## **Getting Started with Replication Programming**

Replication programming contains the following sections.

Topic	Description
Introducing Replication	Describes the benefits and planning
Programming	required to program replication.
<b>Developing Replication Applications</b>	
<u>Using ActiveX Controls</u>	controls, with code samples for each
	control showing various pieces of
	functionality.
Replication ActiveX Control	Describes the objects, collections,
Reference	properties, methods, events, and
	constants available to the ActiveX
	controls.
Developing Replication Merge	Provides an overview of the methods
Conflict Resolvers Through a	available for creating a merge
<u>Custom Resolver</u>	conflict resolver
Programming Replication from	Provides an overview of the methods
Heterogeneous Data Sources	available for making a heterogeneous
	data source a Publisher.
Replication Distributor Interface	Describes the objects, properties,
Reference	methods, and interface structures
	available for the Replication
	Distributor Interface.
Replication Programming Samples	Describes the replication samples
	that are included with Microsoft SQL
	Server™ 2000.

For more information about replication applications that use SQL-DMO, see <u>Developing SQL-DMO Applications</u> and <u>SQL-DMO Reference</u>.

## **Introducing Replication Programming**

As an alternative to using SQL Server Enterprise Manager, you can use the following programming interfaces to implement, administer, and monitor replication:

- Microsoft® ActiveX® controls, when used within custom applications
  that employ Microsoft Visual Basic® or Microsoft Visual C++®,
  provide programmable controls for administering and controlling the
  Snapshot Agent, the Distribution Agent, and the Merge Agent. These
  controls can be used to program activities needed to operate replication.
  Using ActiveX replication controls, you can build custom applications
  to configure and administer replication. Benefits include:
  - Installing a smaller portion of Microsoft SQL Server<sup>™</sup> 2000.
     Because the client has no dependency on the full SQL Server 2000 installation, the user only needs to install Microsoft SQL Server 2000 Desktop Engine.
  - Controlling when replication activity occurs. For example, for an application that provides online and offline capabilities, you may want to expose a Synchronize button. That button can be associated with the merge ActiveX control so that whenever users click the button, the merge ActiveX control connects to the Publisher and activates the Merge Agent for the specified publication, which then merges and synchronizes the data. For more information, see <u>Programming Replication with ActiveX</u> <u>Controls</u>.
- SQL Distributed Management Objects (SQL-DMO) allow you to create custom applications using Visual Basic or C++, which allows you to configure, implement, or maintain your replication topology. SQL-DMO can be used to program replication administration such as configuring distribution and creating subscriptions. The SQL-DMO objects can be used in programming languages such as Visual C++ or Visual Basic.

- The Replication Distributor Interface allows you to replicate data from heterogeneous data sources such as Microsoft Access or Oracle. The Replication Distributor Interface is an OLE DB service provider that allows heterogeneous data sources to publish data to SQL Server Subscribers using snapshot replication or read-only transactional replication. It can be used to develop a custom replication application based on proprietary data sources. For more information, see <u>Programming Replication from Heterogeneous Data Sources</u>.
- Transact-SQL system stored procedures allow you automate some replication tasks, configure replication, and implement subscriptions on multiple servers. Stored procedures are frequently used in scripts that can be run when configuring replication on multiple servers (for example, creating subscriptions to a publication on multiple Subscribers). In most cases, you are better served by using the programming interfaces SQL-DMO and ActiveX replication controls for programming replication rather than writing direct calls to the system stored procedures.

## **Benefits of Programming Replication**

Programming replication allows you to create custom applications with which you can configure and maintain a replication topology. Some benefits of using custom applications include:

- Making it easier for mobile or occasionally connected users to modify data offline and propagate those changes to other locations when they reconnect to the network. The users can enter their changes to the data, and then when connected to the network, use an application to upload and download data changes. The users do not need to have knowledge of Microsoft® SQL Server<sup>TM</sup> or replication to perform these actions.
- Allowing you to program heterogeneous data sources, such as Publishers of data, to SQL Server Subscribers. The Replication Distributor Interface can be programmed to support heterogeneous Publishers for snapshot replication and transactional replication.
- Allowing you to save the initial snapshot on removable media (such as a CD-ROM) and apply the snapshot at the Subscriber from the media, rather than applying the initial snapshot over a slow link.
- Simplifying replication by separating functionality into smaller, reusable pieces that manage and administer the replication process easily from a central location.
- Allowing heterogeneous applications to inherit replication features and act as Publishers so SQL Server Subscribers can gain access to data that is stored on a variety of data sources, including legacy data sources and proprietary data sources.
- Storing replication SQL statements, scripts, and .bcp files (in addition to transactions), and forwarding them to the Subscriber.

• Writing custom applications to resolve merge conflicts that can occur when the same data is modified at multiple places allow the developer to implement specific data or business-decision rules to resolve the conflict. Custom resolvers can be built either as stored procedures or as COM objects written in languages such as Microsoft Visual C++® or Microsoft Visual Basic®. By using merge replication custom conflict resolvers, you can resolve unique business conflicts by writing scripts that can handle any logic required to resolve complex conflict scenarios. For more information, see Merge Replication Conflict Detection and Resolution.

## **Planning for Replication Programming**

When planning to program an application used in replication, decide what replication topology you will use, which replication actions need to be performed programmatically, and which actions will be performed using other Microsoft® SQL Server<sup>TM</sup> 2000 tools.

For the actions that are being controlled programmatically, determine what functionality in your business application will be performed only one time (such as creating the databases, configuring a Publisher, or creating a publication), and what functionality will be performed repeatedly (such as creating subscriptions, synchronizing data between the Publisher and Subscribers, and validating replicated data).

Example

See Also

**Planning for Application Development** 

## **Developing Replication Applications Using ActiveX Controls**

Microsoft® SQL Server™ 2000 provides Microsoft ActiveX® controls that allow custom applications to embed replication functionality. These controls support synchronization and limited administration of push, pull, and anonymous subscriptions. In addition, these controls can be programmed to add, copy, and delete both pull and anonymous subscriptions; create or attach subscription databases; and create new subscriptions to be synchronized. The activity of these controls can be monitored using Replication Monitor in SQL Server Enterprise Manager.

Software developers can use ActiveX replication controls like any standard built-in control. They have been implemented as in-process components and do not have visible user interfaces. The ActiveX replication controls provided are:

- SQL Snapshot control
- SQL Distribution control
- SQL Merge control

These controls are comparable to the replication agents of the same name. The SQL Snapshot control has functions similar to the Snapshot Agent, the SQL Distribution control has functions similar to the Distribution Agent, and the SQL Merge control has functions similar to the Merge Agent.

The Microsoft SQL Server CD-ROM ships with some ActiveX replication control samples. For more information, see <u>Replication ActiveX Control Samples</u>.

#### See Also

Replication ActiveX Control Reference

# Requirements for Using Replication ActiveX Controls in Development Environments

This section explains how to use the SQL Distribution control, SQL Merge control, and SQL Snapshot control in Microsoft® Visual Basic® and Microsoft Visual C++® development environments.

#### **Microsoft Visual Basic**

To use one or more Microsoft ActiveX® replication controls in a Microsoft Visual Basic program: on the **Project/References** menu in the Visual Basic Development Environment, in the **References** dialog box, select the .dll references for the controls you plan to use.

Component	Reference	Library
SQL Distribution	Microsoft SQL Distribution Control	sqldistx.dll
Control	8.0	
SQL Merge Control	Microsoft SQL Merge Control 8.0	sqlmergx.dll
SQL Snapshot Control	Microsoft SQL Snapshot Control 8.0	sqlinitx.dll
Replication Errors	Microsoft SQL Replication Errors	replerrx.dll
	8.0	

If Microsoft SQL Server™ 2000 was installed to the default locations, these component .dlls are located in C:\Program Files\Microsoft SQL Server\80\COM\.

#### Microsoft Visual C++

To use the one or more replication controls in a Microsoft Visual C++ program, include the files from the **Include Files** column of the following table in the appropriate source files of your project. These files are installed on your computer only if **Headers and Libraries** was selected in **Development Tools** during a custom installation of the SQL Server 2000 client tools.

Reference the files from the **Libraries** column of the table in your project or NMAKE file. These files are installed automatically during a typical installation

of the SQL Server 2000 client tools.

Component	Include Files	Libraries
SQL Distribution Control	sqldistx.h repldstx.c	sqldistx.dll
SQL Merge Control	sqlmergx.h replmrgx.c	sqlmergx.dll
SQL Snapshot Control	sqlinitx.h replinix.c	sqlinitx.dll
Replication Errors	replerrx.h	replerrx.dll

If SQL Server 2000 was installed to the default locations, the files in the **Include Files** column are located in C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Include\. The files in the **Libraries** column are located in C:\Program Files\Microsoft SQL Server\80\COM\.

After the controls are referenced, they can be included in the Components toolbar; however, these controls are not user interface controls. Drawing the controls on the form will not instantiate them. The controls need to be instantiated with the NEW keyword. In addition, a variable of the ActiveX object's class can be declared using the WithEvents keyword. This enables the program to receive callbacks from the agents, and the application can cancel the ActiveX object in this callback function. The callback is also available with the **ReplErr** object, although it is a notify callback and not a status callback.

#### See Also

Replication ActiveX Control Reference

# Requirements for Deploying Replication ActiveX Controls

When deploying the Microsoft® ActiveX® replication controls independently of Microsoft SQL Server<sup>TM</sup> 2000, you must include additional files in the installation kit that you use to distribute your application. If you deploy your application using these ActiveX replication controls on a computer where SQL Server 2000 Desktop Engine or SQL Server 2000 client software is also installed, these files will already be present.

If you use Microsoft Visual Studio® and the Deployment Wizard to prepare your installation kit, the wizard will recognize that the files from the **Referenced Libraries** column in the following table are needed, and will include them in the installation kit. The wizard will also indicate that dependency information for these files is not available. Include the files from the **Dependent Libraries** column of the table in your installation kit (click the **Add** button on the Included Files page of the wizard). The paths shown in this table assume SQL Server 2000 was installed to the default locations.

Replication	Referenced	
Control	Libraries	Dependent Libraries
SQL Sqlc	Sqldistx.dll Replerrx.dll	<ul> <li>C:\Program Files\Microsoft SQL Server\80\COM\Rdistcom.dll</li> <li>C:\Program Files\Microsoft SQL Server\MSSQL\Binn\Sqlrepss.dll</li> </ul>
		• \$(WinSysPath)\Sqlwoa.dll
	Sqlmergx.dll Replerrx.dll	• C:\Program Files\Microsoft SQL Server\80\COM\Replprov.dll
		C:\Program Files\Microsoft SQL

		Server\80\COM\Replrec.dll
		• \$(WinSysPath)\Sqlwoa.dll
SQL Snapshot Control	Sqlinitx.dll Replerrx.dll	• C:\Program Files\Microsoft SQL Server\80\COM\Rinitcom.dll
		• \$(WinSysPath)\Sqlwoa.dll

## See Also

Replication ActiveX Control Reference

## **Programming the SQL Snapshot ActiveX Control**

The SQL Snapshot control is implemented as a Microsoft® ActiveX® inprocess component. It provides a way to create snapshots, and is used by all types of replication. The control is comparable to the Snapshot Agent. Its primary class, the **SQLSnapshot** object, creates a snapshot of the specified publication on the specified Distributor.

Dynamic snapshots are supported. There are properties to provide values for the **HOST\_NAME()** and **SUSER\_SNAME()** functions when they appear in the filter criteria of dynamic publications, and to specify the location where the dynamic snapshots are written.

### **Instantiating the SQL Snapshot Control**

This example is a Microsoft® Visual Basic® Sub procedure that creates a snapshot from a publication named **FullSnapPublication** using the database **UE\_PublisherDB** on Publisher **UE\_PUBLISHER** and saves it on Distributor **UE\_DISTRIBUTOR**. Windows Authentication is used for both the Publisher and Distributor connections. The example shows the snapshot code for a snapshot publication. The code for a merge publication requires the ReplicationType property to be set.

#### Sub Main()

Dim oSnapCtl As SQLINITXLib.SQLSnapshot

```
Set oSnapCtl = New SQLINITXLib.SQLSnapshot
oSnapCtl.Publisher = "UE_PUBLISHER"
oSnapCtl.PublisherDatabase = "UE_PublisherDB"
oSnapCtl.PublisherSecurityMode = NT_AUTHENTICATION
oSnapCtl.Publication = "FullSnapPublication"
oSnapCtl.Distributor = "UE_DISTRIBUTOR"
oSnapCtl.DistributorSecurityMode = NT_AUTHENTICATION
oSnapCtl.Initialize
oSnapCtl.Run
```

## oSnapCtl.Terminate End Sub

**Note** Include the call to the **Terminate** method to close connections and release allocated memory.

## **See Also**

**SQLSnapshot Object** 

## **Programming the SQL Distribution ActiveX Control**

The SQL Distribution control is implemented as a Microsoft® ActiveX® inprocess component. It provides a way to move schema and data to Subscribers for snapshot replication and to control the synchronization of subscriptions for transactional replication. The control is comparable to the Distribution Agent. Its primary class, the **SQLDistribution** object, includes the ability to:

- Add, drop, copy, and register subscriptions for synchronization using Windows Synchronization Manager. For more information, see Common SQL Distribution Control and SQL Merge Control Functionality.
- Specify an alternate snapshot folder from which the initial snapshot for a subscription can be applied.
- Specify a working directory to which snapshot files are copied when File Transfer Protocol (FTP) is used to retrieve the initial snapshot.
- Generate a specially formatted subscription file (usually an .msf) in a specified location. These files can be attached to create a synchronized subscription that is registered at the Publisher as part of an attachable subscription database.
- Specify a Data Transformation Services (DTS) package that transforms command rowsets before applying them to a Subscriber.
- Set the Subscriber options to immediate updating or queued updating for changes made at the Subscriber.
- Synchronize push, pull, or anonymous subscriptions in a transactional publication.

For more information, see **SQLDistribution Object**.

#### **Instantiating the SQL Distribution Control**

This code segment demonstrates how a Microsoft Visual Basic® program configures the **SQLDistribution** object to synchronize data with an anonymous SQL Server Subscriber.

In the example, Distributor properties are not set, so the Publisher is also the Distributor. The *SubscriberDatabase* must already exist when this code is executed. Replace the fields in italic with appropriate values.

'SQLDistribution control declaration.

Private mobjDistr As SQLDISTXLib.SQLDistribution
...

Set mobjDistr = New SQLDISTXLib.SQLDistribution

#### With mobjDistr

'Set up the Publisher

.Publisher = "PublisherServer"

.PublisherDatabase = "PublisherDatabase"

.Publication = "PublicationName"

.PublisherSecurityMode = NT\_AUTHENTICATION

'Set up the Subscriber.

.Subscriber = "SubscriberServer"

.SubscriberDatabase = "SubscriberDatabase"

.SubscriberDatasourceType = SQL\_SERVER

.SubscriberSecurityMode = NT\_AUTHENTICATION

'Set up the Subscription.

.SubscriptionType = ANONYMOUS

.SynchronizationType = AUTOMATIC

- 'Synchronize the data.
- .Initialize
- .Run
- .Terminate

End With

**Note** When adding an anonymous subscription, you do not have to call the **AddSubscription** method of the **SQLDistribution** object. If the anonymous subscription does not exist, it will be added automatically when the first synchronization is performed, as in the previous sample.

When using a merge publication, the same example can be used for programming anonymous subscriptions with the **SQLMerge** object. Replace references to the **SQLDistribution** object with references to the **SQLMerge** object.

#### **Creating Anonymous Internet Subscriptions Sample**

ActiveX replication controls can be programmed to synchronize data over the Internet. After the Publisher and Distributor are configured for publishing over the Internet and a publication enabled for anonymous subscriptions is created, an application using an ActiveX replication control can synchronize with the publication data. FTP is used for the snapshot download, both during the initial application and when the subscription is re-initialized. All other synchronizations use SQL packets over TCP/IP to transfer data between Publisher and Subscriber.

#### **Examples**

#### **Distribution Control Using IP Address**

This code segment demonstrates how a Visual Basic program configures the SQL Distribution control to create a Subscriber database and synchronize data with an anonymous SQL Server Subscriber over the Internet. The Distributor and Publisher are reached using an IP address.

In this example, no distributor properties are set, so the Publisher is also used as the Distributor. Replace the fields in italic with appropriate values. The port numbers shown are the numbers that are typically used, but they may also need to be changed.'SQLDistribution control declaration.

Private mobjDistr As SQLDISTXLib.SQLDistribution

. . .

Set mobjDistr = New SQLDISTXLib.SQLDistribution

#### With mobjDistr

- 'Set up the Publisher.
- .Publisher = "PublisherServer"
- .PublisherDatabase = "PublisherDatabase"
- .Publication = "PublicationName"
- .PublisherSecurityMode = DB\_AUTHENTICATION
- .PublisherLogin = "PublisherUserID"
- .PublisherPassword = "PublisherPassword"
- .PublisherAddress = "157.56.17.27,1433"
- .PublisherNetwork = TCPIP SOCKETS

'Set up the Subscriber.

- .Subscriber = "SubscriberServer"
- .SubscriberDatabase = "SubscriberDatabase"
- .SubscriberDatasourceType = SQL\_SERVER
- .SubscriberSecurityMode = NT\_AUTHENTICATION

'Set up the Subscription.

- .SubscriptionType = ANONYMOUS
- .SynchronizationType = AUTOMATIC
- 'Synchronize subscription.
- .Initialize
- .Run
- .Terminate

End With

When using a merge publication, the same example can be used for programming anonymous subscriptions with the SQL Merge control. Replace references to the **SQLDistribution** object with references to the **SQLMerge** object.

#### **Creating Pull Subscriptions to an ODBC Data Source Sample**

The SQL Distribution control can be programmed to synchronize Publishers with ODBC data sources. This feature is one method by which applications can synchronize subscriptions to heterogeneous data sources.

Before using the SQL Distribution control, an ODBC data source name (DSN) must be created at the client computer on which the application hosting the SQL Distribution control is running. If a pull subscription is to be created, the ODBC DSN must also be created at the Distributor if it is a different computer from the client. The DSN at the Distributor must be configured as a Subscriber (for example, by using the **sp\_addsubscriber** stored procedure).

This code segment demonstrates how a Visual Basic program configures the SQL Distribution control to create a pull subscription and synchronize data with an ODBC Subscriber. If an anonymous subscription is used, it is not necessary to call **AddSubscription** or to create the DSN at the Distributor. Replace the fields in italic with appropriate values.

'SQLDistribution object declaration.

Private mobjDistr As SQLDISTXLib.SQLDistribution
...

'Configure the control for an ODBC subscription. Set mobjDistr = New SQLDISTXLib.SQLDistribution

#### With mobjDistr

'Set up the Publisher.

.Publisher = "PublisherServer"

.PublisherDatabase = "PublisherDatabase"

.Publication = "PublicationName"

.PublisherSecurityMode = NT\_AUTHENTICATION

- 'Set up the Distributor.
- .Distributor = "*DistributorServer*"
- .DistributorSecurityMode = NT\_AUTHENTICATION
- 'Set up the Subscriber.
- .Subscriber = "The ODBC DSN"
- .SubscriberDatasourceType = ODBC\_DSN
- .SubscriberSecurityMode = DB\_AUTHENTICATION
- 'Subscriber login information needed only if the
- 'ODBC DSN does not contain login information.
- .SubscriberLogin = "SubscriberUserID"
- .SubscriberPassword = "SubscriberPassword"
- 'Set up the subscription.
- .SubscriptionName = "SubscriptionName"
- .SubscriptionType = PULL
- .SynchronizationType = AUTOMATIC
- 'Create and synchronize the subscription.
- .AddSubscription EXISTING\_DATABASE, NONE
- .Initialize
- .Run
- .Terminate

End With

## **Creating Pull Transactional Subscriptions to an OLE DB Data Source Sample**

You can program the SQL Distribution control to synchronize data with OLE DB data sources. This feature is one method by which applications can synchronize subscriptions to heterogeneous data sources.

Before running the control, a linked server to the OLE DB database must be created at the Distributor. This can be done in SQL Server Enterprise Manager or by using the **sp\_addlinkedserver** stored procedure. A heterogeneous Subscriber using the linked server must be created at the Publisher (for example, by using the **sp\_addsubscriber** stored procedure).

SQL Server 2000 does not support anonymous subscriptions using OLEDB\_DATASOURCE with the **SubscriberDatasourceType** property. You can create OLE DB anonymous subscriptions to instances of SQL Server by setting **SubscriberDatasourceType** to SQL\_SERVER.

This code segment demonstrates how a Visual Basic program can configure the SQL Distribution control to create a pull subscription and synchronize data with an OLE DB Subscriber.Replace the fields in italic with appropriate values.

'Declare the SQLDistribution object.
Private mobjDistr As SQLDISTXLib.SQLDistribution

'Configure the control for an OLE DB subscription. Set mobjDistr = New SQLDISTXLib.SQLDistribution

#### With mobjDistr

'Set up the Publisher.

.Publisher = "PublisherServer"

.PublisherDatabase = "PublisherDatabase"

.Publication = "PublicationName"

.PublisherSecurityMode = NT\_AUTHENTICATION

'Set up the Subscriber.

.Subscriber = "LinkedServerName"

.SubscriberDatasourceType = OLEDB\_DATASOURCE

.SubscriberSecurityMode = DB\_AUTHENTICATION

'Subscriber login information is needed only if the OLE DB

' linked server definition does not include login information.

- .SubscriberLogin = "SubscriberUserID"
- .SubscriberPassword = "SubscriberPassword"
- 'Set up the subscription.
- .SubscriptionName = "SubscriptionName"
- .SubscriptionType = PULL
- .SynchronizationType = AUTOMATIC
- 'Create and synchronize the subscription.
- .AddSubscription EXISTING\_DATABASE, NONE
- .Initialize
- .Run
- .Terminate

End With

#### See Also

Replication and Heterogeneous Data Sources

sp\_addlinkedserver

sp\_addsubscriber

SubscriberDatasourceType Property

## **Programming the SQL Merge ActiveX Control**

The SQL Merge control is implemented as a Microsoft® ActiveX® in-process component. It provides a way to synchronize data in merge subscriptions. It is not used in snapshot replication or transactional replication. Its primary class, the **SQLMerge** object, provides the functionality of the Merge Agent and supports synchronization of push, pull, or anonymous subscriptions to a merge publication. It also includes the options to:

- Add, drop, copy, and register subscriptions for synchronization using Windows Synchronization Manager. For more information, see Common SQL Distribution Control and SQL Merge Control Functionality.
- Apply the initial snapshots to the Subscriber.
- Merge incremental changes that occurred at the Publisher or Subscribers after the initial snapshot was created.
- Reconcile conflicts according to the rules configured, through a COM component custom conflict resolver, or interactively by setting the UseInteractiveResolver property.
- Specify the direction of the synchronization so that it executes only the upload phase, the download phase, or both the upload and download phases.
- Specify an alternate snapshot folder from which the snapshot for a subscription can be applied.
- Specify a client-side working folder to which snapshot files can be copied using FTP.

- Copy a subscription database by generating a specially formatted subscription file in a specified location. These files can be attached to create a synchronized subscription, registered at the Publisher as part of an attachable subscription database.
- Synchronize with a Publisher other than the one at which its subscription originated. This alternate synchronization partner must contain the same schema and data set as the original Publisher.
- Attach a subscription database by specifying Subscriber-side properties only. The **Publisher**, **PublisherDatabase**, **Distributor**, and **Publication** properties do not need to be set while adding attachable subscription databases.

#### **Instantiating the SQL Merge Control**

The SQL Distribution and SQL Merge controls can be used to create a subscription database and a pull subscription, as well as synchronize with the publication data.

This code segment demonstrates how a Microsoft Visual Basic® program can configure the **SQLMerge** object to create a Subscriber database and subscription using the **AddSubscription** method, and then synchronizes data with the SQL Server Publisher.

In the example, Microsoft SQL Server<sup>™</sup> 2000 generates the subscription name. Replace the fields in italic with appropriate values.

'SQLMerge control declaration.

Private mobjMerge As SQLMERGXLib.SQLMerge

Set mobjMerge = New SQLMERGXLib.SQLMerge

With mobjMerge
'Set up the Publisher.
.Publisher = "PublisherServer"

- .PublisherDatabase = "*PublisherDatabase*"
- .Publication = "PublicationName"
- .PublisherSecurityMode = NT\_AUTHENTICATION
- 'Set up the Distributor.
- .Distributor = "*DistributorServer*"
- .DistributorSecurityMode = NT\_AUTHENTICATION
- 'Set up the Subscriber.
- .Subscriber = "SubscriberServer"
- .SubscriberDatabase = "SubscriberDatabase"
- .SubscriberDatasourceType = SQL\_SERVER
- .SubscriberSecurityMode = DB\_AUTHENTICATION
- .SubscriberLogin = "SubscriberUserID"
- .SubscriberPassword = "SubscriberPassword"
- 'Set up the subscription.
- .SubscriptionType = PULL
- .SynchronizationType = AUTOMATIC
- 'Create the database and subscription.
- .AddSubscription CREATE\_DATABASE, NONE
- 'Synchronize the subscription.
- .Initialize
- .Run
- .Terminate

End With

**Note** Subscriptions other than anonymous subscriptions that are added using the **SQLDistribution** or **SQLMerge** objects must be added by explicitly calling the **AddSubscription** method before attempting to initialize and synchronize the subscription for the first time.

Both **SQLDistribution** and **SQLMerge** objects can be used to create a subscription database and a pull subscription, as well as synchronize with the publication data. When using a transactional publication, the previous sample code can be used for creating a Subscriber database and pull subscriptions with the **SQLDistribution** control. Replace references to the **SQLMerge** object with references to the **SQLDistribution** object. For more information about the merge object, see <u>SQLMerge Object</u>.

#### **Creating Anonymous Internet Subscriptions Sample**

ActiveX replication controls can be programmed to synchronize data over the Internet. After the Publisher and Distributor are configured for publishing over the Internet and a publication enabled for anonymous subscriptions is created, an application using an ActiveX replication control can synchronize with the publication data. FTP is used for the snapshot download, both during the initial application and when the subscription is re-initialized. All other synchronizations use SQL packets over TCP/IP to transfer data between Publisher and Subscriber.

This example demonstrates how a Visual Basic program configures the SQL Merge Control to synchronize data for an anonymous subscription over the Internet. The Distributor and Publisher are reached using a Uniform Resource Locator (URL).

The *SubscriberDatabase* must already exist when this code is executed. Because the FTP information has not been provided, the specification of FILETRANSFERFTP for the **FileTransferType** property causes the control to request the FTP information from the Distributor.

A handler for the **Status** event is included. It displays the most recent status message in a label. Replace the fields in italic with appropriate values.

'SQLMerge control declaration.

Private WithEvents mobjMerge As SQLMERGXLib.SQLMerge . . . .

Private Sub RunReplMerge()

## 'Create SQLMerge control. Set mobjMerge = New SQLMERGXLib.SQLMerge

#### With mobjMerge

- 'Set up the Publisher.
- .Publisher = "PublisherServer"
- .PublisherAddress = "publisher.company.com"
- .PublisherNetwork = TCPIP\_SOCKETS
- .PublisherDatabase = "PublisherDatabase"
- .Publication = "PublicationName"
- .PublisherSecurityMode = DB\_AUTHENTICATION
- .PublisherLogin = "*PublisherUserID*"
- .PublisherPassword = "PublisherPassword"

### 'Set up FTP.

.FileTransferType = FILETRANSFERFTP

### 'Set up the Distributor.

- .Distributor = "*DistributorServer*"
- .DistributorAddress = "distributor.company.com"
- .DistributorNetwork = TCPIP\_SOCKETS
- .DistributorSecurityMode = DB\_AUTHENTICATION
- .DistributorLogin = "DistributorUserID"
- .DistributorPassword = "DistributorPassword"

### 'Set up the Subscriber.

- .Subscriber = "SubscriberServer"
- .SubscriberDatabase = "SubscriberDatabase"
- .SubscriberDatasourceType = SQL\_SERVER
- .SubscriberSecurityMode = NT\_AUTHENTICATION

.SubscriptionType = ANONYMOUS

<sup>&#</sup>x27;Set up the subscription.

#### .SynchronizationType = AUTOMATIC

```
'Synchronize the Subscriber.
```

- .Initialize
- .Run
- .Terminate

End With

Exit Sub

**End Sub** 

Private Function mobjMerge\_Status(ByVal Message As String, ByVal As SQLMERGXLib.STATUS\_RETURN\_CODE

'Display most recent status message.

Label1 = Message

**DoEvents** 

**End Function** 

**Note** The URLs *publisher.company.com* and *distributor.company.com* need to be resolvable by an external DNS server. If a listening port other than the default port 1433 is used, it must be explicitly coded. For example, if port 1430 is to be used:

.PublisherAddress = "publisher.company.com,1430"

When using a transactional publication, the same example can be used for synchronizing a Subscriber database with the SQL Distribution control. Replace references to the **SQLMerge** object with references to the **SQLDistribution** object.

#### **Creating and Synchronization Subscriptions to a Jet 4.0 Database**

ActiveX replication controls can be programmed to synchronize data with a Microsoft Jet 4.0 database. This enables applications to synchronize subscriptions to Jet 4.0 databases without having to create an OLE DB data source. The publication must be configured to accept heterogeneous data source

Subscribers. The Subscriber does not need to be configured as a linked server.

This code segment demonstrates how a Visual Basic program configures the SQL Merge control to create a Jet 4.0 database and synchronizes data with it. The **AddSubscription** method need not be used if the database already exists.

**Note** If the publication has not been enabled for heterogeneous Subscribers, the Jet database will be created and the Publisher schema applied, but no data will be copied. No error message or warning is given.

'SQLMerge control declaration.

Private WithEvents mobjMerge As SQLMERGXLib.SQLMerge ....

'Configure the control for a Jet 4.0 database subscription. Set mobjMerge = New SQLMERGXLib.SQLMerge

#### With mobjMerge

'Set up the Publisher.

.Publisher = "PublisherServer"

.PublisherDatabase = "PublisherDatabase"

.Publication = "PublicationName"

.PublisherSecurityMode = NT\_AUTHENTICATION

'Set up the Subscriber.

.Subscriber = "SubscriberServer"

' MDBFileSpecification would be something like C:\ReplDBs\JetPul

.SubscriberDatabasePath = "MDBFileSpecification"

.SubscriberDatasourceType = JET4 DATABASE

.SubscriberSecurityMode = DB\_AUTHENTICATION

' JetDatabaseUserID would be something like "Admin"

.SubscriberLogin = "JetDatabaseUserID"

.SubscriberPassword = "JetDatabasePassword"

- 'Set up the subscription.
- .SubscriptionType = ANONYMOUS
- .SynchronizationType = AUTOMATIC
- 'Synchronize the subscription.
- .Initialize
- .Run
- .Terminate

#### End With

When using a transactional publication, the same example can be used for synchronizing a Subscriber database with the SQL Distribution control. Replace references to the **SQLMerge** object with references to the **SQLDistribution** object.

#### **Providing Status and Handling Cancel Requests**

ActiveX replication controls provide a **Status** event that provides status messages and percent complete during **Initialize**, **Run**, and other replication control methods. These messages can be displayed in the user interface of the application (for example, a label and a progress bar). The event also supports the ability to cancel the control process.

Assume the application hosts a SQL Merge control, and its visible user interface includes these controls.

Control Type	Control Name
Label	lblStatus
Progress Bar	prgStatus
Command Button	cmdCancel

The **Status** event handler updates **lblStatus** and **prgStatus** with the status information. When **cmdCancel** is clicked, a Boolean variable is set. The handler returns a CANCEL notification when it finds the variable set. The **DoEvents** call

should be included to update the visible controls.

If you use a mechanism similar to this, be sure to inhibit subsequent calls to control methods after the cancel request is received.

This code segment demonstrates how a Visual Basic program displays status information and handles a cancel request. Note that the Status callback might be called again after being cancelled. Some operations cannot be cancelled immediately.

Private WithEvents mobjMerge As SQLMERGXLib.SQLMerge Private mblnCancel As Boolean

. . .

Private Sub cmdCancel\_Click()
'Set flag when the Cancel button is clicked.
mblnCancel = True
End Sub

Private Function mobjMerge\_Status(ByVal Message As String, \_
ByVal Percent As Long) As SQLMERGXLib.STATUS\_RETURN
'Display progress and status message.
lblStatus.Caption = Message
prgStatus.Value = Percent

'Cancel if the button was clicked.

If mblnCancel Then mobjMerge\_Status = CANCEL

'Allow screen to update. DoEvents

**End Function** 

#### **Using Error Handling Sample**

ActiveX replication controls provide detailed information about method failures through these mechanisms:

- Returning an error code to the caller of the method. In Microsoft Visual Basic, this is done by raising a runtime error that you can trap in an error handler. In Microsoft Visual C++®, the error code is an HRESULT function return value.
- Referencing a SQLReplErrors collection. Each control has an ErrorRecords property that references a SQLReplErrors collection. To retrieve error information, you can enumerate through each SQLReplError object in the collection.

Usually, any error that is returned to the caller is added to the collection. Collection elements that have an error code equal to 0 are supplemental information such as a stored procedure reference with actual arguments substituted, which applies to the previous collection element. The collection is cleared at each new call to a replication control method to ensure that all elements apply to the most recent call.

This example demonstrates how a Visual Basic program reports detailed error information that might be generated by an ActiveX replication control. Most of the code to set up the control is omitted. The error handler displays the error code and description for the raised error and for each error in the errors collection. When the raised error description is the same as the description of one of the collection elements, it is not duplicated in the display. Both the raised error codes and the duplicate collection element error code are displayed.

Private WithEvents mobjMerge As SQLMERGXLib.SQLMerge Private mobjReplErr As REPLERRXLib.SQLReplError

Private Sub RunReplMerge()
Dim strPhase As String 'setup/initialize/run/terminate

On Error GoTo Error Handler

Set mobjMerge = New SQLMERGXLib.SQLMerge

```
With mobiMerge
    'Set up the SQL Merge control.
    strPhase = "Setup"
    .Publisher = "PublisherServer"
    .PublisherDatabase = "PublisherDatabase"
    .PublisherSecurityMode = NT_AUTHENTICATION
    .SubscriberSecurityMode = NT_AUTHENTICATION
    '<Remainder of properties set here.>
    'Synchronize the subscription.
    strPhase = "Initialize"
    .Initialize
    strPhase = "Run"
    .Run
    strPhase = "Terminate"
    .Terminate
  End With
  Exit Sub
ErrorHandler:
  Dim blnMsgDupl As Boolean 'True: duplicate found in collectio
  Dim strErrMsg
                    As String 'Message buffer
  'Iterate through errors collection.
  For Each mobjReplErr In mobjMerge.ErrorRecords
'The raised error is the same as the collection element; add error code.
    If Err.Description = mobjReplErr.Description Then
       strErrMsg = strErrMsg & vbCrLf & vbCrLf & _
           mobjReplErr.Description & vbCrLf & _
           "Error " & strErrorNumConv(Err.Number)
      blnMsgDupl = True
```

```
'The raised error is not the same as the collection element; add 'descrip'
    Else
       strErrMsg = strErrMsg & vbCrLf & vbCrLf & _
           mobjReplErr.Description
    End If
    'Append error code from collection.
    strErrMsg = strErrMsg & vbCrLf & "Coll. " & _
         strErrorNumConv(mobjReplErr.ErrorNumber)
  Next mobjReplErr
  'Format message, include raised error if duplicate not found.
  strErrMsg = "Error during Merge control " & _
       strPhase & "phase:" & _
       IIf(blnMsgDupl, "", _
         vbCrLf & vbCrLf & Err.Description & vbCrLf & _
         "Error " & strErrorNumConv(Err.Number)) & _
       strErrMsg
  MsgBox strErrMsg, vbExclamation
End Sub
Private Function strErrorNumConv(ByVal lngErrNum As Long) As St
'Convert error number into readable forms,
' hex, and decimal for the low-order word.
  Dim strErrNums
                     As String
  If lngErrNum < 16 And lngErrNum > -16 Then
    strErrNums = CStr(lngErrNum)
  ElseIf lngErrNum < 65536 And lngErrNum > -65536 Then
    strErrNums = "x" & Hex(lngErrNum) & _
         " = " & CStr(lngErrNum)
  Else
```

If this example is run as shown, with part of the control setup missing, the following error is displayed.

Error during Merge control Initialize phase:

The property 'Publication' must be set before initializing the SQL Merg Error Code: x80004005 = x80000000 + 16389 Coll. Code: x7918 = 31000

'The property 'Subscriber' must be set before initializing the SQL Merg Coll. Code: x7918 = 31000

#### See Also

ErrorRecords Property

SQLReplError Object

SQLReplErrors Collection

# Common SQL Distribution Control and SQL Merge Control Functionality

The SQL Distribution control and SQL Merge control have common functionality. This functionality includes:

- Adding subscriptions.
- Registering subscriptions in Microsoft® Windows Synchronization Manager.
- Copying subscriptions.
- Dropping subscriptions.

**Note** This functionality applies only to existing pull and anonymous subscriptions. It cannot be used for push subscriptions.

#### **Adding Subscriptions**

The **SQLDistribution** and **SQLMerge** objects can be configured to add subscriptions using the **AddSubscription** method. When adding a subscription, you can specify to create the Subscriber database, use an existing database, attach a database .mdf file, register an existing subscription, or attach a subscription .msf file.

To add a pull subscription, the Subscriber must already be defined at the Publisher.

Explicitly adding a subscription for an anonymous Subscriber is optional.

#### **Registering Subscriptions**

In addition to adding the subscription, the subscription can be registered in Microsoft Windows Synchronization Manager using the **AddSubscription** method. After being registered, there is no need to use the Microsoft ActiveX®

replication controls to synchronize the Subscriber because you can perform this operation using the Microsoft Windows Synchronization Manager. The Windows Synchronization Manager can be accessed from the **Start** menu by clicking **Programs**, clicking **Accessories**, then clicking **Synchronize**.

To register a subscription in Windows Synchronization Manager, specify SYNC\_MANAGER (instead of the default NONE) for the second argument of **AddSubscription**.

#### **Copying Subscriptions**

The **SQLDistribution** and **SQLMerge** objects can be used to copy a specially prepared database subscription file (typically with an .msf extension) to a Subscriber, attach the subscription, and receive an immediately synchronized subscription at the original Publisher. The **CopySubscription** method creates the .msf file. The **AddSubscription** method with the ATTACH\_SUBSCRIPTION option creates the new subscription from the .msf file.

#### **Dropping Subscriptions**

The **SQLDistribution** and **SQLMerge** objects can be programmed to drop subscriptions using the **DropSubscription** method. When removing the subscription, you can specify if the Subscriber database should be dropped. You can also specify that the subscription be unregistered from Microsoft Windows Synchronization Manager, but not dropped by specifying UNREGISTER\_SUBSCRIPTION for the parameter of **DropSubscription**.

#### See Also

**AddSubscription Method** 

CopySubscription Method

**DBADDOPTION** 

**DBDROPOPTION** 

**DropSubscription Method** 

SUBSCRIPTION\_HOST

# Programming Replication ActiveX Controls Using VBScript

Using the Microsoft® ActiveX® replication controls with Microsoft Visual Basic® Scripting Edition is supported with the following limitations:

- Named constants are not supported. You must specify the actual value represented by the constant; for example, AddSubscription(0, 1) instead of coding AddSubscription(EXISTING\_DATABASE, SYNC\_MANAGER).
- The error handler is not supported. You must use the **ErrorRecords.Count** method to determine if there are failures.

This example demonstrates how an HTML page using Visual Basic Scripting Edition configures the SQL Distribution control to synchronize data for an anonymous subscription over the Internet:

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windov<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>New Page 1</title>
</head>
<body>
<font face="Papyrus">This is only a test.</font>
<object classid="clsid:08B0B2E6-3FB3-11D3-A4DE-00C04F610189"</object>
```

```
<script LANGUAGE="VBScript">
<!--
Sub window_onload()
  Call DoSync()
End Sub
Sub DoSync()
  Dim oErrorObject
  On Error Resume Next
  'Configure the control for an anonymous subscription.
  oSQLDistribution.Publisher = "PUBLISHERNAME"
  oSQLDistribution.PublisherDatabase = "PublishedDBName"
  oSQLDistribution.Publication = "PublicationName"
  oSQLDistribution.PublisherSecurityMode = 0 'DB_AUTHENTICA
  oSQLDistribution.PublisherLogin = "sa"
  oSQLDistribution.PublisherPassword = ""
  oSQLDistribution.SubscriberDatasourceType = 0 ' SQL SERVER
  oSQLDistribution.Subscriber = "SUBSCRIBERNAME"
  oSQLDistribution.SubscriberDatabase = "SubscribingDBName"
  oSQLDistribution.SubscriptionType = 2 'ANONYMOUS
  oSQLDistribution.SubscriberSecurityMode = 0 ' DB_AUTHENTIC
  oSQLDistribution.SubscriberLogin = "sa"
  oSQLDistribution.SubscriberPassword = ""
'Configure the control to access Publisher over the Internet using TCP/
  oSQLDistribution.PublisherNetwork = 1 'TCPIP_SOCKETS
 'Replace 000.00.00.00 with the correct IP address
  oSQLDistribution.PublisherAddress = "000.00.00.00,1433"
  oSQLDistribution.FileTransferType = 1 'FILETRANSFERFTP
  ' Synchronize the data.
```

oSQLDistribution.Initialize

```
If (oSQLDistribution.ErrorRecords.Count > 0) Then
 ' Display each error message.
  For Each oErrorObject in oSQLDistribution.ErrorRecords
   MsgBox oErrorObject.Description, vbCritical, "SQLDist Failure"
   Next
 Else
   oSQLDistribution.Run
   If (oSQLDistribution.ErrorRecords.Count > 0) Then
    ' Display each error message.
    For Each oErrorObject in oSQLDistribution.ErrorRecords
      MsgBox oErrorObject.Description, vbCritical, "SQLDist Failur
      Next
   Else
    oSQLDistribution.Terminate
    If (oSQLDistribution.ErrorRecords.Count > 0) Then
      ' Display each error message.
     For Each oErrorObject in oSQLDistribution.ErrorRecords
       MsgBox oErrorObject.Description, vbCritical, "SQLDist Failur
       Next
     End If
   End If
 End If
End Sub
Sub oSQLDistribution_Status(Message, Percent)
  'Display message here.
End Sub
</script>
</body>
</html>
```

**Note** The properties in this example are the same for both the SQL Merge and

SQL Distribution controls. You can replace references to the **SQLDistribution** object with references to the **SQLMerge** object.

## **Replication ActiveX Control Reference**

Microsoft® ActiveX® replication controls provide a means of programmatically controlling Merge Agent, Distribution Agent, and Snapshot Agent activity using a program written in Microsoft Visual Basic®, Microsoft Visual C++®, or other development languages that support COM.

The Microsoft ActiveX replication controls include:

- SQL Snapshot controls
- SQL Distribution controls
- SQL Merge controls

#### See Also

**Developing Replication Applications Using ActiveX Controls** 

Object Model for ActiveX Controls

Other Replication Control Objects and Collections

**Replication ActiveX Control Properties** 

Replication ActiveX Control Methods

Replication ActiveX Control Events

**Replication ActiveX Control Constants** 

# **Object Model for ActiveX Controls**

The table shows the Microsoft® ActiveX® controls and lists and describes their associated objects and collections.

ActiveX control	Object	Description
SQL Distribution	SQLDistribution	Provides the functionality of
Control	<u>Object</u>	the Distribution Agent and
		supports synchronization of
		push, pull, or anonymous
		subscriptions to a transactional publication.
	SQLReplError Object	Defines an error that occurred
		during processing by an
		ActiveX replication control.
	<u>SQLReplErrors</u>	Contains a <b>SQLReplError</b>
	Collection	object for each error that has
		occurred during the most recent
		method execution of an
		ActiveX replication control.
SQL Merge Control	SQLMerge Object	Provides the functionality of
		the Merge Agent and supports
		synchronization of push, pull,
		or anonymous subscriptions to
		a merge publication.
	<u>AlternateSyncPartners</u>	Returns a reference to the
	<u>Property</u>	AlternateSyncPartners
		collection.
	<u>AlternateSyncPartners</u>	Contains all of the
	Collection	AlternateSyncPartner
		objects.
	<u>AlternateSyncPartner</u>	Defines an alternate
	<u>Object</u>	synchronization partner that a
		subscription (referenced by the
		<b>SQLMerge</b> object) can use if

		the primary Publisher is not available, or if use of the alternate synchronization partner is preferable.
	SQLReplError Object	Defines an error that occurred during processing by an ActiveX replication control.
	SQLReplErrors Collection	Contains a <b>SQLReplError</b> object for each error that has occurred during the most recent method execution of an ActiveX replication control.
SQL Snapshot Control	SQLSnapshot Object	Creates a snapshot of the specified publication on the specified Distributor.
	SQLReplError Object	Defines an error that occurred during processing by an ActiveX replication control.
	SQLReplErrors Collection	Contains a <b>SQLReplError</b> object for each error that has occurred during the most recent method execution of an ActiveX replication control.

## See Also

**How Snapshot Replication Works** 

**How Transactional Replication Works** 

## **SQLSnapshot Object**

The SQL Snapshot control is implemented as a Microsoft® ActiveX® inprocess component. It provides an alternate way, in addition to the Snapshot Agent, to create snapshots. The primary class of the **SQLSnapshot**, the **SQLSnapshot** object, creates a snapshot of the specified publication on the specified Distributor.

Dynamic snapshots are supported. There are properties to provide values for the **HOST\_NAME()** and **sp\_repl\_suser\_sname()** functions when they appear in the filter criteria of dynamic publications, and to specify the location where the dynamic snapshots are written.

### **Properties**

**Distributor Property** 

**DistributorLogin Property** 

**DistributorPassword Property** 

DistributorSecurityMode Property

**DynamicFilterHostName Property** 

**DynamicFilterLogin Property** 

<u>DynamicSnapshotLocation Property</u>

**ErrorRecords Property** 

**LoginTimeout Property** 

**ProfileName Property** 

**Publication Property** 

**Publisher Property** 

PublisherDatabase Property

PublisherLogin Property

**PublisherPassword Property** 

PublisherSecurityMode Property

**QueryTimeout Property** 

ReplicationType Property

#### **Methods**

**Initialize Method** 

Run Method

**Terminate Method** 

#### **Events**

**Status Event** 

#### **Remarks**

To refer to the elements of this object from a Microsoft Visual Basic® application, in the **Project/References** dialog box, select **Microsoft SQL Snapshot Control 8.0**. In a C++ application, include Replinix.c and Sqlinitx.h, and reference Sqlinitx.dll from the project/NMAKE file.

#### See Also

**Developing Replication Applications Using ActiveX Controls** 

## **SQLDistribution Object**

The SQL Distribution control is implemented as a Microsoft® ActiveX® inprocess component. The SQL Distribution control provides an alternate way, in addition to the Distribution Agent, to control synchronization of transactional replication subscriptions. Its primary class, the **SQLDistribution** object, provides the functionality of the Distribution Agent and supports synchronization of push, pull, or anonymous subscriptions to a transactional publication. The **SQLDistribution** object also includes options to:

- Add or drop subscriptions and register them for synchronization using Windows Synchronization Manager.
- Specify an alternate snapshot folder, from which the initial snapshot for a subscription can be applied.
- Specify a working directory to which snapshot files are copied using FTP.
- Generate a specially formatted subscription file in a specified location. These files can then be attached to create a synchronized subscription registered at the Publisher, as part of the attach-and-go functionality.
- Specify a Data Transformation Services (DTS) package that transforms command rowsets before applying them to a Subscriber.
- Set the Subscriber update mode to use immediate or queued updating for changes made at the Subscriber.

#### **Properties**

AltSnapshotFolder Property

Distributor Property

**DistributorAddress Property** 

**DistributorLogin Property** 

**DistributorNetwork Property** 

**DistributorPassword Property** 

<u>DistributorSecurityMode Property</u>

DTSPackageFileName Property

**DTSPackagePassword Property** 

**ErrorRecords Property** 

FileTransferType Property

**FTPAddress Property** 

**FTPLogin Property** 

**FTPPassword Property** 

**FTPPort Property** 

**LoginTimeout Property** 

MaxDeliveredTransactions Property

**ProfileName Property** 

**Publication Property** 

**Publisher Property** 

PublisherAddress Property

PublisherDatabase Property

PublisherLogin Property

**PublisherNetwork Property** 

**PublisherPassword Property** 

PublisherSecurityMode Property

**QueryTimeout Property** 

**SkipErrors Property** 

**Subscriber Property** 

SubscriberDatabase Property

SubscriberDatabasePath Property

SubscriberDatasourceType Property

SubscriberLogin Property

**SubscriberPassword Property** 

SubscriberSecurityMode Property

**SubscriptionName Property** 

SubscriptionType Property

SynchronizationType Property

**UndeliveredCommands Property** 

<u>UndeliveredTransactions Property</u>

**Working Directory Property** 

#### **Methods**

**AddSubscription Method** 

**CopySubscription Method** 

**DropSubscription Method** 

**Initialize Method** 

ReinitializeSubscription Method

Run Method

SetFailoverMode Method

**Terminate Method** 

#### **Events**

#### **Status Event**

#### **Remarks**

To refer to the elements of this object from a Microsoft Visual Basic® application, in the **Project/References** dialog box, select **Microsoft SQL Distribution Control 8.0**. In a C++ application, include Repldstx.c and Sqldistx.h, and reference Sqldistx.dll from the project/NMAKE file.

#### **See Also**

**Developing Replication Applications Using ActiveX Controls** 

## **SQLMerge Object**

The SQL Merge control is implemented as a Microsoft® ActiveX® in-process component. The SQL Merge control provides an alternate way, in addition to the Merge Agent, to synchronize merge subscriptions. Its primary class, the **SQLMerge** object, provides the functionality of the Merge Agent and supports synchronization of push, pull, or anonymous subscriptions to a merge publication. The **SQLMerge** object also includes the options to:

- Add or drop subscriptions and register them for synchronization using Windows Synchronization Manager.
- Specify whether only the upload phase, the download phase, or both phases are run.
- Specify an alternate snapshot folder, from which the initial snapshot for a subscription can be applied.
- Specify a client-side working directory to which snapshot files can be copied using FTP.
- Copy a subscription database by generating a specially formatted subscription file in a specified location. These files can then be attached to create a synchronized subscription registered at the Publisher, as part of the attach-and-go functionality.
- Synchronize with a Publisher other than the one at which its subscription originated. Alternate Publishers must contain the same schema and data set as the original Publisher.
- Access an alternate sync partner when the primary Publisher of the data for the Subscriber is temporarily unavailable, or select an alternate sync partner to use the fastest available connection speed.

- Attach a subscription database by specifying only Subscriber-side properties. The **Publisher**, **PublisherDatabase**, **Distributor**, and **Publication** properties no longer need to be set while adding attachand-go subscriptions.
- Validate that a subscription has the expected data.

#### **Properties**

AlternateSyncPartners Property

AltSnapshotFolder Property

**Distributor Property** 

**DistributorAddress Property** 

**DistributorLogin Property** 

**DistributorNetwork Property** 

**DistributorPassword Property** 

<u>DistributorSecurityMode Property</u>

**DynamicSnapshotLocation Property** 

**ErrorRecords Property** 

**ExchangeType Property** 

FileTransferType Property

**FTPAddress Property** 

**FTPLogin Property** 

FTPPassword Property

**FTPPort Property** 

**HostName Property** 

**LoginTimeout Property** 

**ProfileName Property** 

**Publication Property** 

**Publisher Property** 

**PublisherAddress Property** 

**PublisherChanges Property** 

**PublisherConflicts Property** 

PublisherDatabase Property

PublisherLogin Property

**PublisherNetwork Property** 

**PublisherPassword Property** 

PublisherSecurityMode Property

**QueryTimeout Property** 

**Subscriber Property** 

**SubscriberChanges Property** 

**SubscriberConflicts Property** 

SubscriberDatabase Property

SubscriberDatabasePath Property

SubscriberDatasourceType Property

SubscriberLogin Property

SubscriberPassword Property

SubscriberSecurityMode Property

<u>SubscriptionName Property</u>

**SubscriptionPriority Property** 

 $\underline{SubscriptionPriorityType\ Property}$ 

SubscriptionType Property

SynchronizationType Property

**SyncToAlternate Property** 

**UseInteractiveResolver Property** 

**Validate Property** 

**WorkingDirectory Property** 

#### **Methods**

**AddSubscription Method** 

**CopySubscription Method** 

**DropSubscription Method** 

**Initialize Method** 

ReinitializeSubscription Method

Run Method

**Terminate Method** 

#### **Events**

**Status Event** 

#### Remarks

To refer to the elements of this object from a Microsoft Visual Basic® application, in the **Project/References** dialog box, select **Microsoft SQL Merge Control 8.0**. In a C/C++ application, include Replmrgx.c and Sqlmergx.h, and reference Sqlmergx.dll from the project/NMAKE file.

#### **See Also**

**Developing Replication Applications Using ActiveX Controls** 

# **Other Replication Control Objects and Collections**

The table lists the objects and collections that are used to implement features of the Microsoft® ActiveX® replication controls.

Object/Collection	Description
AlternateSyncPartner Object	Defines an alternate Publisher that a
	subscription can use.
AlternateSyncPartners Collection	Contains the alternate Publishers that
	a subscription can use.
SQLReplError Object	Defines an error that occurred during
	processing by a replication control.
SQLReplErrors Collection	Contains the error objects for a
	replication control.

## AlternateSyncPartner Object

An **AlternateSyncPartner** object defines an alternate Publisher that a subscription referenced by a **SQLMerge** object can use if the primary Publisher is not available, or if use of the alternate Publisher is preferable (for example, it is reachable by a faster or more reliable network connection).

### **Properties**

**Distributor Property** 

FriendlyName Property

**Publication Property** 

**Publisher Property** 

**PublisherDatabase Property** 

#### **Remarks**

To select an alternate synchronization partner, the **Publisher**, **PublisherDatabase**, **Publication**, and **Distributor** properties of the **SQLMerge** object should be set to the corresponding property values of the **AlternateSyncPartner** object before the **Run** method of the **SQLMerge** object is called.

#### See Also

**AlternateSyncPartners Collection** 

**SQLMerge Object** 

## **AlternateSyncPartners Collection**

The **AlternateSyncPartners** collection contains the **AlternateSyncPartner** objects, which define the alternate Publishers that a subscription referenced by a **SQLMerge** object can use if the primary Publisher is not available, or if use of the alternate Publisher is preferable.

### **Applies To**

**SQLMerge Object** 

## **Properties**

**Count Property** 

#### Remarks

The **AlternateSyncPartners** property of a **SQLMerge** object is used to return a reference to the **AlternateSyncPartners** collection.

#### See Also

AlternateSyncPartner Object

AlternateSyncPartners Property

## **SQLReplError Object**

A **SQLReplError** object defines an error that occurred during processing by a Microsoft® ActiveX® replication control.

### **Properties**

**Description Property** 

**ErrorNumber Property** 

**ErrorNumberString Property** 

**Source Property** 

SourceType Property

#### Remarks

**SQLReplError** objects are referenced from the **SQLReplErrors** collection. A reference to this collection is obtained from the **ErrorRecords** property of replication ActiveX controls.

To refer to the elements of this object from a Microsoft Visual Basic® application, in the **Project/References** dialog box, select **Microsoft SQL Replication Errors 8.0**. In a C++ application, include Replerrx.h and reference Replerrx.dll from the project/NMAKE file.

An interface **ISQLReplError** is available directly from the replication ActiveX control components; however, use of the **SQLReplError** object is recommended.

#### See Also

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLReplErrors Collection** 

**SQLS**napshot Object

## **SQLReplErrors Collection**

The **SQLReplErrors** collection contains an **SQLReplError** object for each error that has occurred during the most recent method execution of a Microsoft® ActiveX® replication control.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

## **Properties**

**Count Property** 

#### **Methods**

Add Method

AddReplError Method

#### **Events**

**Notify Event** 

#### Remarks

A reference to the **SQLReplErrors** collection is obtained from the **ErrorRecords** property of replication ActiveX controls. The collection is cleared before the execution of each control method.

To refer to the elements of this collection from a Microsoft Visual Basic® application, in the **Project/References** dialog box, select **Microsoft SQL Replication Errors 8.0**. In a C++ application, include Replerrx.h and reference Replerrx.dll from the project/NMAKE file.

An interface **ISQLReplErrors** is available directly from the replication ActiveX control components; however, using the **SQLReplErrors** collection is recommended.

## **See Also**

**ErrorRecords Property** 

**SQLReplError Object** 

# **Replication ActiveX Control Properties**

The topics in this section define the properties of the Microsoft® ActiveX® replication control objects and collections.

## **Properties**

AlternateSyncPartners Property	PublisherChanges Property
AltSnapshotFolder Property	PublisherConflicts Property
Count Property	PublisherDatabase Property
Description Property	PublisherLogin Property
Distributor Property	PublisherNetwork Property
DistributorAddress Property	PublisherPassword Property
DistributorLogin Property	PublisherRPCLogin Property
DistributorNetwork Property	PublisherRPCPassword Property
DistributorPassword Property	PublisherRPCSecurityMode Property
DistributorSecurityMode Property	PublisherSecurityMode Property
DTSPackageFileName Property	QueryTimeout Property
DTSPackagePassword Property	ReplicationType Property
DynamicFilterHostName Property	SkipErrors Property
DynamicFilterLogin Property	Source Property
DynamicSnapshotLocation Property	SourceType Property
ErrorNumber Property	Subscriber Property
ErrorNumberString Property	SubscriberChanges Property
ErrorRecords Property	SubscriberConflicts Property
ExchangeType Property	SubscriberDatabase Property
FileTransferType Property	SubscriberDatabasePath Property
FriendlyName Property	SubscriberDatasourceType Property
FTPAddress Property	SubscriberLogin Property
FTPLogin Property	SubscriberPassword Property
FTPPassword Property	SubscriberSecurityMode Property
FTPPort Property	SubscriptionName Property

HostName Property	SubscriptionPriority Property
LoginTimeout Property	SubscriptionPriorityType Property
MaxDeliveredTransactions Property	SubscriptionType Property
ProfileName Property	SynchronizationType Property
Publication Property	SyncToAlternate Property
Publisher Property	UndeliveredCommands Property
PublisherAddress Property	UndeliveredTransactions Property
	UseInteractiveResolver Property
	Validate Property
	WorkingDirectory Property

# **AlternateSyncPartners Property**

The **AlternateSyncPartners** property returns a reference to the **AlternateSyncPartners** collection maintained by the **SQLMerge** object.

### **Applies To**

**SQLMerge Object** 

### **Syntax**

object.AlternateSyncPartners

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.

#### **Remarks**

The **AlternateSyncPartners** collection contains the **AlternateSyncPartner** objects, which define the alternate Publishers that a subscription referenced by a **SQLMerge** object can employ if the primary Publisher is not available, or if the use of the alternate Publisher is preferable.

The alternate Publishers must be defined in the publication for the **AlternateSyncPartners** collection to be populated.

### **Data Type**

**AlternateSyncPartners Collection** 

#### **Modifiable**

Read-only

## Prototype (C/C++)

HRESULT AlternateSyncPartners(IAlternateSyncPartners\*\* pVal);

## **AltSnapshotFolder Property**

The **AltSnapshotFolder** property returns or sets the path to the folder that contains the initial snapshot for a subscription.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

### **Syntax**

object.AltSnapshotFolder [= path]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
path	Path to alternate snapshot folder.

#### **Remarks**

The snapshot folder path must include a UNC or FTP designator, for example:

```
objMerge.AltSnapshotFolder = "file:\\Distributor\Backup\Snapshots' objDistr.AltSnapshotFolder = _
```

"ftp://distributor.company.com/backup/snapshots,1433"

It is recommended you use the **sp\_copysnapshot** stored procedure to copy a subscription manually.

### **Data Type**

String

#### **Modifiable**

## Read/write

## Prototype (C/C++)

HRESULT AltSnapshotFolder(BSTR pVal);
HRESULT AltSnapshotFolder(BSTR\* pVal);

## **See Also**

How to Browse and Copy Snapshot Files (Transact-SQL)

# **Count Property**

The **Count** property specifies the number of objects in a collection.

## **Applies To**

AlternateSyncPartners Collection

**SQLReplErrors Collection** 

## **Syntax**

object.Count

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.

## **Data Type**

Long

### **Modifiable**

Read-only

## Prototype (C/C++)

HRESULT Count(long\* cMembers);

# **Description Property**

The **Description** property returns a string that describes the error defined by a **SQLReplError** object.

## **Applies To**

**SQLReplError Object** 

## **Syntax**

object. Description

Part	Description
object	Expression that evaluates to a <b>SQLReplError</b> object.

## **Data Type**

String

### **Modifiable**

Read/write

## Prototype (C/C++)

HRESULT Description(BSTR pVal);

HRESULT Description(BSTR\* pVal);

#### See Also

**ErrorNumber Property** 

**ErrorNumberString Property** 

Source Property

SourceType Property

# **Distributor Property**

The **Distributor** property sets or returns the name of the Distributor server.

### **Applies To**

AlternateSyncPartner Object

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.Distributor [= name]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
name	Name of the Distributor used by the Publisher.

#### Remarks

**Distributor** is a required property for the **SQLSnapshot** object but optional for the other controls.

The Distributor is the instance of Microsoft® SQL Server™ on which the snapshots are stored by default, and on which replication history and statistics are logged. For the SQL Distribution control, it is also the location of the storeand-forward database that contains the replicated transactions.

If the **DistributorNetwork** and **DistributorAddress** properties are specified, they will be used instead of the **Distributor** property when connecting to the Distributor.

If Distributor connection properties are not specified, it is assumed the Publisher and Distributor are on the same instance of SQL Server, and Publisher

connection properties will be used when connecting to the Distributor.

## **Data Type**

String

#### **Modifiable**

Read/write

## Prototype (C/C++)

HRESULT Distributor(BSTR pVal);

HRESULT Distributor(BSTR\* pVal);

## **See Also**

**DistributorAddress Property** 

**DistributorNetwork Property** 

## **DistributorAddress Property**

The **DistributorAddress** property specifies the network address used for connecting to the Distributor when the **DistributorNetwork** property is specified.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

### **Syntax**

object.DistributorAddress [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Network address used when connecting to the Distributor.

#### Remarks

This is a required property only when **DistributorNetwork** is set to a value other than DEFAULT\_NETWORK.

This property is useful when configuring the control to connect to the Distributor without having to use SQL Server Client Network Utility.

For example, if the Distributor is accessed over the Internet, **DistributorNetwork** can be set to TCPIP\_SOCKETS and **DistributorAddress** can be set to a specific IP address.

If the **DistributorNetwork** is TCP/IP\_SOCKETS or MULTI\_PROTOCOL using TCP/IP, the value is in the form of:

'IP address, socket' (i.e. '111.11.11.11,1433")

If the Distributor connection properties are not specified, it is assumed that the Publisher and Distributor are the same instance of Microsoft® SQL Server<sup>TM</sup> and the Publisher connection properties will be used when connecting to the Distributor.

### **Data Type**

String

#### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_DistributorAddress(BSTR \*pVal);

HRESULT put\_DistributorAddress(BSTR newVal);

#### See Also

**Distributor Property** 

**DistributorNetwork Property** 

# **DistributorLogin Property**

The **DistributorLogin** property specifies the login name used when connecting to the Distributor.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

### **Syntax**

object.DistributorLogin [= name]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
name	Name used to log in to the Distributor.

### **Remarks**

This is a required property if **DistributorSecurityMode** is set to **DB\_AUTHENTICATION**.

If the Distributor connection properties are not specified, it is assumed that the Publisher and Distributor are the same instance of Microsoft® SQL Server<sup>TM</sup>, and that the Publisher connection properties are used when connecting to the Distributor.

## **Data Type**

String

#### **Modifiable**

## Read/write

# **Prototype C/C++**

HRESULT get\_DistributorLogin(BSTR \*pVal);
HRESULT put\_DistributorLogin(BSTR newVal);

## **See Also**

**DistributorPassword Property** 

<u>DistributorSecurityMode Property</u>

# **DistributorNetwork Property**

The **DistributorNetwork** property specifies the client Net-Library used when connecting to the Distributor.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

### **Syntax**

object.DistributorNetwork [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	NETWORK_TYPE constant that specifies the client Net-Library used when connecting to the Distributor.

#### Remarks

This option is useful when configuring the control to connect to the Distributor without having to use SQL Server Client Network Utility.

If the value is not DEFAULT\_NETWORK (default), the **DistributorAddress** property must be specified.

### **Data Type**

NETWORK\_TYPE

### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_DistributorNetwork(NETWORK\_TYPE \*pVal);
HRESULT put\_DistributorNetwork(NETWORK\_TYPE newVal);

## **See Also**

**DistributorAddress Property** 

## **DistributorPassword Property**

The **DistributorPassword** property sets or returns the login password used when connecting to the Distributor.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLS**napshot Object

### **Syntax**

object.DistributorPassword [= string]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
string	Password string used when connecting to the Distributor.

#### Remarks

This property is used only when **DistributorSecurityMode** is set to **DB\_AUTHENTICATION**. Specify a NULL password as follows:

objMerge.DistributorPassword = ""

If the Distributor connection properties are not specified, it is assumed that the Publisher and Distributor are the same instance of Microsoft® SQL Server<sup>TM</sup>, and that Publisher connection properties will be used when connecting to the Distributor.

The default is no password.

### **Data Type**

## String

## Modifiable

Read/write

## **Prototype C/C++**

HRESULT get\_DistributorPassword(BSTR \*pVal);

HRESULT put\_DistributorPassword(BSTR newVal);

## See Also

DistributorLogin Property

<u>DistributorSecurityMode Property</u>

## **DistributorSecurityMode Property**

The **DistributorSecurityMode** property sets or returns the security mode used when connecting to the Distributor.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLS**napshot Object

#### **Syntax**

object.DistributorSecurityMode [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list
value	SECURITY_TYPE constant that specifies the mode of security enforced at the Distributor

#### Remarks

If the value is DB\_AUTHENTICATION (default), **DistributorLogin** and **DistributorPassword** will be used when connecting to the Distributor.

NT\_AUTHENTICATION is not supported for **DistributorSecurityMode** unless the Distributor uses the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 operating system. NT\_AUTHENTICATION is not supported for any of **DistributorSecurityMode**, **PublisherSecurityMode**, and **SubscriberSecurityMode** unless the computer on which the replication control is running uses the Windows NT 4.0 or Windows 2000 operating system.

### **Data Type**

SECURITY\_TYPE

### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_DistributorSecurityMode(SECURITY\_TYPE \*pVal);
HRESULT put\_DistributorSecurityMode(SECURITY\_TYPE newVal);

## See Also

**DistributorLogin Property** 

**DistributorPassword Property** 

## **DTSPackageFileName Property**

The **DTSPackageFileName** property returns or sets the name and path of a DTS package used to transform command rowsets before they are applied to a Subscriber.

### **Applies To**

**SQLDistribution Object** 

### **Syntax**

object.DTSPackageFileName [= PackageSpec]

Part	Description
object	Expression that evaluates to a <b>SQLDistribution</b> object.
PackageSpec	Fully qualified path to a DTS package.

### **Remarks**

If the DTS package is stored in Microsoft® SQL Server<sup>TM</sup> or Meta Data Services, the value of this property is the package name. If the package is stored in a file, the property value is the file specification.

## **Data Type**

String

#### **Modifiable**

Read/write

### Prototype (C/C++)

HRESULT DTSPackageFileName(BSTR pVal);

HRESULT DTSPackageFileName(BSTR\* pVal);

# See Also

DTSPackagePassword Property

## **DTSPackagePassword Property**

The **DTSPackagePassword** property returns or sets the **owner** password used to access the DTS package specified by the **DTSPackageFileName** property.

### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.DTSPackagePassword [= string]

Part	Description
object	Expression that evaluates to a <b>SQLDistribution</b> object.
string	Password needed to access the specified DTS package.

#### **Remarks**

The DTS **owner** password must be provided for this property. The required access cannot be obtained with the **user** password.

This property returns the value to which the property was most recently set. It cannot be used to return the actual **owner** password of the DTS package specified by **DTSPackageFileName**, unless it was set to that value previously.

### **Data Type**

String

#### **Modifiable**

Read/write

### Prototype (C/C++)

HRESULT DTSPackagePassword(BSTR pVal);

HRESULT DTSPackagePassword(BSTR\* pVal);

## See Also

DTSPackageFileName Property

## **DynamicFilterHostName Property**

The **DynamicFilterHostName** property returns or sets the host name used when creating a dynamic snapshot. This property provides a value used when the publication is dynamically filtered using the **HOST\_NAME()** function.

### **Applies To**

**SQLSnapshot Object** 

### **Syntax**

object.DynamicFilterHostName [= value]

Part	Description
object	Expression that evaluates to a <b>SQLSnapshot</b> object.
value	Value to be returned by <b>HOST_NAME()</b> in filter
	clauses when creating a dynamic snapshot.

#### Remarks

Use the **DynamicFilterHostName** property to set a value for **HOST\_NAME()** in filtering when a dynamic snapshot is created. For example, if the subset filter clause "rep\_id = HOST\_NAME()" has been specified for an article, and if you set the **DynamicFilterHostName** property to "FBJones" before calling the **Run** method of the **SQLSnapshot** object, only rows having "FBJones" in the **rep\_id** column will be included in the snapshot.

The **DynamicFilterHostName** property applies only to snapshots created for merge publications.

By default, **HOST\_NAME()** evaluates to the name of the computer on which the merge control is executing, unless it is overridden by setting the **DynamicFilterHostName** property.

### **Data Type**

## String

## Modifiable

Read/write

## **Prototype C/C++**

HRESULT get\_DynamicFilterHostName(BSTR \*pVal);
HRESULT put\_DynamicFilterHostName(BSTR newVal);

## See Also

**HOST\_NAME** 

## **DynamicFilterLogin Property**

The **DynamicFilterLogin** property returns or sets the user name used when creating a dynamic snapshot. This property provides a value used when the publication is dynamically filtered using the **SUSER\_SNAME()** function.

### **Applies To**

**SQLSnapshot Object** 

### **Syntax**

object.DynamicFilterLogin [= value]

Part	Description
object	Expression that evaluates to a <b>SQLSnapshot</b> object.
value	Value to be returned by SUSER_NAME( ) in filter
	clauses when creating a dynamic snapshot.

#### Remarks

Use the **DynamicFilterLogin** property to set a value for **SUSER\_SNAME()** in filtering when a dynamic snapshot is created. For example, if the subset filter clause "user\_id = SUSER\_SNAME()" has been specified for an article, and if you set the **DynamicFilterLogin** property to "rsmith" before calling the **Run** method of the **SQLSnapshot** object, only rows having "rsmith" in the **user\_id** column will be included in the snapshot.

The **DynamicFilterLogin** property applies only to snapshots created for merge publications.

### **Data Type**

String

#### **Modifiable**

## Read/write

## **Prototype C/C++**

HRESULT get\_DynamicFilterLogin(BSTR \*pVal);
HRESULT put\_DynamicFilterLogin(BSTR newVal);

## See Also

SUSER\_NAME

# **DynamicSnapshotLocation Property**

The **DynamicSnapshotLocation** property returns or sets the path to a folder to which the files are to be written when a dynamic snapshot is created.

### **Applies To**

**SQLSnapshot Object** 

#### **Syntax**

object.DynamicSnapshotLocation [= path]

Part	Description
object	Expression that evaluates to a <b>SQLSnapshot</b> object.
*	Path of the folder to which the files of a dynamic snapshot are to be written.

#### Remarks

The **DynamicSnapshotLocation** property applies only to snapshots created for Merge publications.

The snapshot folder path must include a UNC or FTP designator, for example:

```
objMerge.DynamicSnapshotLocation = _
   "file:\\Distributor\Backup\Snapshots"
objDistr.DynamicSnapshotLocation = _
   "ftp://distributor.company.com/backup/snapshots,1433"
```

### **Data Type**

String

#### **Modifiable**

Read/write

# **Prototype C/C++**

HRESULT get\_DynamicSnapshotLocation(BSTR \*pVal);
HRESULT put\_DynamicSnapshotLocation(BSTR newVal);

# **ErrorNumber Property**

The **ErrorNumber** property returns the numeric code that identifies the error.

## **Applies To**

**SQLReplError Object** 

### **Syntax**

object.ErrorNumber [= pVal]

Part	Description
object	Expression that evaluates to a <b>SQLReplError</b> object.
pVal	Error number.

#### **Remarks**

The error number is the value assigned by the error source.

### **Data Type**

Long

#### **Modifiable**

Read/write

## Prototype (C/C++)

HRESULT ErrorNumber(long pVal);

HRESULT ErrorNumber(long\* pVal);

### See Also

**Description Property** 

ErrorNumberString Property
Source Property
SourceType Property

# **ErrorNumberString Property**

The **ErrorNumberString** property returns or sets a string representation of the error number.

## **Applies To**

**SQLReplError Object** 

## **Syntax**

object.ErrorNumberString [= string]

Part	Description
object	Expression that evaluates to a <b>SQLReplError</b> object.
string	String representation of the error number.

## **Data Type**

String

#### **Modifiable**

Read/write

### Prototype (C/C++)

HRESULT ErrorNumberString(BSTR pVal);

 $HRESULT\ ErrorNumberString(BSTR*\ pVal);$ 

#### See Also

**Description Property** 

**ErrorNumber Property** 

**Source Property** 

# SourceType Property

## **ErrorRecords Property**

The **ErrorRecords** property returns a reference to the **SQLReplErrors** collection maintained by the replication control.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.ErrorRecords

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.

#### **Remarks**

The **SQLReplErrors** collection is loaded when a control method fails. The error records in the collection are available until the next replication control method is called.

### **Data Type**

**SQLReplErrors Collection** 

#### **Modifiable**

Read-only

### Prototype C/C++

HRESULT ErrorRecords(ISQLReplErrors\*\* pVal);

# See Also

SQLReplError Object

SQLReplErrors Collection

# **ExchangeType Property**

The **ExchangeType** property specifies whether data merges up to the Publisher, down to the Subscriber, or in both directions.

### **Applies To**

**SQLMerge Object** 

### **Syntax**

object.ExchangeType [= value]

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	EXCHANGE_TYPE constant that specifies the
	direction in which data can be merged.

### **Remarks**

The **ExchangeType** property is typically used first to upload all the changes from several Subscribers to a Publisher, and then to download the changes to the Subscribers only after all the uploads have been completed.

BIDIRECTIONAL is the default. You should use a BIDIRECTIONAL synchronization unless you have a specific reason for separating the UPLOAD and DOWNLOAD phases.

### **Data Type**

**EXCHANGE TYPE** 

#### **Modifiable**

Read/write

Prototype C/C++

HRESULT get\_ExchangeType(EXCHANGE\_TYPE \*pVal);
HRESULT put\_ExchangeType(EXCHANGE\_TYPE newVal);

# FileTransferType Property

The **FileTransferType** property returns or sets how the snapshot file is transferred.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

### **Syntax**

object.FileTransferType [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
	Value from the <b>FILE_TRANSFER_TYPE</b> enumeration.

#### **Remarks**

The file transfer options are FTP and UNC (network file copy).

### **Data Type**

FILE TRANSFER TYPE

#### **Modifiable**

Read/write

### Prototype (C/C++)

HRESULT FileTransferType(FILE\_TRANSFER\_TYPE pVal);

HRESULT FileTransferType(FILE\_TRANSFER\_TYPE\* pVal);

# **FriendlyName Property**

The **FriendlyName** property returns or sets a display name by which the association of Publisher, publication, and Distributor that makes up an alternate sync partner can be identified.

## **Applies To**

AlternateSyncPartner Object

### **Syntax**

object.FriendlyName [= name]

Part	Description
object	Expression that evaluates to an <b>AlternateSyncPartner</b>
	object.
name	Friendly name for alternate sync partner.

#### Remarks

An **AlternateSyncPartner** object defines an alternate Publisher that a subscription referenced by the **SQLMerge** object can use if the primary Publisher is not available.

### **Data Type**

String

#### **Modifiable**

Read/write

### Prototype (C/C++)

HRESULT FriendlyName(BSTR pVal);

HRESULT FriendlyName(BSTR\* pVal);

# See Also

SQLMerge Object

## **FTPAddress Property**

The **FTPAddress** property specifies the IP address of the FTP site where publication snapshot files are stored.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.FTPAddress [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Network address of the FTP site.

#### Remarks

**FTPAddress** is an optional property.

If specified, the **FTPLogin**, **FTPPassword**, and **FTPPort** properties are also used.

This option is useful when configuring the control to connect to a Distributor over the Internet, where the Distributor working directory is not directly accessible. Setting this property forces the control to download all publication snapshot files using File Transfer Protocol (FTP) before applying them at the Subscriber.

These properties should not be required when you dynamically determine the FTP login properties by querying the publication. Provided for backward compatibility only: these properties will not be effective when used against Microsoft® SQL Server<sup>TM</sup> 2000 publications unless the Subscriber is a Microsoft SQL Server version 7.0 Subscriber.

**Note** The publication must be enabled for the Internet.

### **Data Type**

String

#### **Modifiable**

Read/write

### **Prototype C/C++**

HRESULT get\_FTPAddress(BSTR \*pVal);

HRESULT put\_FTPAddress(BSTR newVal);

#### See Also

**Distributor Property** 

**DistributorAddress Property** 

**Implementing Replication Over the Internet** 

## FTPLogin Property

The **FTPLogin** property specifies the user name of the FTP site where the publication snapshot files are stored.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.FTPLogin [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	FTP user name of the FTP site.

#### **Remarks**

The default is **anonymous**. These properties should not be required when you dynamically determine the FTP login properties by querying the publication. Provided for backward compatibility only: these properties will not be effective when used against Microsoft® SQL Server<sup>TM</sup> 2000 publications unless the Subscriber is a Microsoft SQL Server version 7.0 Subscriber.

#### **Data Type**

String

#### **Modifiable**

Read/write

**Prototype C/C++** 

HRESULT get\_FTPLogin(BSTR \*pVal);
HRESULT put\_FTPLogin(BSTR newVal);

### See Also

FTPAddress Property

FTPPassword Property

FTPPort Property

## **FTPPassword Property**

The **FTPPassword** property specifies the password of the FTP site where the publication snapshot files are stored.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.FTPPassword [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Password of the FTP site.

#### Remarks

The default is no password. Typically, an e-mail address of the form user@company.com is used for FTP passwords.

These properties should not be required when you dynamically determine the FTP login properties by querying the publication. Provided for backward compatibility only: these properties will not be effective when used against Microsoft® SQL Server<sup>TM</sup> 2000 publications unless the Subscriber is a Microsoft SQL Server version 7.0 Subscriber.

#### **Data Type**

String

#### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_FTPPassword(BSTR \*pVal);
HRESULT put\_FTPPassword(BSTR newVal);

## See Also

FTPAddress Property

FTPLogin Property

**FTPPort Property** 

## **FTPPort Property**

The **FTPPort** property specifies the TCP port number of the FTP site where the publication snapshot files are stored.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.FTPPort [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	TCP port number of the FTP site.

#### **Remarks**

The default value is 21.

These properties should not be required when you dynamically determine the FTP login properties by querying the publication. Provided for backward compatibility only: these properties will not be effective when used against Microsoft® SQL Server<sup>TM</sup> 2000 publications unless the Subscriber is a Microsoft SQL Server version 7.0 Subscriber.

#### **Data Type**

Long

#### **Modifiable**

Read/write

# **Prototype C/C++**

HRESULT get\_FTPPort(long \*pVal);
HRESULT put\_FTPPort(long newVal);

### **See Also**

FTPAddress Property

## **HostName Property**

The **HostName** property returns or sets the host name used when connecting to the Publisher. This property provides a value used when the publication is dynamically filtered using the SQL Server **HOST\_NAME()** function.

#### **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.HostName [= value]

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	Value to be returned by <b>HOST_NAME()</b> in filter
	clauses.

#### Remarks

Use the **HostName** property to set a value for **HOST\_NAME()** in dynamic filtering. For example, if the subset filter clause "rep\_id = HOST\_NAME()" has been specified for an article, and if you set the **HostName** property to "FBJones" before calling the **Run** method of the **SQLMerge** object, only rows having "FBJones" in the **rep\_id** column will participate in the replication of that article.

#### **Data Type**

String

#### Modifiable

Read/write

**Prototype C/C++** 

HRESULT get\_HostName(BSTR \*pVal);
HRESULT put\_HostName(BSTR newVal);

## See Also

**HOST\_NAME** 

## **LoginTimeout Property**

The **LoginTimeout** property specifies the maximum number of seconds to wait for connections to be established.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.LoginTimeout [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Number of seconds for connections to be established.

### Remarks

The default is 15 seconds. A value of 0 specifies an indefinite wait.

### **Data Type**

Integer

### Modifiable

Read/write

### **Prototype C/C++**

HRESULT get\_LoginTimeout(short \*pVal);

HRESULT put\_LoginTimeout(short newVal);

## **MaxDeliveredTransactions Property**

The **MaxDeliveredTransactions** property sets or returns the maximum number of transactions to be downloaded to Subscribers during each **Run** operation.

#### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.MaxDeliveredTransactions [= value]

Part	Description
object	Expression that evaluates to a <b>SQLDistribution</b> object.
value	Maximum number of transactions to be downloaded.

#### **Remarks**

The default is 0, which means that all available transactions are delivered. Other values can be used to control the number of transactions downloaded during a **Run** operation.

#### **Data Type**

Integer

#### **Modifiable**

Read/write

#### Prototype C/C++

HRESULT get\_MaxDeliveredTransactions(long \*pVal);

HRESULT put\_MaxDeliveredTransactions(long newVal);

## See Also

<u>UndeliveredCommands Property</u>

<u>UndeliveredTransactions Property</u>

## **ProfileName Property**

The **ProfileName** property returns or sets the name of the agent profile at the Distributor to be used by the replication control.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.ProfileName [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Agent profile name.

### Remarks

If not specified, the default profile for the agent type is used.

### **Data Type**

String

#### **Modifiable**

Read/write

### **Prototype C/C++**

HRESULT get\_ProfileName(BSTR \*pVal);

HRESULT put\_ProfileName(BSTR newVal);

## See Also

**Agent Profiles** 

How to create a replication agent profile (Enterprise Manager)

## **Publication Property**

The **Publication** property returns or sets the name of the publication.

#### **Applies To**

AlternateSyncPartner Object

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.Publication [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Publication name.

#### Remarks

This is a required property. An error is raised if it is set to an empty string.

If using the SQL Distribution Control to synchronize multiple publications configured to use a shared agent, the value must be ALL. For more information about shared agents, see <a href="mailto:specialcon">specialcon</a>.

#### **Data Type**

String

#### **Modifiable**

Read/write

# **Prototype C/C++**

HRESULT get\_Publication(BSTR \*pVal);
HRESULT put\_Publication(BSTR newVal);

### **See Also**

<u>Publisher Property</u>

PublisherDatabase Property

## **Publisher Property**

The **Publisher** property returns or sets the Publisher name where the publication resides.

#### **Applies To**

AlternateSyncPartner Object

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.Publisher [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Publisher name.

#### **Remarks**

If the **PublisherNetwork** and **PublisherAddress** properties are specified, they will be used instead of the **Publisher** property when connecting to the Publisher; otherwise, **Publisher** is a required property.

#### **Data Type**

String

#### **Modifiable**

Read/write

# **Prototype C/C++**

HRESULT get\_Publisher(BSTR \*pVal);
HRESULT put\_Publisher(BSTR newVal);

### **See Also**

Publication Property

Publisher Database Property

## **PublisherAddress Property**

The **PublisherAddress** property specifies the network address used when connecting to the Publisher when the **PublisherNetwork** property is specified.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.PublisherAddress [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Network connection string.

#### **Remarks**

This is a required property when **PublisherNetwork** is set to a value other than DEFAULT\_NETWORK.

This property is useful when configuring the control to connect to the Publisher without having to use SQL Server Client Network Utility.

For example, if the Publisher is to be accessed over the Internet, **PublisherNetwork** can be set to TCPIP\_SOCKETS and **PublisherAddress** can be set to a specific IP address.

If the **PublisherNetwork** is TCP/IP\_SOCKETS or MULTI\_PROTOCOL over TCP/IP, the value is in the form of:

'**IP** address, socket' (i.e. '111.11.11.11,1433")

#### **Data Type**

### String

### Modifiable

Read/write

## **Prototype C/C++**

HRESULT get\_PublisherAddress(BSTR \*pVal);
HRESULT put\_PublisherAddress(BSTR newVal);

### See Also

**PublisherNetwork Property** 

# **PublisherChanges Property**

The **PublisherChanges** property returns the total number of Publisher changes applied to the Subscriber during the last **Run** operation.

#### **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.PublisherChanges [= value]

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	Total number of Publisher rows inserted, deleted, and
	updated.

### **Data Type**

Long

#### **Modifiable**

Read-only

### Prototype C/C++

HRESULT get\_PublisherChanges(long \*pVal);

### **See Also**

**PublisherConflicts Property** 

SubscriberChanges Property

# **PublisherConflicts Property**

The **PublisherConflicts** property specifies the total number of conflicts that occurred at the Publisher during the last **Run** operation.

#### **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.PublisherConflicts [= value]

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	Number of conflicts that occurred.

### **Data Type**

Long

#### **Modifiable**

Read-only

#### Prototype C/C++

HRESULT get\_PublisherConflicts(long \*pVal);

#### See Also

SubscriberConflicts Property

**PublisherChanges Property** 

# **PublisherDatabase Property**

The **PublisherDatabase** property returns or sets the name of the publication database.

### **Applies To**

AlternateSyncPartner Object

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.PublisherDatabase [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Publication database name.

#### **Remarks**

This is a required property. An error is raised if it is set to an empty string.

### **Data Type**

String

#### **Modifiable**

Read/write

#### Prototype C/C++

HRESULT get\_PublisherDatabase(BSTR \*pVal);
HRESULT put\_PublisherDatabase(BSTR newVal);

# See Also

**Publication Property** 

**Publisher Property** 

# **PublisherLogin Property**

The **PublisherLogin** property sets or returns the login name used when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.PublisherLogin [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	User name used to log in to the Publisher.

#### **Remarks**

This is a required property if **PublisherSecurityMode** is set to DB\_AUTHENTICATION.

### **Data Type**

String

#### **Modifiable**

Read/write

#### Prototype C/C++

HRESULT get\_PublisherLogin(BSTR \*pVal);

HRESULT put\_PublisherLogin(BSTR newVal);

# See Also

PublisherPassword Property

PublisherSecurityMode Property

# **PublisherNetwork Property**

The **PublisherNetwork** property specifies the client Net-Library used when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.PublisherNetwork [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
	NETWORK_TYPE constant that specifies the client Net-Library to use.

#### **Remarks**

This option is useful when configuring the control to connect to the Publisher without having to use SQL Server Client Network Utility.

If the value is not DEFAULT\_NETWORK (default), the **DistributorAddress** property must be specified.

#### **Data Type**

NETWORK\_TYPE

#### **Modifiable**

Read/write

# **Prototype C/C++**

HRESULT get\_PublisherNetwork(NETWORK\_TYPE \*pVal);
HRESULT put\_PublisherNetwork(NETWORK\_TYPE newVal);

### **See Also**

**Publisher Property** 

PublisherAddress Property

# **PublisherPassword Property**

The **PublisherPassword** property sets or returns the login password used when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.PublisherPassword [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Password used to connect to the Publisher.

#### **Remarks**

The default is no password.

This is a required property if **PublisherSecurityMode** is set to DB\_AUTHENTICATION.

### **Data Type**

String

#### **Modifiable**

Read/write

**Prototype C/C++** 

HRESULT get\_PublisherPassword(BSTR \*pVal);
HRESULT put\_PublisherPassword (BSTR newVal);

### See Also

PublisherLogin Property

PublisherSecurityMode Property

# **PublisherRPCLogin Property**

The **PublisherRPCLogin** property determines the login name used by the immediate-updating Subscriber trigger RPC when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.PublisherRPCLogin [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Login to be used when connecting to the Publisher.

#### **Remarks**

This property is relevant only to immediate-updating subscriptions.

#### **Data Type**

String

#### Modifiable

Read/write

#### **Prototype C/C++**

HRESULT get\_PublisherRPCLogin(BSTR \*pVal);

HRESULT put\_PublisherRPCLogin(BSTR newVal);

#### **See Also**

# PublisherRPCPassword PublisherRPCSecurityMode

# **PublisherRPCPassword Property**

The **PublisherRPCPassword** property determines the password used by the immediate-updating Subscriber trigger RPC when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.PublisherRPCPasswrod [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Password string used when connecting to the Publisher.

#### **Remarks**

This property is relevant only to immediate-updating subscriptions.

#### **Data Type**

String

#### **Modifiable**

Read/write

#### **Prototype C/C++**

HRESULT get\_PublisherRPCPassword(BSTR \*pVal);

HRESULT put\_PublisherRPCPassword(BSTR newVal);

#### **See Also**

PublisherRPCLogin
PublisherRPCSecurityMode

# **PublisherRPCSecurityMode Property**

The **PublisherRPCSecurityMode** property determines the security mode used by the immediate-updating Subscriber trigger RPC when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.PublisherRPCPassword [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Specifies the mode of security enforced during immediate-updating Subscriber RPC login at the Publisher. The value is from the REPLRPC_SECURITY_TYPE enumeration.

#### **Remarks**

This property is relevant only to immediate-updating subscriptions.

### **Data Type**

Long, enumerated.

#### **Modifiable**

Read/write

#### Prototype C/C++

HRESULT get\_PublisherRPCPassword(REPLRPC\_SECURITY\_TYPE \*pVal);

 $HRESULT\ put\_PublisherRPCPassword(REPLRPC\_SECURITY\_TYPE\ newVal);$ 

# See Also

<u>PublisherRPCLogin</u>

<u>PublisherRPCPassword</u>

# **PublisherSecurityMode Property**

The **PublisherSecurityMode** property sets or returns the security mode when connecting to the Publisher.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLS**napshot Object

#### **Syntax**

object.PublisherSecurityMode [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	SECURITY_TYPE constant that specifies the Publisher security mode.

#### Remarks

If the value is DB\_AUTHENTICATION (default), **PublisherLogin** and **PublisherPassword** will be used when connecting to the Publisher.

NT\_AUTHENTICATION is not supported for **PublisherSecurityMode** unless the Publisher runs on the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 operating system. NT\_AUTHENTICATION is not supported for any of **DistributorSecurityMode**, **PublisherSecurityMode**, and **SubscriberSecurityMode** unless the computer on which the replication control is running uses the Windows NT 4.0 or Windows 2000 operating system.

#### **Data Type**

SECURITY\_TYPE

#### Modifiable

Read/write

### **Prototype C/C++**

HRESULT get\_PublisherSecurityMode(SECURITY\_TYPE \*pVal);
HRESULT put\_PublisherSecurityMode(SECURITY\_TYPE newVal);

### See Also

PublisherLogin Property

**PublisherPassword Property** 

# **QueryTimeout Property**

The **QueryTimeout** property returns or sets the number of seconds allowed for internal queries to complete.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.QueryTimeout [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Number of seconds allowed for internal queries to be returned.

#### **Remarks**

The default value is 30.

A value of 0 means to wait indefinitely.

### **Data Type**

Integer

#### **Modifiable**

Read/write

**Prototype C/C++** 

HRESULT get\_QueryTimeout(short \*pVal);
HRESULT put\_QueryTimeout(short newVal);

# **ReplicationType Property**

The **ReplicationType** property returns or sets the type of replication for which the snapshot is to be used.

#### **Applies To**

**SQLSnapshot Object** 

#### **Syntax**

object.ReplicationType [= value]

Part	Description
object	Expression that evaluates to a <b>SQLSnapshot</b> object.
value	Value from the <b>REPLICATION_TYPE</b> enumeration.

#### **Remarks**

The types of replication are transactional replication and merge replication.

#### **Data Type**

**REPLICATION\_TYPE** 

#### Modifiable

Read/write

#### Prototype (C/C++)

HRESULT ReplicationType(REPLICATION\_TYPE pVal);

HRESULT ReplicationType(REPLICATION\_TYPE\* pVal);

# **SkipErrors Property**

The **SkipErrors** property returns or sets a list of the errors to be skipped.

#### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.SkipErrors [= list]

Part	Description
object	Expression that evaluates to a <b>SQLDistribution</b> object.
list	List of the errors that are to be skipped by the control.

#### **Remarks**

The SkipErrors list has this format:

<native\_error\_id1>:<native\_error\_id2>:<native\_error\_id3>....

Native error IDs are the error numbers from the underlying database. Only the errors that are encountered when applying replication transactions at the Subscriber can be skipped.

#### **Data Type**

String

#### **Modifiable**

Read/Write

#### Prototype C/C++

HRESULT get\_SkipErrors(BSTR \*pList);

HRESULT put\_SkipErrors(BSTR pList);

# See Also

**Handling Errors and Messages in Applications** 

# **Source Property**

The **Source** property returns or sets the name of the source where the error occurred.

### **Applies To**

**SQLReplError Object** 

#### **Syntax**

object.Source [= pVal]

Part	Description
object	Expression that evaluates to a <b>SQLReplError</b> object.
pVal	Name of the source where the error occurred.

### **Data Type**

String

**Modifiable** 

Read/write

Prototype (C/C++)

HRESULT Source(BSTR pVal);

HRESULT Source(BSTR\* pVal);

#### See Also

**Description Property** 

**ErrorNumber Property** 

**ErrorNumberString Property** 

SourceType Property

# **SourceType Property**

The **SourceType** property specifies the type of the source of the error information.

### **Applies To**

**SQLReplError Object** 

#### **Syntax**

object.SourceType [= value]

Part	Description
object	Expression that evaluates to a <b>SQLReplError</b> object.
value	Value from the <b>ERRORSOURCE_TYPE</b>
	enumeration.

### **Data Type**

**ERRORSOURCE TYPE** 

#### **Modifiable**

Read/write

#### Prototype (C/C++)

HRESULT SourceType(ERRORSOURCE\_TYPE pVal);

HRESULT SourceType(ERRORSOURCE\_TYPE\* pVal);

#### See Also

**Description Property** 

**ErrorNumber Property** 

**ErrorNumberString Property** 

# Source Property

# **Subscriber Property**

The **Subscriber** property specifies the name of the Subscriber.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

## **Syntax**

object.Subscriber [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	Subscriber name.

#### **Remarks**

This is a required property.

## **Data Type**

String

#### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_Subscriber(BSTR \*pVal);

HRESULT put\_Subscriber(BSTR newVal);

## See Also

# SubscriberDatasourceType Property

# **SubscriberChanges Property**

The **SubscriberChanges** property specifies the total number of Subscriber changes applied at the Publisher during the last **Run** operation.

## **Applies To**

**SQLMerge Object** 

## **Syntax**

[value =] object.SubscriberChanges

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	Total number of rows inserted, deleted, and updated.

# **Data Type**

Long

#### **Modifiable**

Read-only

### Prototype C/C++

HRESULT get\_SubscriberChanges(long \*pVal);

#### See Also

**PublisherChanges Property** 

**SubscriberConflicts Property** 

# **SubscriberConflicts Property**

The **SubscriberConflicts** property specifies the total number of conflicts that occurred during the upload operation from the Subscriber.

## **Applies To**

**SQLMerge Object** 

# **Syntax**

object.SubscriberConflicts

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.

## **Data Type**

Long

### **Modifiable**

Read-only

## **Prototype C/C++**

HRESULT get\_SubscriberConflicts(long \*pVal);

### **See Also**

PublisherConflicts Property

**SubscriberChanges Property** 

# **SubscriberDatabase Property**

The **SubscriberDatabase** property specifies the name of the Subscriber database.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriberDatabase [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Subscriber database name.

#### **Remarks**

This is a required property if **SubscriberDatasourceType** is SQLSERVER.

#### **Data Type**

String

#### **Modifiable**

Read/write

### Prototype C/C++

HRESULT get\_SubscriberDatabase(BSTR \*pVal);

HRESULT put\_SubscriberDatabase(BSTR newVal);

# See Also

SubscriberDatabasePath Property

# **SubscriberDatabasePath Property**

The **SubscriberDatabasePath** property specifies the path to a Microsoft® Jet 4.0 database or Microsoft SQL Server<sup>TM</sup> detached database or subscription file.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriberDatabasePath [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
	Path to a Jet database or SQL Server detached database or subscription file.

#### **Remarks**

This is a required property if **SubscriberDatasourceType** is JET4\_DATABASE, or if you are using the DBADDOPTION constants ATTACH\_DATABASE or ATTACH\_SUBSCRIPTION when calling **AddSubscription**.

## **Data Type**

String

#### **Modifiable**

Read/write

### **Prototype C/C++**

HRESULT get\_SubscriberDatabasePath(BSTR \*pVal);
HRESULT put\_SubscriberDatabasePath(BSTR newVal);

# See Also

AddSubscription Method

**DBADDOPTION** 

SubscriberDatasourceType Property

# **SubscriberDatasourceType Property**

The **SubscriberDatasourceType** property specifies the type of Subscriber data source.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

## **Syntax**

object.SubscriberDatasourceType [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
	DATASOURCE_TYPE constant that specifies the type of database at the Subscriber.

### **Remarks**

The default is SQL\_SERVER.

## **Data Type**

DATASOURCE\_TYPE

#### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_SubscriberDatasourceType(
DATASOURCE\_TYPE \*pVal);

HRESULT put\_SubscriberDatasourceType(
DATASOURCE\_TYPE newVal);

# See Also

SubscriberDatabasePath Property

# **SubscriberLogin Property**

The **SubscriberLogin** property specifies the login name used connecting to the Subscriber.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriberLogin [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Subscriber login name.

#### **Remarks**

This is a required property if **SubscriberSecurityMode** is set to DB\_AUTHENTICATION.

#### **Data Type**

String

#### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_SubscriberLogin(BSTR \*pVal);

HRESULT put\_SubscriberLogin(BSTR newVal);

# See Also

SubscriberPassword Property

SubscriberSecurityMode Property

# **SubscriberPassword Property**

The **SubscriberPassword** property specifies the login password used when connecting to the Subscriber.

## **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriberPassword [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Subscriber password string.

#### **Remarks**

This property is used only when **SubscriberSecurityMode** is set to DB\_AUTHENTICATION.

The default is no password.

## **Data Type**

String

#### **Modifiable**

Read/write

## Prototype C/C++

HRESULT get\_SubscriberPassword(BSTR \*pVal);

HRESULT put\_SubscriberPassword(BSTR newVal);

# See Also

SubscriberLogin Property

SubscriberSecurityMode Property

# **SubscriberSecurityMode Property**

The **SubscriberSecurityMode** property specifies the security mode used when connecting to the Publisher.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriberSecurityMode [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
	SECURITY_TYPE constant that specifies the security mode enforced at the Subscriber.

#### Remarks

If the value is DB\_AUTHENTICATION (default), **SubscriberLogin** and **SubscriberPassword** will be used when connecting to the Subscriber.

NT\_AUTHENTICATION is not supported for **SubscriberSecurityMode** unless the Subscriber runs on the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 operating systems. NT\_AUTHENTICATION is not supported for any of **DistributorSecurityMode**, **PublisherSecurityMode**, and **SubscriberSecurityMode** unless the computer on which the replication control is running uses the Windows NT 4.0 or Windows 2000 operating system.

#### **Data Type**

**SECURITY TYPE** 

## **Modifiable**

Read/write

# **Prototype C/C++**

HRESULT get\_SubscriberSecurityMode(SECURITY\_TYPE \*pVal);
HRESULT put\_SubscriberSecurityMode(SECURITY\_TYPE newVal);

## See Also

SubscriberLogin Property

SubscriberPassword Property

# **SubscriptionName Property**

The **SubscriptionName** property specifies a display name for the subscription. This name is used in the Windows Synchronization Manager user interface.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.Subscription [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.
value	Name of the Subscription.

#### **Remarks**

If **SubscriptionName** is not specified, a subscription name is set using Subscriber name, Subscriber database, and publication.

The **SubscriptionName** property is not persisted anywhere except in the operating system registry for the current user Synchronization manager settings. All subsequent merges will show the property to be formatted:

subscribername:subscriberdatabase

This is consistent with the way subscription names are shown in SQL Server Enterprise Manager.

### **Data Type**

String

# Modifiable

Read/write

# Prototype (C/C++)

HRESULT get\_SubscriptionName(BSTR \*pVal);

HRESULT put\_SubscriptionName(BSTR newVal);

# **SubscriptionPriority Property**

The **SubscriptionPriority** returns or sets the priority of the subscription.

#### **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.SubscriptionPriority [= value]

Description
Expression that evaluates to a <b>SQLMerge</b> object.
Subscription priority value from 0.0 through 100.0, inclusive.

#### **Remarks**

If the **SubscriptionPriorityType** property has the value GLOBAL\_PRIORITY, use this property to set the priority.

#### **Data Type**

Single/Float

#### Modifiable

Read/write

## Prototype C/C++

HRESULT get\_SubscriptionPriority(float \*pVal);

HRESULT put\_SubscriptionPriority(float newVal);

#### **See Also**

Subscriber Types and Conflicts

SUBSCRIPTION\_PRIORITY\_TYPE

SubscriptionPriorityType Property

# **SubscriptionPriorityType Property**

The **SubscriptionPriorityType** property specifies how the subscription priority is determined.

## **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.SubscriptionType [= value]

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	SUBSCRIPTION_PRIORITY_TYPE constant that
	specifies how the subscription priority is determined.

#### **Remarks**

The subscription priority can be assigned or assume the priority value of the Publisher. If the value of the property is GLOBAL\_PRIORITY, use the **SubscriptionPriority** property to set the priority.

## **Data Type**

SUBSCRIPTION PRIORITY TYPE

#### **Modifiable**

Read/write

### Prototype C/C++

HRESULT get\_SubscriptionPriorityType(SUBSCRIPTION\_PRIORITY\_TYPE
\*pVal);

HRESULT put\_SubscriptionPriorityType(SUBSCRIPTION\_PRIORITY\_TYPE newVal);

# See Also

Subscriber Types and Conflicts
SubscriptionPriority Property

# **SubscriptionType Property**

The **SubscriptionType** property specifies whether the subscription is push, pull, or anonymous.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriptionType [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	SUBSCRIPTION_TYPE constant that specifies the type of subscription.

#### **Remarks**

The publication must be configured to support the specified subscription type.

The default is ANONYMOUS.

### **Data Type**

SUBSCRIPTION\_TYPE

#### **Modifiable**

Read/write

### Prototype C/C++

HRESULT get\_SubscriptionType(SUBSCRIPTION\_TYPE \*pVal);

 $HRESULT\ put\_SubscriptionType (SUBSCRIPTION\_TYPE\ newVal);$ 

# **SynchronizationType Property**

The **SynchronizationType** property specifies whether the subscription needs to be initially synchronized.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.SubscriptionType [= value]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
value	SYNCHRONIZATION_TYPE constant that specifies whether initial synchronization will occur.

#### **Remarks**

The default is AUTOMATIC.

### **Data Type**

**SYNCHRONIZATION TYPE** 

#### **Modifiable**

Read/write

## **Prototype C/C++**

 $HRESULT\ get\_SynchronizationType(SYNCHRONIZATION\_TYPE\ *pVal);$ 

HRESULT put\_SynchronizationType(SYNCHRONIZATION\_TYPE newVal);

# **SyncToAlternate Property**

The **SyncToAlternate** property returns or sets whether the synchronization is to an alternate synchronization partner.

#### **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.SyncToAlternate [= value]

Part	Description	
object	Expression that evaluates to a <b>SQLMerge</b> object.	
value	If set to <b>True</b> , an alternate synchronization partner is used.	

#### Remarks

To select an alternate synchronization partner, the **Publisher**, **PublisherDatabase**, **Publication**, and **Distributor** properties of the **SQLMerge** object should be set to the corresponding property values of an **AlternateSyncPartner** object before the **Run** method of the **SQLMerge** object is called.

#### **Data Type**

Boolean

#### **Modifiable**

Read/write

#### Prototype (C/C++)

HRESULT SyncToAlternate(VARIANT\_BOOL pVal);

HRESULT SyncToAlternate(VARIANT\_BOOL\* pVal);

### See Also

AlternateSyncPartner Object

AlternateSyncPartners Collection

**Distributor Property** 

**Publication Property** 

**Publisher Property** 

PublisherDatabase Property

# **UndeliveredCommands Property**

The **UndeliveredCommands** property specifies the number of commands currently available to download to the Subscriber.

### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object. Undelivered Commands

Part	Description
object	Expression that evaluates to a <b>SQLDistribution</b> object.

### **Data Type**

Long

### Modifiable

Read-only

#### **Prototype C/C++**

HRESULT get\_UndeliveredCommands(long \*pVal);

#### See Also

MaxDeliveredTransactions Property

<u>UndeliveredTransactions Property</u>

# **UndeliveredTransactions Property**

The **UndeliveredTransactions** property returns the number of transactions currently available to download to the Subscriber.

### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object. Undelivered Transactions

Part	Description
object	Expression that evaluates to a <b>SQLDistribution</b> object.

### **Data Type**

Long

#### **Modifiable**

Read-only

## **Prototype C/C++**

HRESULT get\_UndeliveredTransactions(long \*pVal);

### **See Also**

MaxDeliveredTransactions Property

**UndeliveredCommands Property** 

## **UseInteractiveResolver Property**

The **UseInteractiveResolver** property returns or sets whether the interactive resolver is used during reconciliation.

#### **Applies To**

**SQLMerge Object** 

#### **Syntax**

object.UseInteractiveResolver [= value]

Part	Description	
object	Expression that evaluates to a <b>SQLMerge</b> object.	
value	If set to <b>True</b> , the interactive resolver is used.	

#### **Remarks**

The interactive resolver is displayed for each row in which a conflict is detected. If the property is changed from **True** to **False** during a merge replication, all subsequent conflicts will be handled by the default resolver or the resolver specified when the article was created.

### **Data Type**

**Boolean** 

#### **Modifiable**

Read/write

#### Prototype (C/C++)

HRESULT UseInteractiveResolver(VARIANT\_BOOL pVal);

HRESULT UseInteractiveResolver(VARIANT\_BOOL\* pVal);

# **Validate Property**

The **Validate** property specifies the type of data validation to perform on the Subscriber data at the end of the **Run**.

## **Applies To**

**SQLMerge Object** 

### **Syntax**

object.Validate [= value]

Part	Description
object	Expression that evaluates to a <b>SQLMerge</b> object.
value	VALIDATE_TYPE constant that specifies the type of
	data validation to perform.

## **Data Type**

**VALIDATE TYPE** 

#### **Modifiable**

Read/write

## **Prototype C/C++**

HRESULT get\_Validate(VALIDATE\_TYPE newVal);

HRESULT put\_Validate(VALIDATE\_TYPE\* pVal);

# **WorkingDirectory Property**

The **WorkingDirectory** property returns or sets the working directory to which snapshot files are transferred using FTP when that option is specified.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.WorkingDirectory [= path]

Part	Description
object	Expression that evaluates to an object in the Applies To list.
<u>*</u>	Fully qualified path to the working directory for copied snapshots.

#### **Data Type**

String

#### **Modifiable**

Read/write

#### Prototype (C/C++)

HRESULT WorkingDirectory(BSTR pVal);

HRESULT WorkingDirectory(BSTR\* pVal);

# **Replication ActiveX Control Methods**

#### **Methods**

Add Method

AddReplError Method

**AddSubscription Method** 

**CopySubscription Method** 

**DropSubscription Method** 

**Initialize Method** 

ReinitializeSubscription Method

Run Method

SetFailoverMode Method

**Terminate Method** 

## **Add Method**

The **Add** method adds a **SQLReplError** object to a **SQLReplErrors** collection.

### **Applies To**

**SQLReplErrors Collection** 

#### **Syntax**

collection.Add object

Part	Description
collection	Expression that evaluates to a <b>SQLReplErrors</b> collection.
object	Expression that evaluates to a <b>SQLReplError</b> object.

#### **Remarks**

The **AddReplError** method can be used to add an error to a **SQLReplErrors** collection without creating a **SQLReplError** object first.

#### Prototype (C/C++)

HRESULT Add(
 ISQLReplError\* pISQLReplError);

#### See Also

AddReplError Method

**SQLReplError Object** 

# AddReplError Method

The **AddReplError** method adds a new error to a **SQLReplErrors** collection.

#### **Applies To**

**SQLReplErrors Collection** 

#### **Syntax**

collection.**AddReplError** Description, Source, ErrorNumber, \_ ErrorSourceType, ErrorNumberString

Part	Description	
collection	Expression that evaluates to a <b>SQLReplErrors</b>	
	collection.	
Description	String that describes the error.	
Source	String that describes the component that generated the	
	error.	
ErrorNumber	Long integer code for the error.	
ErrorSourceType	Value from the <b>ERRORSOURCE_TYPE</b>	
	enumerating the type of error source.	
ErrorNumberString	tring String representation of the error number.	

#### Remarks

The **AddReplError** method adds an error to a **SQLReplErrors** collection without the caller explicitly creating a **SQLReplError** object. The **Add** method can be used to add a **SQLReplError** object to a **SQLReplErrors** collection.

#### Prototype (C/C++)

HRESULT AddReplError( BSTR bstrDescription,

BSTR bstrSource, long lErrorNumber, ERRORSOURCE\_TYPE ErrorSourceType, BSTR bstrErrorNumberString);

### **See Also**

Add Method
ERRORSOURCE\_TYPE

**SQLReplError Object** 

## **AddSubscription Method**

The **AddSubscription** method adds a new or existing subscription based on the specified control properties.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.AddSubscription DBAddOption, SubscriptionHost

Part	Description	
object	Expression that evaluates to an object in the Applies To	
	list.	
DBAddOption	Option to add; use a value from <u>DBADDOPTION</u> .	
SubscriptionHost	Subscription host; use a value from	
	SUBSCRIPTION_HOST.	

#### Remarks

Push subscriptions are not currently supported by this method.

If *DBAddOption* is set to ATTACH\_DATABASE and **SubscriberDatasourceType** is set to SQL\_SERVER, **AddSubscription** can work only with single-file databases. Use the **SubscriberDatabasePath** property to specify the name and path of the .mdf file to attach.

If *DBAddOption* is ATTACH\_SUBSCRIPTION, use **SubscriberDatabasePath** to specify the name and path of the Microsoft Subscription File (.msf). This file can be created with the **CopySubscription** method.

**AddSubscription** also supports creating a new Subscriber database, creating a new subscription for an existing database, and registering an existing subscription with Windows Synchronization Manager.

In Microsoft® Visual Basic®, **AddSubscription** is a **Sub** method and does not return a value.

## **Prototype C/C++**

HRESULT AddSubscription(
DBADDOPTION DBAddOption,
SUBSCRIPTION\_HOST SubscriptionHost);

Return code	Description
S_OK	Method succeeded.
E_FAIL	General failure occurred. Check error
	records for detailed information.
REPLX_E_DBEXISTS	Specified database already exists.
REPLX_E_DBNOTFOUND	Specified database does not exist.
REPLX_E_SUBEXISTS	Specified subscription already exists.
REPLX_E_RETRYFAILURE	Failure occurred that might require retrying
	the last operation.

#### See Also

**CopySubscription Method** 

**DropSubscription Method** 

SubscriberDatasourceType Property

SubscriberDatabasePath Property

## **CopySubscription Method**

The **CopySubscription** method copies the entire subscription database to the file location specified by the parameter.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.CopySubscription filespec

Part	Description
object	Expression that evaluates to an object in the Applies To list.
, ,	File name and path to which subscription database is copied.

#### Remarks

**CopySubscription** copies a specially prepared database subscription file (typically with an .msf extension) to a Subscriber, attaches it, and receives an immediately synchronized subscription at the original Publisher.

**CopySubscription** creates the .msf file. Use the **AddSubscription** method with the ATTACH\_SUBSCRIPTION option to create the new subscription from the .msf file.

You can use the **CopySubscription** method to copy a subscription database that contains more than one subscription.

### Prototype (C/C++)

HRESULT CopySubscription(BSTR bstrSubscriptionFileName);

# See Also

AddSubscription Method

**DropSubscription Method** 

## **DropSubscription Method**

The **DropSubscription** method drops the subscription having the specified control properties.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.DropSubscription DBDropOption

Part	Description	
object	Expression that evaluates to an object in the Applies To	
	list.	
DBDropOption	Option to drop; use a value from <a href="DBDROPOPTION">DBDROPOPTION</a> .	

#### Remarks

Push subscriptions are not currently supported by this method.

If DROP\_DATABASE is specified, the database is dropped even when the subscription specified by the **SubscriptionName** property does not exist. Error notification is not provided.

**DropSubscription** also supports dropping the subscription without dropping the database and unregistering the subscription from Windows Synchronization Manager.

In Microsoft® Visual Basic®, **DropSubscription** is a **Sub** method and does not return a value.

### **Prototype C/C++**

HRESULT DropSubscription(

## DBDROPOPTION DBDropOption);

Return code	Description
S_OK	Method succeeded.
E_FAIL	General failure occurred. Check error
	records for detailed information.
REPLX_E_SUBNOTFOUND	Specified subscription does not exist.
REPLX_E_RETRYFAILURE	Failure occurred that might require retrying
	the last operation.

## See Also

AddSubscription Method

CopySubscription Method

**DBDROPOPTION** 

SubscriptionName Property

### **Initialize Method**

The **Initialize** method validates the control properties and establishes all database connections.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLS**napshot Object

#### **Syntax**

object.Initialize

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.

#### Remarks

Control properties that define the Publisher, publication, Distributor, and Subscriber (for the Distribution and Merge controls) must be set before calling **Initialize**.

In Microsoft® Visual Basic®, **Initialize** is a **Sub** method and does not return a value.

**Initialize** should not be called prior to using the **AddSubscription** method to add a new subscription, or prior to using the **DropSubscription** method to drop an existing subscription.

#### Prototype (C/C++)

HRESULT Initialize( );

Return code	Description
-------------	-------------

S_OK	Method succeeded.
E_FAIL	General failure occurred. Check error
	records for detailed information.
REPLX_E_DBNOTFOUND	Specified database does not exist.
REPLX_E_SUBNOTFOUND	Specified subscription does not exist
	(SQLDistribution and SQLMerge
	objects only).
REPLX_E_SUBEXPIRED	Subscription has expired
	(SQLDistribution and SQLMerge
	objects only).
REPLX_E_PROFILENOTFOUND	Specified profile does not exist.
REPLX_E_RETRYFAILURE	Failure occurred that might require
	retrying the last operation.

# See Also

<u>Developing Replication Applications Using ActiveX Controls</u>

### ReinitializeSubscription Method

The **ReinitializeSubscription** method configures a subscription to reapply the initial snapshot and subsequent changes during the next **Run** operation.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

#### **Syntax**

object.ReinitializeSubscription [bUploadBeforeReinit]

Part	Description	
object	Expression that evaluates to a <b>SQLMerge</b> object.	
bUploadBeforeReinit	If <b>True</b> , the changes in the subscription database are	
uploaded to the Publisher before the snapshot is		
	applied at the Subscriber. The default is <b>False</b> .	

#### **Remarks**

**ReinitializeSubscription** is a method of both the **SQLDistribution** and **SQLMerge** objects. **ReinitializeSubscription** for the **SQLMerge** object has the **bUploadBeforeReinit** parameter. For the **SQLDistribution** object, **ReinitializeSubscription** has no parameters.

In Microsoft® Visual Basic®, **ReinitializeSubscription** is a **Sub** method and does not return a value.

#### Prototype (C/C++)

HRESULT ReinitializeSubscription(VARIANT\_BOOL bUploadBeforeReinit);

Return code	Description	
S_OK	Method succeeded.	

E_FAIL	General failure occurred. Check error	
	records for detailed information.	
REPLX_E_DEADLOCK	Deadlock occurred.	
REPLX_E_RETRYFAILURE	Failure occurred that might require retrying	
	the last operation.	

### **Run Method**

The **Run** method executes the replication process using the control properties.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

### **Syntax**

object.Run

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.

#### **Remarks**

The control must call **Initialize** before calling the **Run** method.

In Microsoft® Visual Basic®, **Run** is a **Sub** method and does not return a value.

### Prototype (C/C++)

HRESULT Run();

Return code	Description
S_OK	Method succeeded.
E_FAIL	General failure occurred. Check error records for detailed information.
REPLX_E_DEADLOCK	Deadlock occurred.
REPLX_E_RETRYFAILURE	Failure occurred that might require retrying the last operation.

REPLX_S_VALIDATIONFAILED	Validation failure occurred.
REPLX_S_CONFLICTSOCCURRED	Conflicts occurred while merging
	changes ( <b>SQLMerge</b> object only).

### See Also

**Developing Replication Applications Using ActiveX Controls** 

#### SetFailoverMode Method

The **SetFailoverMode** method sets the Subscriber update mode when adding subscriptions to use immediate (DTC) updates or queued updates, or support immediate updating with a queued updating failover.

### **Applies To**

**SQLDistribution Object** 

#### **Syntax**

object.SetFailoverMode FailoverMode

Part	Description	
object	Expression that evaluates to a <b>SQLDistribution</b> object.	
FailoverMode	Value from the REPL_FAILOVER_MODE	
	enumeration.	

### Prototype (C/C++)

HRESULT SetFailoverMode(REPL\_FAILOVER\_MODE FailoverMode);

#### See Also

REPL\_FAILOVER\_MODE

### **Terminate Method**

The **Terminate** method terminates the replication process and closes all database connections.

### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

#### **Syntax**

object.Terminate

Part	Description
object	Expression that evaluates to an object in the Applies To
	list.

#### **Remarks**

After using **Terminate**, the control host must call **Initialize** before again calling **Run**.

In Microsoft® Visual Basic®, **Terminate** is a **Sub** method and does not return a value.

#### Prototype (C/C++)

HRESULT Terminate( );

Return code	Description
S_OK	Method succeeded.
E_FAIL	General failure occurred. Check error
	records for detailed information.
REPLX_E_RETRYFAILURE	Failure occurred that might require retrying

the last operation.

### See Also

**Developing Replication Applications Using ActiveX Controls** 

# **Replication ActiveX Control Events**

This section defines the events of the replication Microsoft® ActiveX® control objects and collections.

#### **Events**

**Notify Event** 

**Status Event** 

### **Notify Event**

The **Notify** event is raised when an error is added to the **SQLReplErrors** collection by a replication Microsoft® ActiveX® control.

#### **Applies To**

**SQLReplErrors Collection** 

### **Prototype (Visual Basic)**

Sub *Objectvar\_***Notify**( \_
ByVal *Status* As REPLERRXLib.AGENT\_STATUS \_
ByVal *Message* As String)

Part	Description	
Objectvar	Expression that evaluates to a <b>SQLReplErrors</b> collection.	
Status	Value that indicates the type of error from the AGENT_STATUS enumeration.	
Message	Description of the error.	

#### **Remarks**

To receive the **Notify** event, the program must declare *Objectvar* **WithEvents** of type **REPLERRXLib.SQLReplErrors**.

This event is not available through the **ISQLReplErrors** interface of the replication ActiveX controls.

### Prototype (C/C++)

HRESULT Notify(
AGENT\_STATUS Status,
BSTR Message);

### See Also

AGENT\_STATUS

#### **Status Event**

The **Status** event returns information about significant occurrences in the operation of a replication Microsoft® ActiveX® control.

#### **Applies To**

**SQLDistribution Object** 

**SQLMerge Object** 

**SQLSnapshot Object** 

### **Prototype (Visual Basic)**

```
Function Objectvar_Status( _ ByVal Message As String, _ ByVal Percent As Long) _ As STATUS_RETURN_CODE
```

Part	Description	
Objectvar	Expression that evaluates to an object in the Applies To	
	list.	
Message	Description of the significant occurrence.	
Percent	Percentage of the operation completed successfully,	
	from 0 through 100.	

#### **Remarks**

To receive the **Status** event, a Microsoft Visual Basic® program must declare *Objectvar* **WithEvents** of the appropriate type.

STATUS\_RETURN\_CODE is an enumeration that specifies the values that can be returned from the **Status** callback function (event).

#### Prototype (C/C++)

HRESULT Status(
BSTR Message,
Long Percent,
STATUS\_RETURN\_CODE \*pReturnCode);

#### See Also

Developing Replication Applications Using ActiveX Controls

STATUS\_RETURN\_CODE

# **Replication ActiveX Control Constants**

This section defines the enumerated data types that are used as parameters and return values in Microsoft Receive X replication control properties, methods, and events.

### **AGENT\_STATUS**

The **AGENT\_STATUS** constants provide agent status codes that are returned by the **Notify** event of the **SQLReplErrors** collection.

Constant	Value	Description
REPLAGENT_FAIL	4	Agent operation failed.
REPLAGENT_IN_PROGRESS	1	Agent operation in progress.
REPLAGENT_RETRY	3	Agent operation failed.
REPLAGENT_SUCCEED	2	Agent operation completed
		successfully.

#### **Remarks**

REPLAGENT\_RETRY indicates the agent operation failed with errors that may not recur if the operation is retried at a later time. The control does not retry the operation unless the calling program directs it to do so.

#### **See Also**

**Notify Event** 

**SQLReplErrors Collection** 

### DATASOURCE\_TYPE

The **DATASOURCE\_TYPE** constants provide data source type values for the **SubscriberDatasourceType** property.

Constant	Value	Description
DB2_UNIVERSAL	6	DB2 Universal database Subscriber
EXCHANGE	4	Microsoft® Exchange Subscriber
JET4_DATABASE	2	Microsoft Jet 4.0 database Subscriber
ODBC_DSN	1	ODBC data source Subscriber
OLEDB_DATASOURCE	3	OLE DB data source Subscriber
ORACLE	5	Oracle database Subscriber
SQL_SERVER	0	Microsoft SQL Server <sup>™</sup> Subscriber

#### **See Also**

<u>SubscriberDatasourceType Property</u>

### **DBADDOPTION**

The **DBADDOPTION** constants specify if the Subscriber database exists or if it must be created or attached, or if the subscription must be attached, when calling the **AddSubscription** method of the **SQLDistribution** and **SQLMerge** objects.

Constant	Value	Description
ATTACH_DATABASE	2	Attaches a Subscriber database file,
		typically an .mdf (Microsoft® SQL
		Server <sup>TM</sup> Subscribers only).
ATTACH_SUBSCRIPTION	3	Attaches a subscription file,
		typically an .msf (Microsoft
		Subscription File).
CREATE_DATABASE	1	Creates the Subscriber database
		(SQL Server Subscribers only).
EXISTING_DATABASE	0	Uses an existing Subscriber
		database.
REGISTER_SUBSCRIPTION	4	Registers the existing subscription
		with Windows Synchronization
		Manager.

#### See Also

AddSubscription Method

### **DBDROPOPTION**

The **DBDROPOPTION** constants specify whether the Subscriber database should be dropped when calling the **DropSubscription** method of the **SQLDistribution** and **SQLMerge** objects.

Constant	Value	Description
DROP_DATABASE	1	Drops the Subscriber database
		and the subscription, if
		specified.
LEAVE_DATABASE	0	Drops the subscription without
		dropping the Subscriber
		database.
UNREGISTER_SUBSCRIPTION	2	Unregisters, but does not drop,
		the subscription.

#### **See Also**

**DropSubscription Method** 

## ERRORSOURCE\_TYPE

The **ERRORSOURCE\_TYPE** constants provide values for the **SourceType** property of the **SQLReplError** object.

Constant	Value	Description
INVALID_SOURCE_TYPE	0	Error source type is invalid.
MERGE_PROCESS	9	Merge process error.
MERGE_PROVIDER	8	Merge replication provider error.
NET_LIBRARY	6	Net-Library error.
ODBC_API	4	ODBC API error.
OPERATING_SYSTEM	3	Operating system error.
REPL_CONTROL	2	Replication ActiveX® control error.
SQL_COMMAND	1	SQL command error.
SQLDMO	7	SQL DMO error.
SQLSERVER_ENGINE	5	Microsoft® SQL Server™ error.

### **See Also**

SourceType Property

### **EXCHANGE\_TYPE**

The **EXCHANGE\_TYPE** constants are used with the **ExchangeType** property of the **SQLMerge** object to specify whether merge replication changes should be uploaded to the Publisher, downloaded to the Subscriber, or both (uploaded and then downloaded).

Constant	Value	Description
UPLOAD	1	Only merge Subscriber changes with the Publisher.
		Publisher.
DOWNLOAD	2	Only merge Publisher changes with the
		Subscriber.
BIDIRECTIONAL	3	Merge all changes between the Publisher and
		Subscriber (default).

#### **Remarks**

You should use a BIDIRECTIONAL synchronization unless you have a specific reason for separating the UPLOAD and DOWNLOAD phases.

#### See Also

**ExchangeType Property** 

## FILE\_TRANSFER\_TYPE

The **FILE\_TRANSFER\_TYPE** constants specify the type of transfer for snapshot files.

Constant	Value	Description
FILETRANSFERFTP	1	Download snapshot files using FTP.
FILETRANSFERUNC	0	Apply snapshot from a UNC network share.

### **See Also**

FileTransferType Property

# NETWORK\_TYPE

The **NETWORK\_TYPE** constants provide network protocol type values for the **DistributorNetwork** and **PublisherNetwork** properties.

Constant	Value	Description	
DEFAULT	0	Use the current configured client Net-	
		Library (default).	
MULTI_PROTOCOL	2	Multiprotocol Net-Library.	
TCPIP_SOCKETS	1	TCP/IP Sockets Net-Library.	

#### **See Also**

**DistributorNetwork Property** 

**PublisherNetwork Property** 

#### REPL\_FAILOVER\_MODE

The **REPL\_FAILOVER\_MODE** enumeration specifies the Subscriber update mode when adding subscriptions to use immediate updating or queued updating options with transactional replication.

Constant	Value	Description
FAILOVER_IMMEDIATE	1	Immediate updating.
FAILOVER_NONE	0	No updating from Subscriber.
FAILOVER_QUEUED	2	Queued updating.

#### See Also

SetFailoverMode Method

# REPLICATION\_TYPE

The **REPLICATION\_TYPE** enumeration specifies the type of replication for which the snapshot is to be used.

Constant	Value	Description	
MERGE	2	Specifies merge replication.	
TRANSACTIONAL	1	Specifies transactional replication or	
		snapshot replication.	

#### **See Also**

ReplicationType Property

# REPLRPC\_SECURITY\_TYPE

The **REPLRPC\_SECURITY\_TYPE** constants provide data source type values for the **PublisherRPCSecurityMode** property.

Constant	Value	Description
RPC_STANDARD_MODE	0	Dynamic RPC connection is used.
RPC_STATIC_MODE	2	Static <b>sysservers</b> entry is used for
		RPC.

#### **See Also**

 $\underline{Publisher RPC Security Mode}$ 

#### **SECURITY\_TYPE**

The **SECURITY\_TYPE** specifies security type values for the **DistributorSecurityMode**, **PublisherSecurityMode**, and **SubscriberSecurityMode** properties.

Constant	Value	Description
DB_AUTHENTICATION	0	Specifies SQL Server Authentication
		for the connection.
NT_AUTHENTICATION	1	Specifies Windows Authentication for
		the connection (supported by
		Microsoft® SQL Server™ on Microsoft
		Windows NT® 4.0 and Microsoft
		Windows® 2000 operating systems
		only).

#### **Remarks**

If the computer on which the Microsoft ActiveX® replication control is hosted is not running the Windows NT 4.0 or Windows 2000 operating system, NT\_AUTHENTICATION cannot be used on the Publisher, Distributor, or Subscriber.

#### See Also

<u>DistributorSecurityMode Property</u>

SubscriberSecurityMode Property

PublisherSecurityMode Property

# STATUS\_RETURN\_CODE

The **STATUS\_RETURN** specifies the return code values that can be returned from the status callback functions.

Constant	Value	Description
SUCCESS	0	Operation is successful.
CANCEL	1	Operation is canceled.

#### **See Also**

**Status Event** 

#### SUBSCRIPTION\_HOST

The **SUBSCRIPTION\_HOST** specifies subscription host codes for the *SubscriptionHost* parameter of the **AddSubscription** method of the **SQLMerge** and **SQLDistribution** objects.

Constant	Value	Description
NONE	0	Subscription is not registered under other
		hosts.
SYNC_MANAGER	1	Subscription is to be registered in Windows
		Synchronization Manager.

#### **See Also**

**AddSubscription Method** 

#### SUBSCRIPTION\_PRIORITY\_TYPE

The **SUBSCRIPTION\_PRIORITY\_TYPE** constants specify subscription priority type values for the **SubscriptionPriorityType** property of the **SQLMerge** object.

Constant	Value	Description
GLOBAL_PRIORITY	1	Subscription has an assigned priority value.
LOCAL_PRIORITY		Subscription uses the priority value of the Publisher.

#### See Also

SubscriptionPriorityType Property

# SUBSCRIPTION\_TYPE

The **SUBSCRIPTION\_TYPE** specifies subscription type values for the **SubscriptionType** property of the **SQLMerge** and **SQLDistribution** objects.

Constant	Value	Description
ANONYMOUS	2	Anonymous subscription.
PULL	1	Pull subscription.
PUSH	0	Push subscription.

#### **See Also**

SubscriptionType Property

#### SYNCHRONIZATION\_TYPE

The **SYNCHRONIZATION\_TYPE** specifies subscription synchronization type codes for the **SynchronizationType** property of the **SQLMerge** and **SQLDistribution** objects.

Constant	Value	Description
AUTOMATIC		Initial synchronization is to be provided to the Subscriber.
NOSYNC	2	No initial synchronization is needed by the Subscriber.

#### **See Also**

SynchronizationType Property

# VALIDATE\_TYPE

The **VALIDATE\_TYPE** constants specify the type of data validation to perform on the Subscriber data at the end of the run. **VALIDATE\_TYPE** is used with the **Validate** property of the **SQLMerge** object.

Constant	Value	Description
FAST_ROWCOUNT_AND_CHECKSUM	4	Perform a fast row count and checksum validation of the Subscriber data.
FAST_ROWCOUNT_AND_BINARYCHECKSUM	6	Perform a fast row count and binary checksum validation the Subscriber data. BINARYCHECKSU is not supported by Microsoft® SQL Server <sup>TM</sup> 2000 Subscribers.
FAST_ROWCOUNT_ONLY	3	Perform only a fast row count validation of the Subscriber data
NO_VALIDATION	0	Do not validate the Subscriber data (default).
ROWCOUNT_AND_CHECKSUM	2	Perform a full row count and checksum validation of the Subscriber data.
ROWCOUNT_AND_BINARYCHECKSUM	5	Perform a full row count and binary checksum validation the Subscriber data. BINARYCHECKSU

		is not supported by SQL Server 2000 Subscribers.
ROWCOUNT_ONLY	1	Perform only a full
		row count validation
		of the Subscriber data

#### See Also

Help with Replication
Validate Property

#### Developing Replication Merge Conflict Resolvers Through a Custom Resolver

Microsoft® SQL Server™ 2000 supports two types of user-implemented conflict resolvers:

 Custom conflict resolvers built as COM components and compiled into dynamic-link libraries (.dll) through products such as Microsoft Visual Basic® and Microsoft Visual C++®. The COM custom conflict resolver is a DLL that implements the **ICustomResolver** interface, methods, and properties. There are interfaces and type definitions designed specially for conflict resolution. For information about the required header file, see COM Conflict Resolver Header File.

To use a COM object resolver, make sure the DLL is registered at the computer where the Merge Agent runs. For a push subscription, this is the Distributor computer, and for a pull subscription, it is the Subscriber computer. When using Visual C++ or Visual Basic, the name of the project becomes the name of the .DLL. The DLL name must be a unique resolver name. Run the **sp\_enumcustomresolvers** stored procedure to ensure uniqueness. **sp\_enumcustomresolvers** displays all the resolvers currently registered on the system.

 User-built Transact-SQL stored procedures can be used instead of a COM component. The stored procedures must implement a specific set of parameters required for a conflict resolver. For more information about the use of each field defined, see <u>Custom Stored Procedure</u> <u>Conflict Resolver</u>.

The Microsoft SQL Server 2000 CD-ROM ships with some resolver samples. For more information, see <u>Replication Resolver Samples</u>.

#### **See Also**

<u>Custom Stored Procedure Conflict Resolver</u>

Merge Replication Conflict Detection and Resolution

# Other Microsoft Resolvers Replication Architecture

#### **COM Conflict Resolver Header File**

A file named sqlres.h is located in C:\Program Files\Microsoft SQL Server\80\ Dev Tools\include, if the replication sample programs were installed to the default folder. The file, sqlres.h, contains several important definitions with which you should be familiar. Do no modify this file.

The custom resolver must implement the interface **ICustomResolver**, which is defined in this file.

```
#undef INTERFACE
#define INTERFACE ICustomResolver
DECLARE_INTERFACE_(ICustomResolver, Iunknown)
//** IUnknown methods
 STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID *pp
 STDMETHOD_(ULONG,AddRef) (THIS)
                                      PURE:
 STDMETHOD (ULONG, Release) (THIS)
                                      PURE:
//** ICustomResolver methods
 STDMETHOD(Reconcile) (THIS
           IReplRowChange *pRowChange,
           DWORD dwFlags,
           PVOID pvReserved) PURE;
 STDMETHOD(GetHandledStates) (THIS
           DWORD *pResolverBm) PURE;
};
```

**ICustomResolver** inherits from **IUnknown**, similar to all COM classes. The **IUnknown** methods usually do not need to be modified from the supplied resolver, but the **ICustomResolver** methods **Reconcile** and **GetHandledStates** must be implemented. **Reconcile** is the method called for each table row that contains a conflict. **GetHandledStates** defines the conflict conditions that the resolver will handle.

The important parameter for **Reconcile** is a reference to an **IReplRowChange** object; **IReplRowChange** is defined in this include file. Through the method of **IReplRowChange**, the resolver determines the columns in conflict, examines the conflicting data, and then copies the appropriate data to the result row.

Other definitions in the file include **IConnectionInfo**, which is used when a resolver needs to access a stored procedure, and **ITranDataChange**, which is used in a transactional resolver. Only the **Get**<**xxx**> methods in **IConnectionInfo** are accessible to user-implemented resolvers.

Several enumerations are defined in **sqlres.h**. Symbols from these enumerations should be used wherever possible instead of using hard-coded constants.

Enumeration Name	Description
REPOLE_CHANGE_TYPE	Codes for the database operation (insert,
	update, delete), whether there is a conflict,
	and whether column tracking is active.
	Many symbolic definitions of aggregates
	of these change types are also available.
REPOLE_CONFLICT_TYPE	Codes for the database operation and
	whether the failure occurred at upload or
	download.
REPOLE_COLSTATUS_TYPE	Codes for the conflict status of an
	individual column.
REPOLE_PRIORITY_TYPE	Codes for what have higher priority
	(source, destination, neither source nor
	destination).

# Programming Replication from Heterogeneous Data Sources

Microsoft® SQL Server™ enables third-party products to become Publishers within the SQL Server replication framework. The Replication Distributor Interface allows replication from heterogeneous databases that provide 32-bit OLE DB drivers to Subscribers running SQL Server 2000. Heterogeneous data sources include:

- Oracle databases
- DB2 databases
- Microsoft Access databases
- Other databases that comply with SQL Server ODBC or OLE DB Subscriber requirements.

When integrated, the Replication Distributor Interface exposes the SQL Distributor, and allows heterogeneous data sources to store meta data and replicated transactions in the SQL Server Distributor database. The Replication Distributor Interface is an OLE DB service provider that allows users to store replicated SQL statements, scripts, and .bcp files in the Distributor store-and-forward database. The Replication Distributor Interface is based on the OLE DB connection model and supports a subset of the **DataSource**, **Session**, and **Error** objects. An additional **Distribution** object is added to the **Session** object and is used to store transactions marked for replication in a SQL Server distribution database.

**Note** The Replication Distributor Interface is a special purpose OLE DB service that is used only to distribute replicated SQL Server transactions. It does not support the minimal set of interfaces necessary to be considered a standard OLE DB data provider.

The Replication Distributor Interface cannot be used with replication types that

need updates to be made at the Subscriber. The only types of replication that can be used with the Replication Distributor Interface are snapshot replication and transactional replication that has read-only Subscribers. Merge replication, and transactional replication with immediate updating, queued updating, or immediate updating with queued updating as failover are not available from heterogeneous Publishers to SQL Server Subscribers.

Here are the steps to using the Replication Distributor Interface.

- 1. This step uses Microsoft Visual Basic® or Microsoft Visual C++® and the replication SQL-DMO objects, makes calls for setting up the publication, articles, and subscription information. These calls differ from the typical SQL-DMO setup calls because they are made on the distribution server and not at the Publisher. A third-party replication tool can also be used to implement this first step
- 2. This step takes the place of the Snapshot Agent and Log Reader Agent. The Replication Distributor Interface is used to store the replication transactions on a server that is currently acting as the Distributor, which will then be distributed by the Distribution Agent. The following C++ code is a sample of an object that is used to place commands into the distribution database.
  - // Instantiate a data source object for the SQL Server Publishir // provider.
  - hr = CoCreateInstance(CLSID\_SQLDistribution, NULL, CLS
    (void\*\*) &pIDBInit);

Using the Replication Distributor Interface leaves the responsibility of data modification detection to the developers because the Log Reader Agent is not available. The use of the monitoring and troubleshooting tools, alerts, and notifications are still available in SQL Server Enterprise Manager. The Log Reader Agent can be activated by using Microsoft Host Integration Server 2000 and its Distribution Store Interface (**iDistributionStore**). For more information, see the Host Integration Server 2000 documentation.

The Microsoft SQL Server CD-ROM ships with some Replication Distributor Interface samples. For more information, see <u>Replication Distributor Interface</u>

#### Samples.

When deploying an application using the Replication Distributor Interface independently of SQL Server 2000, you must include additional files in the installation kit you use to distribute your application. If you will be deploying your application to a computer where SQL Server 2000 will also be installed, these files will already be present.

#### **SQL-DMO Replication Objects**

SQL-DMO provides a set programming interface for administering and monitoring Microsoft® SQL Server™ replication, including the ability to administer replication from heterogeneous computers. Here are the SQL-DMO objects used in replication:

- **Distributor** object
- **DistributionDatabase** object
- **DistributionPublisher** object
- **DistributionPublication** object
- **DistributionSubscription** object
- **DistributionArticle** object
- **RegisteredSubscriber** object
- Replication object

The SQL-DMO replication objects are a subset of the SQL-DMO COM object model. For information about how these object fit into the larger replication object model, see <u>Developing SQL-DMO Applications</u>.

#### **See Also**

**Distributor Object** 

**DistributionArticle Object** 

**Distribution**Database Object

**DistributionPublication Object** 

DistributionPublisher Object

**DistributionSubscription Object** 

RegisteredSubscriber Object

**Replication Object** 

# **Replication Distributor Interface Reference**

The Replication Distributor Interface can be used to enable Microsoft® SQL Server<sup>TM</sup> 2000 replication services on heterogeneous databases. The Replication Distributor Interface is based on the OLE DB connection model. The objects are available using Microsoft Visual C++® and the OLE library.

The Replication Distributor Interface exposes a **Distribution** object that is generated from a **Session** object. The **Distribution** object is used to store replicated transactions in the distribution database on the SQL Server Distributor, and is used to log history and error information.

The Replication Distributor Interface is implemented using these files:

- Repldist.dll
- Repldist.h

The Replication Distributor Interface objects, methods, properties, events, and structures can be found in the following topics.

Торіс	Description
Replication Distributor Interface	Description of Replication
<u>Objects</u>	Distributor Interface <b>Distribution</b>
	objects.
Replication Distributor Interface	Description of Replication
<u>Properties</u>	Distributor Interface properties.
Replication Distributor Interface	Description of Replication
<u>Methods</u>	Distributor Interface methods.
Replication Distributor Interface	Description of Replication
Structures	Distributor Interface structures.

#### See Also

**Replication Distributor Interface Samples** 

# **Replication Distributor Interface Objects**

The Replication Distributor Interface exposes these objects.

Topic	Description
DistributionLog Object	Stores history and error information
	about the Replication Distributor
	Interface.
DistributionStore Object	Stores transactions in a Distributor.

# **DistributionLog Object**

The **DistributionLog** object stores history and error information about the Replication Distributor Interface. This information is used to monitor replication.

#### **Methods**

AddLog Method

# **DistributionStore Object**

The **DistributionStore** object stores transactions in a Distributor. This is a custom interface supported only by the Distributor.

#### **Methods**

**Abort Method** 

AddTransactionCommands Method

**Commit Method** 

**GetLastTransaction Method** 

**StartTransaction Method** 

# **Replication Distributor Interface Properties**

This section defines the properties of the Replication Distributor Interface.

# **DBPROP\_APPLICATION\_NAME** Property

The **DBPROP\_APPLICATION\_NAME** property specifies the name of the application.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_APPLICATION\_NAME

## **Data Type**

VT\_BSTR

#### **Modifiable**

# **DBPROP\_APPLICATION\_TYPE** Property

The **DBPROP\_APPLICATION\_TYPE** property specifies the Publisher application type. The application type can be either snapshot or transactional (incremental updates).

## **Applies To**

**DistributionStore Object** 

#### **Syntax**

object.DBPROP\_APPLICATION\_TYPE

## **Data Type**

VT I1

#### **Modifiable**

# **DBPROP\_AUTH\_PASSWORD Property**

The **DBPROP\_AUTH\_PASSWORD** property specifies the password used when connecting to the data source.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_AUTH\_PASSWORD

## **Data Type**

VT\_BSTR

#### **Modifiable**

# **DBPROP\_AUTH\_USERID Property**

The **DBPROP\_AUTH\_USERID** property specifies the user ID used when connecting to the data source.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_AUTH\_USERID

## **Data Type**

VT\_BSTR

#### **Modifiable**

# **DBPROP\_DBMSNAME** Property

The **DBPROP\_DBMSNAME** property specifies the name of the product accessed by the provider.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_DBMSNAME

## **Data Type**

VT\_BSTR

#### **Modifiable**

Read-only

# **DBPROP\_DBMSVER Property**

The **DBPROP\_DBMSVER** property specifies the version of the product accessed by the provider.

# **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.**DBPROP\_DBMSVER** 

## **Data Type**

VT\_BSTR

#### **Modifiable**

Read-only

# **DBPROP\_INIT\_DATASOURCE** Property

The **DBPROP\_INIT\_DATASOURCE** property specifies the name of the distribution database to which to connect.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_INIT\_DATASOURCE

## **Data Type**

VT\_BSTR

#### **Modifiable**

# **DBPROP\_INIT\_LOCATION Property**

The **DBPROP\_INIT\_LOCATION** property specifies the location of the Distributor to which to connect (typically, the server name).

## **Applies To**

<u>DistributionStore Object</u>

#### **Syntax**

object.DBPROP\_INIT\_LOCATION

## **Data Type**

VT\_BSTR

#### **Modifiable**

# DBPROP\_INIT\_PUBLISHERDATASOURCE Property

The **DBPROP\_INIT\_PUBLISHERDATASOURCE** property specifies the name of the Publisher database on whose behalf the connection is made.

## **Applies To**

**DistributionStore Object** 

## **Syntax**

object.DBPROP\_INIT\_PUBLISHERDATASOURCE

#### **Data Type**

VT\_BSTR

#### **Modifiable**

# **DBPROP\_INIT\_PUBLISHER\_NAME Property**

The **DBPROP\_INIT\_PUBLISHER\_NAME** property specifies the name of the Publisher on whose behalf the connection is made.

## **Applies To**

**DistributionStore Object** 

#### **Syntax**

object.DBPROP\_INIT\_PUBLISHERNAME

## **Data Type**

VT\_BSTR

#### **Modifiable**

# **DBPROP\_INIT\_TIMEOUT Property**

The **DBPROP\_INIT\_TIMEOUT** property specifies the connection time-out.

## **Applies To**

**DistributionStore Object** 

## **Syntax**

object.DBPROP\_INIT\_TIMEOUT

## **Data Type**

VT\_I4

#### **Modifiable**

# **DBPROP\_INIT\_XACT\_SEQNO\_SIZE** Property

The **DBPROP\_INIT\_XACT\_SEQNO\_SIZE** property specifies the size of the transaction sequence number.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_INIT\_XACT\_SEQNO\_SIZE

## **Data Type**

VT\_I1

#### **Modifiable**

# **DBPROP\_INIT\_XACTID\_SIZE** Property

The **DBPROP\_INIT\_XACTID\_SIZE** property specifies the size of the transaction ID.

## **Applies To**

<u>DistributionStore Object</u>

## **Syntax**

object.DBPROP\_INIT\_XACTID\_SIZE

## **Data Type**

VT\_I1

#### Modifiable

## **DBPROP\_PUBLICATION\_NAME** Property

The **DBPROP\_PUBLICATION\_NAME** property specifies the name of the publication serviced by the application.

### **Applies To**

<u>DistributionStore Object</u>

### **Syntax**

object.DBPROP\_PUBLICATION\_NAME

### **Data Type**

VT\_BSTR

#### **Modifiable**

Read/write

# **Replication Distributor Interface Methods**

This section defines the methods of the Replication Distributor Interface.

## **Abort Method**

The **Abort** method rolls back the storage of transaction commands.

### **Applies To**

<u>DistributionStore Object</u>

### **Syntax**

HRESULT Abort( );

#### Remarks

The method returns S\_OK if it succeeds and DB\_E\_ERROROCCURRED if it fails.

## AddLog Method

The **AddLog** method adds history and error information to a distribution store.

### **Applies To**

**DistributionLog Object** 

#### **Syntax**

HRESULT AddLog(
DWORD dwStatusID,
LPSTR szComment,
BYTE\* pXactSeqno,
ULONG ulTransactions,
ULONG ulCommands,
ULONG cErrorDescs,
const DISTERRORDESC rgErrorDescs[] );

Part	Description	
dwStatusID	Status of the log message:	
	• 1 = STARTUP	
	• 2 = SUCCESS	
	• 3 = INPROGRESS	
	• 4 = RETRY	
	• 5 = FAILURE	
szComment	Log message text	
XactSeqno	Transaction sequence number	

ulTransactions Number of transactions associated with this message		
ulCommands	ds Number of commands associated with this message	
cErrorDescs	Number of DISTCOMMANDDESC structures in the	
	rgCommandDesc array	
rgErrorDescs	Array of DISTERRORDESC structures that describe the	
	replicated command	

### Remarks

The method returns S\_OK if it succeeds and DB\_E\_ERROROCCURRED if it fails.

### **See Also**

**DISTERRORDESC Structure** 

#### AddTransactionCommands Method

The **AddTransactionCommands** method stores a group of replicated transaction commands. If the **StartTransaction** method is not called before this method, **AddTransactionsCommands** uses an implicit transaction.

### **Applies To**

**DistributionStore Object** 

#### **Syntax**

HRESULT AddTransactionCommands(
ULONG cCommandDescs,
const DISTCOMMANDDESC rgCommandDescs []);

Part	Description
cCommandDescs	Number of DISTCOMMANDDESC structures in the
	rgCommandDescs array
rgCommandDescs	Array of DISTCOMMANDDESC structures that
	describe the replicated command

#### **Remarks**

The method returns S\_OK if it succeeds and DB\_E\_ERROROCCURRED if it fails.

#### **See Also**

**DISTCOMMANDDESC Structure** 

## **Commit Method**

The **Commit** method commits the storage of the transaction commands.

### **Applies To**

**DistributionStore Object** 

### **Syntax**

HRESULT Commit( );

#### Remarks

The method returns S\_OK if it succeeds and DB\_E\_ERROROCCURRED if it fails.

### **GetLastTransaction Method**

The **GetLastTransaction** method retrieves information about the last stored transaction.

### **Applies To**

**DistributionStore Object** 

#### **Syntax**

HRESULT GetLastTransaction(
BYTE\* pXactID,
BYTE\* pXactSeqno );

Part	Description
XactID	Transaction identifier that uniquely identifies the transaction. It
	can be up to 255 bytes. The default is NULL.
XactSeqno	Transaction sequence number that identifies the sequence in
	which transactions are committed (in big-endian format).
	Transactions with lower sequence numbers are committed first.

#### **Remarks**

The method returns S\_OK if it succeeds and DB\_E\_ERROROCCURRED if it fails.

### **StartTransaction Method**

The **StartTransaction** method begins a transaction.

### **Applies To**

<u>DistributionStore Object</u>

### **Syntax**

HRESULT StartTransaction( );

#### Remarks

The method returns S\_OK if it succeeds and DB\_E\_ERROROCCURRED if it fails.

# **Replication Distributor Interface Structures**

This section defines the structures exposed by the Replication Distributor Interface.

### **DISTCOMMANDDESC Structure**

This is the definition of the DISTCOMMANDDESC structure.

#### **Syntax**

```
typedef struct tagDISTCOMMANDDESC{
       PublicationID;
INT
INT
       ArticleID;
INT
       CommandID;
DISTCMDTYPE CommandType;
BOOL fPartialCommand;
LPSTR
        szCommand;
BYTE*
        pXactID;
BYTE*
LPSTR
        pXactSeqno;
        szOriginator;
        szOriginatorDB;
LPSTR
} DISTCOMMANDDESC;
```

Part	Description
PublicationID	Publication ID.
ArticleID	Article ID.
CommandID	Uniquely identifies commands within a transaction. Each command added to a transaction should have a unique, monotonically increasing command ID.
CommandType	Identifies the type of the command. The Microsoft® SQL Server™ Distribution Agent can handle the following command types:
	• SQL_CMDTYPE_SQL = Transact-SQL command.
	• SQL_CMDTYPE_SCRIPT = File path to a Transact-SQL script file.

	• SQL_CMDTYPE_NATIVE_BCP = File path to a . <b>bcp</b> file in native format.	
	• SQL_CMDTYPE_CHAR_BCP = File path to a . <b>bcp</b> file in character format.	
	• SQL_CMDTYPE_WORKINGDIR = File path to the directory in which snapshot files are stored. Used by the Distribution Cleanup Agent.	
fPartialCommand	d Determines whether the command wraps more than one row.	
szCommand	Command text.	
pXactID	Pointer to the transaction ID.	
pXactSeqno	Pointer to the transaction sequence number.	
szOriginator	Name of the originating server.	
szOriginatorDB	Name of the originating database.	

## **CommandType Text Formats**

The *CommandType* member of the DISTCOMMANDDESC structure requires specific text formatting when specifying these items

- Transact-SQL data types
- Working directories
- Schema files
- **bcp** files

## **Transact-SQL Data Types**

When using DISTCMDTYPE\_SQL, data types in Transact-SQL statements have these formats.

Data type	Format	Comment
Datetime <b>datetime</b>	{ts 'yyyy'mm'dd hh'mm'ss[.mmm]'}	Milliseconds are optional.
smalldatetime		
Binary binary varbinary	{ <b>b</b> 'data'}	Where data is one or more characters within the range: [0-9a-f]. It should not contain a leading 0x.
Long Binary image	{ <b>lb</b> 'data'}	Where data is one or more characters within the range: [0-9a-f]. It should not include a leading 0x.
Character char varchar	'data'	Where data is any sequence of characters. Single quotation marks within the data portion must be expanded to two adjacent single quotation marks.

Long Character	{lc 'data'}	Where data is any sequence of
text		characters. Single quotation marks
		within the data portion must be
		expanded to two adjacent single
		quotation marks.

#### **Working Directories**

When using DISTCMDTYPE\_WORKINGDIR, include escape characters (\) in the file paths.

```
const char szDir[] = ":\\Program Files\\Microsoft SQL Server\\mssql\\r
DISTCOMMANDDESC
                                                                                                    aCommand[60];
                                       NumCommands = 0:
INT
DistByteArray
                                                              XactId;
                                                              XactSeqno;
DistByteArray
                                                 szWrkDir[ MAX PATH];
CHAR
// Get the current working directory.
sprintf(szWrkDir, "%c", (char)(_getdrive() + 'A' - 1) );
strcat(szWrkDir, szDir);
// Set working directory.
NumCommands++;
aCommand[i].PublicationId = 1;
aCommand[i].ArticleId = 1;
aCommand[i].CommandId = NumCommands;
a Command [i]. Command Type = DISTCMD TYPE\_SQL\_WORKING DISTRIBUTION FOR STANDING PROPRIBUTION 
aCommand[i].fPartialCommand = FALSE;
aCommand[i].pXactId = (BYTE *)&XactId;
aCommand[i].pXactSeqno = (BYTE *)&XactSeqno;
aCommand[i].szOriginator = NULL;
aCommand[i].szOriginatorDB = NULL;
```

```
aCommand[i].szCommand = (LPSTR)szWrkDir; i++;
```

#### Schema Files

When using DISTCMDTYPE\_SCRIPT, include escape characters (\) in file paths.

```
const char szDir[] = ":\\Program Files\\Microsoft SQL Server\\mssql\\r
const char szScriptCmd[] = "%ssamptab.sch";
DISTCOMMANDDESC
                          aCommand[60];
          pszCmdBuf[60][255];
char
          NumCommands = 0;
INT
DistByteArray
                XactId:
DistByteArray
                XactSeqno;
// Get the current working directory.
sprintf(szWrkDir, "%c", (char)(_getdrive() + 'A' - 1) );
strcat(szWrkDir, szDir);
// Execute script - table schema.
NumCommands++;
aCommand[i].PublicationId = 1;
aCommand[i].ArticleId = 1;
aCommand[i].CommandId = NumCommands;
aCommand[i].CommandType = DISTCMDTYPE_SCRIPT;
aCommand[i].fPartialCommand = FALSE;
aCommand[i].pXactId = (BYTE *)&XactId;
aCommand[i].pXactSeqno = (BYTE *)&XactSeqno;
aCommand[i].szOriginator = NULL;
aCommand[i].szOriginatorDB = NULL;
sprintf(pszCmdBuf[i], szScriptCmd, szWrkDir);
aCommand[i].szCommand = pszCmdBuf[i];
```

i++;

Use this format in .sch files:

SET QUOTED IDENTIFIER ON

GO

SET ANSI\_PADDING OFF

GO

CREATE TABLE [Samptbl1] (C1 INT, C2 VARCHAR(20))

GO

#### **bcp Files**

When using DISTCMDTYPE\_CHAR\_BCP or DISTCMDTYPE\_NATIVE\_BCP, you can use these switches with the **sync** command.

Switch	Description
-t	Destination table.
-0	Destination owner.
-d	Data file.
-f	Field delimiter. Default field delimiter: \n <x\$3>\n</x\$3>
-r	Row delimiter. Default row delimiter: \n<,@g>\n
-u	Unicode. This switch applies only when using DISTCMDTYPE_NATIVE_BCP type <b>bcp</b> files.
-m	Denotes that the file is a character <b>bcp</b> file with a Unicode marker at the beginning of the file.

Include escape characters (\) in file paths. Begin switch arguments with quotation marks (") and end the arguments with a backslash and quotation marks (\").

const char szDir[] = ":\\Program Files\\Microsoft SQL Server\\mssql\\r
const char szBCPCmd[] = "sync -t\"SampTable1\" -d\"%ssamptab.bcp\
DISTCOMMANDDESC aCommand[60];

```
pszCmdBuf[60][255];
char
INT
          NumCommands = 0;
                XactId;
DistByteArray
DistByteArray
                XactSeqno;
// Get the current working directory.
sprintf(szWrkDir, "%c", (char)(_getdrive() + 'A' - 1) );
strcat(szWrkDir, szDir);
// Import data - char bcp.
NumCommands++;
aCommand[i].PublicationId = 1;
aCommand[i].ArticleId = 1;
aCommand[i].CommandId = NumCommands;
aCommand[i].CommandType = DISTCMDTYPE_CHAR_BCP;
aCommand[i].fPartialCommand = FALSE;
aCommand[i].pXactId = (BYTE *)&XactId;
aCommand[i].pXactSeqno = (BYTE *)&XactSeqno;
aCommand[i].szOriginator = NULL;
aCommand[i].szOriginatorDB = NULL;
sprintf(pszCmdBuf[i], szBCPCmd, szWrkDir);
aCommand[i].szCommand = pszCmdBuf[i];
i++;
```

## **DISTERRORDESC Structure**

This is the definition of the DISTERRORDESC structure.

### **Syntax**

```
typedef struct tagDISTERRORDESC{
INT SourceType;
CHAR szSourceName[31];
CHAR szErrorCode[31];
BSTR bstrErrorText;
} DISTERRORDESC;
```

Part	Description
	Identifies the type of the command. Microsoft® SQL Server™ 2000 supports the following command types:  • APPLICATION  • DISTOLE
	• OS
szSourceName	Name of the component responsible for the error (for example, ODBC).
szErrorCode	Error code string of the source.
bstrErrorText	Error text.

## **Replication Programming Samples**

The following samples illustrate Microsoft® SQL Server<sup>TM</sup> 2000 replication application development in various environments and languages, including Transact-SQL, Microsoft Visual C++®, and Microsoft Visual Basic®. The samples and associated headers and libraries are required for successful execution of several sample applications.

# To install the samples and related headers and libraries using the SQL Server Installation Wizard

- 1. On the Setup Type page, select **Custom**.
- 2. On the Select Components page, under **Components**, select **Code Samples**.
- 3. In the **Select Sub-Components** dialog box, select **Replication**.
- 4. On the Select Components page, under **Components**, select **Development Tools.**
- 5. In the **Select Sub-Components** dialog box, select **Headers and Libraries**.

After installation is complete, if you have accepted the default installation location, the path C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Samples\sqlrepl will be on your computer. The sqlrepl directory contains an executable file, unzip\_sqlrepl.exe, which expands the samples into useable files. Double-click unzip\_sqlrepl.exe, and you will be prompted to enter that path to the folder where you want the samples stored. The samples assume that the default is selected, and samples are not installed to an alternate location. The expansion adds several subdirectories to sqlrepl.

The Visual C++ samples were tested with Visual C++ version 6.0, Service Pack 3. The Visual Basic samples were tested with Visual Basic version 6.0, Service

Pack 3. The samples have been run on Microsoft Windows NT® version 4.0, Service Pack 6, and on Microsoft Windows® 2000 operating systems. They have not been compiled or tested on any other hardware platform supported by any other compiler.

For Visual C++ samples to compile, header and libraries file paths must be set properly to obtain the required replication files. After installation, set your project options for **include** files to C:\Microsoft SQL Server\80\Tools\Devtools\Include, and set your options for the **library** files to C:\Microsoft SQL Server\80\Tools\Devtools\Lib.

# **Replication Syntax Conventions**

Replication programming samples use the following conventions to distinguish elements of text.

Convention	Used for
UPPERCASE	Constants and enumerated data types.
courier new	Sample command lines and program code.
italic	Information that the user or the application must provide.
bold	Replication component objects; object events, methods or properties; data types; and other syntax that must be typed exactly as shown.

## **Replication ActiveX Control Samples**

Microsoft® SQL Server<sup>TM</sup> 2000 comes with the following sample applications to help you use Microsoft ActiveX® replication controls in your application.

Sample	Description
Using SQL Merge and SQL	Microsoft Visual Basic® sample that
Distribution Controls in a	demonstrates how to include the SQL
Custom Visual Basic	Distribution and SQL Merge controls in a
Application	custom application.
Using SQL Merge and SQL	Microsoft Visual C++® sample that
Distribution Controls in a	demonstrates how to include the SQL
Custom Visual C++	Distribution and SQL Merge controls in a
Application	custom application.
Using SQL Merge and SQL	HTML sample that demonstrates how to
Distribution Controls in a	include the SQL Distribution and SQL Merge
Custom Web Application	controls in a custom application.
Creating a Transformable	Visual Basic sample that creates and saves a
Subscription Using Visual	Data Transformation Services (DTS) package
<u>Basic</u>	to DTS LocalPackages. The package contains
	Data Driven Query code to synchronize a
	subscription, and ActiveX code to transform
	the data before it is entered into the
	subscription database.

# Using SQL Merge and SQL Distribution Controls in a Custom Visual Basic Application

The programs in the **replctrl** folder are samples of how to include the SQL Merge and SQL Distribution controls in a custom application. This sample is located in C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Samples\sqlrepl\replctrl.

### To run the sample programs

- 1. On the computer that will be a Publisher with a local Distributor, verify that the SQL Server Agent is running. If it is not running, start it.
- 2. Open SQL Query Analyzer, open \Samples\sqlrepl\replctrl\instsamp.sql, and then run **instsamp.sql**. This SQL script configures the computer for publishing and distribution, enables the computer as a Subscriber, adds a distribution database with the name **distributor**, creates a transactional publication named SampleTransactionalPublication, and creates a merge publication named SampleMergePublication.
- 3. The **instsamp.sql** script creates pull and push subscriptions for each publication and creates and configures a database called **Northwind\_replica** as the subscription database. Any warnings from the script regarding tables that have been created with a maximum row size that exceeds the maximum number of bytes per row can be ignored. The computer is now configured as a Publisher with a local Distributor and enabled as a Subscriber.

The **Northwind** publication database will have two publications: SampleTransactionalPublication and SampleMergePublication. The instance of SQL Server contains a new subscription database, **Northwind\_replica**.

4. After the **instsamp.sql** script has completed successfully, in SQL

Server Enterprise Manager, expand **Replication Monitor**, expand the **Agents** folder, and then click the **Snapshot Agents** folder. In the right pane, for each agent listed, right-click, and then click **Start Agent**. This starts the Snapshot Agents for each publication and creates a snapshot for each publication.

- 5. Start Visual Basic, and then open \Samples\sqlrepl\replctrl\VB\replsamp.vbp.
- 6. On the **Project** menu, click **References**, and then in the **Available References** list, verify that the following controls are selected:
  - Microsoft SQL Distribution Control 8.0
  - Microsoft SQL Merge Control 8.0
  - Microsoft SQL Snapshot Control 8.0
  - Microsoft SQL Replication Errors 8.0
- 7. On the **File** menu, click **Make ReplSamp.exe**, and then save the executable to a directory.
- 8. Run replsamp.exe. The following options will be displayed:

### **Generate Snapshot for Transactional Publication**

Creates a snapshot for the Northwind transactional publication. The snapshot activity can be monitored interactively in SQL Server Enterprise Manager using Replication Monitor and the Agents folder.

### **Generate Snapshot for Merge Publication**

Creates a snapshot for the Northwind merge publication. The snapshot activity can be monitored interactively in SQL Server Enterprise Manager using Replication Monitor and the Agents folder.

### **Update Transactional Subscription Tables**

Runs the Distribution Agent and applies the snapshot and schema at the subscription database, **Northwind\_replica**. After the **SQL Replication Sample** dialog box shows that the task has completed, the **Northwind\_replica** subscription database shows the new tables. The distribution activity can be monitored interactively in SQL Server Enterprise Manager using Replication Monitor and the Agents folder.

### **Update Merge Subscription Tables**

Runs the Merge Agent and applies the snapshot data and schema at the merge subscription database, **Northwind\_replica**. After the **SQL Replication Sample** dialog box shows that the task has completed, the **Northwind\_replica** subscription database shows the new tables. The merge activity can be monitored interactively in SQL Server Enterprise Manager using Replication Monitor and the Agents folder.

# Using SQL Merge and SQL Distribution Controls in a Custom Visual C++ Application

The programs in the **replctrl** folder are samples of how to include the SQL Merge control and the SQL Distribution control in a custom application. This sample is located in C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Samples\sqlrepl\replctrl.

### To run the sample programs

- 1. On the computer that will be a Publisher with a local Distributor, verify that the SQL Server Agent is running. If it is not running, start it.
- 2. Open SQL Query Analyzer, open \Samples\sqlrepl\replctrl\instsamp.sql, and then run **instsamp.sql**. This SQL script configures the computer for publishing and distribution, enables the computer as a Subscriber, adds a distribution database with the name **distributor**, creates a transactional publication named SampleTransactionalPublication, and creates a merge publication named SampleMergePublication.
- 3. The **instsamp.sql** script creates pull and push subscriptions for each publication and creates and configures a database called **Northwind\_replica** as the subscription database. Any warnings from the script regarding tables that have been created with a maximum row size that exceeds the maximum number of bytes per row can be ignored. The computer is now configured as a Publisher with a local Distributor and is enabled as a Subscriber. The **Northwind** database will have two publications: SampleTransactionalPublication and SampleMergePublication. The instance of SQL Server contains a new subscription database, **Northwind\_replica**.
- 4. After the **instsamp.sql** script has completed successfully, in SQL

Server Enterprise Manager, expand **Replication Monitor**, expand the **Agents** folder, and then click the **Snapshot Agents** folder. In the right pane, for each agent listed, right-click, and then click **Start Agent**. This starts the Snapshot Agent and creates a snapshot for each publication.

To use the Microsoft Visual C++® sample code that demonstrates how to include the SQL Distribution control and the SQL Merge control in a custom application, the two programs must first be built into executables.

### To build the distribution sample executable using Visual C++

- 1. Open Visual C++. On the main menu, click **File**, click **Open Workspace**, navigate to the C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp directory, and then open distsamp.dsw.
- 2. On the **Tools** menu, click **Options**, and then on the **Directories** tab, in the **Show directories for** drop-down list, select **Include files**. Add the path C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Include. This path assumes the samples were installed to the default directory. If this path does not exist, navigate to the path where the include files were installed.
- 3. On the **Directories** tab, in the **Show directories for** drop-down list, click **Library files**, and then add the path C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Lib.
  - This path assumes the samples were installed to the default directory. If this path does not exist, navigate to the path where the Lib files were installed.
- 4. On the **Build** menu, click **Build distsamp.exe**. The default location of the resulting distsamp.exe will be in C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp\Debug or C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp\Release, depending on the build configuration.

### To build the distribution sample executable using a batch command

- 1. Open a command prompt window.
- 2. Navigate to C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp.
- 3. Run the batch file **builddst.cmd** with the following parameters: builddst.cmd [x86] [debug|retail] [clean]
  - Enter builddst.cmd /? for help.

To run the sample, run the executable from its location using a command prompt window.

The **distsamp.exe** sample activates the Distribution Agent and moves the data from the snapshot into the tables defined as articles for the transactional subscription in the database **Northwind\_replica**.

### To build the merge sample executable using Microsoft Visual C++.

- 1. Open Visual C++. On the main menu, click **File**, click **Open Workspace**, and then navigate to the C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp directory and open mergsamp.dsw.
- 2. On the **Tools** menu, click **Options**, and then on the **Directories** tab, in the **Show directories for** drop-down list, click **Include files**. Add the path C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Include.
  - This path assumes the samples were installed to the default directory. If this path does not exist, navigate to the path where the include files were installed.
- 3. On the **Directories** tab, in the **Show directories for** drop-down list, click **Library files**, and then add the path C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Lib.

This path assumes the samples were installed to the default directory. If this path does not exist, navigate to the path where the Lib files were installed.

4. On the **Build** menu, click **Build mergsamp.exe**. The default location of the resulting mergsamp.exe will be in C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp\Debug or C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp\Release, depending on the build configuration.

### To build the merge sample executable using a batch command

- 1. Open a command prompt window.
- 2. Navigate to C:\Program Files Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\replctrl\cpp.
- 3. Run the batch file **buildmrg.cmd** with the following parameters: buildmrg.cmd [x86] [debug|retail] [clean]
  Enter buildmrg.cmd /? for help.

To run the sample, run the executable from its location using a command prompt window.

The **mergsamp.exe** sample activates the Merge Agent and moves data from the snapshot into the tables defined as articles for the merge subscription in the **Northwind\_replica\_html** database.

# Using SQL Merge and SQL Distribution Controls in a Web Application

The programs in the **replctrl** folder are samples of how to include the SQL Merge control and the SQL Distribution control in a custom application. This sample is located in C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Samples\sqlrepl\replctrl.

#### To run the sample programs

- 1. On the computer that will be a Publisher with a local Distributor, verify that the SQL Server Agent is running. If it is not running, start it.
- 2. Open SQL Query Analyzer, open \Samples\sqlrepl\replctrl\instsamp.sql, and then run instsamp.sql. This SQL script configures the computer for publishing and distribution, enables the computer as a Subscriber, adds a distribution database with the name distributor, creates a transactional publication named SampleTransactionalPublication and a merge publication named SampleMergePublication. instsamp.sql creates a Snapshot Agent for both publications.
- 3. The instsamp.sql creates pull and push subscriptions for each publication and creates and configures a database called Northwind\_replica as the subscription database. Any warnings from the script regarding tables that have been created with a maximum row size that exceeds the maximum number of bytes per row can be ignored.

The **Northwind** database will have two publications: SampleTransactionalPublication and SampleMergePublication. The SQL Server contains a new subscription database, **Northwind\_replica**.

4. After the **instsamp.sql** script has completed successfully, in SQL

Server Enterprise Manager, expand **Replication Monitor**, expand the **Agents** folder, and then click the **Snapshot Agents** folder. In the right pane, for each agent listed, right-click, and then click **Start Agent**. This starts the Snapshot Agent and creates a snapshot for each publication.

### To modify the HTML sample program

- 1. Open the \Samples\sqlrepl\replctrl\html\replsamp.htm file in Notepad or other HTML editor. There are comments in the file noting that the local computer name needs to be specified in the line of code following the comment.
- 2. Save and the file.

### To execute the HTML sample program

• Using Microsoft Internet Explorer or another Internet browser, open the replsamp.htm file. These options are available:

### **Synchronize Transactional Subscription**

Runs the Distribution Agent and applies the snapshot data and schema at the subscription database, **Northwind\_replica**. After the HTML page has completed, the snapshot will be applied, and the **Northwind\_replica** database will show the new tables, with each table corresponding to an article from the publication. The distribution activity can be monitored interactively in SQL Server Enterprise Manager using **Replication Monitor** and the **Agents** folder.

### **Synchronize Merge Subscription**

Runs the Merge Agent and applies the snapshot data and schema at the subscription database, **Northwind\_replica**. After the HTML Page has completed, the snapshot will be replicated, and the **Northwind\_replica** database will show the new tables, with each table corresponding to an article from the publication. The merge activity can be monitored interactively in SQL Server Enterprise Manager using **Replication Monitor** 

and the **Agents** folder.

## Creating a Transformable Subscription Using Visual Basic

Advanced users familiar with the Data Transformation Services (DTS) object model can build their own transformable subscription DTS packages in Microsoft® Visual Basic<sup>TM</sup>. For information about programming to the DTS object model, see <u>DTS Programming Reference</u>.

A transformable subscription is a subscription in which the data is modified as it flows from Publisher to Subscriber. In replication programming, a Visual Basic program for a transformable subscription closely resembles that of a DTS Visual Basic program used outside of replication. This sample is located in C:\Program Files\Microsoft SQL Server\80\Tools\DevTools\Samples\sqlrepl\repldts.

### **How to Run the Transformable Subscription Sample**

The program in the **repldts** folder is a sample of how to write a transformable subscription using Visual Basic.

### To run the sample program

- 1. On the computer that will be a Publisher with a local Distributor, verify that the SQL Server Agent is running. If it is not, start it.
- 2. Open Visual Basic 6.0. Open ReplDTS.vbp, and then open the code window for ModReplDTS (ReplDTS.bas).
- 3. Select the following **Project** / **References**:
  - Microsoft DTSPackage Object Library (required)
  - **Microsoft DTSDataPump Scripting Object Library** (required to use a Microsoft ActiveX® script or custom transformation)

- **Microsoft DTS Custom Tasks Object Library** (required to use one of the DTS custom tasks)
- 4. Edit the following line of code to include your own connection information: goPackage.SavetoSQLServer "MyServerName", "sa", ""
- 5. In Visual Basic, start the ReplDTS program.
- 6. When the program is finished executing, you should receive a message indicating the Employees package was saved successfully. When you receive this message, save the project, and then close Visual Basic.
- 7. In SQL Server Enterprise Manager, ensure that your server is configured for replication.
- 8. In SQL Server Enterprise Manager, click **Data Transformation Services**, click **Local Packages**, and then on the **Action** menu, click **Refresh** to refresh the view. The package Employees should appear in the right pane.
- 9. Open SQL Query Analyzer, open the **repldts.sql** script supplied with sample, and then edit the **@subscriber** parameter of **sp\_addsubscription** so that it contains your server name. The **sp\_addsubscription** stored procedure is the last SQL statement in the file.
- 10. Run the **repldts.sql** script. You can run the entire script at once, or you can run the script a block at a time and check each message.
- 11. After the **repldts.sql** script has completed successfully, in SQL Server Enterprise Manager, expand **Replication Monitor**, expand the **Agents** folder, and then click the **Snapshot Agent** folder.

- 12. In the right pane, right-click the Snapshot Agent for the **Employees** publication, and then click **Start Agent**.
- 13. When the Snapshot Agent has completed, view or query the data in **MyEmployees** table of the subscription database **ReplDTS\_SubDB**, and then compare it to the data in the **Employees** table of the publication database ReplDTS\_PubDB.
- 14. The **Address** column of the **MyEmployees** table in the subscription database, **ReplDTS\_subDB**, contains data concatenated from several columns of the **Employees** table of the publication database.

### **Examining the Sample Code**

This section describes key parts of the sample Visual Basic program.

• The Publisher connection is always set to the Microsoft SQL Server Replication OLE DB Provider for DTS. This is specified in the section of code where the connections are created. In addition, a required property specified for this provider is the column list for ConnectionProperties, which provides DTS Designer with the number and names of the source columns in the package. The Subscriber connection (not shown here) is set to the Microsoft OLE DB Provider for SQL Server.

Dim oConnection As DTS.Connection
Set Connection = goPackage.Connections.New("SQLReplication oConnection.Name = "Publisher article 'Employees'" oConnection.ID = 1
oConnection.ConnectImmediate = False

oConnection.ConnectionProperties("Column List") = \_ "[EmployeeID],[LastName],[FirstName],[Title], [BirthDate],[HireDate],[Address],[City],[Region], [PostalCode],[Country],[HomePhone],[Extension] [ReportsTo]"

goPackage.Connections.Add oConnection

### Set oConnection = Nothing

 If the destination table schema is not defined, use an Execute SQL custom task containing an SQL statement to generate the schema for the destination (Subscriber) table.

Use additional Execute SQL custom tasks to define SQL scripts to be applied after the data has been copied to the destination (for example, a script containing index generation statements (not shown)).

Use the following conventions when naming tasks in a replication DTS program (required by the replication agents):

- For a Data Driven Query task, the task name is the same as the article name.
- For custom tasks, such as an Execute SQL task, the name is a concatenated string consisting of the article name, the prefix "pre" (if the task executes before the snapshot data is copied) or "post" (if the task executes after the snapshot data is copied), and an optional part ("ignore\_error") if an instruction is given to continue program execution when a query script error is encountered. To determine what the name should be, run sp\_helparticledts in SQL Query Analyzer. For more information, see sp\_helparticledts.

In the following code sample, the Execute SQL task name "Employees\_pre\_ignore\_error" (line 5) means that the article name is **Employees**, the task occurs before the snapshot data is copied, and that program execution should continue if a script error is encountered.

Dim oTask As DTS.Task
Dim oCustomTask0 As DTS.ExecuteSQLTask
Set oTask = goPackage.Tasks.New("DTSExecuteSQLTask")
Set oCustomTask0 = oTask.CustomTask
oCustomTask0.Name = "Employees\_pre\_ignore\_error"
oCustomTask0.Description = "Pre script for article employees"
oCustomTask0.SQLStatement = \_

```
"If object_id('MyEmployees') is NOT NULL _
 BEGIN Drop Table MyEmployees END _
 Create Table MyEmployees _
 ([EmployeeID] [int] NOT NULL,
  [LastName] [nvarchar] (20) NOT NULL,
  [FirstName] [nvarchar] (10) NOT NULL,
  [Title] [nvarchar] (30) NULL,
  [Birthdate] [datetime] NULL,
  [HireDate] [datetime] NULL,
  [Address] [nvarchar] (255) NULL,
  [HomePhone] [nvarchar] (24) NULL,
  [Extension] [nvarchar] (4) NULL,
  [ReportsTo] [int] NULL,
  CONSTRAINT [PK_Employees] PRIMARY KEY _
   CLUSTERED([EmployeeID]))"
CustomTask0.ConnectionID = 2
goPackage.Tasks.Add oTask
Set CustomTask0 = Nothing
Set oTask = Nothing
```

• With transformable subscriptions, data movement is always done with a Data Driven Query task, never with a Transform Data task, which is commonly used in DTS packages that do not use replication. Therefore, you must define a Data Driven Query custom task:

Dim oTransformation As DTS.Transformation

Dim oTransProps As DTS.Properties

Dim oColumn As DTS.Column

Dim oCustomTask1 As DTS.DataDrivenQueryTask

Set oTask = goPackage.Tasks.New("DTSDataDrivenQueryTasl

Set oCustomTask1 = oTask.CustomTask

oCustomTask1.Name = "Employees"

oCustomTask1.Description = "Transformations for article Emp oCustomTask1.SourceConnectionID = 1

```
oCustomTask1.SourceObjectName = "nothing" 'Experiment oCustomTask1.DestinationConnectionID = 2 oCustomTask1.DestinationObjectName = "MyEmployees"
```

 Among the properties you need to define for the Data Driven Query task are the parameterized queries associated with each type of incremental update (INSERT, UPDATE, and DELETE). The parameterized query works by selecting for data movement source rows that satisfy the conditions in the query statement.

```
oCustomTask1.InsertQuery = "INSERT INTO _
MyEmployees values (?,?,?,?,?,?,?,?)"
oCustomTask1.UpdateQuery = "UPDATE _MyEmployees _
SET LastName=?, FirstName=?, Title=?, Birthdate=?, _
HireDate=?, Address=?, HomePhone=?, Extension=?, _
ReportsTo=? where EmployeeID=?"
oCustomTask1.DeleteQuery = "DELETE MyEmployees _
WHERE EmployeeID = ?"
```

Another essential set of operations is setting the source and destination column collections. Because one of the transform operations performed by the DTS package is a concatenation of several of the source columns, the number of destination columns added to the destination collection is fewer than the number of source columns added to the source collection. An example of one source column and one destination column added to their respective collections is shown here. The column "Address" for the destination is actually the concatenation of the source columns "Address", "City", "Region", and "Postal Code". Set oColumn = oTransformation.DestinationColumns.New("Hi oColumn.Name = "HireDate" oColumn.Ordinal = 6
 oTransformation.DestinationColumns.Add oColumn Set oColumn = Nothing

Set oColumn = oTransformation.DestinationColumns.New("Ac

```
oColumn.Name = "Address"
oColumn.Ordinal = 7
oTransformation.DestinationColumns.Add oColumn
Set oColumn = Nothing
```

 To complete the Data Driven Query task transformation, add column definitions (in sequential order) to each of the incremental update operations described earlier. For example, because an INSERT operation was defined with parameters for the 10 destination columns of the article, INSERT column definition code for each of the 10 columns is required (only the first two are shown here):
 Set oColumn = oCustomTask1.InsertQueryColumns.New("Em

oColumn.Name = "EmployeeID"
oColumn.Ordinal = 1
oCustomTask1.InsertQueryColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oCustomTask1.InsertQueryColumns.New("Las oColumn.Name = "LastName" oColumn.Ordinal = 2 oCustomTask1.InsertQueryColumns.Add oColumn Set oColumn = Nothing

• • •

Similar code is used for the UPDATE and DELETE queries. When defining the columns for those queries, remember that the order of the columns must follow the order of the parameters specified by the question mark (?) characters in the **InsertQuery** and **DeleteQuery** definitions. For example, in this sample, the DELETE query uses only the **EmployeeId** column; therefore, only code for that column is used.

Set oColumn = oCustomTask1.DeleteQueryColumns.New("En oColumn.Name = "EmployeeID" oColumn.Ordinal = 1

### oCustomTask1.DeleteQueryColumns.Add oColumn Set oColumn = Nothing

• In the example, a Microsoft ActiveX® script performs the actual transformation of data, which is the concatenation of several columns. The entire ActiveX script code should be viewed in the sample. This example shows how to set the transform server property to handle scripts, and how to set the line of script code used to perform the column concatenation.

Set oTransProps = oTransformation.TransformServerProperties ...

```
oTransProps("Text") = oTransProps("Text") &
" DTSDestination(""Address"") = DTSSource(""Address"")
& "","" & DTSSource(""City"") &"",""& DTSSource(""Regi
&"",""& DTSSource(""PostalCode"")" & vbCrLf
```

•••

- Transformable subscription DTS packages are typically saved to an instance of SQL Server. They can also be saved as a .dts structured storage file, but cannot be saved to the repository. In the following line of code, the package is saved to an instance of SQL Server: GoPackage.SaveToSQLServer "myServerName", "sa", ""
- DTS includes several options for handling data conversions. These options are handled by a set of transformation flags whose values can be viewed in the Visual Basic Object Browser under the **DTSPump** component, **DTSTransformFlags** enumeration. When building a replication DTS package in Visual Basic, if an ActiveX script transformation is used, as in this sample, the transformation flags do not need to be explicitly set. If you build a replication DTS package without ActiveX scripts (using only Copy Column mappings), the **TransformFlags** property must be explicitly set to **DTSTransformFlag\_Default** (a value of 63).

oTransformation.TransformFlags = 63

## **Merge Replication Samples**

Microsoft® SQL Server $^{\text{TM}}$  2000 comes with the following samples to help you implement merge replication in your application.

Sample	Description
Subscriber-Based Resolver using	Microsoft Visual C++ language
C++ and a Stored Procedure	sample that builds a stored procedure
	custom resolver to use in merge
	replication.
Transact-SQL Custom Stored	Transact-SQL stored procedure that
Procedure Resolver	is the custom resolver used in a
	merge replication.
Generating Merge Dynamic	The procedures in this sample show
Snapshot Jobs	how to enumerate a list of users from
	within a Microsoft Windows® group
	and generate dynamic snapshot jobs
	for each user.

## Subscriber-Based Resolver Using C++ and a Stored Procedure

This sample application builds a custom stored procedure resolver that executes the stored procedure at the Subscriber. It contains a sample script to set up a publication on **Northwind**, and install the sample resolver that calls **sp\_authority\_resolver**. **sp\_authority\_resolver** retrieves the 'authority' value from the table at the Publisher and Subscriber, and the row with the highest authority will win and be returned to the merge process to be applied to both servers. If both rows have the same authority, the source table will win. The Microsoft® Visual C++ sample code is located in the \Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\resolver\subspres directory.

### To run the sample program

- 1. On the computer that will be the Publisher, verify that the SQL Server Agent is running. If it is not running, start it.
- 2. Verify that the computer is configured for publishing and distribution If it is not configured, configure it using the Configure Publishing and Distribution Wizard. The defaults given in the wizard are all acceptable for this sample.

### Transact-SQL Custom Stored Procedure Resolver

The program in the C:\Microsoft SQL

Server\80\Tools\Devtools\Samples\sqlrepl\deflt\_sp directory is a sample application, which builds a custom stored procedure resolver that executes at the Publisher. The resolver uses the **Northwind** sample database, a distributed query to obtain information from the Subscriber, and then computes the average price if the values between the Publisher and Subscriber are different. The resolver then makes the average price the resolved value, and logs the conflict at the Publisher so it can be viewed, and if necessary, changed. Finally, the application sends an e-mail stating that the price was changed to an average due to a conflict. If any columns other than price have changed, the resolver uses the values from the Publisher.

The resolver is designed to run on the **Products** table in **Northwind** sample database. It is assumed that the sample code was saved to the default directory offered during installation, and that the files can be found in C:\Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\deflt\_sp.

**Note** To run this sample, two computers are required. For the distributed query to work, the two computers must both be running the Microsoft Windows NT 4.0 or Windows 2000 operating system, or the Publisher must be running on Windows 2000 with the linked server running on Windows NT 4.0.

### To run the sample program

- 1. On the computer that will be the Publisher, verify that the SQL Server Agent is running. If it is not running, start it.
- 2. Using the **Create Publication Wizard**, create a merge publication based on the **Northwind** database, and then select the **Products** table as an article in the publication.

How to create publications and define articles

### **Generating Merge Dynamic Snapshot Jobs**

This sample generates multiple merge dynamic snapshot jobs for a set of Windows Group users. After you create a merge publication with a dynamic filter and generate a standard snapshot, you can run the extended stored procedure and stored procedures included with this sample to generate dynamic snapshot jobs.

The sample extended stored procedure enumerates a Windows group and identifies the members for which dynamic snapshot jobs will be created. After the members are identified, you can execute the sample stored procedure that will create the actual dynamic snapshot jobs. After the dynamic snapshot jobs are created, you will need to start them manually or ensure they will start according to a schedule.

If you have a dynamically filtered merge publication with multiple subscriptions, this sample shows how you can create multiple dynamic snapshot jobs for the users of those subscriptions. Dynamic snapshots provide the performance advantage of using SQL bulk copy program (**bcp**) files to apply data to a specific Subscriber when applying the initial snapshot. For more information, see <a href="Dynamic Snapshots">Dynamic Snapshots</a>.

The sample code is located in the \Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\dynsnapjob directory.

### To run the sample programs

- 1. In the \Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\dynsnapjob\xp directory, copy the compiled **xp\_enumntusers.dll** into your Windows system32 folder.
- 2. Register the name of the extended stored procedure to Microsoft SQL Server™ by running the following Transact-SQL command in SQL Query Analyzer:

go

use master

exec sp\_addextendedproc 'xp\_enumntusers', 'xp\_enumntusers' go

- 3. To create the stored procedure, open SQL Query Analyzer, and then run **spdynsnapsample.sql**, which is located in the \Samples\sqlrepl\dynsnapjob\sp directory.
- 4. Create a merge publication with dynamic filtering enabled (use a system function such as SUSER\_SNAME() as the criteria for the filter), and then generate a standard snapshot for the publication.
- 5. Allow a Windows Local (or Global) group access to the SQL Server instance that contains the merge publication in Step 4. This group needs to have public access only to the database on which you created the publication. This step can be performed using SQL Server Enterprise Manager.
- 6. Add the Windows Group to the Publication Access List (PAL), which is found in the merge publication properties.

You can now run the extended stored procedure and stored procedure to generate dynamic snapshot jobs for the Windows Group users. For more information about how to execute each procedure, see **Using xp\_enumntusers** and **Using sp\_addntgroupmergedynsnapshotjob**.

### Using xp\_enumntusers

This extended stored procedure enumerates through Microsoft® Windows® Local and Global group users.

```
xp_enumntusers [ @servername = ] 'domain_server_name'
, [ @groupname = ] 'group_name'
```

### **Arguments**

```
[@servername = ] 'domain_server_name'
```

The domain server to retrieve the group from. Use '.' or " for local server.

```
[@groupname = ] 'group_name'
```

The name of the group from which to enumerate users.

Both @servername and @groupname must be server strings.

### Using sp\_addntgroupmergedynsnapshotjob

This stored procedure generates a dynamic snapshot job for each user found when **xp\_enumntusers** was run. The stored procedure accepts parameters that allow you to specify the following:

- Generate a single dynamic snapshot job for all users or generate a separate dynamic snapshot job for each user.
- Filter the enumerated user using the LIKE operand.
- Specify schedule information.

, [ @active\_end\_date = ] active\_end\_date

```
sp_addntgroupmergedynsnapshotjob [ @publication = ] 'publication'
, [ @ntserver_name = ] 'ntserver_name'
, [ @group_name = ] 'group_name'
, [ @destination = ] 'destination'
, [ @job_name = ] 'job_name'
, [ @like_string = ] 'like_string'
, [ @as_one_job = ] as_one_job
, [ @frequency_type = ] frequency_type
, [ @frequency_interval = ] frequency_interval
, [ @frequency_subday = ] frequency_subday
, [ @frequency_subday_interval = ] frequency_relative_interval
, [ @frequency_relative_interval = ] frequency_recurrence_factor
, [ @active_start_date = ] active_start_date
```

, [ @active\_start\_time\_of\_day = ] active\_start\_time\_of\_day

, [ @active\_end\_time\_of\_day = ] active\_end\_time\_of\_day

### **Arguments**

[@publication = ] 'publication'

The name of the publication. *publication* is sysname, with no default.

[@ntserver\_name = ] 'ntserver\_name'

The Windows server or domain name on which to enumerate users. Use '.' for local server. *ntserver\_name* is **nvarchar(100)** with no default.

[@group\_name = ] 'group\_name'

The name of the group on the Windows server or domain. *group\_name* is **nvarchar(256)** with no default.

[@destination = ] 'destination'

The destination directory path. *destination* is **nvarchar(3500)** with no default.

[@job\_name = ] 'job\_name'

The job name. *job\_name* is **sysname** with a default of NULL. If NULL, a default job name is used. This parameter is ignored when @as\_one\_job=0

[@like\_string = ] 'like\_string'

The string to be used in WHERE clause used when creating jobs for user names. For example, the WHERE clause is constructed of "WHERE user name LIKE @like\_string". like\_string can contain any of the valid wildcard characters such as the percent sign ('%'). like\_string is nvarchar(1000) with a default of NULL.

[@as\_one\_job = ] as\_one\_job

Specifies how many dynamic snapshot jobs to create. *as\_one\_job* is a **bit** with a default of 1. **1** specifies that one dynamic snapshot job will be created for all users. **0** specifies that an individual dynamic snapshot job will be created for each user.

[**@frequency\_type** = ] frequency\_type

@frequency\_type int = 4,

Is a value indicating when the job is to be executed. *freq\_type* is **int** with a default of 4, and can be one of these values.

Value	Description
1	Once
4	Daily
8	Weekly
16	Monthly
32	Monthly, relative to
	frequency_interval
64	Run when SQL Server Agent service
	starts
128	Run when computer is idle

### [@frequency\_interval = ] frequency\_interval

Is the days that the job is executed. *freq\_interval* is **int** with a default of **1**, and the value used is dependent on the value of *freq\_type*.

Value of frequency_type	Effect on frequency_interval
1 (Once)	frequency_interval is ignored.
4 (Daily)	Every frequency_internval days.
8 (Weekly)	frequency_interval is one or more of the following (combined with an OR logical operator):  1 = Sunday 2 = Monday 4 = Tuesday 8 = Wednesday 16 = Thursday 32 = Friday 64 = Saturday
16 (Monthly)	On the <i>frequency_interval</i> day of the month.

32 (Monthly relative)	<pre>frequency_interval is one of the following:</pre>
	1 = Sunday 2 = Monday 3 = Tuesday 4 = Wednesday 5 = Thursday 6 = Friday 7 = Saturday 8 = Day 9 = Weekday 10 = Weekend day
64 (When SQL Server Agent service starts)	frequency_interval is unused.
128 (When computer is idle)	frequency_interval is unused.

### [@frequency\_subday = ] frequency\_subday

Specifies the units for *frequency\_subday\_interval*. *frequency\_subday* is **int** with a default of 8, and can be one of these values.

Value	Description of Units
0x1	At the specified time.
0x4	Minutes.
0x8	Hours.

### [@frequency\_subday\_interval = ] frequency\_subday\_interval

Is the number of *frequency\_subday\_type* periods to occur between each execution of the job. *frequency\_subday\_interval* is **int**, with a default of 6.

### [@frequency\_relative\_interval = ] frequency\_relative\_interval

Is the scheduled job's occurrence of *frequency\_interval* in each month, if *frequency\_interval* is 32 (monthly relative). *frequency\_relative\_interval* is

**int** with a default of 1, and can be one of these values.

Value	Description of Units
1	First
2	Second
4	Third
8	Fourth
16	Last

### [@frequency\_recurrence\_factor = ] frequency\_recurrence\_factor

Is the number of weeks or months between the scheduled execution of the job. *frequency\_recurrence\_factor* is used only if *frequency\_type* is **8**, **16**, or **32**. *frequency\_recurrence\_factor* is **int** with a default of 0.

### [@active\_start\_date = ] active\_start\_date

Is the date on which execution of the job can begin. *active\_start\_date* is **int** with a default of 0, which indicates today's date. The date is formatted as YYYYMMDD. If *active\_start\_date* is not NULL, the date must be greater than or equal to 19900101.

### [@active\_end\_date = ] active\_end\_date

Is the date on which execution of the job can stop. *active\_end\_date* is **int** with a default of 99991231, which indicates December 31, 9999. Formatted as YYYYMMDD.

### [@active\_start\_time\_of\_day = ] active\_start\_time\_of\_day

Is the time on any day between *active\_start\_date* and *active\_end\_date* to begin execution of the job. *active\_start\_time* is **int**, with a default of 0, which indicates 12:00:00 A.M. on a 24-hour clock. The value for this parameter must be entered using the form HHMMSS.

### [@active\_end\_time\_of\_day = ] active\_end\_time\_of\_day

Is the time on any day between *active\_start\_date* and *active\_end\_date* to end execution of the job. *active\_end\_time* is int with a default of 235959, which indicates 11:59:59 P.M. on a 24-hour clock. The value for this parameter

must be entered using the form HHMMSS.

## **Replication Distributor Interface Samples**

Microsoft® SQL Server™ 2000 comes with the following sample applications to help you implement the Replication Distributor Interface.

Sample	Description
Programming Snapshot or	Microsoft Visual Basic® SQL-DMO
Transactional Replication	application to configure the Publisher,
from Heterogeneous Data	publication, and a subscription for use in a
<u>Sources</u>	heterogeneous publishing environment. Also
	includes a Microsoft Visual C++® application
	to deliver transactions to the Distributor.

# Programming Snapshot or Transactional Replication from Heterogeneous Data Sources

SQL-DMO and the Replication Distributor Interface sample explains how to support a third-party publication. Because this sample uses transactional replication, this sample will work only with Microsoft® SQL Server™ 2000 Standard Edition and SQL Server 2000 Enterprise Edition.

SQL-DMO configures a third-party publication and Distributor using SQL Server. The Replication Distributor Interface logs any transaction that it marks for replication to this Distributor. The third-party publication uses the Distribution Agent to distribute the transactions as if they were coming from an instance of SQL Server.

If the samples were installed to the default location, the files used in this sample will be located in the C:\Microsoft SQL Server\80\Tools\Devtools\Samples\sqlrepl\samppub directory.

#### To run the sample programs

- 1. On the computer that will be the Publisher, verify that SQL Server Agent is running. If it is not running, start it.
- 2. In SQL Query Analyzer, open \Samples\sqlrepl\samppub\samppub.sql, and then execute it. This script creates two databases, **SampleSubscriberDB1** and **SampleSubscriberDB2**, and sets up a subscription to one of the databases. The script also configures the computer for publishing and distribution, adds a distribution database named **distribution**, and enables the computer as a Subscriber.
- 3. Execute the BAT file **iniwkdir.bat**, in the in \Microsoft SQL Server\80\Tools\Devtools\Samples\Sqlrepl\Samppub directory, by either double-clicking it in Microsoft Windows Explorer or running it at a command line. The result of the execution creates the working directory at C:\mssql8\repldata\UNC\samppub that contains the files **Samptab.sch** and **Samptab.bcp**. These schema and .bcp data files

will be used by the Visual C++ application.

- 4. On the Publisher, open Visual Basic, open the SQL-DMO application workspace, **samppub.vbp**, in the samppub\sqldmo directory. Build the application by selecting Make samppub.exe on the **File** menu. The resulting executable will be named samppub.exe, (the same name as the Visual C++ executable); therefore, it is recommended that this one be kept in the SQLDMO directory or in a directory of your choosing where the two executables can be distinguished from each other.
- 5. Run the Visual Basic version of the **samppub.exe** from inside Visual Basic on the **Run** menu by clicking **Start**, or by starting the executable from where it was saved. The application has one form with four buttons.
- 6. Click the **Create Sample Publication** button to create a publication on a third-party vendor. You can view the results in SQL Server Enterprise Manager. Expand **Replication**, and there will be a new folder called **Heterogeneous Publications**. There will be a folder called Sample Vendor, with a publication called **SamplePublication**. You can right-click **SamplePublication:SampleDatabase** and select **Properties** to view information about the publication. You will see that there is no subscription to this publication. If the **Replication** folder was already expanded, click **Refresh** to show the new objects or close and open SQL Server Enterprise Manager again.
- 7. Click the **Add Subscription** button. This creates a push subscription, with the data going to the **SampleSubscriberDB1** database. If you have not clicked the **Create Sample Publication** button, clicking this button will give an error because there will be no publication on which to add a subscription. After clicking this button, you can view the results in SQL Server Enterprise Manager. Expand **Replication**, expand **Heterogeneous Publications**, expand **Sample Vendor** Rightclick **SamplePublication:SampleDatabase** and select **Properties**. The **Subscriptions** tab shows the new subscription.

- 8. Open Visual C++, and then open the workspace **samppub.dsw** in the samppub\repldist directory. On the **Tools** menu, point to **Options**, and then click the **Directories** tab. In the **Show directories for** box, select **Include files** and point to the \Microsoft SQL Server\80\Tools\Devtools\Include directory. For the **Library** setting, navigate to \Microsoft SQL Server\80\Tools\Devtools\Lib. On the **Build** menu, select **Build samppub.exe**. The application will be compiled and saved to the samppub\repldist directory.
- 9. Run the Visual C++ version of samppub.exe . On the Build menu, click Execute samppub.exe. It can also be executed directly running it from the directory where it was saved. This will deliver transactions to the Distributor. The application will perform Snapshot Agent and Log Reader Agent operations for the publication. When run, messages will appear in the Command window, with the messages:
  Sample Publisher Agent Startup
  Added 1 Transaction(s) consisting of 3 Command(s)
  Sample Publisher Snapshot Agent Succeeded.
  Added 1 Transaction(s) consisting of 30 Command(s)
  Sample Publisher Logreader Agent Succeeded.

After a successful execution, you should see a table **SampleTable1** in the **SampleSubscriberDB1** database. Opening the table shows 20 rows of data.

- 10. Optionally, create a push subscription to the SampleSubscriberDB2 from the publication database using SQL Server Enterprise Manager. When creating this subscription, do not create an initial snapshot. Running the Visual C++ program results in transactions being propagated to both subscriptions.
- 11. Execute the Visual Basic **samppub.exe** file from where it was saved, and then click **Drop Subscription** to drop the push subscription. After clicking this button, you can view the results in SQL Server Enterprise

Manager. Expand the SQL Server Group, the **Replication** folder, **Heterogeneous Publications** folder, and **Sample Vendor**. On **SamplePublication:SampleDatabase**, right-click and select **Properties**. On the **Subscriptions** tab is a **Properties** button that shows the **Subscription Properties** dialog box. There are no subscriptions showing. If you have not clicked the **Create Sample Publication** button, clicking this button will give an error, as there will be no subscription to drop because one has not been created. If you have not clicked the **Add Subscription** button, clicking this button will give an error, as there is no subscription to drop.

12. Click the **Drop Sample Publication** button to removes the publication. In SQL Server Enterprise Manager, in the **Replication** folder, the **Heterogeneous Publications** folder is removed if this was the only heterogeneous publication on the computer. If you have not clicked the **Create Sample Publication** button, clicking this button will give an error, as there will be no Publication to drop. The subscription and the publication have been removed.

Data and schema files were installed on the local computer to a directory created by the **iniwkdir.bat**. The directory is C:\mssql8\repldata\uncsamppub. They were used by the Visual C++ application as the data copied into the **SampleTable1**.