## Microsoft Jet and Replication Objects Overview

Microsoft® Jet and Replication objects (JRO) allow you to add features to your application that are specific to the Microsoft Jet Database Engine. The Jet and Replication objects are based fundamentally on Microsoft® ActiveX® Data Objects (ADO), because they are connected to an ADO Connection object. However, the Jet and Replication objects function only with Microsoft Jet databases. For more information about ADO and the Connection object, refer to the ADO Programmer's Reference or the ADO Web site at http:\\www.microsoft.com\data.

With the Jet and Replication objects you can:

- Create and synchronize database replica sets.
- Compact a database, and specify options for the compacted database, such as passwords and encryption.
- Refresh the memory cache by writing pending data changes to the database, and reading the most recent data from the database to memory.

### The Jet Engine

The Jet database engine is represented by <u>JetEngine</u> objects. With **JetEngine** objects, you can make a compacted copy of a Jet database and include options for encrypting data, changing database engine versions, passing user ids and passwords, specifying the system database or registry information, setting or

overriding locale-specific collating orders, removing relationships, and repairing damaged replicas. You can also force any pending changes to the data to be processed by completely flushing the cache of database operations.

### **Replicas**

Replicated databases are represented by <u>Replica</u> objects. With **Replica** objects, you create new replicated databases, define the options for a specific replica (full, partial, design master), synchronize the changes between two replicas, and determine conflicts that occur during synchronization.

### **Filters**

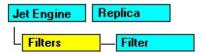
Partial replicas are created using <u>Filter</u> objects, which are collected in the <u>Filters</u> collection of a **Replica** object. Filter objects define criteria that limit the records that become replicated. Filters can be applied to tables or relationships in a partial replica.

For more information, search for a specific object, property, or method, or see the following topics:

- Jet and Replication Objects
- Jet and Replication Collections
- Jet and Replication Methods
- Jet and Replication Properties
- <u>Jet and Replication Examples</u>

# **Jet and Replication Object Model**

The following diagram shows how the Jet and Replication objects are represented and related.



## **Jet and Replication Objects**

### **JRO Object Summary**

Object	Description
<u>Filter</u>	Specifies criteria that limit the recordset of a replicated database.
<u>JetEngine</u>	Provides control over Jet Engine-specific features such as compacting databases and refreshing data from the cache.
<u>Replica</u>	Represents a copy of a replicated database.

The Jet and Replication objects consist of three objects: the **Filter**, the **Replica**, and the **JetEngine**. **Filter** objects can be contained in the <u>Filters</u> collection of a **Replica**.

### **Filter Objects**

You can create partial replicas by specifying criteria for a **Filter** object. With filter criteria, you can select specific data to be replicated. For example, you can choose to replicate only those records from a Customer table from a specific region.

### **JetEngine Objects**

Use **JetEngine** objects to compact databases and refresh data with pending changes from the cache. Compact databases to save space and to specify options for the new database copy such as encrypting the data, changing the database engine version, or setting the collating order by locale. Refresh data from the cache to force all pending database operations to be processed. This keeps data

in a recordset current even in an environment with many users and a high level of data processing.

### **Replica Objects**

Use **Replica** objects to create and maintain replicated databases. With replicas, you identify a design master, create new replicas, obtain information about the replicability of the objects in a database, and synchronize the changes between two replicas.

For more information about the properties and methods of a specific Jet and Replication object, see the specific reference topic for that object.

## Filter Object

Specifies criteria that limit the recordset of a replicated database.



### **Remarks**

Use a **Filter** object to replicate a subset of the records in a database. With filters, you can create partial replicas that contain specific data according to the criteria. A **Filter** object can be contained in the <u>Filters</u> collection of a replica.

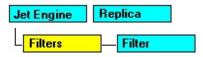
With the properties of a **Filter** object, you can:

• Specify the name of the table to which the filter is applied with the <u>TableName</u> property.

- Use the <u>FilterType</u> property to determine whether the filter is based on a table or a relationship.
- Specify the criteria that a record must satisfy in order to be replicated from the full replica with the <u>FilterCriteria</u> property.

## **JetEngine Object**

Provides control over Jet Engine-specific features such as compacting databases and refreshing data from the cache.



### **Remarks**

Use a **JetEngine** object to access features of the Jet database engine. A JetEngine object functions only when connected to a Jet datasource.

With the methods of a **JetEngine** object, you can:

• Compact a database and modify other database properties with the

## CompactDatabase method.

• Force pending data changes in the cache to be processed, ensuring that a recordset or the database is kept current, with the <a href="RefreshCache">RefreshCache</a> method.

## **Replica Object**

Represents a copy of a replicated database.



#### Remarks

With the collections, methods, and properties of a Replica object, you can do the following:

- Access the filters for a replica with the <u>Filters</u> collection.
- Create a new replica with the <a href="CreateReplica">CreateReplica</a> method.
- Determine whether an object is local or replicated with the

GetObjectReplicability and SetObjectReplicability methods.

- Make a database replicable with the <u>MakeReplicable</u> method.
- Populate a partial replica with the **PopulatePartial** method.
- Synchronize two replicas with the **Synchronize** method.
- Specify the ADO **Connection** object or the valid connection string to which the replica is connected with the <u>ActiveConnection</u> property.
- Specify custom conflict resolution code with the <u>ConflictFunction</u> property.
- Return a **Recordset** containing table conflicts that occurred during synchronization with the <u>ConflictTables</u> property.
- Specify the unique identifier of the design master in a replica set with the <a href="DesignMasterId">DesignMasterId</a> property.
- Determine the relative priority of the replica for use during conflict resolution with the <u>Priority</u> property.
- Specify how long to keep replication histories with the <u>RetentionPeriod</u> property.
- Return the unique identifier of the replica with the <u>Replicald</u> property.
- Determine if the replica is a full, partial, or design master with the ReplicaType property.
- Determine if the replica is a global, local, or anonymous replica with the <u>Visibility</u> property.

## **Jet and Replication Collections**

The Jet and Replication objects contain one collection: <u>Filters</u>. A <u>Replica</u> object can contain a **Filters** collection. The **Filters** collection contains <u>Filter</u> objects, which define criteria that limit the recordset of the replicated database.

The **Filters** collection is a standard ADO collection that uses the following properties and methods to access its members: <u>Count</u>, <u>Append</u>, <u>Delete</u>, <u>Item</u>, and <u>Refresh</u>.

## **Filters Collection**

Contains all of the **Filter** objects for the replica.



### **Remarks**

Replica objects can contain a **Filters** collection.

With the properties and methods of a **Filters** collection, you can:

- Return the number of filters contained in the collection with the <u>Count</u> property.
- Access a column in the collection with the <a>Item</a> method.
- Add a new filter to the collection with the **Append** method.
- Remove a filter from the collection with the **Delete** method.
- Update the objects in the collection to reflect the current database's schema

with the **Refresh** method.

# **Jet and Replication Methods**

## JRO Method Summary

Method	Description
Append	Adds a new <b>Filter</b> object to the <b>Filters</b> collection of a partial <b>Replica</b> .
CompactDatabase	Copies and compacts a closed database, and gives you the option of changing its version, collating order, encryption, and other options.
CreateReplica	Creates a new replica of the current replicable database.
<u>Delete</u>	Removes the <b>Filter</b> object from the <b>Filters</b> collection of a <b>Replica</b> .
<u>GetObjectReplicability</u>	Indicates whether an object is local or replicated.
<u>Item</u>	Returns a specific member of a collection by name or ordinal number.
<u>MakeReplicable</u>	Makes a database replicable.
<u>PopulatePartial</u>	Populates a partial replica.
Refresh	Updates the objects in a collection to reflect objects available from and specific to the provider.
<u>RefreshCache</u>	Forces any pending writes to .mdb files, and refreshes memory with the most current data from the .mdb file.
<u>SetObjectReplicability</u>	Sets whether an object is local or replicated.
<u>Synchronize</u>	Synchronizes two replicas.

## **Append Method**

Adds a new <u>Filter</u> object to the <u>Filters</u> collection of a partial <u>Replica</u>.

## **Syntax**

Filters. Append (Table Name [, Filter Type], Filter Criteria)

#### **Parameters**

*TableName* A **String** value specifying the name of the table to which the filter is applied with the <u>TableName</u> property.

*FilterType* An **Enum** value indicating the <u>FilterType</u> property to determine whether the filter is based on a table or a relationship.

*FilterCriteria* A **String** value specifying the criteria that a record must satisfy in order to be replicated from the full replica with the <u>FilterCriteria</u> property.

#### Remarks

An error will occur if the replica is not a partial replica, as defined by the <a href="ReplicaType">ReplicaType</a> property. An error will occur if a filter with the same name and type

already exists.

An error will occur if you attempt to add a second FilterType **jrFltrTypeTable** with the same *TableName*.

## **CompactDatabase Method**

Copies and compacts a closed database, and gives you the option of changing its version, collating order, encryption, and other options.

### **Syntax**

JetEngine.CompactDatabase(SourceConnection, DestConnection)

#### **Parameters**

*SourceConnection* A **String** value specifying a connection to the source database to be compacted. An error will occur if the database specified by *SourceConnection* is already open.

*DestConnection* A **String** value specifying a connection to the destination database to be created by the compaction. An error will occur if the database specified by *DestConnection* already exists or another file with that name already exists.

#### Remarks

Connection properties are used to specify information for compacting the

database. At a minimum, the Data Source property must be specified for both *SourceConnection* and *DestConnection*. The table below lists the connection properties that you can use with this method. All other connection properties will be ignored.

Property	Source	Destination	Description
Provider	•	•	Indicates the name of the provider to use to connect to the data source. If this property is not specified, the Microsoft.Jet.OLEDB.4.0 provider is assumed.
			An error occurs if the name of the provider in the source string is different from the name of the provider in the destination string.
Data Source	•	•	Indicates the name of the database. This property is required for both the source and destination connection information.
User Id	•		Indicates the name of the user if the source database is secured.
Password	•		Indicates the password for the user if the source database is secured.
			Indicates the locale id for the new database. If this is omitted, the destination database will have the same locale id as the source database.
Locale Identifier		•	Locale specifies the collating order for string comparisons in

the database. For destination database values, see the Remarks section.

Jet
OLEDB:Database 

Password

Indicates the password if the database is secured by a password.

Indicates the version of the source database to open or the version of the new database to be created. The OLE DB engine types that you can use are:

1 for JET Version 1.0

2 for JET Version 1.1

3 for JET Version 2.x

4 for JET Version 3.x

5 for JET Version 4.x

If this is omitted for the destination database, it will be the same version as for the source database. The value for the destination database cannot be a version prior to that of the source database.

Sets information about the Windows Registry key that contains values for the Microsoft Jet Database Engine. Indicates the location of the system database.

Jet OLEDB:Engine , Type

Jet OLEDB:Registry Path

Jet OLEDB:System Database

Jet OLEDB:Encrypt Database	•	Indicates whether to encrypt the new database. If this parameter is omitted, the new database will have the same encryption as the source database.
Jet OLEDB:Don't Copy Locale on Compact	•	Indicates that the database sort order should be used, overriding any per-column locale settings. (Boolean.)
Jet OLEDB:Compact Without Relationships	•	Indicates whether to copy relationships to the new database. (Boolean.)
Jet OLEDB:Compact Without Replica Repair	•	Indicates whether to try and find other replicas to repair damaged replicated databases. (Boolean.)

It is not possible to do an "in place" compact, that is, one in which the source database and destination database are the same, using this method. An error will occur if the provider does not support compacting Jet databases.

### **Remarks**

The following table lists the destination database and Locale Identifier (LCID) values:

Description	LCID
Chinese Punctuation	0x00000804
Chinese Stroke Count	0x00020804
Chinese Stroke Count (Taiwan)	0x00000404
Chinese Bopomofo (Taiwan)	0x00030404
Croatian	0x0000041a
Czech	0x00000405
Estonian	0x00000425
French	0x0000040c

General	0x00000409
Georgian Modern	0x00010437
German Phone Book	0x00010407
Hungarian	0x0000040e
Hungarian Technical	0x0001040e
Icelandic	0x0000040f
Japanese	0x00000411
Japanese Unicode	0x00010411
Korean	0x00000412
Korean Unicode	0x00010412
Latvian	0x00000426
Lithuanian	0x00000427
Macedonian	0x0000042f
Norwegian/Danish	0x00000414
Polish	0x00000415
Romanian	0x00000418
Slovak	0x0000041b
Slovenian	0x00000424
Spanish Traditional	0x0000040a
Spanish Modern	0x00000c0a
Swedish/Finnish	0x0000041d
Thai	0x0000041e
Turkish	0x0000041f
Ukrainian	0x00000422
Vietnamese	0x0000042a

**Note** The General sort order contains the following languages: Africaans, Albanian, Arabic, Basque, Bulgarian, Byelorussian, Catalan, Dutch, English, Faeroese, Farsi, German - Standard, Greek, Hebrew, Hindi, Indonesian, Italian, Malay, Portuguese, Russian, Serbian, Swahili, Urdu.

## **CreateReplica Method**

Creates a new replica of the current replicable database.

## **Syntax**

Replica.CreateReplica(ReplicaName, Description [, ReplicaType] [, Visibility] [, Priority] [, Updatability])

#### **Parameters**

*ReplicaName* A **String** value specifying the name and path of the full or partial replica to be created. An error will occur when specifying a name longer than 255 characters.

*Description* A **String** value describing the replica to be created.

*ReplicaType* Optional. An **Enum** value indicating the type of replica to be created. The default value is **jrRepTypeFull**. See the <u>ReplicaType</u> property for more information. The following constants are valid values for *ReplicaType*:

Constant Description
jrRepTypeFull The replica is a full replica.

### jrRepTypePartial

The replica is a partial replica.

*Visibility* Optional. An **Enum** value indicating the replica's visibility. The default value is **jrRepVisibilityGlobal**. See the <u>Visibility</u> property for more information. The following constants are valid values for *Visibility*:

Constant	Description
jrRepVisibilityGlobal	The replica is global.
jrRepVisibilityLocal	The replica is local.
jrRepVisibilityAnon	The replica is anonymous.

*Priority* Optional. A **Long** value indicating the priority of the replica for use during conflict resolution. The default value is -1, which indicates that the database should determine the default value. For global replicas, the default priority is 90 percent of the parent replica's priority. Also, the valid values for a global replica may be further restricted. If the user is the database administrator, the entire range is valid. Otherwise, the maximum value for priority is 90 percent of the parent replica's priority. For local and anonymous replicas, the value will always be 0 and cannot be changed. This value is forced with the creation of the replica and any other value is ignored. See the <u>Priority</u> property for more information.

*Updatability* Optional. An **Enum** value indicating the type of updates allowed. The default value is **jrRepUpdFull**. The constant **jrRepUpdReadOnly** prevents users from modifying schema and records of replicable objects of the new replica; however, when you synchronize the new replica with another member of the replica set, design and data changes will be propagated to the new replica. The following constants are valid values for *Updatability*:

Constant	Description
jrRepUpdFull	The replica can be updated.
jrRepUpdReadOnly	The replica is read-only.

#### Remarks

An error will occur if the replica was not successfully opened and the <a href="ActiveConnection">ActiveConnection</a> property is not set.

This method can only be used if the database is replicable. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to use this method.

A replica either inherits the exact same characteristics or is more restrictive than the replica that created it. For example, a read-only global replica can only create either a local or anonymous read-only replica, while a local replica can only create another local replica with the same characteristics.

## **Delete Method**

Removes the <u>Filter</u> object from the <u>Filters</u> collection of a <u>Replica</u>.

### **Syntax**

Filters.**Delete**(Index)

### **Parameters**

*Index* A **Variant** value specifying the name or ordinal of the **Filter** object you want to delete.

#### **Remarks**

If two filters have the same name, the first filter will be removed. Use the ordinal value to explicitly delete a filter when more than one filter has the same name.

An error will occur if a filter with the name or ordinal specified does not exist in the collection.

## GetObjectReplicability Method

Indicates whether an object is local or replicated.

## **Syntax**

**Set** ReturnValue = Replica.**GetObjectReplicability**(ObjectName, ObjectType)

### **Return Value**

A **Boolean** value indicating whether the object is replicated. In databases that have not been made replicable, this method returns **True** for all objects by default. **True** indicates that the object will be made replicable if the database is made replicable. In replicable databases, this method returns **False** for all new objects by default. **False** indicates that the object is not replicated.

#### **Parameters**

*ObjectName* A **String** value specifying the name of the object for which to retrieve the replication state.

*ObjectType* A **String** value specifying the type of object specified by *ObjectName*.

### **Remarks**

The **GetObjectReplicability** method indicates whether the object is or will be replicated.

The *ObjectName* and *ObjectType* parameters are strings that indicate the name of the object (for example, Customers) and the object's container (for example, Tables). An error will occur if an object of that name and type does not exist in the database. An error will also occur if either of these strings is longer than 64 characters

See the <u>SetObjectReplicability</u> method for information on how to change an object's replicability.

An error will occur if the object specified by the *ObjectName* and *ObjectType* parameters does not exist.

## **Item Method**

Returns a specific member of a collection by name or ordinal number.

## **Syntax**

Set object = collection.**Item (** *Index* **)** 

### **Return Value**

Returns an object reference.

#### **Parameters**

*Index* A **Variant** that evaluates either to the name or to the ordinal number of an object in a collection.

#### Remarks

Use the **Item** method to return a specific object in a collection. If the method cannot find an object in the collection corresponding to the *Index* argument, an error occurs. Also, some collections don't support named objects; for these collections, you must use ordinal number references.

The **Item** method is the default method for all collections; therefore, the following syntax forms are interchangeable:

collection.**Item (Index)** 

collection (Index)

## MakeReplicable Method

Makes a database replicable.

### **Syntax**

Replica.MakeReplicable([ConnectString] [, ColumnTracking])

#### **Parameters**

*ConnectString* Optional. A **String** value specifying the name and path of the database to make replicable. The *ConnectString* overrides the <u>ActiveConnection</u> property.

*ColumnTracking* Optional. A **Boolean** value that indicates whether to track changes by column or by row. The default value is **True**. Column-level conflict resolution lets you merge two records and only report a conflict if simultaneous changes have been to the same field. If you frequently have overlapping updates in the same row, setting this option could increase performance.

#### Remarks

An error will occur if the *ConnectString* parameter is omitted and the

**ActiveConnection** property has not already been exclusively set. The **ActiveConnection** property will be set if this method is successful.

## **PopulatePartial Method**

Populates a partial replica.

## **Syntax**

Replica. PopulatePartial (Full Replica)

#### **Parameters**

*FullReplica* A **String** value representing the path and file name of the replica to populate with data.

#### Remarks

When you synchronize a partial replica with a full replica, it is possible to create "orphaned" records in the partial replica. For example, suppose you have a Filter for the Customers table with a FilterCriteria of "Region = 'CA'". If a user changes a customer's region from CA to NY in the partial replica, and then a synchronization occurs with the Synchronize method, the change is propagated to the full replica. But the record containing NY in the partial replica is orphaned because it now doesn't meet the replica filter criteria.

To solve the problem of orphaned records, you can use the **PopulatePartial** method. The **PopulatePartial** method is similar to the **Synchronize** method, but it synchronizes any changes in the partial replica with the full replica, removes all records in the partial replica, and then repopulates the partial replica based on the current replica filters. Even if your replica filters have not changed, **PopulatePartial** will always clear all records in the partial replica and repopulate it based on the current filters.

Generally, you should use the **PopulatePartial** method when you create a partial replica and whenever you change your replica filters. If your application changes replica filters, you should follow these steps:

- 1. Synchronize the full replica with the partial replica in which the filters are being changed.
- 2. Use the **Filter** object to make the desired changes to the replica filter.
- 3. Call the **PopulatePartial** method to remove all records from the partial replica and transfer all records from the full replica that meet the new replica filter criteria.

If a replica filter has changed, and the **Synchronize** method is called without first calling **PopulatePartial**, a trappable error occurs.

The **PopulatePartial** method can only be invoked on a partial replica that has been opened for exclusive access. Furthermore, you can not call the **PopulatePartial** method from code running within the partial replica itself. Instead, open the partial replica exclusively from the full replica or another database, then call **PopulatePartial**.

**Note** Although **PopulatePartial** performs a one-way synchronization before clearing and repopulating the partial replica, it is still a good idea to call **Synchronize** before calling **PopulatePartial**. When using the direct or Internet synchronization modes, if the call to **Synchronize** fails, a trappable error occurs. You can use this error to decide whether or not to proceed with the **PopulatePartial** method (which removes all records in the partial replica). For indirect synchronization, a trappable error does not occur; see the **Synchronize** method for more information. If **PopulatePartial** is called by itself and an error occurs while records are being synchronized, records in the partial replica will

still be cleared, which may not be the desired result.

This method can only be used if the database is replicable and it uses a **PartialReplica**. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to use this method.

## **Refresh Method**

Updates the objects in a collection to reflect objects available from and specific to the provider.

**Syntax** 

collection.Refresh

### RefreshCache Method

Forces any pending writes to .mdb files, and refreshes memory with the most current data from the .mdb file.

### **Syntax**

JetEngine.RefreshCache(Connection)

### **Parameters**

Connection The ADO **Connection** object for which to refresh the cache. An error will occur if the connection is not a valid, open ADO **Connection**. An error will occur if the provider used to create the connection does not support the **RefreshCache** method.

#### Remarks

You don't need to use this method in single-user environments unless multiple connections are made to the database. The **RefreshCache** method may increase performance in a multiuser environment because it forces the database engine to write data to disk, releasing locks on memory.

# SetObjectReplicability Method

Sets whether an object is local or replicated.

### **Syntax**

Replica. SetObjectReplicability (Object Name, Object Type, Replicability)

### **Parameters**

*ObjectName* A **String** value specifying the name of the object for which to retrieve the replication state.

*ObjectType* A **String** value specifying the type of object specified by *ObjectName*.

*Replicability* A **Boolean** value specifying whether the object is or will be replicated.

#### Remarks

The **SetObjectReplicability** method makes an object local or replicated. If the database has not been made replicable, setting the *Replicability* parameter to

**False** will indicate that the object should be kept local when the database is made replicable. Objects in non-replicable databases are replicable by default. However, new objects created in a replicable database are not replicable by default. To make a new object in a replicable database replicable, set the *Replicability* parameter to **True**.

The *ObjectName* and *ObjectType* parameters are strings that indicate the name of the object (for example, Customers) and the object's container (for example, Tables). An error will occur if an object of that name and type does not exist in the database. An error will also occur if either of these strings is longer than 64 characters.

**SetObjectReplicability** is ignored on objects in the following Microsoft Access collections: Forms, Reports, DataAccessPages, Macros, and Modules. An Access system table, **MSysAccessObjects**, controls the replicability of these objects and can only be set prior to making the database replicable. The default is **True**.

See the <u>GetObjectReplicability</u> method for information about how to determine an object's replicability.

# **Synchronize Method**

Synchronizes two replicas.

### **Syntax**

Replica.Synchronize(Target [, SyncType] [, SyncMode])

Constant

#### **Parameters**

*Target* A **String** value specifying the path and file name of the replica with which to synchronize, the name of the Synchronizer that manages the target replica, or the name of the Internet server that contains the target replica.

*SyncType* Optional. An **Enum** value specifying the type of synchronization to perform. The default value for the *SyncType* parameter is **jrSyncTypeImpExp**. The following values are valid for *SyncType*:

Constant	Description
jrSyncTypeExport	Sends changes from the current replica to the target replica.
	3 1
	Sends changes from the target repl

Description

jrSyncTypeImport to the current replica.

Default. Sends changes from the jrSyncTypeImpExp current replica to the target replica and

vice-versa.

*SyncMode* Optional. An **Enum** value specifying the method of synchronization. **jrSyncModeIndirect** is the default value for the *SyncMode* parameter. The following values are valid for *SyncMode*:

Constant	Description
jrSyncModeIndirect	Default. Indirect synchronization.
jrSyncModeDirect	Direct synchronization.
irSyncModoIntornot	Indirect synchronization over the
jrSyncModeInternet	Internet.

#### Remarks

The replica identified in *Target* parameter must be part of the same replica set. If both replicas have the same Replicald property setting or are design masters for two different replica sets, the synchronization fails. This is enforced by the provider.

When the *SyncMode* is indirect, the value of the *Target* parameter must be a Synchronizer name. Jet Replication leaves the changes in a "dropbox." The Synchronizer that manages that target replica picks up the changes and applies them. For indirect synchronization to work correctly, a Synchronizer must be running on both the local computer and the target computer.

When the *SyncMode* is Internet, the value of the *Target* parameter must be a Universal Resource Locator (URL) address instead of a local area network path. An error will occur if a URL is specified in the *Target* parameter and the *SyncMode* parameter is not **jrSyncModeInternet**.

When the *SyncMode* is direct, both replicas are opened simultaneously and synchronized. Over a WAN or remote dialup network, reliability and performance are improved by using indirect synchronization. You can also synchronize with an SQL Server replica in a replica set containing both SQL Server and Jet databases by setting the *Target* parameter to **ServerName.Database.Publication** and performing a direct (**jrSyncModeDirect**) synchronization. An error will occur if the *Target* parameter is **ServerName.Database.Publication** and *SyncMode* is other than direct.

This method can only be used if the database is replicable. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to use this method.

Replication Manager is required for installation and configuration of the Synchronizer and Replman should be used to monitor the status for indirect and Internet synchronizations. It is only available in the Microsoft Office 2000, Developer Edition. For more information about Replication Manager, see Replication Manager in Microsoft Access help.

## **Jet and Replication Properties**

### **JRO Property Summary**

Property	Description

Indicates the **Connection** object to ActiveConnection

which the **Replica** belongs.

Indicates the name of the custom conflict resolution function to use in ConflictFunction

resolving conflicts during

synchronization.

Specifies a **Recordset** containing a list ConflictTables

of tables and associated conflict tables.

Indicates the number of objects in a Count

collection.

Uniquely identifies the design master in **DesignMasterId** 

a replica set.

Specifies the criteria that a record must **FilterCriteria** 

satisfy in order to be replicated from

the full replica.

**FilterType** Indicates the type of filter.

Specifies the relative priority of the

replica for use during conflict **Priority** 

resolution.

**ReplicaId** Uniquely identifies a database replica.

**ReplicaType** Indicates the type of replica.

Indicates how many days to keep **RetentionPeriod** 

<u>TableName</u>

**Visibility** 

replication histories.

Indicates the name of the table to which

the filter is applied.

Indicates the visibility of the replica:

Global, Local, or Anonymous.

# **ActiveConnection Property**

Indicates the **Connection** object to which the <u>Replica</u> belongs.

### **Settings and Return Values**

Sets or returns either a **String** containing the definition for a connection or an opened **Connection** object.

### **Remarks**

The default value is **Null**.

If the **ActiveConnection** property is set to nothing, then any related objects will be disconnected from the data source.

# **ConflictFunction Property**

Indicates the name of the custom conflict resolution function to use in resolving conflicts during synchronization.

### **Settings and Return Values**

Sets or returns a **String** value that is the name of a custom function. The **String** must be less than 256 characters long. The default value is an empty string ("").

#### Remarks

The **ConflictFunction** property enables you to use custom conflict resolution code for resolving data conflicts that occur between replicas upon synchronization, or it can designate one replica in the set to view the built-in Conflict Resolver. Set the **ConflictFunction** property to a text string that is the name of the function you wish to call. Note that the setting must be the name of a **Function** procedure; it can't be the name of a **Sub** procedure. If this property has not been set, Microsoft Access calls the built-in Conflict Resolver.

A conflict occurs if the same record in a replicated database has been changed in one or more replicas (or the same column in the same record if column level tracking is enabled for the database). Normally, Microsoft Access calls the builtin Conflict Resolver to resolve these conflicts. The Conflict Resolver wizard presents each conflict to the user, who must manually determine which changed record contains the correct data. You can designate one replica in the replication set to view conflicts with the built-in Conflict Resolver by setting the **ConflictFunction** property to be equal to the ReplicaId (GUID) of that replica.

If you prefer to automate conflict resolution for your application, you can write custom procedures to resolve conflicts, and then override the built-in functionality by setting the **ConflictFunction** property. You should create a single function that acts as a point of entry into your conflict resolution code, then set the **ConflictFunction** property to the name of this function.

This property can only be read or set if the database is replicable. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to read or set this property. An error will also occur if the ReplicaType is **jrRepTypeDesignMaster**.

# **ConflictTables Property**

Specifies a **Recordset** containing a list of tables and associated conflict tables.

#### **Return Values**

Returns a recordset containing two columns: TableName and ConflictTableName for each table that had conflicts during the synchronization of two replicas. **ConflictTables** is read-only.

### **Remarks**

This property can only be read if the database is replicable. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to read or set this property.

### **Count Property**

Indicates the number of objects in a collection.

#### **Return Value**

Returns a **Long** value.

### **Remarks**

Use the **Count** property to determine how many objects are in a given collection.

Because numbering for members of a collection begins with zero, you should always code loops starting with the zero member and ending with the value of the **Count** property minus 1. If you are using Microsoft<sup>®</sup> Visual Basic<sup>®</sup> and want to loop through the members of a collection without checking the **Count** property, use the **For Each...Next** command.

If the **Count** property is zero, there are no objects in the collection.

## **DesignMasterId Property**

Uniquely identifies the design master in a replica set.

### **Settings and Return Values**

Sets or returns a **Variant** value that uniquely identifies the design master in a replica set (a GUID). This property is automatically set when the design master is created. The default value is an empty string ("").

#### Remarks

A design master can set one of its replicas to be the new design master. A replica can make itself the new design master, but it cannot make a different replica the design master.

Under extreme circumstances — for example, if the design master is erased or corrupted — you can set this property at the current replica. However, setting this property at a replica when there is already another design master in the set might partition your replica set into two irreconcilable sets, and prevent any further synchronization of data.

This property can only be read if the database is replicable. An error will occur if

the **ReplicaType** is **jrRepTypeNotReplicable** and the user attempts to read or set this property.

# FilterCriteria Property

Specifies the criteria that a record must satisfy in order to be replicated from the full replica.

### **Settings and Return Values**

Sets or returns a **String** value. For filters based on a table, the string should represent a SQL WHERE clause without the keyword WHERE. For filters based on relationships, the string contains the name of the relationship. **FilterCriteria** is read-only once it has been set, and can only be set using the **Append** method.

#### Remarks

The default value is an empty string ("").

# FilterType Property

Indicates the type of filter.

### **Settings and Return Values**

Sets or returns an **Enum** value. The following constants are valid values for **FilterType**. **FilterType** is read-only once it has been set, and can only be set using the **Append** method.

Constant Description

JrFltrTypeTable Default. The filter is based on a table.

JrFltrTypeRelationship The filter is based on a relationship.

# **Priority Property**

Specifies the relative priority of the replica for use during conflict resolution.

#### **Return Values**

Returns a **Long** value from 0 to 100 that indicates the priority. **Priority** is read-only.

### **Remarks**

The algorithm for resolving conflicts is as follows:

- The replica with the highest priority wins.
- If priorities are equal, the replica with the lowest <u>Replicald</u> wins.

In previous versions, the conflict resolution algorithm was based on the replica with the most changes in the row and the second rule listed above. To ensure compatibility when older databases are converted, the priority for all replicas are equal to 90.

For global replicas, the default priority is 90 percent of the parent replica's

priority. Also, the valid values for a global replica may be further restricted. If the user is the database administrator, the entire range is valid. Otherwise, the maximum value for priority is 90 percent of the parent replica's priority. For local and anonymous replicas, the value will always be 0 and cannot be changed. This value is forced with the creation of the replica and any other value is ignored.

This property can only be read if the database is replicable. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to read this property.

# **Replicald Property**

Uniquely identifies a database replica.

#### **Return Values**

Returns a **Variant** value that uniquely identifies a database replica (a GUID). This property is automatically generated when the replica is created. **Replicald** is read-only.

This property can only be read if the database is replicable. An error will occur if the <u>ReplicaType</u> is **jrRepTypeNotReplicable** and the user attempts to read or set this property.

## ReplicaType Property

Indicates the type of replica.

### **Return Values**

Returns an **Enum** value that can be one of the following constants:

Constant	Description
jrRepTypeNotReplicable	Default. The database is not replicable.
jrRepTypeDesignMaster	The replica is a design master.
jrRepTypeFull	The replica is a full replica.
jrRepTypePartial	The replica is a partial replica.

**ReplicaType** is read-only.

### **Remarks**

The **ReplicaType** property is used to determine whether a database is replicable and if so, what type of replica it is. If **ReplicaType** is **jrRepTypeNotReplicable** then the database has not yet been made replicable. Use the *ReplicaType* parameter of the <u>CreateReplica</u> method to create a new full or partial replica.

## **RetentionPeriod Property**

Indicates how many days to keep replication histories.

### **Settings and Return Values**

Sets or returns an **Integer** value ranging from 5 to 32,000, which represents a number of days.

### **Remarks**

The **RetentionPeriod** property specifies the amount of time, measured in days, that a replica set retains details of deleted records, schema changes, and other system-specific information. If the database was made replicable with ADO, RDO, or the Replication Manger, the default value is 60.

If the database was made replicable with Access, the default value is 1000.

This property can only be set on a design master. An error will occur if the ReplicaType is not **jrTypeDesignMaster**.

# **TableName Property**

Indicates the name of the table to which the filter is applied.

### **Settings and Return Values**

Sets or returns a **String** value specifying a table name. For filters based on a relationship, this is the table on the many side of the relationship. **TableName** is read-only once it has been set, and can only be set using the **Append** method.

### **Remarks**

The default value is an empty string ("").

# Visibility Property

Indicates the visibility of the replica: Global, Local, or Anonymous.

#### **Return Values**

Returns an **Enum** value that indicates the visibility of the replica within the replica set. It can be one of the following constants:

Constant	Description
jrRepVisibilityGlobal	Default. The replica is global.
jrRepVisibilityLocal	The replica is local.
jrRepVisibilityAnon	The replica is anonymous.

**Visibility** is read-only.

### **Remarks**

A global replica is the typical replica from which you can create all other types of replicas. Changes by a global replica are fully tracked and can be exchanged with any other global replica in the set. The global replica can also exchange changes with any local or anonymous replicas for which it becomes the hub.

**Note** The Design Master is a global replica.

Local and anonymous replicas synchronize only with their hub, a global replica. They are not permitted to synchronize with other replicas in the replica set. All local and anonymous replicas always have a priority of 0, therefore, if any of their changes conflict with the global hub replica, the changes will automatically lose in any conflict resolution process. If they convey a non-conflicting change to the hub, the hub assumes authorship of the change.

Only the hub replica is aware of local replicas, and only it can schedule an exchange to a local replica.

No replicas (including the hub replica) are aware of anonymous replicas. The hub replica cannot schedule an exchange to an anonymous replica. However, anonymous replicas are recommended for use on the Internet for mass distribution because system-tracking information is not maintained, and replica size is reduced.

Visibility cannot be changed once the replica is created using the <u>CreateReplica</u> method.

This property can only be read if the database is replicable. An error will occur if the <u>ReplicaType</u> property is not **jrRepTypeNotReplicable** and the user attempts to read this property.

# **Jet and Replication Examples**

Use these topics to learn how to use the following JRO methods.

**Note** Paste the entire code example, from Sub to End Sub, into your code editor. The example may not run correctly if you use partial examples or if paragraph formatting is lost.

The <u>Append method example</u> demonstrates how to append a new filter to a table.

The <u>CompactDatabase method example</u> demonstrates how to compact and encrypt a database with the **CompactDatabase** method.

The <u>CreateReplica method example</u> demonstrates how to create a full replica with the **CreateReplica** method.

The <u>MakeReplicable method example</u> demonstrates how to create a Design Master with the **MakeReplicable** method.

The <u>PopulatePartial</u> method example demonstrates how to update a replica using the **PopulatePartial** method.

The RefreshCache method example demonstrates the RefreshCache method.

The <u>Synchronize method examples</u> demonstrate how to update changes between a Design Master and a replica with the **Synchronize** method; use the **Synchronize** method for Internet synchronization; and use the **Synchronize** method with direct synchronization.

## **Append Method Example**

This example demonstrates how to append a new filter to a table.

## PopulatePartial Method Example

This example demonstrates how to update a replica using the **PopulatePartial** method.

```
Public Sub PartialRep()
' This code demonstrates how to create a partial replica with a
' relationship filter and a table filter.
' NOTE: PopulatePartial requires an exclusive connection.
    Dim repMaster As New JRO.Replica
    Dim repPartial As New JRO.Replica
    Dim flt As JRO.Filter
    repMaster.ActiveConnection =
        "C:\Program Files\Microsoft Office\Office\Samples\Northwind.
    If (Dir("C:\Program Files\" & _
        "Microsoft Office\Office\Samples\" & _
        "Partial of Northwind.mdb") <> "") Then Kill _
        ("C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Partial of Northwind.mdb")
    ' Northwind.mdb is already replicable so you can
    ' create replicas from it.
    repMaster.CreateReplica "C:\Program Files\Microsoft Office\" &
        "Office\Samples\Partial of Northwind.mdb",
        "Partial Replica of Northwind", jrRepTypePartial
    Set repMaster = Nothing
    ' PopulatePartial requires an exclusive connection to the databa
    repPartial.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;
        "Data Source=C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Partial of Northwind.mdb;" & _
        "Mode=Share Exclusive"
    repPartial.Filters.Append "Orders", jrFilterTypeRelationship, _
        "CustomersOrders"
    repPartial.Filters.Append "Customers", jrFilterTypeTable, _
        "CustomerID"
    repPartial.PopulatePartial "C:\Program Files\" & _
```

"Microsoft Office\Office\Samples\Northwind.mdb"

## MakeReplicable Method Example

This example demonstrates how to create a Design Master with the **MakeReplicable** method.

Public Sub MakeRep()

- ' This code example demonstrates how to use
- ' MakeReplicable by specifying the ConnectionString parameter.
- ' NOTE: This connection string does not need to explictly specify
- ' exclusive connection mode, JRO does this automatically.

Dim repDM As New JRO.Replica
' Make northwind.mdb replicable with record level tracking
repDM.MakeReplicable "C:\Program Files\Microsoft Office\" & \_
 "Office\Samples\Northwind.mdb", False

# CreateReplica Method Example

This example demonstrates how to create a full replica with the **CreateReplica** method.

## **CompactDatabase Method Example**

This example demonstrates how to compact and encrypt a database with the **CompactDatabase** method.

```
Public Sub CompactAndEncrypt()
    Dim je As New JRO. Jet Engine
    ' Make sure that a file doesn't exist with the name of
    ' the compacted database.
    If Dir("C:\Program Files\Microsoft Office\" &
        "Office\Samples\Northwind2.mdb") <> "" Then Kill _
        "C:\Program Files\Microsoft Office\Office\Samples\Northwind2
    ' Compacts and encrypts version Northwind database.
    je.CompactDatabase _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=C:\Program Files\Microsoft Office\" &
        "Office\Samples\Northwind.mdb", _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Northwind2.mdb;" &
    "Jet OLEDB:Encrypt Database=True"
```

# RefreshCache Method Example

This example demonstrates the **RefreshCache** method.

```
Public Sub RefCache()
    Dim lateje As JRO. Jet Engine
    Dim conn As New ADODB.Connection
    Dim conn2 As New ADODB.Connection
    Dim rs As ADODB.Recordset
    Dim fld As ADODB.Field
    ' Open both connections to the database.
    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data source=C:\Program Files\Microsoft Office\" &
        "Office\Samples\Northwind.mdb;"
    conn2.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data source=C:\Program Files\Microsoft Office\" &
        "Office\Samples\Northwind.mdb;"
    Set lateje = CreateObject("JRO.JetEngine")
On Error Resume Next
    conn.Execute "drop table tab1"
On Error GoTo 0
    ' Create table and input new values for columns.
    conn.Execute "create table tab1 ( col1 int)"
    conn.Execute "insert into tab1 values (1)"
    conn.Execute "insert into tab1 values (2)"
    ' Flush the data so that conn2 can see the changes from conn1.
    lateje.RefreshCache conn2
    Set rs = conn2.Execute("select * from tab1")
    Set fld = rs.Fields(0)
    Debug.Print "Record 1: " & fld.Name & " = " & Str(fld.Value)
    conn.Close
    conn2.Close
    Set lateje = Nothing
End Sub
```

## **Synchronize Method Example**

The following three examples demonstrate the **Synchronize** method. This example demonstrates how to update changes between a Design Master and a replica with the **Synchronize** method.

```
Public Sub DirectSync()
    Dim repMaster As New JRO.Replica
    Dim RepNorthwind As New JRO.Replica
    Dim conn As New ADODB. Connection
    ' Open the database.
    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data source=C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Northwind.mdb;"
    ' If a replica of the database exists, it is deleted.
    repMaster.ActiveConnection = conn
    ' "C:\Program Files\Microsoft Office\Office\Samples\Northwind.md
    If (Dir("C:\Program Files\Microsoft Office\" &
        "Office\Samples\Replica of Northwind.mdb") <> "") Then Kill
        ("C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Replica of Northwind.mdb")
    ' Create a new replica of the database.
    repMaster.CreateReplica "C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Replica of Northwind.mdb",
        "Replica1 for Northwind.mdb", _
        jrRepTypeFull, jrRepVisibilityGlobal
    ' New values are put into tab1.
On Error Resume Next
    conn.Execute "drop table tab1"
On Error GoTo 0
    conn.Execute "create table tab1 ( col1 int)"
    conn.Execute "insert into tab1 values (1)"
    conn.Execute "insert into tab1 values (2)"
    ' Synchronize the values.
    repMaster.Synchronize "C:\Program Files\Microsoft Office\" &
        "Office\Samples\Replica of Northwind.mdb", jrSyncTypeImpExp,
        jrSyncModeDirect
```

This example demonstrates how to use the **Synchronize** method for Internet synchronization. Note that prior to synchronization, the Internet replication setup must be completed.

This example demonstrates how to use the **Synchronize** method with indirect synchronization.