Samples:

- C++

[Run Sample] | [View Source]

Languages
Declare a variable
Issue a statement
Comment my code
Declare a simple property
Use an indexed property
Declare an indexed property
Declare an enumeration
Enumerate a collection or array
Declare and use a method
Define custom attributes
Declare and use an array
Statically initialize a variable
Write an if statement
Write a case statement
Write a for loop
Write a while loop
Handle exceptions
Concatenate a string
Declare an event
Declare an event handler
Add an event handler
Cast a variable to a type
Convert a variable to type
Declare a class or interface
Inherit from a base class
Implement an interface
Class with a main method
Write a standard module

Back to Index

Get URL for this page

Language Support

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

The exercises and code samples in this tutorial demonstrate how to use C#, Visual Basic, and JScript to build .NET applications. For information regarding the syntax of the other languages, refer to the complete documentation for the .NET Framework SDK.

The following table is provided to help you understand the code samples in this tutorial as well as the differences between the three languages:

Variable Declarations

```
Dim x As Integer
Dim s As String
Dim s1, s2 As String
Dim o 'Implicitly Object
Dim obj As New Object()
Public name As String
```

VB

Statements

```
Response.Write("foo")
```

VB

Comments

```
' This is a comment

' This
' is
' a
' multiline
' comment
```

VB

Accessing Indexed Properties

```
Dim s, value As String
s = Request.QueryString("Name")
value = Request.Cookies("Key").Value

'Note that default non-indexed properties
'must be explicitly named in VB
```

VB

Declaring Indexed Properties

```
' Default Indexed Property
Public Default ReadOnly Property DefaultProperty(Name As String) As String
    Get
        Return CStr(lookuptable(name))
    End Get
End Property
```

VB

Declaring Simple Properties

```
Public Property Name As String

  Get
    ...
    Return ...
  End Get

  Set
    ... = Value
  End Set

End Property
```

VB

Declare and Use an Enumeration

```
' Declare the Enumeration
Public Enum MessageSize

    Small = 0
    Medium = 1
    Large = 2
End Enum

' Create a Field or Property
Public MsgSize As MessageSize

' Assign to the property using the Enumeration values
MsgSize = small
```

VB

Enumerating a Collection

```
Dim S As String
For Each S In Coll
 ...
Next
```

VB

Declare and Use Methods

```
' Declare a void return function
Sub VoidFunction()
 ...
End Sub

' Declare a function that returns a value
Function StringFunction() As String
 ...
    Return CStr(val)
End Function

' Declare a function that takes and returns values
Function ParmFunction(a As String, b As String) As String
 ...
    Return CStr(A & B)
End Function

' Use the Functions
VoidFunction()
Dim s1 As String = StringFunction()
Dim s2 As String = ParmFunction("Hello", "World!")
```

VB

Custom Attributes

```
' Stand-alone attribute
<STAThread>

' Attribute with parameters
<DllImport("ADVAPI32.DLL")>
```

```
' Attribute with named parameters
<DllImport("KERNEL32.DLL", CharSet:=CharSet.Auto)>
```

VB

Arrays

```
Dim a(2) As String
a(0) = "1"
a(1) = "2"
a(2) = "3"

Dim a(2,2) As String
a(0,0) = "1"
a(1,0) = "2"
a(2,0) = "3"
```

VB

Initialization

```
Dim s As String = "Hello World"
Dim i As Integer = 1
Dim a() As Double = { 3.00, 4.00, 5.00 }
```

VB

If Statements

```
If Not (Request.QueryString = Nothing)
  ...
End If
```

VB

Case Statements

```
Select Case FirstName
  Case "John"
    ...
  Case "Paul"
    ...
  Case "Ringo"
    ...
  Case Else
    ...
End Select
```

VB

For Loops

```
Dim I As Integer
For I = 0 To 2
  a(I) = "test"
Next
```

VB

While Loops

```
Dim I As Integer
I = 0
Do While I < 3
```

```
    Console.WriteLine(I.ToString())
    I += 1
Loop
```

VB

Exception Handling

```
Try
    ' Code that throws exceptions
Catch E As OverflowException
    ' Catch a specific exception
Catch E As Exception
    ' Catch the generic exceptions
Finally
    ' Execute some cleanup code
End Try
```

VB

String Concatenation

```
' Using Strings
Dim s1, s2 As String
s2 = "hello"
s2 &= " world"
s1 = s2 & " !!!"

' Using StringBuilder class for performance
Dim s3 As New StringBuilder()
s3.Append("hello")
s3.Append(" world")
s3.Append(" !!!")
```

VB

Event Handler Delegates

```
Sub MyButton_Click(Sender As Object,
                   E As EventArgs)
...
End Sub
```

VB

Declare Events

```
' Create a public event
Public Event MyEvent(Sender as Object, E as EventArgs)

' Create a method for firing the event
Protected Sub OnMyEvent(E As EventArgs)
    RaiseEvent MyEvent(Me, E)
End Sub
```

VB

Add or Remove Event Handlers to Events

```
AddHandler Control.Change, AddressOf Me.ChangeEventHandler
RemoveHandler Control.Change, AddressOf Me.ChangeEventHandler
```

VB

Casting

```
Dim obj As MyObject
Dim iObj As IMyObject
obj = Session("Some Value")
```

```
iObj = CType(obj, IMyObject)
```

VB

Conversion

```
Dim i As Integer
Dim s As String
Dim d As Double

i = 3
s = i.ToString()
d = CDbl(s)

' See also CDbl(...), CStr(...), ...
```

VB

Class Definition with Inheritance

```
Imports System

Namespace MySpace

  Public Class Foo : Inherits Bar

    Dim x As Integer

    Public Sub New()
      MyBase.New()
      x = 4
    End Sub

    Public Sub Add(x As Integer)
      Me.x = Me.x + x
    End Sub

    Overrides Public Function GetNum() As Integer
       Return x
    End Function

  End Class

End Namespace

' vbc /out:libraryvb.dll /t:library
' library.vb
```

VB

Implementing an Interface

```
Public Class MyClass : Implements IEnumerable
 ...

    Function IEnumerable_GetEnumerator() As IEnumerator Implements IEnumerable.GetEnumerator
         ...
    End Function
End Class
```

VB

Class Definition with a Main Method

```
Imports System

Public Class ConsoleVB

  Public Sub New()
    MyBase.New()
    Console.WriteLine("Object Created")
  End Sub

  Public Shared Sub Main()
    Console.WriteLine("Hello World")
    Dim cvb As New ConsoleVB
```

```
  End Sub

End Class

' vbc /out:consolevb.exe /t:exe console.vb
```

VB

Standard Module

```
Imports System

Public Module ConsoleVB

  Public Sub Main()
    Console.WriteLine("Hello World")
  End Sub

End Module

' vbc /out:consolevb.exe /t:exe console.vb
```

VB

Languages

Declare a variable
Issue a statement
Comment my code
Declare a simple property
Use an indexed property
Declare an indexed property
Declare an enumeration
Enumerate a collection or array
Declare and use a method
Define custom attributes
Declare and use an array
Statically initialize a variable
Write an if statement
Write a case statement
Write a for loop
Write a while loop
Handle exceptions
Concatenate a string
Declare an event
Declare an event handler
Add an event handler
Cast a variable to a type
Convert a variable to type
Declare a class or interface
Inherit from a base class
Implement an interface
Class with a main method
Write a standard module

Back to Index

Get URL for this page

Language Support

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

The exercises and code samples in this tutorial demonstrate how to use C#, Visual Basic, and JScript to build .NET applications. For information regarding the syntax of the other languages, refer to the complete documentation for the .NET Framework SDK.

The following table is provided to help you understand the code samples in this tutorial as well as the differences between the three languages:

Variable Declarations

```
Dim x As Integer
Dim s As String
Dim s1, s2 As String
Dim o 'Implicitly Object
Dim obj As New Object()
Public name As String
```

VB

Statements

```
Response.Write("foo")
```

VB

Comments

```
' This is a comment

' This
' is
' a
' multiline
' comment
```

VB

Accessing Indexed Properties

```
Dim s, value As String
s = Request.QueryString("Name")
value = Request.Cookies("Key").Value

'Note that default non-indexed properties
'must be explicitly named in VB
```

VB

Declaring Indexed Properties

```
' Default Indexed Property
Public Default ReadOnly Property DefaultProperty(Name As String) As String
    Get
        Return CStr(lookuptable(name))
    End Get
End Property
```

VB

Declaring Simple Properties

```
Public Property Name As String

  Get
    ...
    Return ...
  End Get

  Set
    ... = Value
  End Set

End Property
```

VB

Declare and Use an Enumeration

```
' Declare the Enumeration
Public Enum MessageSize

    Small = 0
    Medium = 1
    Large = 2
End Enum

' Create a Field or Property
Public MsgSize As MessageSize

' Assign to the property using the Enumeration values
MsgSize = small
```

VB

Enumerating a Collection

```
Dim S As String
For Each S In Coll
 ...
Next
```

VB

Declare and Use Methods

```
' Declare a void return function
Sub VoidFunction()
 ...
End Sub

' Declare a function that returns a value
Function StringFunction() As String
 ...
    Return CStr(val)
End Function

' Declare a function that takes and returns values
Function ParmFunction(a As String, b As String) As String
 ...
    Return CStr(A & B)
End Function

' Use the Functions
VoidFunction()
Dim s1 As String = StringFunction()
Dim s2 As String = ParmFunction("Hello", "World!")
```

VB

Custom Attributes

```
' Stand-alone attribute
<STAThread>

' Attribute with parameters
<DllImport("ADVAPI32.DLL")>
```

```
' Attribute with named parameters
<DllImport("KERNEL32.DLL", CharSet:=CharSet.Auto)>
```

VB

Arrays

```
Dim a(2) As String
a(0) = "1"
a(1) = "2"
a(2) = "3"

Dim a(2,2) As String
a(0,0) = "1"
a(1,0) = "2"
a(2,0) = "3"
```

VB

Initialization

```
Dim s As String = "Hello World"
Dim i As Integer = 1
Dim a() As Double = { 3.00, 4.00, 5.00 }
```

VB

If Statements

```
If Not (Request.QueryString = Nothing)
  ...
End If
```

VB

Case Statements

```
Select Case FirstName
  Case "John"
    ...
  Case "Paul"
    ...
  Case "Ringo"
    ...
  Case Else
    ...
End Select
```

VB

For Loops

```
Dim I As Integer
For I = 0 To 2
  a(I) = "test"
Next
```

VB

While Loops

```
Dim I As Integer
I = 0
Do While I < 3
```

```
  Console.WriteLine(I.ToString())
  I += 1
Loop
```

VB

Exception Handling

```
Try
    ' Code that throws exceptions
Catch E As OverflowException
    ' Catch a specific exception
Catch E As Exception
    ' Catch the generic exceptions
Finally
    ' Execute some cleanup code
End Try
```

VB

String Concatenation

```
' Using Strings
Dim s1, s2 As String
s2 = "hello"
s2 &= " world"
s1 = s2 & " !!!"

' Using StringBuilder class for performance
Dim s3 As New StringBuilder()
s3.Append("hello")
s3.Append(" world")
s3.Append(" !!!")
```

VB

Event Handler Delegates

```
Sub MyButton_Click(Sender As Object,
                   E As EventArgs)
...
End Sub
```

VB

Declare Events

```
' Create a public event
Public Event MyEvent(Sender as Object, E as EventArgs)

' Create a method for firing the event
Protected Sub OnMyEvent(E As EventArgs)
    RaiseEvent MyEvent(Me, E)
End Sub
```

VB

Add or Remove Event Handlers to Events

```
AddHandler Control.Change, AddressOf Me.ChangeEventHandler
RemoveHandler Control.Change, AddressOf Me.ChangeEventHandler
```

VB

Casting

```
Dim obj As MyObject
Dim iObj As IMyObject
obj = Session("Some Value")
```

```
iObj = CType(obj, IMyObject)
```

VB

Conversion

```
Dim i As Integer
Dim s As String
Dim d As Double

i = 3
s = i.ToString()
d = CDbl(s)

' See also CDbl(...), CStr(...), ...
```

VB

Class Definition with Inheritance

```
Imports System

Namespace MySpace

  Public Class Foo : Inherits Bar

    Dim x As Integer

    Public Sub New()
      MyBase.New()
      x = 4
    End Sub

    Public Sub Add(x As Integer)
      Me.x = Me.x + x
    End Sub

    Overrides Public Function GetNum() As Integer
       Return x
    End Function

  End Class

End Namespace

' vbc /out:libraryvb.dll /t:library
' library.vb
```

VB

Implementing an Interface

```
Public Class MyClass : Implements IEnumerable
 ...

    Function IEnumerable_GetEnumerator() As IEnumerator Implements IEnumerable.GetEnumerator
         ...
    End Function
End Class
```

VB

Class Definition with a Main Method

```
Imports System

Public Class ConsoleVB

  Public Sub New()
    MyBase.New()
    Console.WriteLine("Object Created")
  End Sub

  Public Shared Sub Main()
    Console.WriteLine("Hello World")
    Dim cvb As New ConsoleVB
```

```
  End Sub

End Class

' vbc /out:consolevb.exe /t:exe console.vb
```

VB

Standard Module

```
Imports System

Public Module ConsoleVB

  Public Sub Main()
    Console.WriteLine("Hello World")
  End Sub

End Module

' vbc /out:consolevb.exe /t:exe console.vb
```

VB

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Send a message to a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to send a simple message to a message queue. To run the sample, you have to have Message Queuing installed on your system. The sample is a command-line application that takes two command-line arguments. The first argument is a name of a message queue. The path does not have to refer to an existing queue. If the queue does not exist, it will be created for you. The second command-line argument is the message to be written to the queue. For example you can run it as follows:

```
MQSend.exe MyQueue "Hello World!"
```

Now, open the Message Queuing console and examine the MyQueue message queue. It should have a message in it.

In its simplest form, sending a message to a message queue involves:

1. Creating an instance of the **MessageQueue** component and setting its **Path** property:

   ```
   Dim mq As MessageQueue = New MessageQueue(".\MyQueue")
   ```

   VB

2. Calling **Send** method to send the message:

   ```
   mq.Send("Hello World!")
   ```

   VB

Example

VB MQSend.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Messaging
Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Receive a message from a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to receive a message from a message queue. To run the sample you have to have Message Queuing installed on your system. The sample is a command-line application that takes one command-line argument. The argument is the name of a public message queue on your local machine. For example, you can run it as follows (first send a message to the queue using the MQSend sample):

```
> MQReceive.exe MyQueue
```

The sample application will receive and output to the console the first message from the queue. In its simplest form, receiving a message from a message queue involves:

1. Creating an instance of the **MessageQueue** component, setting its **Path**, and initializing the Formatter's **TargetTypeNames** property:

```
Dim mq As MessageQueue = new MessageQueue(mqPath)
Dim formatter As XmlMessageFormatter = CType(mq.Formatter,XmlMessageFormatter)
formatter.TargetTypeNames = new String(){"System.String"}
```

VB

2. Calling **Receive** to receive the message:

```
Dim m As Message = mq.Receive(New TimeSpan(0,0,3))
Console.WriteLine(CStr(m.Body))
```

VB

Example

VB MQReceive.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Pass a complex type through a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue component** to send a complex type through a message queue. To run the sample you have to have Message Queuing installed on your system.

Sending a complex type through a queue is very simple. It is actually as simple as sending a string (see MQSend sample). The **MessageQueue** component handles everything automatically. All you have to do is:

1. Create an instance of the **MessageQueue** component and set its **Path** property:

```
Dim mq As MessageQueue = New MessageQueue(".\MyQueue")
```

VB

2. Calling **Send** method to send a complex object:

```
Dim myType As MyComplexType = New MyComplexType()
mq.Send(myType)
```

VB

Example

VB SendObject.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Messaging
Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Receive asynchronously?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to watch a message queue for arrival of new messages. To run the sample you have to have Message Queuing installed on your system. The sample is a command-line application that takes one command-line argument. The argument is the name of a public message queue on your local machine. For example you can run it as follows:

```
> MQAsync.exe MyQueue
```

Now, use the MQSend sample to send a message to the MyQueue queue. The sample application will be notified when the message arrives at the queue and will output the message to the console.

In its simplest form, asynchronously receiving a message from a message queue involves:

1. Creating an instance of the **MessageQueue** component, and setting its **Path** property and the formatter:

```
Dim mq As MessageQueue = New MessageQueue(".\MyQueue")
Dim formatter As XmlMessageFormatter = CType(mq.Formatter,XmlMessageFormatter)
formatter.TargetTypeNames = new String(){"System.String"}
```

VB

2. Setting up an event handler:

```
AddHandler mq.ReceiveCompleted, New ReceiveCompletedEventHandler(AddressOf OnReceiveCompleted)
```

VB

3. Implementing an event handler:

```
Public Shared Sub OnReceiveCompleted(source As Object, asyncResult As ReceiveCompletedEventArgs)
    Dim mq As MessageQueue = CType(source,MessageQueue)
    Dim m As Message = mq.EndReceive(asyncResult.AsyncResult)
    Console.WriteLine("Message: " + CStr(m.Body))
    mq.BeginReceive()
End sub
```

VB

4. Calling **BeginReceive** to start an asynchronous receive operation:

```
mq.BeginReceive
```

VB

Please note that **BeginReceive** will receive only one message. If you want to keep receiving messages, you have to call **BeginReceive** again (see the event handler implementation in step 3).

Example

VB MQAsync.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Use message formatters?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to use different serialization mechanisms to send objects through a message queue. To run the sample you have to have Message Queuing installed on your system.

Selecting a serialization mechanism is as simple as changing the value of one property. Here are some examples:

1. Send a message using an XML based serialization mechanism:

```
mq.Formatter = New XmlMessageFormatter()
mq.Send(order)
```

VB

2. Send a message using a binary serialization mechanism:

```
mq.Formatter = New BinaryMessageFormatter()
mq.Send(order)
```

VB

3. Send a message using a serialization mechanism which is compatibile with the MSMQ ActiveX Formatter:

```
mq.Formatter = New ActiveXMessageFormatter()
mq.Send(order)
```

VB

Example

VB Formatters.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Messaging
- Send a queued message
- Receive a queued message
- Pass a complex type
- Receive asynchronously
- Use message formatters
- Control a message queue

Back to Index

Get URL for this page

How Do I...Control a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

This sample illustrates how to change some properties of a message queue. It is a small console application that can be run from a command prompt. Run the sample without any command-line arguments and it will print its full usage instructions.

For example, if you want change a queue's label, run:

```
> MQCtrl.exe MyQueue l "New Label"
```

In its simplest form, changing a property of a message queue involves:

1. Creating a new instance of a **MessageQueue** component and pointing it to an existing queue:

   ```
   Dim mq As MessageQueue = New MessageQueue(mqPath)
   ```

   VB

2. Setting the property to a new value:

   ```
   mq.Label = "New Label"
   ```

   VB

Example

VB MQCtrl.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI.

The function **getPage** is where the specific details of making the request can be found. The rest of the code in this sample is for taking command-line parameters as well as displaying help usage for the parameters. This sample program is a command-line utility that runs at the command prompt.

The **getPage** function takes a string parameter, which is the URL (or URI) of the web page you are requesting. This URI is then included as a parameter in a call to **WebRequest.Create** which creates a **WebRequest** object.

The **GetResponse** function of the **WebRequest** object is then used to get a **WebResponse** object. This object can be used to get the status code of the response, as well as the actual response stream (a web page, for instance).

Writing out the stream can take several different forms. This example reads a byte array (of 512 bytes) into the read Byte[] variable with the **Read** function. It then writes out the 512 bytes with the function: Console.Write(System.Text.Encoding.ASCII.GetString(read, 0, bytes));

Note: Behind a proxy server, only internal URI's will work with this sample.

Example

VB ClientGET.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request with SSL?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI with SSL.

There is no actual code difference between this example and the clientGET example. Basically, you can enter a URL with a http**s**:// instead of http:// and the **WebRequest** will work exactly the same.

The function **getPage** is where the specific details of making the request can be found. The rest of the code in this sample is for taking command-line parameters as well as displaying help usage for the parameters. This sample program is a command-line utility that runs at the command prompt.

The **getPage** function takes a string parameter, which is the URL (or URI) of the web page to request. This URI is then included as a parameter in a call to **WebRequest.Create** which creates a **WebRequest** object.

The **GetResponse** function of the **WebRequest** object is then used to get a **WebResponse** object. This object can be used to get the status code of the response, as well as the actual response stream (a web page, for instance).

Writing out the stream can take several different forms. This example reads a byte array (of 512 bytes) into the Byte[] variable **read** with the **Read** function. It then writes out the 512 bytes with a Console.WriteLine function: **Console.Write(System.Text.Encoding.ASCII.GetString(read, 0, bytes));**.

Example

VB ClientGETwithSSL.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request with NTLM?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI with NTLM authentication.

This sample is very similar to the clientGET sample. The difference is in the addition of the **NetworkCredential** object to handle setting a username, password, and domain to allow for NTLM authentication. Other authentication types are also just as easy. A **NetworkCredential** object is created with a username, password, and domain sent in the constructor. The object is then attached to the **WebRequest** object's **Credentials** property.

The function **getPage** is where the specific details of making the request can be found. The rest of the code in this sample is for taking command-line parameters as well as displaying help usage for the parameters. This sample program is a command-line utility that runs at the command prompt.

The **getPage** function takes a string parameter, which is the URL (or URI) of the web page to request. This URI is then included as a parameter in a call to **WebRequest.Create** which creates a **WebRequest** object.

The **GetResponse** function of the WebRequest object is then used to get a **WebResponse** object. This object can be used to get the status code of the response, as well as the actual response stream (a web page, for instance).

Writing out the stream can take several different forms. This example reads a byte array (of 512 bytes) into the Byte[] variable **read** with the **Read** function. It then writes out the 512 bytes with a Console.WriteLine function: **Console.Write(System.Text.Encoding.ASCII.GetString(read, 0, bytes));.**

Example

VB ClientGETwithNTLM.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request through a proxy?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI through a proxy.

This sample is very similar to the clientGET sample. The difference is in the creation and attachment of the proxy object. A **WebProxy** is created and (in the constructor) set to the name and port of the proxy server. This object's **BypassProxyOnLocal** property is set to true, which means to bypass the proxy for local LAN servers. The final step is to set this proxy as the global proxy for all requests with the line **GlobalProxySelection.Select = proxyObject;**.

The function **getPage** is where the specific details of making the request can be found. The rest of the code in this sample is for taking command-line parameters as well as displaying help usage for the parameters. This sample program is a command-line utility that runs at the command prompt.

The **getPage** function takes a string parameter, which is the URL (or URI) of the web page to request. This URI is then included as a parameter in a call to **WebRequest.Create** which creates a **WebRequest** object.

The **GetResponse** function of the **WebRequest** object is then used to get a **WebResponse** object. This object can be used to get the status code of the response, as well as the actual response stream (a web page, for instance).

Writing out the stream can take several different forms. This example reads a byte array (of 512 bytes) into the Byte[] variable **read** with the **Read** function. It then writes out the 512 bytes with a Console.WriteLine function: **Console.Write(System.Text.Encoding.ASCII.GetString(read, 0, bytes));**.

Example

VB ClientGETwithProxy.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request asynchronously?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI asynchronously.

This sample is similar to the other **WebRequest** samples, with the difference found in the Begin and End async-style function calls. Put simply, instead of calling **GetResponse**, a call is made to the function **BeginGetResponse**, with the callback function sent as a parameter.

This function **RespCallback**, takes an object of type **IASyncResult**. This object contains an object **AsyncState** that can be cast as a **HttpWebRequest**. Calling this casted objects **EndGetResponse** method will return a WebResponse object (which is cast as an HttpWebResponse in this case.)

From here, this example is like the others. It pulls out the response from the stream and writes it out to the console. However, this example is using the **StreamReader** and **StreamWriter** classes to do this.

Example

VB ClientGETasync.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request from within ASP.NET?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI from within a ASP.NET application.

This example is exactly the same as the command-line example showing a basic GET request with the net classes. The only difference is that this example shows that you can request another URI from within an .aspx page.

Example

VB WebRequest.aspx

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make an asynchronous GET request from within ASP.NET?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make an asynchronous GET request on a URI from within an ASP.NET application.

This example is exactly the same as the command-line example showing an asynchronous GET request with the net classes. The only difference is that this example shows that you can request another URI from within an .aspx page.

Example

VB WebRequestAsync.aspx

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Handle WebException?

This sample illustrates how to use the **WebException** and **URIFormatException** classes when dealing with exceptions when accessing URIs.

This sample is like the simple clientGET sample, except that it wraps the call in a Try/Catch wrapper. The most important section of code here (other than adding a try and catch to make sure we catch exceptions) is the catch section.

In this catch section, the object that is caught is of type **WebException** or **URIFormatException**. Both exception objects have an exception message that can be written out with the **ToString()** method. In addition, you can examine the error code of the **WebException** with the call to its **Status** property. If this status is 7, then you can get the **WebResponse** and see its status code, which will map to an HTTP status code.

Example

VB ClientException.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a POST request?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a POST request on a URI.

This sample is similar to a simple **WebRequest** with a GET verb. The are two differences here:

1) The verb needs to be changed to be POST.

2) the form information needs to be encoded and sent up the stream.

To change the verb, simple set the **Method** property to "POST".

You must then set the **ContentType** property to "application/x-www-form-urlencoded". At this point, make sure that the string you supply is encoded correctly, and everything will be POSTed correctly.

Example

VB ClientPOST.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a DateTime client with TCPClient?

This sample illustrates using the **TCPClient** class to create a DateTime program.

This application acts as a client for the standard Date and Time protocol. This protocol works on port 13, and responds to any TCP connections to that port with the current date and time.

The application firsts creates a **TCPClient** object with the default constructor. Next, the hostname that was specified at the command line is resolved into an IP address with the static class **DNS**. Next, the **Connect** method is called and the TCPClient connects to the server on port 13.

The **GetStream** method is next called and a **Stream** object is returned that represents the data returned by the server. This data is read from the **Stream** object with the **Read** method. The byte array that the **Read** method returns is turned into a string and displayed.

Example

VB DateTimeClient.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a DateTime server with TCPListener?

This sample illustrates how to use the **TCPListener** class to create a DateTime server program.

This application acts as a server for the standard Date and Time protocol. This protocol works on port 13, and responds to any TCP connections to that port with the current date and time.

The application firsts creates a **TCPListener** object with the constructor that specifies that the Listener should handle all incoming requests on port 13. Next, it goes into a loop and continues to listen for incoming connections. The connections are then accepted with the **Accept** method, and the socket returned is used to **Send** the current date and time.

The **Receive** method could be called to get data from the client, but that method is ignored, as the Date and Time protocol does not need any information that is sent from the client, but automatically returns the information.

Example

VB DateTimeServer.exe

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a chat client with UDPClient?

This sample illustrates using the **UDPClient** class to create a chat client.

This sample is a complete and fully functional chat client. It uses UDP to send and receive chat messages with other users that are also using the client. The program illustrates many important Microsoft .NET Framework concepts, but the most important (for this example) is the use of the **UDPClient** class.

THe **UDPCLient** object is initialized with a constructor call that sets the port that the object will listen on. The object then joins a Multicast group. This code can be found in the **Initialize** function of the sample code.

Once initialized, the UDPClient listens for incoming messages with the **Receive** function, which can be found in the **Listener** function of the sample code. The **Receive** function returns an array of bytes, containing the data sent. This data can be converted into a string with the ASCII object **GetString** function.

Sending data is just as easy. The **UDPClient** includes a function called **Send** which sends out a byte array to every class that is listening on the Multicast group.

To run the sample, execute Chat.exe.

Example

VB Chat.exe

[Run Sample] | [View Source]

Reflection

Get the types in an assembly

List all the members of a type

Invoke methods

Back to Index

Get URL for this page

How Do I...Get the types in an assembly?

This sample illustrates how to retrieve all the types for a given assembly. To look through the types of an assembly, you first need to identify the assembly you want to manipulate. Once you have an object reference to the assembly of interest, you can call the **GetTypes** method on that assembly, which returns an array of all the types in that assembly. You can use control logic to identify the more specific types in that array, and use iterating logic to parse your array, returning the type information to the user if needed. The ability to retrieve type information can be useful for determining alternative types you could use for a given task, or identifying existing elements which could provide you with the functionality you need.

VB GetTypes.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

The first thing to learn when retrieving types from a particular assembly is, how to identify an assembly. This QuickStart demonstrates two ways to retrieve an assembly. The first is by identifying a particular object that you want to find the assembly of, and requesting the assembly for the module of that object (remember that a module is a logical grouping of types and code, such as a .dll or .exe). The second is by using the **LoadFrom** method of the **Assembly** class to load a specific assembly for a named module (such as myapp.exe).

```
'  don't forget your using statements
Imports System
Imports System.Reflection
' ...

'  Getting an Assembly, method 1. Get the mscorlib assembly
'  Note that other types such as String, or Int32 would have worked just as well,
'  since they reside in the same assembly
Dim a As reflection.Assembly = GetType(Object).Module.Assembly

'  Getting an Assembly, method 2. Load a particular assembly, using a reference to a
'  module that is within that assembly. Note that this requires a compiled module for
'  the reference, and when running in an aspx page, will require a fully qualifed path
'  to the file, to ensure it is correctly identified
Dim b As reflection.Assembly = reflection.Assembly.LoadFrom ("GetTypes.exe")

'  note that either of the above methods is viable, depending on the information
'  you have. Since we know the name of the file which houses all of the base system
'  objects, we could do the following to replace the first example, just as effectively
'  (the absolute path may change on your machine)
'  Dim a As reflection.Assembly = reflection.Assembly.LoadFrom _
'                        ("c:/winserv/microsoft.net/framework/v1.0.2230/mscorlib.dll")
```

VB

Having identified the assembly, you can now go ahead and retrieve the types, assigning the return value from the **GetTypes** method to an array of **Type** objects. You can now manipulate the types. In the following sample, you get the types for the core runtime library, and take separate counts of the different styles of the types in that assembly (if you need more information on the foreach (For Each) statement, please look under the topic How Do I...Iterate through a collection?). In this sample, we demonstrate how to count the number of interfaces only, although we could count other members such as classes, and enums.

```
' Get all the types in the assembly identified in the previous example
Dim types() As Type = a.GetTypes ()

Dim numInterfaces As Integer = 0

Dim t As Type

For Each t in types

        ' the following line uses a set of methods which identify what
        ' kind of type we are currently querying
        If t.IsInterface Then
                '  only print out the names of the Interfaces
                Console.WriteLine (t.Name + "'")
                numInterfaces = numInterfaces + 1
        End If
```

```
Next t

'  write out the totals
Console.WriteLine("Out of {0} types in the {1} library:", _
                        types.Length, GetType(Object).Module.ToString())
Console.WriteLine ("{0} are interfaces (listed)", types.Length, numInterfaces)
```

VB

You can also use the second method identified in the first sample to retrieve the types of a given assembly. In the following sample, you will notice that it does not use the same base structure for looping through the types because you do not need keep a track of the different kinds of types. This works well when looking at a small assembly, such as the currently running application (getting a list of all the types for MSCorLib results in a very large list).

```
'  Get all the types in the assembly identified in the previous example (this assembly)
Dim types2() As Type = b.GetTypes ()

Console.WriteLine ("Get all the types from the assembly: '{0}'", b.GetName())

For Each t in types2

        Console.WriteLine(t.FullName) ' not many types, so we can print them all
Next t
```

VB

Reflection
Get the types in an assembly
List all the members of a type
Invoke methods

Back to Index

Get URL for this page

How Do I...List all the members of a type?

This example lets you list the members for a given data type. The ability to list a type's members is a great way to quickly discover which elements are available. It is an important tool for reporting on your system as well as assisting in the development of user documentation. Using the **Reflection** namespace, you can control what kind of members you want to show to your users, as well as other information (such as the visiblity of a particular method). You can also get information on all of the members in a class, or specify only certain subsets (such as methods or fields).

VB ListMembers.aspx

[Run Sample] | [View Source]

You might be wondering why it is important to get information for a particular type. After all, that is what the help system and documentation is for, right? The following example can assist you in creating user documentation, or can be used to assist in dynamically calling methods or setting properties.

There are a few steps you need to perform. First, you need to get the type that the user wants (in the form of a string) to use. Having determined what type you are working with, you need to assign an object to represent that type. This does two things: it creates an object you can work with for the following steps, and it also makes sure that the specified type exists and can be found by the system. The following example assigns an object to the System.String type. Note that although the example gives feedback to the user through the Console object in the sample here, the actual example sends the feedback to an ASP.NET label object. The interpretation, however, is the same.

```
'  don't forget your imports statements at the top of your code...
Imports System
Imports System.Reflection

'  class declaration, and method declaration...

'  remember that this string is case-sensitive, so be careful
Dim t As Type = Type.GetType("System.String")

'  check to see if we have a valid value. If our object is null, the type does not exist...
If t Is Nothing Then
        '  Don't assume that it is a SYSTEM datatype...
        Console.WriteLine("Please ensure you specify only valid types in the type field.")
        Console.WriteLine("REMEMBER: The Case matters (Byte is not the same as byte).")

        Exit Sub ' don't continue processing
End If
```

VB

Now that you have a valid type object, the next question is what kind of members do we want to retrieve for our type? Do we want the methods, the static methods, or perhaps the instance fields? In the **Reflection** namespace, there are a group of **Info** objects, each representing a different set of members of your system. For example, there is a **MethodInfo** object which can represent information about a method. There is also a generic **MemberInfo** object, which represents all of the members that can exist in a given class.

With this information, you can set up the following arrays to look at the type you have just created and find out what kind of information is inside our type. The **BitWise** operator in the sample below (the | symbol, or BitOr in Visual Basic) requests all the information for the type that meets the specified constraints. This sample demonstrates how to get all fields and all methods.

```
'  declare and populate the arrays to hold the information...
Dim fi() As FieldInfo = t.GetFields(BindingFlags.Static BitOr _
                BindingFlags.NonPublic BitOr BindingFlags.Public)  '  fields

Dim mi() As MethodInfo = t.GetMethods(BindingFlags.Static BitOr _
                BindingFlags.NonPublic BitOr BindingFlags.Public)  '  methods
```

VB

The next step is to iterate through each of the arrays and list the elements in the array on the screen (obviously, you would actually process the elements or identify a paricular element in the array). There are a few ways to do this, but this example uses a **foreach** (For Each in Visual Basic) statement.

```
Dim m As MethodInfo
Dim f As FieldInfo

'  iterate through all the method members...
For Each m In mi
        Console.WriteLine(m)
Next m

'  iterate through all the field members
For Each f In fi
        Console.WriteLine(f)
Next f

'  etc.... for each array type
```

VB

The previous code works fine, but notice how similar the two **foreach** (For Each in Visual Basic) statements are. Writing all of those out is going to be laborious and untidy (and may require a lot of maintenance if you change your code later). You can get around this by returning to the **MemberInfo** object. The **MemberInfo** object includes all of the possible information sets (methods and fields and interfaces, and so on). This can assist us because we can write the previous **foreach** statements as one statement, passing in the array to parse.

```
'  call the routine below, passing the relevant array we made in the previous step
PrintMembers( mi )  '  the method information
PrintMembers( fi )  '  the field information

Sub PrintMembers (ms() As MemberInfo)

        Dim m As MemberInfo

        '  MemberInfo is the generic info object. This can be any of the other info objects.
        For Each  m in ms
                Console.WriteLine(m)
        Next m
End Sub
```

VB

You will notice when you run the example, a couple of other things are happening along the way. Rather than hard-coding the System.String object, you can specify which class you want to get information about. It also lets you control whether you want to show static or instance information.

Reflection
Get the types in an assembly
List all the members of a type
Invoke methods

Back to Index

Get URL for this page

How Do I...Invoke methods?

This sample illustrates how to invoke different kinds of methods through reflection. Because the names of the methods being invoked are stored in strings, this mechanism provides the capability to specify methods to be invoked at runtime, rather than at design-time, providing the scope to give your users control over which particular method gets called. Although this demonstration focuses on invoking methods, you can also set and get properties and fields, if required. For another active demonstration of this topic, please see the example under the topic How Do I...Use mathematical functions?.

VB ListMembers.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

In many coding scenarios, you know the task that you want to carry out before you want to do it. Therefore, you can specify the methods that you need to invoke, and the parameters you need to pass them. However, there are also situations where you might want to dynamically invoke methods, based upon specific scenarios, or user actions. This capability is available through the **Reflection** namespace, by using the **InvokeMember** method on the **Type** object.

You can also take other actions, such as getting or setting the value of a specified property. These actions are available through the **BindingFlags** enumeration. The second parameter of **InvokeMethod** is a combination of the **BindingFlags** actions you specify. For example, if you want to invoke a static method on a class, you would include the static element in **BindingFlags**, and the **InvokeMethod BindingFlag**. The following example demonstrates how to invoke a hypothetical method called SayHello, where SayHello is a static method.

```
'  calling a static method, receiving no arguments

'  don't forget that we are using object in the reflection namespace...
Imports System
Imports System.Reflection

Public Class Invoke

        Public Shared Sub Main ()

                '  Declare a type object, used to call our InvokeMember method...
                Dim t As Type = GetType (TestClass)

                '  BindingFlags has three bitor'ed elements. Default indicates that
                '  default binding rules should be applied.
                t.InvokeMember ("SayHello", _
                                BindingFlags.Default BitOr BindingFlags.InvokeMethod _
                                BitOr BindingFlags.Static, nothing, _
                                nothing, new object () {})
        End Sub
End Class
```

VB

Take a quick look at the rest of the parameters that were passed to the **Invoke** method. The first null argument passed is requesting that the default binder be used to bind the method you are invoking. When you invoke the default binder, include the default **BindingFlags**. Instead of null as the third parameter, you can specify a **Binder** object that defines a set of properties and enables binding, which may involve selection of an overloaded method or coercion of argument types. The second null argument is the object on which to invoke the method you chose. Finally, pass an object array of the arguments that the member receives. In this case, the SayHello method receives no arguments, therefore we pass an empty array.

Here is a slightly different stuation. Call another static method called ComputeSum, but in this case, the method needs two arguments. Therefore populate an object array with those arguments, and pass them into **InvokeMember** as the last parameter.

```
'  Calling a static method, which needs arguments
Dim args as object ()
args = new object () {100.09, 184.45}

'  we know that this particular method returns a value, being the computed sum,
'  so we create a variable to hold the return
'  note the datatype of the return is object, the only datatype InvokeMethod returns...
Dim result As object
```

```
'  invoke the method. Note the change in the last parameter: the array we populated...
result = t.InvokeMember ("ComputeSum", BindingFlags.Default BitOr _
                BindingFlags.InvokeMethod BitOr BindingFlags.Static, _
                nothing, nothing, args)

'  write the results to the user's console...
Console.WriteLine ("{0} + {1} = {2}", args(0), args(1), result)
```

VB

In the previous two examples, you have invoked static methods. You can also invoke instance methods. To do this, pass in an object of the type of which you want to invoke your method on as the third parameter. This example also demonstrates that you do not have to have an actual **Type** object to use **InvokeMember**. In this case, you will generally want to use the class instance you have to call GetType, as in the following example. Note that the **BindingFlags** have changed, now that we are not invoking a static method.

```
'  Calling  an instance method
'  we need an object reference to invoke an instance member
Dim c as TestClass
c = new TestClass ()

'  use the instance of our class to call GetType
'  we no longer include the Static element in BindingFlags for our bitor
'  the fourth parameter is no longer null: we instead pass an instance
'  of the object we wish to invoke our method on
c.GetType().InvokeMember ("AddUp", BindingFlags.Default BitOr BindingFlags.InvokeMethod, _
                        nothing, c, new object () {})
c.GetType().InvokeMember ("AddUp", BindingFlags.Default BitOr BindingFlags.InvokeMethod, _
                        nothing, c, new object () {})
```

VB

Sometimes you will not want to invoke methods, but will instead need to invoke other members, such as properties or fields. To achieve this, simply alter your **BindingFlags** combination to include the appropriate element, rather than **InvokeMethod**. The following example demonstrates getting and setting a field value. The field in question is not static, therefore we need to create an instance of an object to request the field. When setting a field value, you need to pass the value you are setting as the only element of our object array parameter. When getting the value, you need to assign the return type from the **InvokeMember** method to an object.

```
'  Setting a field. Assume we are using the same Type and Class declared in the
'  previous examples (t and c). The field we are setting is the Name field
'  note the BindingFlags argument now includes SetField rather thanInvokeMember
'  Further, this is an instance field, so we pass the instance of our class
t.InvokeMember ("Name", BindingFlags.Default BitOr BindingFlags.SetField, _
                nothing, c, new object () {"NewName"})

'  similar usage...
result = t.InvokeMember ("Name", BindingFlags.Default BitOr BindingFlags.GetField, _
                nothing, c, new object () {})

Console.WriteLine ("Name == {0}", result)
```

VB

You can also get and set a property, but in this example, imagine that the property you are setting is an array or collection, which has multiple elements. To specifiy the setting of a particular element, you need to specify which index. To set a property, assign the BindingFlags.SetProperty. To specify an index of a collection or array for the property, place the index value of the element you want to set in the first element of the object array, then the value you want to set as the second element. To get the property back out, pass the index as the only element in the object array, specifying BindingFlags.GetProperty.

```
' Set an indexed property value
Dim  index As Int32 = 3

'  specify BindingFlags.SetProperty, and because this is an instance property,
'  pass the object to call the property on (c). In the object array, make two elements,
'  the first being the index, and the second being the value to set
t.InvokeMember ("Item", BindingFlags.Default BitOr BindingFlags.SetProperty, _
                        nothing, c, new object () {index, "NewValue"})

'  Get an indexed property value
'  specify BindingFlags.GetProperty, and because this is an instance property,
'  pass the object to call the property on (c). In the object array, specify the index only
result = t.InvokeMember ("Item", BindingFlags.Default BitOr BindingFlags.GetProperty, _
                        nothing, c, new object () {index})

Console.WriteLine ("Item[{0}] == {1}", index, result)
```

VB

You can also use named arguments, in which case you need to use a different overloaded version of the **InvokeMember** method. Create the array of object arguments as you have been doing so far, and also create a string array of the names of the parameters being passed. The overloaded method you want to use accepts the list of parameter names as the last parameter, and the list of values you want to set as the fifth parameter. In this demonstration, all other parameters can be null (except, of course, the first two).

```
'  Calling a method using named arguments

'  the argument array, and the parameter name array. Obviously, you will need
'  to determine the names of the parameters in advance
object[] argValues = new object [] {"Mouse", "Micky"};
String [] argNames = new String [] {"lastName", "firstName"};

'  the first five parameters for this overloaded method are the same as the
'  the five parameters we have used to this point. The final parameter needs to be
'  set to the names of the parameters
t.InvokeMember ("PrintName", BindingFlags.Default BitOr BindingFlags.InvokeMethod, _
                        nothing, nothing, argValues, nothing, nothing, argNames)
```

VB

This next example demonstrates how to invoke the default member on a class. Make sure that the class you are invoking upon has a default member specified. Then, in the **InvokeMember** method, do not specify a name for the member to invoke, as in this example.

```
'  our class with it's default member specified, using the defaultmemeber attribute

public  class  TestClass2

        public Sub PrintTime ()
                Console.WriteLine (DateTime.Now)
        End Sub
End Class

'  the client code that uses the above class...
Dim t3 As Type

t3 = GetType (TestClass2)
t3.InvokeMember ("", BindingFlags.Default  BitOr BindingFlags.InvokeMethod, _
                        nothing, new TestClass2(), new object () {})
```

VB

The final example uses a slightly different process to invoke a method. Rather than using the **Type** object directly, create a separate **MethodInfo** object directly to represent the method you will be invoking. Then, call the **Invoke** method on your **MethodInfo** object, passing an instance of the object you need to invoke your method on (if you are invoking an instance method, but null if your method is static). As before, an object array of the parameters is required. This particular sample allows you to pass parameters by reference, if required.

```
'  Invoking a ByRef member
Dim m as MethodInfo =
m = t.GetMethod("Swap")

args = new object() {CObj(1), CObj(2)}


m.Invoke(new TestClass(),args)

Console.WriteLine ("{0}, {1}", args(0), args(1))
```

VB

The following example lists the types of a given assembly. We demonstrate getting types for the assembly this particular code is a member of, as well as getting information from the MSCorLib Assembly. Because MSCorLib has so many types in it, we chose only to display the interfaces in this example (this still results in a list of over 100 interfaces).

```
Get all the types from the assembly: 'mscorlib, Version=1.0.2411.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089''
Only the Interface Types will be listed.'
---------------------------------'
ISerializable'
IComparable'
IFormattable'
IConvertible'
IRegistrationServices'
IContributeDynamicSink'
ISecurityEncodable'
IPermission'
IStackWalk'
IUnrestrictedPermission'
IBuiltInPermission'
IEvidenceFactory'
ICloneable'
IDeserializationCallback'
UCOMITypeInfo'
ISecurityPolicyEncodable'
IMembershipCondition'
IHashCodeProvider'
ISoapXsd'
IConstantMembershipCondition'
IDisposable'
IIdentity'
UCOMIMoniker'
ISurrogateSelector'
ISerializationSurrogate'
UCOMIEnumString'
IObjectReference'
ILogicalThreadAffinative'
IEnumerable'
ICustomAttributeProvider'
ICollection'
IEnumerator'
UCOMIRunningObjectTable'
ICryptoTransform'
ITypeLibImporterNotifySink'
ITypeLibExporterNotifySink'
ITypeLibConverter'
IInternalMessage'
UCOMIEnumVARIANT'
IBuiltInEvidence'
IIdentityPermissionFactory'
ILease'
IContextProperty'
IContributeObjectSink'
ISymbolNamespace'
IDictionaryEnumerator'
ISymbolBinder'
UCOMIEnumConnections'
```

```
ILIst'
IComparer'
IDictionary'
ISponsor'
IMessageCtrl'
IRemotingTypeInfo'
IChannelInfo'
IEnvoyInfo'
IContextAttribute'
ISymbolVariable'
IFormatter'
IRemotingFormatter'
UCOMIEnumConnectionPoints'
IResourceWriter'
ITrace'
IMessageSink'
IChannel'
IChannelSender'
IChannelReceiver'
IFormatterConverter'
IStreamable'
UCOMIConnectionPoint'
UCOMIPersistFile'
IContributeServerContextSink'
IContributeClientContextSink'
UCOMIBindCtx'
IReflect'
IExpando'
IPrincipal'
IFormatProvider'
IMessage'
IServiceProvider'
UCOMITypeLib'
IContributeEnvoySink'
ISymbolReader'
UCOMIReflect'
ISymbolDocumentWriter'
IFieldInfo'
INormalizeForIsolatedStorage'
ICustomMarshaler'
IActivator'
IMethodMessage'
IMethodCallMessage'
IConstructionCallMessage'
IMethodReturnMessage'
IConstructionReturnMessage'
IObjectHandle'
IAsyncResult'
IChannelReceiverHook'
IClientChannelSinkProvider'
IServerChannelSinkProvider'
IClientFormatterSinkProvider'
IServerFormatterSinkProvider'
IChannelSinkBase'
IClientChannelSink'
IServerChannelSink'
IClientFormatterSink'
```

```
IChannelDataStore'
ITransportHeaders'
IClientResponseChannelSinkStack'
IClientChannelSinkStack'
IServerResponseChannelSinkStack'
IServerChannelSinkStack'
ISerializationRootObject'
IResourceReader'
ITokenReader'
ISymbolDocument'
UCOMIEnumMoniker'
ICustomAdapter'
ISymbolMethod'
IDynamicProperty'
IDynamicMessageSink'
ISymbolScope'
ICustomFactory'
UCOMIEnumerator'
ISoapMessage'
ISymbolWriter'
ICustomFormatter'
UCOMIEnumerable'
_AppDomain'
IAppDomainSetup'
UCOMITypeComp'
UCOMIExpando'
UCOMIStream'
IConfigHandler'
IConfigHelper'
ITrackingHandler'
IContextPropertyActivator'
UCOMIConnectionPointContainer'
-----------------------------------'
Out of 1410 types in the CommonLanguageRuntimeLibrary library:'
137 are interfaces (listed)'
325 are value types'
947 are Classes'
0 are Arrays'
-----------------------------------'

Could not find the reference to the file which holds the assembly information
for this application.
Please alter the source file for this page to look in a different directory
-----------------------------------
```

Regular Expressions
Match a pattern
Get multiple matches
Make replacements
Find common patterns

Back to Index

Get URL for this page

How Do I...Use Regular Expressions to match a pattern?

Regular Expressions allow for easy parsing and matching of strings to a specific pattern. Using the objects available in the **RegularExpressions** namespace, it is possible to compare a string against a given pattern, replace a string pattern with another string, or to retrieve only portions of a formatted string.

In the simplest case, Regular Expressions can be used to do string comparisons against a pattern of strings. For instance, it can be useful to only allow strings of a particular length range, especially when accepting passwords. The following code demonstrates creating a **Regex** and testing to see if the string matches the pattern using the **IsMatch** method.

```
Dim emailregex As Regex = New Regex("(?<user>[^@]+)@(?<host>.+)");
Dim ismatch As Boolean = emailregex.IsMatch("johndoe@tempuri.org");
```

VB

This sample illustrates how to create a **Regex** object using a pattern string. The **Match** method is called and a **Match** object is returned. By examining the **Success** property, the sample decides whether to continue processing the **Match** object or to print an error message. If the match is successful, the **Groups** collection of the **Match** object can be queried for ordinal or named groups within the match. The following sample illustrates using named groups in a pattern match to validate an email address and then print the user and host portions.

Example

VB RegexMatch.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Regular Expressions
- Match a pattern
- Get multiple matches
- Make replacements
- Find common patterns

Back to Index

Get URL for this page

How Do I...Get all matches for a pattern

Regular Expressions are often useful when trying to retrieve small portions of text from a large document, result set, or when filtering a stream. The **MatchCollection** object contains all valid **Match** objects for a given regular expression after a successful match occurs.

```
Dim DigitRegex As Regex = New Regex("(?<number>\d+)")
Dim S As String = "abc 123 def 456 ghi 789"
  ...
Dim Mc As MatchCollection = DigitRegex.Matches(S)
```

VB

The following example illustrates how to create a **Regex** that matches numbers in a string. The **Matches** method is callled to return a **MatchCollection**. If the **Count** property equals 0, no successful matches have occured. If there are matches, the results of each are displayed.

Example

VB RegexMatches.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Regular Expressions
Match a pattern
Get multiple matches
Make replacements
Find common patterns

Back to Index

Get URL for this page

How Do I...Use Regular Expressions to make replacements?

The Regular Expressions library can often ease the time it takes to generate string replacement functions. By specifying a pattern of strings to be replaced, you do not have to search for every possible variation of a string. Once a **Regex** object that matches every possible string to be replaced is created, the **Replace** method can be used to generate a result. The **Replace** method can be most easily used by passing in the source string and the replacement string. The **Replace** method will return the results as a String.

```
Dim digitregex As Regex = new Regex("(?<digit>[0-9])")
Dim Before As String = "Here is so4848me te88xt with emb4493edded numbers."
 ...
Dim After As String = DigitRegex.Replace(Before, "")
```

VB

You can also reuse the matched string in the replacement. In the previous snippet we have a named capture of `?<digit>`. This named capture can be reused in the replacement string as `${digit}`. Note that ordinal captures can be used as well, `$123`, which would evaluate to 123 captures in our pattern.

This example illustrates how to use the **Replace** method of **Regex** to remove all digits from the input string.

Example

VB RegexReplace.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Regular Expressions

Match a pattern

Get multiple matches

Make replacements

Find common patterns

Back to Index

Get URL for this page

How Do I...Use common Regular Expressions?

There are a variety of tasks that regular expressions can be used for in regards to input validation and file parsing. Some of the most common expressions are often hard to remember because of the power and options available. Here is a list of commonly used regular expressions:

Use	Expression
Social Security Number	\d{3}-\d{2}-\d{4}
US Phone Number	((\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}
US Postal Code	\d{5}(-\d{4})?
Internet EMail Address	[\w-]+@([\w-]+\.)+[\w-]+
Internet URL	http://([\w-]\.)+[\w-](/[\w-./?%=]*)?
Simple Password (digit)	^(?=.*\d).{4,8}$
Advanced Password (upper, lower, digit)	^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{4,8}$
Common File masks	^(.+)\\(.+)\.(.+)
Major Credit Card	\d{4}-?\d{4}-?\d{4}-?\d{4}

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

Remoting Overview

The common language runtime Remoting infrastructure provides a rich set of classes that allow developers to ignore most of the complexities of deploying and managing remote objects. Even though you are dealing with applications running against different run-time environments, calling methods on remote objects is nearly identical to calling local methods. The purpose of this QuickStart is to provide a brief introduction to Remoting and to describe the principles involved so that you can build more sophisticated applications as needed. A basic understanding of remote objects, HTTP, TCP, and URLs would be helpful for understanding these examples.

Remoting is a framework built into the common language runtime that can be used to build sophisticated distributed applications and network services. Although it has a number of features in common with similar .NET Framework on other platforms, it offers a unique set of features not available anywhere else:

- **Proxy objects.** When a client creates an instance of a remote object, it receives a proxy to the class instance on the server. All methods called on the proxy will automatically be forwarded to the remote class and any results will be returned to the client. From the client's perspective, this process is no different than making a local call. Any exceptions thrown by the remote object will automatically be returned to the client. This enables the client to use normal try and catch blocks around sections of the code to trap and deal with exceptions.

- **Object passing.** All objects created remotely are returned by reference and have to derive from **MarshallByRefObject**. Objects passed as parameters to a remote method call can be forwarded by value or by reference. The default behavior is pass by value provided the object in question is marked by the custom attribute [serializable]. Additionally, the object could implement the **ISerializable** interface, which provides flexibility in how the object should be serialized and deserialized. Objects that are not marshal by reference or marshal by value are not remotable.

- **Activation models.** Remote objects can easily be created from a client by calling **new**. The framework contains enough "intelligence" to realize you are dealing with a remote object and will ensure an instance of the object gets created in the relevant remote application. Creating instances of remote objects is not limited to default constructors; you can even do this using a constructor that requires one or more parameters. The **Activator** class contains two methods, **CreateInstance** and **GetObject**, that can also be used to create an instance of remote objects. The former can be used in place of new to create an object instance while the latter is normally used to connect to an object at a specified URL.

- **Stateless and Stateful objects.** The framework makes a provision for creating remote objects as stateless. When an object is configured as SingleCall, it will be created when a method is called on that object. The object processes the call, returns an optional result, and is then collected by the garbage collector. This way the client is always connected to a fresh object with each call. Configuring an object as a Singleton ensures that all clients will be connected to the same object whenever a call is made to that object. **ClientActivated** objects allow the client to pass parameters to the constructor of a remote object when it gets created. Each activation request for a client activated object

(**Activator.CreateInstance** or new in combination with entries in the configuration file) on the client results in a new object on the server.

- **Channels and Serialization.** When a client calls a method on a remote object, the remoting framework automatically serializes any data associated with the request and transports the data to the remote object using a channel. Some of the more popular channels supported are HTTP, TCP, and SMTP. In the case of HTTP, the framework uses the SOAP protocol to transport data in XML format from the client to the server and back. The default serialization formatter for HTTP is a SOAP formatter. Since programmers can create custom formatters for use with any channel, the remoting framework can be configured to work with any external .NET Framework on other platforms. The TCP channel uses plain sockets and Binary Serialization by default and can be used to communicate with any object on a remote server.
- **Lease based Lifetime.** The lifetime of remote objects is controlled by a leasing mechanism. When an object is first created, it is given a lease time. When the lease time of the object reaches zero, the object will be disconnected from the Remoting infrastructure and when all references to the object has been freed within the **AppDomain**, it will be collected by the garbage collector. A number of mechanisms are provided that allow the client to extend the lease on the object, thereby sustaining its life.
- **Hosting objects in IIS.** Remote objects can easily be hosted in IIS. This allows any client to connect to the object using normal HTTP on port 80. It is also possible to create Web pages that allow a user to connect to a remote object using Internet Explorer.

The remainder of the QuickStart features practical examples of the above concepts.

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

Remoting Overview

The common language runtime Remoting infrastructure provides a rich set of classes that allow developers to ignore most of the complexities of deploying and managing remote objects. Even though you are dealing with applications running against different run-time environments, calling methods on remote objects is nearly identical to calling local methods. The purpose of this QuickStart is to provide a brief introduction to Remoting and to describe the principles involved so that you can build more sophisticated applications as needed. A basic understanding of remote objects, HTTP, TCP, and URLs would be helpful for understanding these examples.

Remoting is a framework built into the common language runtime that can be used to build sophisticated distributed applications and network services. Although it has a number of features in common with similar .NET Framework on other platforms, it offers a unique set of features not available anywhere else:

- **Proxy objects.** When a client creates an instance of a remote object, it receives a proxy to the class instance on the server. All methods called on the proxy will automatically be forwarded to the remote class and any results will be returned to the client. From the client's perspective, this process is no different than making a local call. Any exceptions thrown by the remote object will automatically be returned to the client. This enables the client to use normal try and catch blocks around sections of the code to trap and deal with exceptions.

- **Object passing.** All objects created remotely are returned by reference and have to derive from **MarshallByRefObject**. Objects passed as parameters to a remote method call can be forwarded by value or by reference. The default behavior is pass by value provided the object in question is marked by the custom attribute [serializable]. Additionally, the object could implement the **ISerializable** interface, which provides flexibility in how the object should be serialized and deserialized. Objects that are not marshal by reference or marshal by value are not remotable.

- **Activation models.** Remote objects can easily be created from a client by calling **new**. The framework contains enough "intelligence" to realize you are dealing with a remote object and will ensure an instance of the object gets created in the relevant remote application. Creating instances of remote objects is not limited to default constructors; you can even do this using a constructor that requires one or more parameters. The **Activator** class contains two methods, **CreateInstance** and **GetObject**, that can also be used to create an instance of remote objects. The former can be used in place of new to create an object instance while the latter is normally used to connect to an object at a specified URL.

- **Stateless and Stateful objects.** The framework makes a provision for creating remote objects as stateless. When an object is configured as SingleCall, it will be created when a method is called on that object. The object processes the call, returns an optional result, and is then collected by the garbage collector. This way the client is always connected to a fresh object with each call. Configuring an object as a Singleton ensures that all clients will be connected to the same object whenever a call is made to that object. **ClientActivated** objects allow the client to pass parameters to the constructor of a remote object when it gets created. Each activation request for a client activated object

(**Activator.CreateInstance** or new in combination with entries in the configuration file) on the client results in a new object on the server.

- **Channels and Serialization.** When a client calls a method on a remote object, the remoting framework automatically serializes any data associated with the request and transports the data to the remote object using a channel. Some of the more popular channels supported are HTTP, TCP, and SMTP. In the case of HTTP, the framework uses the SOAP protocol to transport data in XML format from the client to the server and back. The default serialization formatter for HTTP is a SOAP formatter. Since programmers can create custom formatters for use with any channel, the remoting framework can be configured to work with any external .NET Framework on other platforms. The TCP channel uses plain sockets and Binary Serialization by default and can be used to communicate with any object on a remote server.
- **Lease based Lifetime.** The lifetime of remote objects is controlled by a leasing mechanism. When an object is first created, it is given a lease time. When the lease time of the object reaches zero, the object will be disconnected from the Remoting infrastructure and when all references to the object has been freed within the **AppDomain**, it will be collected by the garbage collector. A number of mechanisms are provided that allow the client to extend the lease on the object, thereby sustaining its life.
- **Hosting objects in IIS.** Remote objects can easily be hosted in IIS. This allows any client to connect to the object using normal HTTP on port 80. It is also possible to create Web pages that allow a user to connect to a remote object using Internet Explorer.

The remainder of the QuickStart features practical examples of the above concepts.

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Remoting Works

The .NET Remoting framework provides developers with a modern distributed object model that allows remote method invocation between different common language runtimes across the network or between different **AppDomains** in the same common language runtime. Any interaction with a remote object occurs through a proxy, and a client cannot access a remote object directly since the object is only meaningful inside its own **AppDomain.** The following rules apply:

- All remote objects have to derive from **MarshalByRefObject**.
- Any interaction with a remote object occurs by reference through a proxy. Clients have to obtain this proxy activating a remote object. This can be done by calling **CreateInstance**, **GetObject**, or **new**.
- Local objects can be used as parameters when making remote calls. Since a reference to a local object is meaningless to a remote object, all local objects are passed by value when they are used as parameters in a remote call or returned as a result from a remote call.
- All objects passed by value must be serializable. This can be achieved by implementing the **ISerializable** interface or marking the object with the [serializable] custom attribute.

Remote objects have to be activated before they can be used by a client. The framework supports two activation models: server activation and client activation. Server activation means the object will automatically be created on the server when a client attempts to access the object. Client activation means the object will be created as the result of a deliberate activation request from a client. The reason there are two activation models is that the lifetime of an object often becomes an issue in distributed object models. In some cases, a remote object is required to maintain state across calls. In other cases, a stateless object is required where each caller is given a fresh instance of the object on each call. Sometimes the client requires more control over the lifetime of the object than that provided by server activation. By allowing both server and client activation, developers have the freedom to pick the best activation model for the task at hand.

How can a server object listen for clients when it has not been created? A server object is created and deployed somewhere on the network where it listens for clients to connect. When a client comes along, details about the call and the parameters are transported to the server object, then the method executes, and an optional result is transported back to the client.

The steps for deploying a server object follow:

1. All remote objects have to be registered with the common language runtime before they can be accessed by a client. You need to provide the common language runtime with specific details about the object, such as the name of the assembly that should be loaded to activate the object, the namespace and type name of the object, and the name of the endpoint where the object can be accessed. The framework ensures that an object is not registered more than once in the same **AppDomain.**
2. You also need to register all the channels clients will use to

communicate with the object. Each channel uses a specific protocol such as TCP, HTTP, or SMTP. More than one channel can be registered where applicable.

3. Details about the object are stored in a table managed by the framework. An object reference is then created for object that is also stored. The object reference will be used to represent the registered object in a different **AppDomain** and will be serialized and transported to the client when required.

4. The registered channels then start listening for clients to connect.

Once a remote object has been deployed, clients can connect and invoke methods on the server object.

1. The client registers a channel that will be used to communicate with the remote object. The only requirement here is that the server should be listening on a similar channel.

2. The client then activates the object by calling **new**, **GetObject**, or **CreateInstance.**

3. Sufficient information about the type and location of the remote object is provided with the activation request for the framework to create proxy to represent the remote object. It is important to note that this process is local, and no calls have been made to the remote object at all. The proxy is also stored locally in a table, any additional activation request on the client side receives the same proxy.

4. In the case of server activated object, the client calls a method on the remote object using the proxy it received on activation. The client channel then attempts to establish a connection with the remote channel if not already connected.

5. Once a connection is established, a message sink is connected to either side of the connection that handles the details of the communication using the specified protocol.

6. Details of the method call are loaded in a **Message** object that is serialized and transported to the server. The type of serialization depends on the channel. For example, when the HTTP channel is used, all messages are serialized to XML and transported over SOAP. On the other hand, TCP uses binary serialization.

7. Once the message arrives on the server side, the framework reassembles the original call, activates the target object if it is not already activated, and forwards the call to the object in question. Returning a result back to the client follows the exact opposite path - the result is packaged in a message that is transported back to the client. If server object is of type **SingleCall**, the object is recycled after the result returns to the client.

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

Use the Remoting Samples

The next few sections contain a number of sample programs that demonstrate the most important features of the .NET remoting framework. The examples as concise as possible and in many cases omit error handling and user interfaces in favor of simplicity. The examples are presented in a specific sequence, each one building on the concepts illustrated in the previous one (where applicable). Each example consists of at least two executables called server.exe and client.exe. Some examples contain more than one client that works with the same server.

The three different ways client references to remote objects are resolved at compile time are:

- Compile the server object and specify the EXE or DLL as a reference to the compiler when compiling the client. This is useful when both the client and server components are developed at the same site.
- Derive the server object from an interface class and compile the client with the interface. This is useful when the client and server components are not developed at the same site. The interface(s) can be compiled to a DLL and shipped to the client sites as necessary. Changes to the published interface should be avoided. Refer to the COM guidelines for more information.
- Use the SUDS tool to extract the required metadata from a running server object. This is useful when the client and server components are developed at different sites and no interface classes are available. Point the SUDS tool at a remote URI and generate the required metadata as source or a DLL. Note that the SUDS tool only extracts metadata, and will not generate the source for the remote object.

The majority of the samples presented in this QuickStart compile the server object to a DLL and use this as a reference to compile the server and client executables. In cases where a common class is referenced by both the client and the server, the common class was added to the server object DLL. The sample installation program automatically builds all the samples on the CD using the MAKEFILES provided. The server object is compiled as follows and produces the file share.dll or object.dll.

```
vbc /t:library /debug /r:System.Runtime.Remoting.dll share.vb
```

VB

The server is compiled as follows to produce the file server.exe.

```
vbc /debug /r:Share.dll /r:System.Runtime.Remoting.dll server.vb
```

VB

The client is compiled as follows to produce the file client.exe.

```
vbc /debug /r:Share.dll /r:System.Runtime.Remoting.dll client.vb
```

VB

How to run the samples

Open two command windows and find the Samples\Quickstart\Remoting directory. Start the server in one command window and run the client in the other. Make sure that the server app is closed down before moving on to the next sample. This step is important since most of the servers use TCP port 8085 and any attempt to run more than one server at a time will generate an error.

Getting more information

The samples included with the .NET Framework SDK contain additional remoting examples that demonstrate features not covered in this QuickStart.

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Create a Remote Server?

This sample demonstrates how to create the remoting version of a Hello World server. When a client calls the **HelloMethod** on the **HelloServer** class, the server object appends the string passed from the client to "Hi There" and returns the resulting string back to the client. The following code example demonstrates both the server object and the server application.

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.TCP

Namespace RemotingSamples
    Public Class Sample
        Shared Sub Main
            Dim chan As TCPChannel
            chan = New TCPChannel(8085)
            ChannelServices.RegisterChannel(chan)
            RemotingConfiguration.RegisterWellKnownServiceType( _
                Type.GetType("RemotingSamples.HelloServer, object"), _
                "SayHello", WellKnownObjectMode.SingleCall)
            System.Console.WriteLine("Hit  to exit...")
            System.Console.ReadLine()
        End Sub
    End Class
End Namespace
```

VB

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.TCP

Namespace RemotingSamples
    Public Class HelloServer : Inherits MarshalByRefObject
        Public Sub HelloServer()
            Console.WriteLine("HelloServer activated")
        End Sub

        Public Function HelloMethod(name As String) As String
            Console.WriteLine("Hello.HelloMethod : {0}", name)
            HelloMethod = "Hi there " + name
        End Function
    End Class
End Namespace
```

VB

The class **HelloServer** is derived from **MarshalByRefObject** to make it remotable. When the server is started, you need to create and register a TCP channel that will listen for clients to connect on port 8085. You also need to register the remote object with the remoting framework by calling **RegisterWellKnownType**. The parameters for this call are as follows:

1. The name of the assembly that contains the object being registered. Since the object is contained in an executable, you provide the name of the executable here. Both server and server.exe will work since the normal rules of locating an assembly apply.

2. The next parameter is the full type name of the object being registered, in this case "**RemotingSamples.HelloServer**". You need to specify both the name of the namespace and the classname here. If you don't use namespaces, you pass the classname.

3. Then, you provide the name of the endpoint where the object will be published. Clients need to know this name in order to connect to the object. Any string will work; for this example, use **"SayHello"**. It is also possible to connect to remote objects using ASP.NET and if this is a requirement, the name of the endpoint should be namespace/class.soap. For this example, the endpoint will be RemotingSamples/HelloServer.soap.

4. The final parameter specifies the object mode, which can be **SingleCall** or **Singleton**. For this example, you specify **SingleCall**. The object mode specifies the lifetime of the object when it is activated on the server. In the case of **SingleCall** objects, a new instance of the class will be created for each call made from a client, even if the same client calls the same method more than once. On the other hand, **Singleton** objects are created once only and all clients communicate with the same object.

Now you can compile this example and execute it.

```
csc /debug+ /r:System.Runtime.Remoting.dll server.cs
```

C#

When you start the server, the object will be created as part of the registration process so the framework can extract the relevant metadata from the object. After registration, this object is destroyed and the framework starts listening for clients to connect on the registered channels.

VB Hello Server Sample

[Run Sample] | [View Source]

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Create a Client of a Remote Server?

If you haven't already read the section How Do I Create a Remote Server?, please read this first. The client references the server assembly for metadata, so you have to compile the server before the client. The code for the client is listed below.

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.TCP

Namespace RemotingSamples
    Public Class Client
        Shared Sub Main
            Dim chan As TCPChannel
            chan = New TCPChannel()
            ChannelServices.RegisterChannel(chan)
            Dim obj As HelloServer
            Obj = CType(Activator.GetObject( _
                Type.GetType("RemotingSamples.HelloServer, object"), _
                    "tcp://localhost:8085/SayHello"), _
                HelloServer)
            If obj Is Nothing Then
                System.Console.WriteLine("Could not locate server")
            Else
                Console.WriteLine(obj.HelloMethod("Caveman"))
            End If
        End Sub
    End Class
End Namespace
```

VB

When the client starts up, it registers a TCP channel and proceeds to activate the object by calling the **GetObject** method on the **Activator** class. The parameters for this call are as follows:

1. The type of the name of the class you need to activate, **RemotingSamples.HelloServer**.

2. Specify the URI of the object you need to activate. For this client the URI is **tcp://localhost:8085/SayHello**. Note that the URI includes the protocol, machine name, port number, and the endpoint. If the server is deployed on a host named Sunshine, clients can connect to the server by specifying **tcp://sunshine:8085/SayHello**.

When you run the client, it locates and connects to the server, retrieves a proxy for the remote object, and calls the **HelloMethod** on the remote object, passing the string "Caveman" as a parameter. The server returns "Hi there Caveman".

NOTE: All TCP channels use binary serialization when transporting local objects to and from a remote object.

VB Hello Client Sample

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Create a Remote Object as a Singleton?

This example demonstrates how to modify the Client/Server example by deploying the remote object as a singleton. The client starts two threads and calls the **CountMe** method on the server on two different channels: TCP and HTTP. Although this example is very simple, it does illustrate a few important concepts.

- The **HelloServer** object is only activated once and its destructor is never called.
- When you run the client, thread number two appears to outperform thread number one. The second thread is indeed faster than the first since thread one uses the HTTP channel that transports data in XML using SOAP while TCP uses a binary transport. While TCP clearly outperforms HTTP, the latter is more general and should be used when communicating with unknown endpoints.
- The server object is locked while the counter is incremented in the **CountMe** call.
- Notice that the count returned by the server is not reset between different invocations of the client. This behavior demonstrates that the server does preserve its state during its lifetime.
- Each client thread makes a call to the server and then yields control to its partner to ensure they both get a chance to run.
- Each thread registers its own channel and connects to the server using different URIs. These are **"http://localhost:8086/SayHello"** for thread one and **"tcp://localhost:8085/SayHello"** for thread two.
- The client executable waits for both threads to finish before terminating.

NOTE: Do not specify a port number when registering the client channels. The framework uses the port number on the channels to determine if the channel should listen or connect. When the channel is registered with a port number, the channel listens. If the port number is zero, it attempts to connect to a remote channel at the specified URI.

VB Singleton Sample

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Pass An Object to a Server By Reference?

This example demonstrates how to create an object that derives from **MarshalByRefObject** on the client that is then passed as a parameter to the server. The server calls a method on the remote object it receives. The first step is to create the object you need to pass.

```
Imports System
Namespace RemotingSamples

  Public Class ForwardMe : Inherits MarshalByRefObject

        Sub CallMe(String text)
            Console.WriteLine(text)
        End Sub
    End Class
End Namespace
```

VB

Since the compiler requires the metadata for this object for both the client and server applications, you will compile the class and create a separate assembly for it.

```
vbc /debug+ /target:library /out:share.dll share.vb
```

VB

You can reuse the hello example, but modify the **HelloMethod** to accept two parameters. The first parameter is the string you had before and the second parameter is the new object, **ForwardMe**. The server calls the **CallMe** method on the object it receives before returning the "hello" string to the client.

On the client side, you can create an instance of **ForwardMe** and pass it as a parameter when you call **HelloMethod.** The only catch in this example is the way you create the TCP channel - you have to give it a port number. This port number should be different from the one you are already using to communicate with the server. When the client channel is registered, the framework will start listening on this port for clients to connect, and establish a connection with the remote object. This allows bidirectional communication between the client and the server where the client forwards parameters to the server on one port and the server forwards parameters to the client using the other port.

When you run the example, the client should print the text **"Regards from the server"** to the console window.

VB Passing by Reference

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Pass An Object to a Server By Value?

The Pass an Object to a Server by Reference section illustrated that local objects are always passed by value when you call a remote function. To demonstrate this concept, you need change the previous example.

The first step is to create the object you need to pass to the server.

```
Imports System
Namespace RemotingSamples

     Public Class ForwardMe
        Dim mValue As Integer = 1

        Public Sub CallMe()
            mValue += 1
        End Sub

        Public Function getValue() As Integer
            Return mValue
        End Function
    End Class
End Namespace
```

VB

This class is flagged with the [serializable] custom attribute, which allows it to be streamed to and from the server. When you run the sample, the client reports the value of the counter to be one. Since the server calls the **CallMe** method on the object five times, the value of the counter when the object is returned is six. Note that the client has two instances of the **ForwardMe** class after calling the **remote** method: the instance that was passed and a copy of the instance that was returned.

VB Passing by Value

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Compile a Client Against an Interface?

This example illustrates how to build a client that does not reference a server object at compile time. The following code example demonstrates the client code.

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.TCP

Namespace RemotingSamples
    Public Class Client
        Shared Sub Main
            Dim chan As TCPChannel
            chan = New TCPChannel()
            ChannelServices.RegisterChannel(chan)
            Dim obj As IHello
            obj = CType(Activator.GetObject(Type.GetType("RemotingSamples.IHello,share"),
"tcp://localhost:8085/SayHello"),IHello)
            If obj Is Nothing
                System.Console.WriteLine("Could not locate server")
            Else
                    Console.WriteLine(obj.HelloMethod("Caveman"))
            End If
        End Sub
    End Class
End Namespace
```

VB

The client calls **GetObject** on an endpoint without knowing what the exact object type is. All it knows is that the object implements **IHello.**

VB Interface

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Remoting
Remoting Overview
How Remoting Works
Remoting Samples

The Basics
Create a Remote Server
Create a Client of the Server

Samples
Create a Singleton Object
Pass Objects By Reference
Pass Objects By Value
Compile Against an Interface
Make an Asynchronous Call

Back to Index

Get URL for this page

How Do I...Make an Asynchronous Call to a Remote Object?

All the examples up to this point made synchronous calls to the remote object. This strategy might not always be desirable since the remote object might have to perform a number of time-consuming tasks and it is not advisable to block the client while a call is in progress. This example demonstrates how to make asynchronous calls.

Reuse the server from the hello sample and modify the client to make asynchronous calls. The following code example demonstrates the new client code.

```
Imports System
Imports System.Threading
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Messaging
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.TCP

Namespace RemotingSamples
    Public Class Client
        Public Shared e As ManualResetEvent
        Public Delegate Function MyDelegate(name As String) As String

        Shared Sub Main
            e = New ManualResetEvent(false)
            Dim chan As TCPChannel
            chan = New TCPChannel()
            ChannelServices.RegisterChannel(chan)
            dim obj As HelloServer
            obj = CType(Activator.GetObject( _
                Type.GetType("RemotingSamples.HelloServer,Object"), _
                    "tcp://localhost:8085/SayHello"), _
                HelloServer)
            If obj Is Nothing Then
                System.Console.WriteLine("Could not locate server")
            Else
                Dim cb As AsyncCallback
                    cb = New AsyncCallback(AddressOf Client.MyCallBack)
                Dim d As MyDelegate
                    d = New MyDelegate(AddressOf obj.HelloMethod)
                Dim ar As IAsyncResult
                ar = d.BeginInvoke("Caveman", cb, 0)
            End If

            e.WaitOne()
        End Sub

        Shared Sub MyCallBack(ar As IAsyncResult)
            Dim d As MyDelegate = CType(ar, AsyncResult).AsyncDelegate
            Console.WriteLine(d.EndInvoke("Caveman", ar))
            e.Set()
        End Sub
    End Class
End Namespace
```

VB

An **Event** object is used to prevent the client application from returning from **Main** while the asynchronous call is pending. You need to reset the **Event** object to false on the first line in **Main** and wait for state of the object to change before leaving **Main.** The async programming pattern in the .NET Framework requires a **delegate** to represent the callback function. You need to declare the delegate that is required before it can be used. Delegates are somewhat similar to function pointers on classes in C++. You have to declare a delegate with the same calling parameters and result type as the method the delegate represents. Since you need to define a delegate for the **HelloMethod** you want to call, declare the delegate as follows:

```
Public Delegate Function MyDelegate(name As String) As String
```

VB

The call accepts a string as argument and returns a string as the result. The compiler automatically generates the class **MyDelegate** when it encounters the declaration and adds a **BeginInvoke** and **EndInvoke** method to the delegate that maps to native calls somewhere in the common language runtime. The next is to create the callback function that receives the result from the call.

```
Shared Sub MyCallBack(ar As IAsyncResult)
    Dim d As MyDelegate = CType(ar, AsyncResult).AsyncDelegate
    Console.WriteLine(d.EndInvoke("Caveman", ar))
    e.Set()
End Sub
```

VB

Notice how the callback declares an object of type **IAsyncResult** as the parameter of the callback function. Once the call completes, the framework ensures that the result of the call is placed inside the result object and the callback is then invoked back to the client, forwarding the result object to it. To retrieve the result of the call, extract the delegate from the **AsyncResult** and call **EndInvoke.** The following code example demonstrates calling the **HelloMethod** on the remote object.

```
Dim cb As AsyncCallback
cb = New AsyncCallback(AddressOf Client.MyCallBack)
Dim d As MyDelegate
d = New MyDelegate(AddressOf obj.HelloMethod)
Dim ar As IAsyncResult
ar = d.BeginInvoke("Caveman", cb, 0)
```

VB

Create a delegate for the **callback** method and one for the **remote** method. Next, call the method by calling **BeginInvoke** on the delegate, and wait for the result to return from the server.

VB Acync calls

LateBreaking Samples:

- C++

[Run Sample] | [View Source]

Resources
Create resources
Use resources
Read and write resources
Make a ResourceReader

Back to Index

Get URL for this page

How Do I...Create Resources?

Creating resources can help you develop robust, culture-aware programs without having to recompile your application because the resources have changed. The steps for creating and subsequently using resources are covered in this topic.

VB CreateResources.aspx

[Run Sample] | [View Source]

Resources are an application-building feature that allow you to place culture-specific items inside satellite files, rather than directly in your main application. When you build your application, you can identify aspects that are culture-specific, and make a different resources file for each culture where you think your application may be used. At run time, the appropriate set of resources will be loaded, based on the user's culture settings. The specific setting used is the **CurrentUICulture** for your Thread, which the user can set programmatically.

When you make your application, it is housed in an 'assembly'. An assembly consists of the following:

- The assembly metadata, also called the assembly manifest
- Type metadata
- The Microsoft intermediate language (MSIL) code that implements the types
- A set of resources

The assembly manifest is the only required item from the list, but either types or resources are needed to give the assembly any meaningful functionality. The set of resources is completely optional. When you have created your application, it is part of an assembly, which is referred to as the 'main' assembly, since it is the central part of your application. Any code aspects of your application are part of this main assembly. If you change any code in this main assembly, you need to recompile the entire assembly, which can mean a lot of work redistributing your application.

When you make external sets of resources for your application to use (that is, resources that are not a part of your main assembly), these are referred to as 'satellite assemblies'. They are also compiled, and they have their own assembly. If you ever need to change information in a satellite assembly, you can do so without having to change the information in your main assembly, which saves time for you and your users. Satellite assemblies can consist of resources only, therefore they have no code. Satellite assemblies will be compiled as dll's and will be referenced by your main assembly, although this is done for you by the ResourceManager: you do not need any specific code to tell your program how to reference the satellite assemblies.

So how do you decide what goes in a satellite assembly? Imagine you are designing an application that may be used in many different cultures. If the strings are hard-coded and messages appear as 'welcome to our application', or 'exception: cannot divide by zero', you need to take into consideration the fact that other cultures won't be able to read the messages. Furthermore, if you decide to change any of these messages, which are part of the main assembly, you will need to recompile the application and redistribute it with every change you make. Therefore, all the information that may change with culture (such as message strings or graphics) was placed in resource files. This specifically includes things such as prompts and exception messages, or icons and bitmaps that may not be interpretable for all users if they are only included in (for example) English.

The first step is to identify all the language or culture-specific string message items and place them in a text file. If you have pictures or other data that may need to be culture-specific, note that also. Here is an example of the kind of things we would include in our file, called strings.txt. The name 'strings' becomes the rootname for our resource, which is important in retrieving the resources later. The name is arbitrary, but strings makes sense for a file of string entries.

```
;the welcome statement we want to make...
welcome = Welcome to our application!

;the prompts for the main screen
PromptName = Please enter your name ==>
PromptState = Please enter the State you live in ==>

;the exception message
NameErr = You must specify your last name
AddressErr = Please ensure you enter the address field

; etc...
```

The lines preceded with a semicolon are comments in this file, and are for your own use to help clarify how you have organized the information. They have no final impact on how the information is read or interpreted. Each actual entry in the file consists of a reference name (the item preceding the equals sign) and then the string you want to present (following the equals sign). When you make your first file, you should create entries for the primary culture that this information will be viewed in. In this demonstration, assume that the application will be used primarily by U.S. English users, so make your first set of entries for that particular culture.

After you create the file, you can make other similar files, one for each culture you want your application to be presented in. You can do this at any point, providing users with specific culture support at a later date (even after you ship your product), allowing you to focus on the development of your core application. When you do make files for other cultures, make sure you give the culture-specific files the same name as the base file, only differentiated by embedding the name of the culture inside the name of the file (see the sample below).

It is a good idea to organize the files in subfolders (this helps with the next steps). In this example, you will design the application for U.S., New Zealand, and German usage. Consequently, the primary (strings.txt) file is the default file (in the same location where main assembly will reside), but we have also made two other files, and organized them in the application directory as in the following diagram.

```
app directory
      |
      ----->   strings.txt              default text resources
      ----->   myApp.cs  or  myApp.vb    our application source file (C# or VB)
      |
      ----->   en-NZ                  New Zealand directory
      |          |
      |          -----> strings.en-NZ.txt    New Zealand text resources,
      |                                      culture embedded in filename
      |
      ----->   de                   German directory
                 |
                 -----> strings.de.txt       German text resources,
                                             culture embedded in filename
```

You may be wondering where the names of the subdirectories come from. Each major language is given a two letter acronym in lowercase, such as en for English or bg for Bulgarian. For subcultures or other regions, a second, capitalized acronym is developed. This allows you to identify culture and region specific combinations such as en-CA for Canadian English, or fr-CA for Canadian French. In the example above, de indicates German (regardless of the sub-locale). It is important that these directories are correctly named, since they will be searched later for resources. If they are not named correctly, then your resources will not be found by the system.

Now that you have the directory structure, the next step is to actually create the resource files. Up to this point, you have made human-readable text files, but these need to be translated into files that can be used by your application. This can be done through code (see How Do I...Read and Write Resources), but in this example, the simplest solution is to use an application called Resgen (Resource Generator). Resgen can take .txt files and turn them into .resources files, which can then be used for the next steps. To do this, call Resgen from a command prompt, specifying the .txt file you want to turn into a .resources file, as in this example. You can specify a different name for the output file if you want, as the second parameter to the Resgen utility.

```
;from inside your app directory
;output file is strings.resources, in this directory
resgen strings.txt


;from inside the en-NZ directory
;output file is strings.en-NZ.resources, in this directory
resgen strings.en-NZ.txt
```

The Resgen utility can convert resource files back into text files. Note however that you lose any comments when converting this way (the comments were already stripped as unnecessary information when making the .resources file). If you do convert this way, you must give Resgen the name of an output file; it will not work if you specify the input file only.

Resgen can also convert information into and out of another intermediary format called ResX. The ResX format is an XML-based format that wraps the name-value pairs you place in the .txt file into XML instructions. The advantage a .ResX file gives you over a .txt file is that you can store object structures inside a ResX file, so long as the object is serializable. Remember that resources can contain string information, but they can also contain objects, if this makes sense for your resource-aware application. The kind of objects you might want to store include fonts, or actual picture images such as bitmaps (a typical example is the ability to show a particular nation's flag based on the culture setting used).

As with .txt files, Resgen can be used to convert .ResX files into and out of resource files. The following sample demonstrates converting ResX into resources, and resources back into .txt, or .ResX. Note that once you have a resources file, it does not matter how it was constructed: it can be converted back into either .ResX or .txt files. In addition, .txt and .resx files can be converted into each other directly if required.

```
;from inside your app directory
;convert a .ResX file into a resources file
resgen strings.ResX

;now, convert the resources file just made (strings.resources), back into a .txt file
;the second parameter is the output file. Note that the extensions of the files
;must be correct for resgen to function properly
resgen strings.resources strings.txt

;now, convert the same strings.resources file back into a ResX file
;how the resources file was constructed does not matter.
resgen strings.resources strings.ResX
```

```
;you can also convert .txt or .resx files to and from one another
resgen strings.txt strings.ResX
resgen strings.ResX strings.txt
```

Note that because a ResX file can contain embedded objects, but a .txt file cannot, if you Resgen a ResX file into a .txt file and the ResX file has embedded objects (or if you convert a resources file with objects in it into a .txt file), the objects cannot be preserved. Instead, the datatype of the object is written to the .txt file (to give you some kind of feedback). You will be informed by Resgen that an error occurred, and all string resources will be correctly created.

The following sample is taken directly from a ResX file. You can see in this sample that an image has been embedded directly into the ResX file, which is not possible in a .txt file. The first section is labeled as [header] (this is not allowed in an actual Resx file) so you can see how a ResX file begins. Subsequent [entries] are the name value combinations for strings or objects. The XML code reflects this by having a data structure that defines a name-value pair, just as a .txt file would. The following code example demonstrates using ResX files.

```
[header]




...
[entries: strings]

    Welcome to our application!

    Please enter your name ==>

    Please enter the State you live in ==>
[entries: objects. This is an actual embedded image]

    AAEAAAD/////AQAAAAAAAAAMAgAAADtTeX    ...
...
```

Having constructed the resources, you can now move on to the next step: compiling your main assembly. This consists of compiling your assembly as you would normally, with an extra /res: switch in order to place your primary resources file into your main assembly. This is a very critical step, because this becomes your fallback resource, if others can't be found.

The following rules are used for looking up resources. Suppose someone is trying to access a Mexican Spanish (es-MX) resource inside your application. When your program is running, it has a standard process for looking for any Spanish resources, which includes taking the following steps:

- The global assembly cache is checked to see if it includes an es-MX assembly for your application. The global assembly cache is an area that allows you to reference shared resources for many applications, rather than having to include many copies of the same set of resources in the directory structure of every application you make. If the assembly is found, then that assembly is searched for the specific resource requested, and if found, the requested resource is used. If not found, then the search continues through the next steps.
- The directory of the currently executing assembly is checked to see if it contains an es-MX directory. If it does, that directory is searched to see if it has a valid satellite assembly for the requested culture. That assembly is then searched for the specific resource requested and if found, the requested resource is used. If not found or the assembly did not exist, then the search continues through the next steps.
- The global assembly cache is searched for the parent assembly of the requested resource, for your application. The parent is defined as the appropriate fallback culture. In this case, es-MX (Mexican Spanish) will fallback to es (Spanish). Note that providing any resource is viewed as a better alternative to throwing an exception, which is why this system is adopted. Parents are viewed as 'best-fit' candidates. This process also allows scope for reuse of resources: you need only include a particular resource at the parent level, if the child culture does not need to localize the requested resource. If the parent assembly exists in the global cache, then that assembly is searched for the specific resource requested and if found, the requested resource is used. If it is not found, then the search continues through the next steps.
- The directory of the currently executing assembly is checked to see if it contains a parent directory for es-MX (the parent directory for es-MX is es). If the parent directory exists, that directory is searched to see if it has a valid satellite assembly for the parent culture. If it does, then that assembly is searched for the specific resource requested and if found, the requested resource is used. If not found, then the search continues through the next steps.
- Parent assemblies are continually searched as in the previous step through many potential levels (each culture only has one parent, but a parent may have another parent). After all the original culture and parents have been searched for the resource and it is still not found, then the default is used.
- The default resource is that resource that is included with the main assembly when it is compiled (strings.txt). The default resource is viewed as the absolute fallback (the final parent), and it is strongly suggested that a default set of resources always be included. This helps ensure that exceptions are not thrown. Using resources should not result in exceptions.
- If the resource has not been found, then an exception is thrown indicating that the resource could not be found.

The process for trying to find an appropriate culture without throwing an exception is quite involved. In addition, different locations are searched when trying to find the specified culture. Including a default resource file simply ensures that at least some reasonable value will be found, without causing an exception to be thrown. In most cases, you will need to provide a full set of resources for each localized culture you are providing support for. A missing resources file should never be enough of a cause to throw exceptions back to the user. Your default resource therefore needs to include an entry for all potential resource searches.

Now that you know why a default resources file is important, remember that your default should be that which will be used by most users. To embed it in your main assembly, use the following command-line instruction.

```
' This line makes your main assembly, and links the resources file to it
' The /t: switch specifies the output type (executable, not .dll)
' The /res: switch specifies the resources file to embed
' The final argument is the main source file used to construct our assembly
' This code is run from your main application directory

vbc /t:exe /res:strings.resources myApp.vb
```

VB

You could run the application now, using the default resources. Hopefully, this will satisfy most of your users (who should be requesting the default resources directly); however, now you need to setup your satellite resources. There are few things to note about these other resources before continuing:

- If you want to replace the satellite resources (satellite assemblies once compiled), you can do so at any time, without having to recompile your main assembly
- Remember that satellite assemblies consist of resources only (no code is permitted)
- The resources in the assembly must match those expected as indicated by the name of the file, and the location of the file. Do not create a jp-JP resource (Japanese) and place it in an en (English) directory. It will be ignored.

For each subculture you want to compile as a separate satellite assembly, rename the file as ... The following sample demonstrates how to compile your resources file. Use the Assembly Linker Utility (al), which is a stripped down version of the main compiler. Note that you could use the main compiler if you want, but the Assembly Linker Utility works just as well, and is easier to ship with products.

```
;move into the en-NZ directory (not essential, but it makes things easier)
;ensure your strings file has the appropriate culture name in it;
;For New Zealand, the file should be named strings.en-NZ.txt

;use resgen to generate the resources file from the txt file
the following command will create the file strings.en-NZ.resources
resgen strings.en-NZ.txt

;- Invoke this command-line inside the en-NZ directory
;- The /t: switch specifies the type of the output file; a dll
;- The /embed: switch specifies the filename to include
;  inside the satellite assembly being created
;- The /out: switch specifies the name of the output file
;- It needs to be named according to the following pattern:
;  .resources.dll

al /t:lib /culture:en-NZ /embed:strings.en-NZ.resources /out:myapp.resources.dll

;complete the above steps for each satellite assembly you wish to create,
; changing the culture name appropriately.
```

Note the name of the output file in the above sample. This is critical, because if the files are not named correctly, they will not be found (this is a commonly mistaken step). You need to make sure that the name of the satellite assembly (the dll file) is .resources.dll. Also ensure that you specify the correct culture switch. Without the culture tag, the resources you request cannot be correctly identified.

You may be asking how you get to install resources into the global assembly cache, which we mentioned earlier. Storing resource files in the global assembly cache allows them to be used by multiple applications, without having to store the same resources with many different assemblies. When you use the al utility, you can also use the /install: switch, which allows you to specify the name of your output assembly file. When you use this switch, the resource is saved into the global assembly cache.

At this point, you know the steps needed to create a console application. The critical files and directories are highlighted in the following sample. Although the rest have been helpful to the process, they should be retained by you and not shipped with your product. They can be useful for making changes and creating updates. They are included in the following diagram to remind you what files you will have in your development area. Your directory tree structure will look as follows:

```
app directory
      |
      ----->   strings.txt
      ----->   strings.resources
```

```
-----> myApp.cs   or   myApp.vb   (C# or VB source file)
-----> myApp.exe                  the actual application
|                                 (strings.resources is embedded in this file)
|
-----> en-NZ
|        |
|        -----> strings.en-NZ.txt
|        -----> myApp.resources.dll    The satellite assembly for New Zealand
|
-----> de
         |
         -----> strings.de.txt
         -----> myApp.resources.dll    The satellite assembly for German
```

It is useful to know that there is another format for saving resources called ResX. ResX files are a resource equivalent to .txt files and are comprised of XML code, but can store objects as well as strings. They are generally generated through code (for more information, see How Do I... Use resources). An object such as an image is actually embedded into the ResX file as an equivalent form of binary information. You can only store serializable objects in ResX files, which are by definition available to be used in resource files. Once your .resx file has been created, you can use the Resgen utility on it to create a resources file, just as you would with a .txt file. Note that if you get errors when attempting to use the Resgen utility on a .resx file, a likely cause is that you are attempting to use non-serializable objects.

Up to this point, the discussion has focused on setting up your resources for use with a console application. If you are developing other applications however, such as an ASP.NET application, you may want to set up your resources differently. An example of this different setup is portrayed in the next diagram. Note that the code inside the application changes a little in order to be able to handle the fact that this application uses a different location for its resource files. In this approach, you can store all of your resources files in the **same directory**. The location of this shared directory is not critical, although the suggested location is a subdirectory of your main application. You do not need to differentiate them by their directory structure (although they do still need to be differentiated by name).

You do not compile an aspx page as such, and therefore, it is difficult to identify the assembly for an aspx application. Because of this, you cannot 'embed' your default resource file into your main assembly: it too must be placed in the shared resources directory you create.

up to this point the resources have been assembled into satellite assemblies. However, this example demonstrates that you do not need to actually compile your resources into libraries: you can leave them as .resources files and they are still deployable. Therefore, remember that you only need to generate your resources in this approach; you do not need to manually compile either your main aspx page, or your resources.

```
app directory
      |
      -----> myApp.aspx            an ASP.NET application. You cannot embed a
      |                            resources file into this application!
      |                            The aspx file knows to look in the single resources
      |                            directory for the files.
      |
      -----> resources
                |
                -----> strings.txt
                -----> strings.de.txt
                -----> strings.en-NZ.txt
                |
                -----> strings.resources         the default resources file
                |                                You can tell it is the default, because
                |                                it does not have a specific culture embedded
                |                                in it's name
                -----> strings.en-NZ.resources   the New Zealand resources file
                -----> strings.de.resources   the German resources file

Note: There are no dlls: therefore, there are no satellite assemblies.
      The resources are loaded directly from the resources files.
      This is NOT the suggested approach, but it is viable.
```

The following sample demonstrates how to set up your aspx application. Note that when selecting a site for this application, you must ensure that it is in a Web directory so that the aspx control can behave appropriately.

VB CreateAspxRes.aspx

[Run Sample] | [View Source]

A second approach to using resources in ASP.NET pages is also available. You can actually use satellite assemblies (rather than referencing resources files directly), when you are creating ASP.NET Controls. For more information, see Working with resource files.

You have now finished setting up your resource-aware applications. You can modify the satellite assemblies and resources files without requiring to recompile and distribute your main assembly or application, and you can even add in completely new satellite assemblies, although you may need to modify your code to recognize them.

Resources
Create resources
Use resources
Read and write resources
Make a ResourceReader

Back to Index

Get URL for this page

How Do I...Create Resources?

Creating resources can help you develop robust, culture-aware programs without having to recompile your application because the resources have changed. The steps for creating and subsequently using resources are covered in this topic.

VB CreateResources.aspx

[Run Sample] | [View Source]

Resources are an application-building feature that allow you to place culture-specific items inside satellite files, rather than directly in your main application. When you build your application, you can identify aspects that are culture-specific, and make a different resources file for each culture where you think your application may be used. At run time, the appropriate set of resources will be loaded, based on the user's culture settings. The specific setting used is the **CurrentUICulture** for your Thread, which the user can set programmatically.

When you make your application, it is housed in an 'assembly'. An assembly consists of the following:

- The assembly metadata, also called the assembly manifest
- Type metadata
- The Microsoft intermediate language (MSIL) code that implements the types
- A set of resources

The assembly manifest is the only required item from the list, but either types or resources are needed to give the assembly any meaningful functionality. The set of resources is completely optional. When you have created your application, it is part of an assembly, which is referred to as the 'main' assembly, since it is the central part of your application. Any code aspects of your application are part of this main assembly. If you change any code in this main assembly, you need to recompile the entire assembly, which can mean a lot of work redistributing your application.

When you make external sets of resources for your application to use (that is, resources that are not a part of your main assembly), these are referred to as 'satellite assemblies'. They are also compiled, and they have their own assembly. If you ever need to change information in a satellite assembly, you can do so without having to change the information in your main assembly, which saves time for you and your users. Satellite assemblies can consist of resources only, therefore they have no code. Satellite assemblies will be compiled as dll's and will be referenced by your main assembly, although this is done for you by the ResourceManager: you do not need any specific code to tell your program how to reference the satellite assemblies.

So how do you decide what goes in a satellite assembly? Imagine you are designing an application that may be used in many different cultures. If the strings are hard-coded and messages appear as 'welcome to our application', or 'exception: cannot divide by zero', you need to take into consideration the fact that other cultures won't be able to read the messages. Furthermore, if you decide to change any of these messages, which are part of the main assembly, you will need to recompile the application and redistribute it with every change you make. Therefore, all the information that may change with culture (such as message strings or graphics) was placed in resource files. This specifically includes things such as prompts and exception messages, or icons and bitmaps that may not be interpretable for all users if they are only included in (for example) English.

The first step is to identify all the language or culture-specific string message items and place them in a text file. If you have pictures or other data that may need to be culture-specific, note that also. Here is an example of the kind of things we would include in our file, called strings.txt. The name 'strings' becomes the rootname for our resource, which is important in retrieving the resources later. The name is arbitrary, but strings makes sense for a file of string entries.

```
;the welcome statement we want to make...
welcome = Welcome to our application!

;the prompts for the main screen
PromptName = Please enter your name ==>
PromptState = Please enter the State you live in ==>

;the exception message
NameErr = You must specify your last name
AddressErr = Please ensure you enter the address field

; etc...
```

The lines preceded with a semicolon are comments in this file, and are for your own use to help clarify how you have organized the information. They have no final impact on how the information is read or interpreted. Each actual entry in the file consists of a reference name (the item preceding the equals sign) and then the string you want to present (following the equals sign). When you make your first file, you should create entries for the primary culture that this information will be viewed in. In this demonstration, assume that the application will be used primarily by U.S. English users, so make your first set of entries for that particular culture.

After you create the file, you can make other similar files, one for each culture you want your application to be presented in. You can do this at any point, providing users with specific culture support at a later date (even after you ship your product), allowing you to focus on the development of your core application. When you do make files for other cultures, make sure you give the culture-specific files the same name as the base file, only differentiated by embedding the name of the culture inside the name of the file (see the sample below).

It is a good idea to organize the files in subfolders (this helps with the next steps). In this example, you will design the application for U.S., New Zealand, and German usage. Consequently, the primary (strings.txt) file is the default file (in the same location where main assembly will reside), but we have also made two other files, and organized them in the application directory as in the following diagram.

```
app directory
      |
      ----->   strings.txt              default text resources
      ----->   myApp.cs  or  myApp.vb    our application source file (C# or VB)
      |
      ----->   en-NZ                  New Zealand directory
      |          |
      |          -----> strings.en-NZ.txt    New Zealand text resources,
      |                                      culture embedded in filename
      |
      ----->   de                   German directory
                 |
                 -----> strings.de.txt       German text resources,
                                             culture embedded in filename
```

You may be wondering where the names of the subdirectories come from. Each major language is given a two letter acronym in lowercase, such as en for English or bg for Bulgarian. For subcultures or other regions, a second, capitalized acronym is developed. This allows you to identify culture and region specific combinations such as en-CA for Canadian English, or fr-CA for Canadian French. In the example above, de indicates German (regardless of the sub-locale). It is important that these directories are correctly named, since they will be searched later for resources. If they are not named correctly, then your resources will not be found by the system.

Now that you have the directory structure, the next step is to actually create the resource files. Up to this point, you have made human-readable text files, but these need to be translated into files that can be used by your application. This can be done through code (see How Do I...Read and Write Resources), but in this example, the simplest solution is to use an application called Resgen (Resource Generator). Resgen can take .txt files and turn them into .resources files, which can then be used for the next steps. To do this, call Resgen from a command prompt, specifying the .txt file you want to turn into a .resources file, as in this example. You can specify a different name for the output file if you want, as the second parameter to the Resgen utility.

```
;from inside your app directory
;output file is strings.resources, in this directory
resgen strings.txt


;from inside the en-NZ directory
;output file is strings.en-NZ.resources, in this directory
resgen strings.en-NZ.txt
```

The Resgen utility can convert resource files back into text files. Note however that you lose any comments when converting this way (the comments were already stripped as unnecessary information when making the .resources file). If you do convert this way, you must give Resgen the name of an output file; it will not work if you specify the input file only.

Resgen can also convert information into and out of another intermediary format called ResX. The ResX format is an XML-based format that wraps the name-value pairs you place in the .txt file into XML instructions. The advantage a .ResX file gives you over a .txt file is that you can store object structures inside a ResX file, so long as the object is serializable. Remember that resources can contain string information, but they can also contain objects, if this makes sense for your resource-aware application. The kind of objects you might want to store include fonts, or actual picture images such as bitmaps (a typical example is the ability to show a particular nation's flag based on the culture setting used).

As with .txt files, Resgen can be used to convert .ResX files into and out of resource files. The following sample demonstrates converting ResX into resources, and resources back into .txt, or .ResX. Note that once you have a resources file, it does not matter how it was constructed: it can be converted back into either .ResX or .txt files. In addition, .txt and .resx files can be converted into each other directly if required.

```
;from inside your app directory
;convert a .ResX file into a resources file
resgen strings.ResX

;now, convert the resources file just made (strings.resources), back into a .txt file
;the second parameter is the output file. Note that the extensions of the files
;must be correct for resgen to function properly
resgen strings.resources strings.txt

;now, convert the same strings.resources file back into a ResX file
;how the resources file was constructed does not matter.
resgen strings.resources strings.ResX
```

```
;you can also convert .txt or .resx files to and from one another
resgen strings.txt strings.ResX
resgen strings.ResX strings.txt
```

Note that because a ResX file can contain embedded objects, but a .txt file cannot, if you Resgen a ResX file into a .txt file and the ResX file has embedded objects (or if you convert a resources file with objects in it into a .txt file), the objects cannot be preserved. Instead, the datatype of the object is written to the .txt file (to give you some kind of feedback). You will be informed by Resgen that an error occurred, and all string resources will be correctly created.

The following sample is taken directly from a ResX file. You can see in this sample that an image has been embedded directly into the ResX file, which is not possible in a .txt file. The first section is labeled as [header] (this is not allowed in an actual Resx file) so you can see how a ResX file begins. Subsequent [entries] are the name value combinations for strings or objects. The XML code reflects this by having a data structure that defines a name-value pair, just as a .txt file would. The following code example demonstrates using ResX files.

```
[header]




...
[entries: strings]

    Welcome to our application!

    Please enter your name ==>

    Please enter the State you live in ==>
[entries: objects. This is an actual embedded image]

    AAEAAAD/////AQAAAAAAAAAMAgAAADtTeX    ...
...
```

Having constructed the resources, you can now move on to the next step: compiling your main assembly. This consists of compiling your assembly as you would normally, with an extra /res: switch in order to place your primary resources file into your main assembly. This is a very critical step, because this becomes your fallback resource, if others can't be found.

The following rules are used for looking up resources. Suppose someone is trying to access a Mexican Spanish (es-MX) resource inside your application. When your program is running, it has a standard process for looking for any Spanish resources, which includes taking the following steps:

- The global assembly cache is checked to see if it includes an es-MX assembly for your application. The global assembly cache is an area that allows you to reference shared resources for many applications, rather than having to include many copies of the same set of resources in the directory structure of every application you make. If the assembly is found, then that assembly is searched for the specific resource requested, and if found, the requested resource is used. If not found, then the search continues through the next steps.
- The directory of the currently executing assembly is checked to see if it contains an es-MX directory. If it does, that directory is searched to see if it has a valid satellite assembly for the requested culture. That assembly is then searched for the specific resource requested and if found, the requested resource is used. If not found or the assembly did not exist, then the search continues through the next steps.
- The global assembly cache is searched for the parent assembly of the requested resource, for your application. The parent is defined as the appropriate fallback culture. In this case, es-MX (Mexican Spanish) will fallback to es (Spanish). Note that providing any resource is viewed as a better alternative to throwing an exception, which is why this system is adopted. Parents are viewed as 'best-fit' candidates. This process also allows scope for reuse of resources: you need only include a particular resource at the parent level, if the child culture does not need to localize the requested resource. If the parent assembly exists in the global cache, then that assembly is searched for the specific resource requested and if found, the requested resource is used. If it is not found, then the search continues through the next steps.
- The directory of the currently executing assembly is checked to see if it contains a parent directory for es-MX (the parent directory for es-MX is es). If the parent directory exists, that directory is searched to see if it has a valid satellite assembly for the parent culture. If it does, then that assembly is searched for the specific resource requested and if found, the requested resource is used. If not found, then the search continues through the next steps.
- Parent assemblies are continually searched as in the previous step through many potential levels (each culture only has one parent, but a parent may have another parent). After all the original culture and parents have been searched for the resource and it is still not found, then the default is used.
- The default resource is that resource that is included with the main assembly when it is compiled (strings.txt). The default resource is viewed as the absolute fallback (the final parent), and it is strongly suggested that a default set of resources always be included. This helps ensure that exceptions are not thrown. Using resources should not result in exceptions.
- If the resource has not been found, then an exception is thrown indicating that the resource could not be found.

The process for trying to find an appropriate culture without throwing an exception is quite involved. In addition, different locations are searched when trying to find the specified culture. Including a default resource file simply ensures that at least some reasonable value will be found, without causing an exception to be thrown. In most cases, you will need to provide a full set of resources for each localized culture you are providing support for. A missing resources file should never be enough of a cause to throw exceptions back to the user. Your default resource therefore needs to include an entry for all potential resource searches.

Now that you know why a default resources file is important, remember that your default should be that which will be used by most users. To embed it in your main assembly, use the following command-line instruction.

```
' This line makes your main assembly, and links the resources file to it
' The /t: switch specifies the output type (executable, not .dll)
' The /res: switch specifies the resources file to embed
' The final argument is the main source file used to construct our assembly
' This code is run from your main application directory

vbc /t:exe /res:strings.resources myApp.vb
```

VB

You could run the application now, using the default resources. Hopefully, this will satisfy most of your users (who should be requesting the default resources directly); however, now you need to setup your satellite resources. There are few things to note about these other resources before continuing:

- If you want to replace the satellite resources (satellite assemblies once compiled), you can do so at any time, without having to recompile your main assembly
- Remember that satellite assemblies consist of resources only (no code is permitted)
- The resources in the assembly must match those expected as indicated by the name of the file, and the location of the file. Do not create a jp-JP resource (Japanese) and place it in an en (English) directory. It will be ignored.

For each subculture you want to compile as a separate satellite assembly, rename the file as ... The following sample demonstrates how to compile your resources file. Use the Assembly Linker Utility (al), which is a stripped down version of the main compiler. Note that you could use the main compiler if you want, but the Assembly Linker Utility works just as well, and is easier to ship with products.

```
;move into the en-NZ directory (not essential, but it makes things easier)
;ensure your strings file has the appropriate culture name in it;
;For New Zealand, the file should be named strings.en-NZ.txt

;use resgen to generate the resources file from the txt file
the following command will create the file strings.en-NZ.resources
resgen strings.en-NZ.txt

;- Invoke this command-line inside the en-NZ directory
;- The /t: switch specifies the type of the output file; a dll
;- The /embed: switch specifies the filename to include
;  inside the satellite assembly being created
;- The /out: switch specifies the name of the output file
;- It needs to be named according to the following pattern:
;  .resources.dll

al /t:lib /culture:en-NZ /embed:strings.en-NZ.resources /out:myapp.resources.dll

;complete the above steps for each satellite assembly you wish to create,
; changing the culture name appropriately.
```

Note the name of the output file in the above sample. This is critical, because if the files are not named correctly, they will not be found (this is a commonly mistaken step). You need to make sure that the name of the satellite assembly (the dll file) is .resources.dll. Also ensure that you specify the correct culture switch. Without the culture tag, the resources you request cannot be correctly identified.

You may be asking how you get to install resources into the global assembly cache, which we mentioned earlier. Storing resource files in the global assembly cache allows them to be used by multiple applications, without having to store the same resources with many different assemblies. When you use the al utility, you can also use the /install: switch, which allows you to specify the name of your output assembly file. When you use this switch, the resource is saved into the global assembly cache.

At this point, you know the steps needed to create a console application. The critical files and directories are highlighted in the following sample. Although the rest have been helpful to the process, they should be retained by you and not shipped with your product. They can be useful for making changes and creating updates. They are included in the following diagram to remind you what files you will have in your development area. Your directory tree structure will look as follows:

```
app directory
      |
      ----->   strings.txt
      ----->   strings.resources
```

```
-----> myApp.cs   or   myApp.vb   (C# or VB source file)
-----> myApp.exe                  the actual application
|                                 (strings.resources is embedded in this file)
|
-----> en-NZ
|        |
|        -----> strings.en-NZ.txt
|        -----> myApp.resources.dll    The satellite assembly for New Zealand
|
-----> de
         |
         -----> strings.de.txt
         -----> myApp.resources.dll    The satellite assembly for German
```

It is useful to know that there is another format for saving resources called ResX. ResX files are a resource equivalent to .txt files and are comprised of XML code, but can store objects as well as strings. They are generally generated through code (for more information, see How Do I... Use resources). An object such as an image is actually embedded into the ResX file as an equivalent form of binary information. You can only store serializable objects in ResX files, which are by definition available to be used in resource files. Once your .resx file has been created, you can use the Resgen utility on it to create a resources file, just as you would with a .txt file. Note that if you get errors when attempting to use the Resgen utility on a .resx file, a likely cause is that you are attempting to use non-serializable objects.

Up to this point, the discussion has focused on setting up your resources for use with a console application. If you are developing other applications however, such as an ASP.NET application, you may want to set up your resources differently. An example of this different setup is portrayed in the next diagram. Note that the code inside the application changes a little in order to be able to handle the fact that this application uses a different location for its resource files. In this approach, you can store all of your resources files in the **same directory**. The location of this shared directory is not critical, although the suggested location is a subdirectory of your main application. You do not need to differentiate them by their directory structure (although they do still need to be differentiated by name).

You do not compile an aspx page as such, and therefore, it is difficult to identify the assembly for an aspx application. Because of this, you cannot 'embed' your default resource file into your main assembly: it too must be placed in the shared resources directory you create.

up to this point the resources have been assembled into satellite assemblies. However, this example demonstrates that you do not need to actually compile your resources into libraries: you can leave them as .resources files and they are still deployable. Therefore, remember that you only need to generate your resources in this approach; you do not need to manually compile either your main aspx page, or your resources.

```
app directory
      |
      -----> myApp.aspx            an ASP.NET application. You cannot embed a
      |                            resources file into this application!
      |                            The aspx file knows to look in the single resources
      |                            directory for the files.
      |
      -----> resources
                |
                -----> strings.txt
                -----> strings.de.txt
                -----> strings.en-NZ.txt
                |
                -----> strings.resources         the default resources file
                |                                You can tell it is the default, because
                |                                it does not have a specific culture embedded
                |                                in it's name
                -----> strings.en-NZ.resources   the New Zealand resources file
                -----> strings.de.resources   the German resources file

Note: There are no dlls: therefore, there are no satellite assemblies.
      The resources are loaded directly from the resources files.
      This is NOT the suggested approach, but it is viable.
```

The following sample demonstrates how to set up your aspx application. Note that when selecting a site for this application, you must ensure that it is in a Web directory so that the aspx control can behave appropriately.

VB CreateAspxRes.aspx

[Run Sample] | [View Source]

A second approach to using resources in ASP.NET pages is also available. You can actually use satellite assemblies (rather than referencing resources files directly), when you are creating ASP.NET Controls. For more information, see Working with resource files.

You have now finished setting up your resource-aware applications. You can modify the satellite assemblies and resources files without requiring to recompile and distribute your main assembly or application, and you can even add in completely new satellite assemblies, although you may need to modify your code to recognize them.

Resources
Create resources
Use resources
Read and write resources
Make a ResourceReader

Back to Index

Get URL for this page

How Do I...Use resources?

This example demonstrates how to use resource files, dynamically loading a particular resource based on a selected culture. Resources allow you to develop robust, culture-aware programs without having to recompile your application because the resources have changed. The code required to access elements of a resource is based on the use of the **ResourceManager** object, and requires knowledge of the names of the elements within the resource. In this demonstration, the resources are presumed to already exist. Note that although Japanese is included in this demonstration, your console application must be setup appropriately to use Japanese resources.

VB UseResources.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Resources are an application-building feature that allow you to place culture-specific items inside satellite files, rather than directly in your main application. For example, error messages need to be translated into every language that is using the application. A resource is a formalized mechanism for loading these sets of culture-dependent items.

The code needed to load resources from a file is focused on the **ResourceManager** object. The **ResourceManager** is created by identifying a particular Assembly to base the **ResourceManager** on, or a Type within a particular assembly. This ensures that the correct set of resources is going to be used (resources are applied to assemblies, which is why an assembly needs to be identified). If you specify an assembly, you also need to specify the root name of the resources file to retrieve. For more information, see How Do I...Create Resources.

The following sample demonstrates creating a **ResourceManager**. Because **ResourceManagers** are expensive to use, you may want to declare only one (unless you have a specific need for more), and declare it at the class level. Further, a static (Shared) version of the **ResourceManager** can be used by many instances of your class, saving even further on memory resources.

```
' declare a ResourceManager, using the GetExecutingAssembly method of the Assembly
' class to determine which Assembly the ResourceManager is based on.
' The first parameter is the rootname of the resource file you want to load.
' In this case, our resourcefile's rootname is 'strings'
Public Shared rm As ResourceManager = New ResourceManager("strings", _
                       System.Reflection.Assembly.GetExecutingAssembly())

' This declaration is similar to the previous declaration, but also allows
' you to pass the Type of a specific resourceset object. This constructor should be used
' when you want to develop your own ResourceSet, which could control things such as the
' caching policy for images. When you have designed your own resourcset, you should also
' define your own ResourceReader by implementing IResourceReader, and control how your
' ResourceSet is parsed. To use this constructor, the last parameter needs to be the Type
' of the resourceSet you created.
Dim rs As ResourceSet = New ResourceSet("myResources.resources")
static ResourceManager rm = new ResourceManager("myResources", _
                       Assembly.GetExecutingAssembly(), GetType(rs))

' you can also declare a ResourceManager, passing it a Type. The ResourceManager
' then determines what Assembly that type is a part of, in order to derive information
' about associated resource files
Shared rm AS ResourceManager = New ResourceManager(GetType(ThisClass))
```

VB

Once you have created your **ResourceManager**, the next step is to identify the elements to retrieve. In almost all situations, you will know what your specific resources are, so you will be aware of the names given to them. This demonstration focuses on retrieving string resources; however, you can also retrieve other resource types such as images.

To retrieve string entries from a resources file, use the **GetString** method of **ResourceManager**. The resources file in question usually consists of many string values, each having an identifier (name) and value. When you call **GetString**, pass the name of the entry you want to retrieve; the associated value is returned. You can also pass a **CultureInfo** object to **GetString**, which identifies the culture to retrieve (if you don't, the current UICulture is used unless you specify otherwise). Remember, you will have many resource files that can have entries for the same value (for example, they may all have a 'welcome' entry). Each resource file is associated with a particular culture. When you call **GetString**, you can specify the culture to attempt to retrieve, and if the **ResourceManager** can find an appropriate entry in the resource file, it will be returned. If it cannot be found, then the **ResourceManager** falls back through the parent chain, trying to find an appropriate resource entry (rather than causing an exception to occur). This is demonstrated in the following example.

```
' These two entries are in a strings.resources file, and are in the default
' language of the software developers (which happens to be US-English, en-US). These are
' the neutral resources
' welcome = Hello and welcome to this application. Click 'Go' to continue...
```

```
' goodbye = Thank you for using this application

' This entry is in a strings.en-AU.resources file, identified as the Australian resources file
' welcome = Gidday, ladies and gents. Hit the 'Go button, and hold on to yer seats!

' declare a CultureInfo object, as a US-English resource
Dim ci1 As CultureInfo = New CultureInfo("en-US")

' declare a CultureInfo object, as an Australian-English resource
Dim ci2 AS CultureInfo = New CultureInfo("en-AU")

//get the Welcome and goodbye strings...
rm.GetString( "welcome", ci1 )   ' returns  'Hello and welcome...'
rm.GetString( "welcome", ci2 )   ' returns  'Gidday ladies and...'
rm.GetString( "goodbye", ci1 )   ' returns  'Thank you for using...'
rm.GetString( "goodbye", ci2 )   ' returns  'Thank you for using...'
                     ' the last return is because the specified resource could not
                     ' be found in the specified file, therefore the default is used
```

VB

Try to use Assembly-aware **ResourceManagers** whenever possible. This means you should use a **ResourceManager** by declaring it through one of its constructors, as shown in the first sample. However, there may be situations when you want to place your family of resource files in one directory, rather than having to split them up into the separate subdirectory structure. This method avoids using resources through their associated assemblies, which means that you don't have to associate the resources files with assemblies. Keep in mind that this is not how resources are intended to be used, but is available in certain scenarios.

The following sample uses the **CreateFileBasedResourceManager** method of the **ResourceManager** to create your **ResourceManager**. This method needs the base name of the resources file you will be using (as in one of the previous examples), and the location and name of the resources file you are using. The final parameter is an optional Type of a ResourceSet (you will specify null, indicating the default should be used, but you could specify your own resource format here). Once created, the **ResourceManager** can be used as in the previous examples.

```
' use the assembly-ignorant method to create a resourceManager.
' The file you look at is the Australia strings resources file. Note that all you need to specify
' is the directory you want to look in for the resources. ALL resources can be placed in the
' same directory, as long as each is differentiated by the name of the culture for that resource.
' In this example, you have placed all the resources in a subdirectory of the application,
' called "resDir".
Public Shared rm As ResourceManager = ResourceManager.CreateFileBasedResourceManager( _
               "strings", Server.MapPath("ResDir") & Environment.DirectorySpearatorChar, Nothing)

' Using the ResourceManager just created...
rm.GetString( "welcome")
rm.GetString( "goodbye")
```

VB

You can also save nonstring resources. To access these resources, create a **ResourceManager**, and instead of using the **GetString** method, use **GetObject**. As with a String, the object in question will have been placed into the resources file as a name-value pair, so you can use **GetObject** the same way you would use **GetString**.

```
' create a ResourceManager...
' note you named the base file differently, to reflect the kind of resources
' you are storing. The file has three pictures in it
Public Shared rm As ResourceManager = new ResourceManager("images", _
                      System.Reflection.Assembly.GetExecutingAssembly());

' get the resources out of the resource, specifying different cultures to retrieve
rm.GetObject("flag1", new CultureInfo("ja-JP"))   ' a Japanese flag image
rm.GetObject("car1", new CultureInfo("it-IT"))    ' an Italian car image
rm.GetObject("Sport1", new CultureInfo("ta-IN"))  ' a snapshot of India's national sport
```

VB

To put this concept into practice, in the final example, you will create a small demonstration WinForms application. This application has string resources, but it also has an image resource to demonstrate that objects can also be made into resources. When you made the resources, you used a short program to write all of the resources into the same resources file, which saves work later. The code for the sample simply loads in the appropriate resources, and makes WinForms objects from them, which are then displayed to the user.

To change the particular resources that you see in this demonstration, you can go to your regional options (Start/Settings/Control Panel, and then select Regional Options) and on the General tab, change your locale to English (NZ), English (US), Japanese (which may require installation of the Japanese character set), or German. If you set it to any other culture, the default is used (in this example, U.S. English). Review the code for this demonstration to see how this setting was used (Thread.CurrentThread.CurrentCulture). The executable you need to run is MultipleResources. This executable actually creates the resources that will be used, and then the automation for the page turns the resources into the satellite assemblies you will find in the culture specific subdirectories.

A good test for this demonstration is to set the culture setting to something other than the four supported settings. Because standard fallback is used, the resources displayed should be the default (neutral) resources, which in this case are the U.S. English resources.

VB MultipleResources.exe

[Run Sample] | [View Source]

Resources

Create resources

Use resources

Read and write resources

Make a ResourceReader

Back to Index

Get URL for this page

How Do I...Read and Write Resources?

This example illustrates how to read and write resource files through code. You can use other tools (such as Resgen) to write resource files; however, you can also use the **ResourceWriter** object to create new resource files from scratch, providing the potential to create your own environment for making resource files from the bottom up. This process skips the step of having to create an intermediary text file for your string resources.

VB ReadWriteResources.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

The easiest way to make resource files is through the Resgen utility. This has the advantage of being developed from a pre-existing text file, which can be easily modified. Even if you don't use Resgen to create your resources file, you can use it to retrieve the information back out from a resources file, creating a resultant text file.

The **ResourceWriter** object can be also used to create resource files. The **ResourceWriter** is constructed by specifying the name of the target resources file, and can be used as many times as you would like to add items to the new file. Use the **AddResource** method to specify the name of the item you are adding, and then the value you want to retrieve. The name must be a string, however the value can be a string, or any object if you are using resources for objects. The following example demonstrates creating a **ResourceWriter**, and adding string entries to the file. Note that when you are finished, you call the **Generate** method to save the information you are creating.

```
' Create a .resources file, specifying a new resources file
Dim rw As ResourceWriter = New ResourceWriter("Greeting.resources")

' add some strings to the file
rw.AddResource("Greeting", "Welcome to Microsoft .Net Framework!")
rw.AddResource("PasswordException", "Sorry, that is not the correct password.")
rw.AddResource("Purchase", "Please select an item to purchase from the store:")
rw.AddResource("Goodbye", "Thank you for visiting Microsoft .Net Framework!")

' create and write to the file
rw.Generate()

' close the resourcewriter. Note that if you close without calling generate
' first, the generate method is implicitly called
rw.Close();
```

VB

Having made the file, you can use the **GetString** method on a **ResourceManager** to retrieve the information back out of the resources file just created. You must specify a particular item to retrieve, as in the following sample. Note that you construct the **ResourceManager** using the static (shared) method, **CreateFileBasedResourceManager**.

```
' make a ResourceManager, as an assembly-ignorant ResourceManager through the
' CreateFileBasedResourceManager method. Simply specify the name of the resources
' file, and the location. The single "." indicates that the current directory should
' be searched the null indicates to use the default resourceset
Dim rm As ResourceManager = CreateFileBasedResourceManager( "Greeting", ".", null)

' having got a reference to the resources, retrieve specific strings...
' Note that the CASE MATTERS
rm.GetString("Greeting")
rm.GetString("PasswordException")
rm.GetString("Purchase")
rm.GetString("Goodbye")
```

VB

The **ResourceReader** can be used to parse all the resources in a given resource file, allowing you to quickly retrieve all the resources without having to specify their names (that is, you don't have to know what they are). This can be useful if you want to print out the resources, or remake the original entries to modify them. The following code demonstrates creating a **ResourceReader** for a particular file, and iterating through that file, printing out the names (keys) and values.

```
' declare and create an instance of ResourceReader
' it is declared for a particular .resources file
Dim rr As ResourceReader = New ResourceReader("file.resources")

Dim s As String = ""
Dim d As DictionaryEntry

' iterate through the reader, printing out the name-value pairs
For Each  d in rr
        Console.WriteLine(d.Key & ":   " & d.Value)
Next d

' close the reader
rr.Close()
```

VB

Remember that you can also save objects into your resource files. A perfect example is saving the flag images, so that a particular nation's flag is displayed when you load its corresponding resources. To save nonstring objects, use the **AddResource** method just as you would for strings, but specify the object you want to save, rather than a string. You must still specify a name for the resource you are saving, so that it can be identified and retrieved later.

The following code example demonstrates creating a couple of objects and saving them to file. When you choose objects to save to file, make sure that they are serializable.

```
' declare your ResourceWriter
Dim rw As ResourceWriter = New ResourceWriter("items.resources")

' create an object to save to the resource file... it is the US flag
Dim b As Bitmap = New Bitmap("us50.gif")

' we can create, and add String resources as well, in the same file
Dim message As String = "Hello, and welcome!";

' repeat for other serializable objects...

' add the object and string to the ResourceWriter
rw.AddResource("flag", b)
rw.AddResource("welcome", message)

' Generate the file
rw.Generate()

' repeat this process for each set of resources you want to create,
' such as items.de.resources, or items.ja.resources
```

VB

Just as you can use a **ResourceReader** and **ResourceWriter**, so too can you use a **ResXResourceReader** and **ResXResourceWriter**. A **ResXResourceWriter** creates an intermediate .Resx file, which is a combination of binary information (such as images) and XML data. You can use the **ResXResourceReader** and **ResXResourceWriter** just as you would in the examples above. After you run your finished code, if you are writing out information, note that the output will be a .Resx file, which you need to use the Resgen utility on to turn it into a .resources file.

Resources
Create resources
Use resources
Read and write resources
Make a ResourceReader

Back to Index

Get URL for this page

How Do I...Make a ResourceReader?

There are two steps to creating a custom resource reader. The first step is to create a class that implements **IResourceReader**, which requires a **Close** method and an implementation for **GetEnumerator**. The following code example dmonstrates creating a basic class that implements **IResourceReader**.

```
Class CustomReader : Implements IResourceReader

    Public Sub Close()
        // Close any items allocated for the resources
    End Sub

    Public Function IEnumerable_GetEnumerator() As IEnumerator Implements IEnumerable.GetEnumerator
        return GetEnumerator();
    End Function

    Public Function GetEnumerator() As IDictionaryEnumerator Implements IResourceReader.GetEnumerator
        // Return an enumerator for the resource collection
    End Function

End Class
```

VB

The next step is to generate a custom resource manager that uses the custom reader. To do this, create a derived class of **ResourceManager**, and override **InternalGetResourceSet** to allow for the creation of the resource reader defined earlier. The following code example demonstrates a simple **ResourceManager**.

```
Public Class CustomManager : Inherits ResourceManager
    Protected Overrides Function InternalGetResourceSet( culture As CultureInfo, _
                                                         createIfNotExists As Boolean, _
                                                         tryParents As Boolean ) As ResourceSet
        Return new ResourceSet( new CustomReader() )
    End Function
End Class
```

VB

Example

VB List.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

The following example walks you through the process of creating a resources-aware application. The purpose of this demonstration is to make you more familiar with the steps required to make resources, and use them. It is not intended to show you how to write the code that actually utilises these resources: that is discussed under the other headings in this topic. The sample assumes that the code to use the resources is provided, as is the actual file of strings you want to use for the application you are creating.

The following instructions are for creating your own console, resources application. They are intended to walk you through the process of creating an application with a basic (primary) set of resources, and one satellite assembly. Once this process has been completed, you should be able to run the application and see the different results based on the culture you select. In these instructions, entries enclosed in quotes indicate text you should type directly into the command prompt. Do not include the quotes themselves. A brief list of terms you may be unfamiliar with is included at the bottom of this page.

Steps **Instructions**

1. Open a console window (in Windows, go to Start/Programs/Accessories/Command Prompt)

2. Create a new directory in your root directory (either with the "md myApp" command, or via explorer) named myApp and then move your console into that directory ("cd myApp").

3. Create a new file inside that directory called strings.txt, and open that file for editing ("notepad strings.txt"). When asked, create the file (click Yes).

4. From the code source links page, click the link to the file called "strings.txt". Copy the entries and paste them into the file you just made. Save and close the file. Once done, come back to complete the following steps.

5. Use resgen to compile a resources file from the text file: "resgen strings.txt" makes a file called strings.resources

6. Create a new file in the same directory called myApp.vb, and open that file for editing ("notepad myApp.vb"). When asked, create the file (click Yes).

7. From the code source links page, click the link to the file called "myApp.vb". copy the code, and paste it into the file you just made. Save and close the file. Once done, come back to complete the following steps.

8. Compile your main assembly, linking in your resources file, with the following command: "vbc /t:exe /res:strings.resources myApp.vb"

9. Try out the application ("myApp"). You will notice at this point that it does not matter WHICH culture you select in the application: the questions are all the same. We need to continue by making an extra resources file.

10. Create a sub-directory in your main directory called en-NZ ("md en-NZ"). Change directory to that directory ("cd en-NZ").

11. Create a new file inside that directory called strings.en-NZ.txt, and open that file
for editing ("notepad strings.en-NZ.txt"). When asked, create the file (click Yes).

12. From the code source links page, click the link to the file called "strings.en-NZ.txt".
copy the entries and paste them into the file you just made. Save and close the file. Once done,
come back to complete the following steps.

13. Use resgen to compile a resources file from the text file:
"resgen strings.en-NZ.txt" makes a file called strings.en-NZ.resources

14. Compile the satellite assembly in the current directory:
"al /t:lib /culture:en-NZ /embed:strings.en-NZ.resources /out:myApp.resources.dll"

15. Move back into your main directory ("cd.."). Run the application again ("myApp")
and run it as a US, and a New Zealand application. Notice that the prompts change now,
based on the culture selected.

NB: We did not have to recompile our main assembly, when we added this new
resources file to our project. This can be indespensable when putting out updates to this
kind of information. You need only redistribute your resources (the satellite assemblies),
you do NOT need to redistribute your entire application.

16. Repeat steps 10 - 15 for any other cultures you wish to include, changing the culture
identifier, and making the new strings file. Remember to add in a new string for ALL entries
you wish to see for your new culture, otherwise the system will default back to the main
assembly. Note that for this particular demonstration, if you want to add a brand new culture,
you will also need to modify the .cs file and recompile your main assembly (see step 8) to
recognise the new culture. Normally, you would not request the user for a specific culture:
they would set it via Control Panel/Regional Options.

Term	Description
cd	change directory, to a specified directory
cd..	change directory up, by one directory
md	make directory, as a sub-directory of your current location
al	assembly linker, a minimal form of the compiler
resgen	The utility which generates resource files from text files
myApp	The name of the executable file being made. Typing this runs the executable
notepad	The notepad program, which opens a file you specify for editing, or creates a new file if requested.

```
------- --------------------------------
```

The following example walks you through the process of creating a resources-aware application, for an ASP.NET application. The purpose of this demonstration is to make you more familiar with the steps required to make resources, and use them. It is not intended to show you how to write the code that actually utilises these resources: that is discussed under the other headings in this topic. The sample assumes that the code to use the resources is provided, as is the actual file of strings you want to use for the application you are creating.

The following instructions are for creating your own ASP.NET, resources application. They are intended to walk you through the process of creating an application with a basic (primary) set of resources, and two satellite assemblies. Once this process has been completed, you should be able to run the application and see the different results based on the culture you select. It is essential to remember that you need to create your base directory inside your localhost somewhere, so that the asp code can run as intended. In these instructions, entries enclosed in quotes indicate text you should type directly into the command prompt. Do not include the quotes themselves. A brief list of terms you may be unfamiliar with is included at the bottom of this page.

Steps **Instructions**

1. Open a console window (in Windows, go to Start/Programs/Accessories/Command Prompt)

2. Create a new directory inside your localhost (either with the "md myApp" command, or via explorer) named myApp and then move your console into that directory ("cd myApp"). If you are uncertain as to what your localhost is, run inetmgr from the run command (goto Start/Run... and type in inetmgr), and expand down to your default website. Right-click on the quickstart entry (this entry should exist: it is the location of these examples) and select properties. Note the local path: you need to create your directory somewhere below this path. To invoke your application once built, type 'localhost/quickstart/myApp/CreateAspxRes.aspx' inside Internet Explorer.

3. Create a sub-directory in your myApp folder called resources ("md resources"). Change directory to that directory ("cd resources").

4. Create a new file inside the resources directory called strings.txt, and open that file for editing ("notepad strings.txt"). When asked, create the file (click Yes).

5. From the code source links page, click the link to the file called "strings.txt". Copy the entries and paste them into the file you just made. Save and close the file. Once done, come back to complete the following steps.

6. Use resgen to create a resources file from the text file:
"resgen strings.txt" makes a file called strings.resources

7. Create a new file inside the resources directory called strings.en-NZ.txt, and open that file for editing ("notepad strings.en-NZ.txt"). When asked, create the file (click Yes).

8. From the code source links page, click the link to the file called

"strings.en-NZ.txt".
copy the entries and paste them into the file you just made. Save and close the file. Once done,
come back to complete the following steps.

9. Use resgen to create a resources file from the text file:
"resgen strings.en-NZ.txt" makes a file called strings.en-NZ.resources

NB: You do not need to compile the resources into assemblies in this example.

10. Change directory back to your main directory (cd..)

11. Create a new file inside your myApp directory called createaspxres.aspx, and open that file
for editing ("notepad createaspxres.aspx"). When asked, create the file (click Yes).

12. From the code source links page, click the link to the file called "createAspxRes.aspx" (make sure it is the VB version).
copy the entries and paste them into the file you just made. Save and close the file. Once done,
come back to complete the following steps.

13. Open an instance of internet explorer, and type in the following url to run your application:

localhost/quickstart/myApp/createaspxres.aspx

This will run the demonstration just created.

14. Repeat steps 7 - 10 for any other cultures you wish to include, changing the culture
identifier, and making the new strings file. The German (de) string files are ready for
you to repeat this process on. Remember to add in a new string for ALL entries you wish to
see for your new culture, otherwise the system will default back to the main assembly. Note
that for this particular demonstration, if you want to add a brand new culture, you will
also need to modify the CreateResources.aspx file to recognise the new culture. Normally,
you would not request the user for a specific culture: they would set it via Control
Panel/Regional Options.

Term	Description
cd	change directory, to a specified directory
cd..	change directory up, by one directory
md	make directory, as a sub-directory of your current location
al	assembly linker, a minimal form of the compiler
resgen	The utility which generates resource files from text files
notepad	The notepad program, which opens a file you specify for editing, or creates a new file if requested.

Microsoft .NET Framework
Control access to a component?
Check Windows Identity?
Request permissions?

Administration
Script security policy changes?

ASP.NET Security
Secure an ASP.NET application?
Check Windows Identity?
Apply forms-based authentication?
Authorize users or roles?

Back to Index

Get URL for this page

How Do I...Control access to my shared component?

This topic describes how to control access to an assembly (or individual method within an assembly) using Code Access Security.

Every assembly loaded by the common language runtime is assigned evidence that describes its identity. This evidence can be the path or URL from which the assembly was loaded or it can be a digital signature given to the code by its publisher. To control access to your code, you can simply demand that your caller have a specific identity. For example, to limit access to a shared component to only code from the same publisher, the publisher would sign all their code and then place a demand for that signature within their shared component.

The identity permissions found under the namespace **System.Security.Permissions** are used for this purpose. Identity permissions are provided for the following types of assembly identity: strong name, Authenticode pulisher certificate, URL of origin, site of origin, and Internet Explorer security zone. All identity permissions support three types of identity demands as described below:

- **Demand:** All callers on the call stack are required to have the specified identity. This check is performed at runtime.
- **LinkDemand:** Only the code's immediate caller is required to have the specified identity. The caller must be trusted to not allow misuse of the called code by its own callers. This check is performed during loading.
- **InheritanceDemand:** Requires the specified identity of any code that attempts to inherit from or override a method on the protected code. This check is performed during loading.

Because the last two are performed during loading, they can only be specified declaratively. Declarative security is placed in an assembly's manifest by the compiler where it can be read and acted upon by the common language runtime when loading that assembly. A straight demand, because it occurs at runtime, can be used either imperatively or declaratively.

Below is an example of making a link demand on a method for a strong name identity. The public key has been abbreviated for readability.

```
Public Shared Sub _
  <StrongNameIdentityPermission(SecurityAction.LinkDemand, PublicKey := "002400000...")> _
  ProtectedMethod()
    'do something
End Sub
```

VB

The additional properties **Name** and **Version** can also be provided. By specifying **Name**, **Version**, and **PublicKey**, an exact version of an assembly can be reliably demanded. Specifiying only **Name** and **PublicKey** will allow the demand to succeed if the assembly name and signature match regardless of the assembly version. Specifying only the **PublicKey**, as in the code example above, causes the security system to look for only the required signature and is useful when you desire to limit access to a group of code signed by the same key.

Attaching a strong name signature to your code involves two steps: creating the strong name key and compiling your assembly with that key. The first is accomplished by using the SN utility that is provided with the .NET Framework SDK. Below is the command-line syntax for creating a key pair and viewing the public key portion (you will need to make an identity demand for code signed with the corresponding private key).

```
sn -k keypair.dat

sn -p keypair.dat publickey.dat
sn -tp publickey.dat
```

The second step requires adding a declaration to the assembly to indicate the location of the file generated in step one.

```
<Assembly: AssemblyKeyFile("keypair.dat")>

Public Class MyClass
    'something interesting
End Class
```

VB

It is also possible to delay-sign an assembly. This reserves room for the signature in the assembly manifest but does not actually sign the assembly. Delayed signing is used when the author of the assembly does not have access to the private key that will be used to generate the signature. For more information about delayed signing, see the reference documentation on **AssemblyDelaySignAttribute**.

The following example contains three assemblies: a shared component written by Company A, an application written by Company A that uses the shared component, and an application by Company B that tries to use the shared component from Company A. Run the two executables, noting in their titles whether they are from Company A or B, and observe the results. A SecurityException is thrown when the code from Company B tries to call that protected code from Company A.

VB CodeIdentityDemand

[Run Sample] | [View Source]

Microsoft .NET Framework
Control access to a component?
Check Windows Identity?
Request permissions?

Administration
Script security policy changes?

ASP.NET Security
Secure an ASP.NET application?
Check Windows Identity?
Apply forms-based authentication?
Authorize users or roles?

Back to Index

Get URL for this page

How Do I...Check the Windows Identity in a client application?

User identity is a common means of controlling access to a business application or limiting the options available within that application. The .NET Framework classes under the namespace **System.Security.Principal** are provided to assist in making such role-based security determinations.

These classes and the mechanisms they expose are highly extensible. They allow host code to provide its own user identity and role information, or allow it to expose the user account and group information provided by Windows. For more complete details regarding how to extend the role-based security system, consult the .NET Framework SDK Developer's Guide.

For those who simply need to check the user's Windows user name and group memberships from a client application, this topic will show you how.

The Framework provides a **WindowsIdentity** class that represents an authenticated Windows user and a **WindowsPrincipal** class that encapsulates the **WindowsIdentity** and information about the user's role memberships. These objects representing the current user are accessible in one of two ways: using a static property on the **Thread** object or a static method on the **WindowsIdentity** object. Examples of both are given in the following section.

Accessing the current principal from the **Thread** is the standard approach, and it works for all types of principal objects. But, because this method returns an **IPrincipal**, it must be cast as a **WindowsPrincipal** before it can be used as one. Notice that before accessing the current principal, a call to **SetPrincipalPolicy** is made. This is noteworthy because without it the principal returned would be a **GenericPrincipal** containing no user information. As the call to **SetPrincipalPolicy** requires the ControlPrincipal **SecurityPermission** (one not normally given out to less than fully trusted code) this prevents semi-trusted code (such as that running off the Internet) from gaining access to a user's account name.

```
AppDomain.CurrentDomain.SetPrincipalPolicy(PrincipalPolicy.WindowsPrincipal)

Dim user As WindowsPrincipal = CType(System.Threading.Thread.CurrentPrincipal, WindowsPrincipal)
Dim ident As WindowsIdentity = user.Identity
```

VB

Since checking for a Windows identity is expected to be a very common case, a shortcut is provided using a static method on the **WindowsIdentity** object as shown in the example below. Please note, however, that this method requires the same level of permission as the one above.

```
Dim ident As WindowsIdentity = WindowsIdentity.GetCurrent()
Dim user As New WindowsPrincipal(ident)
```

VB

Once a **WindowsPrincipal** is retrieved, group membership can be checked using the method **IsInRole**. If the goal of checking role group membership is to deny access to an application (vs. customizing the user experience), an even simpler approach is to use the **PrincipalPermission** to demand the required role.

The following example verifies that a user is in a given NT group using both **WindowsPrincipal** and **PrincipalPermission**.

VB WindowsIdentityCheck

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Microsoft .NET Framework
Control access to a component?
Check Windows Identity?
Request permissions?

Administration
Script security policy changes?

ASP.NET Security
Secure an ASP.NET application?
Check Windows Identity?
Apply forms-based authentication?
Authorize users or roles?

Back to Index

Get URL for this page

How Do I...Request the permissions my code needs?

Since you do not necessarily have control over what permissions are assigned to the code you write, the common language runtime provides a mechanism for requesting the permissions that you feel your code must have in order to run properly. If the code is not granted the required permissions, it will not run. And, because permission requests are stored in an assembly's manifest, the end user can run a tool to determine what permissions have been requested by the assembly author and then take the appropriate steps to grant those permissions if they need the code to run on their machine.

Three types of permission requests are supported:

- **RequestMinimum:** The permissions the code must have to run properly. If these permissions cannot be granted, the code will not be executed.
- **RequestOptional:** The permissions that should be granted if allowed by policy. The runtime will attempt to execute code even if permissions it requests as optional have not been granted.
- **RequestRefuse:** The permissions that code should never be granted. Code will not receive these permissions, even if they would normally be granted to it. This is an extra precaution you can take to prevent your from code being misused.

Permission requests can only be made in a declarative fashion and must always be at the assembly level (the assembly is the unit to which permissions are granted by the security system). The following code is a request stating that an assembly must have unrestricted access to the file system in order to function.

```
<Assembly: FileIOPermission(SecurityAction.RequestMinimum, Unrestricted := True)>

Public Class FileMover
    'something interesting
End Class
```

VB

Several requests of the same type can be made, in which case the final permission set requested is the aggregate of all requests of that type. In the example below **RequestMinimum** is used twice with different permissions to state that the assembly must have the ability to use Reflection Emit and perform serialization in order for it to function.

```
<Assembly: ReflectionPermission(SecurityAction.RequestMinimum, ReflectionEmit := True)>
<Assembly: SecurityPermission(SecurityAction.RequestMinimum, SerializationFormatter := True)>

Public Class CodeGenerator
    'something interesting
End Class
```

VB

The same permission can also appear in requests of different types. For instance, the example program at the bottom of this page uses an **EnvironmentPermission** in each of its three requests (Minimum, Optional, and Refuse). This is useful when a permission encompasses a number of operations and you want to ensure that your assembly has the ability to perform some of those operations while being prevented from performing others. It is important to note that any permission you refuse using **RequestRefuse** will not be granted to your assembly even if you request that same permission using **RequestMinimum**.

In addition to requesting individual permissions, entire sets of permissions can be requested in a compact fashion. The example below shows two requests: one stating that an assembly must have unrestricted access to the file system in order to function and one stating that it will take any and all other permissions that the security system is willing to grant it.

```
<Assembly: FileIOPermission(SecurityAction.RequestMinimum, Unrestricted := True)>
<Assembly: PermissionSet(SecurityAction.RequestOptional, Name := "FullTrust")>

Public Class FileMover
    'something interesting
End Class
```

VB

The previous example shows how to request a permission set by name, but it is also possible to use a custom permission set representing the exact permissions you want. For more information on how to do this, search for **PermissionSetAttribute** in the .NET Framework SDK Reference.

The SDK provides a tool called PERMVIEW that is useful for verifying that your permission requests are correct. Running PERMVIEW on a compiled assembly will read the permission requests out of the assembly's manifest and display them as shown below.

```
C:\>permview filemover.exe

Microsoft (R) .NET Framework Permission Request Viewer.  Version 1.0.XXXX.0
Copyright (C) Microsoft Corp. 1998-2001
```

```
minimal permission set:
<PermissionSet class="System.Security.PermissionSet"
               version="1">
   <IPermission class="System.Security.Permissions.FileIOPermission"
                version="1"
                Unrestricted="true"/>
</PermissionSet>


optional permission set:
<PermissionSet class="System.Security.PermissionSet"
               version="1"
               Unrestricted="true"/>


refused permission set:
  Not specified
```

The following example shows a simple program for reading Windows environment variables. The program is intending to collect very basic information about the user's hardware (processor type, # of processors, etc.) but has safeguards in place to prevent it from ever being used to obtain private information (user name, computer name, etc.).

VB RequestPermission

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Microsoft .NET Framework
Control access to a component?
Check Windows Identity?
Request permissions?

Administration
Script security policy changes?

ASP.NET Security
Secure an ASP.NET application?
Check Windows Identity?
Apply forms-based authentication?
Authorize users or roles?

Back to Index

Get URL for this page

How Do I...Script Security Policy Changes?

The common language runtime ships with an advanced security policy system. An assembly receives permissions to access protected resources based on evidence about the code (such as its URL of origin or its publisher certificate). The set of granted permissions is a function of what various policy levels grant to the assembly based on its evidence. There are three policy levels that can be administered: the enterprise policy, machine policy, and user policy level. Each policy level consists of a tree of code groups. Each code group consists of a membership condition and an associated permission set. Code is granted the permission set associated with a code group if it meets the respective membership condition. Changing code groups in the user, machine, or enterprise policy allows administrators to determine what permissions are granted to assemblies. It might become necessary to script policy changes. In these cases, the Code Access Security Policy (Caspol.exe) command line tool can be used to create batch files containing policy change commands.

Note: For all standard administrative tasks, it is highly recommended you use the common language runtime configuration (Mscorcfg.msc) tool.

Note: For more information regarding the policy system please see the security documentation in the Framkeworks SDK.

The Code Access Security Policy (Caspol.exe) command line tool

The SDK ships the caspol command line policy administration tool. This tool can be used to create batch files for scripting security policy changes. Typing **caspol -?** at the command line will show you the available options of the tool.

Scripting Against Named Code Groups

In the default policy shipping with the common language runtime, all code groups have a unique name. If code groups have not been deleted or renamed it is possible to uniquely script changes against these code groups. The most common code group names that will be used for scripting are:

Code Group Name	Description
All_Code	The root code group in every policy level.
My_Computer_Zone	Code group that applies to code on local computer
Internet_Zone	Code group that applies to code from the Internet
LocalIntranet_Zone	Code group that applies to code from the Intranet

To see a complete list of code groups and their names in all policy levels, you can use the following caspol command: **caspol -all -listgroups**. In order to change a code group's permission set, include a command of the following form in your batch script:

```
caspol PolicyLevel -chggroup Name of code group PermissionSetName
```

In order to add a new code group include a command of the following form in your batch script:

```
caspol PolicyLevel -addgroup Name of Parent code group MembershipCondition
PermissionSetName CodeGroupFlags
```

In order to reset policy to the default state at a policy level include a command of the following form in your batch script:

caspol PolicyLevel -reset

Sample Scripts

The following caspol batch script resets policy to the default on all policy levels and grants full trust to intranet applications. Note, since the granted permissions to code are calculated as the intersection between policy levels and since (in default policy) both enterprise and user policy level are set to full trust, you only need to change the machine policy level in order to guarantee that intranet applications receive full trust. In other words, the script below guarantees that intranet applications will run with full trust (while code from other places of origin will run with the permissions given it in default policy)

```
caspol -all -reset
caspol -machine -chggroup LocalIntranet_Zone FullTrust
```

The following script of caspol commands shows how to set policy so code from the internet will not receive any permissions from machine policy (first line). The second command shows how to add a code group for granting full trust to code signed by the publisher that signed Myexe.exe. Note that the new code group is hung off the root of the machine policy.

```
caspol -machine -chggroup Internet_Zone Nothing
caspol -machine -addgroup All_Code -pub -file Myexe.exe FullTrust
```

Serialization
Serialize an object graph
Use ISerializable

Back to Index

Get URL for this page

How Do I...Serialize an Object Graph

This sample shows how to serialize a graph of objects using the serialization infrastructure.

Example

VB Serialize.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Serialization

Serialize an object graph

Use ISerializable

Back to Index

Get URL for this page

How Do I...Use ISerializable to Customize the Serialization of a Class?

This sample illustrates how to use the **ISerializable** interface to customize the serialization of your own objects. This sample serializes objects based on the mscorlib assembly namespace tree structure. Each object contains a property as the name of the item and a list of children as per the real object in the mscorlib assembly. To demonstrate the customization possible with **ISerializable**, "(SAVED)" is appended to the namespace name to indicate that the values are not live. When the data is read back in, this tag could be modified or removed. Note: This sample requires the System.Runtime.Serialization.Formatters.Soap.dll.

Example

VB ISerializable.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Services
Write a simple Service

Back to Index

Get URL for this page

How Do I...Write a Simple Service Application?

Microsoft Windows NT services enable you to create long running executable applications that run in their own Windows sessions. These services can be automatically started when the computer boots, can be paused and restarted, and typically do not show any user interface. This makes services ideal for use on a server or whenever you need long-running functionality that does not interfere with other users who are working on the same computer. You can also run services in the security context of a specific user account that is different from the logged-on user or the default computer account. For more information about services and Windows sessions, see the "About Services" section in the Platform SDK documentation in MSDN Online.

Using the .NET Framework, you can easily create services by creating an application that is installed as a service. For example, suppose you want to monitor performance counter data and react to threshold values. You could do this by writing an NT service and deploying it to the computer that has the counters you want to monitor.

The following sample is a simple service application. You can install it using the .NET Framework SDK Installation Utility. From a command prompt run:

```
> InstallUtil.exe SimpleService.exe
```

After the installation is completed you can start the Hello World Service from the Services Management Console. After the service starts you should see the service's event log being filled with greetings from the sample service. You can use the Services Management Console to pause, continue, start, and stop the service, and observe how such operations affect the messages written to the event log. After you are done experimenting with the sample, you can uninstall the service by running the following command from the command window (you should stop the service before uninstalling it):

```
> InstallUtil.exe /u SimpleService.exe
```

Example

VB SimpleService.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Setup

Create system resources

Execute custom setup code

Back to Index

Get URL for this page

How Do I...Create System Resources During Installation?

An application can consist not only of traditional program files, but also of associated resources such as message queues, event logs, and performance counters that must be created on the deployment destination. You can configure your application to create these resources when your application is installed and to remove them if your application is uninstalled, by using installation components.

This sample illustrates how to build an application that will set up its own message queue and its own event log during installation and delete those two system resources when the application is uninstalled. To run the sample you need to have message queueing installed on your machine. You can install message queueing from Start -> Settings -> Control Panel -> Add/Remove Programs -> Add/Remove System Components.

Run the following command line to install the system resources:

```
> InstallUtil.exe Installers.exe
```

Now, you can verify in the message queuing management console that an "InstallersSample" message queue was installed, and that an "InstallersSample" event log source was set up in the registry.

To uninstall the message queue and the event log source, run the following command:

```
>InstallUtil.exe /u Installers.exe
```

How does it work? The InstallUtil.exe will search, using the Reflection mechanism, all assemblies passed as the command line arguments for classes inheriting from **Installer** class and with **RunInstallerAttribute** set to true. The Installers then handle the install and uninstall actions.

Example

VB Installers.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Setup
Create system resources
Execute custom setup code

Back to Index

Get URL for this page

How Do I...Create System Resources During Installation?

An application can consist not only of traditional program files, but also of associated resources such as message queues, event logs, and performance counters that must be created on the deployment destination. You can configure your application to create these resources when your application is installed and to remove them if your application is uninstalled, by using installation components.

This sample illustrates how to build an application that will set up its own message queue and its own event log during installation and delete those two system resources when the application is uninstalled. To run the sample you need to have message queueing installed on your machine. You can install message queueing from Start -> Settings -> Control Panel -> Add/Remove Programs -> Add/Remove System Components.

Run the following command line to install the system resources:

```
> InstallUtil.exe Installers.exe
```

Now, you can verify in the message queuing management console that an "InstallersSample" message queue was installed, and that an "InstallersSample" event log source was set up in the registry.

To uninstall the message queue and the event log source, run the following command:

```
>InstallUtil.exe /u Installers.exe
```

How does it work? The InstallUtil.exe will search, using the Reflection mechanism, all assemblies passed as the command line arguments for classes inheriting from **Installer** class and with **RunInstallerAttribute** set to true. The Installers then handle the install and uninstall actions.

Example

VB Installers.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Setup

Create system resources

Execute custom setup code

Back to Index

Get URL for this page

How Do I...Execute Custom Code During Installation?

An application can consist not only of the traditional program files, but also of associated resources such as message queues, event logs, and performance counters that must be created on the deployment destination. You can configure your application to create these resources when the application is installed and to remove them if the application is uninstalled, by using installation components.

This sample illustrates how to build an application that will run custom code when installed or uninstalled.

Run the following command line to install the system resources:

```
> InstallUtil.exe CustomAction.exe
```

Note that the message "Installing: Any code could be executed here!" is printed to the console. You can substitute the line that prints the message (see CustomInstaller.cs file) for any custom code you want to execute during installation.

You can execute different custom code while uninstalling:

```
> InstallUtil.exe /u CustomAction.exe
```

Example

VB CustomAction.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Assemblies

Use strong names
Use delay signing
Work with the global cache
Change the search path
Use assembly version policy
Create publisher policy

Back to Index

Get URL for this page

How Do I...Use Assembly Version Policy?

A primary goal of the deployment system in the .NET Framework is to eliminate conflicts between applications caused by shared components and shared states (or DLL conflicts). A key solution to this problem is a robust versioning system. The .NET Framework records information about an application's dependencies in the assembly manifest. This dependency information includes a version number that is used at runtime to load the proper version of a dependency.

By default, the common language runtime will load the version of a dependency that is specified in the manifest. This is preferred in the majority of scenarios. However, there are cases where running an application with a different version of a dependency can be useful. In order to accomplish this, version policies can be included in an application's configuration file. For example, the following XML code fragment redirects references to version 5.0.0.0 of a shared assembly called "caclR" up to version 6.0.0.0 of that assembly:

```
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v

            <dependentAssembly>
                <assemblyIdentity name="calcR"
                                  publicKeyToken="a1690a5ea44ba
                                  culture=""/>

                <bindingRedirect oldVersion="5.0.0.0"
                                 newVersion="6.0.0.0"/>

            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

In addition to policies specified at the application level, the .NET Framework also provides two other policy levels: publisher and administrator. A Publisher policy statement describes the compatibility of an assembly issued by the publisher of that shared assembly. See How Do I ... Create Publisher Policy for details. Administrator level policy is a machine version policy that affects all applications. Administrator policy is created using the same XML syntax as application level policy. The administrator policy file is called machine.config, and resides in the common language runtime install directory. The three policy levels are evaluated in the following order:

- Application Policy
- Publisher Policy
- Administrator Policy

Example

C# Math.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Send a message to a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to send a simple message to a message queue. To run the sample, you have to have Message Queuing installed on your system. The sample is a command-line application that takes two command-line arguments. The first argument is a name of a message queue. The path does not have to refer to an existing queue. If the queue does not exist, it will be created for you. The second command-line argument is the message to be written to the queue. For example you can run it as follows:

```
MQSend.exe MyQueue "Hello World!"
```

Now, open the Message Queuing console and examine the MyQueue message queue. It should have a message in it.

In its simplest form, sending a message to a message queue involves:

1. Creating an instance of the **MessageQueue** component and setting its **Path** property:

   ```
   Dim mq As MessageQueue = New MessageQueue(".\MyQueue")
   ```

   VB

2. Calling **Send** method to send the message:

   ```
   mq.Send("Hello World!")
   ```

   VB

Example

VB MQSend.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Receive a message from a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to receive a message from a message queue. To run the sample you have to have Message Queuing installed on your system. The sample is a command-line application that takes one command-line argument. The argument is the name of a public message queue on your local machine. For example, you can run it as follows (first send a message to the queue using the MQSend sample):

```
> MQReceive.exe MyQueue
```

The sample application will receive and output to the console the first message from the queue. In its simplest form, receiving a message from a message queue involves:

1. Creating an instance of the **MessageQueue** component, setting its **Path**, and initializing the Formatter's **TargetTypeNames** property:

```
Dim mq As MessageQueue = new MessageQueue(mqPath)
Dim formatter As XmlMessageFormatter = CType(mq.Formatter,XmlMessageFormatter)
formatter.TargetTypeNames = new String(){"System.String"}
```

VB

2. Calling **Receive** to receive the message:

```
Dim m As Message = mq.Receive(New TimeSpan(0,0,3))
Console.WriteLine(CStr(m.Body))
```

VB

Example

VB MQReceive.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Pass a complex type through a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue component** to send a complex type through a message queue. To run the sample you have to have Message Queuing installed on your system.

Sending a complex type through a queue is very simple. It is actually as simple as sending a string (see MQSend sample). The **MessageQueue** component handles everything automatically. All you have to do is:

1. Create an instance of the **MessageQueue** component and set its **Path** property:

```
Dim mq As MessageQueue = New MessageQueue(".\MyQueue")
```

VB

2. Calling **Send** method to send a complex object:

```
Dim myType As MyComplexType = New MyComplexType()
mq.Send(myType)
```

VB

Example

VB SendObject.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Messaging
Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Receive asynchronously?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to watch a message queue for arrival of new messages. To run the sample you have to have Message Queuing installed on your system. The sample is a command-line application that takes one command-line argument. The argument is the name of a public message queue on your local machine. For example you can run it as follows:

```
> MQAsync.exe MyQueue
```

Now, use the MQSend sample to send a message to the MyQueue queue. The sample application will be notified when the message arrives at the queue and will output the message to the console.

In its simplest form, asynchronously receiving a message from a message queue involves:

1. Creating an instance of the **MessageQueue** component, and setting its **Path** property and the formatter:

```
Dim mq As MessageQueue = New MessageQueue(".\MyQueue")
Dim formatter As XmlMessageFormatter = CType(mq.Formatter,XmlMessageFormatter)
formatter.TargetTypeNames = new String(){"System.String"}
```

VB

2. Setting up an event handler:

```
AddHandler mq.ReceiveCompleted, New ReceiveCompletedEventHandler(AddressOf OnReceiveCompleted)
```

VB

3. Implementing an event handler:

```
Public Shared Sub OnReceiveCompleted(source As Object, asyncResult As ReceiveCompletedEventArgs)
    Dim mq As MessageQueue = CType(source,MessageQueue)
    Dim m As Message = mq.EndReceive(asyncResult.AsyncResult)
    Console.WriteLine("Message: " + CStr(m.Body))
    mq.BeginReceive()
End sub
```

VB

4. Calling **BeginReceive** to start an asynchronous receive operation:

```
mq.BeginReceive
```

VB

Please note that **BeginReceive** will receive only one message. If you want to keep receiving messages, you have to call **BeginReceive** again (see the event handler implementation in step 3).

Example

VB MQAsync.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Use message formatters?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

The sample illustrates how to use the **MessageQueue** component to use different serialization mechanisms to send objects through a message queue. To run the sample you have to have Message Queuing installed on your system.

Selecting a serialization mechanism is as simple as changing the value of one property. Here are some examples:

1. Send a message using an XML based serialization mechanism:

```
mq.Formatter = New XmlMessageFormatter()
mq.Send(order)
```

VB

2. Send a message using a binary serialization mechanism:

```
mq.Formatter = New BinaryMessageFormatter()
mq.Send(order)
```

VB

3. Send a message using a serialization mechanism which is compatibile with the MSMQ ActiveX Formatter:

```
mq.Formatter = New ActiveXMessageFormatter()
mq.Send(order)
```

VB

Example

VB Formatters.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Messaging

Send a queued message
Receive a queued message
Pass a complex type
Receive asynchronously
Use message formatters
Control a message queue

Back to Index

Get URL for this page

How Do I...Control a message queue?

Message queuing makes it easy for application developers to communicate with application programs quickly and reliably by sending and receiving messages. Messaging provides you with guaranteed message delivery and a robust, fail-safe way to carry out many of your business processes.

The **MessageQueue** component allows you to easily incorporate message-based communication into your applications. Using this component and its associated language features, you can send and receive messages, explore existing queues, create and delete queues, and perform a variety of other operations using a simple programming model.

This sample illustrates how to change some properties of a message queue. It is a small console application that can be run from a command prompt. Run the sample without any command-line arguments and it will print its full usage instructions.

For example, if you want change a queue's label, run:

```
> MQCtrl.exe MyQueue l "New Label"
```

In its simplest form, changing a property of a message queue involves:

1. Creating a new instance of a **MessageQueue** component and pointing it to an existing queue:

   ```
   Dim mq As MessageQueue = New MessageQueue(mqPath)
   ```

   VB

2. Setting the property to a new value:

   ```
   mq.Label = "New Label"
   ```

   VB

Example

VB MQCtrl.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Networking

Make a GET request
Make an SSL GET request
Make an NTLM GET request
Make a proxy GET request
Make an async GET request
Make a GET request in ASP.NET
Async GET request in ASP.NET
Handle WebExceptions
Make a POST request
Write a TCPClient client
Write a TCPListener server
Write a UDPClient client

Back to Index

Get URL for this page

How Do I...Make a GET request asynchronously?

This sample illustrates how to use the **WebRequest** and **WebResponse** classes to make a GET request on a URI asynchronously.

This sample is similar to the other **WebRequest** samples, with the difference found in the Begin and End async-style function calls. Put simply, instead of calling **GetResponse**, a call is made to the function **BeginGetResponse**, with the callback function sent as a parameter.

This function **RespCallback**, takes an object of type **IASyncResult**. This object contains an object **AsyncState** that can be cast as a **HttpWebRequest**. Calling this casted objects **EndGetResponse** method will return a WebResponse object (which is cast as an HttpWebResponse in this case.)

From here, this example is like the others. It pulls out the response from the stream and writes it out to the console. However, this example is using the **StreamReader** and **StreamWriter** classes to do this.

Example

VB ClientGETasync.exe

[Run Sample] | [View Source]

Please enter a URI - [ ] Submit

Please enter a URI - Submit

This is a short demonstration which illustrates how to find specific member information for a given type. It uses classes and methods from the Reflection namespace to look at the actual code you develop with.

We have selected a few common types for you to start your searching with as a suggestion, but you can enter a type of your own, or a different System type if you wish. Remember that if you do enter a type rather than selecting one from the list, you should give the full namespace and class name for your type (the only exception is types inside the System namespace). When looking through the information returned, you are presented with the datatype returned (if applicable), and in the case of methods, you also get the signature for that method. Where the member is implemented or inherited from another object, you will also see the full namespace for that member, in its original location. Happy Searching!

Please select the datatype you wish to search on: Select A Datatype Run Query

Please include the following members: ☐ Fields ☐ Properties ☐ Methods ☐ Events ☐ Constructors

Please include the following styles: ☑ Instance ☐ Static

The following example illustrates how to use the reflection classes to dynamically invoke methods. The InvokeMember method on the Type object is used to specify the name of a method to invoke, as well as its style (instance/static) and any parameters required. In this demonstration, you can also learn how to dynamically invoke fields and properties, or use the MemberInfo object to dynamically invoke a method requiring parameters to be passed by reference. Please see the code creating this page to see how the output was achieved.

```
Invoking a Static method
--------------------------------
Hello

Invoking an Instance method
--------------------------------
AddUp Called 1 times
AddUp Called 2 times

Invoking a method with arguments/parameters
--------------------------------
100.09 + 184.45 = 284.54

Invoking a Field (Setting and Getting)
--------------------------------
Name == initalName
Name == NewName

Invoking an indexed Property (Setting and Getting)
--------------------------------
Item[3] == 3
Item[3] == NewValue

Getting a field or property
--------------------------------
Name == NewName
Value == the value

Invoking a method with named arguments
--------------------------------
Mouse,Micky

Invoking a default member of a type
--------------------------------
2/4/2002 5:22:51 PM

Invoking a Method with ByRef parameters
--------------------------------
2, 1
```

Regular Expressions
Match a pattern
Get multiple matches
Make replacements
Find common patterns

Back to Index

Get URL for this page

How Do I...Use Regular Expressions to match a pattern?

Regular Expressions allow for easy parsing and matching of strings to a specific pattern. Using the objects available in the **RegularExpressions** namespace, it is possible to compare a string against a given pattern, replace a string pattern with another string, or to retrieve only portions of a formatted string.

In the simplest case, Regular Expressions can be used to do string comparisons against a pattern of strings. For instance, it can be useful to only allow strings of a particular length range, especially when accepting passwords. The following code demonstrates creating a **Regex** and testing to see if the string matches the pattern using the **IsMatch** method.

```
Dim emailregex As Regex = New Regex("(?<user>[^@]+)@(?<host>.+)");
Dim ismatch As Boolean = emailregex.IsMatch("johndoe@tempuri.org");
```

VB

This sample illustrates how to create a **Regex** object using a pattern string. The **Match** method is called and a **Match** object is returned. By examining the **Success** property, the sample decides whether to continue processing the **Match** object or to print an error message. If the match is successful, the **Groups** collection of the **Match** object can be queried for ordinal or named groups within the match. The following sample illustrates using named groups in a pattern match to validate an email address and then print the user and host portions.

Example

VB RegexMatch.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Regular Expressions
Match a pattern
Get multiple matches
Make replacements
Find common patterns

Back to Index

Get URL for this page

How Do I...Get all matches for a pattern

Regular Expressions are often useful when trying to retrieve small portions of text from a large document, result set, or when filtering a stream. The **MatchCollection** object contains all valid **Match** objects for a given regular expression after a successful match occurs.

```
Dim DigitRegex As Regex = New Regex("(?<number>\d+)")
Dim S As String = "abc 123 def 456 ghi 789"
  ...
Dim Mc As MatchCollection = DigitRegex.Matches(S)
```

VB

The following example illustrates how to create a **Regex** that matches numbers in a string. The **Matches** method is callled to return a **MatchCollection**. If the **Count** property equals 0, no successful matches have occured. If there are matches, the results of each are displayed.

Example

VB RegexMatches.exe

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Regular Expressions
Match a pattern
Get multiple matches
Make replacements
Find common patterns

Back to Index

Get URL for this page

How Do I...Use Regular Expressions to make replacements?

The Regular Expressions library can often ease the time it takes to generate string replacement functions. By specifying a pattern of strings to be replaced, you do not have to search for every possible variation of a string. Once a **Regex** object that matches every possible string to be replaced is created, the **Replace** method can be used to generate a result. The **Replace** method can be most easily used by passing in the source string and the replacement string. The **Replace** method will return the results as a String.

```
Dim digitregex As Regex = new Regex("(?<digit>[0-9])")
Dim Before As String = "Here is so4848me te88xt with emb4493edded numbers."
 ...
Dim After As String = DigitRegex.Replace(Before, "")
```

VB

You can also reuse the matched string in the replacement. In the previous snippet we have a named capture of `?<digit>`. This named capture can be reused in the replacement string as `${digit}`. Note that ordinal captures can be used as well, `$123`, which would evaluate to 123 captures in our pattern.

This example illustrates how to use the **Replace** method of **Regex** to remove all digits from the input string.

Example

VB RegexReplace.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Regular Expressions

Match a pattern

Get multiple matches

Make replacements

Find common patterns

Back to Index

Get URL for this page

How Do I...Use common Regular Expressions?

There are a variety of tasks that regular expressions can be used for in regards to input validation and file parsing. Some of the most common expressions are often hard to remember because of the power and options available. Here is a list of commonly used regular expressions:

Use	Expression
Social Security Number	\d{3}-\d{2}-\d{4}
US Phone Number	((\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}
US Postal Code	\d{5}(-\d{4})?
Internet EMail Address	[\w-]+@([\w-]+\.)+[\w-]+
Internet URL	http://([\w-]\.)+[\w-](/[\w-./?%=]*)?
Simple Password (digit)	^(?=.*\d).{4,8}$
Advanced Password (upper, lower, digit)	^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{4,8}$
Common File masks	^(.+)\\(.+)\.(.+)
Major Credit Card	\d{4}-?\d{4}-?\d{4}-?\d{4}

Timers
Get timer events

Back to Index

Get URL for this page

How Do I...Get timer events without message pump?

Timer allows you to raise an event on a specified interval. This sample illustrates how to use a **Timer** to raise an event every 1000 milliseconds. Run the sample and observe a new "Hello World" message being printed every second. In its simplest form, using a **Timer** involves:

1. Creating a new instance of **Timer**:

```
Dim aTimer As System.Timers.Timer = New System.Timers.Timer()
```

VB

2. Specifying the event handler:

```
AddHandler aTimer.Tick, AddressOf OnTimer
```

VB

3. Specifying how often to raise the event:

```
aTimer.Interval = 1000
```

VB

4. Enabling the component: aTimer.Enabled = true;
5. Handling the event:

```
Public Shared Sub OnTimer(ByVal source As Object, ByVal e As EventArgs)
    Console.WriteLine("Hello World!")
End Sub
```

VB

Example

VB Timer.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Windows Forms
A default error handler
An Owner-drawn ListBox
Call across thread boundaries

Back to Index

Get URL for this page

Adding an error handler for untrapped exceptions

In your application design you might want to register an exception handler for any exceptions you have not handled in your application. You can create an exception handler for your unhandled exceptions by calling the **Application.AddOnThreadException** method and passing in a reference to a method in your application. This method will then be called for any unhandled exceptions on that thread. The **OnThreadException** event is raised when the Windows Forms window procedure receives an exception.

Windows Forms provides a default exception handler and exception handling dialog named **ThreadExceptionDialog**. This dialog can be used during development, but you will want to supply your own custom exception handlers for your application when it is near completion.

The following example demonstrates how to create a very simple custom exception handler that displays the exception message and stack trace in a **MessageBox**:

```
' The Error Handler class
' We need a class because event handling methods can't be static
Friend Class CustomExceptionHandler

    ' Handle the exception event
    Public Sub OnThreadException(Sender As Object , t As ThreadExceptionEventArgs)

        Dim result As DialogResult = DialogResult.Cancel
        Try
            result = this.ShowThreadExceptionDialog(t.Exception)

        Catch Exc As Exception
            Try
                MessageBox.Show("Fatal Error", _
                                "Fatal Error", _
                                MessageBoxButtons.AbortRetryIgnore, _
                                MessageBoxIcon.Stop)

            Finally
                Application.Exit()
            End Try
        End Try

        If result = DialogResult.Abort Then Application.Exit()

    End Sub

    ' The simple dialog that is displayed when this class catches and exception
    Private Function ShowThreadExceptionDialog(e As Exception) As DialogResult
        Dim errorMsg As String = "An error occurred please contact the adminstrator with" & _
                                 " the following information:" & vbCrLf & vbCrLf
        errorMsg &= e.Message & vbCrLf & vbCrLf & "Stack Trace:" & _
                    vbCrLf & e.StackTrace
        Return MessageBox.Show(errorMsg, _
                               "Application Error", _
                               MessageBoxButtons.AbortRetryIgnore, _
                               MessageBoxIcon.Stop)
    End Function
End Class

....

' Register the custom error handler as soon as we can in Main
' to make sure that we catch as many exceptions as possible
Shared Sub Main()
    Dim eh As New CustomExceptionHandler()
    Application.AddOnThreadException(New ThreadExceptionEventHandler(AddressOf eh.OnThreadException))
    Application.Run(New ErrorHandler())
End Sub
```

VB

VB Error Handling Sample

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Windows Forms

A default error handler

An Owner-drawn ListBox

Call across thread boundaries

Back to Index

Get URL for this page

Creating an Owner-Drawn Listbox

By default, Listboxes display strings. If you want to display something else, such as a bitmap for each item or paint each item in the list box in a different color, then you will need to draw each item in your code. This is known as an owner-drawn ListBox.

In order to put a ListBox into owner-drawn mode, you must set the **DrawMode** property to **DrawMode.OwnerDrawVariable** or **DrawMode.OwnerDrawFixed**. OwnerDrawFixed indicates that the ListBox will be owner-drawn and each item in the ListBox will be the same height. **OwnerDrawVariable** indicates that the ListBox will be owner-drawn and each item in the ListBox can be a different height.

There are two events that you need to handle for an owner-drawn list box:

DrawItem	This event is raised every time the list box needs to paint an item. The handler for this event is passed a **DrawItemEventArgs** object. This object contains the information needed to draw the ListBox item including a **Graphics** object, the index of the item to be painted, the state of the item to be painted, and the bounds of the item to be painted. It also includes utility methods for painting the background of the item and its focus rectangle.
MeasureItem	This event is raised every time a ListBox item is created and the size of the list item needs to be determined. You only need to handle this event if the Listbox DrawMode is **OwnerDrawVariable**. The handler for this event is passed a **MeasureItemEventArgs** object. This object contains the information needed to determine the size of the item, a **Graphics** object, and the index of the item to be measured. To set the height and width of each item use the **ItemHeight** and **ItemWidth** properties on the **MeasureItemEventArgs** object.

The following example demonstrates how to create an owner-drawn ListBox that displays each item in the ListBox in a different color:

```
....

' Set the DrawMode and add the event handlers
Me.listBox1.DrawMode = System.Windows.Forms.DrawMode.OwnerDrawVariable
AddHandler Me.listBox1.DrawItem, AddressOf Me.listBox1_DrawItem
AddHandler Me.listBox1.MeasureItem, AddressOf Me.listBox1_MeasureItem
....

'Return the height of the item to be drawn
Private Sub listBox1_MeasureItem(ByVal sender As Object, ByVal e As MeasureItemEventArgs)
    Dim displayText As String
    Dim stringSize As SizeF

    'Work out what the text will be
    displayText = "ITEM #" & e.Index

    'Get width & height of string
    stringSize = e.Graphics.MeasureString(displayText, Me.Font)

    'Account for top margin
    stringSize.Height += 6

    If listBoxHeights(e.Index) > stringSize.Height Then
        'Now set height to taller of default and text height
        e.ItemHeight = listBoxHeights(e.Index)
    Else
        e.ItemHeight = CInt(stringSize.Height)
    End If
End Sub

....

Private Sub listBox1_DrawItem(ByVal sender As Object, ByVal e As DrawItemEventArgs)
    Dim brush As Brush
    Dim selected As Boolean
    Dim displayText As String

    ' The following method should generally be called before drawing.
    ' It is actually superfluous here, since the subsequent drawing
    ' will completely cover the area of interest.
    e.DrawBackground()

    'The system provides the context
    'into which the owner custom-draws the required graphics.
    'The context into which to draw is e.graphics.
```

```
        'The index of the item to be painted is e.Index.
        'The painting should be done into the area described by e.Bounds.
        brush = listBoxBrushes(e.Index)
        e.Graphics.FillRectangle(brush, e.Bounds)
        e.Graphics.DrawRectangle(SystemPens.WindowText, e.Bounds)

        If (e.State BitAnd DrawItemState.Selected) = DrawItemState.Selected Then
            selected = True
        Else
            selected = False
        End If

        displayText = "ITEM #" & e.Index

        If (selected) Then
            displayText &= " SELECTED"
        End If

        e.Graphics.DrawString(displayText, Me.Font, Brushes.Black, e.Bounds.X, e.Bounds.Y)

        e.DrawFocusRectangle()
    End Sub

    ....
```

VB

VB OwnerDraw ListBox Sample

LateBreaking Samples:

- C++

[Run Sample] | [View Source]

Windows Forms
A default error handler
An Owner-drawn ListBox
Call across thread boundaries

Back to Index

Get URL for this page

Making procedure calls across thread boundaries

Windows Forms controls can only execute on the thread on which they were created, that is, they are not thread-safe. If you want to get or set properties, or call methods, on a control from a background thread, the call must be marshaled to the thread that created the control.

There are five functions on a control that are safe to call from any thread: **InvokeRequired**, **Invoke**, **BeginInvoke**, **EndInvoke** and **CreateGraphics**. For all other method calls, you should use one of the invoke methods.

By default, Windows marshals the calls for you. However, if you are making multiple calls to a control, it is much more efficient to create a method that executes those calls and make the one cross-thread call yourself. You make the cross-thread call by calling one of the **Control.Invoke** methods. The **Invoke** methods take a reference to a delegate. Typically, this delegate is an instance of the **MethodInvoker** delegate.

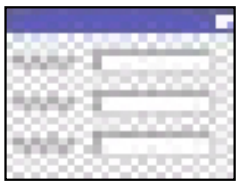
InvokeRequired	public bool InvokeRequired { get ; }
	Returns true if the caller must call Invoke when making method calls to this control.
BeginInvoke	public IAsyncResult BeginInvoke(Delegate method) public IAsyncResult BeginInvoke(Delegate method, Object[] args)
	Executes the given delegate on the thread that owns this Control's underlying window handle. The delegate is called asynchronously and this method returns immediately. You can call this from any thread, even the thread that owns the control's handle. If the control's handle does not exist yet, this will follow up the control's parent chain until it finds a control or form that does have a window handle. If no appropriate handle can be found, **BeginInvoke** will throw an exception. Exceptions within the delegate method are considered untrapped and will be sent to the application's untrapped exception handler.
EndInvoke	public Object EndInvoke(IAsyncResult asyncResult)
	Retrieves the return value of the asynchronous operation represented by the **IAsyncResult** interface passed. If the async operation has not been completed, this function will block until the result is available.
Invoke	public Object Invoke(Delegate method) public Object Invoke(Delegate method, Object[] args)
	Executes the given delegate on the thread that owns this control's underlying window handle. The delegate is called synchronously and this method returns once the invoked method has returned. The return value is the result of the invoked method. It is an error to call this on the same thread that the control belongs to.

The following example demonstrates how to create a background thread that uses a **MethodInvoker** to update a **ProgressBar** control at regular intervals:

```
....

' Start the background thread
timerThread = new Thread(new ThreadStart(AddressOf ThreadProc))
timerThread.IsBackground = True
timerThread.Start()

....

' This function is executed on a background thread - it marshalls calls to
' update the UI back to the foreground thread
Public Sub ThreadProc()

    Try
```

```
            Dim mi As New MethodInvoker(AddressOf Me.UpdateProgress)
            Do While True

                ' Call BeginInvoke on the Form
                Me.BeginInvoke(mi)
                Thread.Sleep(500)
            Loop

        ' Thrown when the thread is interupted by the main thread - exiting the loop
        Catch e As ThreadInterruptedException
            ' Simply exit....

        Catch we As Exception
        End Try
    End Sub

    ....

    ' This function is called from the background thread
    Private Sub UpdateProgress()

        ' Reset to start if required
        If progressBar1.Value = progressBar1.Maximum
            progressBar1.Value = progressBar1.Minimum
        End If


        ' Reset to start if required
        progressBar1.PerformStep()
    End Sub

    ....

    ' Make sure we clean up the background thread in Dispose
    Public Override Sub Dispose()
       If Not timerThread Is Nothing Then
           timerThread.Interrupt()
           timerThread = Nothing
       End If

       MyBase.Dispose()
    End Sub
```

VB



VB Threading Sample

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

XML Framework in .NET

Extensible Markup Language (XML) is a meta-markup language that provides a format for describing structured data. XML enables a new generation of Web-based data viewing and manipulation applications. XML is the universal language for data on the Web. XML gives developers the power to deliver structured data from a wide variety of applications to the desktop for local computation and presentation.

System.Xml Namespace

This set of topics cover the usage of the XML classes in the System.Xml namespace.

This namespace has a comprehensive set of XML classes for parsing, validation, and manipulation of XML data using readers, writers, and World Wide Web Consortium (W3C) DOM-compliant components. It also covers XML Path Language (XPath) queries and Extensible Stylesheet Language Transformations (XSLT). The following list contains the main classes in the XML namespace:

- The **XmlTextReader** class provides fast, non-cached, forward only read access to XML data.
- The **XmlNodeReader** class provides an XmlReader over the given DOM node subtree.
- The **XmlValidatingReader** class provides DTD, XDR and XSD Schema validation.
- The **XmlTextWriter** class provides a fast, forward only way of generating XML.
- The **XmlDocument** class implements the W3C Document Object Model level 1 Core and the Core DOM Level 2
- The **XmlDataDocument** class provides an implementation of an XmlDocument that can be associated with a DataSet. Structured XML can be viewed and manipulated simultaneously through the DataSet's relational representation or the XmlDataDocument's tree representation. See the topic How Do I...Use XML and the DataSet class?
- The **XPathDocument** class provides a fast and performant cache for XML document processing for XSLT.
- The **XPathNavigator** class provides a W3C XPath 1.0 data model over a store with a cursor style model for navigation.
- The **XslTransform** class is a W3C XSLT 1.0 specification compliant XSLT processor for transforming XML documents.
- The **XmlSchema** Object Model classes provide a navigable set of classes which directly reflect the W3C XSD specification. They provide the ability to programmatically create an XSD schema.
- The **XmlSchemaCollection** class provides a library of XDR and XSD schemas. These schemas, cached in memory provide fast, parse-time validation for the XmlValidatingReader.

World Wide Web Consortium (W3C) Conformance

The classes in this namespace conform to the World Wide Web Consortium (W3C) specifications and so provide a comprehensive set of XML tools for building Web applications and any scenario which relies upon XML data exchange.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

XML Framework in .NET

Extensible Markup Language (XML) is a meta-markup language that provides a format for describing structured data. XML enables a new generation of Web-based data viewing and manipulation applications. XML is the universal language for data on the Web. XML gives developers the power to deliver structured data from a wide variety of applications to the desktop for local computation and presentation.

System.Xml Namespace

This set of topics cover the usage of the XML classes in the System.Xml namespace.

This namespace has a comprehensive set of XML classes for parsing, validation, and manipulation of XML data using readers, writers, and World Wide Web Consortium (W3C) DOM-compliant components. It also covers XML Path Language (XPath) queries and Extensible Stylesheet Language Transformations (XSLT). The following list contains the main classes in the XML namespace:

- The **XmlTextReader** class provides fast, non-cached, forward only read access to XML data.
- The **XmlNodeReader** class provides an XmlReader over the given DOM node subtree.
- The **XmlValidatingReader** class provides DTD, XDR and XSD Schema validation.
- The **XmlTextWriter** class provides a fast, forward only way of generating XML.
- The **XmlDocument** class implements the W3C Document Object Model level 1 Core and the Core DOM Level 2
- The **XmlDataDocument** class provides an implementation of an XmlDocument that can be associated with a DataSet. Structured XML can be viewed and manipulated simultaneously through the DataSet's relational representation or the XmlDataDocument's tree representation. See the topic How Do I...Use XML and the DataSet class?
- The **XPathDocument** class provides a fast and performant cache for XML document processing for XSLT.
- The **XPathNavigator** class provides a W3C XPath 1.0 data model over a store with a cursor style model for navigation.
- The **XslTransform** class is a W3C XSLT 1.0 specification compliant XSLT processor for transforming XML documents.
- The **XmlSchema** Object Model classes provide a navigable set of classes which directly reflect the W3C XSD specification. They provide the ability to programmatically create an XSD schema.
- The **XmlSchemaCollection** class provides a library of XDR and XSD schemas. These schemas, cached in memory provide fast, parse-time validation for the XmlValidatingReader.

World Wide Web Consortium (W3C) Conformance

The classes in this namespace conform to the World Wide Web Consortium (W3C) specifications and so provide a comprehensive set of XML tools for building Web applications and any scenario which relies upon XML data exchange.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Read XML from a file?

This sample illustrates how to read XML from a file using the XmlTextReader class. This class provides direct parsing and tokenizing of XML, and implements the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML specifications.

VB ReadXmlFile.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

XmlReader class is the API that provides XML parsing, the XmlTextReader is the implementation designed to handle byte streams.

Typically, you use the XmlTextReader if you need to access the XML as raw data without the overhead of a DOM. Not having to access the DOM results in a faster way to reading XML. For example, an XML document could have a header section used for routing the document for processing elsewhere. The XmlTextReader has different constructors to specify the location of the XML data. This sample loads XML from the books.xml file, as shown in the following code.

```
Dim reader As XmlTextReader = New XmlTextReader ("books.xml")
```

VB

Once loaded, the XmlTextReader moves across the XML data by using the Read method sequentially retrieving the next record from the document. The Read method returns false if there are no more records.

```
Do While (reader.Read())
    ' Do some work here on the data
    ...
Loop
```

VB

To process the XML data, each record has a node type that can be determined from the NodeType property. After the NodeType enumeration returns the node type, the sample tests the node type to see whether it is either an element or document type. If the node is either one of these two types, the sample processes the node using the Name and Value properties to display details about the node. The Name property returns the node name (for instance, the element and attribute names, while the Value property returns the node value (node text) of the current node (record).

```
Do While (reader.Read())
    Select Case reader.NodeType
        Case XmlNodeType.Element ' The node is an Element
            Console.Write("<" + reader.Name)
            while (reader.MoveToNextAttribute()) ' Read attributes
                Console.Write(" " + reader.Name + "='" + reader.Value + "'")
            end while
            Console.Write(">")

        Case XmlNodeType.DocumentType ' The node is a DocumentType
            Console.WriteLine(NodeType & "<" & reader.Name & ">" & reader.Value);
        ...
    End Select
Loop
```

VB

The XmlNodeType returned depends on the XmlReader class being used. For example, the XmlTextReader class never returns an XmlNodeType that is a Document, DocumentFragment, Entity, EndEntity and Notation node. See the .NET Framework Class Library for details on what XmlNodeType are returned by each XmlReader class.

The node types specified for the XmlNodeType are equivalent to the W3C DOM node types, with some extended types required for forward-only reading.

XmlNodeType Enumeration Member	Description

Attribute	An Attribute. Example XML: id='123'. An Attribute node can have the following child node types: Text and EntityReference. The Attribute node does not appear as the child node of any other node type; note that it is not considered a child node of an Element.
CDATA	A CDATA section. Example XML: <![CDATA[my escaped text]]> CDATA sections are used to escape blocks of text that would otherwise be recognized as markup. A CDATASection node cannot have any child nodes. The CDATASection node can appear as the child of the DocumentFragment, EntityReference, and Element nodes.
Comment	A Comment. Example XML: <!-- my comment --> A Comment node cannot have any child nodes. The Comment node can appear as the child of the Document, DocumentFragment, Element, and EntityReference nodes.
Document	A document object, which, as the root of the document tree, provides access to the entire XML document. A Document node can have the following child node types: Element (maximum of one), ProcessingInstruction, Comment, and DocumentType. The Document node cannot appear as the child of any node types.
DocumentFragment	A document fragment. The DocumentFragment node associates a node or subtree with a document without actually being contained within the document. A DocumentFragment node can have the following child node types: Element, ProcessingInstruction, Comment, Text, CDATASection, and EntityReference. The DocumentFragment node cannot appear as the child of any node types.
DocumentType	The document type declaration, indicated by the <!DOCTYPE> tag. Example XML: <!DOCTYPE ...> A DocumentType node can have the following child node types: Notation and Entity. The DocumentType node can appear as the child of the Document node.
Element	An Element. Example XML: <Name> An Element node can have the following child node types: Element, Text, Comment, ProcessingInstruction, CDATA, and EntityReference. The Element node can be the child of the Document, DocumentFragment, EntityReference, and Element nodes.
EndElement	Returned when XmlReader gets to the end of an element. Example XML: </foo>
EndEntity	Returned when XmlReader gets to the end of the entity replacement as a result of a call to ResolveEntity.
Entity	An entity declaration. Example XML: <!ENTITY ...> An Entity node can have child nodes that represent the expanded entity (for example, Text and EntityReference nodes). The Entity node can appear as the child of the DocumentType node.
EntityReference	A reference to an entity. Example XML: &foo; This applies to all entities, including character entity references. An EntityReference node can have the following child node types: Element, ProcessingInstruction, Comment, Text, CDATASection, and EntityReference. The EntityReference node can appear as the child of the Attribute, DocumentFragment, Element, and EntityReference nodes.
None	This is returned by the XmlReader if a Read method has not been called.
Notation	A notation in the document type declaration. Example XML: <!NOTATION ...> A Notation node cannot have any child nodes. The Notation node can appear as the child of the DocumentType node.
ProcessingInstruction	A processing instruction (PI). Example XML: <?pi test?> A PI node cannot have any child nodes. The PI node can appear as the child of the Document, DocumentFragment, Element, and EntityReference nodes.
SignificantWhitespace	Whitespace between markup in a mixed content model, or whitespace within the xml:space= "preserve" scope.

Text	The text content of an element. A Text node cannot have any child nodes. The Text node can appear as the child node of the Attribute, DocumentFragment, Element, and EntityReference nodes.
Whitespace	Whitespace between markup.
XmlDeclaration	The XML declaration node. Example XML: <?xml version='1.0'?> This has to be the first node in the document. It can have no children. It is a child of the root node. It can have attributes that provide version and encoding information.

The Depth property returns the depth of the current node in the XML document and is useful when formatting data. A node at the root level of the document is at depth 0. Combining this depth information with the Name and Value properties, you can create a sample that processes an XML file and formats the output depending on the node type and the depth, gathering statistics as it reads. The following sample code shows how this basic formatting is possible using the Format method. To see the full sample code, see View Source.

```
private shared Sub Format(byref reader as XmlTextReader , nodeType as String)

    ' Format the output
    Console.Write(reader.Depth & " ")
    Console.Write(reader.AttributeCount & " ")

    Dim i as Integer
    for i = 0 to reader.Depth
        Console.Write(Strings.chr(9))
    Next

    Console.Write(nodeType & "<" & reader.Name & ">" & reader.Value)

    'Display the attributes values for the current node
    if (reader.HasAttributes)
        Console.Write(" Attributes:")
        Dim j as Integer
        for j = 0 to reader.AttributeCount - 1
            Console.Write(" [{0}] " & reader(j), j)
        next
    End if

    Console.WriteLine()
End Sub
```

VB

The preceding sample code uses the HasAttributes property to test whether an element node has any attribute nodes, and then uses the node index operators to retrieve each attribute value. This is analogous to a collection of attributes for the node. In addition, the preceding code sample uses the AttributeCount property to return the number of attributes for the current node. Use this approach if all you want are the attribute values and are not concerned with other properties of the attribute nodes (for example, the name of the attribute).

Note: For an alternative approach to accessing attributes by moving to each attribute node, and reading the both the node's name and value, see How Do I...Read XML from a stream?.

The following output results from running this sample with the books.xml file. In this output, the first column is the Depth property and the second column is the AttributeCount property.

```
0 0 XmlDeclaration<xml>version='1.0'
0 0 Comment<> This file represents a fragment of a book store inventory database
0 0 Element<bookstore>
1 3     Element<book> Attributes: [0] autobiography [1] 1981 [2] 1-861003-11-0
2 0             Element<title>
3 0                     Text<>The Autobiography of Benjamin Franklin
2 0             Element<author>
3 0                     Element<first-name>
4 0                             Text<>Benjamin
3 0                     Element<last-name>
4 0                             Text<>Franklin
2 0             Element<price>
3 0                     Text<>8.99
1 3     Element<book> Attributes: [0] novel [1] 1967 [2] 0-201-63361-2
2 0             Element<title>
3 0                     Text<>The Confidence Man
2 0             Element<author>
3 0                     Element<first-name>
4 0                             Text<>Herman
3 0                     Element<last-name>
4 0                             Text<>Melville
2 0             Element<price>
```

```
3 0                    Text<>11.99
1 3     Element<book> Attributes: [0] philosophy [1] 1991 [2] 1-861001-57-6
2 0             Element<title>
3 0                    Text<>The Gorgias
2 0             Element<author>
3 0                    Element<name>
4 0                           Text<>Plato
2 0             Element<price>
3 0                    Text<>9.99

Statistics for books.xml file

XmlDeclaration: 1
ProcessingInstruction: 0
DocumentType: 0
Comment: 1
Element: 18
Attribute: 9
Text: 11
Whitespace: 27
```

Summary

1. The XmlTextReader provides fast, non-cached, forward-only read access to XML data.
2. The XmlTextReader implements the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML specifications.
3. The XmlTextReader provides constructors to read XML from a file, a stream, or a TextReader.
4. The Read method moves the reader sequentially as a node.
5. For element nodes, the value of an attribute can be obtained by using the index operators.
6. Attributes are represented as a node list off the current node and can be discovered through the HasAttributes property.
7. The Depth property reports the depth of the current node and can be useful for formatting. Nodes at the root level are at depth 0.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace

How Do I...Read XML data from a stream?

This sample illustrates how to read XML from a stream using the XmlTextReader class. The stream could have come from a variety of sources such as a byte stream from a server, a file, or from a TextReader.

Note: This sample follows on from the How do I...Read XML from a file? topic.

VB ReadXmlStream.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The XmlTextReader has different constructors to specify the location of the XML data. This sample loads the XmlTextReader from a stream. A stream is an abstract representation of an input or output device that is the source of or destination for data (in this case, XML data). You can write to a stream and read from a stream, which is best visualized as a flow of bytes. A stream provides independence from the device and hence requires no program changes if, for instance, the source of a stream changes.

The following sample code creates a StringReader class that builds up an XML string. Since this is purely a byte stream held in memory, you can get the XmlTextReader to parse this stream as XML. This memory stream has no particular specified encoding. The sample then creates an XmlTextReader that parses the stream and displays the resultant XML.

```
    Dim stream as StringReader
    Dim reader as XmlTextReader = nothing

    try
        Console.WriteLine ("Initializing StringReader ...")
        stream = new StringReader ("<?xml version='1.0'?>" & _
                                   "<!-- This file represents a fragment of a book store inventory database -->" & _
                                   "<bookstore>" & _
                                   " <book genre=""autobiography"" publicationdate=""1981"" ISBN=""1-861003-11-0"">" & _
                                   "   <title>The Autobiography of Benjamin Franklin</title>" & _
                                   "   <author>" & _
                                   "       <first-name>Benjamin</first-name>" & _
                                   "       <last-name>Franklin</last-name>" & _
                                   "   </author>" & _
                                   "   <price>8.99</price>" & _
                                   " </book>" & _
                                   " <book genre=""novel"" publicationdate=""1967"" ISBN=""0-201-63361-2"">" & _
                                   "   <title>The Confidence Man</title>" & _
                                   "   <author>" & _
                                   "       <first-name>Herman</first-name>" & _
                                   "       <last-name>Melville</last-name>" & _
                                   "   </author>" & _
                                   "   <price>11.99</price>" & _
                                   " </book>" & _
                                   "  <book genre=""philosophy"" publicationdate=""1991"" ISBN=""1-861001-57-6"">" & _
```

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
"   <title>The Gorgias</title>" & _
"   <author>" & _
"       <name>Plato</name>" & _
"   </author>" & _
"   <price>9.99</price>" & _
" </book>" & _
"</bookstore>")

    ' Load the XmlTextReader from the stream
    reader = new XmlTextReader (stream)

    Console.WriteLine ("Processing ...")
    Console.WriteLine ()
    FormatXml(reader)

catch e as Exception
    Console.WriteLine ("Exception: {0} ", e.ToString())

finally
    Console.WriteLine()
    Console.WriteLine("Processing of stream complete.")
    If Not reader Is Nothing
        reader.Close()
    end if
end try
```

VB

If a Stream is supplied as input, the XmlTextReader properly decodes this by wrapping the stream in a StreamReader and calling the SwitchEncoding property according to the XML encoding specified. Also, the XmlResolver is used to resolve external resources needed to correctly parse the input - like Document Type Definitions (DTDs) and schemas. Another method of representing a stream is by using a StreamReader class that implements a reader that reads from a character stream. The following code sample illustrates loading a file named books.xml and then using the XmlTextReader to parse the resulting file.

```
Dim streamreader As StreamReader = New StreamReader ("books.xml")
Console.WriteLine ("File books.xml read sucessfully ...")

' Load the XmlTextReader from the StreamReader
Dim xmlreader As XmlTextReader = New XmlTextReader (streamreader)
```

VB

The FormalXml method in this sample shows how to move to the attribute nodes using the MoveToNextAttribute method when the current node is an element node. This allows you not only to access the node's name and value properties but, because you are at a node context, you can also get properties like the current namespace for the attribute. The following code sample also shows the Format method that this time displays the name and value for the current node.

```
private shared sub FormatXml (reader as XmlTextReader)

    Dim piCount, docCount, commentCount, elementCount as Integer
    Dim attributeCount, textCount, whitespaceCount as Integer

    While reader.Read()
```

```
            Select (reader.NodeType)

            case XmlNodeType.ProcessingInstruction:
                Format (reader, "ProcessingInstruction")
                piCount += 1

            case XmlNodeType.DocumentType:
                Format (reader, "DocumentType")
                docCount += 1

            case XmlNodeType.Comment:
                Format (reader, "Comment")
                commentCount += 1

            case XmlNodeType.Element:
                Format (reader, "Element")
                elementCount += 1
                While reader.MoveToNextAttribute()
                    Format (reader, "Attribute")
                end While

                if (reader.HasAttributes)
                    attributeCount += reader.AttributeCount
                end if

            case XmlNodeType.Text:
                Format (reader, "Text")
                textCount += 1

            case XmlNodeType.Whitespace:
                whitespaceCount += 1

            end Select

        end While

        ' Display the Statistics for the file
        Console.WriteLine ()
        Console.WriteLine("Statistics for stream")
        Console.WriteLine ()
        Console.WriteLine("ProcessingInstruction: " & piCount)
        Console.WriteLine("DocumentType: " & docCount)
        Console.WriteLine("Comment: " & commentCount)
        Console.WriteLine("Element: " & elementCount)
        Console.WriteLine("Attribute: " & attributeCount)
        Console.WriteLine("Text: " & textCount)
        Console.WriteLine("Whitespace: " & whitespaceCount)
    end sub

    private shared sub Format(byref reader as XmlTextReader, nodeType as String)

        ' Format the output
        Console.Write(reader.Depth & " ")
        Console.Write(reader.AttributeCount & " ")

        Dim i as Integer
        for i = 0 to reader.Depth
            Console.Write(Strings.chr(9))
        Next
```

```
            Console.Write(reader.Prefix & nodeType & "<" & reader.Name & ">" & reader.Value)
            Console.WriteLine()

        end sub
```

VB

Summary

1. A stream is an abstract representation of an input or output device that is the source of or destination for data.
2. The XmlTextReader provides constructors to read XML from a string representing an URL or a local file name, a stream, or a TextReader.
3. The XmlTextReader provides constructors to read XML from a file name, a stream, or a TextReader.
4. Attribute nodes can be accessed with the MoveToNextAttribute method that allows you to determine the properties of the attribute node.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

How Do I...Read XML from a URL?

This sample illustrates how to read XML from an URL using the XmlTextReader class.

Note: This sample follows on from the How do I...Read XML from a file? topic.

VB XmlReadFromUrl.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The XmlTextReader has different constructors to specify the location of the XML data. This sample loads the XmlTextReader from one of these language specific URLs: http://localhost/quickstart/howto/samples/Xml/XmlReadFromUrl/vb/books.xml or http://localhost/quickstart/howto/samples/Xml/XmlReadFromUrl/cs/books.xml. The following sample code constructs an XmlTextReader.

```
private const URLString as String = "http://localhost/quickstart/howto/samples/Xml/XmlReadFromUrl/vb/books.xml"

' Load the XmlTextReader from the URL
myXmlURLreader = new XmlTextReader (URLString)
```

VB

Once loaded, the sample code calls the FormatXML function. In this function, the XmlTextReader moves across the XML data by using the Read method to perform sequential reads to get the next node. This function returns false if there are no more nodes. For more information about how the Read method works, see How do I...Read XML from a file?

```
While reader.Read()

    Select (reader.NodeType)

        case XmlNodeType.XmlDeclaration:
            Format (reader, "XmlDeclaration")
            declarationCount += 1

        case XmlNodeType.ProcessingInstruction:
            Format (reader, "ProcessingInstruction")
            piCount += 1

        case XmlNodeType.DocumentType:
            Format (reader, "DocumentType")
            docCount += 1

        case XmlNodeType.Comment:
```

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
                Format (reader, "Comment")
                commentCount += 1

            case XmlNodeType.Element:
                Format (reader, "Element")
                elementCount += 1
                if (reader.HasAttributes)
                    attributeCount += reader.AttributeCount
                end if

            case XmlNodeType.Text:
                Format (reader, "Text")
                textCount += 1

            case XmlNodeType.Whitespace:
                whitespaceCount += 1

        End Select

    End While
```

VB

Summary

1. The XmlTextReader provides constructors to read XML from a string representing an URL or a filename, a stream, or a TextReader.
2. Attribute nodes can be accessed with the MoveToNextAttribute method that allows you to determine the properties of the attribute node.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model

How Do I...Write XML to a file?

This sample illustrates how to write XML to a file using the XmlTextWriter class. The writer provides a fast, forward-only way of generating XML and helps you to build XML documents that conform to the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML specifications. The XmlTextWriter writes to a stream rather than using an object model such as the XML DOM, and so gives better performance.

Note: For more information about how to write XML using the DOM, see How Do I...Load and use the XmlDocument (W3C DOM)?

VB WriteXmlFile.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Typically, you use an XmlTextWriter if you need to write XML as raw data without the overhead of a DOM. The XmlTextWriter is an implementation of the XmlWriter class that provides the API which writes XML to file, stream, or a TextWriter. This class provides numerous validation and checking rules to ensure that the XML being written is well formed. When certain violations occur, exceptions are thrown and these exceptions should be caught. The XmlTextWriter has different constructors, each of which specifies a different type of the location to which to write the XML data. This sample uses the constructor that writes XML to a file. In particular, the following sample code constructs an XmlTextWriter with a string representing the file location for the newbooks.xml file.

```
Dim myXmlTextWriter As XmlTextWriter = new XmlTextWriter ("newbooks.xml", nothing)
```

VB

In addition to the file name, this constructor also takes the encoding that you would like to generate. If encoding is "null", the writer writes out UTF-8. For more information about how the encoding used in XML documents, see the W3C XML 1.0 specification.

This next fragment of sample code creates an XML file with a single book element. This code begins by using the Formatting property to specify the formatting of the XML data being written. By setting this property to Indented, the writer indents child elements using the Indentation and IndentChar properties.

```
myXmlTextWriter.Formatting = System.Xml.Formatting.Indented
myXmlTextWriter.WriteStartDocument(false)
myXmlTextWriter.WriteDocType("bookstore", nothing, "books.dtd", nothing)
myXmlTextWriter.WriteComment("This file represents another fragment of a book store inventory database")
myXmlTextWriter.WriteStartElement("bookstore")
    myXmlTextWriter.WriteStartElement("book", nothing)
        myXmlTextWriter.WriteAttributeString("genre","autobiography")
```

Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
            myXmlTextWriter.WriteAttributeString("publicationdate","1979")
            myXmlTextWriter.WriteAttributeString("ISBN","0-7356-0562-9")
            myXmlTextWriter.WriteElementString("title", nothing, "The Autobiography of Mark Twain")
            myXmlTextWriter.WriteStartElement("Author", nothing)
                myXmlTextWriter.WriteElementString("first-name", "Mark")
                myXmlTextWriter.WriteElementString("last-name", "Twain")
            myXmlTextWriter.WriteEndElement()
            myXmlTextWriter.WriteElementString("price", "7.99")
        myXmlTextWriter.WriteEndElement()
    myXmlTextWriter.WriteEndElement()

    'Write the XML to file and close the myXmlTextWriter
    myXmlTextWriter.Flush()
    myXmlTextWriter.Close()
```

VB

In creating this element, the preceding sample code also shows how, for each XML node type, there is corresponding XML write method. For example, writing an element calls the WriteElementString method and writing an attribute calls the WriteAttributeString method. For nested levels, you use the WriteStartElement/WriteEndElement pair and, for more complex attribute creation, you can use the WriteStartAttribute/WriteEndAttribute pair.

When writing XML, notice how the sample code writes the XML declaration with the version "1.0" using the WriteStartDocument method. If you want the writer to check that the document is well formed (XML declaration first, DOCTYPE in prolog, only one root level element, and so on), you must call this optional WriteStartDocument method before calling any other write method. Next, the code calls the WriteDocType method to writes the document type with the name "bookstore". The third parameter in the call to the WriteDocType specifies that the writer is to write SYSTEM "books.dtd". By writing this, the XML file indicates that there is an external DTD to validate against.

Finally, the sample code calls the Flush method to persist the XML data to a file before calling the Close method. (While this sample only really requires the Close method, there are occasions where the XML generated needs to be persisted and the writer reused.)

To check the output from the XmlTextWriter, perform a round trip test by reading in the generated file with an XmlTextReader to validate that the XML is well formed.

Summary

1. The XmlTextWriter provides a fast, forward only way of generating XML.
2. The XmlTextWriter helps you to write XML documents that conform to the W3C Extensible Markup Language (XML) 1.0 specification and the Namespaces in XML specification.
3. The XmlTextWriter provides constructors to write XML to a file, a stream or a TextWriter.
4. For each XML node type there are corresponding XML write methods.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema

How Do I...Apply validation when reading XML?

This sample illustrates how to apply validation when reading and parsing XML data using the XmlValidatingReader class. Validation is the process of enforcing rules on the XML content either via a Document Type Definition (DTD) or a schema. At it's simplest, a schema is an XML representation of a DTD, but it can also infer relationships within the XML data and type information. The XmlValidatingReader class layers validation support onto the XmlTextReader.

Note: This sample follows on from the How do I...Read XML from a file? topic.

VB ValidationReadingXML.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

An XmlValidatingReader can provide validation of a DTD, XML Schema Definition language (XSD) schema, or XML-Data Reduced Language (XDR) schema when parsing the document. You can use the ValidationType property to request which of these validation options to use, as shown in the following table.

Validation Type	Description
Auto	This is the default.
None	No parse time validation.
DTD	Validate according to DTD.
Schema	Validate according to XSD schemas.
XDR	Validate according to XDR schemas.

Setting the ValidationType to Auto does the following:

- If there is no DTD or schema, it will parse the XML without validation.
- If there is a DTD defined in a <!DOCTYPE ...> declaration, it will load the DTD and process the DTD declarations such that default attributes and general entities will be made available. General entities are only loaded and parsed if they are used (expanded).
- If there is no <!DOCTYPE ...> declaration but there is an XSD "schemaLocation" attribute, it will load and process those XSD schemas and it will return any default attributes defined in those schemas.
- If there is no <!DOCTYPE ...> declaration and no XSD or XDR schema information then the parser is a non-validating parser (i.e. ValidationType=ValidationType.None)
- If there is no <!DOCTYPE ...> declaration and no XSD "schemaLocation" attribute but there are some namespaces using the

Use the Xml Schema Object Model

Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

MSXML "x-schema:" URN prefix, it will load and process those schemas and it will return any default attributes defined in those schemas.

- If there is no <!DOCTYPE ...> declaration but there is a schema declaration <schema>, it will validate using the in-line schema.

The sample code (see View Source) validates four files. The first file, BooksDTD.xml, validates against the books.dtd. The second file, BooksDTDFail.xml, fails validation against the books.dtd. The third file, BooksSchema.xml, validates against the schema.xml file. The fourth file, BooksFailSchema.xml, fails validation against the schema.xml. Schema.xml is an XSD schema. The sample code sets the required type of validation by setting the ValidationType property with either the value of DTD or Schema, as shown in the following code.

```
Dim myXmlTextReader As XmlTextReader = New XmlTextReader(document1)
Dim myXmlValidatingReader As XmlValidatingReader = New XmlValidatingReader(myXmlTextReader)
myXmlValidatingReader.ValidationType = ValidationType.DTD
```

VB

The parser does not stop for any kind of validation error and only stops if the parser finds data that is not well-formed. By not stopping for validation errors, you are able to find all the validation errors in one pass without having to repeatedly parse the XML document.

As shown in the following code, a validation error results in the sample calling the handler method that was set by a ValidationEventHandler class before reading the XML document. If this method is not provided, then the XmlValidatingReader throws an XmlSchemaException upon encountering the first validation error. In this code sample, the success variable indicates the validation state.

The parser does not stop for any kind of validation error and only stops if the parser finds data that is not well-formed. By not stopping for validation errors, you are able to find all the validation errors in one pass without having to repeatedly parse the XML document.

```
private sub Validate()
    try
        ' Set the validation event handler
        Dim valdel as ValidationEventHandler = new ValidationEventHandler(addressof ValidationEvent)

        AddHandler myXmlValidatingReader.ValidationEventHandler, valdel

        ' Read XML data
        while myXmlValidatingReader.Read()
        end While

        Dim s as String
        if (success = true)
            s = "successful"
        else
            s = "failed"
        end if
        Console.WriteLine ("Validation finished. Validation {0}", s)
```

```
        catch e as XmlException
            Console.WriteLine ("XmlException: {0} ", e.ToString())

        catch e as Exception
            Console.WriteLine ("Exception: {0} ", e.ToString())
        end try
    end sub

    public sub ValidationEvent (errorid as object, args as ValidationEventArgs)
        Console.Write(Strings.chr(9) & "Validation error: " & args.Message)
    end sub
```

VB

You can use the callback handler to guarantee that the XmlSeverityType enumeration validates an XML instance document against a schema. For all validation errors, this has the value of XmlSeverityType.Error, which indicates a fatal error. However, it is possible that there is simply no schema information available with which to validate the elements and attibutes. In these cases (usually with ErrorCodes, NoElementSchemaFound and NoAttributeSchemaFound) the XmlSeverityType has the value XmlSeverityType.Warning.

Summary

1. Validation can either be against a DTD, an XSD or XDR schema, the type required being set by the Validation property.
2. Validation is performed during the read and parsing operations.
3. The ValidationEventHandler property must be set to receive notification of validation errors.
4. Validation errors do not stop parsing. Parsing only stops if there is a well-formedness error. This enables all errors to be discovered in a single pass.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

How Do I...use multiple XmlReaders

In some situations, you may have one reader that reads part of an XML document, and one or more other readers that read the rest of the document. For example, an e-commerce ordering system may have orders submitted as XML documents. In processing these orders, one reader may process information at the beginning of the file and then route the rest of the document to another reader to handle the remaining order details.

In order to enable a reader to accept a document from another reader, you use one of two XmlTextReader constructors. One constructor takes a string; the other takes a stream. The string or stream passed to the constructor can contain an XML document fragment. Thus, you can have one reader pass a partially read XML document to another reader. The following code shows the syntax for the XmlTextReader constructors that use a string or stream.

```
Overloads Public Sub New(ByVal xmlFragment as String, ByVal fragType as XmlNodeType,
                         ByVal context as XmlParserContext)
Overloads Public Sub New(ByVal xmlFragment as Stream, ByVal fragType as XmlNodeType,
                         ByVal context as XmlParserContext)
```

VB

The XmlParserContext object provides the XmlTextReader with the appropriate Document Type Definition (DTD), base URI for external resources, xml:lang, xml:space, and namespace information. The XmlParserContext constructor takes eight parameters. If only namespace/prefix lookup is required for the second XmlTextReader, you can specify the XmlNamespaceManager and XmlNameTable parameters and pass a null value for the others.

The XmlNamespaceManager passed to the constructor specifies the namespace manager used for looking up namespace information. For example, a document fragment may contain prefixes defined in an earlier portion of the document. The XmlNamespaceManager enables the XmlTextReader to resolve any prefix in the document appropriately. When creating an XmlNamespaceManager to be used by an XmlTextReader, you must use the AddNamespace method to add each prefix and namespace that the XmlTextReader reads to the XmlNamespaceManager to the collection. The default namespace (if one exists) has the prefix of String.Empty. If the XML document contains a prefix that does not have an entry in the XmlNamespaceManager, the XmlTextReader throws an XmlException.

Programmatically retrieving prefixes and namespaces from nodes in an XmlReader (and its derived classes) only takes into account nodes read up to the reader's current position in the XML document. If you want to hand off a partially read stream/string to another XmlReader (using the XmlParserContext), you cannot account for namespaces defined in the unread portion of the stream/string. A better approach is to use the XmlNamespaceManager in situations where you know the namespaces beforehand..

VB MultipleXmlReader.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

The following sample code uses two XmlTextReaders to read an XML document and display the element information. One reader reads the first portion of the document and the second reader reads the remaining part of the document. This code first creates an XmlNamespaceManager, and then adds the prefixes and namespaces (including the default namespace) used in the XML document to the XmlNamespaceManager. The XmlParserContext used to construct the second XmlTextReader then uses this XmlNamespaceManager as the namespace manager to use to look up namespace information.

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
'Create a new file stream for the specified source file.
Dim filestreamSource as FileStream  = new FileStream(args, FileMode.Open, FileAccess.Read)
'Create a new reader with the file stream
Dim reader as XmlTextReader  = new XmlTextReader(filestreamSource)
'Read the first part of the XML document
while(reader.Read())
    'Display the elements and stop reading on the part1 end element tag
    'then go to ReadPart2 to start another reader to read the rest of the file.
    Select (reader.NodeType)

        case XmlNodeType.Element
            Console.WriteLine("Name: {0}", reader.Name)
            Console.WriteLine("  Prefix: {0}", reader.Prefix)
            Console.WriteLine("  LocalName: {0}", reader.LocalName)
            Console.WriteLine("  Namespace: {0}", reader.NamespaceURI)

        case XmlNodeType.EndElement
            'Stop reading at end element for element with localname equal to part1
            if ("part1" = reader.LocalName) then
                Console.WriteLine("End reading part 1...")
                Console.WriteLine()
                goto ReadPart2
            end if

    end Select
end while

'Read the rest of the XML document
ReadPart2:
Console.WriteLine("Begin reading part 2...")
'Create an XmlNamespaceManager and add the namespaces for the document.
Dim nsmanager as XmlNamespaceManager  = new XmlNamespaceManager(reader.NameTable)
'Set default namespace--first param is null.
nsmanager.AddNamespace(String.Empty, "http://tempuri.org/mydefaultnamespace")
nsmanager.AddNamespace("myns", "http://tempuri.org/mynamespace")
nsmanager.AddNamespace("yourns", "http://tempuri.org/yournamespace")
Dim pc as XmlParserContext  = new XmlParserContext(reader.NameTable, nsmanager, reader.XmlLang, XmlSpace.Default)

Dim reader2 as XmlTextReader = new XmlTextReader(filestreamSource, XmlNodeType.Element, pc)

while(reader2.Read())
    Select (reader2.NodeType)

        case XmlNodeType.Element
            Console.WriteLine("Element Name: {0}", reader2.Name)
            Console.WriteLine("  Prefix: {0}", reader2.Prefix)
            Console.WriteLine("  LocalName: {0}", reader2.LocalName)
            Console.WriteLine("  Namespace: {0}", reader2.NamespaceURI)

        case XmlNodeType.EndElement
            'Stop reading at end element for element with localname equal to part2
            if ("part2" = reader2.LocalName) then
                Console.WriteLine("End reading part 2...")
                goto Done
            end if

    end Select
```

```
    end while

    Done:
    Console.WriteLine("Done.")
    reader.Close()
```

VB

Summary

1. The XmlTextReader classes derived from XmlReader can be constructed using a string or stream containing an XML document fragment, such as a string or stream that has been partially read by another XmlReader.
2. The XmlParserContext class provides all the required context information needed to start an XML parser up at any random point within an XML document. This enables the XmlTextReader to read XML fragments.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace

How Do I...Load and use the XmlDocument (W3C DOM)?

This sample illustrates the classes which implement:

- the Document Object Model (Core) Level 1 the World Wide Web Consortium (W3C) Document Object Model (DOM) Level 1 Specification.
- the Document Object Model Core in the W3C Document Object Model (DOM) Level 2 Specification.

The DOM is an in-memory (cache) tree representation of an XML document and enables the navigation and editing of the document.

VB DOMInterfaceXmlDocument.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

There are a number of classes implemented as part of the W3C DOM. Of these, the XmlNode class provides methods and properties to manipulate a node. Many of the other W3C classes are specializations of the XmlNode class. The XmlDocument class represents the XML document and has a Load method that loads the document from a file, stream, or an XmlReader. The list of classes is given in the table below.

Class	Derived from
XmlNode	implements ICloneable, IEnumerable
XmlDocument	XmlNode
XmlElement	XmlLinkedNode
XmlAttribute	XmlNode
XmlCharacterData	XmlLinkedNode
XmlText	XmlCharacterData
XmlComment	XmlCharacterData
XmlCDataSection	XmlCharacterData
XmlWhitespace	XmlCharacterData
XmlSignificantWhitespace	XmlCharacterData
XmlProcessingInstruction	XmlLinkedNode
XmlNodeList	implements IEnumerable
XmlNamedNodeMap	implements IEnumerable
XmlAttributeCollection	XmlNamedNodeMap
XmlDocumentFragment	XmlNode

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

XmlDocumentType	XmlLinkedNode
XmlEntityReference	XmlLinkedNode
XmlEntity	XmlNode
XmlNotation	XmlNode
XmlImplementation	Object
XmlNodeChangedEventArgs	EventArgs

The following code illustrates how to load an XmlDocument from an XmlTextReader, and then display the document's contents. The XmlDocument can also be loaded directly from a URL and then using an XmlTextReader to read the XML document.

```
Dim reader as XmlTextReader = nothing

try
    ' Load the XML from file
    Console.WriteLine ("Reading file {0} ...", args)
    reader = new XmlTextReader (args)
    reader.WhitespaceHandling = WhitespaceHandling.None
    Console.WriteLine ("File {0} read sucessfully ...", args)

    ' Create an XmlDocument from the XmlTextReader
    Dim myXmlDocument as XmlDocument = new XmlDocument()
    myXmlDocument.Load (reader)
    Console.WriteLine ("XmlDocument loaded with XML data successfully ...")

    ' Process the supplied XML file
    Console.WriteLine ("Processing ...")
    Console.WriteLine ()

    ' Start from the document Element
    DisplayTree(myXmlDocument.DocumentElement)
catch e as Exception
    Console.WriteLine ("Exception:" & e.ToString())
finally
    If Not reader Is Nothing
        reader.Close()
    end if
end try
```

VB

To navigate the document, the DisplayTree method recursively iterates over the XmlDocument using the HasChildNodes and FirstChild properties of XmlNode to move down the tree. The NextSibling property moves to the node immediately next to the current node, returning a null value if there is no node to which to move.

```
public sub DisplayTree( node as XmlNode)
    if not isnothing(node) then Format (node)

    if node.HasChildNodes then
        node = node.FirstChild
```

```
        while not isnothing(node)
            DisplayTree(node)
            node = node.NextSibling
        end While
    end if
end sub
```

VB

The Name property gives the name of the node, and the Value property its value depending on the type. The following table shows the value returned for a given node type. The Format method, shown in the following code, displays the details of the current node using the Name and Value properties.

Type	Value
Attribute	Value of attribute
CDATA	Content of the CDATA Section
Comment	Content of the comment
Document	A null reference (Nothing in Visual Basic)
DocumentFragment	A null reference (Nothing in Visual Basic)
DocumentType	A null reference (Nothing in Visual Basic)
Element	A null reference (Nothing in Visual Basic)
Entity	A null reference (Nothing in Visual Basic)
EntityReference	A null reference (Nothing in Visual Basic)
Notation	A null reference (Nothing in Visual Basic)
ProcessingInstruction	Entire content excluding the target
Text	Content of the text node
XmlDeclaration	A null reference (Nothing in Visual Basic)

Note: The XmlNode class also has a NodeType property. For a table that describes the different node types, see How Do I...Read XML from a File?.

For XmlElement node types, the Attributes property provides a list of attributes as an XmlNamedNodeMap class. The XmlNameNodeMap class also implements IEnumerable in order to support the foreach statement in C# and the For EachNext statement in Visual Basic.

```
' Format the output
private sub Format ( node as XmlNode)
    if not node.HasChildNodes then
        Console.WriteLine( Strings.chr(9) & node.Name & "<" & node.Value & ">")

    else
        Console.Write(node.Name)
        if XmlNodeType.Element = node.NodeType then
            Dim map as XmlNamedNodeMap = node.Attributes
            Dim attrnode As Object
```

```
                    for each attrnode in map
                        Console.Write(" " & Ctype(attrnode,XmlNode).Name & "<" & Ctype(attrnode,XmlNode).Value & "> " )
                    next
                end if
                Console.WriteLine()
            end if
        end sub
```

VB

The XmlNode class also has a **NodeType** property. See How Do I...Read XML from a file? for a table of the different node types. The output from running this sample is shown below.

```
Reading file books.xml ...
File books.xml read sucessfully ...
XmlDocument loaded with XML data successfully ...
Processing ...

bookstore
book genre<autobiography>  publicationdate<1981>  ISBN<1-861003-11-0>
title
        #text<The Autobiography of Benjamin Franklin>
author
first-name
        #text<Benjamin>
last-name
        #text<Franklin>
price
        #text<8.99>
book genre<novel>  publicationdate<1967>  ISBN<0-201-63361-2>
title
        #text<The Confidence Man>
author
first-name
        #text<Herman>
last-name
        #text<Melville>
price
        #text<11.99>
book genre<philosophy>  publicationdate<1991>  ISBN<1-861001-57-6>
title
        #text<The Gorgias>
author
name
        #text<Plato>
price
        #text<9.99>
```

Summary

1. The XmlDocument, XmlNode and other classes implement the W3C Document Object Model Level 1 Core and the Core DOM Level 2 specifications.
2. The XmlDocument is an in-memory (cache) tree representation of an XML document.
3. There are different node types specializing from XmlNode to enable you to manipulate the XML document.
4. You can get faster, non-cached, forward-only stream access to XML using the XmlTextReader and XmlTextWriter classes.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema

How Do I...Save XML with the XmlDocument (W3C DOM)?

This sample illustrates how to update and save XML using the XmlDocument class, which is an implementation of the World Wide Web Consortium (W3C) Document Object Model (DOM).

This sample follows on from How do I...Load and use the XmlDocument?.

VB SaveXmlDocument.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The XmlDocument class has a Save method that provides XmlDocument the ability to save XML data to files, streams, and XmlWriters.

This sample application, as shown in the following code, navigates over the file, books.xml, in order to increase all the books prices by 2%. The application then saves the XML to a new file called updatebooks.xml.

```
public sub Run(args As String())
    try
        ' Create an XmlDocument
        Dim myXmlDocument as XmlDocument = new XmlDocument()
        myXmlDocument.Load (args(0))
        Console.WriteLine ("XmlDocument loaded with XML data successfully ...")
        Console.WriteLine()

        IncreasePrice(myXmlDocument.DocumentElement)

        ' Write out data as XML
        myXmlDocument.Save(args(1))
        Console.WriteLine()
        Console.WriteLine("Updated prices saved in file {0}", args(1))

    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())
    end try
end sub
```

VB

Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

The IncreasePrice method recursively iterates over the XML document using the FirstChild method of the XmlNode class to move down the tree. This method returns a null value if there are no child nodes. The NextSibling property moves to the node immediately next to the current node returning a null value if there is no node to which to move. Every time the application finds a node named price, the application updates the price. The following code shows how the IncreasePrice method works.

```
public sub IncreasePrice(node as XmlNode)
    if (node.Name = "price")

        node = node.FirstChild
        Dim price as Decimal
        price = System.Decimal.Parse(node.Value)

        ' Increase all the book prices by 2%
        Dim newprice as String
        newprice = CType(price*(new Decimal(1.02)), Decimal).ToString("#.00")
        Console.WriteLine("Old Price = " & node.Value & Strings.chr(9) & "New price = " & newprice)
        node.Value = newprice
    end if

    node = node.FirstChild
    While Not node Is Nothing
        IncreasePrice(node)
        node = node.NextSibling
    end while
end sub
```

VB

```
XmlDocument loaded with XML data successfully ...

Old Price = 8.99        New price = 9.17
Old Price = 11.99       New price = 12.23
Old Price = 9.99        New price = 10.19

Updated prices saved in file updatebooks.xml
```

Summary

1. The XmlDocument class can save XML to a file, stream, or an XmlWriter.
2. The XmlNode class enables you to navigate and amend the nodes within an XML document.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema

How Do I...Handle Events from XmlDocument?

This sample illustrates how to receive and handle events when nodes in an XML document change. In particular, this sample demonstrates how to catch the NodeChanged and NodeInserted events described in the following paragraphs.

Modifying an XmlDocument can result in one or more of the following events:

- The NodeInserted event occurs when any node belonging to XmlDocument has been inserted into another node. This includes all nodes that were created by this document whether they are inside or outside the document tree, including attribute nodes.
- The NodeRemoved event occurs when any node belonging to this document has been removed from its parent. This includes all nodes that were created by this document whether they are inside or outside the document tree, including attribute nodes.
- The NodeChanged event occurs when the Value property of any node belonging to this document has been changed. This only applies to nodes that have a Value property.
- The NodeInserting, NodeRemoving, or NodeChanging events occur when any node belonging to this document is about to be inserted, removed, or changed.

These events enable you to throw an exception if you want to stop the operation. The XmlDocument guarantees that, after throwing the exception, the document is in the state it was in before the operation was started.

VB XmlDocumentEvent.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

After loading the books.xml file into an XmlDocument, this sample application both modifies the price of the existing books within the document and adds new books to the document. These changes result in NodeChanged and NodeInserted events.

The following sample code constructs an XmlDocument and loads books.xml.

```
Dim myXmlDocument as XmlDocument  = new XmlDocument()
myXmlDocument.Load (args)
```

VB

Once the sample loads books.xml, the sample adds the NodeChanged and a NodeInserted event handlers the XmlDocument. (The code for these handlers appears later in this topic.)

Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
AddHandler myXmlDocument.NodeChanged, new XmlNodeChangedEventHandler(addressof MyNodeChangedEvent)
AddHandler myXmlDocument.NodeInserted, new XmlNodeChangedEventHandler(addressof MyNodeInsertedEvent)
```

VB

The sample application changes data with the document by increasing the price of each book by 2%. To do this, the sample must first select the data to be changed using the SelectNodes method of the XmlNode class. By using the XML Path Language (Xpath) expression, "descendant::book/price", with the SelectNodes method, the sample selects the price child elements of all the book elements. The application then places these selected nodes into an XmlNodeList where they can be edited to reflect a 2% price increase. This price increase operation consists of getting the InnerText value of the price element, increasing it by 2%, and then inserting the modified value back into the element. Each time the price element changes, there is a NodeChanged event that writes the new price to the screen. (The code for the NodeChanged event appears later in this topic.)

```
' Increase all the book prices by 2%
...
' Create a list of the  nodes and change their values
Dim myXmlNodeList as XmlNodeList  = myXmlDocument.SelectNodes("descendant::book/price")
Dim myXmlNode as XmlNode

for each myXmlNode in myXmlNodeList
    Console.WriteLine("<" + myXmlNode.Name + "> " + myXmlNode.InnerText)
    Dim price as Double  = System.Double.Parse(myXmlNode.InnerText)
    myXmlNode.InnerText = CStr(CType(price * 1.02, Double).ToString("#.00"))
next
```

VB

The sample application also inserts a node into the XmlDocument. A simple way to insert a new node is to create an XmlDocumentFragment, determine where in the document you want to insert the fragment, and then use the InsertBefore or InsertAfter methods of the XmlDocument. As shown in the following code, the sample creates the XmlDocumentFragment with a string of prepared XML and then inserts this fragment into myXmlDocument. By inserting a new node, there is a NodeInserted event that writes the new node information to the screen. (The code for the NodeInserted event appears later in this topic.)

```
' Create a new book
Dim myNewBook as XmlDocumentFragment  = myXmlDocument.CreateDocumentFragment()

myNewBook.InnerXml = ("" & _
                      "" & _
                      "" & _
                      "Eric" & _
                      "Gamma" & _
                      "" & _
                      "49.95" & _
                      "")


Dim rootXmlElement as XmlElement  = myXmlDocument.DocumentElement
```

```
' Add the new book to the XmlDocument
rootXmlElement.InsertBefore(myNewBook, rootXmlElement.FirstChild)
...
' Clone the node and note that by cloning the node we
' are inserting it again into the XmlDocument
Dim myNewBook2 as XmlNode  = myXmlDocument.DocumentElement.FirstChild.Clone()
```

VB

The following code describes the functions that handle the NodeChanged and NodeInserted events.

```
' Handle the Node Changed Event
public sub MyNodeChangedEvent(src as Object , args as XmlNodeChangedEventArgs )
    Console.Write("Node Changed Event: <" + args.Node.Name + "> changed")
    if not (args.Node.Value is Nothing) then
        Console.WriteLine(" with value " + args.Node.Value)
    else
        Console.WriteLine("")
    end if
end sub

' Handle the Node Inserted Event
public sub MyNodeInsertedEvent(src as Object , args as XmlNodeChangedEventArgs )
    Console.Write("Node Inserted Event: <" + args.Node.Name + "> inserted")
    if not (args.Node.Value is Nothing) then
        Console.WriteLine(" with value " + args.Node.Value)
    else
        Console.WriteLine("")
    end if
end sub
```

VB

Summary

1. To edit node values, first use SelectNodes to select the nodes and then place the selected nodes into an XmlNodeList.
2. There are six node events that can be handled: NodeInserted, NodeRemoved, NodeChanged, NodeInserting, NodeRemoving, or NodeChanging.
3. To handle nodes, add the node handler to your instance of XmlDocument.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

How Do I...Use an XmlNodeReader?

This sample illustrates how to create and use an XmlNodeReader. An XmlNodeReader is a reader that provides fast, non-cached, forward-only access to XML data in an XmlNode. It has the ability to read an entire XML DOM tree or read from just a subtree. However, the XmlNodeReader does not support Document Type Definition (DTD) or schema validation, and therefore does not validate the XML it is reading.

This sample loads the books.xml into an XmlDocument, and then uses an XmlNodeReader to get each node and its respective value to display to the screen.

VB XmlNodeReader.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

After creating and loading an XmlDocument with the books.xml file, the sample creates an XmlNodeReader that selects a single node from the XmlDocument. The sample passes this XmlNodeReader to the FormatXml function that prints the node data to the screen. The sample then creates another XmlNodeReader that selects a different node to output to the screen.

```
Dim myXmlDocument as XmlDocument  = new XmlDocument()
myXmlDocument.Load (arg)
...
Dim myXmlNodeReader as XmlNodeReader  = new XmlNodeReader (myXmlDocument.SelectSingleNode("bookstore/book[3]"))
FormatXml (myXmlNodeReader)
...
myXmlNodeReader = new XmlNodeReader (myXmlDocument.SelectSingleNode("bookstore/book[2]"))

myXmlNodeReader.Read()
Console.WriteLine (myXmlNodeReader.ReadInnerXml())
```

VB

This part of the sample simply reads each node using the Read method of the XmlNodeReader and then, depending on the node type, prints the node to the screen using the Format function.

```
private sub FormatXml (reader as XmlNodeReader)

    while (reader.Read())

    select case reader.NodeType

        case XmlNodeType.ProcessingInstruction:
            Format (reader, "ProcessingInstruction")
```

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
            case XmlNodeType.DocumentType:
                Format (reader, "DocumentType")

            case XmlNodeType.Document:
                Format (reader, "Document")

            case XmlNodeType.Comment:
                Format (reader, "Comment")

            case XmlNodeType.Element:
                Format (reader, "Element")
                while (reader.MoveToNextAttribute())
                    Format (reader, "Attribute")
                end while

            case XmlNodeType.Text:
                Format (reader, "Text")

            case XmlNodeType.Whitespace:
                Format (reader, "Whitespace")

            end select
    end while

    end sub

    'Format the output
    private sub Format (reader as XmlNodeReader, NodeType as String)

        Dim value as String = System.String.Empty
        Dim i as long
        for i=0  to (reader.Depth - 1)
            Console.Write(Chr(9))
        Next

        if (reader.HasValue)
            value = reader.Value
        end if

        Console.WriteLine(NodeType & "<" & reader.Name & ">" + value)
    end sub
```

VB

Summary

1. XmlNodeReader is a fast, non-cached, forward-only reader to access XML data in an XmlNode.
2. Since XmlNodeReader can also be constructed with any XmlNode within the XmlDocument, XmlNodeReader provides a reader that reads only the subtree of a given node.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Load XML data into an XmlDataDocument?

This sample illustrates how to load XML into the XmlDataDocument class. This class extends the XmlDocument class, allowing structured data to be stored, retrieved, and manipulated through the DataSet class, which provides a relational view onto the loaded XML data. This affiliation between the XmlDocument class and DataSet class allows for components to mix XML and relational views of the same underlying data-irrespective of the source of the data. As detailed in other topics, there are differences between these views. Most notable is that XML data that does not map to the relational view cannot be accessed via relational objects such as tables and rows. It is, however, still possible to easily move from a relational row to the equivalent XML element and hence access this XML data.

You can use an XmlDataDocument anywhere an XmlDocument is used; all topics that describe the XmlDocument apply equally to the XmlDataDocument. Hence, the information covered in the following topics relate to XmlDataDocument as well:

- How Do I...Load and use the XmlDocument?
- How Do I...Save XML with the XmlDocument?

There is also a companion topic, How Do I...Read XML Data from an XmlDataDocument?, that focuses on just using XmlDataDocument to read XML.

In addition, the topic, How Do I...Use XML and the DataSet Class?, gives additional information on how to use the XmlDataDocument and the DataSet.

VB LoadXmlDocument.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

As shown in the following code, this sample simply loads an XmlDataDocument with the XML data in the sample file, books.xml, and displays that data on screen.

```
public class LoadXmlDocumentSample
    private const document as String = "books.xml"

    public shared sub Main()
        Dim myLoadXmlDocumentSample as LoadXmlDocumentSample
        myLoadXmlDocumentSample = new LoadXmlDocumentSample()
        myLoadXmlDocumentSample.Run(document)
    end sub

    public sub Run(args as String)
        try
            'Load the XML from file
            Console.WriteLine ()
            Console.WriteLine ("Loading file {0} ...", args)

            Dim myXmlDocument as XmlDataDocument = new XmlDataDocument()
            myXmlDocument.Load (args)

            Console.WriteLine ("XmlDataDocument loaded with XML data successfully ...")

        catch e as Exception
            Console.WriteLine ("Exception: {0}", e.ToString())
        end try
    end sub
end Class
```

VB

Summary

1. The XmlDataDocument class provides an in memory cache for XML data.
2. The XmlDataDocument extends the XmlDocument class. XmlDataDocument can be used anywhere the XmlDocument class is used.
3. There is a close association between the XmlDataDocument and the DataSet classes. In fact, you can regard the XmlDataDocument as a DataSet-aware XmlDocument.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Read XML from an XmlDataDocument?

This sample illustrates how to create an XmlReader class that reads the XML data stored in the XmlDataDocument. Once the data has been read, the sample applies an XslTransform to the data and then reads the transformed data out from the XmlDataDocument.

Note: This sample continues from How do I...Load XML data into an XmlDataDocument?.

The XmlDataDocument class functions as an in-memory cache for XML data. This class enables you to load either relational or XML data, and then manipulate that data using Extensible Stylesheet Language Transfomations (XSLT).

The XmlDataDocument class also has a close association with the DataSet class. In fact, you can construct an XmlDataDocument class so that it maps to a DataSet. For more information about mapping XmlDataDocument to a DataSet, see How do I...Create DataSet Mappings from an XSD Schema?

VB StreamXmlDocument.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The code for this sample starts by loading an XmlDataDocument with the XML data in the sample file, books.xml. Then, the sample creates an XslTransform object and loads it with the XSLT style sheet, identity.xsl. This style sheet replicates all the XML from the root node.

Before the data can be transformed, the data must be in a form that supports XML Path Language (XPath) queries. Thus, the sample uses the XmlDataDocument to create an XPathNavigator on which the node transformations can be applied. Now, by applying the XslTransform class to the XPathNavigator, the sample is able to transform the data. This transformed data is returned as an XmlReader that the Format function can use to read the XML data in the XmlDataDocument.

Note: For information on the XslTransform class, see How Do I...Apply XSLT to XML Data?

```
private const document as String = "books.xml"
private const newStyleSheet as String = "identity.xsl"

shared sub Main()
    Dim args As String() = {document, newStyleSheet}
    Dim myStreamXmlDocumentSample as StreamXmlDocumentSample
    myStreamXmlDocumentSample = new StreamXmlDocumentSample()
    myStreamXmlDocumentSample.Run(args)
end sub

public sub Run(args As String())
    Dim myXmlReader as XmlReader = nothing
    Dim myXmlDataDocument as XmlDataDocument = new XmlDataDocument()
    Dim myXslTransform as XslTransform = new XslTransform()

    try
        ' Load the XML from file
        Console.WriteLine ("Loading file {0} ...", args(0))
        myXmlDataDocument.Load (args(0))
        Console.WriteLine ("XmlDataDocument loaded with XML data successfully ...")

        Console.WriteLine ("Reading XML ...")
        Console.WriteLine()
        myXslTransform.Load(args(1))

        Dim myXPathNavigator as XPathNavigator = myXmlDataDocument.CreateNavigator()
        myXmlReader = myXslTransform.Transform(myXPathNavigator, nothing)
        FormatXml (myXmlReader)

    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())

    finally
        ' Close the reader
        If Not myXmlReader Is Nothing
            myXmlReader.Close()
        end if

    end try
end sub
```

VB

The FormatXml function is used to demonstrate one technique for reading XML from an XmlDataDocument. We use an **XmlReader** to walk over the entire content of the and write out our custom formated XML content.

```
private sub FormatXml (reader as XmlReader)
    While reader.Read()
        Select (reader.NodeType)
            case XmlNodeType.ProcessingInstruction:
            ...
        end Select
    end While
end sub

private sub Format(byref reader as XmlReader, NodeType as String)
    ' Format the output
    ...
end sub
```

VB

Summary

1. The XmlDataDocument class enables you to load either relational data or XML data.
2. You can create an XmlReader for the XML data in an XmlDataDocument by using an XslTransform.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model

How Do I...Navigate XML with the XPathNavigator?

This sample illustrates how to navigate XML documents with an XPathNavigator created from an XPathDocument. It also demonstrates how to use an XPathNodeIterator. The XPathNavigator class provides a cursor-style model to navigate an XML document. This topic also introduces XML Path Language (XPath) expressions with more detail added in How Do I...Query XML with an XPath expression?.

VB NavigateXmlDocument.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

This sample loads an XPathDocument with the XML data found in books.xml, creates an XPathNavigator as a view onto the data, and then displays the XML by moving through the document. The XPathNavigator class provides an XPath API over data such as the XPathDocument. The following sample code shows the creation of these classes.

```
Dim myXPathDocument as XPathDocument = new XPathDocument(args)
...
' Create an XPathNavigator
Dim myXPathNavigator as XPathNavigator  = myXPathDocument.CreateNavigator()
```

VB

The XPathNavigator enables you to move through both the attributes nodes and the namespace nodes in an XML document.

The methods and properties of the XPathNavigator operate in relation to the current node:

- The MoveToFirstChild method moves to the first child of the current node.
- The MoveToParent method moves to the parent node of the current node. When positioned on an attribute, you must use the MoveToParent method to move to the element that owns the attribute node. This is how you get back to the element after navigating through its attributes using MoveToFirstAttribute and MoveToNextAttribute.
- The HasChildren property indicates whether the current node has child nodes.
- The HasAttributes property indicates whether the current node has attribute nodes.
- The MoveToRoot method sets the navigator to the document node that contains the whole tree of nodes.

These methods enable you to navigate the XML document in a recursive fashion as illustrated in the following code.

```
myXPathNavigator.MoveToRoot() ' Initialise the myXPathNavigator to start at the root
DisplayTree(myXPathNavigator) ' Display all the nodes
```

Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
...
' Walks the XPathNavigator tree recursively
public sub DisplayTree (myXPathNavigator as XPathNavigator )

    if (myXPathNavigator.HasChildren)
        myXPathNavigator.MoveToFirstChild()

        Format (myXPathNavigator)
        DisplayTree (myXPathNavigator)

        myXPathNavigator.MoveToParent()
    end if

    while (myXPathNavigator.MoveToNext())
        Format (myXPathNavigator)
        DisplayTree (myXPathNavigator)
    end while
end sub

' Format the output
private sub Format (myXPathNavigator as XPathNavigator )
    if Not (myXPathNavigator.HasChildren)
        if (myXPathNavigator.NodeType = XPathNodeType.Text)
            Console.WriteLine(myXPathNavigator.Value )
        elseif (myXPathNavigator.Name <> String.Empty)
            Console.WriteLine("<" & myXPathNavigator.Name & ">")
        else
            Console.WriteLine()
        end if

    else
        Console.WriteLine("<" & myXPathNavigator.Name & ">")

        ' Show the attributes if there are any
        if (myXPathNavigator.HasAttributes)
            Console.WriteLine("Attributes of <" & myXPathNavigator.Name & ">")

            while (myXPathNavigator.MoveToNextAttribute())
                Console.Write("<" + myXPathNavigator.Name + "> " + myXPathNavigator.Value + " ")
            end while

            ' Return to the 'Parent' node of the attributes
            myXPathNavigator.MoveToParent()
        end if
    end if
end sub
```

VB

If you know the structure of the XML document, then you can navigate by explicitly moving through the document from node to node. The following code finds the price of the first book node in the sample file, books.xml.

```
'Find the price of the first book. Start at the root node
Console.WriteLine ()
Console.WriteLine ("Find the price of the first book by navigating nodes ...")
myXPathNavigator.MoveToRoot()
DisplayNode (true, myXPathNavigator) ' root node
DisplayNode (myXPathNavigator.MoveToFirstChild(), myXPathNavigator) ' ?xml version='1.0'? node
DisplayNode (myXPathNavigator.MoveToNext(), myXPathNavigator) '!-- This file ... node
DisplayNode (myXPathNavigator.MoveToNext(), myXPathNavigator) ' bookstore element
DisplayNode (myXPathNavigator.MoveToFirstChild(), myXPathNavigator) ' book element
DisplayNode (myXPathNavigator.MoveToFirstChild(), myXPathNavigator) ' title element
DisplayNode (myXPathNavigator.MoveToNext(), myXPathNavigator)' author element
DisplayNode (myXPathNavigator.MoveToNext(), myXPathNavigator) ' price Element
DisplayNode (myXPathNavigator.MoveToFirstChild(), myXPathNavigator) ' value of price element
...
private sub DisplayNode(success as Boolean, myXPathNavigator as XPathNavigator )

    if (success and (myXPathNavigator.NodeType = XPathNodeType.Text))
        Console.WriteLine(myXPathNavigator.Value)
    else if (success and (myXPathNavigator.Name <> String.Empty))
        Console.WriteLine("<" & myXPathNavigator.Name & ">")
    else
        Console.WriteLine()
    end if
end sub
```

VB

This following code illustrates one of the most powerful features of the XPathNavigator-its ability to create a set of selected nodes from an XPath expression using the Select method. XPath is a language that provides filtering and addressing of data in an XML document and is another approach to retrieving a set of nodes. An XPathNavigator can apply an XPath expression and generate a set of nodes that correspond to the selected expresion. You can then navigate this node-set independently since these nodes are returned to an XPathNodeIterator whose MoveNext method allows read-only navigation over the selected node tree. The following code example selects and displays the title child node of each book node in the sample file, books.xml.

```
Dim myXPathNodeIterator as XPathNodeIterator  = myXPathNavigator.Select("descendant::book/title")

while(myXPathNodeIterator.MoveNext())
    Console.WriteLine("<" & myXPathNodeIterator.Current.Name & ">" & myXPathNodeIterator.Current.Value)
end while
```

VB

Summary

1. The XPathNavigator class provides a cursor-style model to navigate an XML document in memory.
2. The XPathNavigator class provides an XPath API over data, such as the XPathDocument.
3. The XPathNavigator enables you to move through both the attributes nodes and namespace nodes in an XML document.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace

How Do I...Query XML with an XPath expression?

This sample illustrates how to query an XPathDocument with an W3C XML Path Language (XPath) expression supplied to the XPathNavigator class. The XPathNavigator class is only for read-only XPath queries over a document and, for Extensible Stylesheet Language Transformations (XSLT) processing, is implemented by the XslTransform class. For XSLT, an XPathNavigator does not need to be created.

Note: XPath is the W3C's general query language specification for addressing parts of an XML document. The .NET Frameworks implementation of XPath conforms to the World Wide Web Consortium (W3C) XML Path Language (XPath) Version 1.0 specification.

For fast, high-performance XML document processing using XSLT, use the XPathDocument class. You can think of the XPathDocument class as similar to the XML DOM, but highly optimized for XSLT processing and the XPath data model. However, unlike the W3C XML DOM classes, the XPathDocument class does not maintain node identity, nor does it do all the rule and validation checking required by the DOM classes.

You can also perform XPath queries against an XmlDocument or XmlDataDocument class. If you use one of these two classes, then you need to use the SelectNodes and SelectSingleNode methods of the XmlNode class.

VB QueryXmlDocumentXPath.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

This sample loads an XPathDocument with the sample file, books.xml. The sample then passes XPath expressions to the XPathQuery function. The first XPath expression to be passed selects all the price nodes for all the book element nodes starting from the root node. The second XPath expression gets the text for the ISBN attribute node associated with the last book element node starting from the root node.

To select nodes that match the given XPath expression, the XPathQuery function uses the Select method of XPathNavigator. The function then creates an XPathNodeIterator, and uses this iterator to repeatedly locate a node and print that node to the screen.

Note: As the XPathNodeIterator represents an XPath node-set, the XPathNodeIterator supports operations on this node-set.

```
private const localURL as String = "http://localhost/quickstart/howto/samples/Xml/QueryXmlDocumentXPath/cs/books.xml"

Shared sub Main
    Dim myQueryXmlDocumentXPathSample as QueryXmlDocumentXPathSample
    myQueryXmlDocumentXPathSample = new QueryXmlDocumentXPathSample()
    myQueryXmlDocumentXPathSample.Run(localURL)
end sub

public sub Run(args as String)
    Console.WriteLine("XPath Test started ...")

    Dim myXPathDocument as XPathDocument = new XPathDocument(args)
    Dim myXPathNavigator as XPathNavigator = myXPathDocument.CreateNavigator()

    ' Get all the book prices
```

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
    XPathQuery(myXPathNavigator, "descendant::book/price")

    ' Get the ISBN of the last book
    XPathQuery(myXPathNavigator, "bookstore/book[3]/@ISBN")
end sub

private sub XPathQuery(myXPathNavigator as XPathNavigator, xpathexpr as String )

    try
        Console.WriteLine("XPath query: " + xpathexpr)

        ' Create a node interator to select nodes and move through them (read-only)
        Dim myXPathNodeIterator as XPathNodeIterator  =  myXPathNavigator.Select (xpathexpr)

        while (myXPathNodeIterator.MoveNext())
            Console.WriteLine("<" & myXPathNodeIterator.Current.Name + "> " & myXPathNodeIterator.Current.Value)
        end while

        Console.WriteLine()
    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())
    end try
end sub
```

VB

The first query selects all the 'price' nodes for all the books starting from the root node. The second query gets the 'ISBN' attribute text for the last book element node starting from the root node.

The query can also be performed using an XPathExpression class . There are two main reasons for using the XPathExpression class over using the string expression in the S elect method (on the XPathNavigator) :
1) The expression is compiled only once and can be reused several times thus improving performance
2) Prefixes (s) used in the XPath expression can be bound to a namespace using a XmlNamespaceManager class - this allows for prefixed XPath expressions when selecting a set of nodes.
The following code conveys the general usage which selects all the book whose ISBN attribute are in the urn:samples namespace.

```
' compile the expression
Dim expr as XPathExpression = nav.Compile("book/@mybk:ISBN")
' set namespace(s) used in the expression
Dim mngr as XmlNamespaceManager = new XmlNamespaceManager(new NameTable())
mngr.AddNamespace("mybk","urn:samples")
expr.SetContext(mngr)
' select the nodes
Dim myXPathNodeIterator as XPathNodeIterator = nav.Select(expr)
```

VB

If the XPathExpression does not include a prefix, it is assumed that the namespace URI is the empty namespace. If your XML includes a default namespace, you must use still use the SetContext method and provide an XmlNamespaceManager containing a prefix and namespace URI to handle the default namespace. For more information on using the XmlNamespaceManager, please refer to the "XmlNameSpace Manager & NameTable" topic.

Summary

1. To query an XPathDocument, XmlDataDocument, or XmlDocument, you can supply an XPath expression to an XPathNavigator class.
2. The XPathNavigator class is only for read-only XPath queries over an XML document.
3. The XPathDocument class provides a fast, high-performance oriented cache for XML document processing using XSLT.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model

How Do I...Apply an XSL Transformation to XML?

This sample illustrates how to apply an Extensible Stylesheet Language Transformations (XSLT) file to an XML document using the XslTransform class. This class implements the World Wide Web Consortium (W3C) W3C XSL Transformations (XSLT) version 1.0 specification. (For further reference, you might want to refer to W3C W3C Extensible Stylesheet Language (XSL) Version 1.0 specification.)

VB TransformXML.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

XSLT is an XML-based language designed to transform one XML document into another XML document or an XML document into any other structured document. XSLT uses the XML Path Language (XPath) to perform queries on an XML document in order to specify a particular part of the document. (For an example of how to use XPath to query an XML document, see How Do I...Query XML with an XPath expression?).

Often, the structure of an XML document does not match the structure needed to process the XML data. To transform the existing structure into one that can be processed, you use an XSLT processor and an XSLT style sheet (or XSLT file) that defines how to do the transformation. The XSLT file consists of templates that specify how each node of the source XML document should appear in the destination XML document. Thus, the XSLT processor holds three trees:

1. one tree for the source XML
2. one tree for the destination structure
3. one tree for the XSLT file

The following diagram illustrates the transformation process: taking in an XML document, applying the XSLT file to the nodes of the XML document, and generating another document of the desired format.

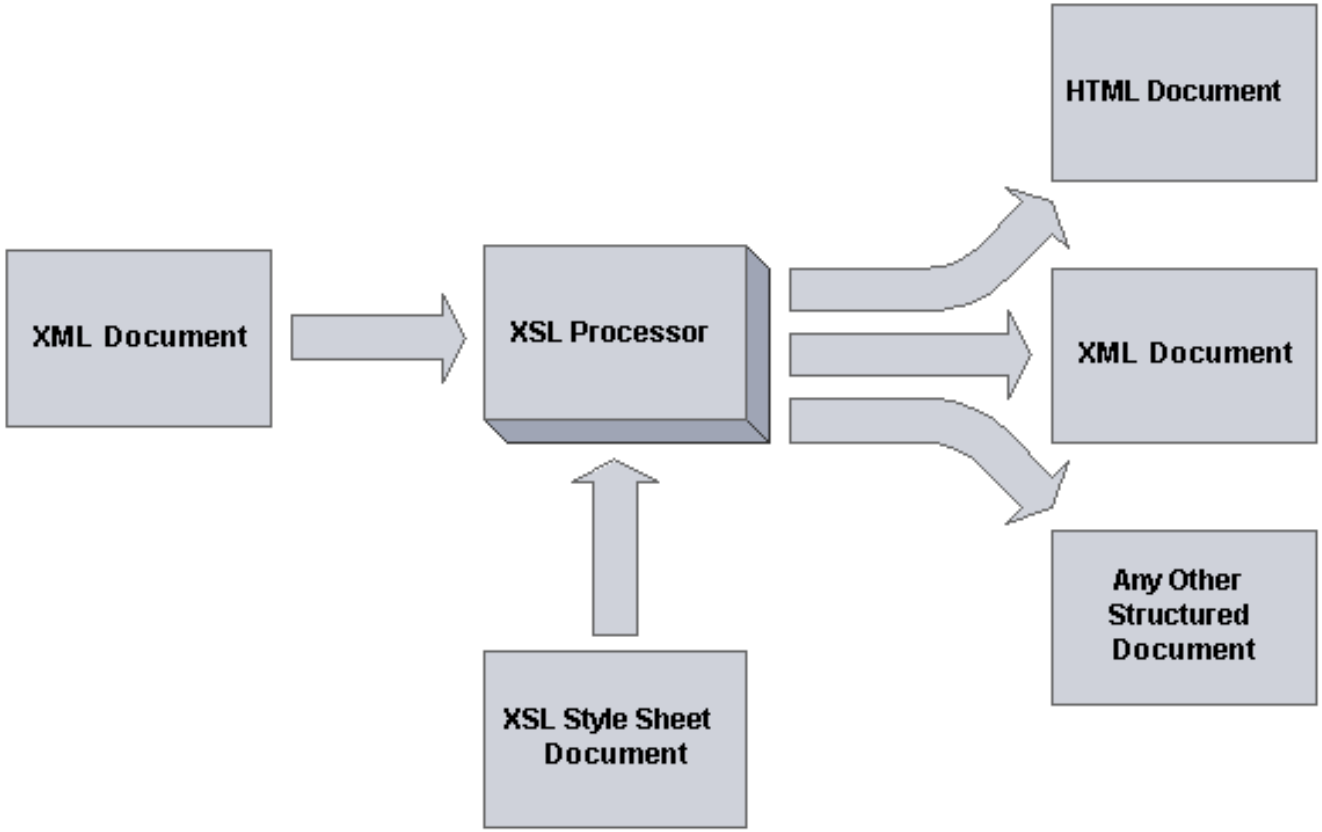
Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page



This sample consists of two functions. The first function simply reads an XML document and then transforms the data using an XSLT file. The second function reads an XML document, transforms the data, and then writes a new XML document from the transformed data.

The sample applies these two functions to three sets of documents:

1. The documents in the first set are the XML document, books.xml, and the XSLT file, books.xsl. The XSLT file consists of templates that get all the books in the bookstore element and then, for each book element, get the values for the ISBN attribute and the price element. There is also a template that tells the XSLT processor to start processing from the root ("/") node.
2. The documents in the second set are the XML document, ProcessParametersA.xml, and the XSLT file, StyleSheetGenerator.xsl. In this case, the sample reads the XML document to find out which of two style sheets contained in the XSLT file to use. The sample then generates the specified style sheet.
3. The documents in the third set are the XML document, ProcessParametersB.xml, and the XSLT file, StyleSheetGenerator.xsl. The sample processes these documents in the same way that it processes the second set of documents. However, in this case, the sample generates a different style sheet.

The following code shows the first of the two functions. This is the function that simply reads and transforms data. To do this, the function first creates an XPathDocument object and an XslTransform object. It then uses the Load method to load the XslTransform object with the XSLT file. To initiate the transformation, the function calls the Transform method passing in the XPathDocument.

The transformation process returns an XmlReader that is used to read the transformed data.

Note: As discussed in How Do I...Query XML with an XPath Expression?, the XPathDocument class provides a fast, high-performance cache for XML document processing using XSLT.

```
' args[0] is the xml file and args[1] is the stylesheet file.
public sub ReadTransform(args As String())
    try
        Dim myXPathDocument as XPathDocument = new XPathDocument (args(0))
        Dim myXslTransform as XslTransform = new XslTransform()
        myXslTransform.Load(args(1))
        Dim reader as XmlReader = myXslTransform.Transform(myXPathDocument, nothing)

        FormatXml(reader)

    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())
    end try
end sub
```

VB

Besides returning an XmlReader from a transformation, it is possible supply an XmlWriter to the Transform method. Supplying a writer enables the transformed data to be written to a file or stream. The second of the two functions in this sample shows how to incorporate an XmlWriter into the transformation process. The following code shows how this second function writes the transformed data to a file called transform.xml and then prints this file to the console.

```
' args[0] is the xml file and args[1] is the stylesheet file.
public sub ReadTransformWrite(args As String())
    Dim stream as StreamReader = nothing

    try
        Dim myXPathDocument as XPathDocument = new XPathDocument (args(0))
        Dim myXslTransform as XslTransform = new XslTransform()
        Dim writer as XmlTextWriter = new XmlTextWriter("transform.xml", nothing)
        myXslTransform.Load(args(1))
        myXslTransform.Transform(myXPathDocument, nothing, writer)

        writer.Close()

        stream = new StreamReader ("transform.xml")
        Console.Write(stream.ReadToend())
    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())
    finally
        If Not stream Is Nothing
            stream.Close()
        end if
    end try
```

```
end sub
```

VB

Although not shown in this sample, you could use the Transform method of XslTransform to load a document and write to file directly. This is a convenience method to load an XML document from file, apply the XSLT file, and write the output to a file. This prevents the user from having to create and load the input source document, and then write to a file stream. This method uses the XPathDocument to transform the XML, as shown in the following code.

```
Dim xsltransform as XslTransform = new XslTransform()
xsltransform.Load("favorite.xsl")
xsltransform.Transform("MyDocument.xml", "TransformResult.xml")
```

VB

Summary

1. XSLT is a language for transforming XML documents into other XML documents using XPath as the query language.
2. Use the Load method of the XslTransform class to load an XSLT style sheet (or XSLT file).
3. The Transform method of the XslTransform class uses an XPathDocument, and either returns an XmlReader or writes the transformation into a supplied XmlWriter or stream.
4. The XslTransform class can transform over any data that implements XPathNavigator and the IXPathNavigable interface.
5. You can combine XSLT files to produce the required structure output format (for example, XML or HTML).
6. XSLT processors do not manipulate documents. They manipulate document structure.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

How Do I...Use an XPathExpression class?

This sample illustrates how to use an XPathExpression class to compile an XML Path Language (XPath) expression to select a node-set from an XML file. Then the sample demonstrates how to add a custom sort routine and use an XPathExpression to sort the elements returned by the compiled XPath. Finally, it also demonstrates how to use Extensible Stylesheet Language Transformations (XSLT) parameters for sorting with the reuse of a compiled XPath.

This sample uses the XPathNavigator class that provides cursor-style navigation over an XPathDocument. This sample also uses the XPathExpression class that represents a compiled XPath expression. This XPathExpression class is returned from the Compile method of XPathNavigator and is supplied to the Select, Evaluate, and Matches methods.

VB XPathExpression.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

First, the sample compiles an XPath expression to select a node-set from an XML file. To do this, the sample begins by creating an XPathDocument and loading the books.xml file into that object. Then, the sample creates an XPathNavigator from the XPathDocument. The sample uses the XPathNavigator object to move over the document using XPathNodeIterator.

```
Dim myXPathDocument as XPathDocument  = new XPathDocument(args)

' Create an XPathNavigator
Dim myXPathNavigator as XPathNavigator  = myXPathDocument.CreateNavigator()
```

VB

After creating an XPath expression that selects the book elements of the XPathDocument, the sample compile the expression into the XPathNavigator and then uses a Select method to execute the XPath expression. The sample displays the resulting XML data using the FormatXml function. By passing an XPathNodeIterator to this function, the function is able to consecutively select each node for formatting.

```
' Get the Book elements
selectExpr = "bookstore/book"

Console.WriteLine("XPath query: " + selectExpr)

' Ensure we are at the root node
myXPathNavigator.MoveToRoot()

myXPathExpr = myXPathNavigator.Compile(selectExpr)
...
```

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
' Create an XPathNodeIterator to walk over the selected nodes
Dim myXPathNodeIterator as XPathNodeIterator  = myXPathNavigator.Select (myXPathExpr)

FormatXml(myXPathNodeIterator)
```

VB

The second part of this sample shows how to use an XPath expression to sort a node-set.

This second section uses the same compiled XPathExpression used in the first part of the sample to select the book elements. However, the code in the second part adds a custom sort routine to sort the results by the attribute value for the ISBN attribute. The sample then displays the resulting XML using the FormatXml function and an XPathNodeIterator.

The AddSort method of an XPathExpression supports a generic sort functionality that the XSLT processor applies to nodes in the node-set. This sorting occurs through the IComparer interface. This interface provides an implementation of the Compare method that compares the objects on their data type.

```
' Create comparision class for the ISBN number
Dim myISBNCompare as ISBNCompare  = new ISBNCompare()

' Add a comparer to do a string compare on the ISBN number
myXPathExpr.AddSort("@ISBN", myISBNCompare)

myXPathNodeIterator = myXPathNavigator.Select (myXPathExpr)

FormatXml(myXPathNodeIterator)
```

VB

To demonstrate the last part of this sample where sorting is through XSLT parameters, there is a custom sort routine that uses the AddSort method of the XPathExpression to implement a new IComparer routine. By providing this custom sort capability, you are able to sort user-defined classes. This is useful for data that cannot be easily compared, like a string for example.

```
public Class ISBNCompare : Implements IComparer
    'Use a function name other than 'Compare'
    Function MyCompare( First As Object, Second As Object ) As Integer Implements IComparer.Compare
        Dim s1 as String  = CStr(First)
        Dim s2 as String  = CStr(Second)

        'Console.WriteLine("ISBN's to Compare: 1. {0} 2. {1}", s1, s2);
        return s1.ToString().CompareTo (s2.ToString ())
    end function
end Class 'ISBNCompare
```

VB

The sample then creates a second XPathNavigator on the XPathDocument and adds a sort based on an XSLT parameter to the XPathExpression.

The XPathExpression supports the functionality and adds the specified sort criteria to the XPath. Upon calling the XPathExpression, the XSLT processor sorts the nodes in the node-set according to the supplied values. Typically, you use the sort method through the element in an XSLT file.

```
' Create Second XPathNavigator
Dim myXPathNavigator2 as XPathNavigator  = myXPathDocument.CreateNavigator()

' Ensure we are at the root node
myXPathNavigator2.MoveToRoot()
myXPathExpr = myXPathNavigator2.Compile(selectExpr)
...
' Add an XSLT based sort
myXPathExpr.AddSort("@publicationdate", XmlSortOrder.Descending, XmlCaseOrder.None, "", XmlDataType.Text)

myXPathNodeIterator = myXPathNavigator2.Select (myXPathExpr)

FormatXml(myXPathNodeIterator)
```

VB

Summary

1. Use an XPathExpression class to compile an XPath expression to select a node-set from an XML file.
2. Use the XPathNavigator to provide fast navigation over an XPathDocument

---

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Use XML and the DataSet?

These topics provide an introduction to the interaction between the DataSet and XMLDataDocument objects.

XML and Relational Data. Same Data, Two Views.

XML often has a relational structure (the fictional books in a bookstore for example), as well as being unstructured. This set of topics cover the concept of storing this data centrally and providing different views on this data, either as XML or relationally as tables, columns and rows with relationships. This disconnected store of data which, for example, could represent a business object in the middle tier that enforces business rules, could provide its data as XML to a browser via Extensible Stylesheet Language Transformations (XSLT), across the Internet to another Web site, or to a local application via the relational tables. However, the data gets amended, you can update that data to a database on a transaction basis.

A DataSet represents an in-memory cache of data as a collection of tables and relationships between those tables. It is, in effect, a locally cached database. This provides a disconnected cache of data, like a message, that enables dealing with chunks of data. The DataSet has no knowledge of where the data came from. It may have come from a file, a database connection, or from a stream. A DataSet provides a relational view onto this stored data.

The XmlDataDocument provides XML APIs for accessing this in-memory cache of data, as well as supporting reading and writing XML. The XmlDataDocument is a DataSet-aware object. Creation of an XmlDataDocument implicitly creates a DataSet (accessed as a property) that provides a relational view onto the data XML data. This symbiotic relationship between these two objects provides a powerful technique for accessing data either relationally or as XML, irrespective of the mechanism by which the data was sourced.

XmlDataDocument and DataSet Differences

There are differences between these two objects and their views. XML loaded via the Load method on the XmlDataDocument maintains full fidelity-that is, all the XML structure is maintained and can be accessed via the XML API. The ReadXml method on the DataSet, however, only loads the XML that conforms to the internally created mappings (schema). You lose white space and element ordering, as well as data that does not conform to the relational mapping. Hence, the DataSet does not retain XML fidelity. Also data which has been loaded through an XmlDataDocument and which does not conform to the internal mappings on the DataSet cannot be viewed using the relational API. Whereas, relational data can always be viewed via the XML API.

Both the DataSet and XmlDataDocument can save data. However, they do not save the data identically. Using the WriteXml method, the DataSet is able to save a normalized view of the relationally mapped data (that is, the XML data that

conforms to the internal schema, with elements ordered according to the relational schema). This contrasts to the Save method of the XmlDataDocument that saves the XML data with full fidelity -all the XML that has been loaded into the XmlDataDocument, including the preservation of white space and element ordering.

Topics

The following topics work with XML data, DataSets and the XmlDataDocument:

1. How do I...Create DataSet mappings from an XSD schema? covers how a DataSet can build a relational view according to a supplied XML Schema Definition language (XSD) schema
2. How do I...Load a DataSet with XML? covers how XML data gets loaded into the DataSet and accessed as relational data.
3. How do I...Infer DataSet mappings from XML? covers how to infer a relational mapping from XML data, rather than an XSD schema.
4. How do I...Save DataSet mappings to an XSD schema file? shows how to save an inferred schema generated from loaded XML data.
5. How do I...Save a DataSet as XML? covers how to access and save XML that has been loaded into the DataSet relationally.

Related Topics

The following topics cover many of the features of the XmlDataDocument object that utilize the XML API. Refer to these topics for information about those scenarios in which you want to access data as XML.

1. How do I...Load and use the XmlDataDocument?
2. How do I...Read XML from an XmlDataDocument?

For coverage of the interaction between the DataSet and databases using the .NET data providers, see How do I...Get an Overview of ADO.NET?

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema

How Do I...Create DataSet mappings from an XSD schema?

This sample illustrates how to create DataSet mappings from a supplied XML Schema Definition language (XSD) schema. Generally, schemas are metadata, or data about data, but XSD schemas also cover the relationships between data types. From a schema, you can create a relational structure of tables and columns to store data that fits the supplied schema. This is the DataSet relational mapping of the schema.

VB DataSetMapXSDSchema.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The DataSet class has a close affiliation with the XmlDataDocument class. The DataSet class provides a relational view of the XML data loaded through the XmlDataDocument. To build a relational mapping between these two views, you need to use the ReadXmlSchema method of the DataSet. You use this method when you create the XmlDataDocument as in the following call, XmlDataDocument.DataSet.ReadXmlSchema. Now, any changes made to the DataSet are reflected in the XmlDataDocument and vice versa.

To illustrate this relationship, the following sample code reads and loads the books.xsd schema file into the DataSet property on an XmlDataDocument. Notice how the ReadXmlSchema method takes the schema loaded into a StreamReader and generates the relational mappings. If there is already a relational view defined in the XmlDataDocument, the sample throws an exception.

```
Dim myStreamReader as StreamReader = nothing
try
    myStreamReader = new StreamReader(schema)
    Console.WriteLine("Reading Schema file ...")
    myXmlDataDocument.ReadXmlSchema(myStreamReader)
catch e as exception
    Console.WriteLine ("Exception: " & e.ToString())
finally
    If Not myStreamReader Is Nothing
        myStreamReader.Close()
    end if
end try
```

VB

So how do you know what the internal tables generated from the schema look like? Well, the DataSet has a Tables property that is a collection of the internal tables. Each table has a Columns collection and each column has a ColumnName and a DataType. You can simply iterate over these collections and format the output to display the internal table structure built from the supplied schema, as

Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

shown in the following code.

```
' Displays the DataSet tables structure
private sub DisplayTableStructure()
    Console.WriteLine()
    Console.WriteLine("Table structure")
    Console.WriteLine()
    Console.WriteLine("Tables count=" & myXmlDataDocument.Tables.Count.ToString())

    Dim i,j as integer

    for i = 0 to (myXmlDataDocument.Tables.Count - 1)
        Console.WriteLine("TableName='" & myXmlDataDocument.Tables(i).TableName & "'.")
        Console.WriteLine("Columns count=" & myXmlDataDocument.Tables(i).Columns.Count.ToString())

        for j = 0 to (myXmlDataDocument.Tables(i).Columns.Count - 1)
            Console.WriteLine(Strings.chr(9) & "ColumnName='" & _
                              myXmlDataDocument.Tables(i).Columns(j).ColumnName & _"', type = " & _
                              myXmlDataDocument.Tables(i).Columns(j).DataType.ToString())
        next
        Console.WriteLine()
    next
end sub
```

VB

The following output shows the table names, column names, and the column types displayed by the DisplayTableStructure method for the books.xsd schema.

```
Table structure

Tables count=3
        TableName='bookstore'.
        Columns count=1
                ColumnName='bookstore_Id', type = System.Int32
        TableName='book'.
        Columns count=5
                ColumnName='title', type = System.String
                ColumnName='price', type = System.Decimal
                ColumnName='genre', type = System.String
                ColumnName='book_Id', type = System.Int32
                ColumnName='bookstore_Id', type = System.Int32
        TableName='author'.
        Columns count=3
                ColumnName='first-name', type = System.String
                ColumnName='last-name', type = System.String
                ColumnName='book_Id', type = System.Int32
```

Summary

1. Generally, schemas are metadata, or data about data, but XSD schemas also cover the relationships between types of data.
2. From an XSD schema, you can create a relational structure of tables and columns to store data that fits the supplied schema. This is the DataSet mapping.
3. The ReadXmlSchema method of the DataSet generates the internal mappings from the supplied schema.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

How Do I...Load a DataSet with XML?

This sample illustrates how to load a DataSet with XML Data. The sample builds on the topic, How Do I...Create DataSet mappings from an XSD schema?, by first loading XML data into an XmlDataDocument and then accessing this data from the DataSet. The DataSet has previously loaded a schema in order to create the internal mappings. This sample in effect shows the transition between XML data and the creation of relational objects to access that XML data.

VB LoadDataSetXMLData.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

As shown in the following code, this sample first uses the ParseSchema method to load the XML Schema Definition language (XSD) schema, books.xsd, into the DataSet property on the XmlDataDocument. Then, the sample uses the Load method of the XmlDataDocument to load the XML file, books.xml.

```
private const document as string = "books.xml"
private const myLoadSchema as string = "books.xsd"
private myXmlDataDocument as XmlDataDocument

shared sub Main()
    Dim args As String() = {document, myLoadSchema}
    Dim myLoadDataSetXMLDataSample as LoadDataSetXMLDataSample
    myLoadDataSetXMLDataSample = new LoadDataSetXMLDataSample()
    myLoadDataSetXMLDataSample.Run(args)
end sub

public sub Run(args As String())
    try
        Console.WriteLine("Creating an XmlDataDocument ...")
        myXmlDataDocument = new XmlDataDocument()
        ParseSchema(args(1))
        DisplayTableStructure()

        myXmlDataDocument.Load(args(0))

        DisplayTables(myXmlDataDocument.DataSet)

    catch e as exception
        Console.WriteLine ("Exception: " & e.ToString())
    end try
end sub
```

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

VB

As in How Do I...Create DataSet Mappings from an XSD schema?, the DisplayTableStructure method (built with the books.xsd schema file) enables the sample to display the internal table structure by simply iterating over the collections of Tables, Columns, and Rows, and then formatting the output. This sample extends this concept with the DisplayTables method (shown in the following code) which enables the sample to display the contents of the XML file. This sample uses the For Each keyword instead of a For loop to illustrate an alternative mechanism for iterating over the collections.

```
' Displays the contents of the DataSet tables
private sub DisplayTables(myDataset as DataSet)

    ' Navigate Dataset
    Console.WriteLine()
    Console.WriteLine("Content of Tables ...")

    Dim table as DataTable
    for each table in myDataset.Tables

        Console.WriteLine("TableName = " & table.TableName.ToString())
        Console.WriteLine ("---------")
        Console.WriteLine("Columns ...")

        Dim column as DataColumn
        for each column in table.Columns

            Console.Write("{0,-22}",column.ColumnName.ToString())
        next
        Console.WriteLine()
        Console.WriteLine("Number of rows = {0}", table.Rows.Count.ToString())
        Console.WriteLine("Rows ...")

        Dim row as DataRow
        for each row in table.Rows

        Dim value as object
            for each value in row.ItemArray
                Console.Write("{0,-22}",value.ToString())
            Next
            Console.WriteLine()
        Next
        Console.WriteLine()
    Next
end sub
```

VB

The following output shows the table names, column names, and row contents of books.xml as displayed by the DisplayTables method.

```
Creating an XmlDataDocument ...
Reading Schema file ...
```

```
Table structure

Tables count=3
        TableName='bookstore'.
        Columns count=1
                ColumnName='bookstore_Id', type = System.Int32
        TableName='book'.
        Columns count=5
                ColumnName='title', type = System.String
                ColumnName='price', type = System.Decimal
                ColumnName='genre', type = System.String
                ColumnName='book_Id', type = System.Int32
                ColumnName='bookstore_Id', type = System.Int32
        TableName='author'.
        Columns count=3
                ColumnName='first-name', type = System.String
                ColumnName='last-name', type = System.String
                ColumnName='book_Id', type = System.Int32
Content of Tables ...

TableName = bookstore
---------
Columns ...

bookstore_Id

Number of rows = 1
Rows ...

0

TableName = book
---------
Columns ...

title                price                genre                book_Id              bookstore_Id

Number of rows = 3
Rows ...

The Autobiography of Benjamin Franklin8.99                autobiography        0                    0
The Confidence Man   11.99                novel                1                    0
The Gorgias          9.99                 philosophy           2                    0

TableName = author
---------
Columns ...

first-name           last-name            book_Id

Number of rows = 3
Rows ...

Benjamin             Franklin             0
Herman               Melville             1
Sidas                Plato                2
```

Summary

1. XML data that has been loaded into an XmlDataDocument can be accessed via the relational methods on the DataSet property.
2. XML data can also be read when relational data has been entered through the DataSet property of XmlDataDocument.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

How Do I...Infer DataSet mappings from XML?

This sample illustrates how to create DataSet mappings from XML Data. This sample uses the concept of inference, whereby the actual structure of the XML data itself is the basis for a relational schema. The sample then creates a relational structure of tables and columns to store data in a DataSet that conforms to the inferred schema.

VB DataSetMapXMLData.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

In order to infer a schema from the XML data structure, the following sample code demonstrates how you should call DataSet.ReadXml using InferSchema for the XmlReadMode. If you do not specify an XmlReadMode, the ReadXml method infers a schema if both the dataset does not already have a schema and the document does not contain an inline schema.

```
' Infer the DataSet schema from the XML data and load the XML Data
datadoc.DataSet.ReadXml(new StreamReader("books.xml"), XmlReadMode.InferSchema)
```

VB

As in How Do I...Load a DataSet with XML?, this sample also uses the DisplayTables method to display the structure and contents of the DataSet. The following output shows the table names, column names, and row contents generated when the DisplayTables method uses the schema inferred from the books.xml file.

```
Creating an XmlDataDocument ...
Content of Tables ...

TableName = book
---------
Columns ...

title                book_Id              price                genre                publicationdate       ISBN

Number of rows = 3
Rows ...

The Autobiography of Benjamin Franklin 0                    8.99                 autobiography         1981
1-861003-11-0
The Confidence Man   1                    11.99                novel                1967                  0-201-
63361-2
The Gorgias          2                    9.99                 philosophy           1991                  1-861001-
57-6

TableName = author
---------
Columns ...

first-name           last-name            book_Id
```

Back to Index

Get URL for this page

```
Number of rows = 3
Rows ...

Benjamin            Franklin            0
Herman              Melville            1
Sidas               Plato               2
```

Summary

1. Inference is the mechanism whereby the structure of the XML data is the basis for a relational schema.
2. The ReadXml method of the DataSet builds relational mapping using the supplied XML file in a StreamReader.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model

How Do I...Save DataSet mappings to an XSD schema file?

This sample illustrates how to save the internal DataSet mappings to an XML Schema Definition language (XSD) schema file. This sample builds the mappings by using the relational methods on the DataSet to create tables and columns. The sample then writes an XSD schema representation of these mappings out to a file.

VB SaveDataSetMapXSDSchema.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The DataSet and XmlDataDocument classes both represent an in-memory data cache. The DataSet provides relationally orientated navigational and editing methods, while the XmlDataDocument provides XML navigational and editing methods.

This sample gets the DataSet property from the XmlDataDocument, uses it to build a set of tables and columns, and then populate those tables and columns. The sample then writes out the internally generated schema.

The following sample code builds two tables, one of people and the other of pets. The sample uses an ID as a primary key into each of the tables, and builds a relationship table between the people and their pets. To do this, the sample first creates an instance of an XmlDataDocument and then passes the DataSet associated with the XmlDataDocument to the LoadDataSet method.

```
Dim datadoc As XmlDataDocument = New XmlDataDocument()
LoadDataSet(datadoc.DataSet)
```

VB

The LoadDataSet method loads the dataset with relational data.

```
' Load a DataSet with relational data
private sub LoadDataSet(dataset as DataSet)

    try
        Console.WriteLine("Loading the DataSet ...")

        ' Set DataSet name
        dataset.DataSetName = "PersonPet"

        ' Create tables for people and pets
        Dim people as DataTable = new DataTable("Person")
        Dim pets as DataTable = new DataTable("Pet")
```

Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
' Set up the columns in the Tables
Dim personname as DataColumn = new DataColumn ("Name", GetType (System.String))
Dim personAge as DataColumn = new DataColumn ("Age", GetType (System.Int32))

Dim petname as DataColumn = new DataColumn ("Name", GetType (System.String))
Dim pettype as DataColumn = new DataColumn ("Type", GetType (System.String))

' Add columns to person table
Dim id as DataColumn = people.Columns.Add("ID", GetType (System.Int32))
id.AutoIncrement = true
Dim primarykey as DataColumn() = new DataColumn() {id}
people.PrimaryKey = primarykey
people.Columns.Add (personname)
people.Columns.Add (personAge)

' Add columns to pet table
id = pets.Columns.Add("ID", GetType (System.Int32))
id.AutoIncrement = true
pets.PrimaryKey = new DataColumn() {id}
id.AutoIncrement = true
Dim ownerid as DataColumn = pets.Columns.Add("OwnerID", GetType (System.Int32))
Dim foreignkey as DataColumn() = new DataColumn() {ownerid}
pets.Columns.Add (petname)
pets.Columns.Add (pettype)

' Add tables to the DataSet
dataset.Tables.Add (people)
dataset.Tables.Add (pets)

' Add people
Dim mark as DataRow = people.NewRow()
mark(personname) = "Mark"
mark(personAge) = 18
people.Rows.Add(mark)

Dim william as DataRow = people.NewRow()
william(personname) = "William"
william(personAge) = 12
people.Rows.Add(william)

Dim james as DataRow = people.NewRow()
james(personname) = "James"
james(personAge) = 19
people.Rows.Add(james)

Dim levi as DataRow = people.NewRow()
levi(personname) = "Levi"
levi(personAge) = 4
people.Rows.Add(levi)

' Add relationships
Console.WriteLine("Creating relationships between people and pets ...")
Dim personpetrel as DataRelation = new DataRelation ("PersonPet", primarykey, foreignkey, false)
dataset.Relations.Add (personpetrel)
```

```
            ' Add pets
            Dim row as DataRow = pets.NewRow()
            row("OwnerID") = mark("ID")
            row(petname) = "Frank"
            row(pettype) = "cat"
            pets.Rows.Add(row)

            row = pets.NewRow()
            row("OwnerID") = william("ID")
            row(petname) = "Rex"
            row(pettype) = "dog"
            pets.Rows.Add(row)

            row = pets.NewRow()
            row("OwnerID") = james("ID")
            row(petname) = "Cottontail"
            row(pettype) = "rabbit"
            pets.Rows.Add(row)

            row = pets.NewRow()
            row("OwnerID") = levi("ID")
            row(petname) = "Sid"
            row(pettype) = "snake"
            pets.Rows.Add(row)

            row = pets.NewRow()
            row("OwnerID") = levi("ID")
            row(petname) = "Tickles"
            row(pettype) = "spider"
            pets.Rows.Add(row)

            row = pets.NewRow()
            row("OwnerID") = william("ID")
            row(petname) = "Tweetie"
            row(pettype) = "canary"
            pets.Rows.Add(row)

            ' commit changes
            dataset.AcceptChanges()

        catch e as Exception
            Console.WriteLine ("Exception: {0}", e.ToString())
        end try
    end sub
```

VB

The AcceptChanges method of the DataSet commits all the changes that have been made to this DataSet since it was loaded or the last time AcceptChanges was called. All new and modified rows become unchanged, and deleted rows get removed. For more detail on the other DataSet relational methods, see How do I...Get an Overview of ADO.NET?

To save the schema to a file, the sample calls the WriteXmlSchema method of the DataSet, passing a StreamWriter class that represents the

destination file.

```
Dim writer as StreamWriter

try
    Console.WriteLine("Writing the schema to {0} ...", mySaveSchema)
    writer = new StreamWriter(mySaveSchema)
    datadoc.DataSet.WriteXmlSchema(writer)
catch e as Exception
    Console.WriteLine ("Exception: {0}", e.ToString())
finally
    If Not writer Is Nothing
        writer.Close()
    end if
end try
```

VB

The following output shows the tables built in the DataSet by the DisplayTables method. For more information about the DisplayTables method, see How do I...Infer DataSet Mappings from XML?. The sample writes the inferred schema to the PersonPet.xsd file.

```
Loading the DataSet ...
Creating relationships between people and pets ...

DataSet:
PersonPet contains ...
No of Tables: 2  Table content ...

TableName = Person
---------
Columns ...

ID                    Name                  Age

Number of rows = 4
Rows ...
0                     Mark                  18
1                     William               12
2                     James                 19
3                     Levi                  4

TableName = Pet
---------
Columns ...

ID                    OwnerID               Name                  Type

Number of rows = 6
Rows ...
0                     0                     Frank                 cat
1                     1                     Rex                   dog
```

```
2                   2                   Cottontail          rabbit
3                   3                   Sid                 snake
4                   3                   Tickles             spider
5                   1                   Tweetie             canary

PersonPet

Name = Mark owns
        Pet = Frank the cat

Name = William owns
        Pet = Rex the dog
        Pet = Tweetie the canary

Name = James owns
        Pet = Cottontail the rabbit

Name = Levi owns
        Pet = Sid the snake
        Pet = Tickles the spider

Writing the schema to PersonPet.xsd ...
```

Summary

1. The WriteXmlSchema method saves the mappings for the internal structure of the relation data in the DataSet as an XSD Schema.
2. The XmlDataDocument has a DataSet property that enables you to view and manage structured data relationally within an XML document.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Save a DataSet as XML?

This sample illustrates how to use XmlDataDocument to save relational data from a DataSet to an XML file using the. This demonstrates the transition from relational data to XML data.

VB SaveDataSetXMLData.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Having built the relational tables for the DataSet in How Do I...Save DataSet Mappings to an XSD schema file?, this sample now saves these tables as XML data. This is the process of generating hierarchical XML based upon and validated against the internally generated XML Schema Definition language (XSD) schema.

As shown in the following code, the sample first creates an XmlDataDocument for the DataSet.

```
' Load the DataSet with relation data
Dim dataset As New DataSet
LoadDataSet(dataset)
' Create an XmlDataDocument for the DataSet
Dim datadoc As XmlDataDocument = New XmlDataDocument(dataset)
```

VB

This example creates a DataSet using the LoadDataSet function. To ensure the correct loading of the DataSet, the sample displays the DataSet to the screen. The sample then uses the WriteXmlSchema method of the DataSet to write out the schema created by the DataSet. To output the contents of the DataSet as XML, the sample calls the WriteXml method of the DataSet using a file name. Then, the sample reads in the XML that it just wrote and display the data on the screen.

```
'Load the DataSet with relation data
Dim myDataSet as DataSet = new DataSet()
LoadDataSet(myDataSet)

DisplayTables(myDataSet)

'Write out schema representation
myDataSet.WriteXmlSchema(m_SchemaFile)

'Write out XML data form relational data
myDataSet.WriteXml(m_XmlFile, XmlWriteMode.IgnoreSchema)

'Create an XmlDataDocument for the DataSet
Dim datadoc as XmlDataDocument = new System.Xml.XmlDataDocument(myDataSet)

'Display the XML
DisplayXMLData(datadoc)
```

VB

The sample writes the XML data to the file PersonPet.xml. The XML data in XmlDataDocument can now be read with an XmlReader and displayed, showing the XML for the relational data. For more information about reading XML documents, see How do I...Read XML from an XmlDataDocument?.

Summary

1. You can construct the XmlDataDocument from a DataSet. This provides an XML API onto the relational data.
2. You can access data entered through the relational methods of the DataSet by using the XML methods of the XmlDataDocument.
3. You can save XML data by either using the WriteXml method of the DataSet or the Save method of the XmlDataDocument. The former saves a normalized view of the relationally mapped data, while the latter saves the XML with full fidelity. If the data has only been entered through the DataSet, then these methods are equivalent.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Read and Write a Schema?

This sample illustrates how to read an XML Schema Definition language (XSD) file into the Schema Object Model (SOM). Then, the sample writes the XSD schema which has been loaded into memory to a StringWriter for display to the screen. The schema could also be written to a file using an XmlWriter.

This SOM provides a set of classes which directly reflect the World Wide Web Consortium (W3C) XSD specification. The classes, methods, and properties of XmlSchema provide the ability to create an in-memory version of the schema that can be complied and used in validation through XmlSchemaCollection.

XmlSchema can load and save valid XSD schemas. It also has strongly typed classes which can be used to create in-memory schemas. For validation purposes, XmlSchema can be used by the XmlSchemaCollection and XmlValidatingReader classes.

VB XmlReadWriteSchema.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Typically, you use the XmlSchema if you need to load and/or create a valid schema. This sample simply loads a simple schema from a file.

```
private const document as String = "sample.xsd"
...
Dim mySchema as XmlSchema  = XmlSchema.Read(new XmlTextReader(document))
```

VB

Once loaded, the sample writes the schema to a StringWriter and displays the StringWriter on the screen.

```
mySchema.Write(myXmlWriter)
Console.WriteLine(myStringWriter.ToString())
```

VB

Summary

1. The Schema Object Model (SOM) provides a navigatable set of classes which directly reflect the W3C XSD specification.
2. A SOM is built for each of the imported and included schemas and these are held in the includes collection
3. The items collection is a list of all the schema element types at the schema level from the loaded schema. This is used for persistence.
4. Hashtables from Elements, Attributes, and so on, are built which reference all the schema element types at the schema level for all referenced schemas (that is, those in the includes collection), as well as this one. This provides an easy lookup mechanism on the element name.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model

How Do I...use the Schema Object Model?

This sample illustrates how to read two XML Schema Definition language (XSD) schemas into a SchemaCollection, navigate through the schemas that they represent, and create a fictious XML output that is representative of the schema's.

This sample shows how to use the XmlSchemaCollection to cache and retrieve multiple schemas, as well as demonstrates that the Schema Object Model (SOM) loads and saves valid XSD schemas. You can also use the SOM to create in-memory schemas using strongly typed classes.

To demonstrate how to navigate the SOM, this sample outputs a formatted version of the two XSD schemas that were loaded into the SchemaCollection. Then, the sample creates an XML stream to give an example of what the given schema usage might look like.

VB XmlSchemaObjectModel.aspx

[Run Sample] | [View Source]

This sample first creates an XmlTextWriter that eventually writes out to the screen. The sample does this to take advantage of the various methods of the XmlTextWriter that produce well-formed XML. Some examples of these methods are the WriteStartElement, WriteEndElement, and WriteAttributeString. Then, the sample creates an XmlNameTable and adds the name table to the XmlSchemaCollection. Into the XmlSchemaCollection, the sample read two unique XSD schemas. Finally, the sample writes each schema in the XmlSchemaCollection to the screen, followed by an example of that schema would be used against some example XML.

```
...
myXmlTextWriter = new XmlTextWriter(Console.Out)
myXmlTextWriter.Formatting = Formatting.Indented
myXmlTextWriter.Indentation = 2
...
Dim myXmlNameTable as XmlNameTable  = new NameTable()
...
Dim myXmlSchemaCollection as XmlSchemaCollection  = new XmlSchemaCollection(myXmlNameTable)
...
myXmlSchemaCollection.Add(nothing, args(0))
...
myXmlSchemaCollection.Add(nothing, args(1))
...

Dim myTempXmlSchema as XmlSchema
for each myTempXmlSchema in myXmlSchemaCollection
    myXmlSchema = myTempXmlSchema
...
```

Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
    ' Write out the various schema parts
    WriteXSDSchema()
...
    ' Write out an example of the XML for the schema
    WriteExample()
...
next
```

VB

The functions in the following code write the XSD schema to the screen. Essentially, these functions loop through each item in the XmlSchema and, after determining its type, format the item for output to the screen.

```
' Write out the XSD
private sub WriteXSDSchema()
    myXmlTextWriter.WriteStartElement("schema", XmlSchema.Namespace)
    myXmlTextWriter.WriteAttributeString("targetNamespace", myXmlSchema.TargetNamespace)
    Dim include as XmlSchemaInclude
    for each include in myXmlSchema.Includes
        myXmlTextWriter.WriteStartElement("include", XmlSchema.Namespace)
        myXmlTextWriter.WriteAttributeString("schemaLocation", include.SchemaLocation)
        myXmlTextWriter.WriteEndElement()
    next

    Dim item as object
    for each item in myXmlSchema.Items

        if (TypeOf item is XmlSchemaAttribute)
            WriteXmlSchemaAttribute(CType(item, XmlSchemaAttribute)) 'attribute
        else if (TypeOf item is XmlSchemaComplexType)
            WriteXmlSchemaComplexType(CType(item, XmlSchemaComplexType)) 'complexType
        else if (TypeOf item is XmlSchemaSimpleType)
            WriteXmlSchemaSimpleType(CType(item, XmlSchemaSimpleType)) 'simpleType
        else if (TypeOf item is XmlSchemaElement)
            WriteXmlSchemaElement(CType(item, XmlSchemaElement)) 'element
        else if (TypeOf item is XmlSchemaAnnotation)
            WriteXmlSchemaAnnotation(CType(item, XmlSchemaAnnotation)) 'annotation
        else if (TypeOf item is XmlSchemaAttributeGroup)
            WriteXmlSchemaAttributeGroup(CType(item, XmlSchemaAttributeGroup)) 'attributeGroup
        else if (TypeOf item is XmlSchemaNotation)
            WriteXmlSchemaNotation(CType(item, XmlSchemaNotation)) 'notation
        else if (TypeOf item is XmlSchemaGroup)
            WriteXmlSchemaGroup(CType(item, XmlSchemaGroup)) 'group
        else
            Console.WriteLine("Not Implemented.")
        end if
    next
    myXmlTextWriter.WriteEndElement()
end sub
```

VB

The functions in the following code write out what the XML that is validated to the schema might look like. Essentially, these functions loop through each item in the XmlSchema and, after determining its type, format the item with some sample data for output to the screen.

```
' Write out the example of the XSD usage
private sub WriteExample()
    Dim element as XmlSchemaElement
    for each element in myXmlSchema.Elements.Values
        WriteExampleElement(element)
    next
end sub

' Write some example elements
private sub WriteExampleElement(element as XmlSchemaElement )
    myXmlTextWriter.WriteStartElement(element.QualifiedName.Name, element.QualifiedName.Namespace)
    if (TypeOf element.ElementType is XmlSchemaComplexType)
        Dim type as XmlSchemaComplexType  = CType(element.ElementType, XmlSchemaComplexType)
        if not type.ContentModel is nothing
            Console.WriteLine("Not Implemented for this ContentModel")
        end if
        WriteExampleAttributes(type.Attributes)
        WriteExampleParticle(type.Particle)
    else
        WriteExampleValue(element.ElementType)
    end if
    myXmlTextWriter.WriteEndElement()
end sub
```

VB

Summary

1. The Schema Object Model (SOM) loads and saves valid XSD schemas.
2. The SOM provides an easy way to create in-memory schemas using strongly typed classes.
3. You can use the XmlSchemaCollection object to cache and retrieve multiple schemas.
4. The SOM works with the XmlValidatingReader through the XmlSchemaCollection.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Use XML Schema Collection?

This sample illustrates how to use the XmlSchemaCollection class used by the XmlValidatingReader for efficient data validation. The XmlSchemaCollection class allows you to maintain a cache of multiple XML Schema Definition language (XSD) and XML-Data Reduced (XDR) schemas which can then be used when a schema is required. However, while this class stores both XSD and XDR schemas, only XSD schemas apply to any method or property which takes or returns an XmlSchema.

Schemas in the XmlSchemaCollection can be used to validate XML data. When reading XML data, you can use an XmlValidatingReader to apply this validation. For information on how to use the features of the XmlValidatingReader, refer to How Do I...Apply validation when reading XML?.

Validation is the process of enforcing rules on the XML content either via a Document Type Definition (DTD) or a schema. At its simplest, a schema is an XML representation of a DTD, but it can also infer relationships within the XML data and type information. The XmlValidatingReader class layers validation support onto the XmlTextReader.

VB SchemaCollection.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

This sample shows how an XmlValidatingReader can use an XmlSchemaCollection for efficient data validation. To the schema collection, the sample adds the XSD schema, xmlsc.xsd, that is to be used for validation. Then, the sample creates an XmlValidatingReader and adds the schema from the schema collection to the reader. Finally, the sample uses the reader to read through the data and formats the output for display.

```
...
Dim reader as XmlValidatingReader
Dim xsc as XmlSchemaCollection = new XmlSchemaCollection()

try
    xsc.Add(args(1), new XmlTextReader(args(1)))
    reader = new XmlValidatingReader(new XmlTextReader(args(0)))
    reader.Schemas.Add(xsc)
    reader.ValidationType = ValidationType.Schema

    while (reader.Read())
      ...
    end while
    ...
end try
```

VB

Summary

1. You can use the XMLSchemaCollection as a cache of preloaded schemas.
2. These preloaded schemas get loaded once, then reused. This results in much better parsing performance.
3. The XmlSchemaCollection supports both XDR and XSD schemas.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...use XmlNamespaceManager and NameTable?

This sample shows how to select namespace specific XML data from an XmlDocument. It demonstrates the use of the XmlNamespaceManager class. This class encapsulates a collection of namespaces and provides a variety of namespace management tasks, such as resolving, adding, and removing namespaces to a collection, and providing scope management for the namespaces.

The XmlNameTable provides management of the string names in the XML stream. It does this by creating a table of atomized string objects. This table provides an efficient means for the XML parser to use the same String object for all repeated element and attribute names in an XML document.

This sample uses XML Path Language (XPath) expressions to query the namespaces. For more information about XPath expressions, see How Do I...Query XML with an XPath expression?

VB XmlNameSpace.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

First, the sample creates an XmlDocument and loads the orders.xml file into the XmlDocument.

```
'Create a new XmlDocument for the specified source file and load it.
Dim myXmlDocument as XmlDocument  = new XmlDocument()
myXmlDocument.Load("orders.xml")
```

VB

To get all the item elements for the yourns1 namespace, the samples uses the following XPath expression: //yourns1:item. The sample also creates an XmlNamespaceManager that contains the prefixes and namespaces in the XML document.

The orders.xml document defines two prefixes: myns for http://tempuri.org/myUSorderprocessornamespace and yourns1 for http://tempuri.org/USvendor1namespace. The document also has the default namespace (with a null prefix) of http://tempuri.org/myUSordersnamespace.

```
'Create a string containing the XPATH expression to evaluate.
sExpr = String.Format("//{0}:item", "yourns1")

'Create an XmlNamespaceManager and add the namespaces for the document.
Dim nsmanager as XmlNamespaceManager  = new XmlNamespaceManager(doc.NameTable)

'Set default namespace--first param is null.
nsmanager.AddNamespace(String.Empty, "http:'tempuri.org/myUSordersnamespace")
nsmanager.AddNamespace("myns", "http:'tempuri.org/myUSorderprocessornamespace")
nsmanager.AddNamespace("yourns1", "http:'tempuri.org/USvendor1namespace")
nsmanager.AddNamespace("yourns2", "http:'tempuri.org/USvendor2namespace")
```

VB

This sample then calls the SelectNodes method on the XmlDocument. In this case, the SelectNodes method finds all item nodes with a yourns1 prefix that resolves to the namespace, http://tempuri.org/USvendor1namespace.

The SelectNodes method returns an XmlNodeList. The sample then display this node list.

```
Dim nodelist as XmlNodeList = myXmlDocument.SelectNodes(exprString, nsmanager)

Dim myXmlNode as XmlNode
for each myXmlNode in nodelist
    DisplayTree(myXmlNode)
next
```

VB

Summary

1. The XmlNamespaceManager class resolves, adds, and removes namespaces to a collection.
2. An application can use an XmlNameTable to improve XML parser performance.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Use XML and the DataSet?

These topics provide an introduction to the interaction between the DataSet and XMLDataDocument objects.

XML and Relational Data. Same Data, Two Views.

XML often has a relational structure (the fictional books in a bookstore for example), as well as being unstructured. This set of topics cover the concept of storing this data centrally and providing different views on this data, either as XML or relationally as tables, columns and rows with relationships. This disconnected store of data which, for example, could represent a business object in the middle tier that enforces business rules, could provide its data as XML to a browser via Extensible Stylesheet Language Transformations (XSLT), across the Internet to another Web site, or to a local application via the relational tables. However, the data gets amended, you can update that data to a database on a transaction basis.

A DataSet represents an in-memory cache of data as a collection of tables and relationships between those tables. It is, in effect, a locally cached database. This provides a disconnected cache of data, like a message, that enables dealing with chunks of data. The DataSet has no knowledge of where the data came from. It may have come from a file, a database connection, or from a stream. A DataSet provides a relational view onto this stored data.

The XmlDataDocument provides XML APIs for accessing this in-memory cache of data, as well as supporting reading and writing XML. The XmlDataDocument is a DataSet-aware object. Creation of an XmlDataDocument implicitly creates a DataSet (accessed as a property) that provides a relational view onto the data XML data. This symbiotic relationship between these two objects provides a powerful technique for accessing data either relationally or as XML, irrespective of the mechanism by which the data was sourced.

XmlDataDocument and DataSet Differences

There are differences between these two objects and their views. XML loaded via the Load method on the XmlDataDocument maintains full fidelity-that is, all the XML structure is maintained and can be accessed via the XML API. The ReadXml method on the DataSet, however, only loads the XML that conforms to the internally created mappings (schema). You lose white space and element ordering, as well as data that does not conform to the relational mapping. Hence, the DataSet does not retain XML fidelity. Also data which has been loaded through an XmlDataDocument and which does not conform to the internal mappings on the DataSet cannot be viewed using the relational API. Whereas, relational data can always be viewed via the XML API.

Both the DataSet and XmlDataDocument can save data. However, they do not save the data identically. Using the WriteXml method, the DataSet is able to save a normalized view of the relationally mapped data (that is, the XML data that

conforms to the internal schema, with elements ordered according to the relational schema). This contrasts to the Save method of the XmlDataDocument that saves the XML data with full fidelity -all the XML that has been loaded into the XmlDataDocument, including the preservation of white space and element ordering.

Topics

The following topics work with XML data, DataSets and the XmlDataDocument:

1. How do I...Create DataSet mappings from an XSD schema? covers how a DataSet can build a relational view according to a supplied XML Schema Definition language (XSD) schema
2. How do I...Load a DataSet with XML? covers how XML data gets loaded into the DataSet and accessed as relational data.
3. How do I...Infer DataSet mappings from XML? covers how to infer a relational mapping from XML data, rather than an XSD schema.
4. How do I...Save DataSet mappings to an XSD schema file? shows how to save an inferred schema generated from loaded XML data.
5. How do I...Save a DataSet as XML? covers how to access and save XML that has been loaded into the DataSet relationally.

Related Topics

The following topics cover many of the features of the XmlDataDocument object that utilize the XML API. Refer to these topics for information about those scenarios in which you want to access data as XML.

1. How do I...Load and use the XmlDataDocument?
2. How do I...Read XML from an XmlDataDocument?

For coverage of the interaction between the DataSet and databases using the .NET data providers, see How do I...Get an Overview of ADO.NET?

XML Serialization
XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I create an XML Schema Definition (XSD) File From a Class?

This topic describes how to build an XSD file for classes that you save and load from XML using the XML Serialization classes. The XML Serialization classes allow you to serialize an object as XML . Often you will start with an XML Schema Definition (XSD) file, and use the XSD.exe tool to build the classes you save and load from . However, if you started by building your classes first, and then decided to save them as XML, you might want to generate an XML Schema Definition (XSD) file that describes the format that the serializer will use for your objects. This file provides the information that consumers of your persisted classes need to know to properly read them.

1. Locate the assembly and class you want to export. You only need to know the name of the class that corresponds to the root element of the XML document. This example will use the following **PurchaseOrder** type:

   VB PurchaseOrder.vb

   [Run Sample] | [View Source]

   In the code, you will notice additional "Custom Metadata Attributes" above the field declarations. These are used to guide the Xml Serializer classes to properly read and write the XML. They are basically hints to the parser. For example, the **Items** field has a custom attribute applied to it to tell the serializer to that the **Items** field should contain an array of item objects which get created when the XML is loaded. Since they affect what XML is used to save the object, they also affect the schema that is generated:

   ```
   <System.Xml.Serialization.XmlArrayAttribute(IsNullable:=false), _
    System.Xml.Serialization.XmlArrayItemAttribute("item", IsNullable:=false) _
   Public items() As ItemsItem
   ```

   VB

   For detailed information on these attributes, see the reference documentation in the SDK. They are located in the System.Xml.Serialization namespace.

2. Use the xsd.exe tool in the SDK to generate an XSD schema from the class and all classes used by that class. The **-t** switch tells the program only to generate a schema for one particular type in the test.exe file:

   ```
   xsd.exe -t:PurchaseOrder PurchaseOrder.dll
   ```

3. The result is an XSD schema that describes what the XML file will look like when you serialize your object using the XML Serialization classes:

   PurchaseOrder.XSD

   [Run Sample] | [View Source]

XML Serialization
XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I create an XML Schema Definition (XSD) File From a Class?

This topic describes how to build an XSD file for classes that you save and load from XML using the XML Serialization classes. The XML Serialization classes allow you to serialize an object as XML . Often you will start with an XML Schema Definition (XSD) file, and use the XSD.exe tool to build the classes you save and load from . However, if you started by building your classes first, and then decided to save them as XML, you might want to generate an XML Schema Definition (XSD) file that describes the format that the serializer will use for your objects. This file provides the information that consumers of your persisted classes need to know to properly read them.

1. Locate the assembly and class you want to export. You only need to know the name of the class that corresponds to the root element of the XML document. This example will use the following **PurchaseOrder** type:

   VB PurchaseOrder.vb

   [Run Sample] | [View Source]

   In the code, you will notice additional "Custom Metadata Attributes" above the field declarations. These are used to guide the Xml Serializer classes to properly read and write the XML. They are basically hints to the parser. For example, the **Items** field has a custom attribute applied to it to tell the serializer to that the **Items** field should contain an array of item objects which get created when the XML is loaded. Since they affect what XML is used to save the object, they also affect the schema that is generated:

   ```
   <System.Xml.Serialization.XmlArrayAttribute(IsNullable:=false), _
    System.Xml.Serialization.XmlArrayItemAttribute("item", IsNullable:=false) _
   Public items() As ItemsItem
   ```

   VB

   For detailed information on these attributes, see the reference documentation in the SDK. They are located in the System.Xml.Serialization namespace.

2. Use the xsd.exe tool in the SDK to generate an XSD schema from the class and all classes used by that class. The **-t** switch tells the program only to generate a schema for one particular type in the test.exe file:

   ```
   xsd.exe -t:PurchaseOrder PurchaseOrder.dll
   ```

3. The result is an XSD schema that describes what the XML file will look like when you serialize your object using the XML Serialization classes:

   PurchaseOrder.XSD

   [Run Sample] | [View Source]

XML Serialization

XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I create classes from an XSD schema?

This topic describes how to create classes that can be used to read and write XML that is defined by an XML Schema Description (XSD) file. Once you understand how to generate the classes for your XSD schema, you should read the "How Do I Read and Write Objects From XML?" topic to find out how to actually load XML into them, and save XML from them.

1. Locate an XSD file. In this example, we will use a purchase order schema, as follows:

PurchaseOrder.XSD

[Run Sample] | [View Source]

2. Use the xsd.exe tool in the SDK to generate source code for the classes you want. Do this by specifying the name of the XSD file as an argument to the executable. The example below creates classes in C# using the purchaseorder.xsd schema and puts them in the **XmlSerializationHowTo** namespace:

```
xsd.exe -c -l:c# -n:XmlSerializationHowTo purchaseorder.xsd
```

3. The tool uses the XSD file to create a set of classes that can read XML files that use the schema that you specified. Here is the code that gets created for the above schema:

VB PurchaseOrder.vb

[Run Sample] | [View Source]

In the code, you will notice additional "Custom Metadata Attributes" above the field declarations. These are used to guide the Xml Serializer classes in properly reading and writing the XML. They are basically hints to the parser. For example, the **Items** field has a custom attribute applied to it to tell the serializer to that the **Items** field should contain an array of item objects which get created when the XML is loaded.

```
<System.Xml.Serialization.XmlArrayAttribute(IsNullable:=false),  _
 System.Xml.Serialization.XmlArrayItemAttribute("item", IsNullable:=false)>  _
Public items() As ItemsItem
```

VB

For detailed information on these attributes, see the reference documentation in the SDK. They are located in the System.Xml.Serialization namespace.

Now that you understand how to generate the classes for your XSD schema, you should read the "How Do I Read and Write Objects From XML?" topic to find out how to actually load XML into them, and save XML from them.

XML Serialization
XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I read/write objects into xml?

This topic describes how to automatically map a particular XML stream into a set of objects designed to hold the XML using the XML Serialization classes.

1. First, you will need classes designed to work with the XML Serialization classes. If you have an XSD schema that describes the format of the XML file you want to load or save, you are almost there. Use the XSD.exe tool to create these classes automatically. You can also build the classes manually. You can find more information on how to do this in the "How do I Create an XSD schema from a class?" topic.

   This example will be using the **PurchaseOrder** classes that were generated to read XML in the format defined by the purchase order schema. Both the classes and the schema are shown below.

   VB PurchaseOrder.vb

   [View Source]

   PurchaseOrder.XSD

   [View Source]

2. Create an instance of the **XmlSerializer**, passing the type of the object you would like to deserialize. Here we will use the **PurchaseOrder** type that was defined earlier.

   ```
   XmlSerializer serializer = new XmlSerializer(typeof(PurchaseOrder));
   ```

3. To read the file, call the Deserialize method passing in a Stream, TextReader, or XmlReader. A purchase order will be returned.

   ```
   Dim reader As TextReader = New StreamReader("po.xml")
   Dim po As PurchaseOrder = CType(serializer.Deserialize(reader), PurchaseOrder)
   reader.Close()
   ```

   VB

4. To write the file, call the Serialize method passing in a **Stream**, **TextReader**, or **XmlReader** as well as an instance of the purchase order.

   ```
   Dim writer As TextWriter = New StreamWriter("po2.xml")
   serializer.Serialize(writer, po)
   writer.Close()
   ```

   VB

The following example puts these ideas together by reading the purchase order XML file shown below, and then writing it back out to another file.

VB ReadWriteXML

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

PurchaseOrder.xml

[View Source]

XML Serialization
XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I read/write objects into xml?

This topic describes how to automatically map a particular XML stream into a set of objects designed to hold the XML using the XML Serialization classes.

1. First, you will need classes designed to work with the XML Serialization classes. If you have an XSD schema that describes the format of the XML file you want to load or save, you are almost there. Use the XSD.exe tool to create these classes automatically. You can also build the classes manually. You can find more information on how to do this in the "How do I Create an XSD schema from a class?" topic.

   This example will be using the **PurchaseOrder** classes that were generated to read XML in the format defined by the purchase order schema. Both the classes and the schema are shown below.

   VB PurchaseOrder.vb

   [View Source]

   PurchaseOrder.XSD

   [View Source]

2. Create an instance of the **XmlSerializer**, passing the type of the object you would like to deserialize. Here we will use the **PurchaseOrder** type that was defined earlier.

   ```
   XmlSerializer serializer = new XmlSerializer(typeof(PurchaseOrder));
   ```

3. To read the file, call the Deserialize method passing in a Stream, TextReader, or XmlReader. A purchase order will be returned.

   ```
   Dim reader As TextReader = New StreamReader("po.xml")
   Dim po As PurchaseOrder = CType(serializer.Deserialize(reader), PurchaseOrder)
   reader.Close()
   ```

   VB

4. To write the file, call the Serialize method passing in a **Stream**, **TextReader**, or **XmlReader** as well as an instance of the purchase order.

   ```
   Dim writer As TextWriter = New StreamWriter("po2.xml")
   serializer.Serialize(writer, po)
   writer.Close()
   ```

   VB

The following example puts these ideas together by reading the purchase order XML file shown below, and then writing it back out to another file.

VB ReadWriteXML

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

PurchaseOrder.xml

[View Source]

XML Serialization

XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I create classes from an XSD schema?

This topic describes how to create classes that can be used to read and write XML that is defined by an XML Schema Description (XSD) file. Once you understand how to generate the classes for your XSD schema, you should read the "How Do I Read and Write Objects From XML?" topic to find out how to actually load XML into them, and save XML from them.

1. Locate an XSD file. In this example, we will use a purchase order schema, as follows:

PurchaseOrder.XSD

[Run Sample] | [View Source]

2. Use the xsd.exe tool in the SDK to generate source code for the classes you want. Do this by specifying the name of the XSD file as an argument to the executable. The example below creates classes in C# using the purchaseorder.xsd schema and puts them in the **XmlSerializationHowTo** namespace:

```
xsd.exe -c -l:c# -n:XmlSerializationHowTo purchaseorder.xsd
```

3. The tool uses the XSD file to create a set of classes that can read XML files that use the schema that you specified. Here is the code that gets created for the above schema:

VB PurchaseOrder.vb

[Run Sample] | [View Source]

In the code, you will notice additional "Custom Metadata Attributes" above the field declarations. These are used to guide the Xml Serializer classes in properly reading and writing the XML. They are basically hints to the parser. For example, the **Items** field has a custom attribute applied to it to tell the serializer to that the **Items** field should contain an array of item objects which get created when the XML is loaded.

```
<System.Xml.Serialization.XmlArrayAttribute(IsNullable:=false),  _
 System.Xml.Serialization.XmlArrayItemAttribute("item", IsNullable:=false)>  _
Public items() As ItemsItem
```

VB

For detailed information on these attributes, see the reference documentation in the SDK. They are located in the System.Xml.Serialization namespace.

Now that you understand how to generate the classes for your XSD schema, you should read the "How Do I Read and Write Objects From XML?" topic to find out how to actually load XML into them, and save XML from them.

The following example makes four arrays of types int, double, string, and MyType. The arrays are displayed unsorted, sorted (ascending), sorted (descending where displayed), and in the case of MyType, sorted by Age, and by Name.

```
Formatting in a table
| 0   Mary-Beth            |
| 1   Aunt Alma            |
| 2   Sue                  |
| 3   My Really Long Name  |
| 4   Matt                 |

Enum Formatting
Name: Red, Value: 0

DateTime Formatting:  Predefined formats

Code    Format
----    ------
d       2/4/2002
D       Monday, February 04, 2002
f       Monday, February 04, 2002 5:24 PM
F       Monday, February 04, 2002 5:24:07 PM
g       2/4/2002 5:24 PM
G       2/4/2002 5:24:07 PM
m       February 04
r       Mon, 04 Feb 2002 17:24:07 GMT
s       2002-02-04T17:24:07
t       5:24 PM
T       5:24:07 PM
u       2002-02-04 17:24:07Z
U       Tuesday, February 05, 2002 1:24:07 AM

DateTime Formatting:  Picture formats
Code    Format
----    ------
ddd     Mon
MMMM dd, yyyy   February 04, 2002

Numeric Formatting:  Predefined formats
Code    Format
----    ------
c       $12.96
d       103
e       1.296250e+001
f       12.96
g       12.9625
n       12.96
x       44e

Numeric Formatting:  Picture formats
Code    Format
----    ------
#       25
###     25
#.00    25.00
```

```
%#      %25
D4      0025

Formatting Custom Types
Code    Format
----    ------
b       101011
o       53
d       43
h       2b
System.Object[] in the custom B8 format is _ASP.stringformat_aspx+BaseFormatter
System.Object[] in the custom B16 format is _ASP.stringformat_aspx+BaseFormatter
```

The following examples show various way to convert datatypes from one form to another. These conversions were achieved with either the Parse or ToString methods, or through the methods provided inside the Convert class.

```
Converting an integer value to a string
---------------------------------------
The string value is 76
The string value is 76
The string value is 76

Using The Convert object's methods
------------------------------
Using Convert.ToString, result = 76
Using Convert.ToBoolean, result = True
Using Convert.ToByte, result = 76
Using Convert.ToChar, result =  L
Using Convert.ToDouble, result = 76

Using the non-instantiated Parse method
---------------------------------------
Parsing true = True
Parsing 00.987 = 0.987
Parsing 127 = 127
Parsing 42 = 42

Using Exception Handling when converting
----------------------------------------
The value 'three' cannot be converted to a integer value.

Attempting an illegal, implicit conversion
------------------------------------------
Adding the following code to a program would cause a compiler error in C#
String strExample = intExample
The error received will be: cannot implicitly convert type 'int' to 'string'

Converting Strings to Dates
---------------------------
Parsing 03/17/77, using the Parse method.
The date parsed as 3/17/1977 12:00:00 AM

Parsing Thursday, March 17, 1977, using the ParseExact method.
The date parsed as 3/17/1977 12:00:00 AM

Converting String information to other datatypes
------------------------------------------------
The Double value when using Double.Parse is 123456
The Int32 value when using Int32.Parse is 123456

Extending the use of the Parse method
-------------------------------------
Parsing the string "   (555)   " which contains leading/trailing white space and
parentheses
The string parsed to -555.
```

The following example demonstrates using the Is keyword, the = operator, and the Equals method to compare objects, and primitive (base) types to eachother. When using the Is keyword, objects will be compared for referential consistency, while using the Equals method, objects will be compared for value equality. Strings and other base datatypes should be compared for value equality using the = operator or the Equals method only, in which case the results will be the same.

```
Comparing Strings
-----------------
Does the String hello = the String goodbye? No
Does the String hello = the String hello (Different Reference)? Yes
Does the String hello = the String hello (Same Reference)? Yes

Does the String hello .Equals the String goodbye? No
Does the String hello .Equals the String hello (Different Reference)? Yes
Does the String hello .Equals the String hello (Same Reference)? Yes


Comparing base Boolean datatypes
--------------------------------
Does the Boolean False = the Boolean True? No
Does the Boolean False = the Boolean False (Different Reference)? Yes
Does the Boolean False = the Boolean False (Same Reference)? Yes

Does the Boolean False .Equals the Boolean True? No
Does the Boolean False .Equals the Boolean False (Different Reference)? Yes
Does the Boolean False .Equals the Boolean False (Same Reference)? Yes


Comparing CultureInfo object references
-----------------------------------
Is the CultureInfo en-NZ Equal to the CultureInfo en-US, using Is? No
Is the CultureInfo en-NZ Equal to the CultureInfo en-NZ, using Is (Different
Reference)? No
Is the CultureInfo en-NZ Equal to the CultureInfo en-NZ, using Is (Same Reference)?
Yes

Is the cultureInfo en-NZ Equal to the CultureInfo en-US, using .Equals? No
Is the cultureInfo en-NZ Equal to the CultureInfo en-NZ, using .Equals (Different
Reference)? Yes
Is the cultureInfo en-NZ Equal to the CultureInfo en-NZ, using .Equals (Same
Reference)? Yes


Object Reference Example
------------------------
I just made a Car. Its color is Red.
YourCar and MyCar refer to the same car. YourCar's color is therefore Red.
If you change the color of YourCar to Pink, MyCar's color also changes to Pink.

In contrast, Strings are different. MyName is Kayleen.
Initially, I can set YourName to MyName, so YourName is Kayleen.
But if I change YourName to Mary, MyName (Kayleen) remains the same.
```

The following example shows what happens when you implement the IComparable interface to compare objects to one another. You can compare base datatypes, or define your own objects for comparison, so long as they implement IComparable. In this example, we define a Max method to compare any two objects that implement IComparable.

```
We have defined a Max method, that works on any types which implement the IComparable
interface....

Comparing Integers:     Max (5, 10) =                   10

Comparing Doubles:      Max (4.3, 2.5) =                4.3

Comparing Decimals:     Max (1235698, -234238) =        1235698

Comparing Strings:      Max (Mathew, Mark) =            Mathew

Comparing Chars:        Max (t, a) =                    t

Comparing Custom Types: Max (MyType(12), MyType(17)) =  MyType(17)
```

The following example illustrates using methods inside the Math class. To learn more about these, and other methods in the math class, please see the SDK documentation. The methods are intuitive and easy to use, and provide you with many basic mathematical functions.

```
Constants
-----------------------------------
The constant value e is 2.71828182845905
The constant value PI is 3.14159265358979

Using the Abs Method
-----------------------------------
The absolute value of -10 is 10
The absolute value of 39.25 is 39.25
The absolute value of -27.84 is 27.84

Using the Floor Method
-----------------------------------
Using the Floor method to round down -10 you get -10
Using the Floor method to round down 39.25 you get 39
Using the Floor method to round down -27.84 you get -28

Using the Ceiling Method
-----------------------------------
Using the Ceiling method to round down -10 you get -10
Using the Ceiling method to round down 39.25 you get 40
Using the Ceiling method to round down -27.84 you get -27

Using the Max Method
-----------------------------------
The maximum value of -10 and 39.25 is 39.25
The maximum value of -27.84 and -10 is -10
The maximum value of 39.25 and -27.84 is 39.25

Using the Sqrt Method
-----------------------------------
The square root of 39.25 is 6.26498204307083
The square root of -10 is NaN

Using the Pow Method
-----------------------------------
-10 raised to the power 2 is 100
39.25 raised to the power 0.3 is 3.00712788821071
-27.84 raised to the power -4 is 1.66465054096883E-06
```

This sample demonstrates the methods available in the Math class. You can invoke any of the non-instantiated methods on the Math class. However, this particular example always calls the overloaded constructor, which accepts a double or double values. The sample does this so that the exact signature of the method you invoke does not need to be determined.

You will not be able to determine from the code how the exact method you chose gets invoked, because this sample uses the Reflection class to find all the methods of the Math class at runtime and invoke them dynamically. Please refer to the reflection quickstart sample on invoking methods for more details. This approach results in less hard-coded code, but results in a more involved example. If you want to see some example lines of code that you would need to type to invoke a particular method, please see the discussion page associated with this example.

The purpose of each method should be fairly self-explanatory, but you should consult the SDK documentation when you are in doubt. Remember to give the sample a good test, and try out negative and positive values of both whole and floating numbers.

Please select the Math method you wish to invoke: Select A Method

Run Query

Collections

Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I....Create and Use a Hashtable?

This example illustrates how to create and use a hash table. A hash table is a collection that consists of key-value combinations, organized into 'buckets' for fast searching. You can search a hash table either by the keys, or by the associated values; however, if fast searching is your objective, searching by key will generally be faster.

VB HashTable.aspx

[Run Sample] | [View Source]

As mentioned in the introduction, a hash table is a type of collection. When you populate a hash table, you give it a key to add to the table, and a value that goes with the key. In the following example, employee id numbers are used for the key, and then the employee names, to give you an idea of the kind of combinations you can use. Note that the key does not need to be numeric: any object is acceptable.

Why is a hash table useful? When one of these key-value pairs is added to the hash table, the hash table looks at the key and gives it a hash code, a number that identifies the key (each key you add gets a hash code). These hash codes are then placed in 'buckets' to help organize the entries in the table. This can help later when you try to find something in the table.

Imagine for a second that these buckets aren't used. If you wanted to find something in an array, you would have to go through every element of the array, looking for the element. You have no idea where it is, so there is no fast way to search through the entries. By contrast, a hash table can search for a particular key faster, because once you give it a key it knows which bucket to look in. This means it only has to look through a subset of all the elements.

It is important to remember that this faster searching mechanism is only available when searching by keys. Searching by a value is available, but is potentially slower than key searching. Therefore, where possible you should perform key searches.

To add elements to a hash table, call the **Add** method, passing in the key-value pairs you want to add.

```
'  Here we'll create a hashtable of employee numbers and employee names
Dim table As Hashtable = New Hashtable()

'  now, we will add elements to the hashtable, as key-value pairs
table.Add(5123, "Jay")
table.Add(1829, "Tom")
table.Add(2882, "Matt")
```

VB

Now that the hash table is populated, you can search for elements in it. You are going to search both by key and value. Note that in the actual example, you will accept a value or key to find from the user, so the implementation is slightly different, but the concepts are the same.

```
'  SEARCHING BY VALUE. Use the method ContainsValue
Dim valueToFind as String = "Jay"
```

```
If table.ContainsValue(valueToFind) Then
        Console.WriteLine("Found {0} in the list.", valueToFind )
End If

'  SEARCHING BY KEY. Use the method Contains
Dim keyToFind As Integer = 5123

If table.Contains(keyToFind) Then
        Console.WriteLine("Found {0} in the list.", keyToFind)
End If
```

VB

Of course, you can do more than simply searching. You can remove entries from the hash table by using the **Remove** method, which you pass a key identifying the element you want to remove. You can also enumerate over the hash table using the Keys collection or the Values collection inside your instance of the hash table.

```
'  using the Remove method to delete a particular item...
table.Remove(5123);

Dim o As Object

'  enumerating over the collection with a foreach statement
'  Note the alternate of iterating through by value is demonstrated
'  For Each o In table.Values
For Each o In table.Keys
        Console.WriteLine(o.ToString())
Next
```

VB

Those are the basics of using a hash table. It is important to know how to iterate through your hash table, once you have it populated with entries. This important topic is covered in the section How Do I...Iterate over a collection? The following code shows you how to use a foreach (For Each in VB) statement to move through our list, and print each of the key-value combinations to the screen. To do this, you also need to reference the **Key** and **Value** properties of each element. The following sample demonstrates iterating through the collection.

```
'  if you are wondering where DictionaryEntry comes from, it is
'  the generic object type of each element in the hashtable
Dim d As DictionaryEntry

For Each d in table
        '  use the Key and Value properties. Note chr(9) is a tab
        Console.WriteLine ("{0}" & chr(9) & "{1}", d.Key, d.Value)
Next d
```

VB

The following example demonstrates searching for elements in a hashtable. You can search either by key, or by value, however searching by key is faster. Often, the functionality that you want to provide your users may mean you have to search by value. In addition, the ability to add elements from the hashtable is demonstrated.

Find an existing Entry

Enter a value (employee name) to find: [ ] Find Value

Enter a key (employee id) to find: [ ] Find Key

Hashtable Entries

- Lakshan
- Jay
- Brad
- Rajesh
- Brian
- Tom
- Stephen
- Seth

Reset List

New Entry

Key: [ ]

Value: [ ]

Add Entry

Collections

Iterate over a collection
Use a Hashtable collection
Choose a collection to use
Implement a collection
Clone a collection

Back to Index

Get URL for this page

How Do I....Choose a collection to use?

This example discusses the storage procedures and capabilities of three different collections: Queue, SortedList, and Hashtable. The primary purpose of this demonstration is to help developers understand the functions of the different collection types, so that they can choose one that will meet their needs.

VB Collections.aspx

[Run Sample] | [View Source]

At first, you may find it difficult to determine which collection best suits your needs. With experience, you will become familiar with each of the different collections you can use, and even customize your own by extending ICollection or IList. The three collections discussed here are: Queue, SortList, and Hashtable. For more detailed information on Hashtable, see How Do I...Use a hashtable?.

A Queue is an unsorted collection, which is meant to be used on a 'first-in, first-out' basis.

- Do use a Queue when you need entries to be stored in the same order they were entered
- Do use a Queue when you want to be able to easily add elements to the end of the Queue. The **Enqueue** method provides this functionality
- Do use a Queue if you want to easily be able to read and remove the top item from the Queue. This is the capability provided by the **Dequeue** method. If you don't want to remove the element but still what to see what it is, use the **Peek** method
- Don't use a Queue if you need fast searching capabilities. Queue searching is not fast
- Don't use a Queue if you need to remove a particular element specified by you or the user
- Don't use a Queue if you need to be able to reorder or sort the elements.

A SortedList is exactly that: a collection of sorted items. A SortedList consists of key-value pairs, and the sort is organized by the key, not the values.

- Do use a SortedList if you want to sort your items, without having to do much work
- Do use a SortedList if you want to control how the elements are sorted (how they are compared)
- Do use a SortedList if you can use fast searching capabilities of the keys of the SortedList
- Do use a SortedList if you want to be able to remove specific elements (use the **Remove** or **RemoveAt** methods)
- Don't use a SortedList if you need to insert an element in a particular location. It will just be sorted anyway
- Don't use a SortedList if you want to store multiple keys with the same value. All keys inside a SortedList must be unique
- Consider using a different collection (such as Hashtable) if fast searching (particularly by Value) is your highest priority.

A hash table is a set of elements also requiring key-value pairs. It sorts the key-value pairs using an internal algorithm, which is not known to the user. The key-value pairs are stored in 'buckets' which makes searching for values very fast.

- Use a hash table if searching is a priority. Hash tables provide very quick search mechanisms when searching by key, and fairly good searching when searching by

value
- Use a hash table if you want to be able to remove specific elements (use the **Remove** method)
- Use a hash table when the order the elements is stored is irrelevant to you
- Don't use a hash table if you need the elements in some specific order. You cannot rely on how a Hashtable will sort its elements
- Don't use a hash table if you need to insert an element in a particular location
- Don't use a hash table if you want to store multiple keys with the same value. All keys inside a hash table must be unique

The next step is to look at declaring each collection, and populating each with the same set of information. Remember that the Hashtable and the SortedList both need key-value combinations, but in this example, you are going to use the same entries for both the key and the value. That is, you will make collections of last names, and use the same last name for the key, and the value of a particular pair (for example, one pair will be "Smith"-"Smith").

```
// don't forget to include those 'Imports' statements
Imports System
Imports System.Collections

'  ... class and method declarations...

'  the declaration for an array of information
Dim names() As String = New String() {"Simpson","Bristow","Walls", _
                "Gilligan","Davidson","Laurence", _
                "Jackson","Singh","O'Connor","Moynagh"}

'  the declarations for each of the three collections
Dim q As Queue q = new Queue()
Dim sl As SortedList = New SortedList()
Dim ht As Hashtable = New Hashtable()

Dim s As String

'  we use a foreach statement to populate our collections
For Each s In names

        '  to add an element to the queue, use Enqueue
        '  A queue only receives the value you want to add
        q.Enqueue(s)

        '  Use Add of SortedList/Hashtable to add elements
        '  Note that you must add key-value pairs
        sl.Add(s, s)
        ht.Add(s, s)
Next s
```

VB

You can also remove items from collections, although different methods must be used depending on the collection in question. Many collections support the **Remove** method to remove a specified element. Both Hashtable and SortedList support this method, to remove the element you specify. **RemoveAt** is also supported by many collections; however, out of the three collections mentioned in this discussion, only SortedList supports the **RemoveAt** method. **RemoveAt** requires you to supply an index of the element to remove, which doesn't really make sense in the case of a Queue or Hashtable (which is why they don't support it). A Queue is a special case since removing elements in a Queue can be performed only on the top element off the Queue. Use the **Dequeue** method to achieve this, as in the following sample.

```
' to remove an element with a specific key, use the Remove method
' where the concept of a key is invalid, specify the object to remove
sl.Remove(keyValue); '  removes the arbitrary keyvalue
ht.Remove(keyValue); '  removes the arbitrary keyvalue
```

```
' to remove an element at a specific index, use the RemoveAt method
' This does not make sense for collections such as a Hashtable/Queue
sl.RemoveAt(indexValue); ' removes the specified index

' A Queue does not have the concept of an arbitrary remove
' Use Dequeue to read, AND remove the top element off the Queue
q.Dequeue(); ' remove the top value
```

VB

After you create the collection, you can look at the information inside those collections by using a simple iteration statement to print the values out. It is difficult to show here what the results will be, so it requires you to test the output. Note that the Queue will not change the order of the items, the SortedList will sort them alphabetically, and the Hashtable will change the order, but not necessarily to anything that you or I could understand: it uses its own sorting mechanism.

```
Dim s As String

Console.WriteLine("The Entries In The Queue")
For Each s In q
        Console.WriteLine(s)
Next s

'  note that we have to parse the Values of the SortedList, rather than
'  the actual SortedList itself
Console.WriteLine("The Entries In The SortedList")
For Each s In sl.Values
        Console.WriteLine(s)
Next s

'  note that we have to parse the Values of the Hashtable, rather than
'  the actual Hashtable itself
Console.WriteLine("The Entries In The SortedList")
For Each s In ht.Values
        Console.WriteLine(s)
Next s
```

VB

So far, you have looked at the code that demonstrates how each of the collections is organized, and how to retrieve their values. However, you also need to consider how to search for values inside each of the different collections. The following code example shows you the commands needed to find a value inside each collection. The methods to search by key and by value are displayed for both SortedList and Hashtable. Recall that hash tables are very good for searching, and sorted lists are fairly good, especially when searching by key. Queues also provide a searching capability, but this is not really a Queue's purpose, and the search mechanism is slower.

```
'  use these lines of code to retrieve the time before the search...
Dim startTime as Int64
startTime = Environment.TickCount

'  use these lines of code to retrieve the time, after the search.
'  You can subtract this value from the above, to get a duration.
Dim endTime as Int64
endTime = Environment.TickCount

'  use Contains to search a Queue for an entry,
'  or a Hashtable/SortedList for a KEY
'  Contains returns a boolean, indicating if value/key was found
Dim foundValue As Boolean
Dim valToFind as Int64
foundValue = q.Contains(valToFind)
```

```
foundValue = sl.Contains(valToFind)
foundValue = ht.Contains(valToFind)

'  use ContainsValue to search a SortedList/Hashtable for a VALUE
foundValue = sl.ContainsValue(valToFind)
foundValue = ht.ContainsValue(valToFind)
```

VB

Summary

Within the **Collections** namespace, there are many collections that providing different functionality. It is a good idea to familiarize yourself with the different collections available, so that you know which collection to use in each situation you encounter. For information on more specialized collections, see the **Collections.Specialized** namespace.

The following example demonstrates using three different types of collections: a Queue, a SortedList, and a Hashtable. A Queue stores elements in the same order they are placed in it, while a SortedList stores them in the fashion specified by the comparer for the SortedList (we use the default comparer, which stores strings in alphabetical order). You can control the way in which the SortedList sorts its elements. Finally, a Hashtable stores elements in a sequence of its own devising, meaning it will not really make sense to us, but will probably not be the same way as we placed the elements in the Hashtable. Searching for items in SortedLists and Hastables is generally faster than when searching in a Queue, especially when searching by key, rather than by value.

Each of these collections has its own particular uses, and capabilities.

The order of the entries in the Array: Simpson, Bristow, Walls, Gilligan, Davidson, Laurence,
Jackson, Singh, O'Connor, Barclay

The Entries In The Queue	The Entries In The Sorted List	The Entries In The Hashtable
Simpson	Barclay	O'Connor
Bristow	Bristow	Bristow
Walls	Davidson	Gilligan
Gilligan	Gilligan	Simpson
Davidson	Jackson	Jackson
Laurence	Laurence	Davidson
Jackson	O'Connor	Singh
Singh	Simpson	Laurence
O'Connor	Singh	Walls
Barclay	Walls	Barclay

Comparing Seek Times

Total Seek time for Queue:	125
Total Seek time for SortedList, by Key:	0
Total Seek time for Hashtable, by Key:	0
Total Seek time for SortedList, by Value:	62
Total Seek time for Hashtable, by Value:	16

The following example demonstrates implementing the ICollection interface. In this example we make a fictitious EmployeeCollection, for a company which likes employing people, but doesn't like them leaving. Therefore, we allow elements to be added and modified, but you can't actually remove any elements from the collection unless you clear the entire collection. If you have a look at the code, you will see that we can use a For Each loop to parse the information in the collection, a reflection of the fact we have included an IEnumerator for the collection.

```
Adding elements to the EmployeeCollection
----------------------------------------------------------------
1234, John
5678, Lisa
3283, Max
1928, Regina
4019, Clarence

Attempting to add a duplicate employee id...
----------------------------------------------------------------
Sorry, the employee id 5678 already existed in the collection.

Setting the name of existing employee with id 5678...
----------------------------------------------------------------
1234, John
5678, Katherine
3283, Max
1928, Regina
4019, Clarence

Retrieving the employee with id 4019...
----------------------------------------------------------------
4019, Clarence

Clearing the collection...displaying the elements (no elements left...)
----------------------------------------------------------------
```

The following example demonstrates cloning collections. In this example, the first three entries in the collection are object references to Employee objects, while the third and fourth entries are literal strings. If you follow through the example, you can see that the cloned ArrayList refers to the same objects as the original ArrayList, but the Strings are unique to each list since they are not reference-copies but are instead, value copies. In contrast, when we copy the ArrayList, both of the ArrayLists are completely unique such that modifications made to one will not affect the other.

```
Performing a CLONE of the ArrayList.
Modifications in the copy will also affect the original.
-------------------------------------------------------
The entries in the original ArrayList
-------------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2

The entries in the copied ArrayList
-------------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
```

```
Performing a COPY of the array.
Modifications in the copy will not affect the original.
-------------------------------------------------------
The entries in the original ArrayList
-------------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2

The entries in the copied ArrayList
-------------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
```

```
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2

The entries in the modified, copied ArrayList
-----------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
Modified String Entry 1
Modified String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2

The entries in the original ArrayList
-------------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
```

```
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2

The entries in the modified, copied ArrayList
-----------------------------------------
7182, Mary van der Beer
2637, Alexei Vyernov
4921, Chris Hayfield
Repeated Modify of String Entry 1
Repeated Modify of String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2

The entries in the original ArrayList
-------------------------------------------
7182, Mary Smith
1827, Francoi Michaels
9912, Francesca Davies
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
7182, Mary Greerson
2183, Abdul Hania
7162, Brenda Simons
String Entry 1
String Entry 2
```

Component Model
Use the CodeDOM

Back to Index

Get URL for this page

How Do I...Use the CodeDOM to emit source code?

What is the CodeDOM

The CodeDOM is an API that gives you the ability to create a programming structure of namespaces, objects, and programming constructs by adding items to different collections. All CodeDOM programs start out with the creation of a **ICodeGenerator** for the target language. The following code shows the creation of the **ICodeGenerator** for C#.

```
Dim cdp As CSharpCodeProvider = New CSharpCodeProvider()
Dim cg As ICodeGenerator = cdp.CreateGenerator()
```

VB

The **ICodeGenerator** is used to generate the actual code once a CodeDOM tree is built. To begin building your tree, you need to start with a **CodeNamespace** object. This object will contain any types and import directives. With the **CodeNamespace** object you can begin the tree. For the following sample, several **CodeNamespaceImport** objects are used to generate import directives for commonly used namespaces. The sample also needs to generate a class construct so by using a **CodeTypeDeclaration** and setting the **IsClass** property to true. The following code demonstrates these concepts.

```
Dim cnamespace As CodeNamespace = New CodeNamespace("Microsoft.Com2Sdk.TypedCollectio
 ...
cnamespace.Imports.Add (New CodeNamespaceImport ("System") )
 ...
Dim co As CodeTypeDeclaration = New CodeTypeDeclaration (typeName +"List")
co.IsClass = true;
cnamespace.Types.Add (co)
```

VB

At the end of the previous example, the class type definition is added to the **Types** collection of the **CodeNamespace** object. This is how a CodeDOM tree is built. Starting with a **CodeNamespace** object and working down through classes to class members. Class members in turn contain different types of code statements and expressions. Once the entire tree is built, the entire source tree can be written with a call to **GenerateCodeFromNamespace**, a method on **ICodeGenerator**. The method is being passed the **CodeNamespace** object and a **TextWriter**.

```
baseCompiler.GenerateCodeFromNamespace (cnamespace, w, Nothing)
```

VB

The following sample demonstrates the techniques described above by generating the code for a strongly typed List. It also contains additional information and code for generating code statements and expressions that were left out of the above documentation for brevity.

Example

VB ListBuilder.exe

[Run Sample] | [View Source]

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Update a Database from a DataSet

This topic illustrates how to update data in a database using a DataSet. It is important to remember that you can also insert, update, and delete data in a database directly using a **SqlCommand**. Understanding the concepts covered in Populate a DataSet from a Database will help you understand the current topic.

Some of the topics covered in Populate a DataSet from a Database include retrieving data from a database and into a DataSet, and how the DataSet is separate and distinct from the database. Once the **DataSet** is loaded, you can modify the data, and the DataSet will track the changes.

The **DataSet** can be considered an in-memory cache of data retrieved from a database. The **DataSet** consists of a collection of tables, relationships, and constraints. In this example we will show how to use the **Add** method on the DataTable to add new data to a DataSet. The **Add** method takes either an array of the expected data columns, or a DataRow.

```
' Create a new Connection and SqlDataAdapter
Dim myConnection as SqlConnection
Dim mySqlDataAdapter as SqlDataAdapter

' Warning: Embedded usernames/passwords in database connection strings are inherently
' insecure, and should only be used on platforms (for example, Win9x) that do not support Integrated
' Security.  See the ASP.NET Quickstart for examples of Integrated Security.
myConnection = new SqlConnection("server=(local)\NetSDK;uid=QSUser;pwd=QSPassword;database=northwind")
mySqlDataAdapter = new SqlDataAdapter("Select * from Customers", myConnection)

Dim myDataSet as DataSet = new DataSet()
Dim myDataRow As DataRow

' Create command builder. This line automatically generates the update commands for you, so you don't
' have to provide or create your own.
Dim myDataRowsCommandBuilder as SqlCommandBuilder = new SqlCommandBuilder(mySqlDataAdapter)

' Set the MissingSchemaAction property to AddWithKey because Fill will not cause primary
' key & unique key information to be retrieved unless AddWithKey is specified.
mySqlDataAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey

mySqlDataAdapter.Fill(myDataSet, "Customers")

myDataRow = myDataSet.Tables("Customers").NewRow()
myDataRow("CustomerId") = "NewID"
myDataRow("ContactName") = "New Name"
myDataRow("CompanyName") = "New Company Name"

myDataSet.Tables("Customers").Rows.Add(myDataRow)
```

VB

Note that the DataTable must return a DataRow through the **NewRow** method. The method returns a DataRow object with the appropriate schema of the DataTable. The new DataRow is independent of the table until it is added to the RowsCollection.

You can change data in a DataRow by accessing the DataRow. You can use the index of the row in the RowsCollection accessed through the **Rows** property:

```
myDataSet.Tables("Customers").Rows(0)("ContactName")="Peach"
```

VB

You can also access a specific row by the Primary Key value:

```
Dim myDataRow1 As DataRow = myDataSet.Tables("Customers").Rows.Find("ALFKI")
myDataRow1("ContactName")="Peach"
```

VB

where "ALFKI" is the value of the Primary Key "CustomerID" in the "Customers" table. When using the SqlDataAdapter, the Key is established from the database. You can also set the Key if you are not using the database through the **PrimaryKey** property.

Use the **Delete** method to remove the Row. Note that a logical deletion occurs in the DataSet, which only results in a hard deletion once the DataSet is updated to the database. Similarly you can use **RejectChanges** on the DataSet, in which case the Row is restored.

```
myDataSet.Tables("Customers").Rows(0).Delete()
```

VB

The original and new values are maintained in the row. The **RowChanging** event allows you to access both original and new values to decide whether you want the edit to proceed. Because we maintain original and new values, we can establish scenarios such as optimistic locking and key changes.

Before submitting the update back to the database, you need to setup the **InsertCommand**, **UpdateCommand**, and **DeleteCommand** to reconcile the changes to the database. For limited scenarios you can use the **SqlCommandBuilder** to automatically generate those for you, as is shown in the following example:

```
Dim myDataRowsCommandBuilder as SqlCommandBuilder = new SqlCommandBuilder(mySqlDataAdapter)
```

VB

To submit the data from the DataSet into the database, use the **Update** method on the SqlDataAdapter.

```
mySqlDataAdapter.Update(myDataSet, "Customers")
```

VB

The following example demonstrates how to get data from a database using a SqlDataAdapter, modify the data within the DataSet, and then submit the data back to the database through the SqlDataAdapter.

VB UpdatingData.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Get Out Parameters from a Stored Procedure

Some stored procedures return values through parameters. When a parameter in a SQL statement or stored procedure is declared as "out", the value of the parameter is returned back to the caller; the value is stored in a parameter in the Parameters collection on the **OleDbCommand** or **SqlCommand** objects.

Unlike the sample below, if the connection and command name are not set, you can still establish the parameters, but you have to create the collection of parameters and expected types.

```
workParam = myCommand.Parameters.Add("@CustomerID", SQLDataType.NChar, 5)
workParam.Value = "CUSTID"
```

VB

The following sample shows how to use a stored procedure that returns output parameters. The command is executed to create a stored procedure in the Northwind database.

VB OutParamsWithACommand.aspx

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Both Records are written to the database!

CompanyName = Alfreds Futterkiste

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Read and Write XML

ADO.NET and DataSet can read and write XML schema and XML. For more information, see How do I...Use XML and the DataSet?

Schemas, or the tables, columns, constraints and so forth, of a DataSet can be defined in several ways. One method is to create them using properties and methods (**PrimaryKey**, **Table.Add**, **Column.Add**, and so on). This establishes a schema within your DataSet that can be used as a container to hold data. Another way is to use the SqlDataAdapter or OleDbDataAdapter. When you use these commands, if the schema does not exist in a DataSet, it is created for you.

XML is an intrinsic element of ADO.NET. Several methods that support XML have been implemented. The **GetXmlSchema** method allows you to get the schema of a DataSet using an XSD schema. For example, if you load a DataSet from the database, you could get an XSD schema from it:

```
Console.WriteLine(myDataSet.GetXmlSchema())
```

VB

This returns an XSD compliant XML schema for the schema in your DataSet.

XML representation is also retrieved through the **GetXml** method.

```
Console.WriteLine(myDataSet.GetXml())
```

VB

The sample below loads data from a database, and then outputs the XSD Schema and XML data.

VB ReadAndWriteXML.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

Data and ADO.NET
Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Work with Relational Data

A DataSet can contain either related or unrelated data, and can have either two unrelated tables or two related tables. You can think of a DataSet as a document of data. In fact, an XML data document is like this, except it is based on a hierarchical paradigm. Because data is often stored in relational databases, the DataSet can handle both hierarchical relationships and key/foreign key relationships. Relationships can also have different types of enforcement. By default, deletions and updates are cascaded: if you delete a Customer row, the related Orders rows are also deleted; if you Update the key of a Customer row, the associated foreign key values in the Orders table is also updated.

A DataSet contains a Relations collection. You can add a relationship to this collection using the column or columns (in a multicolumn key) of the related tables. The following code example creates a relation between **Customers** data and **Orders** data, and names the relation CustOrders.

```
myDataSet.Relations.Add("CustOrders",myDataSet.Tables("Customers").Columns("CustomerID"), _
                        myDataSet.Tables("Orders").Columns("CustomerID"))
```

VB

The following code example adds a relation between the CustomerID Key on the Customers table and the CustomerID foreign key on the Orders table in the DataSet. Now that a relationship exists, you can iterate through the data.

```
Dim myDataRow1 as DataRow
for each myDataRow1 in myDataSet.Tables("Customers").Rows
  Console.WriteLine("Customer: " + myDataRow1("ContactName").ToString())

  ' Iterate over orders data
  Dim myDataRow2 as DataRow
  for each myDataRow2 in myDataRow1.GetChildRows(myDataSet.Relations("CustOrders"))
    Console.WriteLine("Order #" + myDataRow2("OrderID").ToString())
  next

  Console.WriteLine()
next
```

VB

The following sample iterates over the data and outputs it to a Web page hierarchically.

VB RelationalData.aspx

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

ABC
ALFKI
ANATR
ANTON
AROUT
BERGS
BLAUS
BLONP
BOLID
BONAP
BOTTM
BSBEV
CACTU
CENTC
CHOPS
COMMI
CONSH
DRACD
DUMON
EASTC
ERNSH
FAMIA
FISSA
FOLIG
FOLKO
FRANK
FRANR
FRANS
FURIB
GALED
GODOS
GOURL
GREAL
GROSR
HANAR
HILAA
HUNGC
HUNGO
ISLAT
KOENE
LACOR
LAMAI
LAUGB
LAZYK
LEHMS
LETSS

LILAS
LINOD
LONEP
MAGAA
MAISD
MEREP
MORGK
NewID
NORTS
OCEAN
OLDWO
OTTIK
PARIS
PERIC
PICCO
PRINI
QUEDE
QUEEN
QUICK
RANCH
RATTC
REGGC
RICAR
RICSU
ROMEY
SANTG
SAVEA
SEVES
SIMOB
SPECD
SPLIR
SUPRD
THEBI
THECR
TOMSP
TORTU
TRADH
TRAIH
VAFFE
VICTE
VINET
WANDK
WARTH
WELLI
WHITC
WILMK

WOLZA

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Populate a DataSet from a Database

Getting data from a database is easy, and working with data is easier than before. If you want the results from a database as a forward only, read-only stream of data, you can execute a command and retrieve the results using the **DataReader**. For examples on how to use a **DataReader**, see Retrieve Data Using OLE DB and Retrieve Data from SQL Server. For more interactive operations such as binding to, navigating through, or remoting the results of a database query, you can place the results in a DataSet as shown in this example.

The most important concept to remember is that the DataSet is a data structure separate and distinct from a data store. Although you get data from a database in this example, it doesn't matter where the data comes from; the DataSet will always present a consistent programming model. It is a simple collection of data with relational database characteristics. There is no Load, Open, or Execute on a DataSet because it doesn't know where it gets its data from. This section describes how to use a SqlDataAdapter to get data from a database into a DataSet.

You can add data to the DataSet using the **Add** method (see Update a Database from a DataSet), or directly push data in with the intrinsic XML methods (see Read and Write XML Data). SqlDataAdapter can be used to retrieve data from a database but can also push data back into the database. Here, the focus is on getting data out.

The first step is to construct a SqlDataAdapter. This is done in the same way a SqlCommand is constructed.

```
Dim SelectCmdString As String = "select * from customers"
Dim mySqlDataAdapter As SqlDataAdapter = New SqlDataAdapter(SelectCmdString, myConnection)
'...or this can also be done as follows:

Dim mySelectCommand as SqlCommand = New SqlCommand("select * from customers", myConnection)
Dim mySqlDataAdapter as SqlDataAdapter = new SqlDataAdapter(mySelectCommand)
```

VB

The SqlDataAdapter and SqlCommand are very similar, except for the **Fill** and **Update** methods. The **Fill** method populates a DataSet. The **Update** method takes changes from a DataSet and pushes them back into the database. This is accomplished by four commands specified on the DataAdapter. These commands are: SelectCommand, UpdateCommand, InsertCommand, and DeleteCommand. You can explicitly set these commands to control the statements used at run time to resolve changes, including the use of stored procedures. For ad hoc scenarios, a **CommandBuilder** object can generate these at run time based on a select statement (see Update a Database from a DataSet). However, this run-time generation requires an extra round trip to the server to gather required metadata, so explicitly providing the insert, update, and delete commands at design time will result in better run-time performance.

NOTE: Visual Studio adds great value to establishing typed SqlDataAdapters and DataSets, and eventually creates Stored Procedures for you. Explore this feature by using the **ComponentDesigner** and **Database** objects.

Once your SqlDataAdapter is established you can pass it a DataSet to populate:

```
myDataSet = New DataSet()
mySqlDataAdapter.Fill(myDataSet,"Customers")
```

VB

The DataSet now holds the results of your query. In fact, the DataSet can hold results from multiple queries and even relate them (see Work with Relational Data). Because it holds multiple results, the DataSet contains a collection of tables. Notice that the **Fill** method has "Customers" as the second argument. This is the name of the table to fill in the DataSet. If that table does not exist, it is created for you.

Because the data is stored in a collection of rows in the table, you can easily use a **foreach** statement to iterate through the rows:

```
Dim myDataRow as DataRow
For Each myDataRow In myDataSet.Tables("Customers").Rows
  Console.WriteLine(myDataRow("CustomerId").ToString())
Next
```

VB

In fact, you can use **foreach** over the columns as well. The following example demonstrates placing together all the code in this document. To learn how to update this data back into the database see Update a Database from a DataSet.

VB GettingData.aspx

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Copyright 2001 Microsoft Corporation. All rights reserved.

Loaded data from Customers table into dataset.
Loaded data from Orders table into dataset.
Inserted new row into Customers.
Sent Update to database.
DataSet processing has completed successfully!

Data and ADO.NET

Overview of ADO.NET
Execute a Command
Get Out Parameters from a Stored Procedure
Populate a DataSet from a Database
Update a Database from a DataSet
Read and Write XML
Use Database Transactions
Handle Errors
Work with Relational Data
Work with Typed Data
Filter Data
Use Connection Pooling
Use Distributed Transactions
Generate XML with data from SQL Server
Load a native ADO recordset into a DataSet
Retrieve Data Using OLE DB
Retrieve Data from SQL Server
Convert an ADO Application to ADO.NET

Back to Index

Get URL for this page

ADO.NET: Read and Write XML

ADO.NET and DataSet can read and write XML schema and XML. For more information, see How do I...Use XML and the DataSet?

Schemas, or the tables, columns, constraints and so forth, of a DataSet can be defined in several ways. One method is to create them using properties and methods (**PrimaryKey**, **Table.Add**, **Column.Add**, and so on). This establishes a schema within your DataSet that can be used as a container to hold data. Another way is to use the SqlDataAdapter or OleDbDataAdapter. When you use these commands, if the schema does not exist in a DataSet, it is created for you.

XML is an intrinsic element of ADO.NET. Several methods that support XML have been implemented. The **GetXmlSchema** method allows you to get the schema of a DataSet using an XSD schema. For example, if you load a DataSet from the database, you could get an XSD schema from it:

```
Console.WriteLine(myDataSet.GetXmlSchema())
```

VB

This returns an XSD compliant XML schema for the schema in your DataSet.

XML representation is also retrieved through the **GetXml** method.

```
Console.WriteLine(myDataSet.GetXml())
```

VB

The sample below loads data from a database, and then outputs the XSD Schema and XML data.

VB ReadAndWriteXML.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

Copyright 2001 Microsoft Corporation. All rights reserved.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Use XML and the DataSet?

These topics provide an introduction to the interaction between the DataSet and XMLDataDocument objects.

XML and Relational Data. Same Data, Two Views.

XML often has a relational structure (the fictional books in a bookstore for example), as well as being unstructured. This set of topics cover the concept of storing this data centrally and providing different views on this data, either as XML or relationally as tables, columns and rows with relationships. This disconnected store of data which, for example, could represent a business object in the middle tier that enforces business rules, could provide its data as XML to a browser via Extensible Stylesheet Language Transformations (XSLT), across the Internet to another Web site, or to a local application via the relational tables. However, the data gets amended, you can update that data to a database on a transaction basis.

A DataSet represents an in-memory cache of data as a collection of tables and relationships between those tables. It is, in effect, a locally cached database. This provides a disconnected cache of data, like a message, that enables dealing with chunks of data. The DataSet has no knowledge of where the data came from. It may have come from a file, a database connection, or from a stream. A DataSet provides a relational view onto this stored data.

The XmlDataDocument provides XML APIs for accessing this in-memory cache of data, as well as supporting reading and writing XML. The XmlDataDocument is a DataSet-aware object. Creation of an XmlDataDocument implicitly creates a DataSet (accessed as a property) that provides a relational view onto the data XML data. This symbiotic relationship between these two objects provides a powerful technique for accessing data either relationally or as XML, irrespective of the mechanism by which the data was sourced.

XmlDataDocument and DataSet Differences

There are differences between these two objects and their views. XML loaded via the Load method on the XmlDataDocument maintains full fidelity-that is, all the XML structure is maintained and can be accessed via the XML API. The ReadXml method on the DataSet, however, only loads the XML that conforms to the internally created mappings (schema). You lose white space and element ordering, as well as data that does not conform to the relational mapping. Hence, the DataSet does not retain XML fidelity. Also data which has been loaded through an XmlDataDocument and which does not conform to the internal mappings on the DataSet cannot be viewed using the relational API. Whereas, relational data can always be viewed via the XML API.

Both the DataSet and XmlDataDocument can save data. However, they do not save the data identically. Using the WriteXml method, the DataSet is able to save a normalized view of the relationally mapped data (that is, the XML data that

conforms to the internal schema, with elements ordered according to the relational schema). This contrasts to the Save method of the XmlDataDocument that saves the XML data with full fidelity -all the XML that has been loaded into the XmlDataDocument, including the preservation of white space and element ordering.

Topics

The following topics work with XML data, DataSets and the XmlDataDocument:

1. How do I...Create DataSet mappings from an XSD schema? covers how a DataSet can build a relational view according to a supplied XML Schema Definition language (XSD) schema
2. How do I...Load a DataSet with XML? covers how XML data gets loaded into the DataSet and accessed as relational data.
3. How do I...Infer DataSet mappings from XML? covers how to infer a relational mapping from XML data, rather than an XSD schema.
4. How do I...Save DataSet mappings to an XSD schema file? shows how to save an inferred schema generated from loaded XML data.
5. How do I...Save a DataSet as XML? covers how to access and save XML that has been loaded into the DataSet relationally.

Related Topics

The following topics cover many of the features of the XmlDataDocument object that utilize the XML API. Refer to these topics for information about those scenarios in which you want to access data as XML.

1. How do I...Load and use the XmlDataDocument?
2. How do I...Read XML from an XmlDataDocument?

For coverage of the interaction between the DataSet and databases using the .NET data providers, see How do I...Get an Overview of ADO.NET?

Select View Source to see XMLSchema

```
<xsd:schema id="NewDataSet" targetNamespace="" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xsd:element name="NewDataSet" msdata:IsDataSet="true">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="Customers">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="CustomerID" type="xsd:string" minOccurs="0" />
              <xsd:element name="CompanyName" type="xsd:string" minOccurs="0" />
              <xsd:element name="ContactName" type="xsd:string" minOccurs="0" />
              <xsd:element name="ContactTitle" type="xsd:string" minOccurs="0" />
              <xsd:element name="Address" type="xsd:string" minOccurs="0" />
              <xsd:element name="City" type="xsd:string" minOccurs="0" />
              <xsd:element name="Region" type="xsd:string" minOccurs="0" />
              <xsd:element name="PostalCode" type="xsd:string" minOccurs="0" />
              <xsd:element name="Country" type="xsd:string" minOccurs="0" />
              <xsd:element name="Phone" type="xsd:string" minOccurs="0" />
              <xsd:element name="Fax" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="Orders">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="OrderID" type="xsd:int" minOccurs="0" />
              <xsd:element name="CustomerID" type="xsd:string" minOccurs="0" />
              <xsd:element name="EmployeeID" type="xsd:int" minOccurs="0" />
              <xsd:element name="OrderDate" type="xsd:dateTime" minOccurs="0" />
              <xsd:element name="RequiredDate" type="xsd:dateTime" minOccurs="0" />
              <xsd:element name="ShippedDate" type="xsd:dateTime" minOccurs="0" />
              <xsd:element name="ShipVia" type="xsd:int" minOccurs="0" />
              <xsd:element name="Freight" type="xsd:decimal" minOccurs="0" />
              <xsd:element name="ShipName" type="xsd:string" minOccurs="0" />
              <xsd:element name="ShipAddress" type="xsd:string" minOccurs="0" />
              <xsd:element name="ShipCity" type="xsd:string" minOccurs="0" />
              <xsd:element name="ShipRegion" type="xsd:string" minOccurs="0" />
              <xsd:element name="ShipPostalCode" type="xsd:string" minOccurs="0" />
              <xsd:element name="ShipCountry" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Data

```
<NewDataSet>
  <Customers>
    <CustomerID>ABC  </CustomerID>
    <CompanyName>ABC Company</CompanyName>
```

```
  <ContactName>Peach</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>One My Way</Address>
</Customers>
<Customers>
  <CustomerID>ALFKI</CustomerID>
  <CompanyName>Alfreds Futterkiste</CompanyName>
  <ContactName>Peach</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Obere Str. 57</Address>
  <City>Berlin</City>
  <PostalCode>12209</PostalCode>
  <Country>Germany</Country>
  <Phone>030-0074321</Phone>
  <Fax>030-0076545</Fax>
</Customers>
<Customers>
  <CustomerID>ANATR</CustomerID>
  <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
  <ContactName>Ana Trujillo</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Avda. de la Constitución 2222</Address>
  <City>México D.F.</City>
  <PostalCode>05021</PostalCode>
  <Country>Mexico</Country>
  <Phone>(5) 555-4729</Phone>
  <Fax>(5) 555-3745</Fax>
</Customers>
<Customers>
  <CustomerID>ANTON</CustomerID>
  <CompanyName>Antonio Moreno Taquería</CompanyName>
  <ContactName>Antonio Moreno</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Mataderos  2312</Address>
  <City>México D.F.</City>
  <PostalCode>05023</PostalCode>
  <Country>Mexico</Country>
  <Phone>(5) 555-3932</Phone>
</Customers>
<Customers>
  <CustomerID>AROUT</CustomerID>
  <CompanyName>Around the Horn</CompanyName>
  <ContactName>Thomas Hardy</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>120 Hanover Sq.</Address>
  <City>London</City>
  <PostalCode>WA1 1DP</PostalCode>
  <Country>UK</Country>
  <Phone>(171) 555-7788</Phone>
  <Fax>(171) 555-6750</Fax>
</Customers>
<Customers>
  <CustomerID>BERGS</CustomerID>
  <CompanyName>Berglunds snabbköp</CompanyName>
```

```
  <ContactName>Christina Berglund</ContactName>
  <ContactTitle>Order Administrator</ContactTitle>
  <Address>Berguvsvägen  8</Address>
  <City>Luleå</City>
  <PostalCode>S-958 22</PostalCode>
  <Country>Sweden</Country>
  <Phone>0921-12 34 65</Phone>
  <Fax>0921-12 34 67</Fax>
</Customers>
<Customers>
  <CustomerID>BLAUS</CustomerID>
  <CompanyName>Blauer See Delikatessen</CompanyName>
  <ContactName>Hanna Moos</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Forsterstr. 57</Address>
  <City>Mannheim</City>
  <PostalCode>68306</PostalCode>
  <Country>Germany</Country>
  <Phone>0621-08460</Phone>
  <Fax>0621-08924</Fax>
</Customers>
<Customers>
  <CustomerID>BLONP</CustomerID>
  <CompanyName>Blondesddsl père et fils</CompanyName>
  <ContactName>Frédérique Citeaux</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>24, place Kléber</Address>
  <City>Strasbourg</City>
  <PostalCode>67000</PostalCode>
  <Country>France</Country>
  <Phone>88.60.15.31</Phone>
  <Fax>88.60.15.32</Fax>
</Customers>
<Customers>
  <CustomerID>BOLID</CustomerID>
  <CompanyName>Bólido Comidas preparadas</CompanyName>
  <ContactName>Martín Sommer</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>C/ Araquil, 67</Address>
  <City>Madrid</City>
  <PostalCode>28023</PostalCode>
  <Country>Spain</Country>
  <Phone>(91) 555 22 82</Phone>
  <Fax>(91) 555 91 99</Fax>
</Customers>
<Customers>
  <CustomerID>BONAP</CustomerID>
  <CompanyName>Bon app'</CompanyName>
  <ContactName>Laurence Lebihan</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>12, rue des Bouchers</Address>
  <City>Marseille</City>
  <PostalCode>13008</PostalCode>
  <Country>France</Country>
```

```
  <Phone>91.24.45.40</Phone>
  <Fax>91.24.45.41</Fax>
</Customers>
<Customers>
  <CustomerID>BOTTM</CustomerID>
  <CompanyName>Bottom-Dollar Markets</CompanyName>
  <ContactName>Elizabeth Lincoln</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>23 Tsawassen Blvd.</Address>
  <City>Tsawassen</City>
  <Region>BC</Region>
  <PostalCode>T2F 8M4</PostalCode>
  <Country>Canada</Country>
  <Phone>(604) 555-4729</Phone>
  <Fax>(604) 555-3745</Fax>
</Customers>
<Customers>
  <CustomerID>BSBEV</CustomerID>
  <CompanyName>B's Beverages</CompanyName>
  <ContactName>Victoria Ashworth</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Fauntleroy Circus</Address>
  <City>London</City>
  <PostalCode>EC2 5NT</PostalCode>
  <Country>UK</Country>
  <Phone>(171) 555-1212</Phone>
</Customers>
<Customers>
  <CustomerID>CACTU</CustomerID>
  <CompanyName>Cactus Comidas para llevar</CompanyName>
  <ContactName>Patricio Simpson</ContactName>
  <ContactTitle>Sales Agent</ContactTitle>
  <Address>Cerrito 333</Address>
  <City>Buenos Aires</City>
  <PostalCode>1010</PostalCode>
  <Country>Argentina</Country>
  <Phone>(1) 135-5555</Phone>
  <Fax>(1) 135-4892</Fax>
</Customers>
<Customers>
  <CustomerID>CENTC</CustomerID>
  <CompanyName>Centro comercial Moctezuma</CompanyName>
  <ContactName>Francisco Chang</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>Sierras de Granada 9993</Address>
  <City>México D.F.</City>
  <PostalCode>05022</PostalCode>
  <Country>Mexico</Country>
  <Phone>(5) 555-3392</Phone>
  <Fax>(5) 555-7293</Fax>
</Customers>
<Customers>
  <CustomerID>CHOPS</CustomerID>
  <CompanyName>Chop-suey Chinese</CompanyName>
```

```
  <ContactName>Yang Wang</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Hauptstr. 29</Address>
  <City>Bern</City>
  <PostalCode>3012</PostalCode>
  <Country>Switzerland</Country>
  <Phone>0452-076545</Phone>
</Customers>
<Customers>
  <CustomerID>COMMI</CustomerID>
  <CompanyName>Comércio Mineiro</CompanyName>
  <ContactName>Pedro Afonso</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>Av. dos Lusíadas, 23</Address>
  <City>Sao Paulo</City>
  <Region>SP</Region>
  <PostalCode>05432-043</PostalCode>
  <Country>Brazil</Country>
  <Phone>(11) 555-7647</Phone>
</Customers>
<Customers>
  <CustomerID>CONSH</CustomerID>
  <CompanyName>Consolidated Holdings</CompanyName>
  <ContactName>Elizabeth Brown</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Berkeley Gardens 12  Brewery</Address>
  <City>London</City>
  <PostalCode>WX1 6LT</PostalCode>
  <Country>UK</Country>
  <Phone>(171) 555-2282</Phone>
  <Fax>(171) 555-9199</Fax>
</Customers>
<Customers>
  <CustomerID>DRACD</CustomerID>
  <CompanyName>Drachenblut Delikatessen</CompanyName>
  <ContactName>Sven Ottlieb</ContactName>
  <ContactTitle>Order Administrator</ContactTitle>
  <Address>Walserweg 21</Address>
  <City>Aachen</City>
  <PostalCode>52066</PostalCode>
  <Country>Germany</Country>
  <Phone>0241-039123</Phone>
  <Fax>0241-059428</Fax>
</Customers>
<Customers>
  <CustomerID>DUMON</CustomerID>
  <CompanyName>Du monde entier</CompanyName>
  <ContactName>Janine Labrune</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>67, rue des Cinquante Otages</Address>
  <City>Nantes</City>
  <PostalCode>44000</PostalCode>
  <Country>France</Country>
  <Phone>40.67.88.88</Phone>
```

```
  <Fax>40.67.89.89</Fax>
</Customers>
<Customers>
  <CustomerID>EASTC</CustomerID>
  <CompanyName>Eastern Connection</CompanyName>
  <ContactName>Ann Devon</ContactName>
  <ContactTitle>Sales Agent</ContactTitle>
  <Address>35 King George</Address>
  <City>London</City>
  <PostalCode>WX3 6FW</PostalCode>
  <Country>UK</Country>
  <Phone>(171) 555-0297</Phone>
  <Fax>(171) 555-3373</Fax>
</Customers>
<Customers>
  <CustomerID>ERNSH</CustomerID>
  <CompanyName>Ernst Handel</CompanyName>
  <ContactName>Roland Mendel</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>Kirchgasse 6</Address>
  <City>Graz</City>
  <PostalCode>8010</PostalCode>
  <Country>Austria</Country>
  <Phone>7675-3425</Phone>
  <Fax>7675-3426</Fax>
</Customers>
<Customers>
  <CustomerID>FAMIA</CustomerID>
  <CompanyName>Familia Arquibaldo</CompanyName>
  <ContactName>Aria Cruz</ContactName>
  <ContactTitle>Marketing Assistant</ContactTitle>
  <Address>Rua Orós, 92</Address>
  <City>Sao Paulo</City>
  <Region>SP</Region>
  <PostalCode>05442-030</PostalCode>
  <Country>Brazil</Country>
  <Phone>(11) 555-9857</Phone>
</Customers>
<Customers>
  <CustomerID>FISSA</CustomerID>
  <CompanyName>FISSA Fabrica Inter. Salchichas S.A.</CompanyName>
  <ContactName>Diego Roel</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>C/ Moralzarzal, 86</Address>
  <City>Madrid</City>
  <PostalCode>28034</PostalCode>
  <Country>Spain</Country>
  <Phone>(91) 555 94 44</Phone>
  <Fax>(91) 555 55 93</Fax>
</Customers>
<Customers>
  <CustomerID>FOLIG</CustomerID>
  <CompanyName>Folies gourmandes</CompanyName>
  <ContactName>Martine Rancé</ContactName>
```

```
  <ContactTitle>Assistant Sales Agent</ContactTitle>
  <Address>184, chaussée de Tournai</Address>
  <City>Lille</City>
  <PostalCode>59000</PostalCode>
  <Country>France</Country>
  <Phone>20.16.10.16</Phone>
  <Fax>20.16.10.17</Fax>
</Customers>
<Customers>
  <CustomerID>FOLKO</CustomerID>
  <CompanyName>Folk och fä HB</CompanyName>
  <ContactName>Maria Larsson</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Åkergatan 24</Address>
  <City>Bräcke</City>
  <PostalCode>S-844 67</PostalCode>
  <Country>Sweden</Country>
  <Phone>0695-34 67 21</Phone>
</Customers>
<Customers>
  <CustomerID>FRANK</CustomerID>
  <CompanyName>Frankenversand</CompanyName>
  <ContactName>Peter Franken</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>Berliner Platz 43</Address>
  <City>München</City>
  <PostalCode>80805</PostalCode>
  <Country>Germany</Country>
  <Phone>089-0877310</Phone>
  <Fax>089-0877451</Fax>
</Customers>
<Customers>
  <CustomerID>FRANR</CustomerID>
  <CompanyName>France restauration</CompanyName>
  <ContactName>Carine Schmitt</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>54, rue Royale</Address>
  <City>Nantes</City>
  <PostalCode>44000</PostalCode>
  <Country>France</Country>
  <Phone>40.32.21.21</Phone>
  <Fax>40.32.21.20</Fax>
</Customers>
<Customers>
  <CustomerID>FRANS</CustomerID>
  <CompanyName>Franchi S.p.A.</CompanyName>
  <ContactName>Paolo Accorti</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Via Monte Bianco 34</Address>
  <City>Torino</City>
  <PostalCode>10100</PostalCode>
  <Country>Italy</Country>
  <Phone>011-4988260</Phone>
  <Fax>011-4988261</Fax>
```

```
</Customers>
<Customers>
  <CustomerID>FURIB</CustomerID>
  <CompanyName>Furia Bacalhau e Frutos do Mar</CompanyName>
  <ContactName>Lino Rodriguez</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>Jardim das rosas n. 32</Address>
  <City>Lisboa</City>
  <PostalCode>1675</PostalCode>
  <Country>Portugal</Country>
  <Phone>(1) 354-2534</Phone>
  <Fax>(1) 354-2535</Fax>
</Customers>
<Customers>
  <CustomerID>GALED</CustomerID>
  <CompanyName>Galería del gastrónomo</CompanyName>
  <ContactName>Eduardo Saavedra</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>Rambla de Cataluña, 23</Address>
  <City>Barcelona</City>
  <PostalCode>08022</PostalCode>
  <Country>Spain</Country>
  <Phone>(93) 203 4560</Phone>
  <Fax>(93) 203 4561</Fax>
</Customers>
<Customers>
  <CustomerID>GODOS</CustomerID>
  <CompanyName>Godos Cocina Típica</CompanyName>
  <ContactName>José Pedro Freyre</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>C/ Romero, 33</Address>
  <City>Sevilla</City>
  <PostalCode>41101</PostalCode>
  <Country>Spain</Country>
  <Phone>(95) 555 82 82</Phone>
</Customers>
<Customers>
  <CustomerID>GOURL</CustomerID>
  <CompanyName>Gourmet Lanchonetes</CompanyName>
  <ContactName>André Fonseca</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>Av. Brasil, 442</Address>
  <City>Campinas</City>
  <Region>SP</Region>
  <PostalCode>04876-786</PostalCode>
  <Country>Brazil</Country>
  <Phone>(11) 555-9482</Phone>
</Customers>
<Customers>
  <CustomerID>GREAL</CustomerID>
  <CompanyName>Great Lakes Food Market</CompanyName>
  <ContactName>Howard Snyder</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>2732 Baker Blvd.</Address>
```

```
  <City>Eugene</City>
  <Region>OR</Region>
  <PostalCode>97403</PostalCode>
  <Country>USA</Country>
  <Phone>(503) 555-7555</Phone>
</Customers>
<Customers>
  <CustomerID>GROSR</CustomerID>
  <CompanyName>GROSELLA-Restaurante</CompanyName>
  <ContactName>Manuel Pereira</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>5ª Ave. Los Palos Grandes</Address>
  <City>Caracas</City>
  <Region>DF</Region>
  <PostalCode>1081</PostalCode>
  <Country>Venezuela</Country>
  <Phone>(2) 283-2951</Phone>
  <Fax>(2) 283-3397</Fax>
</Customers>
<Customers>
  <CustomerID>HANAR</CustomerID>
  <CompanyName>Hanari Carnes</CompanyName>
  <ContactName>Mario Pontes</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>Rua do Paço, 67</Address>
  <City>Rio de Janeiro</City>
  <Region>RJ</Region>
  <PostalCode>05454-876</PostalCode>
  <Country>Brazil</Country>
  <Phone>(21) 555-0091</Phone>
  <Fax>(21) 555-8765</Fax>
</Customers>
<Customers>
  <CustomerID>HILAA</CustomerID>
  <CompanyName>HILARION-Abastos</CompanyName>
  <ContactName>Carlos Hernández</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Carrera 22 con Ave. Carlos Soublette #8-35</Address>
  <City>San Cristóbal</City>
  <Region>Táchira</Region>
  <PostalCode>5022</PostalCode>
  <Country>Venezuela</Country>
  <Phone>(5) 555-1340</Phone>
  <Fax>(5) 555-1948</Fax>
</Customers>
<Customers>
  <CustomerID>HUNGC</CustomerID>
  <CompanyName>Hungry Coyote Import Store</CompanyName>
  <ContactName>Yoshi Latimer</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>City Center Plaza 516 Main St.</Address>
  <City>Elgin</City>
  <Region>OR</Region>
  <PostalCode>97827</PostalCode>
```

```
  <Country>USA</Country>
  <Phone>(503) 555-6874</Phone>
  <Fax>(503) 555-2376</Fax>
</Customers>
<Customers>
  <CustomerID>HUNGO</CustomerID>
  <CompanyName>Hungry Owl All-Night Grocers</CompanyName>
  <ContactName>Patricia McKenna</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>8 Johnstown Road</Address>
  <City>Cork</City>
  <Region>Co. Cork</Region>
  <Country>Ireland</Country>
  <Phone>2967 542</Phone>
  <Fax>2967 3333</Fax>
</Customers>
<Customers>
  <CustomerID>ISLAT</CustomerID>
  <CompanyName>Island Trading</CompanyName>
  <ContactName>Helen Bennett</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>Garden House Crowther Way</Address>
  <City>Cowes</City>
  <Region>Isle of Wight</Region>
  <PostalCode>PO31 7PJ</PostalCode>
  <Country>UK</Country>
  <Phone>(198) 555-8888</Phone>
</Customers>
<Customers>
  <CustomerID>KOENE</CustomerID>
  <CompanyName>Königlich Essen</CompanyName>
  <ContactName>Philip Cramer</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>Maubelstr. 90</Address>
  <City>Brandenburg</City>
  <PostalCode>14776</PostalCode>
  <Country>Germany</Country>
  <Phone>0555-09876</Phone>
</Customers>
<Customers>
  <CustomerID>LACOR</CustomerID>
  <CompanyName>La corne d'abondance</CompanyName>
  <ContactName>Daniel Tonini</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>67, avenue de l'Europe</Address>
  <City>Versailles</City>
  <PostalCode>78000</PostalCode>
  <Country>France</Country>
  <Phone>30.59.84.10</Phone>
  <Fax>30.59.85.11</Fax>
</Customers>
<Customers>
  <CustomerID>LAMAI</CustomerID>
  <CompanyName>La maison d'Asie</CompanyName>
```

```
  <ContactName>Annette Roulet</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>1 rue Alsace-Lorraine</Address>
  <City>Toulouse</City>
  <PostalCode>31000</PostalCode>
  <Country>France</Country>
  <Phone>61.77.61.10</Phone>
  <Fax>61.77.61.11</Fax>
</Customers>
<Customers>
  <CustomerID>LAUGB</CustomerID>
  <CompanyName>Laughing Bacchus Wine Cellars</CompanyName>
  <ContactName>Yoshi Tannamuri</ContactName>
  <ContactTitle>Marketing Assistant</ContactTitle>
  <Address>1900 Oak St.</Address>
  <City>Vancouver</City>
  <Region>BC</Region>
  <PostalCode>V3F 2K1</PostalCode>
  <Country>Canada</Country>
  <Phone>(604) 555-3392</Phone>
  <Fax>(604) 555-7293</Fax>
</Customers>
<Customers>
  <CustomerID>LAZYK</CustomerID>
  <CompanyName>Lazy K Kountry Store</CompanyName>
  <ContactName>John Steel</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>12 Orchestra Terrace</Address>
  <City>Walla Walla</City>
  <Region>WA</Region>
  <PostalCode>99362</PostalCode>
  <Country>USA</Country>
  <Phone>(509) 555-7969</Phone>
  <Fax>(509) 555-6221</Fax>
</Customers>
<Customers>
  <CustomerID>LEHMS</CustomerID>
  <CompanyName>Lehmanns Marktstand</CompanyName>
  <ContactName>Renate Messner</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Magazinweg 7</Address>
  <City>Frankfurt a.M.</City>
  <PostalCode>60528</PostalCode>
  <Country>Germany</Country>
  <Phone>069-0245984</Phone>
  <Fax>069-0245874</Fax>
</Customers>
<Customers>
  <CustomerID>LETSS</CustomerID>
  <CompanyName>Let's Stop N Shop</CompanyName>
  <ContactName>Jaime Yorres</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>87 Polk St. Suite 5</Address>
  <City>San Francisco</City>
```

```
  <Region>CA</Region>
  <PostalCode>94117</PostalCode>
  <Country>USA</Country>
  <Phone>(415) 555-5938</Phone>
</Customers>
<Customers>
  <CustomerID>LILAS</CustomerID>
  <CompanyName>LILA-Supermercado</CompanyName>
  <ContactName>Carlos González</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</Address>
  <City>Barquisimeto</City>
  <Region>Lara</Region>
  <PostalCode>3508</PostalCode>
  <Country>Venezuela</Country>
  <Phone>(9) 331-6954</Phone>
  <Fax>(9) 331-7256</Fax>
</Customers>
<Customers>
  <CustomerID>LINOD</CustomerID>
  <CompanyName>LINO-Delicateses</CompanyName>
  <ContactName>Felipe Izquierdo</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Ave. 5 de Mayo Porlamar</Address>
  <City>I. de Margarita</City>
  <Region>Nueva Esparta</Region>
  <PostalCode>4980</PostalCode>
  <Country>Venezuela</Country>
  <Phone>(8) 34-56-12</Phone>
  <Fax>(8) 34-93-93</Fax>
</Customers>
<Customers>
  <CustomerID>LONEP</CustomerID>
  <CompanyName>Lonesome Pine Restaurant</CompanyName>
  <ContactName>Fran Wilson</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>89 Chiaroscuro Rd.</Address>
  <City>Portland</City>
  <Region>OR</Region>
  <PostalCode>97219</PostalCode>
  <Country>USA</Country>
  <Phone>(503) 555-9573</Phone>
  <Fax>(503) 555-9646</Fax>
</Customers>
<Customers>
  <CustomerID>MAGAA</CustomerID>
  <CompanyName>Magazzini Alimentari Riuniti</CompanyName>
  <ContactName>Giovanni Rovelli</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>Via Ludovico il Moro 22</Address>
  <City>Bergamo</City>
  <PostalCode>24100</PostalCode>
  <Country>Italy</Country>
  <Phone>035-640230</Phone>
```

```
  <Fax>035-640231</Fax>
</Customers>
<Customers>
  <CustomerID>MAISD</CustomerID>
  <CompanyName>Maison Dewey</CompanyName>
  <ContactName>Catherine Dewey</ContactName>
  <ContactTitle>Sales Agent</ContactTitle>
  <Address>Rue Joseph-Bens 532</Address>
  <City>Bruxelles</City>
  <PostalCode>B-1180</PostalCode>
  <Country>Belgium</Country>
  <Phone>(02) 201 24 67</Phone>
  <Fax>(02) 201 24 68</Fax>
</Customers>
<Customers>
  <CustomerID>MEREP</CustomerID>
  <CompanyName>Mère Paillarde</CompanyName>
  <ContactName>Jean Fresnière</ContactName>
  <ContactTitle>Marketing Assistant</ContactTitle>
  <Address>43 rue St. Laurent</Address>
  <City>Montréal</City>
  <Region>Québec</Region>
  <PostalCode>H1J 1C3</PostalCode>
  <Country>Canada</Country>
  <Phone>(514) 555-8054</Phone>
  <Fax>(514) 555-8055</Fax>
</Customers>
<Customers>
  <CustomerID>MORGK</CustomerID>
  <CompanyName>Morgenstern Gesundkost</CompanyName>
  <ContactName>Alexander Feuer</ContactName>
  <ContactTitle>Marketing Assistant</ContactTitle>
  <Address>Heerstr. 22</Address>
  <City>Leipzig</City>
  <PostalCode>04179</PostalCode>
  <Country>Germany</Country>
  <Phone>0342-023176</Phone>
</Customers>
<Customers>
  <CustomerID>NewID</CustomerID>
  <CompanyName>New Company Name</CompanyName>
  <ContactName>New Name</ContactName>
</Customers>
<Customers>
  <CustomerID>NORTS</CustomerID>
  <CompanyName>North/South</CompanyName>
  <ContactName>Simon Crowther</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>South House 300 Queensbridge</Address>
  <City>London</City>
  <PostalCode>SW7 1RZ</PostalCode>
  <Country>UK</Country>
  <Phone>(171) 555-7733</Phone>
  <Fax>(171) 555-2530</Fax>
```

```
</Customers>
<Customers>
  <CustomerID>OCEAN</CustomerID>
  <CompanyName>Océano Atlántico Ltda.</CompanyName>
  <ContactName>Yvonne Moncada</ContactName>
  <ContactTitle>Sales Agent</ContactTitle>
  <Address>Ing. Gustavo Moncada 8585 Piso 20-A</Address>
  <City>Buenos Aires</City>
  <PostalCode>1010</PostalCode>
  <Country>Argentina</Country>
  <Phone>(1) 135-5333</Phone>
  <Fax>(1) 135-5535</Fax>
</Customers>
<Customers>
  <CustomerID>OLDWO</CustomerID>
  <CompanyName>Old World Delicatessen</CompanyName>
  <ContactName>Rene Phillips</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>2743 Bering St.</Address>
  <City>Anchorage</City>
  <Region>AK</Region>
  <PostalCode>99508</PostalCode>
  <Country>USA</Country>
  <Phone>(907) 555-7584</Phone>
  <Fax>(907) 555-2880</Fax>
</Customers>
<Customers>
  <CustomerID>OTTIK</CustomerID>
  <CompanyName>Ottilies Käseladen</CompanyName>
  <ContactName>Henriette Pfalzheim</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Mehrheimerstr. 369</Address>
  <City>Köln</City>
  <PostalCode>50739</PostalCode>
  <Country>Germany</Country>
  <Phone>0221-0644327</Phone>
  <Fax>0221-0765721</Fax>
</Customers>
<Customers>
  <CustomerID>PARIS</CustomerID>
  <CompanyName>Paris spécialités</CompanyName>
  <ContactName>Marie Bertrand</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>265, boulevard Charonne</Address>
  <City>Paris</City>
  <PostalCode>75012</PostalCode>
  <Country>France</Country>
  <Phone>(1) 42.34.22.66</Phone>
  <Fax>(1) 42.34.22.77</Fax>
</Customers>
<Customers>
  <CustomerID>PERIC</CustomerID>
  <CompanyName>Pericles Comidas clásicas</CompanyName>
```

<ContactName>Guillermo Fernández</ContactName>
<ContactTitle>Sales Representative</ContactTitle>
<Address>Calle Dr. Jorge Cash 321</Address>
<City>México D.F.</City>
<PostalCode>05033</PostalCode>
<Country>Mexico</Country>
<Phone>(5) 552-3745</Phone>
<Fax>(5) 545-3745</Fax>
</Customers>
<Customers>
<CustomerID>PICCO</CustomerID>
<CompanyName>Piccolo und mehr</CompanyName>
<ContactName>Georg Pipps</ContactName>
<ContactTitle>Sales Manager</ContactTitle>
<Address>Geislweg 14</Address>
<City>Salzburg</City>
<PostalCode>5020</PostalCode>
<Country>Austria</Country>
<Phone>6562-9722</Phone>
<Fax>6562-9723</Fax>
</Customers>
<Customers>
<CustomerID>PRINI</CustomerID>
<CompanyName>Princesa Isabel Vinhos</CompanyName>
<ContactName>Isabel de Castro</ContactName>
<ContactTitle>Sales Representative</ContactTitle>
<Address>Estrada da saúde n. 58</Address>
<City>Lisboa</City>
<PostalCode>1756</PostalCode>
<Country>Portugal</Country>
<Phone>(1) 356-5634</Phone>
</Customers>
<Customers>
<CustomerID>QUEDE</CustomerID>
<CompanyName>Que Delícia</CompanyName>
<ContactName>Bernardo Batista</ContactName>
<ContactTitle>Accounting Manager</ContactTitle>
<Address>Rua da Panificadora, 12</Address>
<City>Rio de Janeiro</City>
<Region>RJ</Region>
<PostalCode>02389-673</PostalCode>
<Country>Brazil</Country>
<Phone>(21) 555-4252</Phone>
<Fax>(21) 555-4545</Fax>
</Customers>
<Customers>
<CustomerID>QUEEN</CustomerID>
<CompanyName>Queen Cozinha</CompanyName>
<ContactName>Lúcia Carvalho</ContactName>
<ContactTitle>Marketing Assistant</ContactTitle>
<Address>Alameda dos Canàrios, 891</Address>
<City>Sao Paulo</City>
<Region>SP</Region>
<PostalCode>05487-020</PostalCode>

```
  <Country>Brazil</Country>
  <Phone>(11) 555-1189</Phone>
</Customers>
<Customers>
  <CustomerID>QUICK</CustomerID>
  <CompanyName>QUICK-Stop</CompanyName>
  <ContactName>Horst Kloss</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>Taucherstraße 10</Address>
  <City>Cunewalde</City>
  <PostalCode>01307</PostalCode>
  <Country>Germany</Country>
  <Phone>0372-035188</Phone>
</Customers>
<Customers>
  <CustomerID>RANCH</CustomerID>
  <CompanyName>Rancho grande</CompanyName>
  <ContactName>Sergio Gutiérrez</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Av. del Libertador 900</Address>
  <City>Buenos Aires</City>
  <PostalCode>1010</PostalCode>
  <Country>Argentina</Country>
  <Phone>(1) 123-5555</Phone>
  <Fax>(1) 123-5556</Fax>
</Customers>
<Customers>
  <CustomerID>RATTC</CustomerID>
  <CompanyName>Rattlesnake Canyon Grocery</CompanyName>
  <ContactName>Paula Wilson</ContactName>
  <ContactTitle>Assistant Sales Representative</ContactTitle>
  <Address>2817 Milton Dr.</Address>
  <City>Albuquerque</City>
  <Region>NM</Region>
  <PostalCode>87110</PostalCode>
  <Country>USA</Country>
  <Phone>(505) 555-5939</Phone>
  <Fax>(505) 555-3620</Fax>
</Customers>
<Customers>
  <CustomerID>REGGC</CustomerID>
  <CompanyName>Reggiani Caseifici</CompanyName>
  <ContactName>Maurizio Moroni</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>Strada Provinciale 124</Address>
  <City>Reggio Emilia</City>
  <PostalCode>42100</PostalCode>
  <Country>Italy</Country>
  <Phone>0522-556721</Phone>
  <Fax>0522-556722</Fax>
</Customers>
<Customers>
  <CustomerID>RICAR</CustomerID>
  <CompanyName>Ricardo Adocicados</CompanyName>
```

<Customers>
<ContactName>Janete Limeira</ContactName>
<ContactTitle>Assistant Sales Agent</ContactTitle>
<Address>Av. Copacabana, 267</Address>
<City>Rio de Janeiro</City>
<Region>RJ</Region>
<PostalCode>02389-890</PostalCode>
<Country>Brazil</Country>
<Phone>(21) 555-3412</Phone>
</Customers>
<Customers>
<CustomerID>RICSU</CustomerID>
<CompanyName>Richter Supermarkt</CompanyName>
<ContactName>Michael Holz</ContactName>
<ContactTitle>Sales Manager</ContactTitle>
<Address>Grenzacherweg 237</Address>
<City>Genève</City>
<PostalCode>1203</PostalCode>
<Country>Switzerland</Country>
<Phone>0897-034214</Phone>
</Customers>
<Customers>
<CustomerID>ROMEY</CustomerID>
<CompanyName>Romero y tomillo</CompanyName>
<ContactName>Alejandra Camino</ContactName>
<ContactTitle>Accounting Manager</ContactTitle>
<Address>Gran Vía, 1</Address>
<City>Madrid</City>
<PostalCode>28001</PostalCode>
<Country>Spain</Country>
<Phone>(91) 745 6200</Phone>
<Fax>(91) 745 6210</Fax>
</Customers>
<Customers>
<CustomerID>SANTG</CustomerID>
<CompanyName>Santé Gourmet</CompanyName>
<ContactName>Jonas Bergulfsen</ContactName>
<ContactTitle>Owner</ContactTitle>
<Address>Erling Skakkes gate 78</Address>
<City>Stavern</City>
<PostalCode>4110</PostalCode>
<Country>Norway</Country>
<Phone>07-98 92 35</Phone>
<Fax>07-98 92 47</Fax>
</Customers>
<Customers>
<CustomerID>SAVEA</CustomerID>
<CompanyName>Save-a-lot Markets</CompanyName>
<ContactName>Jose Pavarotti</ContactName>
<ContactTitle>Sales Representative</ContactTitle>
<Address>187 Suffolk Ln.</Address>
<City>Boise</City>
<Region>ID</Region>
<PostalCode>83720</PostalCode>
<Country>USA</Country>

```
  <Phone>(208) 555-8097</Phone>
</Customers>
<Customers>
  <CustomerID>SEVES</CustomerID>
  <CompanyName>Seven Seas Imports</CompanyName>
  <ContactName>Hari Kumar</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>90 Wadhurst Rd.</Address>
  <City>London</City>
  <PostalCode>OX15 4NB</PostalCode>
  <Country>UK</Country>
  <Phone>(171) 555-1717</Phone>
  <Fax>(171) 555-5646</Fax>
</Customers>
<Customers>
  <CustomerID>SIMOB</CustomerID>
  <CompanyName>Simons bistro</CompanyName>
  <ContactName>Jytte Petersen</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Vinbæltet 34</Address>
  <City>Kobenhavn</City>
  <PostalCode>1734</PostalCode>
  <Country>Denmark</Country>
  <Phone>31 12 34 56</Phone>
  <Fax>31 13 35 57</Fax>
</Customers>
<Customers>
  <CustomerID>SPECD</CustomerID>
  <CompanyName>Spécialités du monde</CompanyName>
  <ContactName>Dominique Perrier</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>25, rue Lauriston</Address>
  <City>Paris</City>
  <PostalCode>75016</PostalCode>
  <Country>France</Country>
  <Phone>(1) 47.55.60.10</Phone>
  <Fax>(1) 47.55.60.20</Fax>
</Customers>
<Customers>
  <CustomerID>SPLIR</CustomerID>
  <CompanyName>Split Rail Beer &amp;amp; Ale</CompanyName>
  <ContactName>Art Braunschweiger</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>P.O. Box 555</Address>
  <City>Lander</City>
  <Region>WY</Region>
  <PostalCode>82520</PostalCode>
  <Country>USA</Country>
  <Phone>(307) 555-4680</Phone>
  <Fax>(307) 555-6525</Fax>
</Customers>
<Customers>
  <CustomerID>SUPRD</CustomerID>
  <CompanyName>Suprêmes délices</CompanyName>
```

```
  <ContactName>Pascale Cartrain</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>Boulevard Tirou, 255</Address>
  <City>Charleroi</City>
  <PostalCode>B-6000</PostalCode>
  <Country>Belgium</Country>
  <Phone>(071) 23 67 22 20</Phone>
  <Fax>(071) 23 67 22 21</Fax>
</Customers>
<Customers>
  <CustomerID>THEBI</CustomerID>
  <CompanyName>The Big Cheese</CompanyName>
  <ContactName>Liz Nixon</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>89 Jefferson Way Suite 2</Address>
  <City>Portland</City>
  <Region>OR</Region>
  <PostalCode>97201</PostalCode>
  <Country>USA</Country>
  <Phone>(503) 555-3612</Phone>
</Customers>
<Customers>
  <CustomerID>THECR</CustomerID>
  <CompanyName>The Cracker Box</CompanyName>
  <ContactName>Liu Wong</ContactName>
  <ContactTitle>Marketing Assistant</ContactTitle>
  <Address>55 Grizzly Peak Rd.</Address>
  <City>Butte</City>
  <Region>MT</Region>
  <PostalCode>59801</PostalCode>
  <Country>USA</Country>
  <Phone>(406) 555-5834</Phone>
  <Fax>(406) 555-8083</Fax>
</Customers>
<Customers>
  <CustomerID>TOMSP</CustomerID>
  <CompanyName>Toms Spezialitäten</CompanyName>
  <ContactName>Karin Josephs</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Address>Luisenstr. 48</Address>
  <City>Münster</City>
  <PostalCode>44087</PostalCode>
  <Country>Germany</Country>
  <Phone>0251-031259</Phone>
  <Fax>0251-035695</Fax>
</Customers>
<Customers>
  <CustomerID>TORTU</CustomerID>
  <CompanyName>Tortuga Restaurante</CompanyName>
  <ContactName>Miguel Angel Paolino</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>Avda. Azteca 123</Address>
  <City>México D.F.</City>
  <PostalCode>05033</PostalCode>
```

```
  <Country>Mexico</Country>
  <Phone>(5) 555-2933</Phone>
</Customers>
<Customers>
  <CustomerID>TRADH</CustomerID>
  <CompanyName>Tradição Hipermercados</CompanyName>
  <ContactName>Anabela Domingues</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Av. Inês de Castro, 414</Address>
  <City>Sao Paulo</City>
  <Region>SP</Region>
  <PostalCode>05634-030</PostalCode>
  <Country>Brazil</Country>
  <Phone>(11) 555-2167</Phone>
  <Fax>(11) 555-2168</Fax>
</Customers>
<Customers>
  <CustomerID>TRAIH</CustomerID>
  <CompanyName>Trail's Head Gourmet Provisioners</CompanyName>
  <ContactName>Helvetius Nagy</ContactName>
  <ContactTitle>Sales Associate</ContactTitle>
  <Address>722 DaVinci Blvd.</Address>
  <City>Kirkland</City>
  <Region>WA</Region>
  <PostalCode>98034</PostalCode>
  <Country>USA</Country>
  <Phone>(206) 555-8257</Phone>
  <Fax>(206) 555-2174</Fax>
</Customers>
<Customers>
  <CustomerID>VAFFE</CustomerID>
  <CompanyName>Vaffeljernet</CompanyName>
  <ContactName>Palle Ibsen</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>Smagsloget 45</Address>
  <City>Århus</City>
  <PostalCode>8200</PostalCode>
  <Country>Denmark</Country>
  <Phone>86 21 32 43</Phone>
  <Fax>86 22 33 44</Fax>
</Customers>
<Customers>
  <CustomerID>VICTE</CustomerID>
  <CompanyName>Victuailles en stock</CompanyName>
  <ContactName>Mary Saveley</ContactName>
  <ContactTitle>Sales Agent</ContactTitle>
  <Address>2, rue du Commerce</Address>
  <City>Lyon</City>
  <PostalCode>69004</PostalCode>
  <Country>France</Country>
  <Phone>78.32.54.86</Phone>
  <Fax>78.32.54.87</Fax>
</Customers>
<Customers>
```

```
  <CustomerID>VINET</CustomerID>
  <CompanyName>Vins et alcools Chevalier</CompanyName>
  <ContactName>Paul Henriot</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>59 rue de l'Abbaye</Address>
  <City>Reims</City>
  <PostalCode>51100</PostalCode>
  <Country>France</Country>
  <Phone>26.47.15.10</Phone>
  <Fax>26.47.15.11</Fax>
</Customers>
<Customers>
  <CustomerID>WANDK</CustomerID>
  <CompanyName>Die Wandernde Kuh</CompanyName>
  <ContactName>Rita Müller</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Adenauerallee 900</Address>
  <City>Stuttgart</City>
  <PostalCode>70563</PostalCode>
  <Country>Germany</Country>
  <Phone>0711-020361</Phone>
  <Fax>0711-035428</Fax>
</Customers>
<Customers>
  <CustomerID>WARTH</CustomerID>
  <CompanyName>Wartian Herkku</CompanyName>
  <ContactName>Pirkko Koskitalo</ContactName>
  <ContactTitle>Accounting Manager</ContactTitle>
  <Address>Torikatu 38</Address>
  <City>Oulu</City>
  <PostalCode>90110</PostalCode>
  <Country>Finland</Country>
  <Phone>981-443655</Phone>
  <Fax>981-443655</Fax>
</Customers>
<Customers>
  <CustomerID>WELLI</CustomerID>
  <CompanyName>Wellington Importadora</CompanyName>
  <ContactName>Paula Parente</ContactName>
  <ContactTitle>Sales Manager</ContactTitle>
  <Address>Rua do Mercado, 12</Address>
  <City>Resende</City>
  <Region>SP</Region>
  <PostalCode>08737-363</PostalCode>
  <Country>Brazil</Country>
  <Phone>(14) 555-8122</Phone>
</Customers>
<Customers>
  <CustomerID>WHITC</CustomerID>
  <CompanyName>White Clover Markets</CompanyName>
  <ContactName>Karl Jablonski</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>305 - 14th Ave. S. Suite 3B</Address>
  <City>Seattle</City>
```

```
  <Region>WA</Region>
  <PostalCode>98128</PostalCode>
  <Country>USA</Country>
  <Phone>(206) 555-4112</Phone>
  <Fax>(206) 555-4115</Fax>
</Customers>
<Customers>
  <CustomerID>WILMK</CustomerID>
  <CompanyName>Wilman Kala</CompanyName>
  <ContactName>Matti Karttunen</ContactName>
  <ContactTitle>Owner/Marketing Assistant</ContactTitle>
  <Address>Keskuskatu 45</Address>
  <City>Helsinki</City>
  <PostalCode>21240</PostalCode>
  <Country>Finland</Country>
  <Phone>90-224 8858</Phone>
  <Fax>90-224 8858</Fax>
</Customers>
<Customers>
  <CustomerID>WOLZA</CustomerID>
  <CompanyName>Wolski  Zajazd</CompanyName>
  <ContactName>Zbyszek Piestrzeniewicz</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Address>ul. Filtrowa 68</Address>
  <City>Warszawa</City>
  <PostalCode>01-012</PostalCode>
  <Country>Poland</Country>
  <Phone>(26) 642-7012</Phone>
  <Fax>(26) 642-7012</Fax>
</Customers>
<Orders>
  <OrderID>10248</OrderID>
  <CustomerID>VINET</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-07-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>32.38</Freight>
  <ShipName>Vins et alcools Chevalier</ShipName>
  <ShipAddress>59 rue de l'Abbaye</ShipAddress>
  <ShipCity>Reims</ShipCity>
  <ShipPostalCode>51100</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10249</OrderID>
  <CustomerID>TOMSP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-07-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>11.61</Freight>
```

```
  <ShipName>Toms Spezialitäten</ShipName>
  <ShipAddress>Luisenstr. 48</ShipAddress>
  <ShipCity>Münster</ShipCity>
  <ShipPostalCode>44087</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10250</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>65.83</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10251</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-07-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>41.34</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10252</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>51.3</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10253</OrderID>
```

```
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-07-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-07-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>58.17</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10254</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-07-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>22.98</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10255</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1996-07-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>148.33</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10256</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-07-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>13.97</Freight>
  <ShipName>Wellington Importadora</ShipName>
```

```
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10257</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>81.91</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10258</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-07-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>140.51</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10259</OrderID>
  <CustomerID>CENTC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>3.25</Freight>
  <ShipName>Centro comercial Moctezuma</ShipName>
  <ShipAddress>Sierras de Granada 9993</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05022</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10260</OrderID>
```

```
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>55.09</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10261</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.05</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10262</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-07-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>48.29</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10263</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1996-07-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-31T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>146.06</Freight>
```

```
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10264</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-07-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>3.67</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10265</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-07-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>55.28</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10266</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-07-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-07-31T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>25.73</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10267</OrderID>
```

```
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-07-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>208.58</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10268</OrderID>
  <CustomerID>GROSR</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-07-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>66.29</Freight>
  <ShipName>GROSELLA-Restaurante</ShipName>
  <ShipAddress>5ª Ave. Los Palos Grandes</ShipAddress>
  <ShipCity>Caracas</ShipCity>
  <ShipRegion>DF</ShipRegion>
  <ShipPostalCode>1081</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10269</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-07-31T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.56</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10270</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-08-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>136.54</Freight>
```

```
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10271</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-08-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.54</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10272</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-08-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>98.03</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10273</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-08-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>76.07</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10274</OrderID>
  <CustomerID>VINET</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-08-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>6.01</Freight>
  <ShipName>Vins et alcools Chevalier</ShipName>
  <ShipAddress>59 rue de l'Abbaye</ShipAddress>
  <ShipCity>Reims</ShipCity>
  <ShipPostalCode>51100</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10275</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-08-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>26.93</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10276</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-08-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>13.84</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10277</OrderID>
  <CustomerID>MORGK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-08-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>125.77</Freight>
  <ShipName>Morgenstern Gesundkost</ShipName>
```

```
  <ShipAddress>Heerstr. 22</ShipAddress>
  <ShipCity>Leipzig</ShipCity>
  <ShipPostalCode>04179</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10278</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-08-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>92.69</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10279</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-08-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>25.83</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10280</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-08-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.98</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10281</OrderID>
  <CustomerID>ROMEY</CustomerID>
  <EmployeeID>4</EmployeeID>
```

```
  <OrderDate>1996-08-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-08-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>2.94</Freight>
  <ShipName>Romero y tomillo</ShipName>
  <ShipAddress>Gran Vía, 1</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28001</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10282</OrderID>
  <CustomerID>ROMEY</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-08-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>12.69</Freight>
  <ShipName>Romero y tomillo</ShipName>
  <ShipAddress>Gran Vía, 1</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28001</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10283</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-08-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>84.81</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10284</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-08-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>76.56</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
```

```
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10285</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-08-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>76.83</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10286</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-08-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>229.24</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10287</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-08-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>12.76</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10288</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-08-23T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1996-09-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>7.45</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10289</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-08-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-08-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>22.77</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10290</OrderID>
  <CustomerID>COMMI</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-08-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>79.7</Freight>
  <ShipName>Comércio Mineiro</ShipName>
  <ShipAddress>Av. dos Lusíadas, 23</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05432-043</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10291</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-08-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>6.4</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
```

```
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10292</OrderID>
  <CustomerID>TRADH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-08-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.35</Freight>
  <ShipName>Tradiçao Hipermercados</ShipName>
  <ShipAddress>Av. Inês de Castro, 414</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05634-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10293</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-08-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>21.18</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10294</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-08-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>147.26</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10295</OrderID>
  <CustomerID>VINET</CustomerID>
  <EmployeeID>2</EmployeeID>
```

```
  <OrderDate>1996-09-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-09-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.15</Freight>
  <ShipName>Vins et alcools Chevalier</ShipName>
  <ShipAddress>59 rue de l'Abbaye</ShipAddress>
  <ShipCity>Reims</ShipCity>
  <ShipPostalCode>51100</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10296</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-09-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.12</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10297</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-09-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>5.74</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10298</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-09-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>168.22</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
```

```
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10299</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-09-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>29.76</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10300</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-09-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>17.68</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10301</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-09-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>45.08</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10302</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-09-10T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1996-10-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>6.27</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10303</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-09-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>107.83</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10304</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-09-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>63.79</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10305</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-09-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>257.62</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
```

```
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10306</OrderID>
  <CustomerID>ROMEY</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-09-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>7.56</Freight>
  <ShipName>Romero y tomillo</ShipName>
  <ShipAddress>Gran Vía, 1</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28001</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10307</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-09-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>0.56</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10308</OrderID>
  <CustomerID>ANATR</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-09-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.61</Freight>
  <ShipName>Ana Trujillo Emparedados y helados</ShipName>
  <ShipAddress>Avda. de la Constitución 2222</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05021</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10309</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-09-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-17T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShippedDate>1996-10-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>47.3</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10310</OrderID>
  <CustomerID>THEBI</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-09-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>17.52</Freight>
  <ShipName>The Big Cheese</ShipName>
  <ShipAddress>89 Jefferson Way Suite 2</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97201</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10311</OrderID>
  <CustomerID>DUMON</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-09-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-09-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>24.69</Freight>
  <ShipName>Du monde entier</ShipName>
  <ShipAddress>67, rue des Cinquante Otages</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10312</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-09-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>40.26</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
```

```
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10313</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-09-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.96</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10314</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-09-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>74.16</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10315</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-09-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>41.76</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10316</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-09-27T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1996-10-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>150.15</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10317</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-09-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>12.69</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10318</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-10-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.73</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10319</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-10-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>64.5</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
```

```
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10320</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-10-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>34.57</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10321</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-10-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.43</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10322</OrderID>
  <CustomerID>PERIC</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-10-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>0.4</Freight>
  <ShipName>Pericles Comidas clásicas</ShipName>
  <ShipAddress>Calle Dr. Jorge Cash 321</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10323</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>4</EmployeeID>
```

```
  <OrderDate>1996-10-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.88</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10324</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1996-10-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>214.27</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10325</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-10-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-10-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1996-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>64.86</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10326</OrderID>
  <CustomerID>BOLID</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>77.92</Freight>
  <ShipName>Bólido Comidas preparadas</ShipName>
  <ShipAddress>C/ Araquil, 67</ShipAddress>
  <ShipCity>Madrid</ShipCity>
```

```
  <ShipPostalCode>28023</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10327</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-10-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-08T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>63.36</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10328</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>87.03</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10329</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>191.67</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10330</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-10-16T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1996-11-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>12.75</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10331</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1996-10-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>10.19</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10332</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-10-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>52.84</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10333</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-10-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>0.59</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
```

```
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10334</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-10-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>8.56</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10335</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-10-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>42.11</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10336</OrderID>
  <CustomerID>PRINI</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-10-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>15.51</Freight>
  <ShipName>Princesa Isabel Vinhos</ShipName>
  <ShipAddress>Estrada da saúde n. 58</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1756</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10337</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-21T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1996-10-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>108.26</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10338</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1996-11-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-10-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>84.21</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10339</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-10-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>15.66</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10340</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-10-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>166.31</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
```

```
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10341</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-10-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>26.78</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10342</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>54.83</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10343</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-10-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>110.37</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10344</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-11-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-05T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>23.29</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10345</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-11-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>249.06</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10346</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-11-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>142.08</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10347</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-11-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>3.1</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
```

```
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10348</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-11-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>0.78</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10349</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-11-08T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.63</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10350</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-11-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>64.19</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10351</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-11-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-09T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1996-11-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>162.33</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10352</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-11-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-11-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.3</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10353</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-11-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>360.63</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10354</OrderID>
  <CustomerID>PERIC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-11-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>53.8</Freight>
  <ShipName>Pericles Comidas clásicas</ShipName>
  <ShipAddress>Calle Dr. Jorge Cash 321</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10355</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-11-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>41.95</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10356</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-11-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>36.71</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10357</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-11-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>34.88</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10358</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-11-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-18T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1996-11-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>19.64</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10359</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-11-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>288.43</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10360</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-11-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>131.7</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10361</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-11-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>183.17</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10362</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-11-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-11-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>96.04</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10363</OrderID>
  <CustomerID>DRACD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-11-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>30.54</Freight>
  <ShipName>Drachenblut Delikatessen</ShipName>
  <ShipAddress>Walserweg 21</ShipAddress>
  <ShipCity>Aachen</ShipCity>
  <ShipPostalCode>52066</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10364</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-11-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>71.97</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10365</OrderID>
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-11-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>22</Freight>
```

```
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10366</OrderID>
  <CustomerID>GALED</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-11-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>10.14</Freight>
  <ShipName>Galería del gastronómo</ShipName>
  <ShipAddress>Rambla de Cataluña, 23</ShipAddress>
  <ShipCity>Barcelona</ShipCity>
  <ShipPostalCode>8022</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10367</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1996-11-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>13.55</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10368</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-11-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>101.95</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10369</OrderID>
  <CustomerID>SPLIR</CustomerID>
```

```
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-12-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>195.68</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10370</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-12-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.17</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10371</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1996-12-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.45</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10372</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-12-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>890.78</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
```

```
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10373</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-12-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>124.12</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10374</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>3.94</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10375</OrderID>
  <CustomerID>HUNGC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-12-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>20.12</Freight>
  <ShipName>Hungry Coyote Import Store</ShipName>
  <ShipAddress>City Center Plaza 516 Main St.</ShipAddress>
  <ShipCity>Elgin</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97827</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10376</OrderID>
  <CustomerID>MEREP</CustomerID>
```

```
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>20.39</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10377</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>22.21</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10378</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-12-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>5.44</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10379</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-12-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-08T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>45.03</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
```

```
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10380</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-12-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>35.03</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10381</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-12-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>7.99</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10382</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-12-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>94.77</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10383</OrderID>
  <CustomerID>AROUT</CustomerID>
```

```
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-12-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>34.24</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10384</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-12-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>168.64</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10385</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>30.96</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10386</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1996-12-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>13.99</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
```

```
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10387</OrderID>
  <CustomerID>SANTG</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>93.63</Freight>
  <ShipName>Santé Gourmet</ShipName>
  <ShipAddress>Erling Skakkes gate 78</ShipAddress>
  <ShipCity>Stavern</ShipCity>
  <ShipPostalCode>4110</ShipPostalCode>
  <ShipCountry>Norway</ShipCountry>
</Orders>
<Orders>
  <OrderID>10388</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-12-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>34.86</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10389</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1996-12-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>47.42</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10390</OrderID>
```

```
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-12-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>126.38</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10391</OrderID>
  <CustomerID>DRACD</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1996-12-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1996-12-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>5.45</Freight>
  <ShipName>Drachenblut Delikatessen</ShipName>
  <ShipAddress>Walserweg 21</ShipAddress>
  <ShipCity>Aachen</ShipCity>
  <ShipPostalCode>52066</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10392</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-12-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-01T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>122.46</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10393</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>126.56</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
```

```
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10394</OrderID>
  <CustomerID>HUNGC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>30.34</Freight>
  <ShipName>Hungry Coyote Import Store</ShipName>
  <ShipAddress>City Center Plaza 516 Main St.</ShipAddress>
  <ShipCity>Elgin</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97827</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10395</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1996-12-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>184.41</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10396</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1996-12-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>135.35</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10397</OrderID>
```

```
  <CustomerID>PRINI</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1996-12-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>60.26</Freight>
  <ShipName>Princesa Isabel Vinhos</ShipName>
  <ShipAddress>Estrada da saúde n. 58</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1756</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10398</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1996-12-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>89.16</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10399</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1996-12-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>27.36</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10400</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-01-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>83.93</Freight>
  <ShipName>Eastern Connection</ShipName>
```

```
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10401</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-01-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>12.51</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10402</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-01-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>67.88</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10403</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>73.79</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10404</OrderID>
```

```
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-01-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-01-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>155.97</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10405</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-01-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-22T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>34.82</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10406</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-01-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>108.04</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10407</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-01-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>91.48</Freight>
```

```
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10408</OrderID>
  <CustomerID>FOLIG</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-01-08T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>11.26</Freight>
  <ShipName>Folies gourmandes</ShipName>
  <ShipAddress>184, chaussée de Tournai</ShipAddress>
  <ShipCity>Lille</ShipCity>
  <ShipPostalCode>59000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10409</OrderID>
  <CustomerID>OCEAN</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>29.83</Freight>
  <ShipName>Océano Atlántico Ltda.</ShipName>
  <ShipAddress>Ing. Gustavo Moncada 8585 Piso 20-A</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10410</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>2.4</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10411</OrderID>
```

```
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-01-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>23.65</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10412</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-01-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.77</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10413</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>95.66</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10414</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-01-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-17T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>21.48</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
```

```
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10415</OrderID>
  <CustomerID>HUNGC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.2</Freight>
  <ShipName>Hungry Coyote Import Store</ShipName>
  <ShipAddress>City Center Plaza 516 Main St.</ShipAddress>
  <ShipCity>Elgin</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97827</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10416</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-01-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>22.72</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10417</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>70.29</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10418</OrderID>
```

```
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>17.55</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10419</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>137.35</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10420</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>44.12</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10421</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-01-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>99.23</Freight>
  <ShipName>Que Delícia</ShipName>
```

```
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10422</OrderID>
  <CustomerID>FRANS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-01-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>3.02</Freight>
  <ShipName>Franchi S.p.A.</ShipName>
  <ShipAddress>Via Monte Bianco 34</ShipAddress>
  <ShipCity>Torino</ShipCity>
  <ShipPostalCode>10100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10423</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-01-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>24.5</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10424</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-01-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-01-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>370.61</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10425</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-01-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.93</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10426</OrderID>
  <CustomerID>GALED</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>18.69</Freight>
  <ShipName>Galería del gastronómo</ShipName>
  <ShipAddress>Rambla de Cataluña, 23</ShipAddress>
  <ShipCity>Barcelona</ShipCity>
  <ShipPostalCode>8022</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10427</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>31.29</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10428</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-01-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>11.09</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
```

```
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10429</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>56.63</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10430</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>458.78</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10431</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-01-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>44.17</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10432</OrderID>
  <CustomerID>SPLIR</CustomerID>
```

```
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-01-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-02-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.34</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10433</OrderID>
  <CustomerID>PRINI</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>73.83</Freight>
  <ShipName>Princesa Isabel Vinhos</ShipName>
  <ShipAddress>Estrada da saúde n. 58</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1756</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10434</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>17.92</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10435</OrderID>
  <CustomerID>CONSH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>9.21</Freight>
  <ShipName>Consolidated Holdings</ShipName>
  <ShipAddress>Berkeley Gardens 12  Brewery</ShipAddress>
```

```
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX1 6LT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10436</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>156.66</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10437</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>19.97</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10438</OrderID>
  <CustomerID>TOMSP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>8.24</Freight>
  <ShipName>Toms Spezialitäten</ShipName>
  <ShipAddress>Luisenstr. 48</ShipAddress>
  <ShipCity>Münster</ShipCity>
  <ShipPostalCode>44087</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10439</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-02-07T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1997-03-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.07</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10440</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-02-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>86.53</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10441</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>73.02</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10442</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>47.94</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
```

```
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10443</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>13.95</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10444</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>3.5</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10445</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>9.3</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10446</OrderID>
  <CustomerID>TOMSP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-02-14T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1997-03-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>14.68</Freight>
  <ShipName>Toms Spezialitäten</ShipName>
  <ShipAddress>Luisenstr. 48</ShipAddress>
  <ShipCity>Münster</ShipCity>
  <ShipPostalCode>44087</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10447</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-02-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>68.66</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10448</OrderID>
  <CustomerID>RANCH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-02-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>38.82</Freight>
  <ShipName>Rancho grande</ShipName>
  <ShipAddress>Av. del Libertador 900</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10449</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-02-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>53.3</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
```

```
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10450</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.23</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10451</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-02-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>189.09</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10452</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>140.26</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10453</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-02-21T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1997-03-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>25.36</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10454</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-02-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>2.74</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10455</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>180.45</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10456</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-02-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>8.12</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
```

```
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10457</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-02-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>11.57</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10458</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>147.06</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10459</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-02-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-02-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>25.09</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10460</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-02-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-03T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>1</ShipVia>
  <Freight>16.27</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10461</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-02-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>148.61</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10462</OrderID>
  <CustomerID>CONSH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-03-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>6.17</Freight>
  <ShipName>Consolidated Holdings</ShipName>
  <ShipAddress>Berkeley Gardens 12  Brewery</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX1 6LT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10463</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-03-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>14.78</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10464</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-03-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>89</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10465</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-03-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>145.04</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10466</OrderID>
  <CustomerID>COMMI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-03-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>11.93</Freight>
  <ShipName>Comércio Mineiro</ShipName>
  <ShipAddress>Av. dos Lusíadas, 23</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05432-043</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10467</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-03-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>4.93</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10468</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-03-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>44.12</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10469</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-03-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>60.18</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10470</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-03-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>64.56</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10471</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-03-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>45.59</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10472</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-03-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.2</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10473</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-03-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-03-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>16.37</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10474</OrderID>
  <CustomerID>PERIC</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-03-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>83.49</Freight>
```

```
  <ShipName>Pericles Comidas clásicas</ShipName>
  <ShipAddress>Calle Dr. Jorge Cash 321</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10475</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-03-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>68.52</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10476</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-03-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.41</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10477</OrderID>
  <CustomerID>PRINI</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-03-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>13.02</Freight>
  <ShipName>Princesa Isabel Vinhos</ShipName>
  <ShipAddress>Estrada da saúde n. 58</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1756</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10478</OrderID>
```

```
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-03-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-03-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.81</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10479</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-03-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>708.95</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10480</OrderID>
  <CustomerID>FOLIG</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-03-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.35</Freight>
  <ShipName>Folies gourmandes</ShipName>
  <ShipAddress>184, chaussée de Tournai</ShipAddress>
  <ShipCity>Lille</ShipCity>
  <ShipPostalCode>59000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10481</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-03-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>64.33</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
```

```
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10482</OrderID>
  <CustomerID>LAZYK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-03-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>7.48</Freight>
  <ShipName>Lazy K Kountry Store</ShipName>
  <ShipAddress>12 Orchestra Terrace</ShipAddress>
  <ShipCity>Walla Walla</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>99362</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10483</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-03-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>15.28</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10484</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-03-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-01T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>6.88</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10485</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-03-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>64.45</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10486</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-03-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>30.53</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10487</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-03-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-03-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>71.07</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10488</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-03-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-02T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>4.93</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10489</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-03-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>5.29</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10490</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-03-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>210.19</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10491</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-03-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>16.96</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10492</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-04-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>62.89</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10493</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>10.64</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10494</OrderID>
  <CustomerID>COMMI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-04-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>65.99</Freight>
  <ShipName>Comércio Mineiro</ShipName>
  <ShipAddress>Av. dos Lusíadas, 23</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05432-043</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10495</OrderID>
  <CustomerID>LAUGB</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-04-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-05-01T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShippedDate>1997-04-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.65</Freight>
  <ShipName>Laughing Bacchus Wine Cellars</ShipName>
  <ShipAddress>2319 Elm St.</ShipAddress>
  <ShipCity>Vancouver</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>V3F 2K1</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10496</OrderID>
  <CustomerID>TRADH</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-04-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-05-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>46.77</Freight>
  <ShipName>Tradiçao Hipermercados</ShipName>
  <ShipAddress>Av. Inês de Castro, 414</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05634-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10497</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-04-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-05-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>36.21</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10498</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-04-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>29.75</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
```

```
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10499</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>102.02</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10500</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-04-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>42.68</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10501</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-04-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>8.85</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10502</OrderID>
  <CustomerID>PERIC</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-04-10T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1997-05-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>69.32</Freight>
  <ShipName>Pericles Comidas clásicas</ShipName>
  <ShipAddress>Calle Dr. Jorge Cash 321</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10503</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-04-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>16.74</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10504</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>59.13</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10505</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-04-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>7.13</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
```

```
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10506</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-04-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>21.19</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10507</OrderID>
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-04-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>47.45</Freight>
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10508</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-04-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.99</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10509</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-15T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShippedDate>1997-04-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.15</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10510</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-04-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>367.63</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10511</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>350.64</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10512</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-04-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.53</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
```

```
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10513</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-04-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>105.65</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10514</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-04-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>789.95</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10515</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-04-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>204.47</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10516</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-04-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-01T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>62.78</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10517</OrderID>
  <CustomerID>NORTS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-04-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-04-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>32.07</Freight>
  <ShipName>North/South</ShipName>
  <ShipAddress>South House 300 Queensbridge</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>SW7 1RZ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10518</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>218.15</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10519</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-04-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>91.76</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10520</OrderID>
  <CustomerID>SANTG</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-04-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>13.37</Freight>
  <ShipName>Santé Gourmet</ShipName>
  <ShipAddress>Erling Skakkes gate 78</ShipAddress>
  <ShipCity>Stavern</ShipCity>
  <ShipPostalCode>4110</ShipPostalCode>
  <ShipCountry>Norway</ShipCountry>
</Orders>
<Orders>
  <OrderID>10521</OrderID>
  <CustomerID>CACTU</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-04-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>17.22</Freight>
  <ShipName>Cactus Comidas para llevar</ShipName>
  <ShipAddress>Cerrito 333</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10522</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-04-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>45.33</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10523</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-05-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>77.63</Freight>
  <ShipName>Seven Seas Imports</ShipName>
```

```
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10524</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-05-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>244.79</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10525</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-05-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>11.06</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10526</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-05-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>58.59</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10527</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>7</EmployeeID>
```

```
  <OrderDate>1997-05-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>41.9</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10528</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-05-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.35</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10529</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-05-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>66.69</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10530</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-05-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>339.22</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
```

```
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10531</OrderID>
  <CustomerID>OCEAN</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-05-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.12</Freight>
  <ShipName>Océano Atlántico Ltda.</ShipName>
  <ShipAddress>Ing. Gustavo Moncada 8585 Piso 20-A</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10532</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-05-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>74.46</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10533</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-05-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>188.04</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10534</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-05-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-09T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShippedDate>1997-05-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>27.94</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10535</OrderID>
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-05-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>15.64</Freight>
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10536</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-05-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>58.88</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10537</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-05-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-05-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>78.85</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10538</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-05-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.87</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10539</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-05-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>12.36</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10540</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-05-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1007.64</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10541</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-05-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
```

```
  <Freight>68.65</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10542</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-05-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>10.95</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10543</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-05-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>48.17</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10544</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-05-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>24.91</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10545</OrderID>
  <CustomerID>LAZYK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-05-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>11.92</Freight>
  <ShipName>Lazy K Kountry Store</ShipName>
  <ShipAddress>12 Orchestra Terrace</ShipAddress>
  <ShipCity>Walla Walla</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>99362</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10546</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-05-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>194.72</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10547</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-05-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>178.43</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10548</OrderID>
  <CustomerID>TOMSP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-05-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-02T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>1.43</Freight>
  <ShipName>Toms Spezialitäten</ShipName>
  <ShipAddress>Luisenstr. 48</ShipAddress>
  <ShipCity>Münster</ShipCity>
  <ShipPostalCode>44087</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10549</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-05-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-05-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>171.24</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10550</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-05-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.32</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10551</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-05-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>72.95</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10552</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-05-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>83.22</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10553</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-05-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>149.49</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10554</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-05-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>120.97</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10555</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-06-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>252.49</Freight>
```

```
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10556</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-06-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>9.8</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10557</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-06-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-06-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>96.72</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10558</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-06-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>72.97</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10559</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-06-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.05</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10560</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-06-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>36.65</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10561</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-06-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>242.21</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10562</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-06-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>22.95</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
```

```
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10563</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-06-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>60.43</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10564</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-06-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>13.75</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10565</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-06-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.15</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10566</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-06-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>88.4</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10567</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-06-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>33.97</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10568</OrderID>
  <CustomerID>GALED</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-06-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>6.54</Freight>
  <ShipName>Galería del gastronómo</ShipName>
  <ShipAddress>Rambla de Cataluña, 23</ShipAddress>
  <ShipCity>Barcelona</ShipCity>
  <ShipPostalCode>8022</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10569</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-06-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>58.98</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
```

```
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10570</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-06-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>188.99</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10571</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-06-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>26.06</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10572</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-06-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>116.43</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10573</OrderID>
```

```
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-06-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>84.84</Freight>
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10574</OrderID>
  <CustomerID>TRAIH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-06-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>37.6</Freight>
  <ShipName>Trail's Head Gourmet Provisioners</ShipName>
  <ShipAddress>722 DaVinci Blvd.</ShipAddress>
  <ShipCity>Kirkland</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98034</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10575</OrderID>
  <CustomerID>MORGK</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-06-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>127.34</Freight>
  <ShipName>Morgenstern Gesundkost</ShipName>
  <ShipAddress>Heerstr. 22</ShipAddress>
  <ShipCity>Leipzig</ShipCity>
  <ShipPostalCode>04179</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10576</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-06-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>18.56</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
```

```
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10577</OrderID>
  <CustomerID>TRAIH</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-06-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-06-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>25.41</Freight>
  <ShipName>Trail's Head Gourmet Provisioners</ShipName>
  <ShipAddress>722 DaVinci Blvd.</ShipAddress>
  <ShipCity>Kirkland</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98034</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10578</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-06-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>29.6</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10579</OrderID>
  <CustomerID>LETSS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-06-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>13.73</Freight>
  <ShipName>Let's Stop N Shop</ShipName>
  <ShipAddress>87 Polk St. Suite 5</ShipAddress>
  <ShipCity>San Francisco</ShipCity>
  <ShipRegion>CA</ShipRegion>
  <ShipPostalCode>94117</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10580</OrderID>
```

```
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-06-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>75.89</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10581</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-06-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>3.01</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10582</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-06-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>27.71</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10583</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-06-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.28</Freight>
  <ShipName>Wartian Herkku</ShipName>
```

```
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10584</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-06-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>59.14</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10585</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-07-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>13.41</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10586</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-07-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.48</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10587</OrderID>
```

```
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>62.52</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10588</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-07-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-31T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>194.67</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10589</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-07-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.42</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10590</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-07-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>44.77</Freight>
```

```
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10591</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-07-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>55.92</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10592</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-07-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>32.1</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10593</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-07-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>174.2</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10594</OrderID>
```

```
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-07-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>5.24</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10595</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-07-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>96.78</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10596</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-07-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>16.34</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10597</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-07-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>35.12</Freight>
```

```
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10598</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>44.42</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10599</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-07-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>29.98</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10600</OrderID>
  <CustomerID>HUNGC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-07-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>45.13</Freight>
  <ShipName>Hungry Coyote Import Store</ShipName>
  <ShipAddress>City Center Plaza 516 Main St.</ShipAddress>
  <ShipCity>Elgin</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97827</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10601</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-07-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>58.3</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10602</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-07-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>2.92</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10603</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-07-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>48.77</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10604</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
```

```
  <Freight>7.46</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10605</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>379.13</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10606</OrderID>
  <CustomerID>TRADH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-07-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-31T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>79.4</Freight>
  <ShipName>Tradiçao Hipermercados</ShipName>
  <ShipAddress>Av. Inês de Castro, 414</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05634-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10607</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-07-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-25T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>200.24</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10608</OrderID>
  <CustomerID>TOMSP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-07-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>27.79</Freight>
  <ShipName>Toms Spezialitäten</ShipName>
  <ShipAddress>Luisenstr. 48</ShipAddress>
  <ShipCity>Münster</ShipCity>
  <ShipPostalCode>44087</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10609</OrderID>
  <CustomerID>DUMON</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-07-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-07-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.85</Freight>
  <ShipName>Du monde entier</ShipName>
  <ShipAddress>67, rue des Cinquante Otages</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10610</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-07-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>26.78</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10611</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-07-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>80.65</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10612</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>544.08</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10613</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-07-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>8.11</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10614</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-07-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.93</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10615</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-07-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>0.75</Freight>
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10616</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-07-31T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>116.53</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10617</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-07-31T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-08-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>18.53</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10618</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-08-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-08T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>1</ShipVia>
  <Freight>154.68</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10619</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-08-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>91.05</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10620</OrderID>
  <CustomerID>LAUGB</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-08-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>0.94</Freight>
  <ShipName>Laughing Bacchus Wine Cellars</ShipName>
  <ShipAddress>2319 Elm St.</ShipAddress>
  <ShipCity>Vancouver</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>V3F 2K1</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10621</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>23.73</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
```

```
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10622</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>50.97</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10623</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-08-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-12T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>97.18</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10624</OrderID>
  <CustomerID>THECR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>94.8</Freight>
  <ShipName>The Cracker Box</ShipName>
  <ShipAddress>55 Grizzly Peak Rd.</ShipAddress>
  <ShipCity>Butte</ShipCity>
  <ShipRegion>MT</ShipRegion>
  <ShipPostalCode>59801</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10625</OrderID>
  <CustomerID>ANATR</CustomerID>
  <EmployeeID>3</EmployeeID>
```

```
  <OrderDate>1997-08-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>43.9</Freight>
  <ShipName>Ana Trujillo Emparedados y helados</ShipName>
  <ShipAddress>Avda. de la Constitución 2222</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05021</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10626</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-08-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>138.69</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10627</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-08-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>107.46</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10628</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>30.36</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
```

```
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10629</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>85.46</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10630</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-08-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>32.35</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10631</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-08-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.87</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10632</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-08-14T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1997-09-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>41.38</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10633</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-08-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>477.9</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10634</OrderID>
  <CustomerID>FOLIG</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>487.38</Freight>
  <ShipName>Folies gourmandes</ShipName>
  <ShipAddress>184, chaussée de Tournai</ShipAddress>
  <ShipCity>Lille</ShipCity>
  <ShipPostalCode>59000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10635</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-08-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>47.46</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10636</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.15</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10637</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-08-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>201.29</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10638</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-08-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>158.44</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10639</OrderID>
  <CustomerID>SANTG</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-08-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-17T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShippedDate>1997-08-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>38.64</Freight>
  <ShipName>Santé Gourmet</ShipName>
  <ShipAddress>Erling Skakkes gate 78</ShipAddress>
  <ShipCity>Stavern</ShipCity>
  <ShipPostalCode>4110</ShipPostalCode>
  <ShipCountry>Norway</ShipCountry>
</Orders>
<Orders>
  <OrderID>10640</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>23.55</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10641</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>179.61</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10642</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-08-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>41.89</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10643</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-08-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>29.46</Freight>
  <ShipName>Alfreds Futterkiste</ShipName>
  <ShipAddress>Obere Str. 57</ShipAddress>
  <ShipCity>Berlin</ShipCity>
  <ShipPostalCode>12209</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10644</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-08-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>0.14</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10645</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-02T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>12.41</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10646</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-08-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-08T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShippedDate>1997-09-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>142.33</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10647</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-08-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>45.54</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10648</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-08-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>14.25</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10649</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-08-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-08-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>6.2</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
```

```
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10650</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-08-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>176.81</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10651</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-09-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>20.6</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10652</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-09-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.14</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10653</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-02T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1997-09-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>93.25</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10654</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-09-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>55.26</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10655</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-11T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.41</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10656</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-09-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>57.15</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
```

```
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10657</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-02T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>352.69</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10658</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-09-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>364.15</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10659</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-09-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-03T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>105.81</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10660</OrderID>
  <CustomerID>HUNGC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-09-08T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1997-10-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>111.29</Freight>
  <ShipName>Hungry Coyote Import Store</ShipName>
  <ShipAddress>City Center Plaza 516 Main St.</ShipAddress>
  <ShipCity>Elgin</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97827</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10661</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-09-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>17.55</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10662</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-09-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.28</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10663</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-09-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>113.15</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
```

```
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10664</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.27</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10665</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-11T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>26.31</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10666</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-09-12T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>232.42</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10667</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-09-12T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1997-10-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>78.09</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10668</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>47.22</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10669</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>24.39</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10670</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-09-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-18T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>203.48</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10671</OrderID>
  <CustomerID>FRANR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>30.34</Freight>
  <ShipName>France restauration</ShipName>
  <ShipAddress>54, rue Royale</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10672</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-09-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>95.75</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10673</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-19T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>22.76</Freight>
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10674</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-09-18T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>0.9</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10675</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-09-19T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>31.85</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10676</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>2.01</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10677</OrderID>
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.03</Freight>
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10678</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-09-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>388.98</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10679</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-09-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>27.94</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10680</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-09-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-26T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>26.61</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10681</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-09-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-30T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>76.13</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10682</OrderID>
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-09-25T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>36.13</Freight>
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10683</OrderID>
  <CustomerID>DUMON</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.4</Freight>
  <ShipName>Du monde entier</ShipName>
  <ShipAddress>67, rue des Cinquante Otages</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10684</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-09-26T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-09-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>145.63</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10685</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-09-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>33.75</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10686</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-09-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>96.5</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10687</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-09-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>296.43</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10688</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>299.09</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10689</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>13.42</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10690</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-02T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-03T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>15.8</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10691</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-10-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>810.05</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10692</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-03T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>61.02</Freight>
  <ShipName>Alfred's Futterkiste</ShipName>
  <ShipAddress>Obere Str. 57</ShipAddress>
  <ShipCity>Berlin</ShipCity>
  <ShipPostalCode>12209</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10693</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1997-10-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>139.34</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10694</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>398.36</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10695</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-10-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>16.72</Freight>
```

```
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10696</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>102.55</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10697</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>45.52</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10698</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>272.47</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10699</OrderID>
  <CustomerID>MORGK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>0.58</Freight>
  <ShipName>Morgenstern Gesundkost</ShipName>
  <ShipAddress>Heerstr. 22</ShipAddress>
  <ShipCity>Leipzig</ShipCity>
  <ShipPostalCode>04179</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10700</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>65.1</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10701</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-10-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>220.31</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10702</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>23.94</Freight>
```

```
  <ShipName>Alfred's Futterkiste</ShipName>
  <ShipAddress>Obere Str. 57</ShipAddress>
  <ShipCity>Berlin</ShipCity>
  <ShipPostalCode>12209</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10703</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-10-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>152.3</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10704</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-10-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.78</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10705</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-10-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.52</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10706</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>135.63</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10707</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-10-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>21.74</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10708</OrderID>
  <CustomerID>THEBI</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-10-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>2.96</Freight>
  <ShipName>The Big Cheese</ShipName>
  <ShipAddress>89 Jefferson Way Suite 2</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97201</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10709</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-20T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>210.8</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10710</OrderID>
  <CustomerID>FRANS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.98</Freight>
  <ShipName>Franchi S.p.A.</ShipName>
  <ShipAddress>Via Monte Bianco 34</ShipAddress>
  <ShipCity>Torino</ShipCity>
  <ShipPostalCode>10100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10711</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-10-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-12-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>52.41</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10712</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>89.93</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10713</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>167.05</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10714</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-10-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>24.49</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10715</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>63.2</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10716</OrderID>
  <CustomerID>RANCH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-21T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1997-10-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>22.57</Freight>
  <ShipName>Rancho grande</ShipName>
  <ShipAddress>Av. del Libertador 900</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10717</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1997-11-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>59.25</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10718</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-10-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>170.88</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10719</OrderID>
  <CustomerID>LETSS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>51.44</Freight>
  <ShipName>Let's Stop N Shop</ShipName>
  <ShipAddress>87 Polk St. Suite 5</ShipAddress>
  <ShipCity>San Francisco</ShipCity>
  <ShipRegion>CA</ShipRegion>
  <ShipPostalCode>94117</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10720</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>9.53</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10721</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-10-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-10-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>48.92</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10722</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>74.58</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10723</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-10-30T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1997-11-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>21.72</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10724</OrderID>
  <CustomerID>MEREP</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-10-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>57.75</Freight>
  <ShipName>Mère Paillarde</ShipName>
  <ShipAddress>43 rue St. Laurent</ShipAddress>
  <ShipCity>Montréal</ShipCity>
  <ShipRegion>Québec</ShipRegion>
  <ShipPostalCode>H1J 1C3</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10725</OrderID>
  <CustomerID>FAMIA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-10-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>10.83</Freight>
  <ShipName>Familia Arquibaldo</ShipName>
  <ShipAddress>Rua Orós, 92</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05442-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10726</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-11-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>16.56</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
```

```
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10727</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-11-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>89.9</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10728</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-11-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>58.33</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10729</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-11-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>141.06</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10730</OrderID>
  <CustomerID>BONAP</CustomerID>
```

```
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-11-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>20.12</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10731</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-11-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>96.65</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10732</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>16.97</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10733</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-11-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>110.11</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
```

```
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10734</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-11-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.63</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10735</OrderID>
  <CustomerID>LETSS</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-11-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-08T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>45.97</Freight>
  <ShipName>Let's Stop N Shop</ShipName>
  <ShipAddress>87 Polk St. Suite 5</ShipAddress>
  <ShipCity>San Francisco</ShipCity>
  <ShipRegion>CA</ShipRegion>
  <ShipPostalCode>94117</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10736</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-11-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>44.1</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10737</OrderID>
  <CustomerID>VINET</CustomerID>
  <EmployeeID>2</EmployeeID>
```

```
  <OrderDate>1997-11-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.79</Freight>
  <ShipName>Vins et alcools Chevalier</ShipName>
  <ShipAddress>59 rue de l'Abbaye</ShipAddress>
  <ShipCity>Reims</ShipCity>
  <ShipPostalCode>51100</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10738</OrderID>
  <CustomerID>SPECD</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-11-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>2.91</Freight>
  <ShipName>Spécialités du monde</ShipName>
  <ShipAddress>25, rue Lauriston</ShipAddress>
  <ShipCity>Paris</ShipCity>
  <ShipPostalCode>75016</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10739</OrderID>
  <CustomerID>VINET</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-17T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>11.08</Freight>
  <ShipName>Vins et alcools Chevalier</ShipName>
  <ShipAddress>59 rue de l'Abbaye</ShipAddress>
  <ShipCity>Reims</ShipCity>
  <ShipPostalCode>51100</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10740</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-11-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>81.88</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
```

```
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10741</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-11-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-11-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>10.96</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10742</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>243.73</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10743</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-11-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>23.72</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10744</OrderID>
  <CustomerID>VAFFE</CustomerID>
```

```
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-11-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>69.19</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10745</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-11-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>3.52</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10746</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-11-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>31.43</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10747</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-11-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>117.33</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
```

```
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10748</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>232.55</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10749</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-11-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>61.53</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10750</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-11-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>79.3</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10751</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>3</EmployeeID>
```

```
  <OrderDate>1997-11-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>130.79</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10752</OrderID>
  <CustomerID>NORTS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-11-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.39</Freight>
  <ShipName>North/South</ShipName>
  <ShipAddress>South House 300 Queensbridge</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>SW7 1RZ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10753</OrderID>
  <CustomerID>FRANS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>7.7</Freight>
  <ShipName>Franchi S.p.A.</ShipName>
  <ShipAddress>Via Monte Bianco 34</ShipAddress>
  <ShipCity>Torino</ShipCity>
  <ShipPostalCode>10100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10754</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-11-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>2.38</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
```

```
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10755</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-11-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-11-28T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>16.71</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10756</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-11-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>73.21</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10757</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-11-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.19</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10758</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-28T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1997-12-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>138.17</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10759</OrderID>
  <CustomerID>ANATR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-11-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>11.99</Freight>
  <ShipName>Ana Trujillo Emparedados y helados</ShipName>
  <ShipAddress>Avda. de la Constitución 2222</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05021</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10760</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>155.64</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10761</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1997-12-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>18.66</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
```

```
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10762</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>328.74</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10763</OrderID>
  <CustomerID>FOLIG</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>37.35</Freight>
  <ShipName>Folies gourmandes</ShipName>
  <ShipAddress>184, chaussée de Tournai</ShipAddress>
  <ShipCity>Lille</ShipCity>
  <ShipPostalCode>59000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10764</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-12-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>145.45</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10765</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-09T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>42.74</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10766</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>157.55</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10767</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.59</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10768</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-08T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>146.32</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10769</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-08T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>65.06</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10770</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-12-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-17T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>5.32</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10771</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-12-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>11.19</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10772</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-07T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-19T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>91.28</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10773</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-08T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>96.43</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10774</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>48.2</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10775</OrderID>
  <CustomerID>THECR</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-12-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>20.25</Freight>
  <ShipName>The Cracker Box</ShipName>
  <ShipAddress>55 Grizzly Peak Rd.</ShipAddress>
  <ShipCity>Butte</ShipCity>
  <ShipRegion>MT</ShipRegion>
  <ShipPostalCode>59801</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10776</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>351.53</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10777</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-12-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.01</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10778</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>6.79</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10779</OrderID>
  <CustomerID>MORGK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>58.13</Freight>
  <ShipName>Morgenstern Gesundkost</ShipName>
  <ShipAddress>Heerstr. 22</ShipAddress>
  <ShipCity>Leipzig</ShipCity>
  <ShipPostalCode>04179</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10780</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-12-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1997-12-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>42.13</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10781</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-12-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>73.16</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10782</OrderID>
  <CustomerID>CACTU</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-12-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-14T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-22T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.1</Freight>
  <ShipName>Cactus Comidas para llevar</ShipName>
  <ShipAddress>Cerrito 333</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10783</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>124.98</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10784</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-22T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>70.09</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10785</OrderID>
  <CustomerID>GROSR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-15T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.51</Freight>
  <ShipName>GROSELLA-Restaurante</ShipName>
  <ShipAddress>5ª Ave. Los Palos Grandes</ShipAddress>
  <ShipCity>Caracas</ShipCity>
  <ShipRegion>DF</ShipRegion>
  <ShipPostalCode>1081</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10786</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-12-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
```

```
  <Freight>110.87</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10787</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-12-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>249.93</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10788</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>42.7</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10789</OrderID>
  <CustomerID>FOLIG</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>100.6</Freight>
  <ShipName>Folies gourmandes</ShipName>
  <ShipAddress>184, chaussée de Tournai</ShipAddress>
  <ShipCity>Lille</ShipCity>
  <ShipPostalCode>59000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10790</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-12-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>28.23</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10791</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-12-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-01T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>16.85</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10792</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>23.79</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10793</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.52</Freight>
```

```
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10794</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-12-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>21.49</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10795</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1997-12-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-21T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>126.66</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10796</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>26.52</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10797</OrderID>
  <CustomerID>DRACD</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1997-12-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-22T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>33.35</Freight>
  <ShipName>Drachenblut Delikatessen</ShipName>
  <ShipAddress>Walserweg 21</ShipAddress>
  <ShipCity>Aachen</ShipCity>
  <ShipPostalCode>52066</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10798</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-12-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>2.33</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10799</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1997-12-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>30.76</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10800</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-12-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
```

```
  <Freight>137.44</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10801</OrderID>
  <CustomerID>BOLID</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1997-12-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>97.09</Freight>
  <ShipName>Bólido Comidas preparadas</ShipName>
  <ShipAddress>C/ Araquil, 67</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28023</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10802</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>257.26</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10803</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>55.23</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10804</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1997-12-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>27.33</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10805</OrderID>
  <CustomerID>THEBI</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1997-12-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>237.34</Freight>
  <ShipName>The Big Cheese</ShipName>
  <ShipAddress>89 Jefferson Way Suite 2</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97201</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10806</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1997-12-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>22.11</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10807</OrderID>
  <CustomerID>FRANS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1997-12-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-28T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.36</Freight>
```

```
  <ShipName>Franchi S.p.A.</ShipName>
  <ShipAddress>Via Monte Bianco 34</ShipAddress>
  <ShipCity>Torino</ShipCity>
  <ShipPostalCode>10100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10808</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>45.53</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10809</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-01-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.87</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10810</OrderID>
  <CustomerID>LAUGB</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-29T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-07T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.33</Freight>
  <ShipName>Laughing Bacchus Wine Cellars</ShipName>
  <ShipAddress>2319 Elm St.</ShipAddress>
  <ShipCity>Vancouver</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>V3F 2K1</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10811</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-08T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>31.22</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10812</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-01-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>59.78</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10813</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>47.38</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10814</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-02T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1998-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>130.94</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10815</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>14.62</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10816</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-01-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>719.78</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10817</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>306.07</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
```

```
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10818</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-01-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>65.48</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10819</OrderID>
  <CustomerID>CACTU</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>19.76</Freight>
  <ShipName>Cactus Comidas para llevar</ShipName>
  <ShipAddress>Cerrito 333</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10820</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-07T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>37.52</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10821</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-08T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-05T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1998-01-15T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>36.68</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10822</OrderID>
  <CustomerID>TRAIH</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-01-08T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>7</Freight>
  <ShipName>Trail's Head Gourmet Provisioners</ShipName>
  <ShipAddress>722 DaVinci Blvd.</ShipAddress>
  <ShipCity>Kirkland</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98034</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10823</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-01-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>163.97</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10824</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.23</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
```

```
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10825</OrderID>
  <CustomerID>DRACD</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-14T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>79.25</Freight>
  <ShipName>Drachenblut Delikatessen</ShipName>
  <ShipAddress>Walserweg 21</ShipAddress>
  <ShipCity>Aachen</ShipCity>
  <ShipPostalCode>52066</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10826</OrderID>
  <CustomerID>BLONP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-01-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>7.09</Freight>
  <ShipName>Blondel père et fils</ShipName>
  <ShipAddress>24, place Kléber</ShipAddress>
  <ShipCity>Strasbourg</ShipCity>
  <ShipPostalCode>67000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10827</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>63.54</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10828</OrderID>
  <CustomerID>RANCH</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-01-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-01-27T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1998-02-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>90.85</Freight>
  <ShipName>Rancho grande</ShipName>
  <ShipAddress>Av. del Libertador 900</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10829</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-01-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>154.72</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10830</OrderID>
  <CustomerID>TRADH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-01-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>81.83</Freight>
  <ShipName>Tradiçao Hipermercados</ShipName>
  <ShipAddress>Av. Inês de Castro, 414</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05634-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10831</OrderID>
  <CustomerID>SANTG</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>72.19</Freight>
  <ShipName>Santé Gourmet</ShipName>
  <ShipAddress>Erling Skakkes gate 78</ShipAddress>
  <ShipCity>Stavern</ShipCity>
  <ShipPostalCode>4110</ShipPostalCode>
```

```
  <ShipCountry>Norway</ShipCountry>
</Orders>
<Orders>
  <OrderID>10832</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-14T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>43.26</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10833</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-01-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>71.49</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10834</OrderID>
  <CustomerID>TRADH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>29.78</Freight>
  <ShipName>Tradiçao Hipermercados</ShipName>
  <ShipAddress>Av. Inês de Castro, 414</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05634-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10835</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-15T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-12T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1998-01-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>69.53</Freight>
  <ShipName>Alfred's Futterkiste</ShipName>
  <ShipAddress>Obere Str. 57</ShipAddress>
  <ShipCity>Berlin</ShipCity>
  <ShipPostalCode>12209</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10836</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-01-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-21T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>411.88</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10837</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-01-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>13.32</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10838</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>59.28</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10839</OrderID>
  <CustomerID>TRADH</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-22T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>35.43</Freight>
  <ShipName>Tradiçao Hipermercados</ShipName>
  <ShipAddress>Av. Inês de Castro, 414</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05634-030</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10840</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-01-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>2.71</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10841</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-01-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>424.3</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10842</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-17T00:00:00.0000000-08:00</RequiredDate>
```

```
  <ShippedDate>1998-01-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>54.42</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10843</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-01-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>9.26</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10844</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>25.22</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10845</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-21T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>212.98</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10846</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>56.46</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10847</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-01-22T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>487.57</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10848</OrderID>
  <CustomerID>CONSH</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-01-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-29T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>38.24</Freight>
  <ShipName>Consolidated Holdings</ShipName>
  <ShipAddress>Berkeley Gardens 12  Brewery</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX1 6LT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10849</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-01-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>0.56</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10850</OrderID>
  <CustomerID>VICTE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>49.19</Freight>
  <ShipName>Victuailles en stock</ShipName>
  <ShipAddress>2, rue du Commerce</ShipAddress>
  <ShipCity>Lyon</ShipCity>
  <ShipPostalCode>69004</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10851</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-01-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>160.55</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10852</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-01-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>174.05</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10853</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-01-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>53.83</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10854</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>100.22</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10855</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>170.97</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10856</OrderID>
  <CustomerID>ANTON</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>58.43</Freight>
  <ShipName>Antonio Moreno Taquería</ShipName>
  <ShipAddress>Mataderos  2312</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05023</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10857</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-28T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>188.85</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10858</OrderID>
  <CustomerID>LACOR</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-01-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>52.51</Freight>
  <ShipName>La corne d'abondance</ShipName>
  <ShipAddress>67, avenue de l'Europe</ShipAddress>
  <ShipCity>Versailles</ShipCity>
  <ShipPostalCode>78000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10859</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-01-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>76.1</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10860</OrderID>
  <CustomerID>FRANR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-01-29T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>19.26</Freight>
  <ShipName>France restauration</ShipName>
  <ShipAddress>54, rue Royale</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10861</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-01-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-17T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>14.93</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10862</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-01-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>53.23</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10863</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-17T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>30.26</Freight>
```

```
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10864</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.04</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10865</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-02-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-02-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>348.14</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10866</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-02-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>109.11</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
```

```
  <OrderID>10867</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-02-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.93</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10868</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-02-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>191.27</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10869</OrderID>
  <CustomerID>SEVES</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-02-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>143.28</Freight>
  <ShipName>Seven Seas Imports</ShipName>
  <ShipAddress>90 Wadhurst Rd.</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>OX15 4NB</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10870</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-02-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-04T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
```

```
  <Freight>12.04</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10871</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-02-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>112.27</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10872</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-02-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-05T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>175.32</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10873</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.82</Freight>
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10874</OrderID>
```

```
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-02-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>19.58</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10875</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-06T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>32.37</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10876</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-02-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>60.42</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10877</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-09T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>38.06</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
```

```
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10878</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>46.69</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10879</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-02-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-10T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>8.5</Freight>
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10880</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-02-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>88.01</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10881</OrderID>
  <CustomerID>CACTU</CustomerID>
  <EmployeeID>4</EmployeeID>
```

```
  <OrderDate>1998-02-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>2.84</Freight>
  <ShipName>Cactus Comidas para llevar</ShipName>
  <ShipAddress>Cerrito 333</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10882</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>23.1</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10883</OrderID>
  <CustomerID>LONEP</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-02-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>0.53</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10884</OrderID>
  <CustomerID>LETSS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>90.97</Freight>
  <ShipName>Let's Stop N Shop</ShipName>
  <ShipAddress>87 Polk St. Suite 5</ShipAddress>
```

```
  <ShipCity>San Francisco</ShipCity>
  <ShipRegion>CA</ShipRegion>
  <ShipPostalCode>94117</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10885</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-02-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-12T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>5.64</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10886</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>4.99</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10887</OrderID>
  <CustomerID>GALED</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-02-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-13T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.25</Freight>
  <ShipName>Galería del gastronómo</ShipName>
  <ShipAddress>Rambla de Cataluña, 23</ShipAddress>
  <ShipCity>Barcelona</ShipCity>
  <ShipPostalCode>8022</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10888</OrderID>
  <CustomerID>GODOS</CustomerID>
```

```
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>51.87</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10889</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-02-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>280.61</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10890</OrderID>
  <CustomerID>DUMON</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-02-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-16T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>32.76</Freight>
  <ShipName>Du monde entier</ShipName>
  <ShipAddress>67, rue des Cinquante Otages</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10891</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-02-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>20.37</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
```

```
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10892</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-17T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>120.27</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10893</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-02-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>77.78</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10894</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>116.13</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10895</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>3</EmployeeID>
```

```
  <OrderDate>1998-02-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-18T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>162.75</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10896</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-02-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>32.45</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10897</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-02-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-19T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>603.54</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10898</OrderID>
  <CustomerID>OCEAN</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.27</Freight>
  <ShipName>Océano Atlántico Ltda.</ShipName>
  <ShipAddress>Ing. Gustavo Moncada 8585 Piso 20-A</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
```

```
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10899</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-02-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>1.21</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10900</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.66</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10901</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>62.09</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10902</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>1</EmployeeID>
```

```
  <OrderDate>1998-02-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-23T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>44.15</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10903</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-02-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>36.71</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10904</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-02-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>162.95</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10905</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-02-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>13.72</Freight>
  <ShipName>Wellington Importadora</ShipName>
```

```
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10906</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-11T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>26.29</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10907</OrderID>
  <CustomerID>SPECD</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-02-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-25T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-02-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>9.19</Freight>
  <ShipName>Spécialités du monde</ShipName>
  <ShipAddress>25, rue Lauriston</ShipAddress>
  <ShipCity>Paris</ShipCity>
  <ShipPostalCode>75016</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10908</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-06T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>32.96</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10909</OrderID>
  <CustomerID>SANTG</CustomerID>
```

```
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-10T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>53.05</Freight>
  <ShipName>Santé Gourmet</ShipName>
  <ShipAddress>Erling Skakkes gate 78</ShipAddress>
  <ShipCity>Stavern</ShipCity>
  <ShipPostalCode>4110</ShipPostalCode>
  <ShipCountry>Norway</ShipCountry>
</Orders>
<Orders>
  <OrderID>10910</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>38.11</Freight>
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10911</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>38.19</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10912</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>580.91</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
```

```
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10913</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-02-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>33.05</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10914</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-02-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>21.19</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10915</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-02-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.51</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10916</OrderID>
  <CustomerID>RANCH</CustomerID>
  <EmployeeID>1</EmployeeID>
```

```
  <OrderDate>1998-02-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-27T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>63.77</Freight>
  <ShipName>Rancho grande</ShipName>
  <ShipAddress>Av. del Libertador 900</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10917</OrderID>
  <CustomerID>ROMEY</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>8.29</Freight>
  <ShipName>Romero y tomillo</ShipName>
  <ShipAddress>Gran Vía, 1</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28001</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10918</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>48.83</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10919</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-04T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>19.8</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
```

```
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10920</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>29.61</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10921</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-09T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>176.48</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10922</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-03-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-31T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-05T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>62.74</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10923</OrderID>
  <CustomerID>LAMAI</CustomerID>
```

```
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-03-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>68.26</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10924</OrderID>
  <CustomerID>BERGS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>151.52</Freight>
  <ShipName>Berglunds snabbköp</ShipName>
  <ShipAddress>Berguvsvägen  8</ShipAddress>
  <ShipCity>Luleå</ShipCity>
  <ShipPostalCode>S-958 22</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10925</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>2.27</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10926</OrderID>
  <CustomerID>ANATR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-04T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-01T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-11T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>39.92</Freight>
  <ShipName>Ana Trujillo Emparedados y helados</ShipName>
  <ShipAddress>Avda. de la Constitución 2222</ShipAddress>
```

```
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05021</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10927</OrderID>
  <CustomerID>LACOR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>19.79</Freight>
  <ShipName>La corne d'abondance</ShipName>
  <ShipAddress>67, avenue de l'Europe</ShipAddress>
  <ShipCity>Versailles</ShipCity>
  <ShipPostalCode>78000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10928</OrderID>
  <CustomerID>GALED</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.36</Freight>
  <ShipName>Galería del gastronómo</ShipName>
  <ShipAddress>Rambla de Cataluña, 23</ShipAddress>
  <ShipCity>Barcelona</ShipCity>
  <ShipPostalCode>8022</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10929</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-05T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>33.93</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10930</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-06T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1998-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>15.55</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10931</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-20T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>13.6</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10932</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>134.64</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10933</OrderID>
  <CustomerID>ISLAT</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-06T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-03T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>54.15</Freight>
  <ShipName>Island Trading</ShipName>
  <ShipAddress>Garden House Crowther Way</ShipAddress>
  <ShipCity>Cowes</ShipCity>
  <ShipRegion>Isle of Wight</ShipRegion>
  <ShipPostalCode>PO31 7PJ</ShipPostalCode>
```

```
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10934</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-12T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>32.01</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10935</OrderID>
  <CustomerID>WELLI</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>47.59</Freight>
  <ShipName>Wellington Importadora</ShipName>
  <ShipAddress>Rua do Mercado, 12</ShipAddress>
  <ShipCity>Resende</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>08737-363</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10936</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-09T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>33.68</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10937</OrderID>
  <CustomerID>CACTU</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-03-10T00:00:00.0000000-08:00</OrderDate>
```

```
  <RequiredDate>1998-03-24T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>31.51</Freight>
  <ShipName>Cactus Comidas para llevar</ShipName>
  <ShipAddress>Cerrito 333</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10938</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>31.89</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10939</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-10T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-13T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>76.33</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10940</OrderID>
  <CustomerID>BONAP</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>19.77</Freight>
  <ShipName>Bon app'</ShipName>
  <ShipAddress>12, rue des Bouchers</ShipAddress>
  <ShipCity>Marseille</ShipCity>
  <ShipPostalCode>13008</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10941</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-03-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>400.81</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10942</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-03-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>17.95</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10943</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-11T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>2.17</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10944</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-26T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-13T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>52.92</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10945</OrderID>
  <CustomerID>MORGK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-18T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>10.22</Freight>
  <ShipName>Morgenstern Gesundkost</ShipName>
  <ShipAddress>Heerstr. 22</ShipAddress>
  <ShipCity>Leipzig</ShipCity>
  <ShipPostalCode>04179</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10946</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-12T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-09T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>27.2</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10947</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-16T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.26</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10948</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-19T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>23.39</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10949</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-13T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-10T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-17T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>74.44</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>10950</OrderID>
  <CustomerID>MAGAA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>2.5</Freight>
  <ShipName>Magazzini Alimentari Riuniti</ShipName>
  <ShipAddress>Via Ludovico il Moro 22</ShipAddress>
  <ShipCity>Bergamo</ShipCity>
  <ShipPostalCode>24100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>10951</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-03-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-07T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>30.85</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10952</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>40.42</Freight>
  <ShipName>Alfred's Futterkiste</ShipName>
  <ShipAddress>Obere Str. 57</ShipAddress>
  <ShipCity>Berlin</ShipCity>
  <ShipPostalCode>12209</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10953</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-03-16T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-03-30T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-03-25T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>23.72</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10954</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-03-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>27.91</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10955</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>3.26</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10956</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-17T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-20T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>44.65</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10957</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>105.36</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10958</OrderID>
  <CustomerID>OCEAN</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-03-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-27T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>49.56</Freight>
  <ShipName>Océano Atlántico Ltda.</ShipName>
  <ShipAddress>Ing. Gustavo Moncada 8585 Piso 20-A</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10959</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-18T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.98</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10960</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-02T00:00:00.0000000-08:00</RequiredDate>
  <ShippedDate>1998-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>2.08</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10961</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>104.47</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
```

```
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10962</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-23T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>275.79</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10963</OrderID>
  <CustomerID>FURIB</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-03-19T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>2.7</Freight>
  <ShipName>Furia Bacalhau e Frutos do Mar</ShipName>
  <ShipAddress>Jardim das rosas n. 32</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1675</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>10964</OrderID>
  <CustomerID>SPECD</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-24T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>87.38</Freight>
  <ShipName>Spécialités du monde</ShipName>
  <ShipAddress>25, rue Lauriston</ShipAddress>
  <ShipCity>Paris</ShipCity>
  <ShipPostalCode>75016</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10965</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-30T00:00:00.0000000-08:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>144.38</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10966</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-20T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>27.19</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10967</OrderID>
  <CustomerID>TOMSP</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>62.22</Freight>
  <ShipName>Toms Spezialitäten</ShipName>
  <ShipAddress>Luisenstr. 48</ShipAddress>
  <ShipCity>Münster</ShipCity>
  <ShipPostalCode>44087</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10968</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-01T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>74.6</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10969</OrderID>
  <CustomerID>COMMI</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-23T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-30T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>0.21</Freight>
  <ShipName>Comércio Mineiro</ShipName>
  <ShipAddress>Av. dos Lusíadas, 23</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05432-043</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10970</OrderID>
  <CustomerID>BOLID</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-03-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>16.16</Freight>
  <ShipName>Bólido Comidas preparadas</ShipName>
  <ShipAddress>C/ Araquil, 67</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28023</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>10971</OrderID>
  <CustomerID>FRANR</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>121.82</Freight>
  <ShipName>France restauration</ShipName>
  <ShipAddress>54, rue Royale</ShipAddress>
  <ShipCity>Nantes</ShipCity>
  <ShipPostalCode>44000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10972</OrderID>
  <CustomerID>LACOR</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-26T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>0.02</Freight>
  <ShipName>La corne d'abondance</ShipName>
  <ShipAddress>67, avenue de l'Europe</ShipAddress>
  <ShipCity>Versailles</ShipCity>
  <ShipPostalCode>78000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10973</OrderID>
  <CustomerID>LACOR</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-03-24T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>15.17</Freight>
  <ShipName>La corne d'abondance</ShipName>
  <ShipAddress>67, avenue de l'Europe</ShipAddress>
  <ShipCity>Versailles</ShipCity>
  <ShipPostalCode>78000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>10974</OrderID>
  <CustomerID>SPLIR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>12.96</Freight>
  <ShipName>Split Rail Beer &amp;amp; Ale</ShipName>
  <ShipAddress>P.O. Box 555</ShipAddress>
  <ShipCity>Lander</ShipCity>
  <ShipRegion>WY</ShipRegion>
  <ShipPostalCode>82520</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10975</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-27T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>32.27</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10976</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-25T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-05-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>37.97</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10977</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>208.5</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10978</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-03-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>32.82</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>10979</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-26T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-03-31T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>353.07</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10980</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-03-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-05-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.26</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10981</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>193.37</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>10982</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>14.01</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10983</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-27T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>657.54</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10984</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-03-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>211.22</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10985</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>91.51</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10986</OrderID>
  <CustomerID>OCEAN</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-30T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-21T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>217.86</Freight>
  <ShipName>Océano Atlántico Ltda.</ShipName>
  <ShipAddress>Ing. Gustavo Moncada 8585 Piso 20-A</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>10987</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-03-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>185.48</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>10988</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-03-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>61.14</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10989</OrderID>
  <CustomerID>QUEDE</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-03-31T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-02T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>34.76</Freight>
  <ShipName>Que Delícia</ShipName>
  <ShipAddress>Rua da Panificadora, 12</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-673</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>10990</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-05-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>117.61</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>10991</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-07T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>38.51</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>10992</OrderID>
  <CustomerID>THEBI</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-03T00:00:00.0000000-08:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.27</Freight>
  <ShipName>The Big Cheese</ShipName>
  <ShipAddress>89 Jefferson Way Suite 2</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97201</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>10993</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-01T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>3</ShipVia>
  <Freight>8.81</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>10994</OrderID>
  <CustomerID>VAFFE</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-16T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-09T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>65.53</Freight>
  <ShipName>Vaffeljernet</ShipName>
  <ShipAddress>Smagsloget 45</ShipAddress>
  <ShipCity>Århus</ShipCity>
  <ShipPostalCode>8200</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>10995</OrderID>
  <CustomerID>PERIC</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>46</Freight>
  <ShipName>Pericles Comidas clásicas</ShipName>
  <ShipAddress>Calle Dr. Jorge Cash 321</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>10996</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-02T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-30T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>1.12</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
```

```
<Orders>
  <OrderID>10997</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-05-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>73.91</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>10998</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-04-17T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>20.31</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>10999</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-04-03T00:00:00.0000000-08:00</OrderDate>
  <RequiredDate>1998-05-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>96.35</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11000</OrderID>
  <CustomerID>RATTC</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
```

```
  <Freight>55.12</Freight>
  <ShipName>Rattlesnake Canyon Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipRegion>NM</ShipRegion>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11001</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-14T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>197.3</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>11002</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>141.16</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11003</OrderID>
  <CustomerID>THECR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-04T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-08T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>14.91</Freight>
  <ShipName>The Cracker Box</ShipName>
  <ShipAddress>55 Grizzly Peak Rd.</ShipAddress>
  <ShipCity>Butte</ShipCity>
  <ShipRegion>MT</ShipRegion>
  <ShipPostalCode>59801</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>11004</OrderID>
  <CustomerID>MAISD</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>44.84</Freight>
  <ShipName>Maison Dewey</ShipName>
  <ShipAddress>Rue Joseph-Bens 532</ShipAddress>
  <ShipCity>Bruxelles</ShipCity>
  <ShipPostalCode>B-1180</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>11005</OrderID>
  <CustomerID>WILMK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>0.75</Freight>
  <ShipName>Wilman Kala</ShipName>
  <ShipAddress>Keskuskatu 45</ShipAddress>
  <ShipCity>Helsinki</ShipCity>
  <ShipPostalCode>21240</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11006</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-07T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-05T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>25.19</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11007</OrderID>
  <CustomerID>PRINI</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-13T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>202.24</Freight>
  <ShipName>Princesa Isabel Vinhos</ShipName>
  <ShipAddress>Estrada da saúde n. 58</ShipAddress>
  <ShipCity>Lisboa</ShipCity>
  <ShipPostalCode>1756</ShipPostalCode>
  <ShipCountry>Portugal</ShipCountry>
</Orders>
<Orders>
  <OrderID>11008</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-06T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>3</ShipVia>
  <Freight>79.46</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>11009</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-08T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>59.11</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>11010</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>28.71</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>11011</OrderID>
```

```
  <CustomerID>ALFKI</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>1.21</Freight>
  <ShipName>Alfred's Futterkiste</ShipName>
  <ShipAddress>Obere Str. 57</ShipAddress>
  <ShipCity>Berlin</ShipCity>
  <ShipPostalCode>12209</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11012</OrderID>
  <CustomerID>FRANK</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-04-23T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-17T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>242.95</Freight>
  <ShipName>Frankenversand</ShipName>
  <ShipAddress>Berliner Platz 43</ShipAddress>
  <ShipCity>München</ShipCity>
  <ShipPostalCode>80805</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11013</OrderID>
  <CustomerID>ROMEY</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-09T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-07T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-10T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>32.99</Freight>
  <ShipName>Romero y tomillo</ShipName>
  <ShipAddress>Gran Vía, 1</ShipAddress>
  <ShipCity>Madrid</ShipCity>
  <ShipPostalCode>28001</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>11014</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-15T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>23.6</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
```

```
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>11015</OrderID>
  <CustomerID>SANTG</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-04-24T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>4.62</Freight>
  <ShipName>Santé Gourmet</ShipName>
  <ShipAddress>Erling Skakkes gate 78</ShipAddress>
  <ShipCity>Stavern</ShipCity>
  <ShipPostalCode>4110</ShipPostalCode>
  <ShipCountry>Norway</ShipCountry>
</Orders>
<Orders>
  <OrderID>11016</OrderID>
  <CustomerID>AROUT</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-04-10T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-08T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-13T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>33.8</Freight>
  <ShipName>Around the Horn</ShipName>
  <ShipAddress>Brook Farm Stratford St. Mary</ShipAddress>
  <ShipCity>Colchester</ShipCity>
  <ShipRegion>Essex</ShipRegion>
  <ShipPostalCode>CO7 6JX</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>11017</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-04-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>754.26</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>11018</OrderID>
  <CustomerID>LONEP</CustomerID>
```

```
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-11T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>11.65</Freight>
  <ShipName>Lonesome Pine Restaurant</ShipName>
  <ShipAddress>89 Chiaroscuro Rd.</ShipAddress>
  <ShipCity>Portland</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97219</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11019</OrderID>
  <CustomerID>RANCH</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-04-13T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-11T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>3</ShipVia>
  <Freight>3.17</Freight>
  <ShipName>Rancho grande</ShipName>
  <ShipAddress>Av. del Libertador 900</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>11020</OrderID>
  <CustomerID>OTTIK</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-16T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>43.3</Freight>
  <ShipName>Ottilies Käseladen</ShipName>
  <ShipAddress>Mehrheimerstr. 369</ShipAddress>
  <ShipCity>Köln</ShipCity>
  <ShipPostalCode>50739</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11021</OrderID>
  <CustomerID>QUICK</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-21T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>297.18</Freight>
  <ShipName>QUICK-Stop</ShipName>
  <ShipAddress>Taucherstraße 10</ShipAddress>
  <ShipCity>Cunewalde</ShipCity>
```

```
  <ShipPostalCode>01307</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11022</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-04-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>6.27</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>11023</OrderID>
  <CustomerID>BSBEV</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-14T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-04-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>123.83</Freight>
  <ShipName>B's Beverages</ShipName>
  <ShipAddress>Fauntleroy Circus</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>EC2 5NT</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>11024</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>74.36</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>11025</OrderID>
  <CustomerID>WARTH</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-04-15T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1998-05-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>29.17</Freight>
  <ShipName>Wartian Herkku</ShipName>
  <ShipAddress>Torikatu 38</ShipAddress>
  <ShipCity>Oulu</ShipCity>
  <ShipPostalCode>90110</ShipPostalCode>
  <ShipCountry>Finland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11026</OrderID>
  <CustomerID>FRANS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-15T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-13T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>47.09</Freight>
  <ShipName>Franchi S.p.A.</ShipName>
  <ShipAddress>Via Monte Bianco 34</ShipAddress>
  <ShipCity>Torino</ShipCity>
  <ShipPostalCode>10100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>11027</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-20T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>52.52</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>11028</OrderID>
  <CustomerID>KOENE</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>29.59</Freight>
  <ShipName>Königlich Essen</ShipName>
  <ShipAddress>Maubelstr. 90</ShipAddress>
  <ShipCity>Brandenburg</ShipCity>
  <ShipPostalCode>14776</ShipPostalCode>
```

```
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11029</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-16T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-14T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>47.84</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11030</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>830.75</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11031</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-04-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>227.22</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11032</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-17T00:00:00.0000000-07:00</OrderDate>
```

```
  <RequiredDate>1998-05-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>606.19</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11033</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-17T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-15T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-23T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>84.74</Freight>
  <ShipName>Richter Supermarkt</ShipName>
  <ShipAddress>Starenweg 5</ShipAddress>
  <ShipCity>Genève</ShipCity>
  <ShipPostalCode>1204</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11034</OrderID>
  <CustomerID>OLDWO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>40.32</Freight>
  <ShipName>Old World Delicatessen</ShipName>
  <ShipAddress>2743 Bering St.</ShipAddress>
  <ShipCity>Anchorage</ShipCity>
  <ShipRegion>AK</ShipRegion>
  <ShipPostalCode>99508</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11035</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-24T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>0.17</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
```

```
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>11036</OrderID>
  <CustomerID>DRACD</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-20T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-22T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>149.47</Freight>
  <ShipName>Drachenblut Delikatessen</ShipName>
  <ShipAddress>Walserweg 21</ShipAddress>
  <ShipCity>Aachen</ShipCity>
  <ShipPostalCode>52066</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11037</OrderID>
  <CustomerID>GODOS</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-27T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>3.2</Freight>
  <ShipName>Godos Cocina Típica</ShipName>
  <ShipAddress>C/ Romero, 33</ShipAddress>
  <ShipCity>Sevilla</ShipCity>
  <ShipPostalCode>41101</ShipPostalCode>
  <ShipCountry>Spain</ShipCountry>
</Orders>
<Orders>
  <OrderID>11038</OrderID>
  <CustomerID>SUPRD</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-19T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>29.59</Freight>
  <ShipName>Suprêmes délices</ShipName>
  <ShipAddress>Boulevard Tirou, 255</ShipAddress>
  <ShipCity>Charleroi</ShipCity>
  <ShipPostalCode>B-6000</ShipPostalCode>
  <ShipCountry>Belgium</ShipCountry>
</Orders>
<Orders>
  <OrderID>11039</OrderID>
  <CustomerID>LINOD</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-04-21T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-19T00:00:00.0000000-07:00</RequiredDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>65</Freight>
  <ShipName>LINO-Delicateses</ShipName>
  <ShipAddress>Ave. 5 de Mayo Porlamar</ShipAddress>
  <ShipCity>I. de Margarita</ShipCity>
  <ShipRegion>Nueva Esparta</ShipRegion>
  <ShipPostalCode>4980</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>11040</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-20T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>3</ShipVia>
  <Freight>18.84</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11041</OrderID>
  <CustomerID>CHOPS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-28T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>48.22</Freight>
  <ShipName>Chop-suey Chinese</ShipName>
  <ShipAddress>Hauptstr. 31</ShipAddress>
  <ShipCity>Bern</ShipCity>
  <ShipPostalCode>3012</ShipPostalCode>
  <ShipCountry>Switzerland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11042</OrderID>
  <CustomerID>COMMI</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-06T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>29.99</Freight>
  <ShipName>Comércio Mineiro</ShipName>
  <ShipAddress>Av. dos Lusíadas, 23</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05432-043</ShipPostalCode>
```

```
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>11043</OrderID>
  <CustomerID>SPECD</CustomerID>
  <EmployeeID>5</EmployeeID>
  <OrderDate>1998-04-22T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-20T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>8.8</Freight>
  <ShipName>Spécialités du monde</ShipName>
  <ShipAddress>25, rue Lauriston</ShipAddress>
  <ShipCity>Paris</ShipCity>
  <ShipPostalCode>75016</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>11044</OrderID>
  <CustomerID>WOLZA</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.72</Freight>
  <ShipName>Wolski Zajazd</ShipName>
  <ShipAddress>ul. Filtrowa 68</ShipAddress>
  <ShipCity>Warszawa</ShipCity>
  <ShipPostalCode>01-012</ShipPostalCode>
  <ShipCountry>Poland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11045</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>6</EmployeeID>
  <OrderDate>1998-04-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-21T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>70.58</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>11046</OrderID>
  <CustomerID>WANDK</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-23T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-21T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-24T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>2</ShipVia>
  <Freight>71.64</Freight>
  <ShipName>Die Wandernde Kuh</ShipName>
  <ShipAddress>Adenauerallee 900</ShipAddress>
  <ShipCity>Stuttgart</ShipCity>
  <ShipPostalCode>70563</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11047</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>46.62</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>11048</OrderID>
  <CustomerID>BOTTM</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-30T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>24.12</Freight>
  <ShipName>Bottom-Dollar Markets</ShipName>
  <ShipAddress>23 Tsawassen Blvd.</ShipAddress>
  <ShipCity>Tsawassen</ShipCity>
  <ShipRegion>BC</ShipRegion>
  <ShipPostalCode>T2F 8M4</ShipPostalCode>
  <ShipCountry>Canada</ShipCountry>
</Orders>
<Orders>
  <OrderID>11049</OrderID>
  <CustomerID>GOURL</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-24T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-22T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>8.34</Freight>
  <ShipName>Gourmet Lanchonetes</ShipName>
  <ShipAddress>Av. Brasil, 442</ShipAddress>
  <ShipCity>Campinas</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>04876-786</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>11050</OrderID>
  <CustomerID>FOLKO</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>59.41</Freight>
  <ShipName>Folk och fä HB</ShipName>
  <ShipAddress>Åkergatan 24</ShipAddress>
  <ShipCity>Bräcke</ShipCity>
  <ShipPostalCode>S-844 67</ShipPostalCode>
  <ShipCountry>Sweden</ShipCountry>
</Orders>
<Orders>
  <OrderID>11051</OrderID>
  <CustomerID>LAMAI</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-25T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>3</ShipVia>
  <Freight>2.79</Freight>
  <ShipName>La maison d'Asie</ShipName>
  <ShipAddress>1 rue Alsace-Lorraine</ShipAddress>
  <ShipCity>Toulouse</ShipCity>
  <ShipPostalCode>31000</ShipPostalCode>
  <ShipCountry>France</ShipCountry>
</Orders>
<Orders>
  <OrderID>11052</OrderID>
  <CustomerID>HANAR</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>1</ShipVia>
  <Freight>67.26</Freight>
  <ShipName>Hanari Carnes</ShipName>
  <ShipAddress>Rua do Paço, 67</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>05454-876</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>11053</OrderID>
  <CustomerID>PICCO</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-27T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-25T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-04-29T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
```

```
  <Freight>53.05</Freight>
  <ShipName>Piccolo und mehr</ShipName>
  <ShipAddress>Geislweg 14</ShipAddress>
  <ShipCity>Salzburg</ShipCity>
  <ShipPostalCode>5020</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>11054</OrderID>
  <CustomerID>CACTU</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-26T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>1</ShipVia>
  <Freight>0.33</Freight>
  <ShipName>Cactus Comidas para llevar</ShipName>
  <ShipAddress>Cerrito 333</ShipAddress>
  <ShipCity>Buenos Aires</ShipCity>
  <ShipPostalCode>1010</ShipPostalCode>
  <ShipCountry>Argentina</ShipCountry>
</Orders>
<Orders>
  <OrderID>11055</OrderID>
  <CustomerID>HILAA</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-04-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-26T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-05T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>120.92</Freight>
  <ShipName>HILARION-Abastos</ShipName>
  <ShipAddress>Carrera 22 con Ave. Carlos Soublette #8-35</ShipAddress>
  <ShipCity>San Cristóbal</ShipCity>
  <ShipRegion>Táchira</ShipRegion>
  <ShipPostalCode>5022</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>11056</OrderID>
  <CustomerID>EASTC</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-04-28T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-12T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>278.96</Freight>
  <ShipName>Eastern Connection</ShipName>
  <ShipAddress>35 King George</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>WX3 6FW</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>11057</OrderID>
```

```
  <CustomerID>NORTS</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-27T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-01T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>3</ShipVia>
  <Freight>4.13</Freight>
  <ShipName>North/South</ShipName>
  <ShipAddress>South House 300 Queensbridge</ShipAddress>
  <ShipCity>London</ShipCity>
  <ShipPostalCode>SW7 1RZ</ShipPostalCode>
  <ShipCountry>UK</ShipCountry>
</Orders>
<Orders>
  <OrderID>11058</OrderID>
  <CustomerID>BLAUS</CustomerID>
  <EmployeeID>9</EmployeeID>
  <OrderDate>1998-04-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-27T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>3</ShipVia>
  <Freight>31.14</Freight>
  <ShipName>Blauer See Delikatessen</ShipName>
  <ShipAddress>Forsterstr. 57</ShipAddress>
  <ShipCity>Mannheim</ShipCity>
  <ShipPostalCode>68306</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11059</OrderID>
  <CustomerID>RICAR</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-29T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-10T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>85.8</Freight>
  <ShipName>Ricardo Adocicados</ShipName>
  <ShipAddress>Av. Copacabana, 267</ShipAddress>
  <ShipCity>Rio de Janeiro</ShipCity>
  <ShipRegion>RJ</ShipRegion>
  <ShipPostalCode>02389-890</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>11060</OrderID>
  <CustomerID>FRANS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-04-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>10.98</Freight>
  <ShipName>Franchi S.p.A.</ShipName>
  <ShipAddress>Via Monte Bianco 34</ShipAddress>
  <ShipCity>Torino</ShipCity>
```

```
  <ShipPostalCode>10100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>11061</OrderID>
  <CustomerID>GREAL</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-11T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>3</ShipVia>
  <Freight>14.01</Freight>
  <ShipName>Great Lakes Food Market</ShipName>
  <ShipAddress>2732 Baker Blvd.</ShipAddress>
  <ShipCity>Eugene</ShipCity>
  <ShipRegion>OR</ShipRegion>
  <ShipPostalCode>97403</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11062</OrderID>
  <CustomerID>REGGC</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-04-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-28T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>29.93</Freight>
  <ShipName>Reggiani Caseifici</ShipName>
  <ShipAddress>Strada Provinciale 124</ShipAddress>
  <ShipCity>Reggio Emilia</ShipCity>
  <ShipPostalCode>42100</ShipPostalCode>
  <ShipCountry>Italy</ShipCountry>
</Orders>
<Orders>
  <OrderID>11063</OrderID>
  <CustomerID>HUNGO</CustomerID>
  <EmployeeID>3</EmployeeID>
  <OrderDate>1998-04-30T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-28T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>81.73</Freight>
  <ShipName>Hungry Owl All-Night Grocers</ShipName>
  <ShipAddress>8 Johnstown Road</ShipAddress>
  <ShipCity>Cork</ShipCity>
  <ShipRegion>Co. Cork</ShipRegion>
  <ShipCountry>Ireland</ShipCountry>
</Orders>
<Orders>
  <OrderID>11064</OrderID>
  <CustomerID>SAVEA</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-05-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-04T00:00:00.0000000-07:00</ShippedDate>
```

```
  <ShipVia>1</ShipVia>
  <Freight>30.09</Freight>
  <ShipName>Save-a-lot Markets</ShipName>
  <ShipAddress>187 Suffolk Ln.</ShipAddress>
  <ShipCity>Boise</ShipCity>
  <ShipRegion>ID</ShipRegion>
  <ShipPostalCode>83720</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11065</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-05-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-29T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>1</ShipVia>
  <Freight>12.91</Freight>
  <ShipName>LILA-Supermercado</ShipName>
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>11066</OrderID>
  <CustomerID>WHITC</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-05-01T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-29T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-04T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>44.72</Freight>
  <ShipName>White Clover Markets</ShipName>
  <ShipAddress>1029 - 12th Ave. S.</ShipAddress>
  <ShipCity>Seattle</ShipCity>
  <ShipRegion>WA</ShipRegion>
  <ShipPostalCode>98124</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
</Orders>
<Orders>
  <OrderID>11067</OrderID>
  <CustomerID>DRACD</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-05-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-05-18T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>7.98</Freight>
  <ShipName>Drachenblut Delikatessen</ShipName>
  <ShipAddress>Walserweg 21</ShipAddress>
  <ShipCity>Aachen</ShipCity>
  <ShipPostalCode>52066</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
```

```
</Orders>
<Orders>
  <OrderID>11068</OrderID>
  <CustomerID>QUEEN</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-05-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-01T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>81.75</Freight>
  <ShipName>Queen Cozinha</ShipName>
  <ShipAddress>Alameda dos Canàrios, 891</ShipAddress>
  <ShipCity>Sao Paulo</ShipCity>
  <ShipRegion>SP</ShipRegion>
  <ShipPostalCode>05487-020</ShipPostalCode>
  <ShipCountry>Brazil</ShipCountry>
</Orders>
<Orders>
  <OrderID>11069</OrderID>
  <CustomerID>TORTU</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-05-04T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-01T00:00:00.0000000-07:00</RequiredDate>
  <ShippedDate>1998-05-06T00:00:00.0000000-07:00</ShippedDate>
  <ShipVia>2</ShipVia>
  <Freight>15.67</Freight>
  <ShipName>Tortuga Restaurante</ShipName>
  <ShipAddress>Avda. Azteca 123</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>11070</OrderID>
  <CustomerID>LEHMS</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-05-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-02T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>1</ShipVia>
  <Freight>136</Freight>
  <ShipName>Lehmanns Marktstand</ShipName>
  <ShipAddress>Magazinweg 7</ShipAddress>
  <ShipCity>Frankfurt a.M.</ShipCity>
  <ShipPostalCode>60528</ShipPostalCode>
  <ShipCountry>Germany</ShipCountry>
</Orders>
<Orders>
  <OrderID>11071</OrderID>
  <CustomerID>LILAS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1998-05-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-02T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>1</ShipVia>
  <Freight>0.93</Freight>
  <ShipName>LILA-Supermercado</ShipName>
```

```
  <ShipAddress>Carrera 52 con Ave. Bolívar #65-98 Llano Largo</ShipAddress>
  <ShipCity>Barquisimeto</ShipCity>
  <ShipRegion>Lara</ShipRegion>
  <ShipPostalCode>3508</ShipPostalCode>
  <ShipCountry>Venezuela</ShipCountry>
</Orders>
<Orders>
  <OrderID>11072</OrderID>
  <CustomerID>ERNSH</CustomerID>
  <EmployeeID>4</EmployeeID>
  <OrderDate>1998-05-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-02T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>258.64</Freight>
  <ShipName>Ernst Handel</ShipName>
  <ShipAddress>Kirchgasse 6</ShipAddress>
  <ShipCity>Graz</ShipCity>
  <ShipPostalCode>8010</ShipPostalCode>
  <ShipCountry>Austria</ShipCountry>
</Orders>
<Orders>
  <OrderID>11073</OrderID>
  <CustomerID>PERIC</CustomerID>
  <EmployeeID>2</EmployeeID>
  <OrderDate>1998-05-05T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-02T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>24.95</Freight>
  <ShipName>Pericles Comidas clásicas</ShipName>
  <ShipAddress>Calle Dr. Jorge Cash 321</ShipAddress>
  <ShipCity>México D.F.</ShipCity>
  <ShipPostalCode>05033</ShipPostalCode>
  <ShipCountry>Mexico</ShipCountry>
</Orders>
<Orders>
  <OrderID>11074</OrderID>
  <CustomerID>SIMOB</CustomerID>
  <EmployeeID>7</EmployeeID>
  <OrderDate>1998-05-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-03T00:00:00.0000000-07:00</RequiredDate>
  <ShipVia>2</ShipVia>
  <Freight>18.44</Freight>
  <ShipName>Simons bistro</ShipName>
  <ShipAddress>Vinbæltet 34</ShipAddress>
  <ShipCity>Kobenhavn</ShipCity>
  <ShipPostalCode>1734</ShipPostalCode>
  <ShipCountry>Denmark</ShipCountry>
</Orders>
<Orders>
  <OrderID>11075</OrderID>
  <CustomerID>RICSU</CustomerID>
  <EmployeeID>8</EmployeeID>
  <OrderDate>1998-05-06T00:00:00.0000000-07:00</OrderDate>
  <RequiredDate>1998-06-03T00:00:00.0000000-07:00</RequiredDate>
```

```
    <ShipVia>2</ShipVia>
    <Freight>6.19</Freight>
    <ShipName>Richter Supermarkt</ShipName>
    <ShipAddress>Starenweg 5</ShipAddress>
    <ShipCity>Genève</ShipCity>
    <ShipPostalCode>1204</ShipPostalCode>
    <ShipCountry>Switzerland</ShipCountry>
  </Orders>
  <Orders>
    <OrderID>11076</OrderID>
    <CustomerID>BONAP</CustomerID>
    <EmployeeID>4</EmployeeID>
    <OrderDate>1998-05-06T00:00:00.0000000-07:00</OrderDate>
    <RequiredDate>1998-06-03T00:00:00.0000000-07:00</RequiredDate>
    <ShipVia>2</ShipVia>
    <Freight>38.28</Freight>
    <ShipName>Bon app'</ShipName>
    <ShipAddress>12, rue des Bouchers</ShipAddress>
    <ShipCity>Marseille</ShipCity>
    <ShipPostalCode>13008</ShipPostalCode>
    <ShipCountry>France</ShipCountry>
  </Orders>
  <Orders>
    <OrderID>11077</OrderID>
    <CustomerID>RATTC</CustomerID>
    <EmployeeID>1</EmployeeID>
    <OrderDate>1998-05-06T00:00:00.0000000-07:00</OrderDate>
    <RequiredDate>1998-06-03T00:00:00.0000000-07:00</RequiredDate>
    <ShipVia>2</ShipVia>
    <Freight>8.53</Freight>
    <ShipName>Rattlesnake Canyon Grocery</ShipName>
    <ShipAddress>2817 Milton Dr.</ShipAddress>
    <ShipCity>Albuquerque</ShipCity>
    <ShipRegion>NM</ShipRegion>
    <ShipPostalCode>87110</ShipPostalCode>
    <ShipCountry>USA</ShipCountry>
  </Orders>
</NewDataSet>
```

DataTable Customers has 2 Error(s)!
Row Error for row ABC -- Error Msg=An Error was added
Row Error for row ALFKI -- Error Msg=This is another error message

Customer: Peach

Customer: Peach
Order #10643
Order #10692
Order #10702
Order #10835
Order #10952
Order #11011

Customer: Ana Trujillo
Order #10308
Order #10625
Order #10759
Order #10926

Customer: Antonio Moreno
Order #10365
Order #10507
Order #10535
Order #10573
Order #10677
Order #10682
Order #10856

Customer: Thomas Hardy
Order #10355
Order #10383
Order #10453
Order #10558
Order #10707
Order #10741
Order #10743
Order #10768
Order #10793
Order #10864
Order #10920
Order #10953
Order #11016

Customer: Christina Berglund
Order #10278
Order #10280
Order #10384
Order #10444
Order #10445
Order #10524

Order #10572
Order #10626
Order #10654
Order #10672
Order #10689
Order #10733
Order #10778
Order #10837
Order #10857
Order #10866
Order #10875
Order #10924

Customer: Hanna Moos
Order #10501
Order #10509
Order #10582
Order #10614
Order #10853
Order #10956
Order #11058

Customer: Frédérique Citeaux
Order #10265
Order #10297
Order #10360
Order #10436
Order #10449
Order #10559
Order #10566
Order #10584
Order #10628
Order #10679
Order #10826

Customer: Martín Sommer
Order #10326
Order #10801
Order #10970

Customer: Laurence Lebihan
Order #10331
Order #10340
Order #10362
Order #10470
Order #10511
Order #10525

Order #10663
Order #10715
Order #10730
Order #10732
Order #10755
Order #10827
Order #10871
Order #10876
Order #10932
Order #10940
Order #11076

Customer: Elizabeth Lincoln
Order #10389
Order #10410
Order #10411
Order #10431
Order #10492
Order #10742
Order #10918
Order #10944
Order #10949
Order #10975
Order #10982
Order #11027
Order #11045
Order #11048

Customer: Victoria Ashworth
Order #10289
Order #10471
Order #10484
Order #10538
Order #10539
Order #10578
Order #10599
Order #10943
Order #10947
Order #11023

Customer: Patricio Simpson
Order #10521
Order #10782
Order #10819
Order #10881
Order #10937
Order #11054

Customer: Francisco Chang
Order #10259

Customer: Yang Wang
Order #10254
Order #10370
Order #10519
Order #10731
Order #10746
Order #10966
Order #11029
Order #11041

Customer: Pedro Afonso
Order #10290
Order #10466
Order #10494
Order #10969
Order #11042

Customer: Elizabeth Brown
Order #10435
Order #10462
Order #10848

Customer: Sven Ottlieb
Order #10363
Order #10391
Order #10797
Order #10825
Order #11036
Order #11067

Customer: Janine Labrune
Order #10311
Order #10609
Order #10683
Order #10890

Customer: Ann Devon
Order #10364
Order #10400
Order #10532
Order #10726
Order #10987
Order #11024

Order #11047
Order #11056

Customer: Roland Mendel
Order #10258
Order #10263
Order #10351
Order #10368
Order #10382
Order #10390
Order #10402
Order #10403
Order #10430
Order #10442
Order #10514
Order #10571
Order #10595
Order #10633
Order #10667
Order #10698
Order #10764
Order #10771
Order #10773
Order #10776
Order #10795
Order #10836
Order #10854
Order #10895
Order #10968
Order #10979
Order #10990
Order #11008
Order #11017
Order #11072

Customer: Aria Cruz
Order #10347
Order #10386
Order #10414
Order #10512
Order #10581
Order #10650
Order #10725

Customer: Diego Roel

Customer: Martine Rancé

Order #10408
Order #10480
Order #10634
Order #10763
Order #10789

Customer: Maria Larsson
Order #10264
Order #10327
Order #10378
Order #10434
Order #10460
Order #10533
Order #10561
Order #10703
Order #10762
Order #10774
Order #10824
Order #10880
Order #10902
Order #10955
Order #10977
Order #10980
Order #10993
Order #11001
Order #11050

Customer: Peter Franken
Order #10267
Order #10337
Order #10342
Order #10396
Order #10488
Order #10560
Order #10623
Order #10653
Order #10670
Order #10675
Order #10717
Order #10791
Order #10859
Order #10929
Order #11012

Customer: Carine Schmitt
Order #10671
Order #10860

Order #10971

Customer: Paolo Accorti
Order #10422
Order #10710
Order #10753
Order #10807
Order #11026
Order #11060

Customer: Lino Rodriguez
Order #10328
Order #10352
Order #10464
Order #10491
Order #10551
Order #10604
Order #10664
Order #10963

Customer: Eduardo Saavedra
Order #10366
Order #10426
Order #10568
Order #10887
Order #10928

Customer: José Pedro Freyre
Order #10303
Order #10550
Order #10629
Order #10872
Order #10874
Order #10888
Order #10911
Order #10948
Order #11009
Order #11037

Customer: André Fonseca
Order #10423
Order #10652
Order #10685
Order #10709
Order #10734
Order #10777
Order #10790

Order #10959
Order #11049

Customer: Howard Snyder
Order #10528
Order #10589
Order #10616
Order #10617
Order #10656
Order #10681
Order #10816
Order #10936
Order #11006
Order #11040
Order #11061

Customer: Manuel Pereira
Order #10268
Order #10785

Customer: Mario Pontes
Order #10250
Order #10253
Order #10541
Order #10645
Order #10690
Order #10770
Order #10783
Order #10886
Order #10903
Order #10922
Order #10925
Order #10981
Order #11022
Order #11052

Customer: Carlos Hernández
Order #10257
Order #10395
Order #10476
Order #10486
Order #10490
Order #10498
Order #10552
Order #10601
Order #10613
Order #10641

Order #10705
Order #10796
Order #10863
Order #10901
Order #10957
Order #10960
Order #10976
Order #11055

Customer: Yoshi Latimer
Order #10375
Order #10394
Order #10415
Order #10600
Order #10660

Customer: Patricia McKenna
Order #10298
Order #10309
Order #10335
Order #10373
Order #10380
Order #10429
Order #10503
Order #10516
Order #10567
Order #10646
Order #10661
Order #10687
Order #10701
Order #10712
Order #10736
Order #10897
Order #10912
Order #10985
Order #11063

Customer: Helen Bennett
Order #10315
Order #10318
Order #10321
Order #10473
Order #10621
Order #10674
Order #10749
Order #10798
Order #10829

Order #10933

Customer: Philip Cramer
Order #10323
Order #10325
Order #10456
Order #10457
Order #10468
Order #10506
Order #10542
Order #10630
Order #10718
Order #10799
Order #10817
Order #10849
Order #10893
Order #11028

Customer: Daniel Tonini
Order #10858
Order #10927
Order #10972
Order #10973

Customer: Annette Roulet
Order #10350
Order #10358
Order #10371
Order #10413
Order #10425
Order #10454
Order #10493
Order #10500
Order #10610
Order #10631
Order #10787
Order #10832
Order #10923
Order #11051

Customer: Yoshi Tannamuri
Order #10495
Order #10620
Order #10810

Customer: John Steel
Order #10482

Order #10545

Customer: Renate Messner
Order #10279
Order #10284
Order #10343
Order #10497
Order #10522
Order #10534
Order #10536
Order #10557
Order #10592
Order #10593
Order #10772
Order #10862
Order #10891
Order #10934
Order #11070

Customer: Jaime Yorres
Order #10579
Order #10719
Order #10735
Order #10884

Customer: Carlos González
Order #10283
Order #10296
Order #10330
Order #10357
Order #10381
Order #10461
Order #10499
Order #10543
Order #10780
Order #10823
Order #10899
Order #10997
Order #11065
Order #11071

Customer: Felipe Izquierdo
Order #10405
Order #10485
Order #10638
Order #10697
Order #10729

Order #10811
Order #10838
Order #10840
Order #10919
Order #10954
Order #11014
Order #11039

Customer: Fran Wilson
Order #10307
Order #10317
Order #10544
Order #10662
Order #10665
Order #10867
Order #10883
Order #11018

Customer: Giovanni Rovelli
Order #10275
Order #10300
Order #10404
Order #10467
Order #10635
Order #10754
Order #10784
Order #10818
Order #10939
Order #10950

Customer: Catherine Dewey
Order #10529
Order #10649
Order #10760
Order #10892
Order #10896
Order #10978
Order #11004

Customer: Jean Fresnière
Order #10332
Order #10339
Order #10376
Order #10424
Order #10439
Order #10505
Order #10565

Order #10570
Order #10590
Order #10605
Order #10618
Order #10619
Order #10724

Customer: Alexander Feuer
Order #10277
Order #10575
Order #10699
Order #10779
Order #10945

Customer: New Name

Customer: Simon Crowther
Order #10517
Order #10752
Order #11057

Customer: Yvonne Moncada
Order #10409
Order #10531
Order #10898
Order #10958
Order #10986

Customer: Rene Phillips
Order #10305
Order #10338
Order #10441
Order #10594
Order #10680
Order #10706
Order #10808
Order #10855
Order #10965
Order #11034

Customer: Henriette Pfalzheim
Order #10260
Order #10407
Order #10508
Order #10554
Order #10580
Order #10684

Order #10766
Order #10833
Order #10999
Order #11020

Customer: Marie Bertrand

Customer: Guillermo Fernández
Order #10322
Order #10354
Order #10474
Order #10502
Order #10995
Order #11073

Customer: Georg Pipps
Order #10353
Order #10392
Order #10427
Order #10489
Order #10530
Order #10597
Order #10686
Order #10747
Order #10844
Order #11053

Customer: Isabel de Castro
Order #10336
Order #10397
Order #10433
Order #10477
Order #11007

Customer: Bernardo Batista
Order #10261
Order #10291
Order #10379
Order #10421
Order #10587
Order #10647
Order #10720
Order #10794
Order #10989

Customer: Lúcia Carvalho
Order #10372

Order #10406
Order #10487
Order #10637
Order #10659
Order #10704
Order #10728
Order #10786
Order #10868
Order #10913
Order #10914
Order #10961
Order #11068

Customer: Horst Kloss
Order #10273
Order #10285
Order #10286
Order #10313
Order #10345
Order #10361
Order #10418
Order #10451
Order #10515
Order #10527
Order #10540
Order #10549
Order #10588
Order #10658
Order #10691
Order #10694
Order #10721
Order #10745
Order #10765
Order #10788
Order #10845
Order #10865
Order #10878
Order #10938
Order #10962
Order #10991
Order #10996
Order #11021

Customer: Sergio Gutiérrez
Order #10448
Order #10716

Order #10828
Order #10916
Order #11019

Customer: Paula Wilson
Order #10262
Order #10272
Order #10294
Order #10314
Order #10316
Order #10346
Order #10401
Order #10479
Order #10564
Order #10569
Order #10598
Order #10761
Order #10820
Order #10852
Order #10889
Order #10988
Order #11000
Order #11077

Customer: Maurizio Moroni
Order #10288
Order #10428
Order #10443
Order #10562
Order #10586
Order #10655
Order #10727
Order #10812
Order #10908
Order #10942
Order #11010
Order #11062

Customer: Janete Limeira
Order #10287
Order #10299
Order #10447
Order #10481
Order #10563
Order #10622
Order #10648
Order #10813

Order #10851
Order #10877
Order #11059

Customer: Michael Holz
Order #10255
Order #10419
Order #10537
Order #10666
Order #10751
Order #10758
Order #10931
Order #10951
Order #11033
Order #11075

Customer: Alejandra Camino
Order #10281
Order #10282
Order #10306
Order #10917
Order #11013

Customer: Jonas Bergulfsen
Order #10387
Order #10520
Order #10639
Order #10831
Order #10909
Order #11015

Customer: Jose Pavarotti
Order #10324
Order #10393
Order #10398
Order #10440
Order #10452
Order #10510
Order #10555
Order #10603
Order #10607
Order #10612
Order #10627
Order #10657
Order #10678
Order #10700
Order #10711

Order #10713
Order #10714
Order #10722
Order #10748
Order #10757
Order #10815
Order #10847
Order #10882
Order #10894
Order #10941
Order #10983
Order #10984
Order #11002
Order #11030
Order #11031
Order #11064

Customer: Hari Kumar
Order #10359
Order #10377
Order #10388
Order #10472
Order #10523
Order #10547
Order #10800
Order #10804
Order #10869

Customer: Jytte Petersen
Order #10341
Order #10417
Order #10556
Order #10642
Order #10669
Order #10802
Order #11074

Customer: Dominique Perrier
Order #10738
Order #10907
Order #10964
Order #11043

Customer: Art Braunschweiger
Order #10271
Order #10329
Order #10349

Order #10369
Order #10385
Order #10432
Order #10756
Order #10821
Order #10974

Customer: Pascale Cartrain
Order #10252
Order #10302
Order #10458
Order #10463
Order #10475
Order #10767
Order #10841
Order #10846
Order #10885
Order #10930
Order #11035
Order #11038

Customer: Liz Nixon
Order #10310
Order #10708
Order #10805
Order #10992

Customer: Liu Wong
Order #10624
Order #10775
Order #11003

Customer: Karin Josephs
Order #10249
Order #10438
Order #10446
Order #10548
Order #10608
Order #10967

Customer: Miguel Angel Paolino
Order #10276
Order #10293
Order #10304
Order #10319
Order #10518
Order #10576

Order #10676
Order #10842
Order #10915
Order #11069

Customer: Anabela Domingues
Order #10292
Order #10496
Order #10606
Order #10830
Order #10834
Order #10839

Customer: Helvetius Nagy
Order #10574
Order #10577
Order #10822

Customer: Palle Ibsen
Order #10367
Order #10399
Order #10465
Order #10591
Order #10602
Order #10688
Order #10744
Order #10769
Order #10921
Order #10946
Order #10994

Customer: Mary Saveley
Order #10251
Order #10334
Order #10450
Order #10459
Order #10478
Order #10546
Order #10806
Order #10814
Order #10843
Order #10850

Customer: Paul Henriot
Order #10248
Order #10274
Order #10295

Order #10737
Order #10739

Customer: Rita Müller
Order #10301
Order #10312
Order #10348
Order #10356
Order #10513
Order #10632
Order #10640
Order #10651
Order #10668
Order #11046

Customer: Pirkko Koskitalo
Order #10266
Order #10270
Order #10320
Order #10333
Order #10412
Order #10416
Order #10437
Order #10455
Order #10526
Order #10553
Order #10583
Order #10636
Order #10750
Order #10781
Order #11025

Customer: Paula Parente
Order #10256
Order #10420
Order #10585
Order #10644
Order #10803
Order #10809
Order #10900
Order #10905
Order #10935

Customer: Karl Jablonski
Order #10269
Order #10344
Order #10469

Order #10483
Order #10504
Order #10596
Order #10693
Order #10696
Order #10723
Order #10740
Order #10861
Order #10904
Order #11032
Order #11066

Customer: Matti Karttunen
Order #10615
Order #10673
Order #10695
Order #10873
Order #10879
Order #10910
Order #11005

Customer: Zbyszek Piestrzeniewicz
Order #10374
Order #10611
Order #10792
Order #10870
Order #10906
Order #10998
Order #11044

CustomerID

ABC
ALFKI
ANATR
ANTON
AROUT
BERGS
BLAUS
BLONP
BOLID
BONAP
BOTTM
BSBEV
CACTU
CENTC
CHOPS
COMMI
CONSH
DRACD
DUMON
EASTC
ERNSH
FAMIA
FISSA
FOLIG
FOLKO
FRANK
FRANR
FRANS
FURIB
GALED
GODOS
GOURL
GREAL
GROSR
HANAR
HILAA
HUNGC
HUNGO
ISLAT
KOENE
LACOR
LAMAI
LAUGB
LAZYK

LEHMS
LETSS
LILAS
LINOD
LONEP
MAGAA
MAISD
MEREP
MORGK
NewID
NORTS
OCEAN
OLDWO
OTTIK
PARIS
PERIC
PICCO
PRINI
QUEDE
QUEEN
QUICK
RANCH
RATTC
REGGC
RICAR
RICSU
ROMEY
SANTG
SAVEA
SEVES
SIMOB
SPECD
SPLIR
SUPRD
THEBI
THECR
TOMSP
TORTU
TRADH
TRAIH
VAFFE
VICTE
VINET
WANDK
WARTH
WELLI

WHITC
WILMK
WOLZA

ALFKI - Alfreds Futterkiste

Open two connections.
Return both of the connections to the pool.
Open a connection from the pool.
Open a second connection from the pool.
Open a third connection.
Return all three connections to the pool.

Data has been writen to the output files: data1.xml and data2.xml

FillCount=7

```
<NewDataSet>
  <ADODB.RecordSet>
    <RegionID>1</RegionID>
    <RegionDescription>Eastern                                        </RegionDescription>
  </ADODB.RecordSet>
  <ADODB.RecordSet>
    <RegionID>2</RegionID>
    <RegionDescription>Changed this region desc                       </RegionDescription>
  </ADODB.RecordSet>
  <ADODB.RecordSet>
    <RegionID>3</RegionID>
    <RegionDescription>Northern                                       </RegionDescription>
  </ADODB.RecordSet>
  <ADODB.RecordSet>
    <RegionID>4</RegionID>
    <RegionDescription>Southern                                       </RegionDescription>
  </ADODB.RecordSet>
  <ADODB.RecordSet>
    <RegionID>901</RegionID>
    <RegionDescription>A new region                                   </RegionDescription>
  </ADODB.RecordSet>
  <ADODB.RecordSet>
    <RegionID>100</RegionID>
    <RegionDescription>MidWestern                                     </RegionDescription>
  </ADODB.RecordSet>
  <ADODB.RecordSet>
    <RegionID>101</RegionID>
    <RegionDescription>MidEastern                                     </RegionDescription>
  </ADODB.RecordSet>
</NewDataSet>
```

EmployeeID	Name	Title	ReportsTo
1	Nancy Davolio	Sales Representative	2
2	Andrew Fuller	Vice President, Sales	N/A
3	Janet Leverling	Sales Representative	2
4	Margaret Peacock	Sales Representative	2
5	Steven Buchanan	Sales Manager	2
6	Michael Suyama	Sales Representative	5
7	Robert King	Sales Representative	5
8	Laura Callahan	Inside Sales Coordinator	2
9	Anne Dodsworth	Sales Representative	5

The following example demonstrates reading and writing resource files through code. This demonstration uses objects from the Resources namespace ResourceManager, ResourceWriter, and ResourceReader. In this sample, you can specify name and value pairs to add to an arbitrary resource file, which you can subsequently retrieve particular entries from, and parse all entries using the ResourceReader.

In this demonstration, you can enter the fully qualified path to an image file for the value, and when you click Find Entry, the image will be displayed on the page. This is a small demonstration of how you can quickly display stored graphics back to the user. Do not confuse this particular demonstration with actually saving objects into a resources file, which is also useful, but not part of this demonstration.

The resources file to create (E.g. strings): Make File

Enter a Resource Filename

Serialization
Serialize an object graph
Use ISerializable

Back to Index

Get URL for this page

How Do I...Serialize an Object Graph

This sample shows how to serialize a graph of objects using the serialization infrastructure.

Example

VB Serialize.exe

[Run Sample] | [View Source]

LateBreaking Samples:
- C++

Serialization

Serialize an object graph

Use ISerializable

Back to Index

Get URL for this page

How Do I...Use ISerializable to Customize the Serialization of a Class?

This sample illustrates how to use the **ISerializable** interface to customize the serialization of your own objects. This sample serializes objects based on the mscorlib assembly namespace tree structure. Each object contains a property as the name of the item and a list of children as per the real object in the mscorlib assembly. To demonstrate the customization possible with **ISerializable**, "(SAVED)" is appended to the namespace name to indicate that the values are not live. When the data is read back in, this tag could be modified or removed. Note: This sample requires the System.Runtime.Serialization.Formatters.Soap.dll.

Example

VB ISerializable.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Services
Write a simple Service

Back to Index

Get URL for this page

How Do I...Write a Simple Service Application?

Microsoft Windows NT services enable you to create long running executable applications that run in their own Windows sessions. These services can be automatically started when the computer boots, can be paused and restarted, and typically do not show any user interface. This makes services ideal for use on a server or whenever you need long-running functionality that does not interfere with other users who are working on the same computer. You can also run services in the security context of a specific user account that is different from the logged-on user or the default computer account. For more information about services and Windows sessions, see the "About Services" section in the Platform SDK documentation in MSDN Online.

Using the .NET Framework, you can easily create services by creating an application that is installed as a service. For example, suppose you want to monitor performance counter data and react to threshold values. You could do this by writing an NT service and deploying it to the computer that has the counters you want to monitor.

The following sample is a simple service application. You can install it using the .NET Framework SDK Installation Utility. From a command prompt run:

```
> InstallUtil.exe SimpleService.exe
```

After the installation is completed you can start the Hello World Service from the Services Management Console. After the service starts you should see the service's event log being filled with greetings from the sample service. You can use the Services Management Console to pause, continue, start, and stop the service, and observe how such operations affect the messages written to the event log. After you are done experimenting with the sample, you can uninstall the service by running the following command from the command window (you should stop the service before uninstalling it):

```
> InstallUtil.exe /u SimpleService.exe
```

Example

VB SimpleService.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Timers
Get timer events

Back to Index

Get URL for this page

How Do I...Get timer events without message pump?

Timer allows you to raise an event on a specified interval. This sample illustrates how to use a **Timer** to raise an event every 1000 milliseconds. Run the sample and observe a new "Hello World" message being printed every second. In its simplest form, using a **Timer** involves:

1. Creating a new instance of **Timer**:

```
Dim aTimer As System.Timers.Timer = New System.Timers.Timer()
```

VB

2. Specifying the event handler:

```
AddHandler aTimer.Tick, AddressOf OnTimer
```

VB

3. Specifying how often to raise the event:

```
aTimer.Interval = 1000
```

VB

4. Enabling the component: aTimer.Enabled = true;
5. Handling the event:

```
Public Shared Sub OnTimer(ByVal source As Object, ByVal e As EventArgs)
    Console.WriteLine("Hello World!")
End Sub
```

VB

Example

VB Timer.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Input file: books.xml

Run

Output from reading the file...

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace

How Do I...Read XML data from a stream?

This sample illustrates how to read XML from a stream using the XmlTextReader class. The stream could have come from a variety of sources such as a byte stream from a server, a file, or from a TextReader.

Note: This sample follows on from the How do I...Read XML from a file? topic.

VB ReadXmlStream.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The XmlTextReader has different constructors to specify the location of the XML data. This sample loads the XmlTextReader from a stream. A stream is an abstract representation of an input or output device that is the source of or destination for data (in this case, XML data). You can write to a stream and read from a stream, which is best visualized as a flow of bytes. A stream provides independence from the device and hence requires no program changes if, for instance, the source of a stream changes.

The following sample code creates a StringReader class that builds up an XML string. Since this is purely a byte stream held in memory, you can get the XmlTextReader to parse this stream as XML. This memory stream has no particular specified encoding. The sample then creates an XmlTextReader that parses the stream and displays the resultant XML.

```
    Dim stream as StringReader
    Dim reader as XmlTextReader = nothing

    try
        Console.WriteLine ("Initializing StringReader ...")
        stream = new StringReader ("<?xml version='1.0'?>" & _
                                   "<!-- This file represents a fragment of a book store inventory database -->" & _
                                   "<bookstore>" & _
                                   " <book genre=""autobiography"" publicationdate=""1981"" ISBN=""1-861003-11-0"">" & _
                                   "   <title>The Autobiography of Benjamin Franklin</title>" & _
                                   "   <author>" & _
                                   "       <first-name>Benjamin</first-name>" & _
                                   "       <last-name>Franklin</last-name>" & _
                                   "   </author>" & _
                                   "   <price>8.99</price>" & _
                                   " </book>" & _
                                   " <book genre=""novel"" publicationdate=""1967"" ISBN=""0-201-63361-2"">" & _
                                   "   <title>The Confidence Man</title>" & _
                                   "   <author>" & _
                                   "       <first-name>Herman</first-name>" & _
                                   "       <last-name>Melville</last-name>" & _
                                   "   </author>" & _
                                   "   <price>11.99</price>" & _
                                   " </book>" & _
                                   "  <book genre=""philosophy"" publicationdate=""1991"" ISBN=""1-861001-57-6"">" & _
```

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

```
"   <title>The Gorgias</title>" & _
"   <author>" & _
"       <name>Plato</name>" & _
"   </author>" & _
"   <price>9.99</price>" & _
" </book>" & _
"</bookstore>")

    ' Load the XmlTextReader from the stream
    reader = new XmlTextReader (stream)

    Console.WriteLine ("Processing ...")
    Console.WriteLine ()
    FormatXml(reader)

catch e as Exception
    Console.WriteLine ("Exception: {0} ", e.ToString())

finally
    Console.WriteLine()
    Console.WriteLine("Processing of stream complete.")
    If Not reader Is Nothing
        reader.Close()
    end if
end try
```

VB

If a Stream is supplied as input, the XmlTextReader properly decodes this by wrapping the stream in a StreamReader and calling the SwitchEncoding property according to the XML encoding specified. Also, the XmlResolver is used to resolve external resources needed to correctly parse the input - like Document Type Definitions (DTDs) and schemas. Another method of representing a stream is by using a StreamReader class that implements a reader that reads from a character stream. The following code sample illustrates loading a file named books.xml and then using the XmlTextReader to parse the resulting file.

```
Dim streamreader As StreamReader = New StreamReader ("books.xml")
Console.WriteLine ("File books.xml read sucessfully ...")

' Load the XmlTextReader from the StreamReader
Dim xmlreader As XmlTextReader = New XmlTextReader (streamreader)
```

VB

The FormalXml method in this sample shows how to move to the attribute nodes using the MoveToNextAttribute method when the current node is an element node. This allows you not only to access the node's name and value properties but, because you are at a node context, you can also get properties like the current namespace for the attribute. The following code sample also shows the Format method that this time displays the name and value for the current node.

```
private shared sub FormatXml (reader as XmlTextReader)

    Dim piCount, docCount, commentCount, elementCount as Integer
    Dim attributeCount, textCount, whitespaceCount as Integer

    While reader.Read()
```

```
            Select (reader.NodeType)

            case XmlNodeType.ProcessingInstruction:
                Format (reader, "ProcessingInstruction")
                piCount += 1

            case XmlNodeType.DocumentType:
                Format (reader, "DocumentType")
                docCount += 1

            case XmlNodeType.Comment:
                Format (reader, "Comment")
                commentCount += 1

            case XmlNodeType.Element:
                Format (reader, "Element")
                elementCount += 1
                While reader.MoveToNextAttribute()
                    Format (reader, "Attribute")
                end While

                if (reader.HasAttributes)
                    attributeCount += reader.AttributeCount
                end if

            case XmlNodeType.Text:
                Format (reader, "Text")
                textCount += 1

            case XmlNodeType.Whitespace:
                whitespaceCount += 1

            end Select

        end While

        ' Display the Statistics for the file
        Console.WriteLine ()
        Console.WriteLine("Statistics for stream")
        Console.WriteLine ()
        Console.WriteLine("ProcessingInstruction: " & piCount)
        Console.WriteLine("DocumentType: " & docCount)
        Console.WriteLine("Comment: " & commentCount)
        Console.WriteLine("Element: " & elementCount)
        Console.WriteLine("Attribute: " & attributeCount)
        Console.WriteLine("Text: " & textCount)
        Console.WriteLine("Whitespace: " & whitespaceCount)
    end sub

    private shared sub Format(byref reader as XmlTextReader, nodeType as String)

        ' Format the output
        Console.Write(reader.Depth & " ")
        Console.Write(reader.AttributeCount & " ")

        Dim i as Integer
        for i = 0 to reader.Depth
            Console.Write(Strings.chr(9))
        Next
```

```
            Console.Write(reader.Prefix & nodeType & "<" & reader.Name & ">" & reader.Value)
            Console.WriteLine()

        end sub
```

VB

Summary

1. A stream is an abstract representation of an input or output device that is the source of or destination for data.
2. The XmlTextReader provides constructors to read XML from a string representing an URL or a local file name, a stream, or a TextReader.
3. The XmlTextReader provides constructors to read XML from a file name, a stream, or a TextReader.
4. Attribute nodes can be accessed with the MoveToNextAttribute method that allows you to determine the properties of the attribute node.

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model
Use XML Schema Collection

XML Namespace
XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

How Do I...Read XML from a file?

This sample illustrates how to read XML from a file using the XmlTextReader class. This class provides direct parsing and tokenizing of XML, and implements the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML specifications.

VB ReadXmlFile.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

XmlReader class is the API that provides XML parsing, the XmlTextReader is the implementation designed to handle byte streams.

Typically, you use the XmlTextReader if you need to access the XML as raw data without the overhead of a DOM. Not having to access the DOM results in a faster way to reading XML. For example, an XML document could have a header section used for routing the document for processing elsewhere. The XmlTextReader has different constructors to specify the location of the XML data. This sample loads XML from the books.xml file, as shown in the following code.

```
Dim reader As XmlTextReader = New XmlTextReader ("books.xml")
```

VB

Once loaded, the XmlTextReader moves across the XML data by using the Read method sequentially retrieving the next record from the document. The Read method returns false if there are no more records.

```
Do While (reader.Read())
    ' Do some work here on the data
    ...
Loop
```

VB

To process the XML data, each record has a node type that can be determined from the NodeType property. After the NodeType enumeration returns the node type, the sample tests the node type to see whether it is either an element or document type. If the node is either one of these two types, the sample processes the node using the Name and Value properties to display details about the node. The Name property returns the node name (for instance, the element and attribute names, while the Value property returns the node value (node text) of the current node (record).

```
Do While (reader.Read())
    Select Case reader.NodeType
        Case XmlNodeType.Element ' The node is an Element
            Console.Write("<" + reader.Name)
            while (reader.MoveToNextAttribute()) ' Read attributes
                Console.Write(" " + reader.Name + "='" + reader.Value + "'")
            end while
            Console.Write(">")

        Case XmlNodeType.DocumentType ' The node is a DocumentType
            Console.WriteLine(NodeType & "<" & reader.Name & ">" & reader.Value);
        ...
    End Select
Loop
```

VB

The XmlNodeType returned depends on the XmlReader class being used. For example, the XmlTextReader class never returns an XmlNodeType that is a Document, DocumentFragment, Entity, EndEntity and Notation node. See the .NET Framework Class Library for details on what XmlNodeType are returned by each XmlReader class.

The node types specified for the XmlNodeType are equivalent to the W3C DOM node types, with some extended types required for forward-only reading.

XmlNodeType Enumeration Member	Description

Attribute	An Attribute. Example XML: id='123'. An Attribute node can have the following child node types: Text and EntityReference. The Attribute node does not appear as the child node of any other node type; note that it is not considered a child node of an Element.
CDATA	A CDATA section. Example XML: <![CDATA[my escaped text]]> CDATA sections are used to escape blocks of text that would otherwise be recognized as markup. A CDATASection node cannot have any child nodes. The CDATASection node can appear as the child of the DocumentFragment, EntityReference, and Element nodes.
Comment	A Comment. Example XML: <!-- my comment --> A Comment node cannot have any child nodes. The Comment node can appear as the child of the Document, DocumentFragment, Element, and EntityReference nodes.
Document	A document object, which, as the root of the document tree, provides access to the entire XML document. A Document node can have the following child node types: Element (maximum of one), ProcessingInstruction, Comment, and DocumentType. The Document node cannot appear as the child of any node types.
DocumentFragment	A document fragment. The DocumentFragment node associates a node or subtree with a document without actually being contained within the document. A DocumentFragment node can have the following child node types: Element, ProcessingInstruction, Comment, Text, CDATASection, and EntityReference. The DocumentFragment node cannot appear as the child of any node types.
DocumentType	The document type declaration, indicated by the <!DOCTYPE> tag. Example XML: <!DOCTYPE ...> A DocumentType node can have the following child node types: Notation and Entity. The DocumentType node can appear as the child of the Document node.
Element	An Element. Example XML: <Name> An Element node can have the following child node types: Element, Text, Comment, ProcessingInstruction, CDATA, and EntityReference. The Element node can be the child of the Document, DocumentFragment, EntityReference, and Element nodes.
EndElement	Returned when XmlReader gets to the end of an element. Example XML: </foo>
EndEntity	Returned when XmlReader gets to the end of the entity replacement as a result of a call to ResolveEntity.
Entity	An entity declaration. Example XML: <!ENTITY ...> An Entity node can have child nodes that represent the expanded entity (for example, Text and EntityReference nodes). The Entity node can appear as the child of the DocumentType node.
EntityReference	A reference to an entity. Example XML: &foo; This applies to all entities, including character entity references. An EntityReference node can have the following child node types: Element, ProcessingInstruction, Comment, Text, CDATASection, and EntityReference. The EntityReference node can appear as the child of the Attribute, DocumentFragment, Element, and EntityReference nodes.
None	This is returned by the XmlReader if a Read method has not been called.
Notation	A notation in the document type declaration. Example XML: <!NOTATION ...> A Notation node cannot have any child nodes. The Notation node can appear as the child of the DocumentType node.
ProcessingInstruction	A processing instruction (PI). Example XML: <?pi test?> A PI node cannot have any child nodes. The PI node can appear as the child of the Document, DocumentFragment, Element, and EntityReference nodes.
SignificantWhitespace	Whitespace between markup in a mixed content model, or whitespace within the xml:space= "preserve" scope.

Text	The text content of an element. A Text node cannot have any child nodes. The Text node can appear as the child node of the Attribute, DocumentFragment, Element, and EntityReference nodes.
Whitespace	Whitespace between markup.
XmlDeclaration	The XML declaration node. Example XML: <?xml version='1.0'?> This has to be the first node in the document. It can have no children. It is a child of the root node. It can have attributes that provide version and encoding information.

The Depth property returns the depth of the current node in the XML document and is useful when formatting data. A node at the root level of the document is at depth 0. Combining this depth information with the Name and Value properties, you can create a sample that processes an XML file and formats the output depending on the node type and the depth, gathering statistics as it reads. The following sample code shows how this basic formatting is possible using the Format method. To see the full sample code, see View Source.

```
private shared Sub Format(byref reader as XmlTextReader , nodeType as String)

    ' Format the output
    Console.Write(reader.Depth & " ")
    Console.Write(reader.AttributeCount & " ")

    Dim i as Integer
    for i = 0 to reader.Depth
        Console.Write(Strings.chr(9))
    Next

    Console.Write(nodeType & "<" & reader.Name & ">" & reader.Value)

    'Display the attributes values for the current node
    if (reader.HasAttributes)
        Console.Write(" Attributes:")
        Dim j as Integer
        for j = 0 to reader.AttributeCount - 1
            Console.Write(" [{0}] " & reader(j), j)
        next
    End if

    Console.WriteLine()
End Sub
```

VB

The preceding sample code uses the HasAttributes property to test whether an element node has any attribute nodes, and then uses the node index operators to retrieve each attribute value. This is analogous to a collection of attributes for the node. In addition, the preceding code sample uses the AttributeCount property to return the number of attributes for the current node. Use this approach if all you want are the attribute values and are not concerned with other properties of the attribute nodes (for example, the name of the attribute).

Note: For an alternative approach to accessing attributes by moving to each attribute node, and reading the both the node's name and value, see How Do I...Read XML from a stream?.

The following output results from running this sample with the books.xml file. In this output, the first column is the Depth property and the second column is the AttributeCount property.

```
0 0 XmlDeclaration<xml>version='1.0'
0 0 Comment<> This file represents a fragment of a book store inventory database
0 0 Element<bookstore>
1 3     Element<book> Attributes: [0] autobiography [1] 1981 [2] 1-861003-11-0
2 0             Element<title>
3 0                     Text<>The Autobiography of Benjamin Franklin
2 0             Element<author>
3 0                     Element<first-name>
4 0                             Text<>Benjamin
3 0                     Element<last-name>
4 0                             Text<>Franklin
2 0             Element<price>
3 0                     Text<>8.99
1 3     Element<book> Attributes: [0] novel [1] 1967 [2] 0-201-63361-2
2 0             Element<title>
3 0                     Text<>The Confidence Man
2 0             Element<author>
3 0                     Element<first-name>
4 0                             Text<>Herman
3 0                     Element<last-name>
4 0                             Text<>Melville
2 0             Element<price>
```

```
3 0                    Text<>11.99
1 3     Element<book> Attributes: [0] philosophy [1] 1991 [2] 1-861001-57-6
2 0             Element<title>
3 0                    Text<>The Gorgias
2 0             Element<author>
3 0                    Element<name>
4 0                            Text<>Plato
2 0             Element<price>
3 0                    Text<>9.99

Statistics for books.xml file

XmlDeclaration: 1
ProcessingInstruction: 0
DocumentType: 0
Comment: 1
Element: 18
Attribute: 9
Text: 11
Whitespace: 27
```

Summary

1. The XmlTextReader provides fast, non-cached, forward-only read access to XML data.
2. The XmlTextReader implements the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML specifications.
3. The XmlTextReader provides constructors to read XML from a file, a stream, or a TextReader.
4. The Read method moves the reader sequentially as a node.
5. For element nodes, the value of an attribute can be obtained by using the index operators.
6. Attributes are represented as a node list off the current node and can be discovered through the HasAttributes property.
7. The Depth property reports the depth of the current node and can be useful for formatting. Nodes at the root level are at depth 0.

Input file: XML representation of the stream

Run

Output from reading the stream ...

Input file: books.xml

Run

Output from loading file...

Output file: newbooks.xml

Run

Output from writing file...

DTD for booksDTD.xml: books.dtd
XML document that validates against the books.dtd: booksDTD.xml
XML document that fails validation against the books.dtd: booksDTDFail.xml
Schema for booksSchema.xml: schema.xml
XML document that validates against the schema.xml: booksSchema.xml
XML document that fails validation against the schema.xml: booksSchemaFail.xml

Run

Output from validation...

Input file: twopart.xml

Run

Output...

Input file: books.xml

Run

Output from reading the file...

.NET Framework can handle multiple encodings seamlessly. From an ASP.NET page, the @ResponseEncoding directive can be used so that each individual page can be sent to clients with the encoding specified.

http://samples.gotdotnet.com/quickstart/howto/samples/globalization/multipleencodings/data/windows1252.aspx
Content-Type header value : text/html; charset=Windows-1252
This page is sent to clients encoded in Windows-1252.

http://samples.gotdotnet.com/quickstart/howto/samples/globalization/multipleencodings/data/shiftjis.aspx
Content-Type header value : text/html; charset=shift_jis
This page is sent to clients encoded in Shift_JIS.

http://samples.gotdotnet.com/quickstart/howto/samples/globalization/multipleencodings/data/utf16.aspx
Content-Type header value : text/html; charset=utf-16
< h t m l > < h e a d > < / h e a d > < b o d y > < b > T h i s p a g e i s s e n t t o c l i e n t s e n c o d e d i n U T F - 1 6 . < / b > < b r > < / b o d y > < / h t m l >
http://samples.gotdotnet.com/quickstart/howto/samples/globalization/multipleencodings/data/iso-8859-1.aspx
Content-Type header value : text/html; charset=iso-8859-1
This page is sent to clients encoded in iso-8859-1.

Euro : 123.456,00 €
German Mark : 123.456,00 DM
US Dollar : $123,456.00

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Read a text file?

This sample retrieves information from a text file, and uses an **ArrayList** to display that information for the user.

Opening and reading files for read access is an important part of IO functionality, even if you do not need to write to the file in question. In this example, we open a file for reading. This is useful for reading text files, but inappropriate for binary files. In this example, we use one of many methods available for opening the file. Although many data structures are available for storing information retrieved from the file, an **ArrayList** is the easiest to use. A more detailed analysis of the code follows the example.

VB Anagrams.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

To open a file, you need to manipulate a number of objects found in the **IO** namespace. The first class of interest is the **File** class, which defines the general methods and rules for dealing with files. The **File** class has a number of different methods for opening files, but for this specific scenario, the most appropriate is **OpenText**. This is because this particular method returns a **StreamReader** object, which is ideal for reading lines of information from a standard text file. The following code demonstrates using the **OpenText** method, passing it the name of the file you intend to open. You can include a fully-qualified path when specifying the file.

```
Dim din As StreamReader = File.OpenText ("words.txt")
```

VB

Here are some other ways in which you can open files. Each of these alternative samples could replace the previous sample, although with this code, you have to do a bit more work to translate the return value back to a **StreamReader** object. This is because each of these methods returns a **Stream** object, which can be used to construct a new instance of **StreamReader**. While you can use a **Stream** to read information from a file, a **StreamReader** lessens your workload by providing you with a richer set of methods that you can use to get information from the file that you have opened.

```
'  Use OpenRead which opens a file in read-only mode
Dim din As StreamReader = New StreamReader( File.OpenRead("words.txt") )

'  Use Open, which accepts the FileMode to open the file in.
'  Here we request FileMode.Open, but an alternative would have been Create, or OpenOrCreate
Dim din As StreamReader = New StreamReader( File.Open("words.txt", FileMode.Open) )
```

VB

Once you have opened your file, you will need to get the information from it. As mentioned earlier, the **StreamReader** is good for this because it provides you with the **ReadLine** method which enables you to get one line at a time. Of course, you need to know how the information in your file is stored to interpret it correctly. In this sample, the input file consists of one-word lines of information, making it easier to interpret.

You can place the information you retrieve into a number of datastores. In this example, you could just place each line in a string, but that is not typically what would happen. An array is a possible solution, but there is no way to tell in advance how many lines it is going to read. It would have to keep resizing, or remaking, the array (which would require extra work). The ideal solution in this scenario is to choose an **ArrayList**, a collection on which we can simply use the **Add** method to add more elements. This saves a lot of effort.

The following sample reads each new line from our opened file and places each new piece of information into an **ArrayList**. You can detect the end of the file by checking to see if the last thing that was read was null. If it was, then it has reached the end of the file (note that an empty line will not equal null).

```
Dim str As String
Dim al As ArrayList = New ArrayList() ' make our temporary storage object

' loop through all the rows, stopping when we reach the end of file
Do
        str = din.ReadLine()

        If str <> Nothing Then
                al.Add(str)           ' add each element to our ArrayList
        End If
```

```
    Loop Until str = Nothing
```

VB

Now that you have all your information from the file, you can manipulate it at will. In this example, you can choose to simply print out the information for the user. To do this, you can use a **for each** statement to go through all the elements in the **ArrayList**, printing each item to the **Console** object.

```
    Dim s As String

    For Each s in al
            Console.WriteLine (s)
    Next s
```

VB

Summary

You can read information from a text file in a number of ways. One of the easiest is to use the **File** object's **OpenText** method, in combination with a **StreamReader** object to to retrieve one line of information at a time. It is recommended that you store the information you are retrieving in a temporary store such as an array, or an **ArrayList**.

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Enumerate directories and their contents?

This sample illustrates how to use the **DirectoryInfo** class to create a directory listing. This is a great way to quickly list files for users or reports, or to find basic information about a specific directory. When used in combination with **FileInfo**, you can get all the information you need about files and directories in a specific directory.

VB Directory.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

The ability to look over files and subdirectories for a given directory is essential for many programming tasks. This QuickStart shows you how to use the **FileInfo** and **DirectoryInfo** classes. Why not the **File** and **Directory** classes? This QuickStart intends to display information such as the size, and creation date for a given file. In this case, it is easier to use **FileInfo** and **DirectoryInfo** objects. The **File** class focuses on the use of non-instantiated (static in C#, or shared in Visual Basic) methods, while the **FileInfo** class offers methods based on an instance of a **FileInfo** object. **Directory** and **DirectoryInfo** are similar.

The first task explains how create an instance of the **DirectoryInfo** object (based on the current directory), then find the fully-qualified path of that directory. Next, you can make a table of all the files and directories inside our current directory, so it will print the full path of the current directory at the top of that table. We use the **GetFullPath** method of the Path class to determine the fully-qualified path of our directory.

```
' don't forget your Imports statements at the top of your code...
Imports System
Imports System.IO
' ...

'  make an object which represents our current directory. The dot (".") represents this directory
Dim dir As DirectoryInfo = new DirectoryInfo(".")

'  make the header for our table, letting the user know the path we are in...
Console.WriteLine("Following is a listing for directory: {0}", Path.GetFullPath(dir.ToString()))
```

VB

The objects inside the directory can be files or directories. You can iterate through the directory twice, looking for files first, and directories next. An alternate solution is to use the **FileSystemInfo** object, which can represent a **FileInfo** or a **DirectoryInfo** object. This means you only have to iterate through the collection once. Then, call the **GetFileSystemInfos** method of the object created in the code above, which returns an array of **FileSystemInfo** objects. You can iterate through that array with the **foreach** (**For Each** in Visual Basic), as the following code demonstrates.

```
Dim fsi As FileSystemInfo

'  loop through our array of FileSysteminfo objects
For Each fsi In dir.GetFileSystemInfos()

        '  ... the code in the following samples goes here.

Next fsi
```

VB

So now you can see what kind of object each element of the **FileSystemInfo** array is, and process it accordingly. First test the instance of the **FileSystemObject** to see if it is a file. If it is, create a new instance of the **FileInfo** object to represent it. This enables you to call the methods particular to that kind of object. If you do not do this, you could not call properties such as **Length**, which does not exist in **FileSystemInfo**. In contrast, **CreationTime** is a property of the **FileSystemInfo** object, which is why you can deal with it before you enter your **If** statement, regardless of whether you have a file or a directory.

Notice that you are shortening the name you display to the user, primarily to save screen space. Because there can be file access or other related exceptions, execute this code in a **Try** block, to make sure it does not end abnormally. Put this code together with the code in the previous examples, and you have your program (once inside a method, of course). Note that this only demonstrates how to process a file.

```
Try
        Dim creationTime As DateTime = fsi.CreationTime
        Dim subLength As Integer = 25

        If (fsi is FileInfo) Then '  check to see if the current object is a file...

                Dim f As FileInfo = CType(fsi, FileInfo)
```

```
                    '  this if statement simply ensures that we do not shorten the
                    '  name of the file too much!
                    if f.Name.Length < subLength Then
                           subLength = f.Name.Length
                    End If

                    Dim name As string = f.Name.Substring(0, subLength)
                    Dim size As long = f.Length

                    '  format the output to the screen
                    Console.WriteLine("{0, -25} {1,-12:N0} {2, -12} {3,-20:g}", _
                                  name, (size + " KB").PadLeft(12), "File", creationTime)

            Else   '  it must be a directory
                    '  ...
            End If
    Catch                 '  ignore!!! errors such as 'file being used' and so on...
    End Try
```

VB

Summary

You can working with files and directories using **FileInfo** and **DirectoryInfo**, although you can also use **File** and **Directory** as alternatives. By working with the Info-based objects, you need an instance of the class, but you can easily obtain specific information about a file or directory (such as size, or creation date).

The following example prints the folders and files for the specified directory, to the screen. It uses the FileInfo and DirectoryInfo objects to obtain this information, and uses formatting to make sure it is displayed correctly.

Please enter a directory to list: ==>
Example c:\Program Files

c:\ List Directory

The following example creates a log file in this directory, which can be viewed by users directly. This example reads the file, in order to print the information to the screen, and also updates the file by writing to it when you click the update button. Note that if the file does not exist, it will be created.

Please Enter a message to Add to the log file.

Update File

The following textbox displays the entries in the file.

```
Log Entry : 10:15:13 PM Tuesday, October 16, 2001

-------------------------------------

Log Entry : 10:15:15 PM Tuesday, October 16, 2001

-------------------------------------

Log Entry : 10:15:16 PM Tuesday, October 16, 2001

-------------------------------------

Log Entry : 10:15:16 PM Tuesday, October 16, 2001

-------------------------------------
```

Input and Output
Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Read and write large files?

This sample demonstrates how to write a large file and read it back in. This sample also shows how to time such operations. It accepts a file size (measured in kilobytes) from the user, indicating the size of the file to create. The larger the specified size, the longer the process will take to both write and read the temporary file created. When working with large files, it is a good idea to give consideration to mechanisms to divert a user's attention away from the processing, such as progress indicators, or splash screens.

VB LargeReadWrite.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

This sample uses objects found in the **IO** namespace, but now you will write a large amount to a file. This concept builds on the issues discussed in How Do I...Create a log file?. However, this time you will use Binary readers and writers. This sample will also demonstrate how to measure the length of the process.

The first thing you have to do is determine the size of the file to create. This example asks the user to specify a file size, making sure that they are warned in advance about specifying numbers which are too large. The information comes from the user through a text field on our aspx page. For more details on creating the ASP.NET aspects of our example application, please review the topics listed under Welcome to the ASP.Net QuickStart Tutorial.

Once you have the file size from the user, the next step is to create a file of that size. You could create a file that specifies the size, but you generally do not know in advance how big you are going to make your file (the situation we have here is fictitious, and just for demonstration). Therefore, this example only writes one byte at a time to the file. As you can imagine, this could potentially be a very long process.

There are a few steps required to make the file. First, you need to take the size specifed by the user, and convert it to kilobytes, which is the measurement the example asked the user to specify, so you multiply their number by 1024. Now that you have the correct total, you need to create a reference to the temporary file. This example uses a **FileStream** object. When you make your reference, you need to specify a file name (make sure it is a test file because you do not want to accidently replace a necessary file), and the **FileMode** in which to open the file. Since you are not concerned if the file already exists (just in case you forgot to remove it previously), this example specifies how to open the file in **OpenOrCreate FileMode**. This means that it will open the file if it exists, or create it if it does not. Having made the reference, you can ensure the file is empty by setting the length of the **Stream** to zero, as in the following sample.

```
' a variable to hold the size of the file as specified by the user (fileSize)
Dim origTestSize As double = fileSize

' a variable to hold the actual size of the file we want to make
Dim testSize As double = origTestSize * 1024

Console.WriteLine ("Running test with size {0} KB", origTestSize.ToString())
Console.WriteLine ("This may take a while...")

FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate)
fs.SetLength(0)
```

VB

Now that you have your file ready to populate, you can begin the writing process. In this example, you will make a binary file, so you are going to create a **BinaryWriter** object. You need to make sure that the writer is pointing to the beginning of our file, so use the **Seek** method to move to the front of the file, and then loop through the file, writing the specified number of bytes to the writer. The nature of the information does not matter, since the process is important here, so we are just writing a series of ones. Once finished, you force the changes back into the file by flushing the stream. Note that at the point you write the first byte to the file, you get the current **TickCount** from the **Environment** you are operating in. You will use this to determine when you began the writing process. You can also take a count once the write is over, to determine the total duration for the write process.

```
Dim w As BinaryWriter = New BinaryWriter(fs)      ' create a BinaryWriter

w.BaseStream.Seek(0,SeekOrigin.Begin)             ' seek the beginning of the file
Dim beginWrite As Integer = Environment.TickCount ' get the start time

Dim intCount As Integer

For intCount = 0 To testSize - 1                  ' loop through the file, writing the bytes

        w.Write( CType(1, byte))                  ' do the write
```

```
        intCount = intCount + 1
Next intCount

fs.Flush()

Dim endWrite As Integer = Environment.TickCount   ' get the end time
' ... feed the information back to the user ...
```

VB

Having performed the Write, it is important to let the user know the results, so you will notice that the complete sample feeds back the information to the console. The next important step is to get the information out of the file, to see how long it takes us to read the file. In a fashion similar to writing, create a **BinaryReader**, once again setting the current postiion to the beginning of the file using the **Seek** method. Again take a **TickCount** to determine the start time, and loop through the file, reading each subsequent byte until the end of the file. Once again, flush the information from the **Stream**, and take a count of the ticks at the end of the process. Once you give the information back to the user, it's important to close the **FileStream**, and because you are making a demonstration file, use the **Delete** method of the **File** class to remove the file.

```
Dim r As BinaryReader = New BinaryReader(fs)     ' create a BinaryReader

w.BaseStream.Seek(0,SeekOrigin.Begin)            ' set the file pointer to the beginning
Dim beginRead As Integer = Environment.TickCount ' get the start time

For intCount = 0 To testSize - 1                 ' loop through, reading the bytes

        dummyInt = r.ReadByte()                  ' do the read
        intCount = intCount + 1
Next intCount

fs.Flush()

Dim endRead As Integer = Environment.TickCount   ' determine the end time
' ... feed the information back to the user ...

fs.Close()                                       ' close the file
System.IO.File.Delete("data.bin")                ' delete the file
```

VB

Creating and reading a large file is not much different from creating files of other types. It is good to be aware of the specific methods and properties associated with different input and output techniques, and the classes available to assist you with your **IO** tasks, such as Binary Readers and Writers. Make sure that you keep in mind your current position when dealing with large files.

The following example asks the user to specify a size, measured in Kilobytes, of a file to create on disk. The file is then made, and the time taken to create, and subsequently read the file, is measured. This process demonstrates many useful techniques for dealing with file input and output.

Please specify a file size (in Kilobytes): KB Create File

Although the upper limit on the size you can specify is large, it is suggested you not exceed an entry of 10000 (10 Megabytes).

The following example creates a binary file using the BinaryWriter object, and subsequently reads that information using a BinaryReader. You can select different to write the informatino you want, to file This demonstration is meant to highlight that you need to know how to read a binary file you have written. Once you have written data in one format, you can only read that information in the same format. However, you can write a lot of different datatypes to file. In this demonstration, enter any strings and read them as strings, enter only integer numeric entries for Integers (try floating point numbers and see what happens...), or the words 'false' and 'true' for boolean entries.

Please Select the type of data to save to a binary file...

Boolean
String
Integer

Write/Read File

The following textbox displays the entries in the binary file, or the exception information returned.

Input and Output

Read a text file
List a directory
Create a log file
Read and write large files
Read and write binary data
Watch file system changes

Back to Index

Get URL for this page

How Do I...Watch file system changes?

Use the **FileSystemWatcher** component to monitor a file system and react when changes to it occur. This makes it possible for you to quickly and easily launch business processes when certain files or directories are created, modified, or deleted. For example, suppose you and a group of coworkers are collaborating on a document that is stored on a shared directory on your server. Using the **FileSystemWatcher** component, you can easily program your application to watch for changes to the shared directory. When a change is detected, the component can run processing that notifies each of the group members through email.

This sample illustrates how to use a **FileSystemWatcher** to watch for any changes, renaming, creation, and deletion of any files in the specified directory. The application takes the directory that you wish to watch as the only argument.

Try running the sample as follows:

```
> Watcher.exe c:\
```

Now, open the c:\ directory and try creating, modifying and deleting a file in the directory. Observe the sample application printing an appropriate message to the console window.

In its simplest form, using a FileSystemWatcher involves:

1. Creating a new instance of the component and specifing the directory to watch:

```
Dim watcher As FileSystemWatcher = New FileSystemWatcher()
watcher.Path= "c:\Foo"
```

VB

2. Adding event handler[s]:

```
AddHandler watcher.Changed, AdressOf OnChanged
AddHandler watcher.Created, AdressOf OnChanged
```

VB

3. Specifying what the application should do when a change occurs:

```
Public Sub OnChanged(source As Object, e As FileSystemEventArgs)
  Console.Write("File: {0} {1}", e.FullPath, e.ChangeType)
End Sub
```

VB

4. Enabling the component:

```
watcher.EnableRasingEvents = True
```

VB

Example

VB Watcher.exe

[Run Sample] | [View Source]

LateBreaking Samples:

- C++

Input file: books.xml Output file: updatebooks.xml

Run

Output from reading and saving file...

Input file: books.xml

Run

Output from loading file...

Input file: books.xml

Run

Output from loading file...

Input file: books.xml

Run

Output from loading file...

Input file: books.xml Stylesheet: identity.xsl

Run

Output from reading file...

Input file: books.xml

Run

Output from navigating file...

Input file: books.xml

Run

Output From Query...

XML Data
Overview of XML

XML Readers and Writers
Read XML from a file
Read XML from a stream
Read XML from a URL
Write XML to a file
Apply validation when reading XML
MultipleXmlReader

XML Document Object Model
Load and use the XmlDocument (w3c DOM)
Save XML with the XmlDocument (w3c DOM)
Use XmlDocument class and handle events
Use XmlNodeReader class to get a reader over the XmlDocument

XmlDataDocument
Load and use the XmlDataDocument
Read XML from XmlDataDocument

XPath and XSL Transformations
Navigate with XPathNavigator
Query XML with an XPath Expression
Apply an XSL transformation to XML
Use new XPathExpression class

XML and Relational Data
Use XML and the DataSet class
Create DataSet mappings from an XSD schema
Load a DataSet with XML
Infer DataSet mappings from XML
Save DataSet mappings to an XSD schema file
Save a DataSet as XML

XML Schema
Read and Write a Schema
Use the Xml Schema Object Model

How Do I...Apply an XSL Transformation to XML?

This sample illustrates how to apply an Extensible Stylesheet Language Transformations (XSLT) file to an XML document using the XslTransform class. This class implements the World Wide Web Consortium (W3C) W3C XSL Transformations (XSLT) version 1.0 specification. (For further reference, you might want to refer to W3C W3C Extensible Stylesheet Language (XSL) Version 1.0 specification.)

VB TransformXML.aspx

LateBreaking Samples:
- C++

[Run Sample] | [View Source]

XSLT is an XML-based language designed to transform one XML document into another XML document or an XML document into any other structured document. XSLT uses the XML Path Language (XPath) to perform queries on an XML document in order to specify a particular part of the document. (For an example of how to use XPath to query an XML document, see How Do I...Query XML with an XPath expression?).

Often, the structure of an XML document does not match the structure needed to process the XML data. To transform the existing structure into one that can be processed, you use an XSLT processor and an XSLT style sheet (or XSLT file) that defines how to do the transformation. The XSLT file consists of templates that specify how each node of the source XML document should appear in the destination XML document. Thus, the XSLT processor holds three trees:

1. one tree for the source XML
2. one tree for the destination structure
3. one tree for the XSLT file

The following diagram illustrates the transformation process: taking in an XML document, applying the XSLT file to the nodes of the XML document, and generating another document of the desired format.

Use XML Schema Collection

XML Namespace

XmlNameSpace Manager & NameTable

Back to Index

Get URL for this page

This sample consists of two functions. The first function simply reads an XML document and then transforms the data using an XSLT file. The second function reads an XML document, transforms the data, and then writes a new XML document from the transformed data.

The sample applies these two functions to three sets of documents:

1. The documents in the first set are the XML document, books.xml, and the XSLT file, books.xsl. The XSLT file consists of templates that get all the books in the bookstore element and then, for each book element, get the values for the ISBN attribute and the price element. There is also a template that tells the XSLT processor to start processing from the root ("/") node.
2. The documents in the second set are the XML document, ProcessParametersA.xml, and the XSLT file, StyleSheetGenerator.xsl. In this case, the sample reads the XML document to find out which of two style sheets contained in the XSLT file to use. The sample then generates the specified style sheet.
3. The documents in the third set are the XML document, ProcessParametersB.xml, and the XSLT file, StyleSheetGenerator.xsl. The sample processes these documents in the same way that it processes the second set of documents. However, in this case, the sample generates a different style sheet.

The following code shows the first of the two functions. This is the function that simply reads and transforms data. To do this, the function first creates an XPathDocument object and an XslTransform object. It then uses the Load method to load the XslTransform object with the XSLT file. To initiate the transformation, the function calls the Transform method passing in the XPathDocument.

The transformation process returns an XmlReader that is used to read the transformed data.

Note: As discussed in How Do I...Query XML with an XPath Expression?, the XPathDocument class provides a fast, high-performance cache for XML document processing using XSLT.

```
' args[0] is the xml file and args[1] is the stylesheet file.
public sub ReadTransform(args As String())
    try
        Dim myXPathDocument as XPathDocument = new XPathDocument (args(0))
        Dim myXslTransform as XslTransform = new XslTransform()
        myXslTransform.Load(args(1))
        Dim reader as XmlReader = myXslTransform.Transform(myXPathDocument, nothing)

        FormatXml(reader)

    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())
    end try
end sub
```

VB

Besides returning an XmlReader from a transformation, it is possible supply an XmlWriter to the Transform method. Supplying a writer enables the transformed data to be written to a file or stream. The second of the two functions in this sample shows how to incorporate an XmlWriter into the transformation process. The following code shows how this second function writes the transformed data to a file called transform.xml and then prints this file to the console.

```
' args[0] is the xml file and args[1] is the stylesheet file.
public sub ReadTransformWrite(args As String())
    Dim stream as StreamReader = nothing

    try
        Dim myXPathDocument as XPathDocument = new XPathDocument (args(0))
        Dim myXslTransform as XslTransform = new XslTransform()
        Dim writer as XmlTextWriter = new XmlTextWriter("transform.xml", nothing)
        myXslTransform.Load(args(1))
        myXslTransform.Transform(myXPathDocument, nothing, writer)

        writer.Close()

        stream = new StreamReader ("transform.xml")
        Console.Write(stream.ReadToend())
    catch e as Exception
        Console.WriteLine ("Exception: {0}", e.ToString())
    finally
        If Not stream Is Nothing
            stream.Close()
        end if
    end try
```

```
end sub
```

VB

Although not shown in this sample, you could use the Transform method of XslTransform to load a document and write to file directly. This is a convenience method to load an XML document from file, apply the XSLT file, and write the output to a file. This prevents the user from having to create and load the input source document, and then write to a file stream. This method uses the XPathDocument to transform the XML, as shown in the following code.

```
Dim xsltransform as XslTransform = new XslTransform()
xsltransform.Load("favorite.xsl")
xsltransform.Transform("MyDocument.xml", "TransformResult.xml")
```

VB

Summary

1. XSLT is a language for transforming XML documents into other XML documents using XPath as the query language.
2. Use the Load method of the XslTransform class to load an XSLT style sheet (or XSLT file).
3. The Transform method of the XslTransform class uses an XPathDocument, and either returns an XmlReader or writes the transformation into a supplied XmlWriter or stream.
4. The XslTransform class can transform over any data that implements XPathNavigator and the IXPathNavigable interface.
5. You can combine XSLT files to produce the required structure output format (for example, XML or HTML).
6. XSLT processors do not manipulate documents. They manipulate document structure.

Input file1: books.xml Stylesheet1: books.xsl Input file2: ProcessParametersA.xml Stylesheet2: StyleSheetGenerator.xsl Input file3: ProcessParametersB.xml Stylesheet3: StyleSheetGenerator.xsl

Run

Output from the transform...

Input file: books.xml

Run

Output From Query...

Schema file: books.xsd

Run

Output from creating tables from schema...

Input file: books.xml Schema file: books.xsd

Run

Output from loading XML data with schema...

Schema file: books.xml

Run

Output from infering schema and creating tables...

Schema file: PersonPet.xsd

Run

Output from saving DataSet mappings to a schema...

Run

Output from saving the DataSet relational tables as XML data...

Input file: sample.xsd

Run

Output from loading file...

Input file1: books.xsd Stylesheet1: poschema.xsd

Run

Output ...

XML file: xmlsc.xml XSD file: xmlsc.xsd

Run

Output from reading the input files...

Input file: orders.xml

Run

Output ...

XML Serialization

XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I create classes from an XSD schema?

This topic describes how to create classes that can be used to read and write XML that is defined by an XML Schema Description (XSD) file. Once you understand how to generate the classes for your XSD schema, you should read the "How Do I Read and Write Objects From XML?" topic to find out how to actually load XML into them, and save XML from them.

1. Locate an XSD file. In this example, we will use a purchase order schema, as follows:

PurchaseOrder.XSD

[Run Sample] | [View Source]

2. Use the xsd.exe tool in the SDK to generate source code for the classes you want. Do this by specifying the name of the XSD file as an argument to the executable. The example below creates classes in C# using the purchaseorder.xsd schema and puts them in the **XmlSerializationHowTo** namespace:

```
xsd.exe -c -l:c# -n:XmlSerializationHowTo purchaseorder.xsd
```

3. The tool uses the XSD file to create a set of classes that can read XML files that use the schema that you specified. Here is the code that gets created for the above schema:

VB PurchaseOrder.vb

[Run Sample] | [View Source]

In the code, you will notice additional "Custom Metadata Attributes" above the field declarations. These are used to guide the Xml Serializer classes in properly reading and writing the XML. They are basically hints to the parser. For example, the **Items** field has a custom attribute applied to it to tell the serializer to that the **Items** field should contain an array of item objects which get created when the XML is loaded.

```
<System.Xml.Serialization.XmlArrayAttribute(IsNullable:=false),  _
 System.Xml.Serialization.XmlArrayItemAttribute("item", IsNullable:=false)>  _
Public items() As ItemsItem
```

VB

For detailed information on these attributes, see the reference documentation in the SDK. They are located in the System.Xml.Serialization namespace.

Now that you understand how to generate the classes for your XSD schema, you should read the "How Do I Read and Write Objects From XML?" topic to find out how to actually load XML into them, and save XML from them.

XML Serialization
XSD schema from a class
Class from an XSD schema
Read and Write Objs as XML

Back to Index

Get URL for this page

How do I create an XML Schema Definition (XSD) File From a Class?

This topic describes how to build an XSD file for classes that you save and load from XML using the XML Serialization classes. The XML Serialization classes allow you to serialize an object as XML . Often you will start with an XML Schema Definition (XSD) file, and use the XSD.exe tool to build the classes you save and load from . However, if you started by building your classes first, and then decided to save them as XML, you might want to generate an XML Schema Definition (XSD) file that describes the format that the serializer will use for your objects. This file provides the information that consumers of your persisted classes need to know to properly read them.

1. Locate the assembly and class you want to export. You only need to know the name of the class that corresponds to the root element of the XML document. This example will use the following **PurchaseOrder** type:

   VB PurchaseOrder.vb

   [Run Sample] | [View Source]

   In the code, you will notice additional "Custom Metadata Attributes" above the field declarations. These are used to guide the Xml Serializer classes to properly read and write the XML. They are basically hints to the parser. For example, the **Items** field has a custom attribute applied to it to tell the serializer to that the **Items** field should contain an array of item objects which get created when the XML is loaded. Since they affect what XML is used to save the object, they also affect the schema that is generated:

   ```
   <System.Xml.Serialization.XmlArrayAttribute(IsNullable:=false), _
    System.Xml.Serialization.XmlArrayItemAttribute("item", IsNullable:=false) _
   Public items() As ItemsItem
   ```

   VB

   For detailed information on these attributes, see the reference documentation in the SDK. They are located in the System.Xml.Serialization namespace.

2. Use the xsd.exe tool in the SDK to generate an XSD schema from the class and all classes used by that class. The **-t** switch tells the program only to generate a schema for one particular type in the test.exe file:

   ```
   xsd.exe -t:PurchaseOrder PurchaseOrder.dll
   ```

3. The result is an XSD schema that describes what the XML file will look like when you serialize your object using the XML Serialization classes:

   PurchaseOrder.XSD

   [Run Sample] | [View Source]