



# **IBP Replicator 5.0**

© 2016 IBPhoenix Editors

**User Manual**

Build 5.0.04  
28/05/2016

# IBPhoenix *Replicator* 5.0

for Firebird, Avalerion,  
InterBase and Oracle databases



Build 5.004 28/05/2016 © 2016 IBPhoenix Editors

IBPhoenix Replicator 5 is a toolset for implementing and controlling database replication on all Firebird server versions, InterBase versions 5.x and higher and Oracle 9 and higher. From this release forward, Replicator can be plugged into the Avalerion build of Firebird 3-series servers. It can also replicate from a Firebird, InterBase or Oracle source to an ODBC datasource target that can be any DBMS with a proper ODBC driver.

Replicator is software that integrates with your databases, allowing users to replicate data seamlessly and transparently between databases on both local and remote sites.

This manual provides an overview of IBP Replicator and how IBP Replicator and its tools can be used to meet your replication needs. It provides a step by step breakdown on how to install, configure and use the Replicator software. Included are descriptions of the various replication models you can implement and how to use the Replication Manager tool to set up your replication schemas and to operate and schedule replications.

Chapter 1 describes the platform, database engine, system and licensing requirements.

Chapter 2 is an overview of the IBP Replicator toolset.

Chapter 3 describes the architecture and terminology of IBP Replicator and Chapter 4 goes on to explain how the replication actually works

In Chapter 5, you will find a discussion of the planning you will need to consider in designing your custom Replicator system.

Chapter 6 contains instructions for installing the pieces of your replication system on source and target servers.

In Chapter 7, you will find the details for defining a replication schema--the rules.

Chapter 8 explains how to run and manage your replication server.

In Chapter 9 you will find a miscellany of topics concerning issues that will arise in more complex setups and how to deal with them.

The Version Notes at the end of this volume contain details of enhancements in Version 5.0.

Installation and configuration of the Avalerion server are not discussed in this manual. Please consult the documentation accompanying the Avalerion server software.

# Table of Contents

<b>Chapter 1 - IBP Replicator Requirements</b>	<b>6</b>
<b>Chapter 2 - Overview</b>	<b>9</b>
<b>Chapter 3 - IBP Replicator Architecture</b>	<b>12</b>
Terminology .....	13
<b>Chapter 4 - How IBP Replicator Works</b>	<b>16</b>
<b>Chapter 5 - Planning for Replication</b>	<b>19</b>
Choosing a Replication Scheme .....	19
Database Design Issues .....	23
Physical Distribution Considerations .....	25
Conflict Resolution .....	26
<b>Chapter 6 - Installation and Setup</b>	<b>28</b>
First Configuration Database .....	29
Registering Databases .....	32
About Licences .....	42
<b>Chapter 7 - Defining a Replication Schema</b>	<b>46</b>
Creating the Schema .....	47
Choosing the Source Database .....	51
Choosing the Target Database .....	57
Choosing Replicated Tables .....	61
Table Settings .....	71
One-time Synchronization .....	76
Creating Source System Objects .....	80
Customising the Global Settings .....	82
<b>Chapter 8 - Managing Replication</b>	<b>87</b>
Windows Operation .....	87
Command-Line Operation .....	91
Linux Operation .....	93
Installing and Configuring on Linux .....	94
Controlling replserver Processes .....	95
Loading Licences Manually .....	99
Management Tools .....	100
Replication Monitor .....	102
Replication Scheduler .....	104

Notify Server .....	107
Conflict Resolution .....	108
License Manager .....	110
Schema View .....	113
Reporting Errors .....	115

## Chapter 9 - Advanced Topics 118

Schema Numbering .....	118
Metadata Changes .....	121
Adding Timefields .....	121
Advanced Mapping Techniques .....	123
Complex Schemas .....	125
Off-line Replication and Synchronization .....	129
Configuring the Offline Schema .....	131
Running an Offline Replication .....	133
Inbound Offline Replication .....	134
Off-line Synchronization .....	136

## Appendices 142

Appendix I: Supported Data Types and Compatibilites .....	142
Appendix II: Command-Line Editor: cfged .....	146
Appendix III: Version Notes .....	172

## Index 174



# Chapter 1

## Chapter 1 - IBP Replicator Requirements

## Chapter 1 - IBP Replicator Requirements

### Operating System

- IBPhoenix Replicator Server runs on Windows and Linux platforms, with both 32-bit and 64-bit versions available.
- The current version of the Replication Manager application needs to run on a Windows host, which may be a Wine environment on a Linux server. It can access a remote Replicator server on Linux.

### Database Engine

Replicator supports replication to and from databases running under

- Firebird (any released version, including the special Avalerion build with plug-in capability)
- InterBase 5.x and later versions
- Oracle 9 and later versions

Replication from any of these sources can be targeted to an ODBC system DSN connecting to other database management systems, if the ODBC driver support is reasonably well implemented.

### Names of Client Libraries

The client libraries must be named according to their "native" conventions. On Windows, this means fbclient.dll for Firebird, gds32.dll for InterBase and oci.dll for Oracle; on Linux it means libfbclient.so for Firebird, libgds.so for InterBase and libclntsh.so for Oracle versions that can run on Linux.

### Disk Space

To install the IBPhoenix Replicator software you need at least 5.25MB of available disk space.

### Database Installation

Both the Replication Server and the Manager require at least the appropriate client library for the database to be installed. Installation and configuration will involve creating a configuration database, so these activities must take place on a running instance of the database server.

### Licensing

There are two types of licence: Server and Replicant. Licences must be installed into the configuration database that you want used for your active replication.

- a Server licence allows the Replicator application to run

- a Replicant licence allows a database to be accessed as a source or target
- free evaluation licensing is included in the software, supplying a minimal, single Server licence and four Replicant licences. They are valid for 14 days from the date the IBPhoenix Replicator software is installed.

### Existing Licences

Existing IBPhoenix Replicator V.3 licences are valid for V.4 and will be preserved. Licences from versions prior to V.3 are not valid for Replicator 4 and will be removed from the configuration altogether.

There are more notes about licences in [Chapter 6](#), About Licences. Instructions for installing them can be found in the topic about the [License Manager](#) tool.

[Chapter 5](#) describes some possible configuration models, which may help to clarify the licensing requirements.



# Chapter 2

## Chapter 2 - Overview



## Chapter 2 - Overview

IBPhoenix Replicator is a management system that is added to servers hosting Firebird, Interbase or Oracle databases. It keeps track of data changes in databases designated as sources and enables users to synchronise two or more databases at one or more sites, locally or remotely.

Replication makes it possible to maintain identical copies of databases, database tables or even pre-selected sets between source databases and target databases. The Replication Server's typical activity is to work behind the scenes, replicating data at set intervals, or according to a schedule, or in response to specific events.

Replicator is designed as a software layer, transparent to ordinary users, that sits between databases and applications.

Replicator can also replicate to or from a file, to cater for conditions where two-tier networks connections are too slow or unreliable to sustain "live" replication.

An important component of the Replication Server suite is a graphical, Windows-based Replication Manager program for defining, managing and monitoring the operations and for resolving replication issues.

Replicator connects to database servers directly through the client APIs. No middleware or third party driver is used, overhead is low and the system is small, robust and fast.

### Support for Database Features

Replicator's replication can handle

- any data type supported by Firebird or InterBase, including BLOBs and arrays
- columns defined using quote-delimited reserved words as identifiers
- other SQL quote-delimited identifiers
- multi-segment primary keys
- international character sets

Replication can be configured to respond to database events.

Replication to or from databases other than Firebird or InterBase, notably Oracle 9 and higher and to ODBC targets, is supported.

### Replicator Security

Security provisions use the same mechanisms as those used to secure your normal databases. User names and passwords that Replicator uses to connect to source and target databases are stored in the regular server security structures of the respective server installations.

### Benefits of Replicator

Replication can enhance the usability of your system in a number of ways. For example,

**Data availability** – Work groups can have their own copy of a replicated database, enabling them to function independently within a larger network. A local copy of a database means that the work group does not have to compete for larger network resources or suffer response delays by accessing databases on remote sites. Decision support applications can be separated from high-volume transactional applications if necessary, which can reduce network traffic and improve data availability and system performance.

**Reliability, load balancing and failover** – Replicated databases can be maintained for load balancing and/or improved system fault tolerance. Applications can switch from original source databases to replicated copies in the event of system overload or outright failure. Partially-replicated databases can be dedicated to heavy read-only operations, freeing up resources for interactive users.

**Flexibility of data distribution** – Any organisation has its own replication requirements and could be operating over a variety of databases. With IBP Replicator you can replicate full databases, a subset of tables, selected rows and columns from tables in a source database to a number of target databases. You can specify when you want automatic replication to occur; on a time-based schedule (hourly, daily, weekly), or when a specific event occurs; or you can specify to initiate replication manually. You can define multiple replication schemas, if you need to, that will enable different source database subsets to be replicated to specific multiple targets.



# Chapter 3

## Chapter 3 - IBP Replicator Architecture

## Chapter 3 - IBP Replicator Architecture

This chapter introduces the components of IBP Replicator and explains the role of each in the replication system.

### Replicator Server

The IBPhoenix Replicator Server (ReplServer) is the program that performs replication operations. To determine what schemas need to be replicated where, it uses a configuration database. A single server can replicate multiple source databases cyclically. On Windows server platforms it can be run as a service (recommended) or as an application.

### Replication Manager

The Replication Manager is a Windows graphical tool for the user to define, manage and monitor replication schemas.

### Configuration Database

IBP Replicator's configuration database is where the program saves and maintains the details of what is to be replicated and where it should be replicated from (the source) and to (the target). The specifications for a replication operation are referred to as a replication schema.

You can have Replicator create multiple configuration databases to handle multiple schemas, or you can store multiple schemas in one single configuration database.

### Replication Processing Tables

**IMPORTANT:** For IBPhoenix Replicator to work properly, each base table that is to be involved in replication must be uniquely identifiable. Ideally, each table involved should have its own unique or primary key; However, unique identification in the absence of keys is sufficient.  
Refer to the Appendix [Supported Data Types and their Compatibilities](#) for information about matching columns with non-identical data types.

The Create System Objects function generates potentially three kinds of triggers--insert, update and/or delete triggers--to track changes that are made to the database tables. Changes are recorded in the Log Table.

The Log Table stores the key information on all the changes that have been made to the tables in the source database.

The Manual Conflict Resolution Log contains a list of operations from the Log Table that were not successfully replicated because of a data conflict problem and now require manual user intervention to correct the problem.

## 3.1 Terminology

### Source (database)

When we talk about the source database, we refer to the database containing the "master version" of data that is intended to be copied to one or more other databases.

### Target (database)

A target database is one that receives changes to the contents of its tables (inserts, updates and deletions) from a source database, generally on a regular basis.

### Replication

Replication is the process of passing newer versions of table data from a source to a target database, thus "replaying" changes that have occurred in the source tables. Replication of subsets (parts of records, filtered sets) is also possible. This process is sometimes referred to as "log shipping".

### Synchronization

Synchronization is an occasional (usually once-only) method of replication, where the entire contents of tables from a source database are pumped into tables in the target. Synchronization is useful for adding data to a new target database from a source database to ensure that the two databases are in synch. If a target database has simply been offline, there is no need to perform synchronization, as normal replication will catch up with any backlog of changes.

If the target tables are "empty" before synchronization, the process is referred to as symmetric synchronization.

### Replicant

A replicant is a database that participates in replication.

### Registering a database

is the action of configuring an IBP Replicator installation to find and recognise a database.

### Asynchronous Replication

Replication is described as asynchronous where the architecture of the database engine and/or the method of replication precludes a situation wherein the target database is kept in synch with the source data in real time. IBP Replicator works with transactional database engines and is event or schedule driven. Thus, even though the process of replication can be effectively continual, the replication performed is always asynchronous.

### Synchronous Replication

In situations where databases must always be kept up-to-date, synchronous replication would replicate changes as and when they occur, i.e., before changes in the source

database are committed. IBP Replicator does not support synchronous replication.

Event-driven replication can be employed, whereby critical data is replicated "in near-to-real time" in response to an event. The cost is a heavy overhead in resource usage. Generally, a compromise would be reached, to replicate the critical data in response to an event while continuously replicating less critical data at suitable intervals according to a schedule.

### Data Subset(ting)

When only certain rows in a table are replicated to a target, it is called data subsetting. It is entirely possible that replication from a single source table could replicate different subsets to multiple targets.



# Chapter 4

## **Chapter 4 - How IBP Replicator Works**

## Chapter 4 - How IBP Replicator Works

The purpose of IBPhoenix Replicator is to create and maintain identical data in database tables distributed across multiple local and remote sites. This chapter provides a technical overview of how the various replication components work together to allow replication to take place.

### Replication Schema Definition

A replication schema defines

- a source and target pair of databases
- the objects in the source database that are to provide data for replication
- the objects in the target database that are to receive data from each particular source object
- the rules for running the replications

As a replication schema is defined, the Replication Manager stores its details in a configuration database. Once a schema's outline is defined, you tell the Replication Manager to create system objects in the specified source database[s] and the schema is completed.

### System Objects

System Objects can be regarded as the metadata of a replication schema. They comprise triggers and tables that ensure that, for each changed row in the source database, the table identifier, primary key and the action (insert, update, delete) are logged.

The logic of the triggers handles new, updated or deleted data created on the local (source) database and, if applicable, incoming data from other databases that are replicating back to the local database. The log table forms a queue of committed work waiting to be replicated.

As an insert, update or delete takes place on a specified table or set, a trigger fires to write information to a log table when the transaction is committed. It is this underlying mechanism that prevents information from being replicated before it is committed in the source database: no commit, no log entry, no replication.

### Executing Replication

The replication server can receive the instruction to replicate from these sources:

- the internal scheduler
- an explicit request to replicate
- in response to an event

### Sequence of events



Whenever the server receives the instruction to replicate, it follows this sequence:

1. It looks at the data that has been read from the configuration database to determine the source and target pairs of databases.
2. Next, it looks at the replication log table in the source database to find out which changes in the source database need to be replicated to the target.
3. The server processes the rows by querying their data from the source database tables and duplicating the logged action (insert, update, delete) to the target database, according to the specifications defined by the replication schema.

Rows are processed in FIFO order, i.e., in exactly the same order that they were put into the log by the triggers. The operation specified by the log record is applied to rows and columns of the tables in the target database :

- An insert is duplicated by inserting a new target row having the same primary key values and the same values for each column that is specified for replication.
- An update is replicated by finding the row with the same primary key and updating the replicated columns to their new values.
- A delete is replicated by finding the row with the same primary key and deleting it.

Clearly, it is essential that both source and target table must have primary keys or, at least, unique column sets that can be treated as keys. The corresponding keys or key-like sets must be compatible.



# Chapter 5

## Chapter 5 - Planning for Replication

## Chapter 5 - Planning for Replication

This chapter describes the planning and preparations that are necessary for implementing replication schemas that IBP Replicator can manage and process.

[Choosing a Replication Scheme](#)

[Database Design Issues](#)

[Physical Distribution Considerations](#)

[Conflict Resolution](#)

### 5.1 Choosing a Replication Scheme

You can choose from a variety of replication models to propagate data throughout your network. Your choice will reflect the requirements of your system. Models can be combined to fulfil needs that are too complex to be satisfied by a single scheme. Multiple schemas can be set up to fulfil different requirements, too.

#### Central-to-Standby

This simple model fulfils a requirement to replicate one way from a source database to a target. It is the scheme typically used to fulfil a "failover" requirement, whereby a standby database must be available to resume activity if a source database should fail.



A Central-to-Standby scheme can be set up with a Peer-to-Peer schema (see below) to enable changes in a failover database to be applied to the source database once it comes back on line.

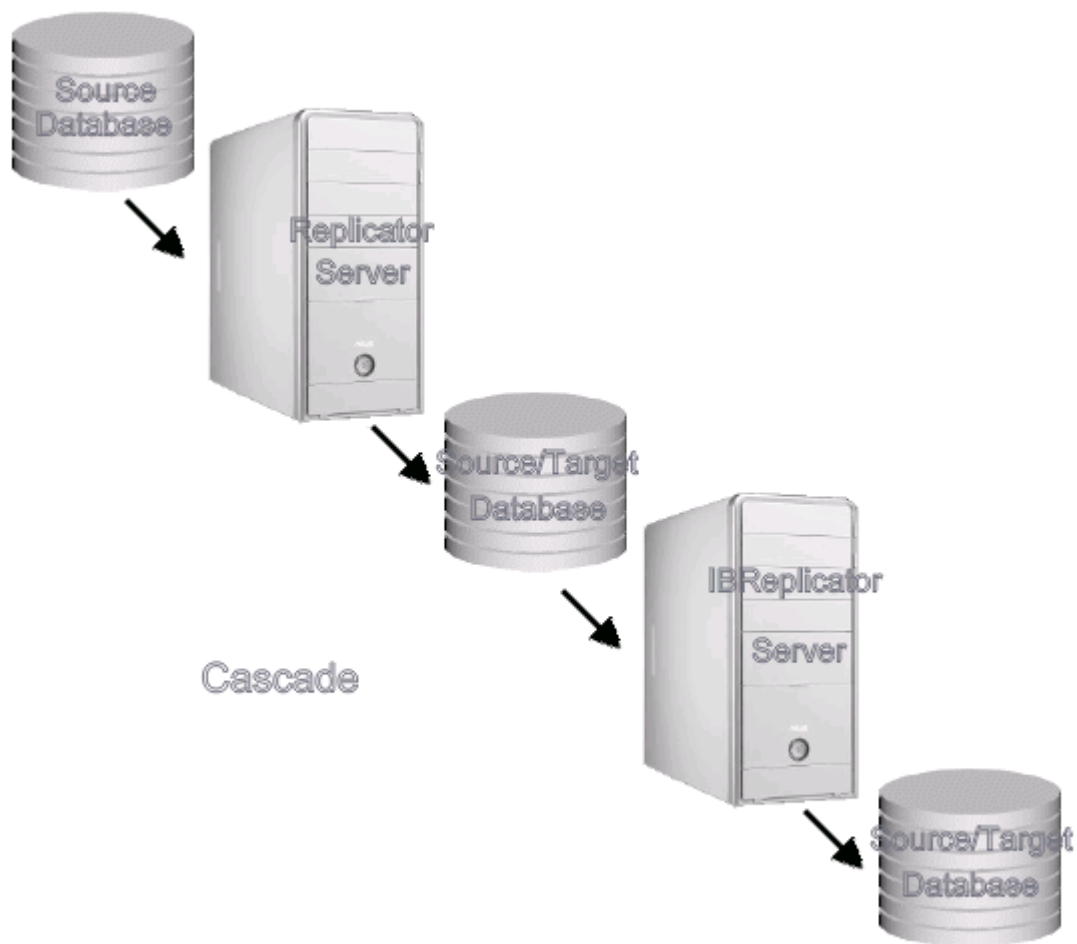
#### Peer to Peer

In a Peer-to-Peer environment, bi-directional (two-way) replication allows source databases to send changes to targets and vice-versa. Each database (source or target) can function autonomously at its own site. Changes from one database are replicated to the other to bring all databases up-to-date.



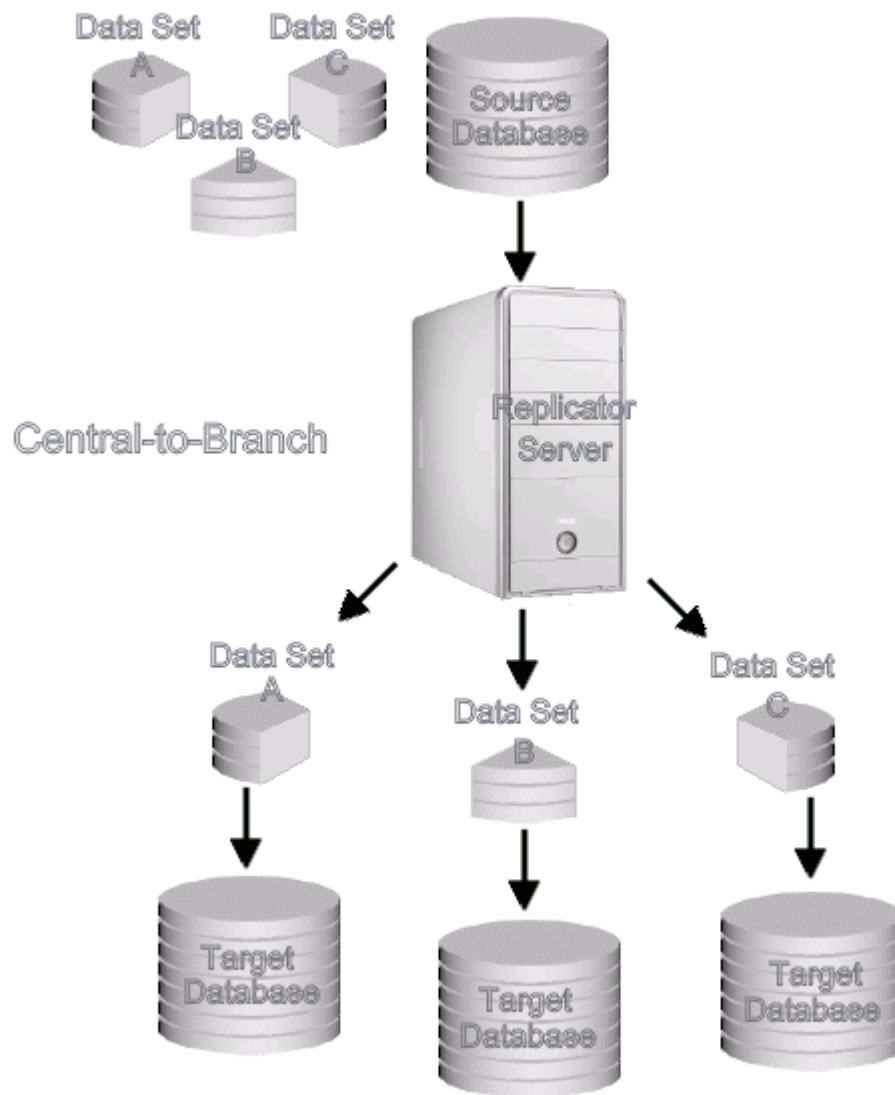
### Cascade

Cascade replication takes data from a source database and moves it to a target that, in turn, becomes a source for another target, and so on, down a hierarchy. An example of cascading replication might be where Head Office replicates to a regional office which then replicate to a branch. The cascade model might be combined with a Central-to-Branch schema (see below) so that each level is replicating separate sets of data to a number of levels below it.



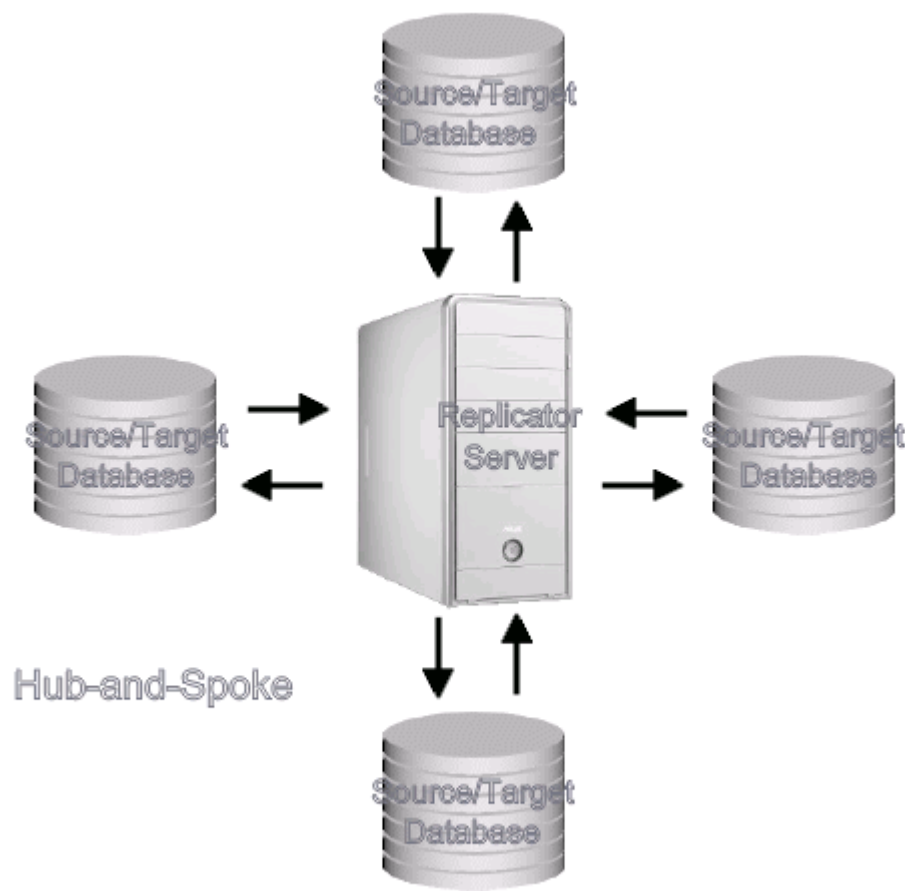
### Central-to-Branch

In a Central-to-Branch schema, subsets of the central database are designated for replication to branch sites. Data at the centre is horizontally partitioned, branch by branch. Usually, central data sent to branches is read-only. However, this schema is sometimes combined with Peer-to-Peer, so that branches update their sets locally and it is replicated back to the centre.



### Hub-and-Spoke

In a Hub-and-Spoke schema, a central database (the hub) replicates bi-directionally to multiple databases. It has a Peer-to-Peer relationship with each of its spokes. Cascade replication is implicit in this environment, because each of the spokes receives replicated data whenever the hub database or any of the other spokes is updated.



## 5.2 Database Design Issues

Certain aspects of your database design and structure will be important as you prepare for replication--particularly the unique or primary key constraints that will logically link source and target sets and the `TIMESTAMP` columns that you may wish to use for conflict resolution.

### Unique Keys

All tables must have unique column sets as keys. As a rule, the unique key used to map a source row to a target row should be the primary key, although it can be any column set that resolves as unique.

**IMPORTANT:** Firebird and Oracle allow null elements in uniquely constrained keys. Such keys cannot be used for mapping source and target tables for replication.

### Distributed Primary Keys

Maintaining keys that are unique across database boundaries are always going to be the

major issue when you are setting out to merge records from different databases. In scenarios where rows are being inserted in multiple databases using generated primary keys, it will be necessary to implement some changes to avoid unresolvable key violation conflicts.

- In a Central-to-Branch scenario a multi-segment primary key can be implemented, consisting of a Branch code + a generated ID. For existing databases, this solution will entail a large amount of work, both in the databases and in existing applications.
- An alternative to augmenting the primary keys is to use GUIDs. Implementing a GUID primary key would avoid impacting the existing single-column PK structure and possibly reduce the amount of change required in application code. On the database side, it requires a type change—simpler to do if you are using domains—and alteration of any trigger and stored procedure code that you have to generate values for keys.
- It is possible to use a uniquely constrained key that is not the primary key. This might be necessary in some heterogeneous schemas.
- Another alternative is to replicate changes, from all of the targets that are going to receive an update replication, back to the central database first, before anything gets replicated back out to the targets. All inserts from the targets go into the central database via an insert trigger which generates a new primary key for them from the central database. Replication out to the targets just has to be able to ascertain which of the "new inserts" originated from the target in focus and treat them as updates.

The main need would be to tag records on both sides, so that both the central and the target databases know that they need special treatment. How (or whether) this could be done would depend on considerations such as the impact if records that had been locally inserted by the target changed their primary keys after replication, whether a target's original key needed to persist for local usage, and so on.

### Timestamps for Replication Use

If your chosen method of [conflict resolution](#) is going to be by Timefield, it will be necessary to include a column in every replicated table—both source and target—to record the date and time of the changes. Triggers should be included, to ensure that every INSERT and UPDATE operation on those tables gets timestamped.

It is recommended that the timefield in each case be defined as `TIMESTAMP`. However, the source-target pairs can be of any comparable data types.

**IMPORTANT** The identifier (name) of the Timefield columns is case-sensitive in the replication schema. All Timefield identifiers must be identical and must be entered into the schema (without double-quotes) exactly as they are defined in the database.

### DATE vs TIMESTAMP

Timefield conflict resolution does not fail if either or both of the compared fields are of `DATE` type rather than `TIMESTAMP`. However, use of the date-only type is not recommended because replication might not occur where the source and target rows



have the same date in the timefield.

The choice is not an issue if the databases are all Dialect 1, since a date-only type is not supported there.

### Clock Synchronisation

Don't overlook the need to implement mechanisms to synchronise the system clocks of the source and target computers and, where necessary, to adjust for time zone differences.

## 5.3 Physical Distribution Considerations

The general purpose of replication is to ensure that changes made to a source database are duplicated to another database, so that the target database contains the same data as the source. Although the source and target databases can be on the same system, they are usually spread around a local network or distributed across a wider network.

### Distribution Conditions

Replication can be organised to distribute data to meet distribution needs in a virtually infinite variety of ways.

### One-way Replication

In one-way replication, one database is always the source. In physical terms, its targets receive data that they do not overwrite, append to or delete. Replication refreshes the content of the targets without further modifying the source database.

### N-way Replication ("Multi-master")

An n-way replication scheme, also known as multi-master replication, involves databases that act as both a source and a target to other databases. One form of n-way replication is where two databases replicate back and forth to each other. The "n" can be more than 2, since reciprocal replication schemes often involve schemas where the same databases can be sources and targets to a number of others.

### Heterogenous Replication

Source and target databases do not always have the same structure. One may contain tables that don't exist in the other. Tables may have different numbers of columns and different names for columns. A source dataset may replicate different sets of data to different targets. Any of these heterogeneous schemas is possible.

## 5.4 Conflict Resolution

Sometimes, a replication server may find a problem in trying to replicate an update to a row that is either missing or has already been updated. This is likely to show up in situations where the users of a target database are performing inserts, deletes and updates without being aware that changes may be happening elsewhere in a source database.

Issues like this generate conflicts during replication. Not all conflicts will be of the same type and different types of conflicts can be handled in various ways. Each source and target pair of databases can have its own conflict resolution settings.

### Handling Conflicts

IBP Replicator provides three mechanisms for handling potential conflicts.

#### Priority-based Resolution

Databases can be given priorities. The database with the higher priority number takes precedence. For example, if a source database has precedence, conflicts are resolved as follows:

- An update finding no identical key record in the target database is converted into an insert
- An insert finding a record in the target database with an identical key is converted into an update
- A delete finding no identical key record in the target database is ignored.

#### Master-Slave

The source database always takes precedence: resolution will be as described in the example above.

#### Timestamping

This method determines the newer version of data by comparing server timestamps written into the data. It needs a suitable timestamp column defined in both source and target tables. Older rows in the source database will not overwrite newer rows in the target.



# Chapter 6

## Chapter 6 - Installation and Setup

## Chapter 6 - Installation and Setup

### IMPORTANT!

The configuration database structure in IBP Replicator 5 is not compatible with those of releases before v.4.

Please take copies of your configuration databases before installing.

The newer Replication Manager will automatically upgrade old configuration databases on opening. Licences from Replicator versions older than v.4 are not valid for Replicator 5. Existing version 3 licences will be preserved and marked invalid. Any licences older than version 3 will be removed from the configuration altogether.

### Windows Installer Program

The IBP Replicator software for Windows comes as an executable Windows installer program named IBPReplicatorSetup-n.n.n-WinXX.exe, where the 'n' characters represent numbers and 'XX' is 32 or 64, indicating the 32-bit or 64-bit version. The first 'n' represents the major release number, e.g. 5; the second the sub-release number; and the third, if present, identifies it as a patch release.

The installer program may be compressed in Winzip format and distributed as an archive named IBPReplicatorSetup-n.n.n-WinXX.zip. Decompress the archive into a temporary location of your choice, using your favourite decompression tool--WinZip, 7Zip, WinRAR, etc.



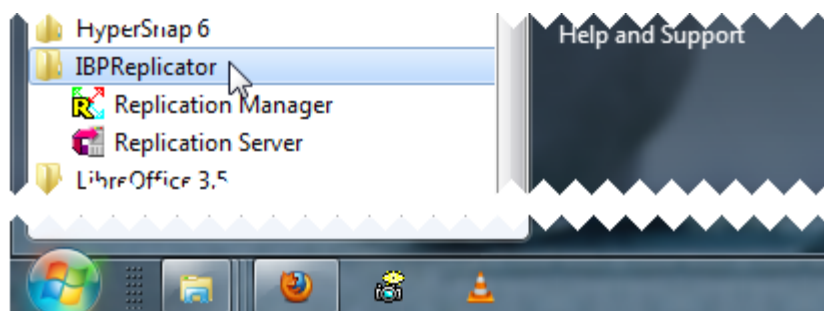
To install IBP Replicator to your hard disk, double-click the IBPReplicatorSetup-n.n.n-WinXX.exe file's icon in its directory

Follow the dialogs as the installer runs, providing a custom root location for the software if you don't want the defaults.

Once the installer has finished, close it. If instructed to do so, close down any applications you have running and reboot the machine.

### Start Menu Shortcuts

The installer program should have installed these shortcuts for you:



As a quick summary, here is what each of the shortcuts takes you to:



Starts the Replication Manager console program. See the next topic, [First Configuration Database](#).



Starts the Replicator Server as an application. Used when running a replication that is not the default one.

## ODBC

Replicator provides support for databases accessed through ODBC system DSN connections. The software for creating and managing a system DSN is provided in your operating system, either as a default application or (on non-Windows platforms) as an installable option. An ODBC driver for the database management system that Replicator will connect to is required. These components do not come with Replicator. They will need to be installed and working before you begin configuring the replication schema.

### Known Issues with Tested Drivers

MySQL: if it is likely the same record will be modified more than once in one replication session, you must enable the DSN setting "Details -> Cursors/Results -> Return matched rows instead of affected rows".

PostgreSQL: To replicate BLOBs to the PostgreSQL bytea type, the DSN setting ByteaAsLongVarBinary must be enabled.

You do not need ODBC to work with or between Firebird, InterBase or Oracle databases.

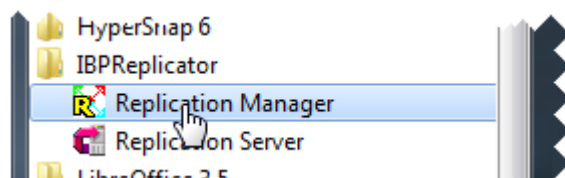
## 6.1 First Configuration Database

The Replication Server will use the configuration database to determine what it is supposed to replicate, where it should find the data (the source) and where the data is to be replicated to (the target).

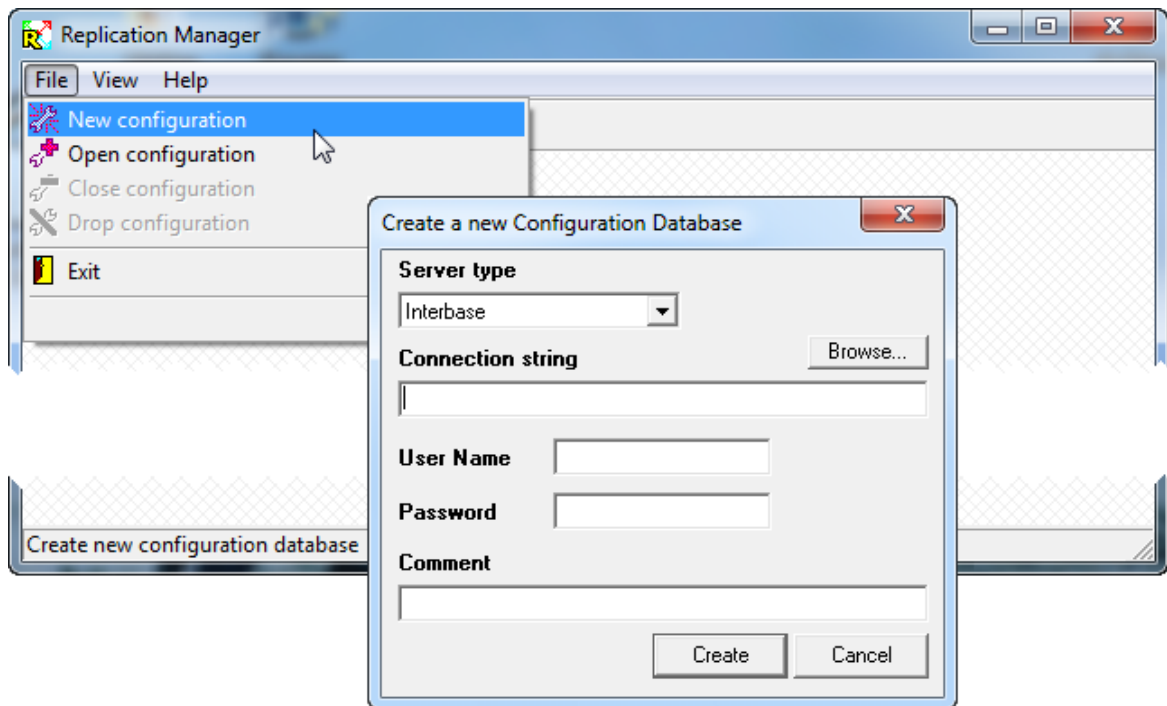
**ODBC** An ODBC datasource cannot be used as a configuration database.

### Creating a Configuration Database from Scratch

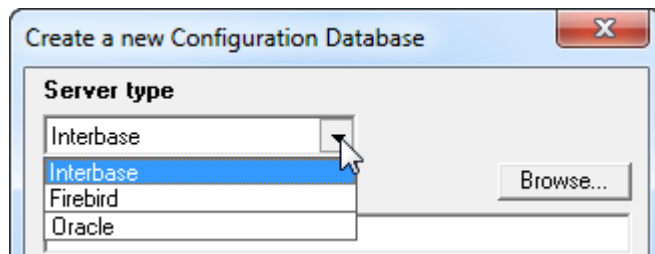
To create a configuration database, start Replication Manager from the Start Menu:



When the Replication Manager console appears, use File|New Configuration:

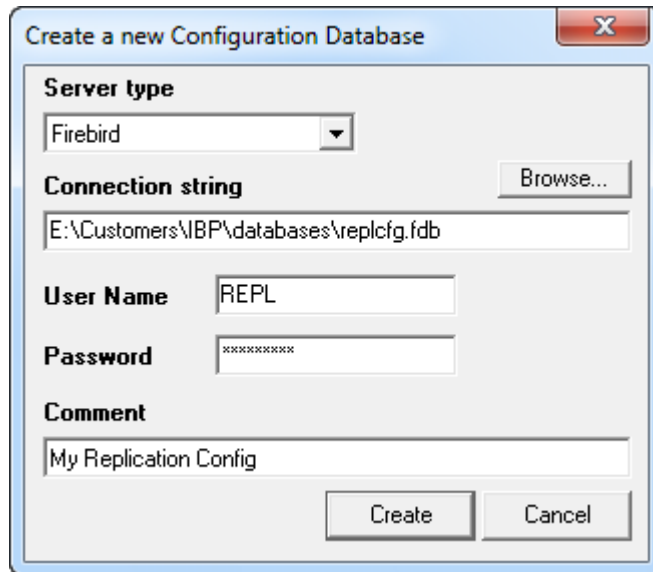


The configuration database doesn't have to be on the same host server as the replication service. The dialog starts by prompting you to select the server type (actual list depends on DBMS clients you have installed in your system) that the database will run on. This enables the programs to take advantage of features in Firebird 1.5 and higher that are not available on InterBase, such as database aliasing:



The rest of the configuration dialog prompts you to define connection string for the configuration database and a valid username and password. The Comment field is optional: it may be used in the title bar of some displays.

For example, suppose you decide to store your configuration databases in a subdirectory "databases" in some folder structure on your server. Your setup entries might be similar to this:



The server connection string in the screenshot is an example only, using a "serverless" protocol that is available locally with some models of Firebird and InterBase. Use the connection string that is appropriate for the client server environment in which you are working. If you are uncertain about it, check with your network administrator or advisor. For Firebird, consult the Quick Start Guide applicable to your server version.

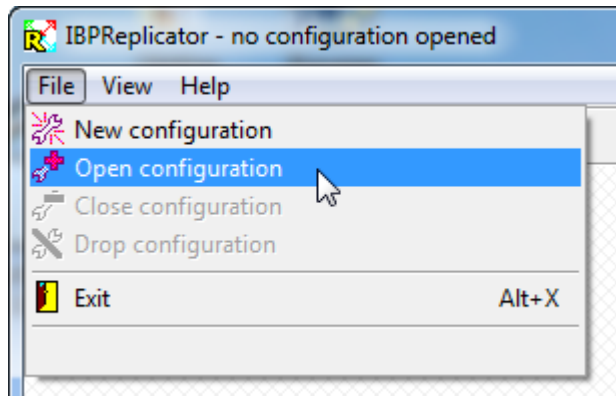
The User Name is that of a user on the server that is to be the owner of the new database. The user ADMIN in the example is an ordinary user name on the host server. Some sites might prefer that the SYSDBA user be the owner of replication configuration databases.

Creating a configuration database adds it to the list of known configuration databases, any of which can be accessed quickly and simply through the File menu. The menu also provides for opening and closing configuration databases and for dropping (deleting) them.

When you are ready to create your configuration database, just click the "Create" button and you're done.

## Using an Existing Database

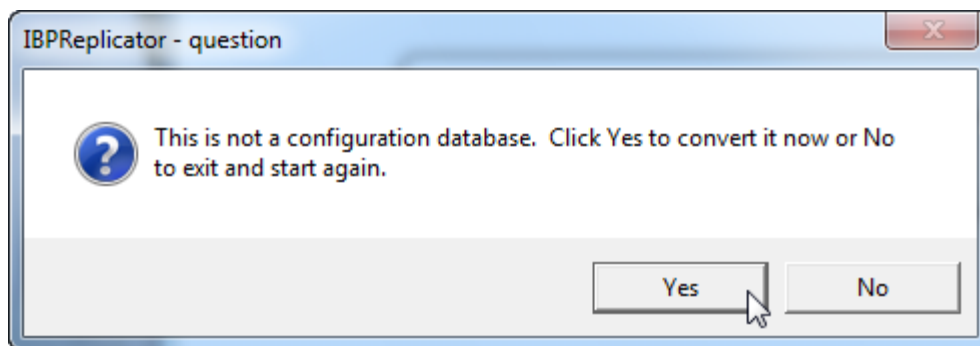
It is possible to store a replication configuration in an existing database, specifically the source database, for an [event-triggered replication](#). For this, use the Open configuration dialog from the File menu:



You will get a dialog almost exactly like the one for creating the regular configuration database. Fill in the fields to access the database, remembering to include a valid username and password, and submit it.

You might like to include a useful nickname for the configuration in the Comment field, e.g. "Synch Checkouts" might be appropriate for a replication that is to be used to keep point-of-sale terminals in synch with a Sales database.

You will be asked if you want to make it into a configuration database:



(For Oracle, the message may differ slightly but the same question is implicit.)

Click Yes to have the program create the replication metadata in the source database. If you don't want to proceed, click No to abandon the idea.

**NOTE** Normally, you won't want the overhead of performing all replication instantly; it is perfectly OK to break your replications out so that most of the replication runs asynchronously and the source database configures just the essential "close-to-real-time" replication.

The database will be added to the selection list beneath the File menu.

## 6.2 Registering Databases

The registration of a database identifies the database to IBP Replicator and provides all the information needed to connect to the relevant databases. All databases that are involved in replication need to be registered, both source databases and target databases.



Once the configuration database has been created, you can register the databases that are going to be involved in replication.

To start registering a database, stay on the Databases tab. Above the panes is a toolbar.

## The Toolbar

The toolbar gives access to the commands available during database registration. They provide shortcuts to the same commands that are available from the Databases menu. The active buttons at any point appear in an "enabled" state, while unavailable buttons are greyed out. Hover hints are available on all of the buttons.



Add Database is for adding a new database or DSN to the configuration



Edit Database is for doing any modifications required to the database or DSN currently selected



Remove Database is for taking a database or DSN out of the configuration. It doesn't drop the database, of course!



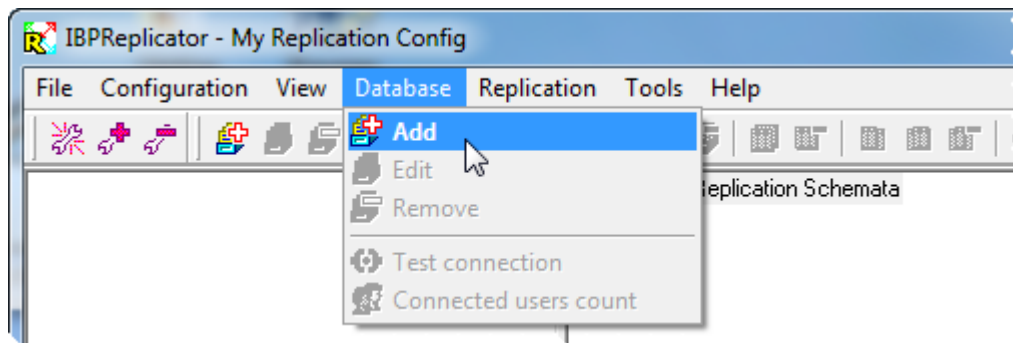
Test Connection does the same thing as right-clicking on a database or DSN and selecting Test Connection



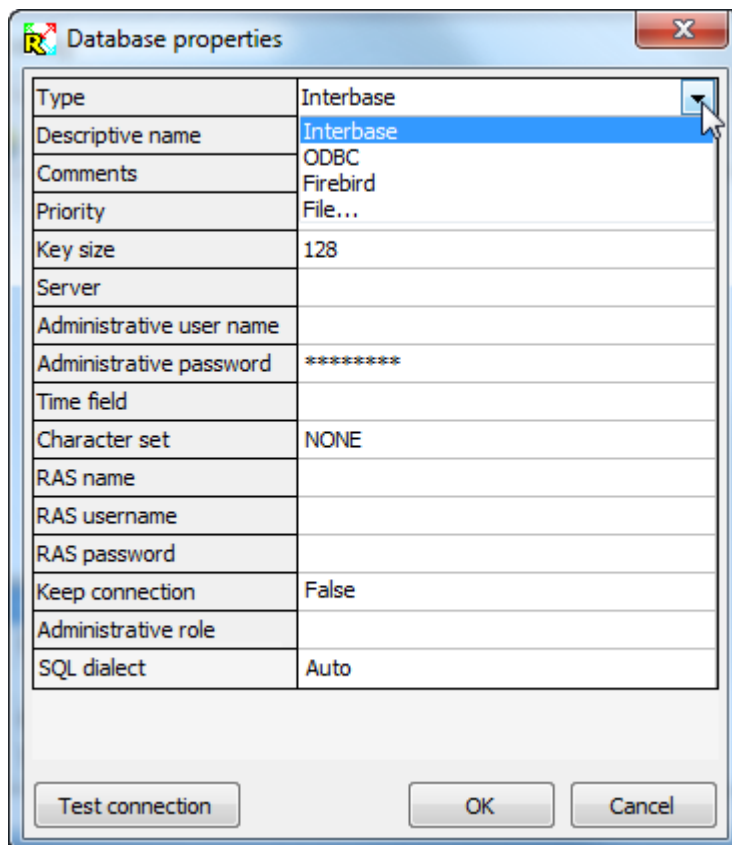
New Schema launches the creation of the replication schema that you are about to define

## Registering a Database

Click the Add button in the toolbar:



A new database record is initiated in the fields editor, ready for adding the necessary information for the replication server to find the database:



Because of variations between the different database drivers, the set of fields displayed may not be exactly as shown in the screenshots.

### Fields for the Database Record

Type	Firebird
Descriptive name	Source Database
Comments	
Priority	Highest
Key size	128
Server	localhost:replsource
Administrative user name	ADM
Administrative password	*****
Character set	NONE
Time field	
RAS name	
RAS username	
RAS password	
Keep connection	False
Administrative role	REPL_ADMIN

[Database] Type: Select the server type (Firebird|ODBC|InterBase|Oracle|File..) of the database you are registering. The fields that appear after this selection has been chosen will vary, according to the database type selected.



Versions of IBPReplicator before v.4.1 did not behave correctly with the Firebird driver if the source database was Dialect 1. The Firebird driver now selects the correct dialect automatically. If you are upgrading Replicator from v.4.0 or an earlier version on a Dialect 1 database, any workaround that involved using the InterBase driver should be corrected to use the Firebird driver, regardless of dialect.

**Descriptive name (required):** A "user-friendly" name (not the database file name) that will be meaningful to anyone managing the replications. For example, "Head Office Source" is more meaningful than "New database"! It is not a formal identifier, so you may include spaces and punctuation as you wish.

**Comments (not required):** This property is entirely for your convenience, to document and explain each database you register.

**Priority:** Used when priority-based conflict resolution is being set for the replication. In this case, replication conflicts are resolved by ensuring that data from the database with the higher priority number is preserved when conflicts arise.



**Key Size:** Size of the fields in the operation log table that contain the old and new key values of every replicated record. This setting is used, with other system objects, during creation of the table and is ignored later. The value must be big enough to allow the field to contain a string representation of your longest primary key. However, setting it to an unnecessarily large value will degrade database performance.



**Server (required):** This is the network path (where applicable) and the full file path (or database alias, for Firebird) of the database. It follows the syntax used by the Firebird, InterBase or Oracle client, respectively, to connect to databases. It should accord correctly with the format required by the network transport protocol and the file system of the host server that is running the database engine.

For Oracle, this will be the URL, TNS or equivalent that will find the database according to your server's configuration.

For Firebird and InterBase, suppose the database file name is "source.fdb" and it is located on a Windows server named "dev" in a directory called G:\IBPhoenix\IBP Replicator\data.

For TCP/IP protocol, the server entry should be

```
dev:G:\IBPhoenix\IBP Replicator\data\source.fdb
```

For Named Pipes (sometimes referred to as NetBEUI) protocol, the server entry should be

```
\\dev\G:\IBPhoenix\IBP Replicator\data\source.fdb
```

For an older InterBase database, named "source.gdb" on Novell, with the G-drive mapped as "vol5" it should be

```
dev@vol5:\IBPhoenix\IBP Replicator\data\source.gdb
```

For a Linux or UNIX host, our example path would be of the form

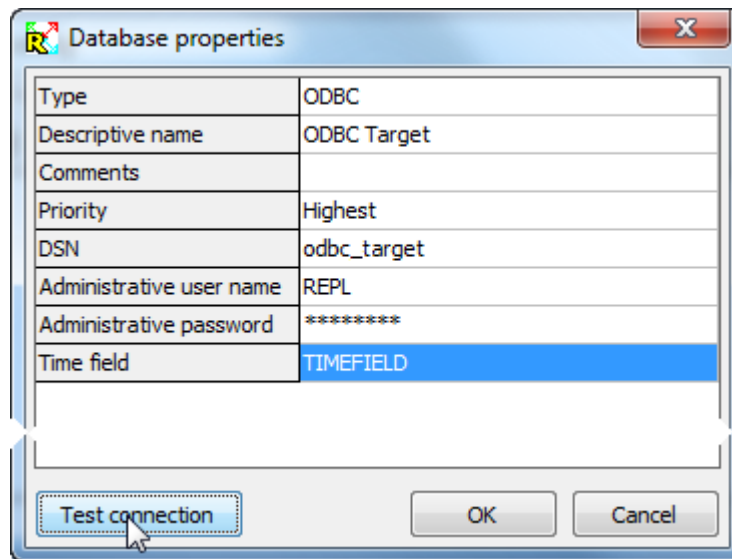
```
dev:/IBPhoenix/IBP Replicator/data/source.fdb
```



When focus is in the Server field, you can click a button to launch a Find File dialog:



#### ODBC



**DSN** Appears only if the selected server type is ODBC. A selector button will appear when you move focus into the field. Use it to select the system DSN you have prepared for this database.

**Administrative user name** (required for InterBase, optional for Firebird and Oracle if using trusted authentication): The database server authentication username that the Replication Manager should use when connecting to and administering the database being registered. It can be a different user name from the [Replication username](#).

For a database used as a source for any schema the user must be one of following:

- SYSDBA
- The Owner of all replicated tables
- A user that has been granted the RDB\$ADMIN role. In this case, the [Administrative Role](#) must be registered as RDB\$ADMIN.

#### IMPORTANT

Failure to comply with this rule will prevent the Replication Manager and cfgd from creating the logging triggers on replicated tables.

If this is an ODBC database, use this field if the connection requires it; otherwise leave blank.

**Administrative password** (required for InterBase, optional for Firebird and Oracle if using trusted authentication): The password for the administrative user name. The username and password belong together, as the two parts of a user login defined in the security database of Firebird or InterBase. See also Administrative Role (below).

If this is an ODBC database, use this field if the connection requires it; otherwise leave blank.

When you start to make an entry in one of these fields, a dialog will pop up for completion of both entries.

**Character set** (required if the default character set of the database is other than NONE or, for Oracle, is not the same as the one in the environment variable NLS\_LANG.) It should be a character set that is compatible with the DEFAULT CHARACTER SET for the database.

#### ORACLE

For Oracle, it can be set to a character set different to the system-wide default (NLS\_LANG) if the source and target databases have incompatible character sets. An example would be where one database in a replication pair was created with WE8WIN252 and loaded with data in CL8WIN1251. With other replicators, the faulty database could not be replicated to one that had the correct character set.

The drop-down list of character sets is not comprehensive. The name of a missing character set can be entered manually.

**Time field:** Used when time-stamped conflict resolution is being set for the replication. If you require this form of conflict resolution, your tables should all include a column with exactly the same name for each, that contains the date and time when each row was inserted or last updated. If this functionality has been implemented, IBP Replicator will ensure that the more recent row is preserved when conflicts arise. Refer to the [design notes about Timefields](#) for discussion about choosing the right date/time type for this style of resolution.

**IMPORTANT** The Timefield name is case-sensitive AND it should be entered without double-quotes. You must type the name of the field exactly as the server sees it in the database. That means it should be in all upper case characters unless the field has been defined with a quoted identifier.

For example, if it was defined by `..ADD "TimeFieldForRepl" TIMESTAMP`, enter it as `TimeFieldForRepl`, without the quotes. If it was defined by `..ADD TimeFieldForRepl TIMESTAMP`, enter it as `TIMEFIELDFORREPL`.



Administrative role: Similarly to the administrative user name and password, this is the SQL role that is used by the Replication Manager tool when administering the database being registered. If the Administrative User parameter is a regular user that has been granted the RDB\$ADMIN role, then only RDB\$ADMIN is valid here.

The replication server uses another "replication role" that is provided when databases are specified in a replication schema.

**ODBC** Because the ODBC interface does not support roles, it is not possible to assign an administrative role for an ODBC-accessed database, even if the target server supports them.

## Remote Access Parameters

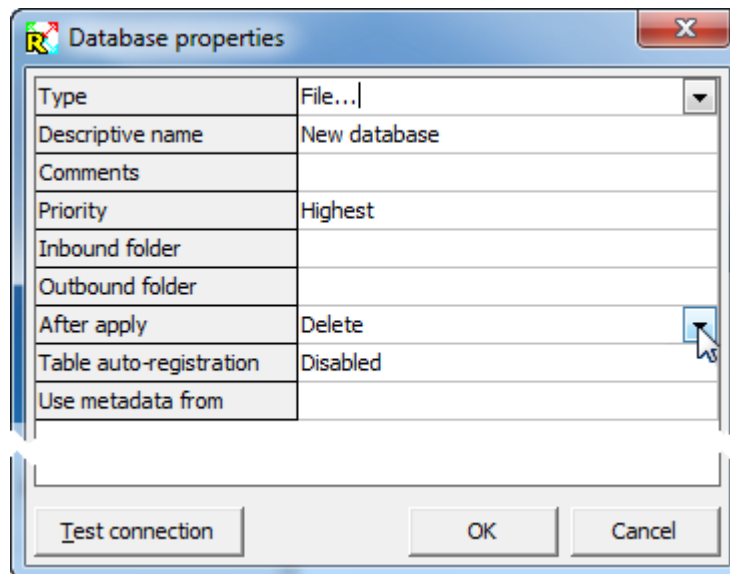
IBP Replicator can use the Windows Remote Access Services (RAS) to replicate across a dial-up connection. The following database parameters control how it works:



RAS name	This is the unique name of the dial-up connection as defined in the Windows "Dial-up Networking" dialog
RAS username	The dial-up username that the replication should use when dialling out
RAS password	The password that the replication should use when dialling out
Keep connection	Controls whether the replication hangs up as soon as it is disconnected from the database, or stays connected until the end of the replication cycle (all schemas) leaving a user to terminate the RAS connection

## Fields for Replicating to Files

If you select Files.. as the Database type, the fields that appear are fewer:



These fields refer only to off-line replication. For more information, see the advanced topic [Offline Replication and Synchronization](#) in Chapter 9.

**Inbound folder:** the full path to the folder where replicated data in off-line files will be received. If you are using Avalerion, make this folder the same one that is configured in `ibpr_cdc.conf` as `OutputDirectory`.

**Outbound folder:** the full path to the folder where replicated data in files will be written for subsequent transfer by external means.

**After apply:** in this and higher versions of Replicator, you can opt either to delete the offline files after they have been applied or to have Replicator preserve and rename them automatically with an extension of ".done".

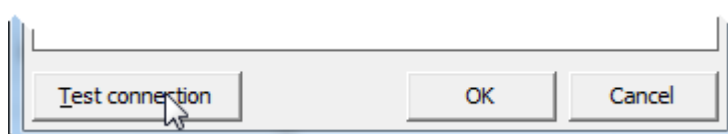
**Table auto-registration:** enable this property if your Firebird server is Avalerion and it is configured for embedded replication using the `ibpr_cdc` plug-in. If the databases have the same metadata structure, no further schema configuration will be required.

**Use metadata from:** You first need to have registered a database whose table definitions will be used to determine the format of data to be replicated off-line. When you focus in this field, the descriptive names of registered databases will be available to select from.

## Checking the Connection

When you have defined the database, test the connection to make sure that you have input the details correctly. You can do this before saving the current parameters.

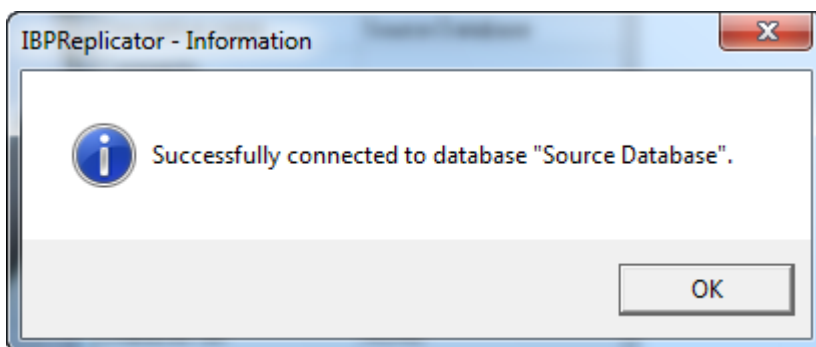
Click on the button in the registration pane:





or click the  button on the toolbar.

If all is well, you should see this message:

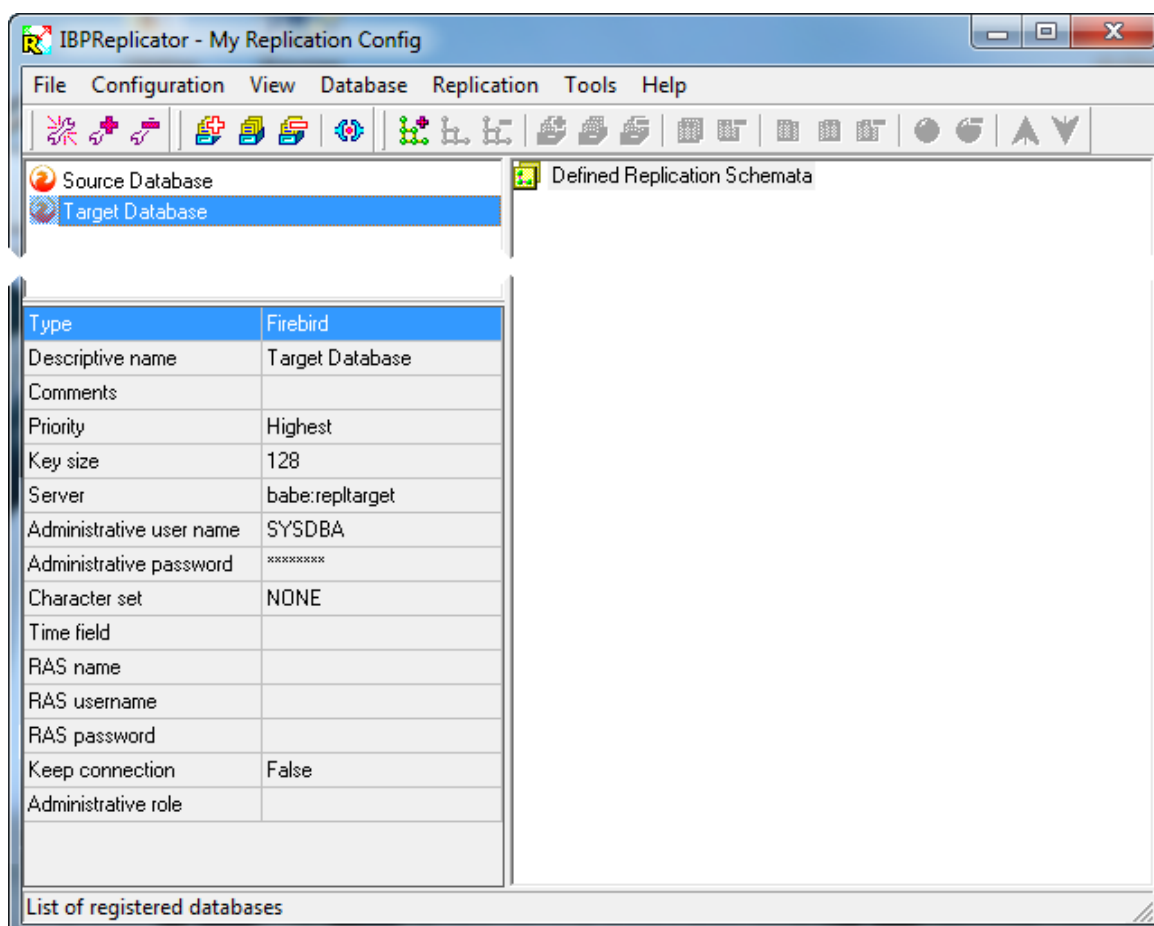


If the connection attempt fails, check and correct the parameters until the connection test succeeds.

### Registering the Database

Finally, to register the database, just click the OK button. The newly registered database's descriptive name is added to the list.

It takes at least two registered databases to make a replication configuration, so carry on and register each database as required.



Once you save the last of your databases, you are ready to move on and [define a replication schema](#).



This is the Toolbar button that launches the creation of the replication schema. You can also right click on the legend in the right-hand pane and select New from its context menu or use Replication>Schema>New in the menu at the top of the screen.

## 6.2.1 About Licences

You can use the Replication Manager for many of your initial configuration tasks without needing to install your licences. However, you won't be able to start a replication server or activate any of your replications until you install your licences.

Existing IBPhoenix Replicator V.3 licences are valid for V.4 and will be preserved. Licences from versions prior to V.3 are not valid for Replicator 4 and will be removed from the configuration altogether.

## Licences Required

Two types of licence are needed for replication: Server and Replicant. Licences must be installed into the configuration database that is to be used for your active replication.

- a Server licence enables the Replicator application to run
- a Replicant licence allows a database to be accessed as a source or target

Licences are entered using the License Manager dialog, which you access from the Tools Menu of Replication Manager.

As a rule of thumb, you require one Replicator licence to run a replication server and one Replicant licence for each database that is a source or a target. Replicator server and Replicant licences are bundled in denominations to suit your particular needs. One Replicant licence is needed for each database that is involved, including the Source database.

For example, the minimum IBP Replicator configuration of one source database being replicated one-way to one target database requires a single Server licence and two Replicant licences.

A Replication Server licence is bundled with one Replicant licence automatically. For this version of Replicator, the included Replicant licence is issued separately to facilitate its use with off-line replication.

Replicant licences are counted when the Server process checks that there are enough Replicant licences available for all of the databases defined in the current replication schema. Once a database is licensed for replication, it can be involved in multiple replication schemas without the need to add more Replicant licences for it.

Licences are "server agnostic": you do not need different licences for Firebird, InterBase or Oracle.

### Licence Counting

Many configurations are possible. Checking is concerned only with ensuring that the count of Replicant licences is at least equal to the number of databases configured in the schema. Additional Replicant licences are available.

### Evaluation Licences

You do not need to buy licences for testing IBPhoenix Replicator. Just start by using the EVAL license for testing a simple, one-way replication schema. It will supply four Replicant licences for your tests for 14 days.

- For more details about IBP Replicator licensing, see the topic in the [IBP Replicator Requirements](#) section.
- For instructions about installing licences, go to the [License Manager](#) topic.





# Chapter 7

## **Chapter 7 - Defining a Replication Schema**

## Chapter 7 - Defining a Replication Schema

After registering all the databases that are to be involved in your replication schema, the next task is to identify the data that will be replicated.

### Summary of Steps

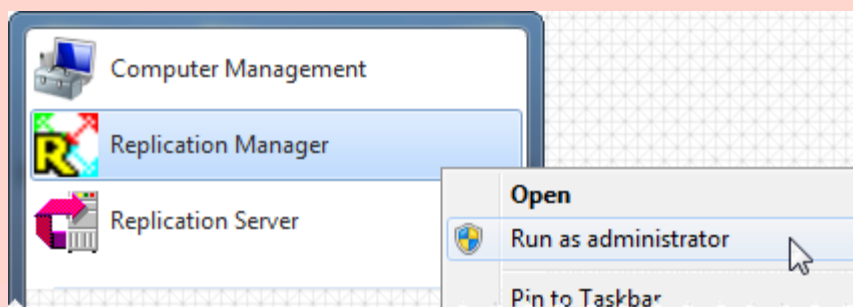
The steps involved are.–

1. [Creating the schema](#)
2. [Choosing the source database](#)
3. [Choosing the target database](#)
4. [Choosing replicated tables](#)
5. [Table settings](#)
6. [One-time synchronization](#)
7. [Identifying primary keys](#)
8. [Choosing replicated columns](#)
9. [Creating system objects in the source database](#)
10. [Customising the global settings](#) (optional)

### Assigning the Default Configuration Database

One configuration database should be assigned as the default. Once the Replication Server is running, it will automatically connect to your default configuration database and use the information there to do its work.

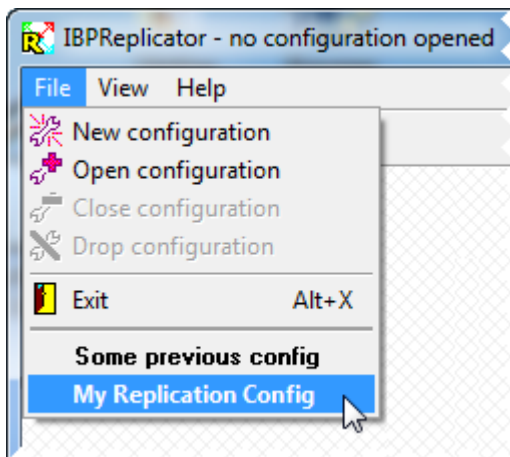
Important :: This task will write to the Windows Registry, which requires Replication Manager to run with escalated privileges on Windows 7, Windows 2008 server and higher Windows versions. To do this, right-click on the shortcut for Replication Manager and select Run as administrator:



and answer YES in the security dialog that appears.

In Replication Manager you can pick up the configuration you want to work with by

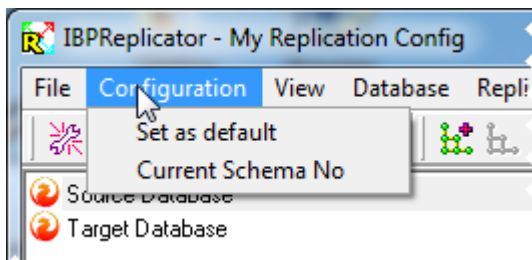
clicking [File](#) on the menu bar and selecting one from the drop-down list:



If a default configuration database is currently assigned, it appears in bold-face. You can change the default one by selecting a different one from the list.

Select the configuration database you want to be the default and click to open it.

Now, select [Configuration](#) from the menu and simply click on "Set as Default":



## Next

Your next step will be to [create the Schema](#) for your replication.

However, you may wish to look over the [default settings](#) first and decide whether you want to [customise the options](#).

## 7.1 Creating the Schema

A replication schema contains information about what should be replicated, from where, to where. For the purposes of this help document, we'll assume that you are working through the steps previously described and are creating your first schema.

However, you can define multiple schemata for multiple replications that are to be cycled through in order. Typically you would already have registered databases, defaults, etc. already defined, and would wish to pick up definitions from some level of the "tree" to

generate successive definitions in your new schema.

## The Replication Tree

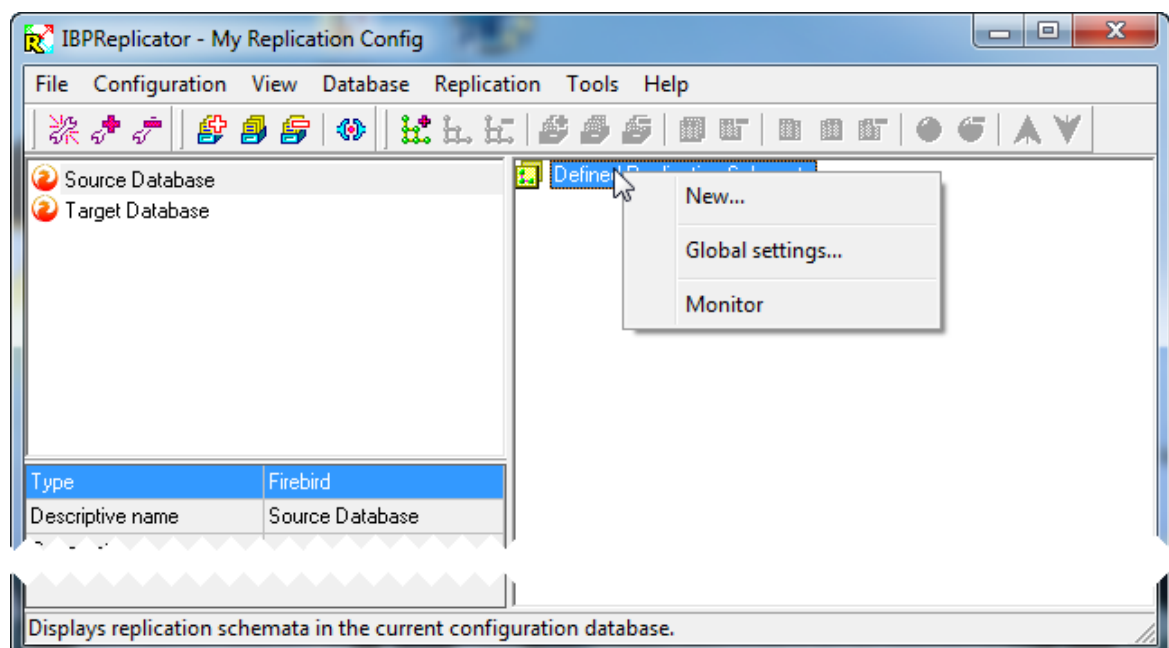
The replication tree—the right pane of the Replications tab in Replication Manager—provides the running details of everything you define in the currently selected configuration database. As you progress through your definitions, the tree will expand and enable you to access aspects of your schema at each level of detail.

### Ways to Work on Definitions

The interface provides several different ways you can approach definition and maintenance tasks at various levels:

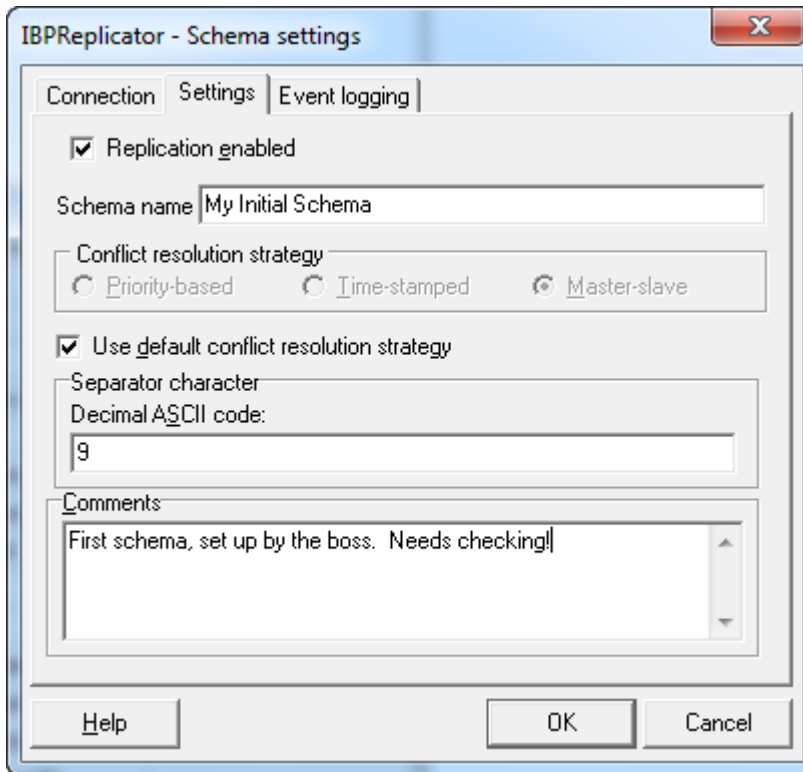
- using the right-click context menus on the tree nodes
- double-clicking the tree node
- making selections from the Replication menu

You can mix the tricks in whatever way suits you best. Following are some examples of the various approaches.

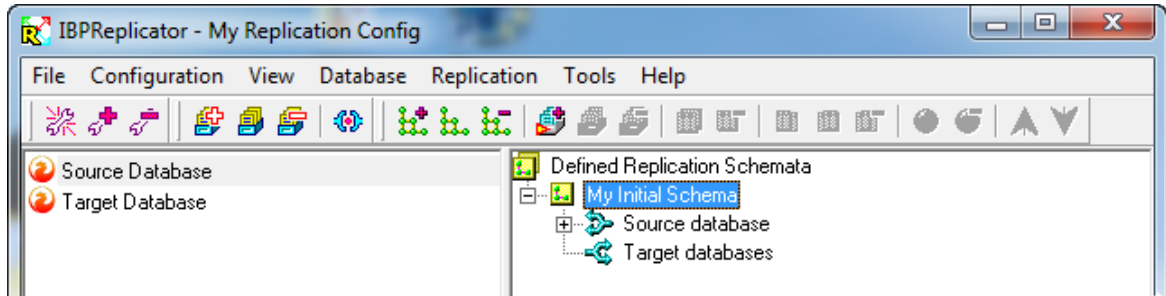


Here, the root node of the replication tree is selected and the user right-clicks on it to display the context menu. The New... option is selected and a left click will provide the input dialog for defining a new schema:





After you click OK, the new schema appears in the tree with nodes for the source and target databases:



As another approach, you could have double-clicked on the "New schema" icon in the task bar or you could have used the Replication > Schema > New in the Menu bar.

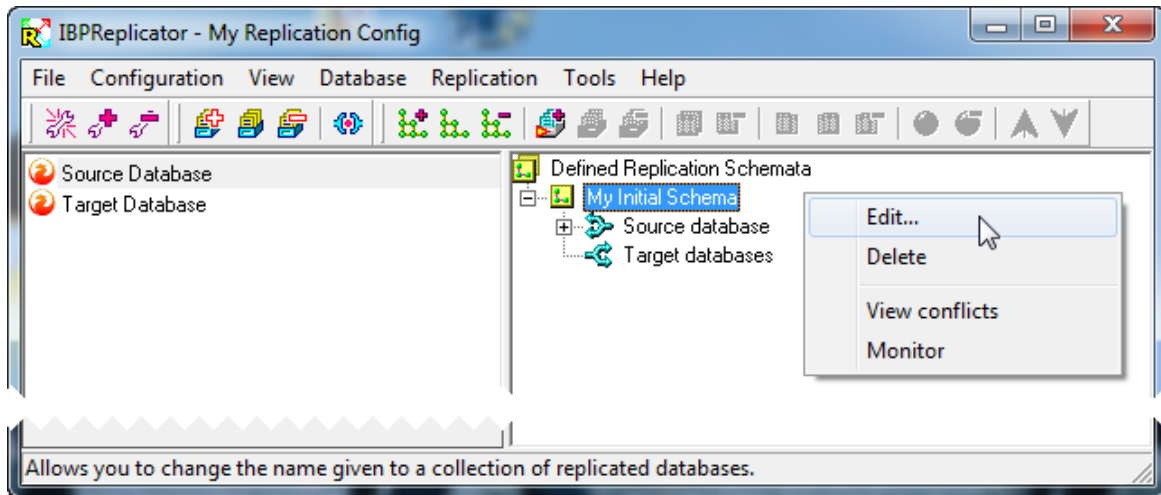
**TIP** Use schema names that will be meaningful not just to you but to anyone else who is going to manage your replications!

Note that you could have stayed within the schema definition dialog to develop the schema definition further, as described below, clicking OK once you were ready to save the work.

### Tasks Pane Context

Notice how the icons in the Tasks pane change, according to where the focus in the tree pane is. For example, if you single-click the newly created Schema node you will see a

new icon in the Tasks pane enabling you to add another target database to the schema. As with other nodes, you also have similar contextual options by selecting and right-clicking the schema's node directly in the tree. In each approach, you will be able to edit the name of the schema or delete it entirely:



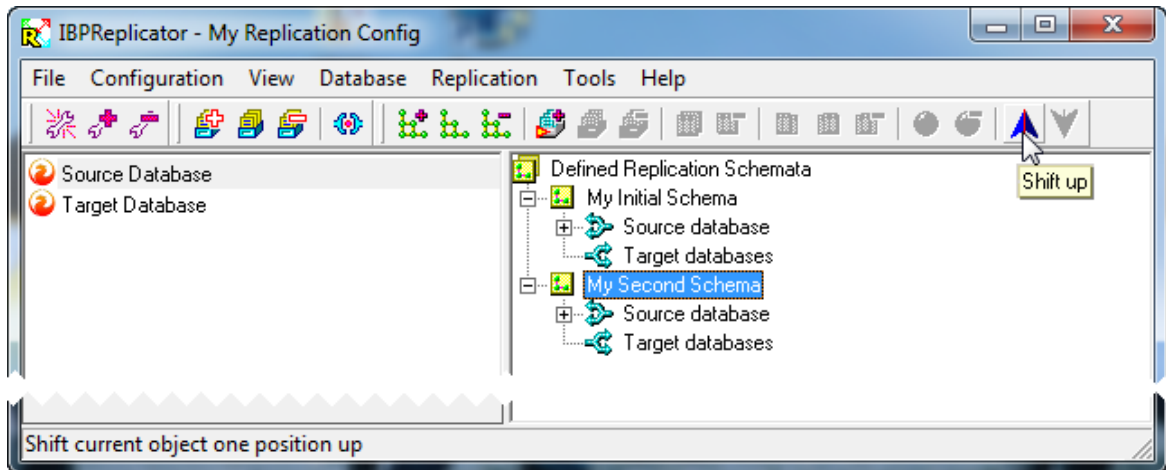
Be aware that deleting a schema will remove all replication information for that schema, including anything stored in lower nodes. Once the schema is gone, the replication cannot occur!

Expanding the schema node reveals two new nodes labelled [source database](#) and [target database](#). These nodes can be approached in the succeeding steps.

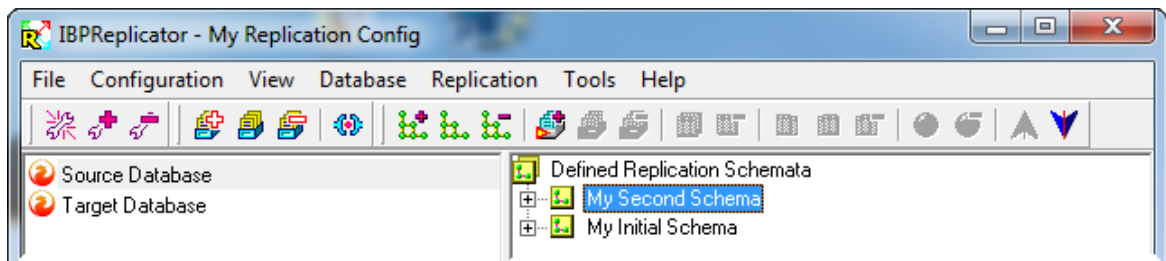
## Adding More Schemata and Ordering Them

To add another schema, simply select the Defined Replication Schemata node again and repeat what you did for the initial one.

Once you have more than one schema, the Shift Up and Shift Down buttons are enabled in the toolbar when a schema node is selected. They provide the means to alter the order in which your replications will take place in the cycle:



Click on the Shift Up button to move the selected schema up the order or Shift Down to move it down:



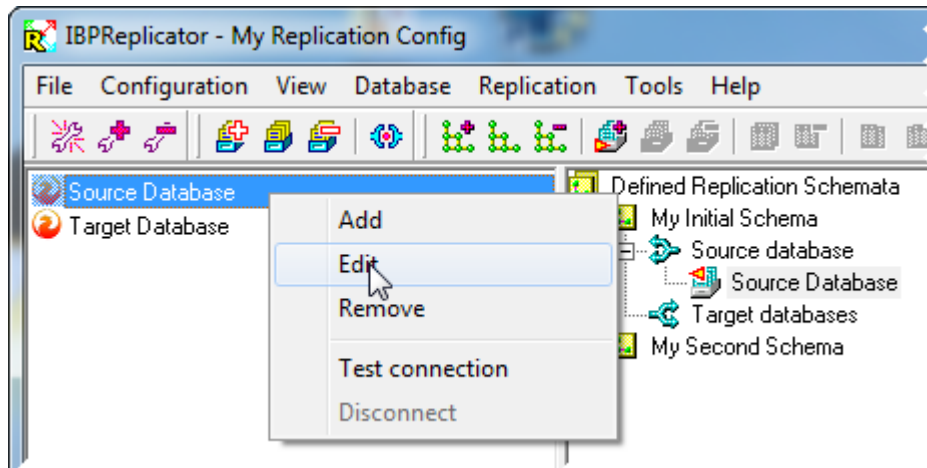
## 7.2 Choosing the Source Database

The source database is the one in your replicant pair that will be replicated from. You already defined at least your source database at an earlier step, when [registering databases](#).

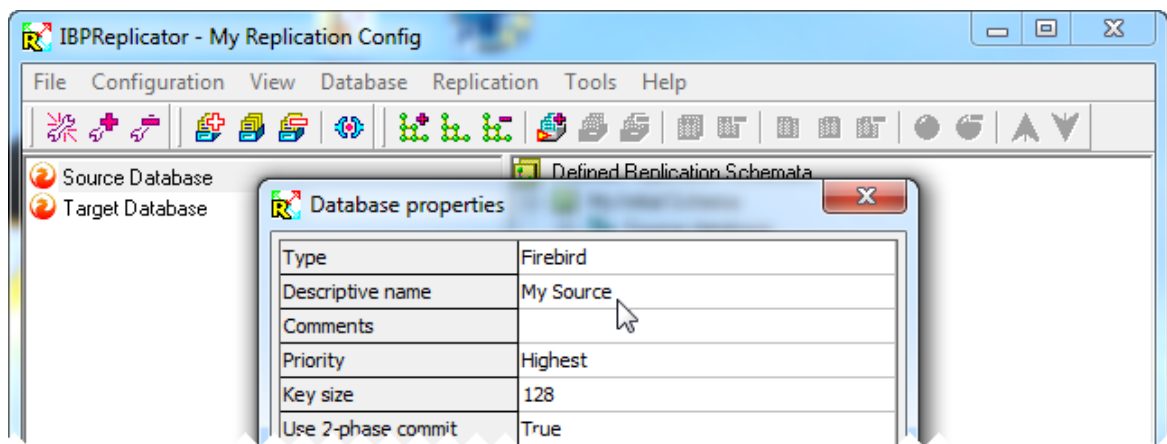
You cannot set up an ODBC datasource as a source database using Replication Manager. [More..](#)

### Editing Database Attributes

Suppose we decide that our descriptive names—Source Database and Target Database—are not to our liking. We can use the the Edit... option on each of their nodes in the left pane to change them:



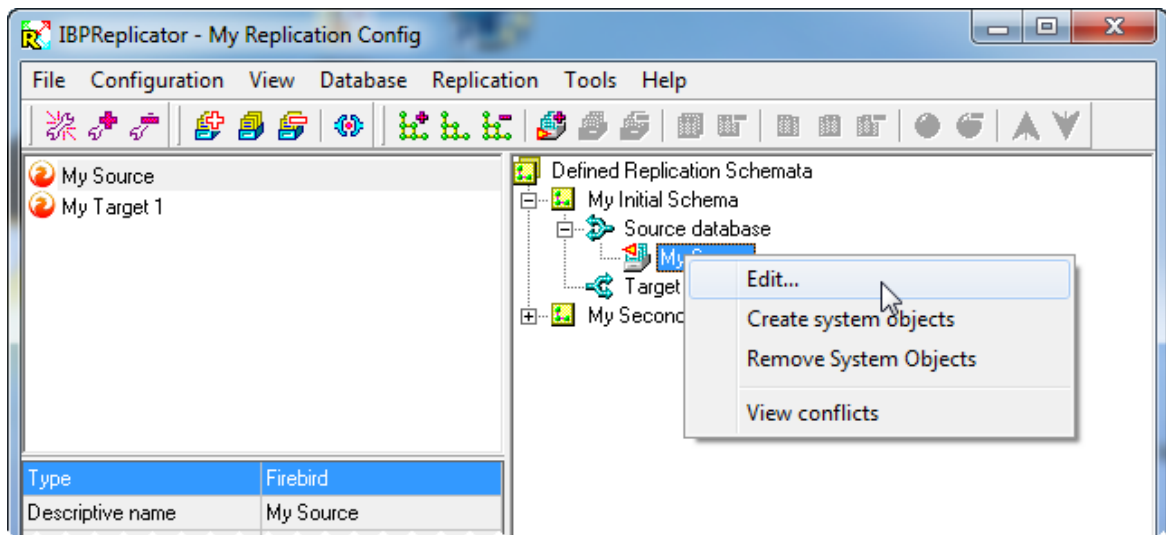
We'll change "Source Database" to "My Source":



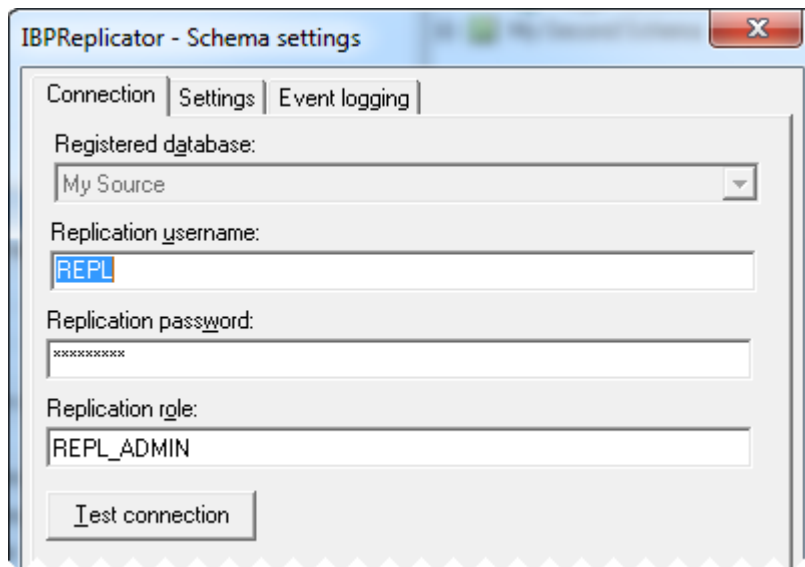
You can change other attributes of registered databases, too. If you change the path of a database, don't forget to test the connection before clicking OK to save the changes.

## Source Database

If you want to edit some characteristics of the source database with respect to the schema, or just check those characteristics, expand the schema node, right-click the named node beneath the Source database node and select Edit...:



The Edit dialog that appears contains the characteristics, spread across three tabs:



## Connection Tab

The first tab of the dialog shows the database that will be the source of your replication.

### Registered database

Because there can be only one source database for a replication, the "friendly name" of your source database is greyed out and not editable.

### Username, password and role

You can change the username, password and role (if applicable) to be used by the replication server when it connects to the source database.

The name of the "replication user" does not have to be the SYSDBA or that of an Administrator of the operating system.

For example, the REPL (replicator) user can be used for any replication. It must be used if you plan to implement bi-directional or n-way replication.

### About the REPL user

The user named REPL is recognised by IBP Replicator as a special user. Any changes made to the source database by the user REPL will NOT be replicated. This is by design, to prevent replications from a target database back to a source from going into a loop.

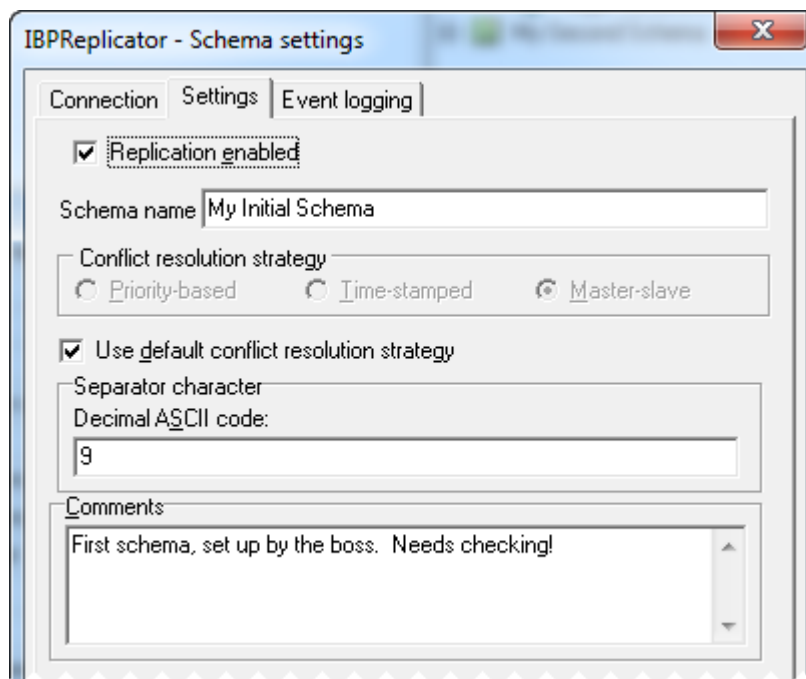
### Test the connection

Whatever user you assign, make sure that it is added to the relevant security databases on each machine involved in the replication. Once you have done that, click the Test Connection button. If login is unsuccessful, review your entries and make any necessary corrections before trying again.

It is pointless proceeding now unless you can make a successful connection test.

## Settings

The second tab enables you to customise settings for this source–target pairs in this schema. They are initialised to the global default settings, but can be modified without affecting either the defaults you have already set or any settings you might have customised for another replication.



You can add any free-form comments you like in the Comments box, to augment or override any comments inherited from the defaults.

### Replication enabled

This check box should be checked ON unless you need to prevent replication from occurring. For example, you would check it off if you were taking the network down or powering one of the servers down for maintenance.

## Conflict resolution strategy

The replication is initialised to use the default conflict resolution strategy that you might have customised previously—refer back to [Customising the Default Settings](#). The current default setting appears above, greyed out. In the illustration it is Master–slave.

You can override the default conflict resolution strategy by checking off Use default conflict resolution strategy and specifying one.

Don't select Time–stamped unless you have the necessary Timefield columns in all of your source and target tables.

## Separator character

IBP Replicator stores the primary keys in text format. The separator character is used to separate the values in primary keys with multiple columns. Thus it is essential that this be a character that cannot appear in a key value.

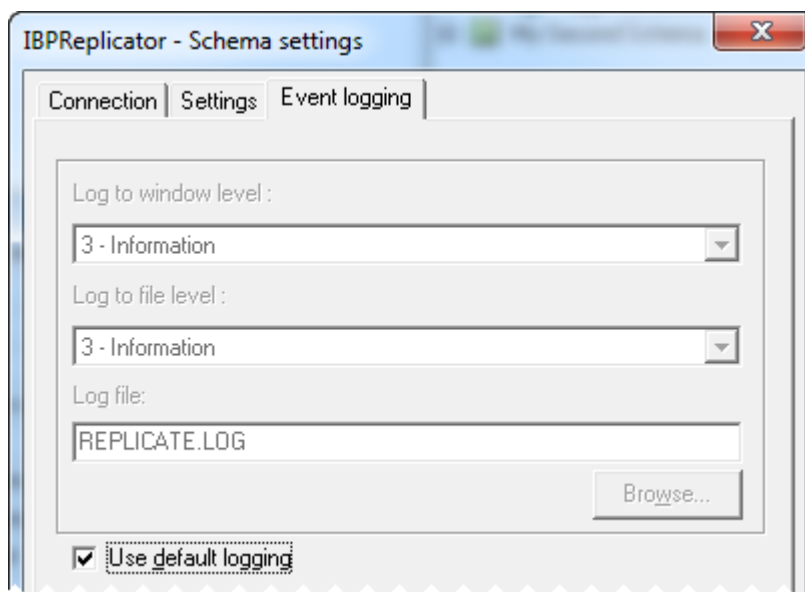
The default character is <TAB> (ASCII 9), a non–printable character sometimes represented by the musical "minim" symbol or Ctrl–I. In the unlikely event that ASCII 9 would conflict with values in your data's primary keys, provide the code for one that is suitably improbable.

## Comments

The comment field is entirely for your own convenience. Use it for any form of documentation that is useful to you.

## Event Logging

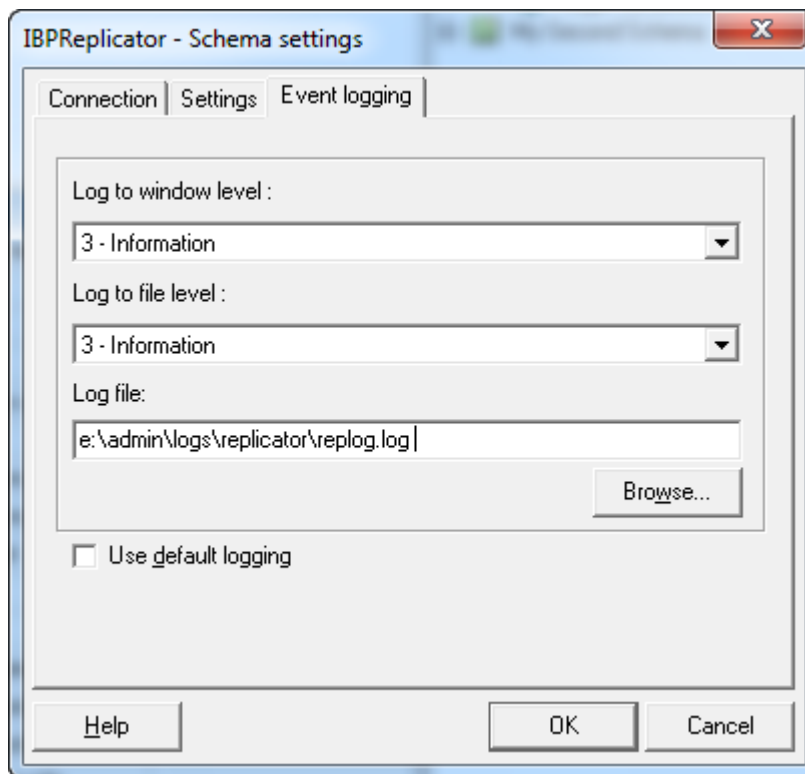
Use the default logging strategy, or specify a specific logging strategy for this source target pair. A newly configured source database is initialised to use the server defaults:



Default logging causes the run–time replication log to be named replication.log and to be written to default locations:

- On Windows, replicate.log is written to c:\Programs and Settings\All Users\Application Data\IBPReplicator if a custom location was not defined for it
- On Linux, replicate.log is written to /var/log if Replicator runs as a service; otherwise it is written to the current working directory

If you are defining multiple replication schemas, you might wish to have the server log each one to its own log file. In that case, check off Use Default Logging and enter a path that exists on the filesystem of the IBP Replicator host machine and, if needed, a different log file name that can be identified easily by the system administrator:



Click OK to update the details of the source database node.

The commands for creating a removing system objects are discussed in [Creating Source System Objects](#).

Each schema can only have one source database, so you need to define a schema for each of your replications. A source database in one schema can be a target database in another schema, and a single source can replicate to multiple targets.

## An ODBC Datasource as a Source Database?

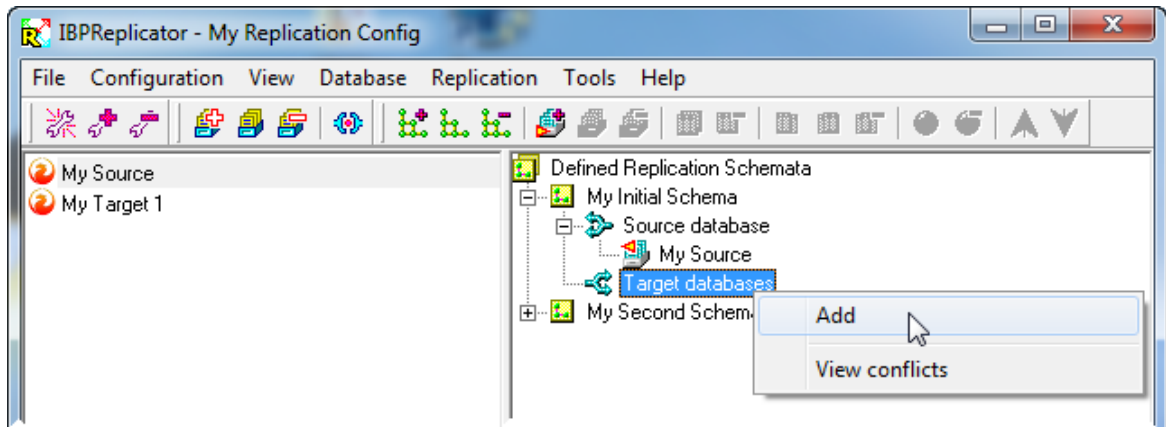
An ODBC datasource cannot be specified as a Source database because there is no way for the Replication Manager to know or access the database engine in order to create the system objects required for a replication.



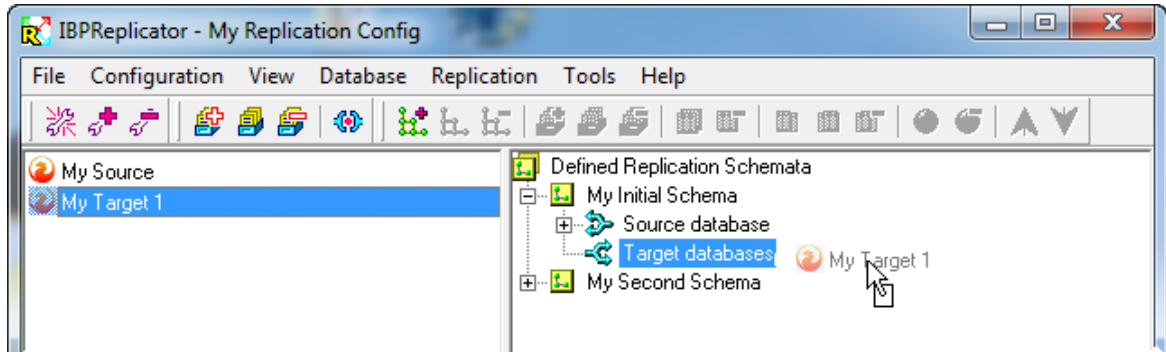
## 7.3 Choosing the Target Database

A target database is a database to which replicated data is sent. When identical data is to be sent to multiple target databases it is possible to define an initial target database and then copy (clone) that definition and apply it to other target databases.

To define a target database, right-click the Target databases node and select the [Add Target](#) task.



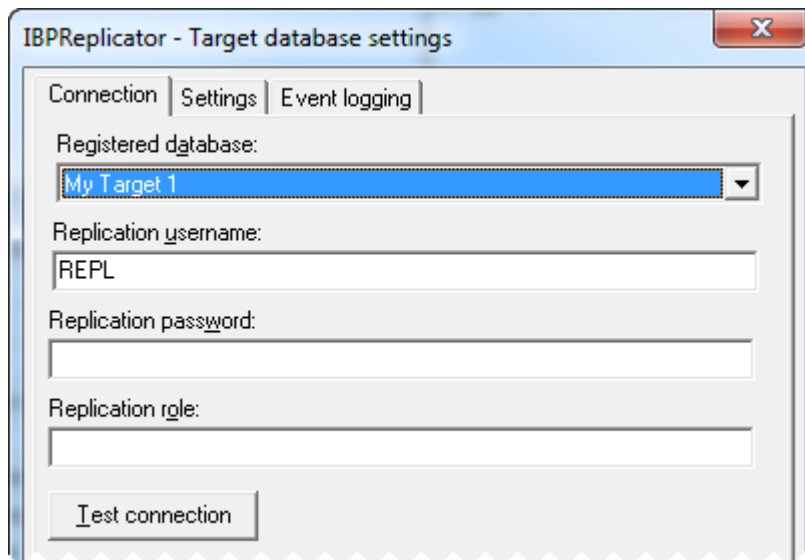
Alternatively, you could drag the target database from the list of registered databases in the left pane over to the Target databases node:



or you could click the Add Target Database button 

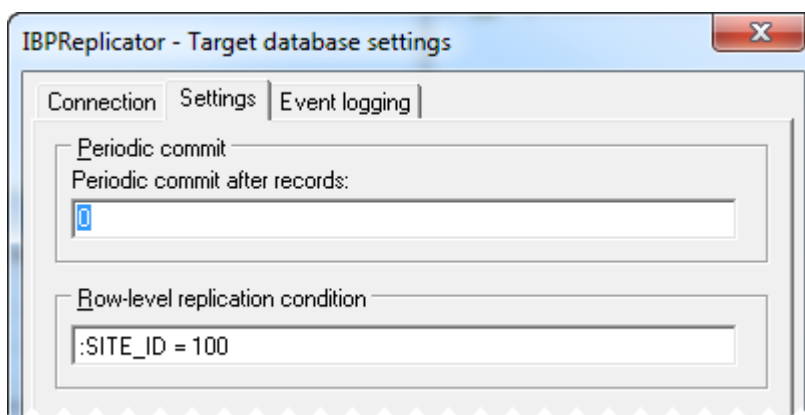
## Target Database Settings

### Connection Tab



- If necessary, drop down the list of registered databases and choose the one that is to be the target database.
- Supply the username, password and, if used, the role to be used by the replication server when it connects to the target database. The name of the "replication user" does not have to be the SYSDBA or that of an Administrator of the operating system. The default user provided in the dialog is REPL, which can be used with a suitable role for any replication and must be used if you plan to implement bi-directional or n-way replication.

### Settings Tab



### Periodic commit

Ordinarily IBP Replicator commits changes once per replication, after all the logged changes have been successfully replicated to the target database. If a network error occurs during replication, all of the changes will be rolled back and IBP Replicator will try again.

If the network connection is unreliable, or the number of changes to be replicated is large, you can specify for changes to be committed periodically as the replication proceeds.

In the field labelled Periodic commit after records set a positive value representing the number of successfully replicated rows that IBP Replicator is to group and commit together.

**CAVEAT** Don't implement this option unless it is actually necessary. It can impact the general performance of the database over time and breaking a replication into numerous separate transactions breaks the overall atomicity of a replication.

If the conditions are sufficiently poor to necessitate resorting to periodic commit, it recommended that you use offline replication instead.

### Row-level replication condition

This is an optional setting that allows rows to be selected for replication on the basis of some value in the data.

For example, if you have defined a multi-site database with conditional replication in view, you might have a key on some tables that identifies the site when the data in a record pertains to that site and not to other sites in the schema. This setting can define this condition.

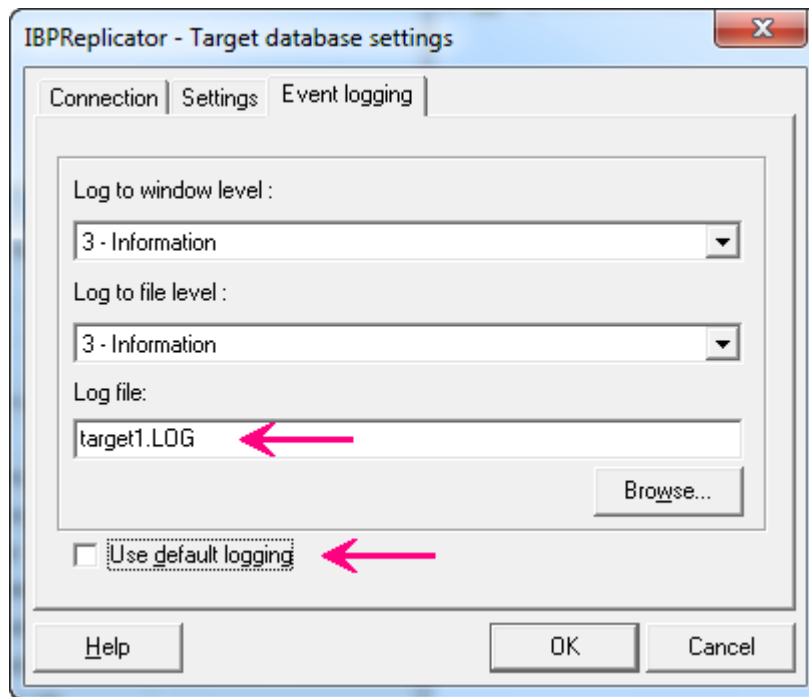
To implement this, any tables that you want to apply this global condition to should include a column with an identifier that is the same for all such tables. Express the condition like a predicate inside an IF() clause.

In the example illustrated the condition :SITE\_ID = 100 will tell IBP Replicator to look out for columns named "SITE\_ID" in the replications it performs to this target and ignore any rows that have any value other than 100 in this column.

Row-level conditioning can also be done at individual table level. For details and more information about the row-level conditioning expressions to use, [look here](#).

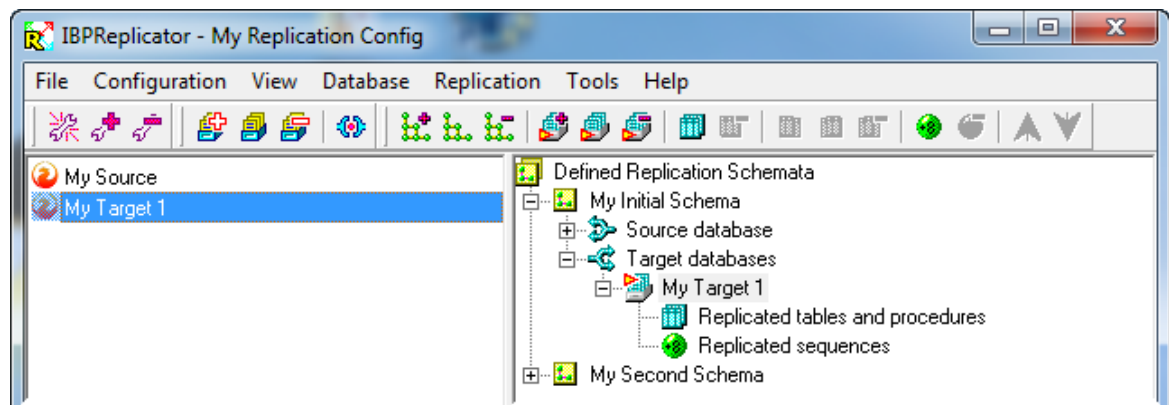
### Event Logging Tab

You can optionally override the default log file and define a specific log file and logging levels for this particular target database:



Check off "Use default logging" and adjust the event logging settings to suit your needs.

When you are done, click OK to save the details of the target database node to the tree. The descriptive name of your target database appears, along with two new nodes—Replicated Tables and Procedures and Replicated Sequences:



In the following sections, you can find out how to

- [select and map the tables and procedures](#) for the replication
- [identify the primary keys](#)
- choose [which columns are to be replicated](#)
- tell the Replication Manager to [create the system objects](#) in the source database

## Accessing the Node Again

Whenever you need to access a particular target database again for these tasks, or you want to edit or remove the target, just select it in the [Target databases](#) node and use one of the [previously described methods](#) to select the task you want.

## Cloning Targets

A schema can support one or multiple target databases for a source database, and different data can be replicated to different targets—see [Choosing replicated tables](#) (next). A separate definition of the tables, primary keys, columns and rows to be replicated is required for each target database involved. That makes it possible to have very fine-grained control for any schema.

It is common for a schema to encompass a number of targets that are going to be replicated with the same, or very similar, data. Because setting up numerous such target databases manually could be repetitious and time-consuming, it is possible to set up one target it and clone its definitions to other targets.

Cloning a target can be done several ways, much as you do with other tasks already discussed. You can

- drag the specific target database node that you want to clone and drop it onto the general target database node or another schema node
- select the specific target database, right click and choose Clone from the context menu
- use [Replication](#) | [Target](#) | [Clone](#) from the replication menu

Once the new target is in the tree, supply the name by selecting from the list of databases that has been registered. Click OK and a copy of the schema defined for the original target database will be created for the cloned target database. Any differences required can be tidied up by editing the cloned definitions.

**TIP** Perform any cloning once you have defined most or all of the mappings for your "base" target.

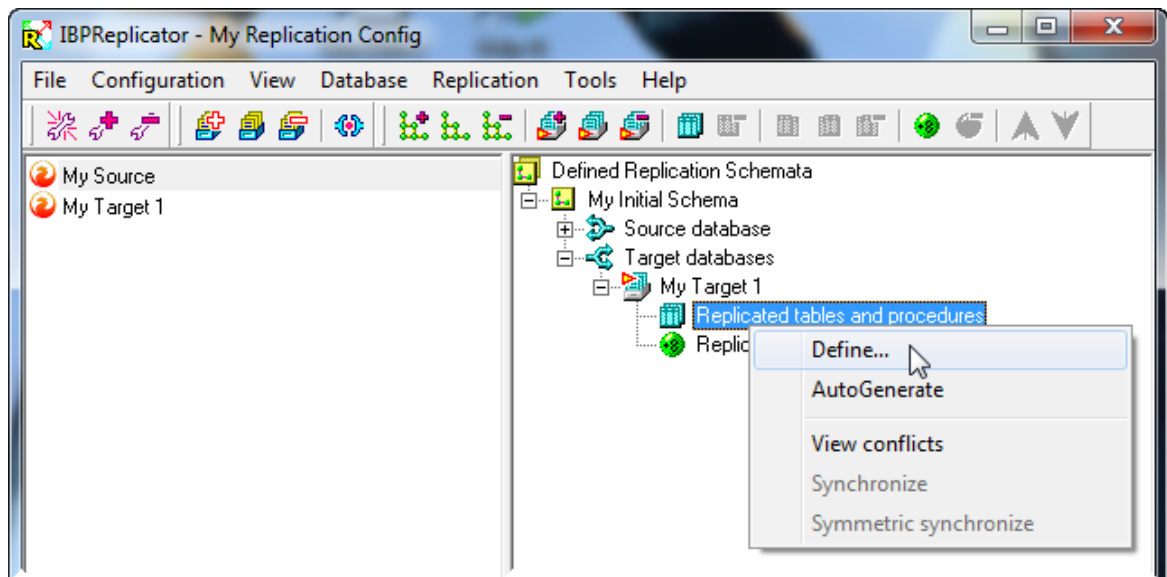
## 7.4 Choosing Replicated Tables

Replication is essentially a matter of identifying data changes in tables in the source database and writing the effects of those changes to tables in one or more target databases. This step is the key one in defining your replication schema: deciding which source tables you want to replicate in your target[s].

It is probably stating the obvious to remind you that you do need to start out with both a source database and a target database that already have at least the relation objects you want to replicate defined in their database schemata!


Don't forget the two essential metadata attributes needed in tables you are going to be

replicating from and to: they must have compatible sets of unique columns and, if you intend to use timestamping for conflict resolution, both the source and target tables must have compatible TIMEFIELD columns that can be referred to for an age reference.



Expand the node for a specific target database and select the Replicated tables and procedures node.

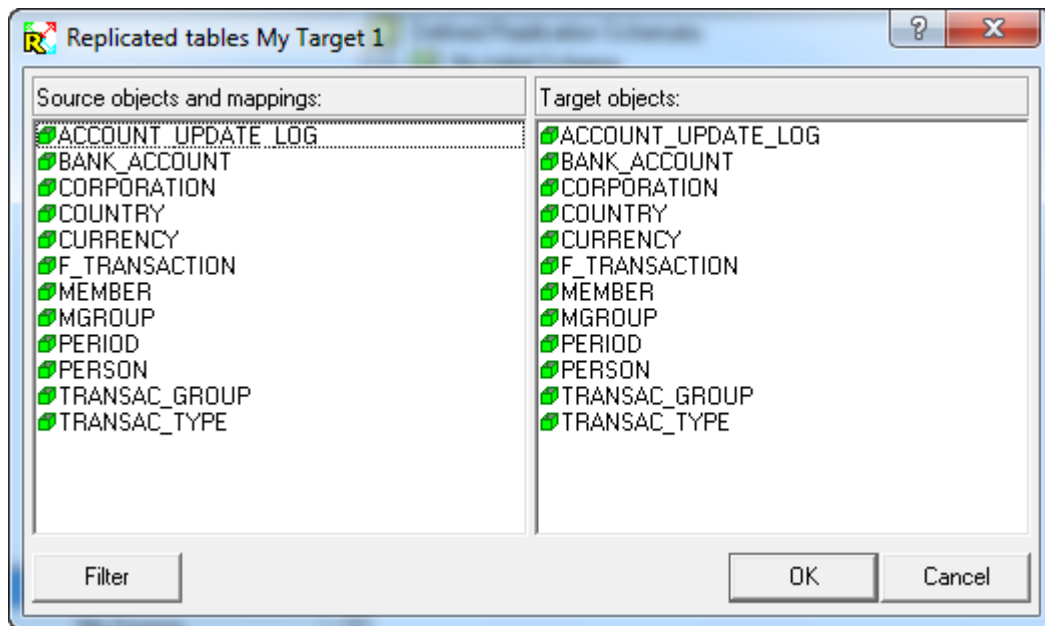
To open the Replicated tables and procedures dialog, either

- right-click and select Define from the context menu; or
- double-click on the Define Tables icon  in the toolbar; or
- select Replication | Tables | Define from the Replication menu.



You can use this selection to access the mappings pane at any time, once you have table objects defined in your schema.

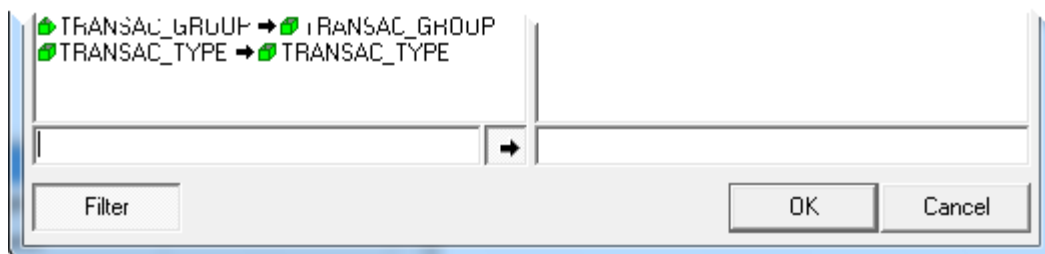
The Mapping Panels



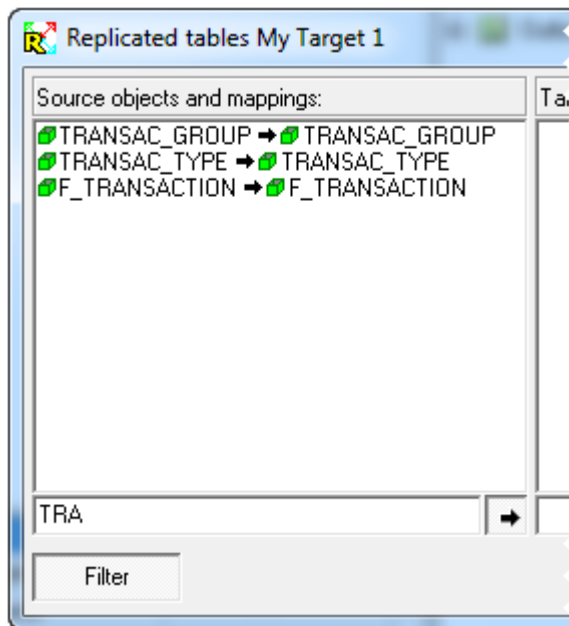
You can use Ctrl-A to select all items in a pane.

## Filtering

If the lists of objects are too big to manage conveniently, you can set a filter on the lists in the two panes. Click the Filter button to display the filter bar below the lists:



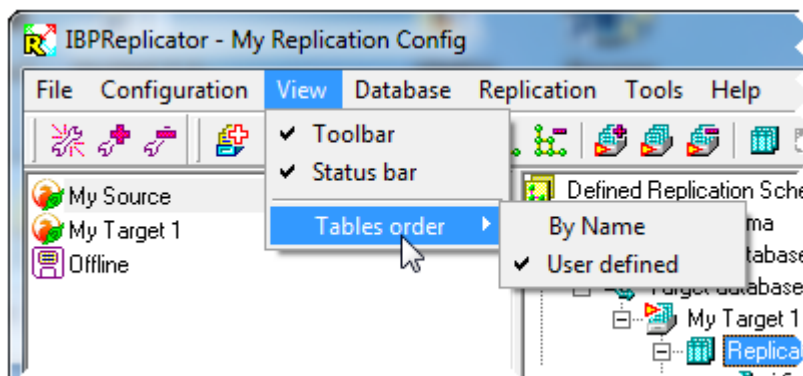
Type in letters to filter the list down to just names containing those letters. You can aim at single letters or substrings—the filtering takes place as you type:



The excluded items have not been unmapped: they are simply invisible. If you use Ctrl-A or the conventional Shift-click/Ctrl-click to select multiple items, only the visible ones are selected.

## Changing the Table Order

Another trick for managing large lists is to use Tables order option in the View menu of the main screen's toolbar to switch the table ordering from User defined to By name:

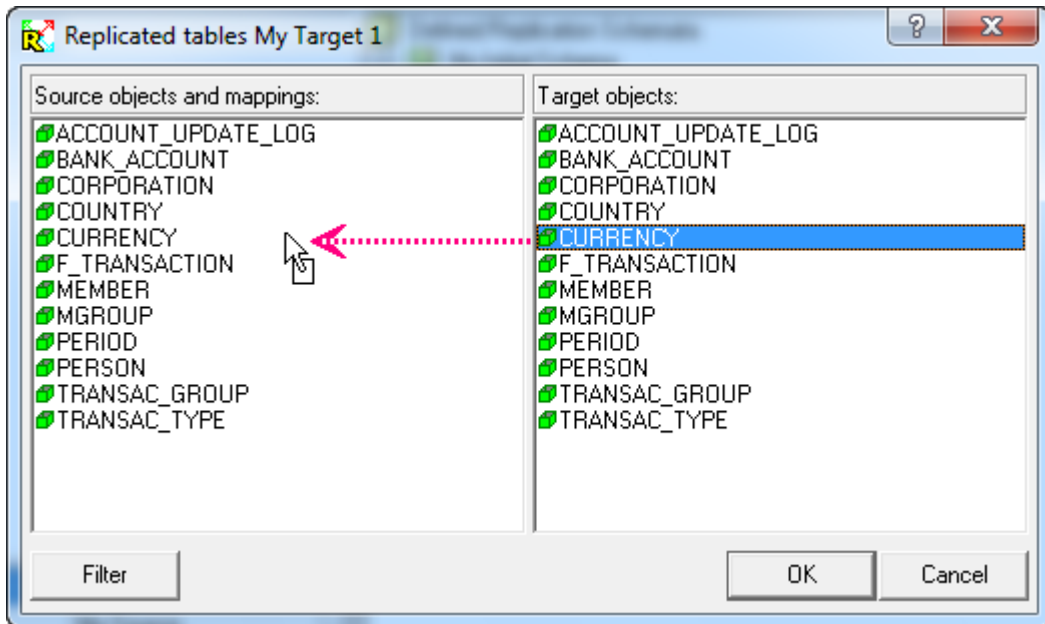


The Tables order setting is persistent. Later, you may need to revert to User defined if it is necessary to set a custom order for synchronizing table objects. For more information about custom ordering for synchronization, refer to the topic [Customising the Order of Tables](#) later in this chapter.

## Mapping the Tables

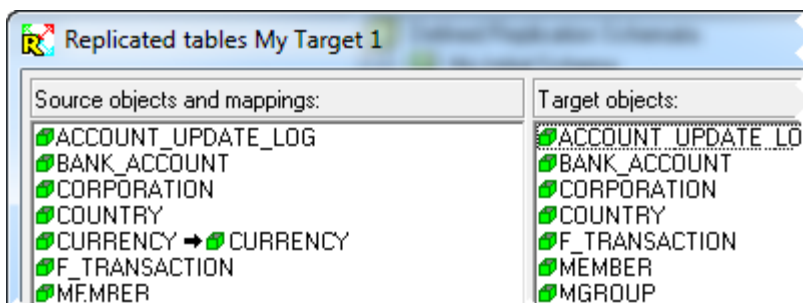
Now you can map the tables in the source database by selecting and dragging table objects from the **Target objects** panel to the **Source objects and mappings** panel:





Stored procedures and views are listed along with tables in the Target pane.

When the target object is dropped onto the source object, the mapping is added and the object disappears from the list of target objects:



- You can multi-select more than one table, view or procedure (using shift-click or ctrl-click, as required) and drag them as whole
- Alternatively, just double-click on the target object to set the mapping(s) automatically between objects with identical names
- Dragging from the source database to the target allows you to break an already established mapping
- Double-clicking on a mapped source table will also break its mapping



Only stand-alone procedures can be replicated to, not procedures in packets. The function must have a minimum of three input parameters and exactly one output parameter, an integer whose "OK" result shall be 0, its conflict result 1 and error for any other value.

Example:

```
REPL_PROC
(K1 INTEGER,
 K2 VARCHAR2,
 D1 DATE,
 D2 CLOB,
 OLD_K1 INTEGER,
 OLD_K2 VARCHAR2,
 OP CHAR,
 RESULT OUT INTEGER)
```

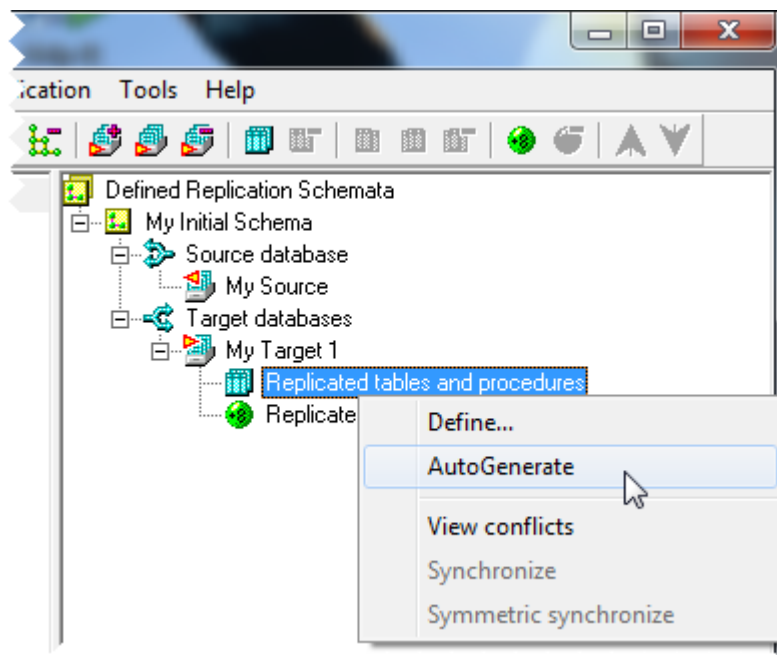
## Mapping in Bulk

The mapping of sequences (generators), tables, primary keys and fields can be done automatically by

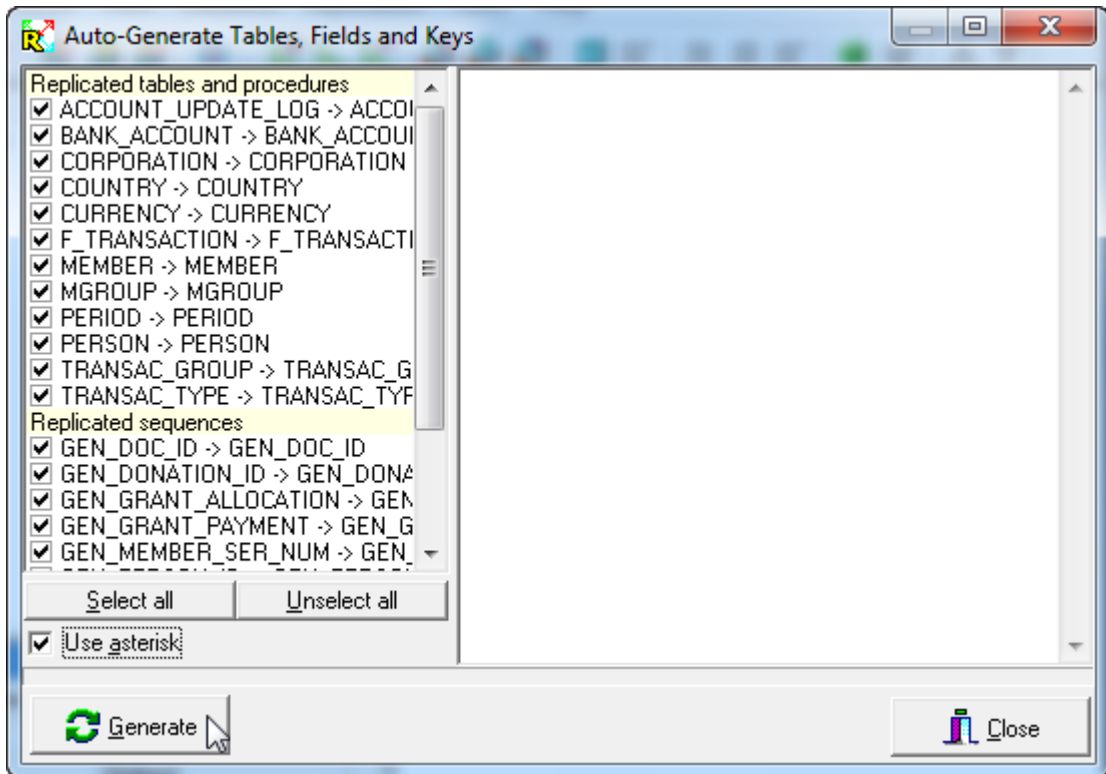
- double clicking on the Generate Tables, Keys and Fields icon; or
- selecting Tables | Autogenerate from the Replication menu or any context menu that offers it

This choice causes all tables, keys and fields in the source database to be automatically selected for mapping to corresponding objects of the same name in the target database. Previous mappings will be respected.

You can multi-select a selective group of tables and autogenerate the mappings and objects for them.



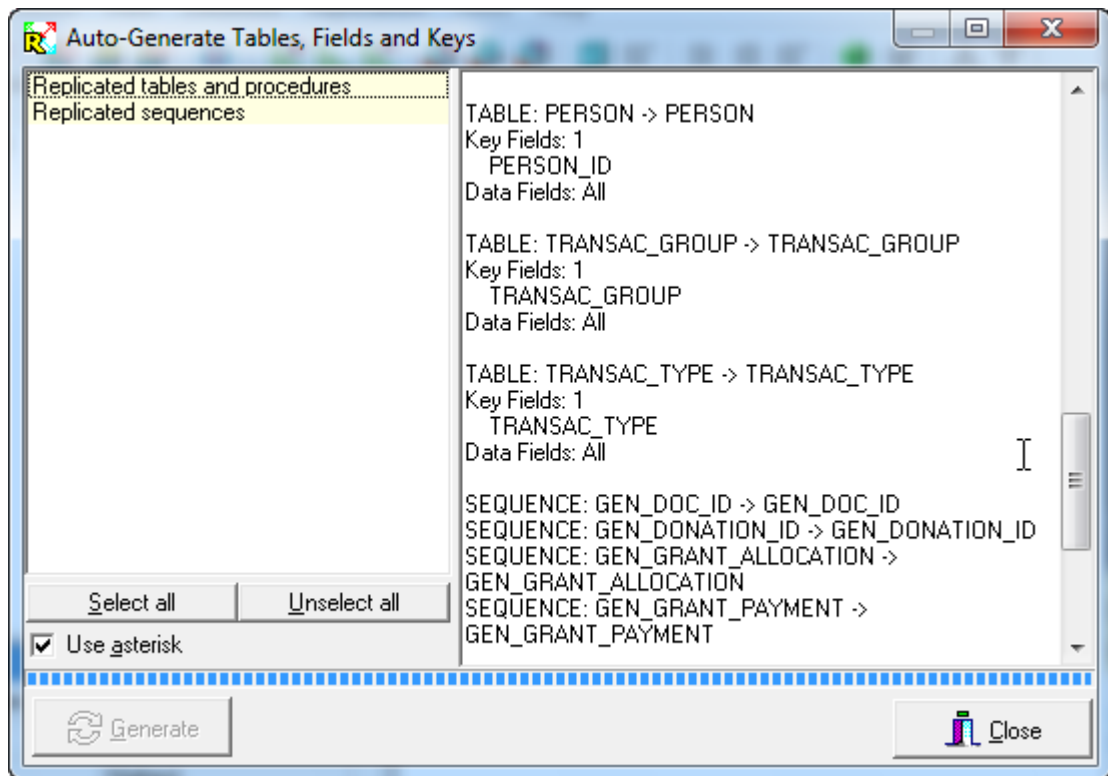
The Autogenerate dialog appears with all of your unmapped objects—tables, primary keys, procedures, views and sequences (generators)—selected with best-guess source–target mappings based on object names:



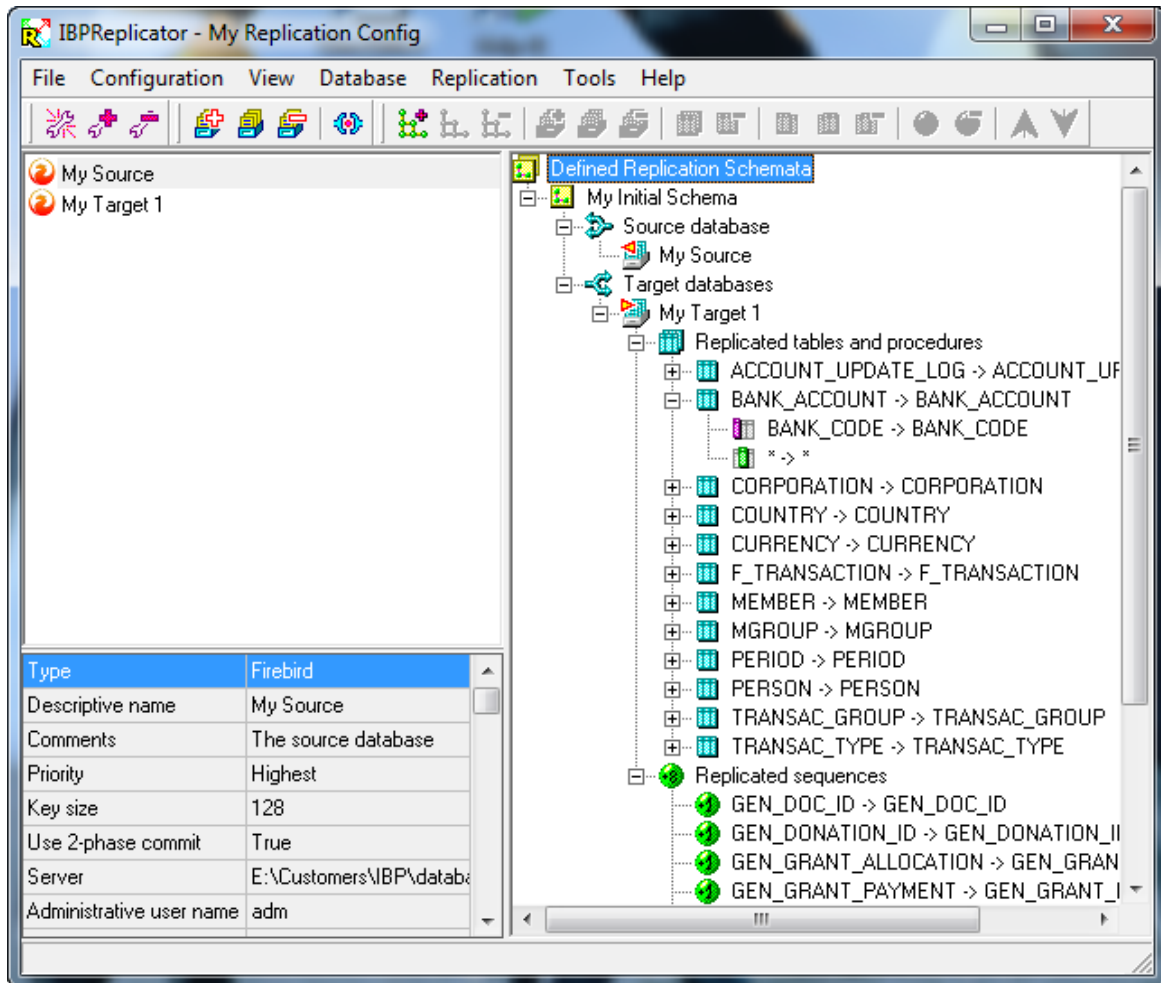
### Use asterisk

Check Use asterisk if you would like to have Replication Manager also map all of the source and target objects automatically at the same time, by mapping column names to column names. Although this is a time-saver if your source and target databases have the same structure, its main purpose is to enable mappings to be easily updated if the table structure should change. In that event, editing of the schema would not be required: it would be enough just to reload the configuration.

Click the Generate button...



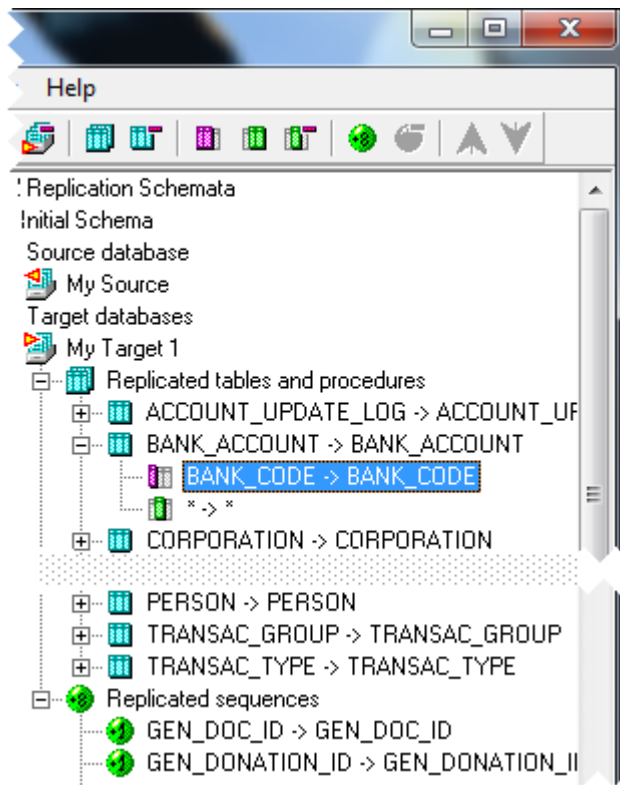
...and it is done! Now, the mappings appear in new nodes in the tree:



Check off any objects you do not want to autogenerate objects for, e.g. because you do not want to replicate them.

### New Toolbar Icons

Notice that icons now appear for adding, editing and removing objects when you select a lower-level node in the tree:



This button takes you to a dialog where you can map or remap the columns involved in primary keys or unique constraints. In the tree, a similar icon marks the node where you can do that task.



This one takes you to the dialog for mapping the non-key fields manually. Similar icons in the tree mark the nodes for each non-key column or for the "asterisk" mapping (as in the screenshot) which does all of the column mappings by matching column names in source and target.



Use this to "unmap" a column pair.



Use this one to work with the mappings of sequences (generators).

## Primary Keys and Column Mappings

The "best guess" mappings for the primary keys have occurred where primary key names and the key fields involved are the same in both source and target or key mappings were deduced from foreign key metadata. The heuristics of this "best-guessing" are not infallible: you should verify the mappings.

Tables that do not yet have their primary keys mapped are marked with a red exclamation point (!).

In tables that have no primary keys, but a matching UNIQUE constraint exists in both tables, that mapping will show instead. You can use the edit tools to tidy up individual key and field mappings, as required.

Once a field has been mapped as a key, or as an element in a multi-column key, it requires no other mapping and is removed from the list of mappable columns. Make sure that all the elements of the key are mapped.

### Heterogeneous Mappings

It is entirely feasible to create heterogeneous table mappings where source and target tables have different table names and even if they have the same table names but different column names. However both tables must have unique sets of same-named columns that identify ("key") the rows to be replicated and the columns being mapped must have compatible data types.

Refer to the Appendix [Supported Data Types and their Compatibilities](#) for information about matching columns with non-identical data types.

### Replicated Sequences (Generators)

The replication can replicate the values of sequences (generators) from source to target.

Be careful with this: replicating sequences in a multi-master scenario will corrupt targets in a heartbeat. If your schema is replicating in such a scenario, you MUST check all sequences (generators) OFF.

### Sequences and ODBC

Because the ODBC interface is, by design, "blind" to the technical capabilities of the datasource, replication of sequences is not available for ODBC targets.

### Removing Mappings

To remove replicated tables select the table node, then either

- right-click and select Remove from the context menu; or
- double-click on the Remove table mapping icon; or
- select Replication | Tables | Remove from the Replication menu

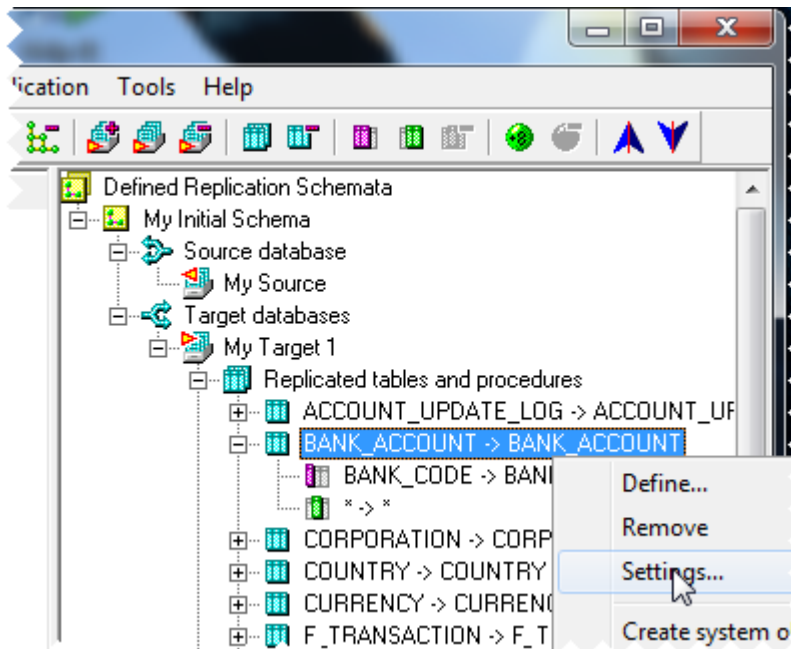
Primary key, column and generator mappings can be similarly removed by selecting the appropriate node and working with the tools activated at that level.

## 7.5 Table Settings

The Table settings dialog allows you to change the default settings for your source-target mappings for an individual table.

To access this dialog, select the table's node in the tree and either

- right click and select Settings; or
- double-click on the Table settings icon; or
- choose Replication | Tables | Settings from the Replication menu.





## The Table Settings Dialog

### Fields on the Table mapping properties Dialog

#### Row-level replication condition

This is a condition that would be inserted into the IF() predicate of a branching statement in a trigger to limit the rows selected.

Given the basic syntax of a PSQL condition:

```
IF (<some_condition>) THEN
BEGIN
...
END
```

you might, for example, replicate only unpaid invoices using

```
IF (NEW.PAID <> 'Y') THEN ...
```

You would pick up the <some\_condition> part of the predicate and enter that into the row level replication condition as

```
:PAID <> 'Y'
```

Note the colon prefixed to the column name.

Although this example is simple, there is no limit to the complexity of the condition: any condition that would be valid in a normal IF(..) statement in PSQL can be used as a replication condition.

### Synchronization enabled

By default, when synchronization is performed, this object is synchronized. Uncheck it only if you want to exclude this object from synchronization.

### Keep statistics

Check this box to use the Monitor tool to review IBP Replicator's performance. By default, it is checked OFF for all tables, to reduce performance overhead by allowing replserver to work with read-only access to the tables in the configuration database.

NOTE :: Monitoring of the table is disabled while Keep Statistics is checked off.

### Conflict resolution strategy

By default, conflicts are resolved using the default method for the schema. If a source and target pair of tables needs a different strategy for resolving conflicts, it can be specified individually, here, by checking the required strategy instead of the one selected from the existing default.

Priority-based Resolution :: the relation in the [higher priority database](#) takes precedence. For example, if a source relation has precedence, conflicts are resolved as follows:

- An update finding no identical key record in the target database is converted into an insert
- An insert finding a record in the target database with an identical key is converted into an update
- A delete finding no identical key record in the target database is ignored.

Master-Slave :: the source object always takes precedence, regardless of the priority settings: resolution will be as described above for the higher-priority database.

Timestamping :: this method determines the newer version of data by comparing server timestamps written on records. It needs a suitable timestamp column defined in both source and target tables. Older rows in the source database will not overwrite newer rows in the target.

Default :: if you later want the strategy for this object to revert to that specified for the schema, select this option.

#### ODBC

When replicating to an ODBC datasource, the replicator cannot distinguish between violations of primary key constraints and others because ODBC does not provide a way to do so. Where

an insert is presented for a record that already exists, it is handled as a foreign key violation in accord with the [setting for foreign key violation](#).

### Time field name

If you are using the time-stamped conflict resolution strategy, both the source and target tables need a timestamp column from which IBP Replicator is to read for comparing the ages of the rows to resolve conflicts. It should be a `TIMESTAMP` type (`DATE` type in IB 5.x) into which a regular Before Insert or Update trigger (separate triggers for InterBase) writes the system timestamp [ `CAST ('NOW' AS TIMESTAMP)` ] whenever an insert or update occurs.

#### IMPORTANT

The time field must have the same name (identifier) in both the source and target tables.

The field name is case-sensitive: it must be entered, without quotes, exactly as it is seen by the server.

Thus, for example, if the identifier was defined with double quotes as "TimeFieldForReplication" then type `TimeFieldForReplication` here. If it was defined without the quotes, e.g., `TimeFieldForReplication`, then type `TIMEFIELDFORREPLICATION` here.

The Time field name is where you name and map this column for the two tables.

### Separator character

Use this if the separator character for the elements of composite primary keys in an individual table need to be different from the [global separator character](#).

### Foreign key violation behavior

If you want a foreign key violation to be treated differently for this relation than the method specified in [Global Settings](#), you can specify it here. Initially, the global setting is selected by default. Choose an alternative behavior as follows:

Error :: causes an error to be written to the log and replication is aborted.

Conflict :: treats the violation as a conflict. The conflict will be written to the manual log table and replication continues.

Ignore :: skips the offending record without logging anything.

Default :: uses the global setting.

### Replicated operations

By default, all operations (insert, update and delete) will be replicated. You can choose to exclude an operation by checking it off.

### Comments

Use this box for any free-form documentation about the object that could be useful to you and others in future.

## 7.6 One-time Synchronization

Synchronization allows the IBP Replicator to get the source and target tables into the state where data matches in both, i.e. a one-time "global" replication.

There are two types:

- asymmetric, where the target table is synchronized from the source table. The target table becomes an exact copy of the source table with IBP Replicator performing all the necessary inserts and deletes. The source "database" for an asymmetric synchronization can be a suite of files previously off-loaded to a Files.. target database. This off-line asymmetric synchronization is described in more detail in [a later chapter](#).
- symmetric, which does not perform any deletes but, instead, inserts any records missing on the target from the source and vice-versa. It also performs updates according to the current conflict resolution strategy.

Synchronization is not replication. It is a one-time operation that can be run for a new or modified schema.

### Preparing for a Synchronization

- Before setting out to perform a synchronization, make certain that no user—including Replicator itself—is working in the databases.
- The table REPL\$LOG (or (REPL\_LOG for InterBase) must be empty.
- It is a good precaution to remove system objects before synchronization, especially in a multi-master schema.
- Deactivating triggers and non-constraint indexes can improve performance.

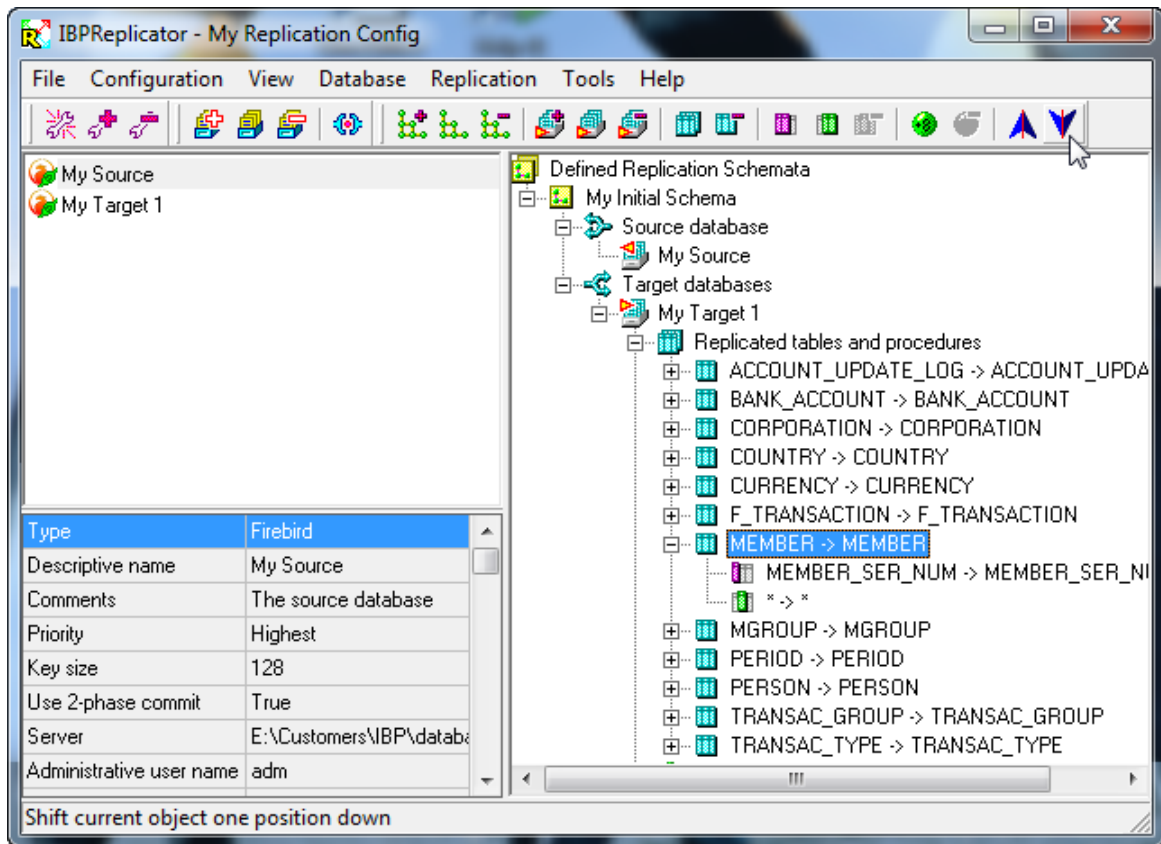
### Customising the Order of Tables

It is not necessary to synchronize tables one-by-one to avoid the dependency conflicts that could occur when a bulk operation follows the default (alphabetical) order that accords with the system tables. You can define your own order for the synchronization to avoid these conflicts and then perform the synchronization for all tables in one hit.

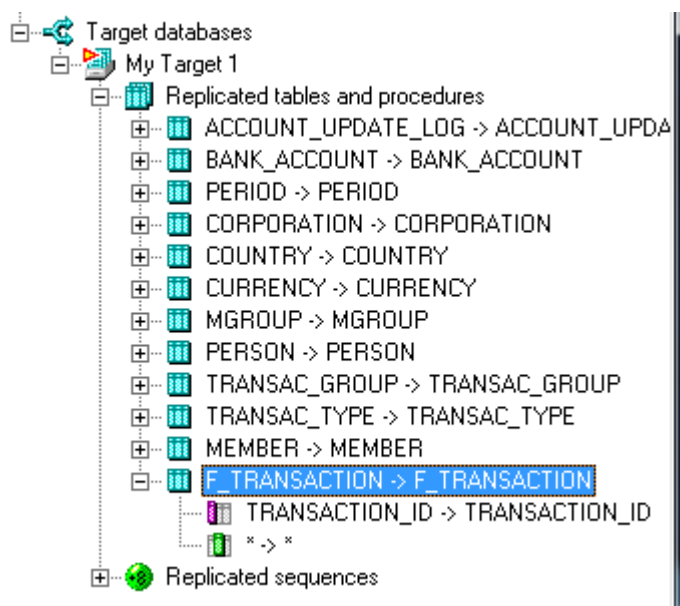


You won't be able to perform this customisation if you have the interface explicitly set up to order the objects alphabetically. Before you start, you will need to use the View>Tables order option in the toolbar of the main display to switch to "User defined" ordering.

Select the Replicated Tables and Procedures node of the target database and select a table that you want to move. Then click either the Up or the Down button in the toolbar, to shift it up or down one position:



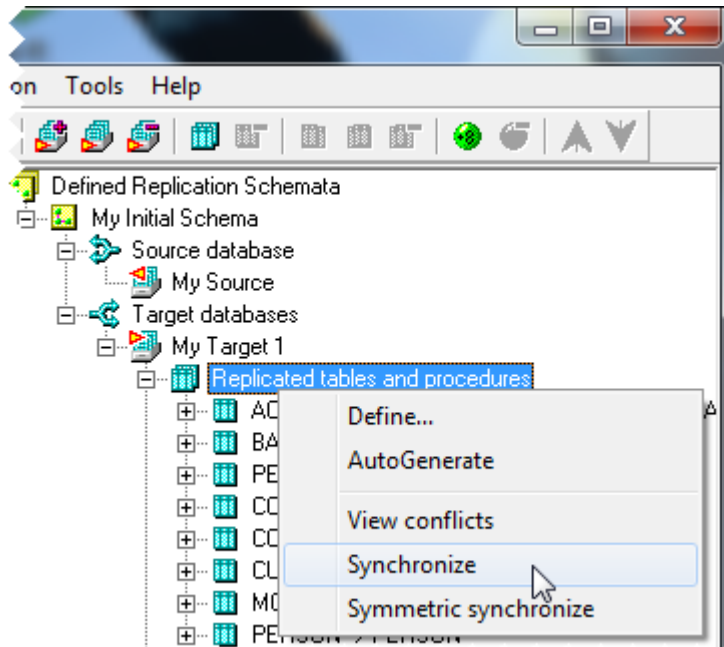
In this example, we want to move MEMBER down, because it has dependencies on nearly all of the other tables. F\_TRANSACTION has to go to the very last position because it depends on all of the other tables:



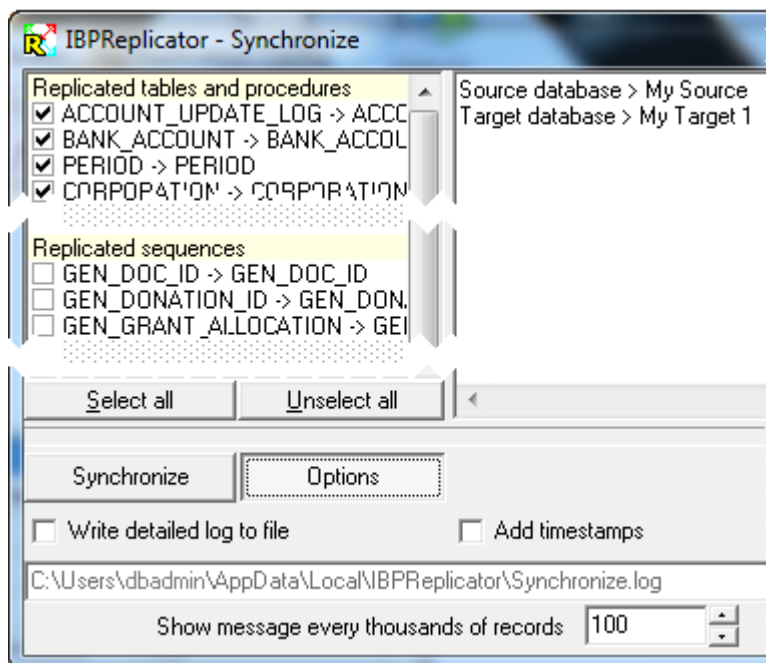
Simply continue to shift tables and procedures down and up until you are happy that you have taken care of the dependency issues.

### Performing a Global Synchronization

Select the database node of the target you want to synchronize with the source, right-click and select the type of synchronization you want:



The dialog window shows all of the synchronizations checked for the mapped tables (and Oracle procedures). If you want to synchronize sequences (generators) as well, check them on manually. You can also check off any objects you don't want synchronized.



## Options

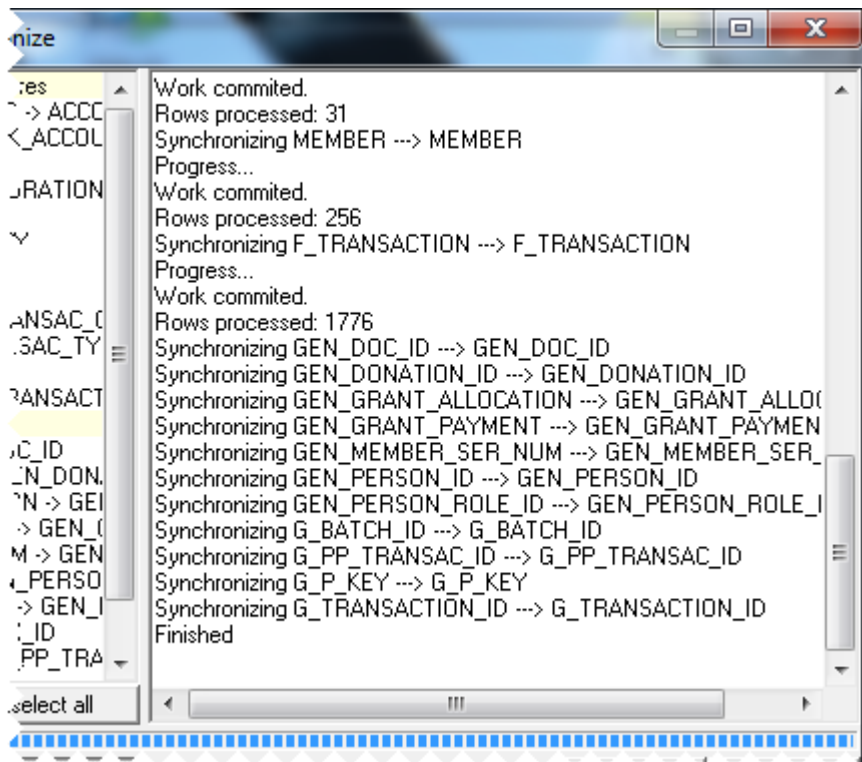
**Write detailed log to file ::** Check this on if you want a log of the synchronization. The field beneath shows the default location for logs on your system, according to your global or schema-level setting. When logging is selected, this field becomes writable and you can specify a custom location for this log if you wish.

Make sure you enter a file path that already exists.

**Add timestamps ::** Select this if you want the screen and log messages to include timestamps.

**Show message every ... ::** the process will return a progress message to the window every 100,000 records. Change this if you want messages to be more or less frequent. Numbers are in thousands.

A monitor window will pop up and display the synchronization as it happens. The window will fill up very fast but you can scroll back to check what has been done:



## Single-table Synchronization

If you just want to synchronize an individual table, select the table in the Replicated Tables and Procedures node of the tree, right-click and select a synchronization option from the context menu. The same dialog appears but, this time, with just that table selected.

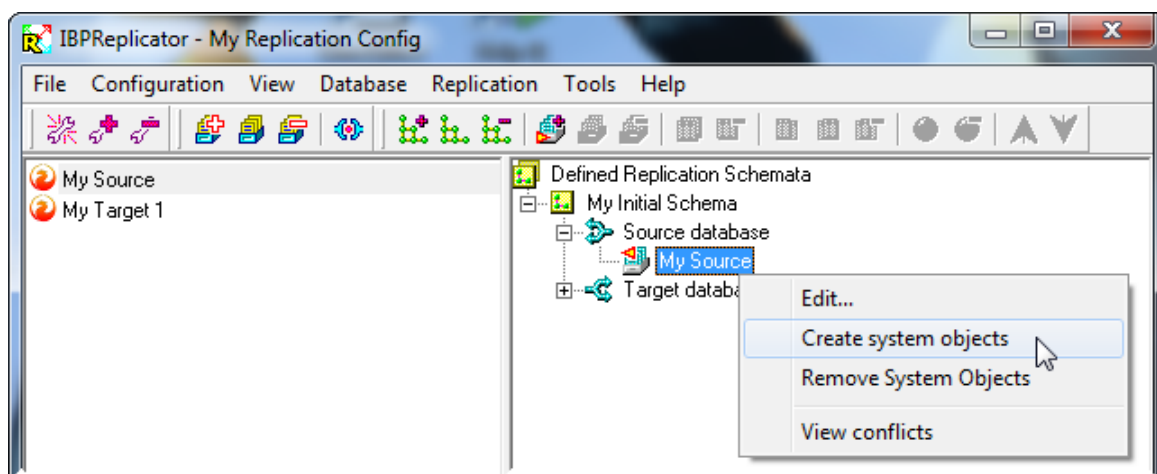
## 7.7 Creating Source System Objects

The final step, once the rest of the replication schema has been defined, is to ask the Replication Manager to create the tables, triggers, etc., in the source database that will be used for the replication operations. These structures are referred to as System Objects.

It is recommended that you perform Create system objects when you are the exclusive user on the source database, particularly if the source database is on a Firebird Classic server. The triggers created by IBP Replicator will be seen by other database users when the database metadata cache is refreshed. On Classic, each client has its own cache and won't see the changes until next time that user logs in. It is less of an issue with Superserver, since the cache is shared by all clients and none will see the changes until after all users have logged out.

As usual, you have several ways to make this request. This time, you will be operating on the Source database, so select it in the Replication tree and do one of the following:

- right click on the Source database node and select **Create System Objects** from the context menu
- choose Replication | Source | Create system objects from the menu

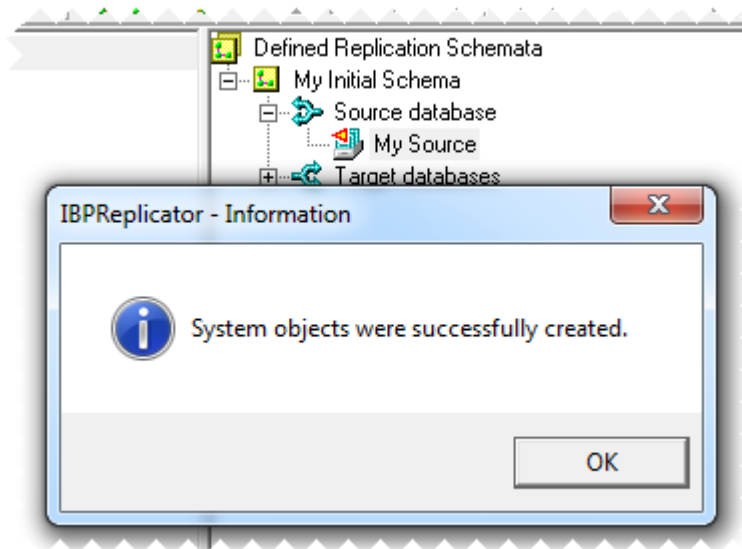


**TIP** You can create all of the system objects globally by using the main menu option.

If there are any problems—for example, missing primary key assignments—you will see an error message and no system objects will be created in this operation. Since each successful operation is committed, previously created system objects will persist. Correct the source of the current exception and try again.

If all goes well, you should see the success message:





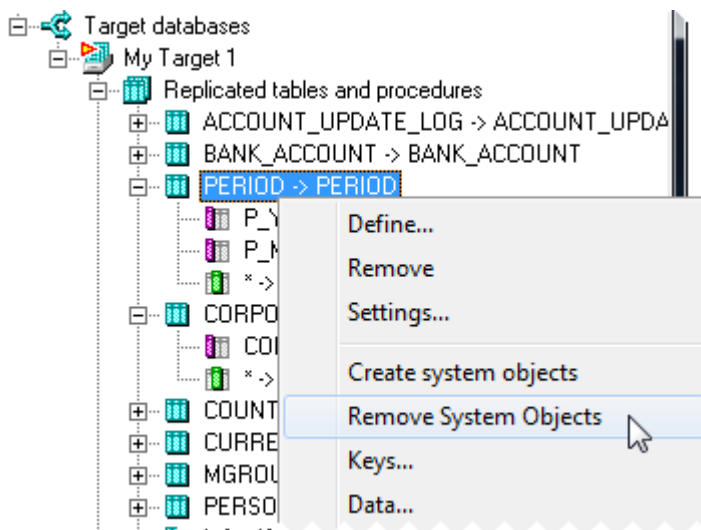
## Creating System Objects from a Target Node

You can also create the system objects when a target database node, or even a target table, is selected. In that case, system objects will be created in the source database for just that selected database (or table, if initiated from a table node).

## Removing System Objects

Select the source database in the Replication tree and double-click on the Remove system objects icon to remove all the IBP Replicator defined system objects from a source database.

You can also choose to remove just one or more system objects selectively. Select the object in the Target Database node and then right-click>Remove System Objects.



Click OK to confirm and a message will inform you that the object has been excluded.

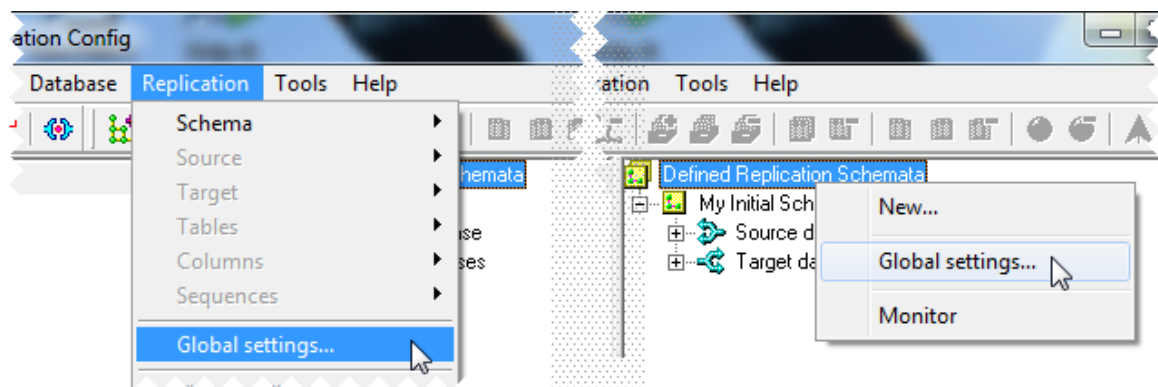
## 7.8 Customising the Global Settings

The global settings are stored in the configuration database when it is created. Although optional, custom global settings will be useful if you are running multiple replications with the same settings. You can configure settings for any individual schema with its own settings if you like: it will overrule the server's global settings for that schema only.

Changing the global settings does not change the settings of schemata that have already been defined; only newly defined schemata will have their settings initialized to the new values.

For schemata you are going to define in future, custom default behaviour can be set up to

- control the way foreign key violations are handled
- set the conflict resolution strategy
- set the levels of logging you want for replication events
- configure email notification of failed or interrupted replications



To begin modifying any of the global default settings, either:

1. Access them from the menu via Replication | Global settings... OR
2. Right click on the root node labelled Defined Replication Schemata and select Global settings... from the context menu

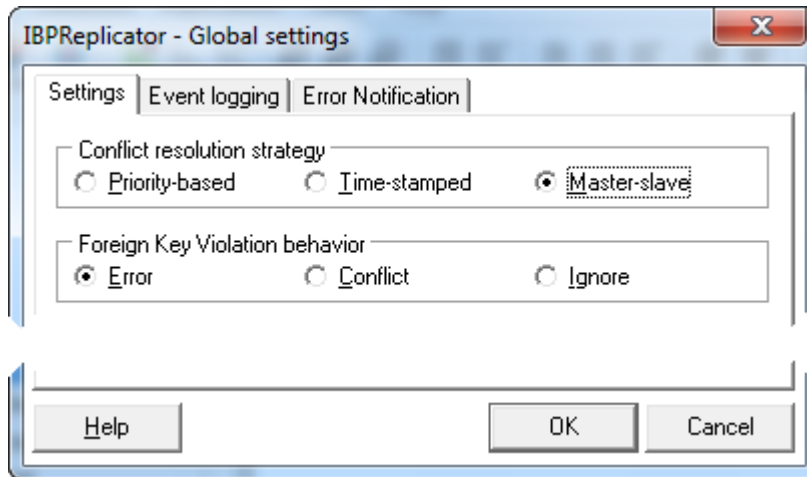
### Global Settings

If you are upgrading from an old version of IBP Replicator, you may notice that the default replication scheduling attributes no longer appear in this display. It disappeared in v.2.5, when replication scheduling was moved into a new engine within Replication Manager. Please refer to the [revised notes about the Scheduler](#).

### Conflict resolution strategy

When replicating new data from a source database to a target database where the

primary key for the replicated row already exists, IBP Replicator can be defaulted to use a specific strategy to resolve conflicts between source and target records:



**Priority-based:** When you register a database, you can give it a priority so that when conflicts occur, the database with the higher priority number takes precedence. If the source database has the higher precedence, the conflicted row is over-written on the target, but if the target has the higher priority, the conflicted row is preserved in the target and the replicated row is queued for manual conflict resolution.

**Time-stamped:** The more recent of the conflicting rows is preserved. This strategy requires the presence of matching timestamp fields in both source and target rows.

**Master-slave:** The source database's row always replaces the conflicting row in the target.

Conflicts that cannot be resolved directly by the conflict resolution strategy are logged for manual intervention using the Conflicts tool.

### Foreign Key Violation behavior

The default behaviour when the Replicator encounters a Foreign Key Violation is to log an error in the replication.log and abort the replication. You can configure different behaviour:

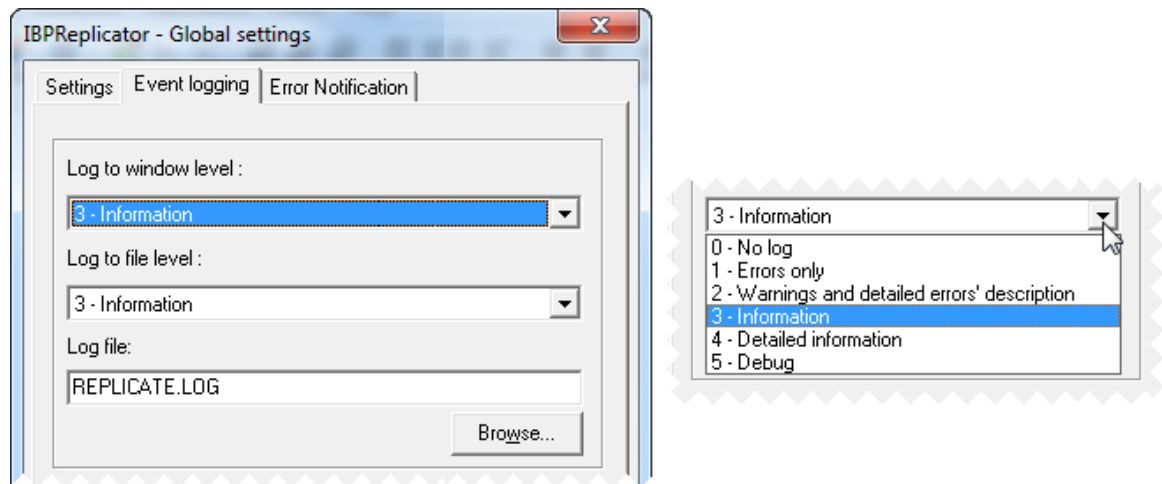
**Conflict:** Treat it as a conflict and write the replication record to the manual log table

**Ignore:** Skip over the offending record without logging anything



You can override the foreign key violation behaviour for specific tables in specific schemas. For more information, see the topic [Foreign key violation behavior](#) in Table Settings.

### Event Logging



Each configuration database can have its own error/information logging settings and log file. Allowing IBP Replicator to record its activity in either a window and/or on a log file on disk. The user defines what is written (level) and where logs are written.

The logging levels are

- 0: No Logging
- 1: Errors Only
- 2: Warnings and Detailed Error Description
- 3: Information
- 4: Detailed Information
- 5: Debug

Logs can be written to a disk file in either the default path (which depends on platform and run mode) or in a different one, which you specify.

The default log file, named Replicate.log, is placed as follows:

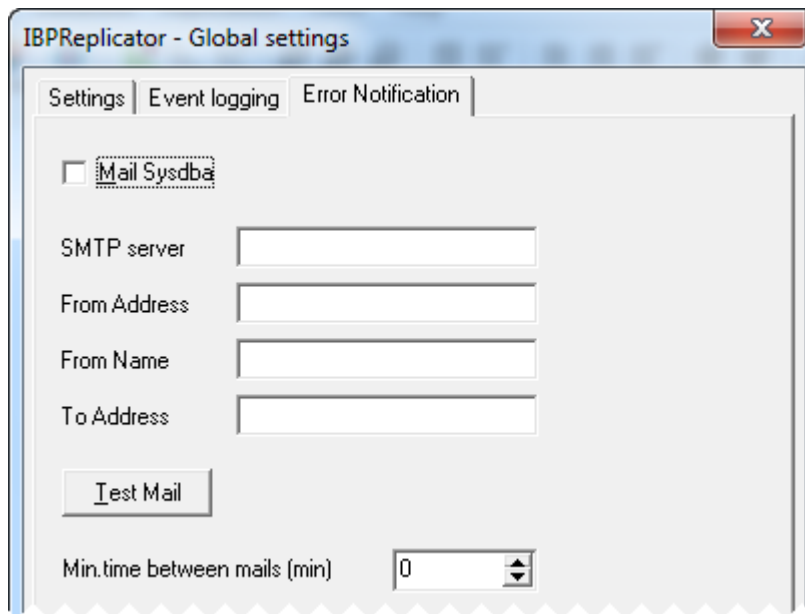
- ☐ Linux: in /var/log if Replicator is running in daemon mode; in the current directory if in application mode
- ☐ Windows: in the IBPReplicator subfolder of the "All Users" Application Data folder

If you are setting up more than one replication you will probably want to specify non-default paths and/or file names for each one's log.

### Error Notification by Email

To enable email notification, make sure the Mail Sysdba checkbox is checked. An email notification will be sent to the designated TO address each time Replicator encounters an error.

Once you have set up the notification details, click the Test Mail button to verify that it works.



To prevent your mailbox from overflow, you can set a minimum interval (in minutes) between email notifications. If it is set, Replication Server will swallow mail messages for the specified number of minutes after successfully sending one. The default setting of 0 minutes disables this protection.



# Chapter 8

## Chapter 8 - Managing Replication

## Chapter 8 - Managing Replication

The topics in this chapter concern the actual running of your replication services and the tools and utilities provided to do so.

On Windows, you will (ideally) run IBP Replicator as a service. On Windows, the installer will set everything up to run the service according to your preferences.

For running IBP Replicator as an application, a graphical interface is available from the Start Menu to set up how you want it to run. Some control aspects can also be run from the [command shell](#), either directly or in a .bat file.

If you need to control IBP Replicator wholly in Linux, only a command-line interface is available currently for controlling the services.

### 8.1 Windows Operation

#### The Windows Executable

The executable that provides the replication services on Windows is ReplServer.exe. It incorporates the scheduler.

#### Essentials Before the Replication Server Will Run

Make sure that

- your configuration is in a sufficiently complete state that replication can occur
- your server and replicant licences are installed and that you have enough replicants to perform the configured replication[s]
- the source and target databases have the necessary permissions granted for the REPL user or, if you are using a role (e.g. REPL\_ADMIN as in our examples) that the role has been created and the permissions granted. (And remember to grant the role to REPL!)
- you have set the default configuration database—otherwise it won't be found and the server program will refuse to start!
- for Firebird or InterBase sources and targets, the respective servers are running



The executable Replserver-debug.exe in the package is intended for troubleshooting and provides extra levels of logging.

#### Running IBP Replicator

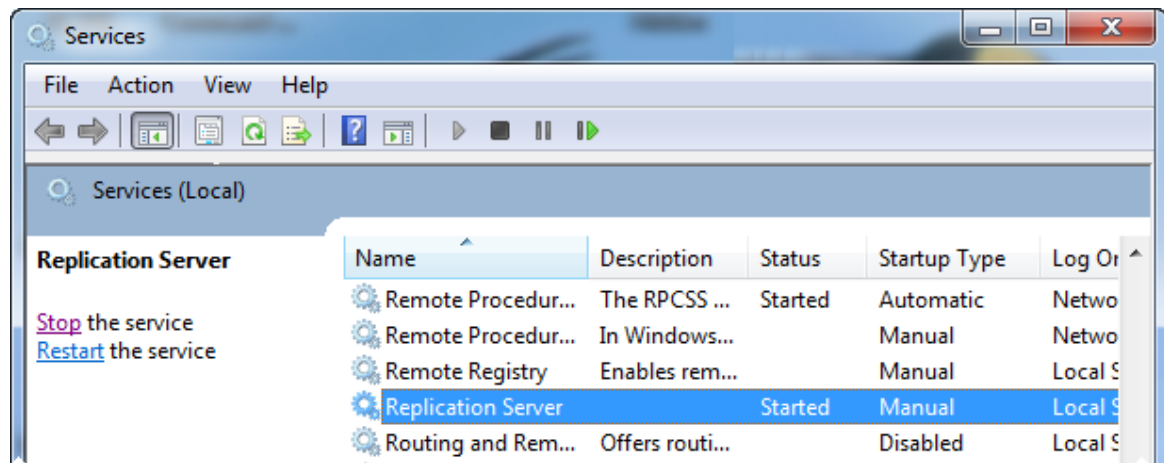
The replication server program can run on Windows as a service or as an application.

##### As a Service

As a rule, on the Windows service-capable platforms—Windows NT, 2000, XP Professional and the 200X Server versions—you will be running replication as a service.

### Starting the Service

The Windows installer should have installed the service for you. You can check this by inspecting the Services display by way of the administration tools on your server:



- If you elected to have the service start automatically, it will have the Startup Type "Automatic" and should be running ("Started") unless someone has stopped it.
- If you elected to have it start manually, or it has been stopped (shut down), you can start it the same way as you would start any other service, i.e. by right-clicking on its entry in the list and selecting "Start" or "Restart", or by selecting its property sheet and manipulating it from there.

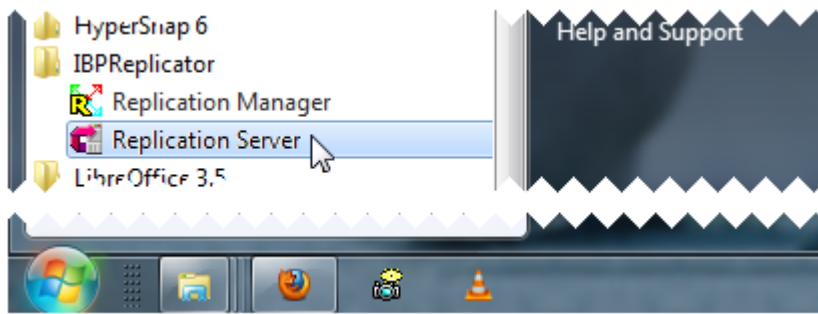
### As an Application

Resources permitting, it is possible to run ReplServer.exe as a stand-alone application. It can be started from the supplied graphical interface or from the command-line (including a batch file).

- On Windows 9x and ME you have no alternative, since these platforms do not support services.
- On the other platforms, you can run it as an application to perform occasional replication runs, that can use an alternative configuration database, if necessary.

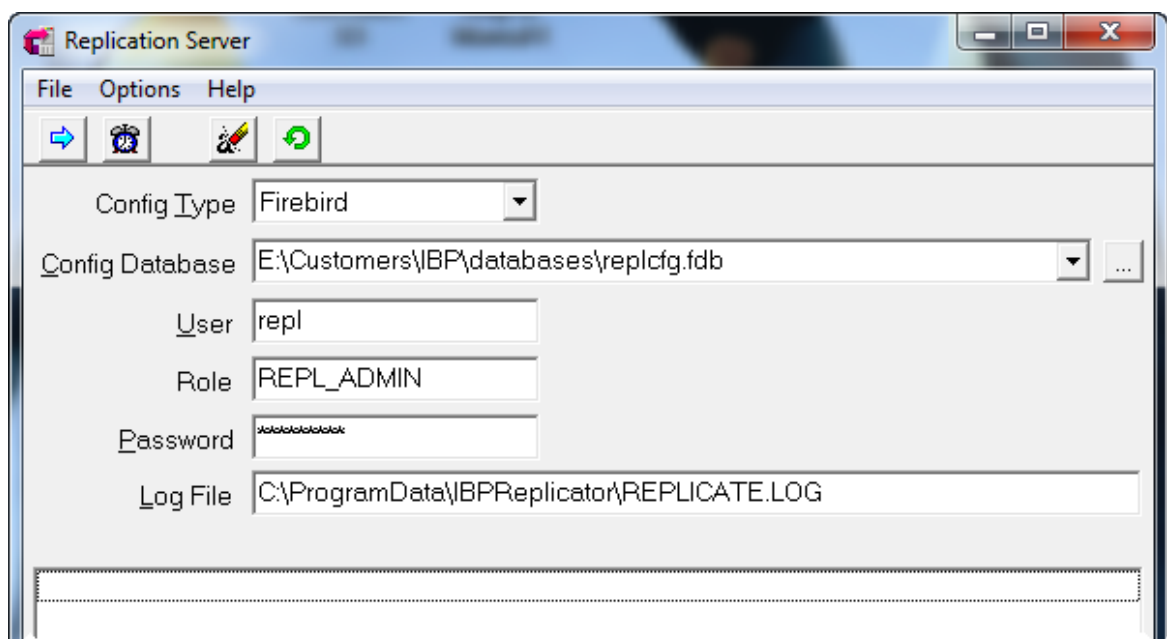
To start ReplServer.exe as an application from the graphical interface, click its icon in the Start Menu:





### The Application Interface

The graphical interface appears, populated with the parameters for the default replication that it has started:



If you watch long enough, the log window will fill up with an account of the replication that is going on. If it is doing what you want, just minimise the display and let 'er rip. You can maximise it and look in on your replication at your pleasure:

```

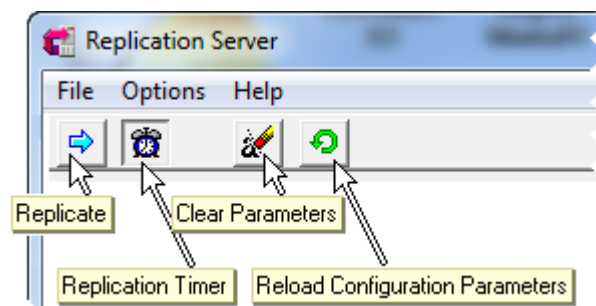
09/26/12 15:24:35: ***** Start Replication *****
Replicate My Initial Schema, My Source -> My Target 1
09/26/12 15:24:36: Connected to DB: E:\Customers\IBP\databases\source.fdb
09/26/12 15:24:37: Connected to DB: babe:d:\databases\IBP\target.fdb
09/26/12 15:24:38: Disconnected from DB: babe:d:\databases\IBP\target.fdb
09/26/12 15:24:38: Disconnected from DB: E:\Customers\IBP\databases\source.fdb
TOTAL STATISTICS:
0 Hours, 0 Minutes, 0.187 Seconds
09/26/12 15:24:38: ***** Replication Finished *****

```

Idle 0

### The Toolbar Buttons

You can use the toolbar buttons to load a different set of parameters and run a different replication:



In the example, the Timer button is depressed, indicating that the configuration is doing (or will do) a timed replication. If you are going to do a different replication as a one-off, you would click this button off.

Click the Clear Parameters button. Note that the application will start to nag at you to enter something if you wait around too long! Proceed to make your alternative entries:

- In the Config Database field, browse for (or type in) the full connection path to an alternative configuration database
- Fill in the User and Password fields and, if needed, the Role
- Change the default log to a different valid path and filename if you wish

Click the Reload button to load the new parameters.

Click the Replicate button to start replicating (or to do a one-off replication, if the Timer button is off).

## Managing the Server

Once the server is running, you can stop and restart it. How you do so depends on whether it is running as an application or a service.

### To stop the server when it is running as a service

Either:

- Method 1: Open the Replication Manager and connect to the appropriate configuration database. Execute (from the Menu)  
Replication|Tools|Notify Server|Shutdown Server or press Ctrl-U.  
For InterBase or Firebird, the events mechanism must be working.
- Method 2: Stop the service from the Services applet in the Administration Tools

Note also that you can stop and restart the service from the Services applet by choosing Restart.

### To stop the server when it is running as an application

Simply close (Exit) the application. You can do this either

- from the menu of the graphical interface program; or
- by selecting it in the Windows Task Manager list and clicking End Task

## 8.1.1 Command-Line Operation

The server program on Windows (ReplServer.exe) can be run as an application from the command line using option switches.

### Program Switches for ReplServer.exe

Switch and Argument	Effect	Comment
--application or -a	Run the server as an application	The default is for it to run as a service—recommended if available. However, you can run a non-default replication as an application whenever needed. It is probably a good idea to avoid running two replications simultaneously, though, in case it causes lock conflicts.
-t=<type>	Configuration database type: <type> interbase   firebird   oracle	Case-insensitive
--min or	Run (with -a) minimised	Minimises the shell window to the Task Bar

-m		
--go	Start replication automatically and quit after replication is complete	Convenient for running replication from another application/script
--log_to_file=n	n= 1: Errors Only 2: Warnings and Detailed Error Description 3: Information 4: Detailed Information 5: Debug 6 & 7: Thorough debug, applicable to debug builds only	Default is 2 (warnings and detailed error description).  If a support consultant asks you for a "debug log", level 7 is the one to generate. Logging levels above 5 apply to a debug build, which is not part of the normal software distribution.
--log_to_window=n	Values for n as above. Echoes the log to the command shell	Default is 2 (warnings and detailed error description).
--log-file file-path or -l file-path	Provides an initial log file for use between server start and configuration load	Enables troubleshooting for cases where ReplServer.exe has problems loading the configuration but the default location for the log file is unavailable for some reason
-c=<cfg>	<cfg> is the full connection string for a specified configuration database	Can be used to run an alternatively configured replication
-p=<password>	<password> may be needed if the config database specified is not the default one and the access password is different to that of the default config DB	
-u=<username>	<username> if needed for the situation described above	
-r=<role>	<role> if needed for the situation described above	
--debug	Run in debug mode	This is logging level 7, the most verbose level. If a support consultant asks you for a "debug log", this is the one to generate.
--help or -h	Prints succinct help for these switches	

## Installation Switches

In case you need to know the installation switches for making your own service installer script, they are as follows:

/install	Install the replicator service
/uninstall	Uninstall the replicator service. (This uninstalls the service, not the software!)
/auto	Have the service start automatically
/dep=<service name>	Used for passing the <service name> of a service that the replication service depends on

## 8.2 Linux Operation

### The Linux Executable

In this version there is just one executable—replserver—for running IBP Replicator on a Linux server. If you ran a previous version of IBP Replicator on Linux, some things have changed.

- replserver can now be run as a service under chkconfig control, or as an application launched with command-line switches
- the command-line switches are in line with those used with the Windows version, although not all of the Windows switches apply to Linux
- the Replicator program causes a pid file to be created, which is used as a locking mechanism to prevent another instance of the program from being started
- Refer to the [list of command-line switches](#) for details of the options available. You can also just run `./replserver --help` for a succinct list of these options



The executable Replserver-debug in the package is intended for troubleshooting and provides extra levels of logging.

## Replication Management on Linux

All management tasks for the Linux server can be done either through the daemon interface or via direct commands with switches. Process instances not under daemon control are stopped using the Linux kill/killall commands with signals.

NOTE IF you do not have a Windows machine available for defining your schemas and installing licences, it is possible to run the Windows ReplMgr.exe program in Wine, the Linux Windows emulation (compatibility) system.

## Topics

[Installing and Configuring on Linux](#)

[Controlling replserver Processes](#)

[Loading Licences Manually](#)

### 8.2.1 Installing and Configuring on Linux

#### Installing the Tarball

As root, download the tarball to any repository location that is convenient for you. Unpacking the tarball will place the contents into the correct sub-directories as long as you include the `-P` switch in your `tar` command:

```
tar -zxPf IBPReplicator-4.0.n.i586.tar.gz
```

(The `-n.n` character in the tarball name will vary according to sub-release version.)

- The replserver executable will be installed into `/opt/IBPReplicator`
- The IBPReplicator.conf configuration file will go into `/etc`
- The IBP Replicator shell script will go into `/etc/init.d`

#### Important

The default replication server configuration is made to run replserver at OS runlevel 3. If your Linux distro is running at runlevel 2 (the installation default for Ubuntu, for example), replserver will not run on system boot. You can either raise the runlevel to 3 in the file `/etc/inittab` or edit the file `/etc/init.d/IBPReplicator`, change the lines `Default-Start` and `Default-Stop` and then reinstall the service.

#### Configuring the Daemon

You need to configure some information that the replication daemon uses to start up. To make these settings, open the configuration file `/etc/IBPReplicator.conf` as root, using any plain text editor you prefer, such as `mcedit` or `nano`. Some sample entries are there, commented with `#`. To make a setting, delete the `#` comment marker and type in the appropriate value, taking care to observe case correctness and to follow the format as written:

```
#-----
# Parameters settings to start replicator
# Uncomment and modify as appropriate
```

```
# Configuration type. Can be 'interbase' or 'firebird'
#CONFIG_TYPE=interbase

# Configuration DB connection path
#CONFIG_DB=/opt/IBPReplicator/Config.fdb [path to default configuration database]

# Configuration DB user
#CONFIG_USER=SYSDBA [see note below]

# Configuration DB password
#CONFIG_PASSWORD=masterkey [see note below]

# Configuration DB role
#CONFIG_ROLE=repl_admin [see note below]

# Name and location of pid-file (optional)
#PID_FILE=/var/run/IBPReplicator.pid [see note below]
#-----
```

### User and Password

The settings `CONFIG_USER` and `CONFIG_PASSWORD` refer to the user name and password for the administrator of the configuration database, as you defined them (or have yet to define) when setting up the configuration using the Replication Manager utility, either from a Windows client on the network or installed locally under wine.

### Configuration Database Role

If you needed to create an SQL role with permissions to enable a non-SYSDBA and non-Owner user to operate on the configuration database then configure this parameter; otherwise, ignore it.

**WARNING** Never use the name 'REPL' for a role.

### Location of PID File

This setting determines where the replserver will write the pid file that behaves as a lock to prevent two instances running concurrently. It can be reconfigured here or, alternatively, with a command-line switch.

### Starting replserver Automatically as a Daemon

To set up replserver to run automatically as a daemon at boot-up:

```
chkconfig IBPReplicator on
```

**TIP ::** You can run multiple replserver daemons, as long as each uses its own PID file.

## 8.2.2 Controlling replserver Processes

### Controlling *replserver* as a Daemon

When the replication daemon is under `chkconfig` control, you can use the following service commands to control it during operation.

To start the daemon:

```
service IBPReplicator start
```

To stop and restart the daemon:

```
service IBPReplicator restart
```

To stop the daemon without restarting it:

```
service IBPReplicator stop
```

To reload the configuration without stopping the daemon:

```
service IBPReplicator reload
```

### Log Files

If replserver is run as a daemon, the replication log is written in `/var/log`.

## Controlling *replserver* with Switches

You can use the command `replserver` with the appropriate switches to start the replication server directly, as a daemon or as an application. Some switches are required, others are optional and some take parameters.

You can use this method to run multiple instances of `replserver` as applications. You should not try to run it as a daemon if there is a service already running under `chkconfig` control.

### Switch Format

Note that the syntax and semantics of the switches are now the same as those used for command-line operation on Windows, although not all of the switches available on Windows are available on Linux.

The format for supplying switches and their arguments is

```
-x=parameter
```

where `-x` is a hyphen followed by a character representing the switch and `parameter` is the actual value for the argument.

### Required Switches

The `replserver` command must include these switches:



Switch and Argument	Parameter	Comment
-t=<type>	<type> interbase   firebird   oracle	Case-insensitive
-u=username	Should be the name of the user that is connecting to the configuration database.	Make sure this user exists in the security database on the server where the configuration database resides. Not case-sensitive.
-p=password	The password of this user.	Passwords are case-sensitive.
-c=<cfg>	<cfg> is the connection string for the configuration database that is to be used for this process instance.	Because events are used to communicate with a running replication server, it is essential to use the same configuration database parameter when performing each action regarding a specific instance of the replication server.  Remember that filesystem names on Linux are case-sensitive.

### Optional Switches

These switches are optional:

Switch and Argument	Parameter	Comment
-a, --application	Run as application	Use this switch when you want to run replserver as an application.  NOTE In application mode the logs are placed in the in working directory.
-r=<role>	Valid role identifier of a role (package of privileges) optionally defined in the configuration database for the user doing the replication.	The environment variable ISC_ROLE, if defined, will be used as the default role if the -r switch is not used; otherwise the role is treated as a no-op (ROLE NONE on Firebird).
-?	Help (no parameters)	Displays a tight summary of these switches and parameters.
--debug	Run in debug mode.	This is logging level 7, the most verbose level. If a support consultant asks you for a "debug log", run your replication process with this switch.
--go	Start replication automatically and quit after replication is complete	Convenient for running replication from another application/script
--log_to_window=logginglevel	1: Errors Only 2: Warnings and Detailed Error Description 3: Information 4: Detailed Information 5: Debug	The logging switches are optional. You can use a logging level switch here to override the default logging level defined in the configuration. For example, Debug level (5) can be useful when you need to troubleshoot a configuration.
--log_to_file=logginglevel	1 to 5, as above.	Same as above, writing to the log file. You can log to window and log to file simultaneously if you wish.
--log-file file-path or -l file-path	Provides an initial log file for use between server start and configuration load	Enables troubleshooting for cases where replserver has problems loading the configuration but the default location for the log file is unavailable for some reason
--pid-file=filespec	<filespec> Location and name of the file that Replicator uses to prevent multiple instances from running concurrently out of the same config file	Default when only one instance of replserver is running is /var/run/IBPReplicator.pid. Use different names for multiple instances.

## Controlling Running replserver Processes

A process can be controlled using signals sent via the `kill` and `killall` commands. Use `kill` to target a specific process by its process ID ( `pid` ) or `killall` to target all processes named `replserver`. The available parameters are:

### TERM

TERM (terminate, 15) is the default signal. It stops the current replication and shuts down the replication server:

```
kill TERM <pid>
[ or ]
killall TERM replserver
```

### HUP

HUP (hang-up, 01) allows you to pause the process and reload the configuration:

```
kill HUP <pid>
[ or ]
killall HUP replserver
```

### USR1

USR1 is a custom signal allowing you to request an immediate replication:

```
kill USR1 <pid>
[ or ]
killall USR1 replserver
```

## Multiple Concurrent Replication Processes

Multiple replication processes can run concurrently on the same host system, provided each instance of the replication server uses a different configuration database.

Trying to run multiple instances of `replserver` against the same configuration database will have unpredictable results.

### 8.2.3 Loading Licences Manually

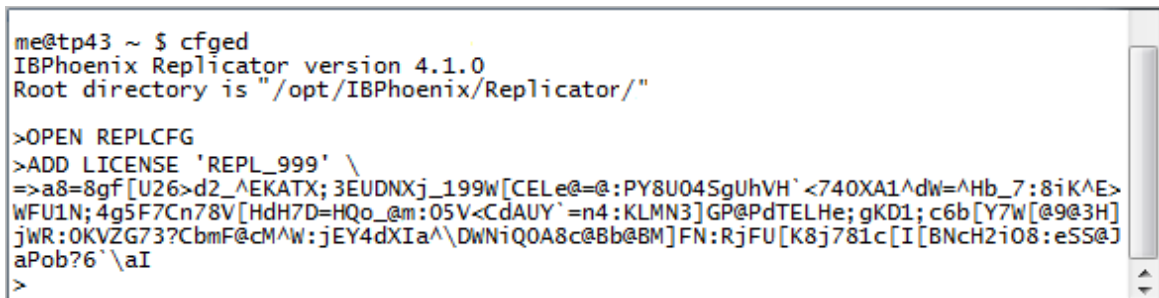
#### Installing Licences for a Linux Host

If you are already using a Windows client or a Wine server to run Replication Manager for configuring your replications, this section need not apply.

In the event that you have created or restored your configuration database on a Linux server and you don't have the Windows Replication Manager connected, you can install your licences manually using the command-line configuration editor, `cfged`.

In a graphics terminal, go to the directory where the Replicator executables are installed and start `cfged`. Open the text email attachment file that contains your licences in a text editor. Note the licence ID, e.g., 'REPL\_99' in our example, and start the `ADD LICENSE` command as indicated in the screenshot. Type a backslash followed by a Return to get a continuation prompt.

Copy the license key to your clipboard application and paste it beside the `=>` continuation prompt:



```
me@tp43 ~ $ cfged
IBPhoenix Replicator version 4.1.0
Root directory is "/opt/IBPhoenix/Replicator/"

>OPEN REPLCFG
>ADD LICENSE 'REPL_999' \
=>a8=8gf[U26>d2_^EKATX; 3EUDNXj_199W[CELe@=:PY8U04SgUhVH' <740XA1^dw=^Hb_7:8iK^E>
wFU1N; 4g5F7Cn78V[HdH7D=HQo_@m:05V<CdAUy`=n4:KLMN3]GP@PdTELHe; gKD1; c6b[Y7W[@9@3H]
jWR:OKVZG73?CbmF@cM^W:jEY4dXIa^DWNiQ0A8c@Bb@BM]FN:RjFU[K8j781c[I[BNCh2i08:eSS@]
aPob?6` \aI
>
```

Continue to add any additional licences you have in the same manner. Use `EXIT` to end your `cfged` session.

For more information about `cfged`, refer to the [Configuration Editor](#) topic in Chapter 9, Advanced Topics.

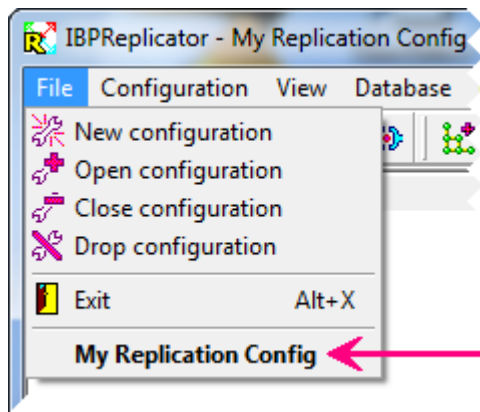
## 8.3 Management Tools

The Replication Manager encompasses a number of utilities through which you can manage and monitor replications in your network at any time.

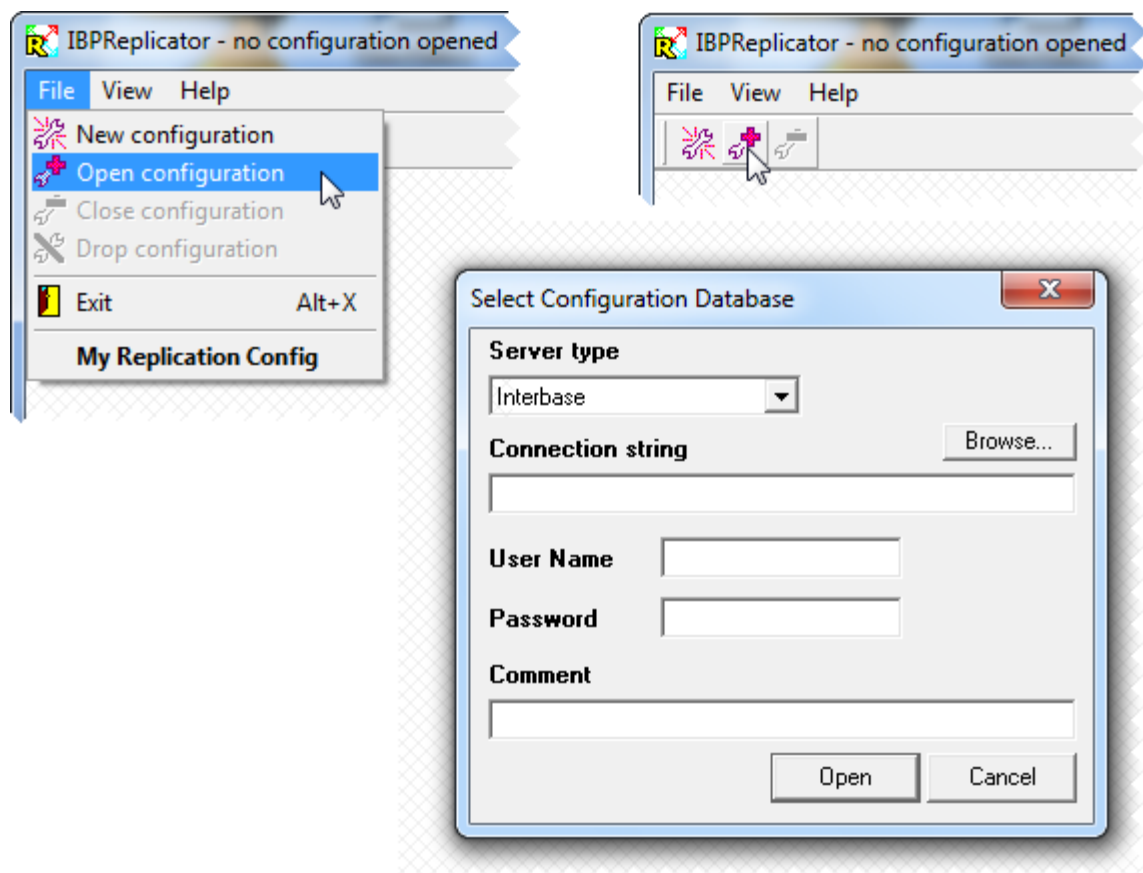
### Accessing the Replication Setups

All of the replication setups that you can access on your network are available through Replication manager. Each setup is in a configuration database.

Recently visited configurations are stored in the Registry of the machine from which they are accessed. Replication manager retrieves the paths to these configurations, enabling "quickstart" access to the config database via a picklist in the File.. menu:



For existing configuration databases that have not been accessed before from this machine, you can use File..Open or click the Open button on the toolbar. An input form will appear, similar to that used for creating a new configuration database:



If you need to remind yourself what to enter into these input fields, [refer back](#) to the notes about filling these fields when creating your first configuration database.

## The Tools

[Replication Monitor](#)

[Replication Scheduler](#)

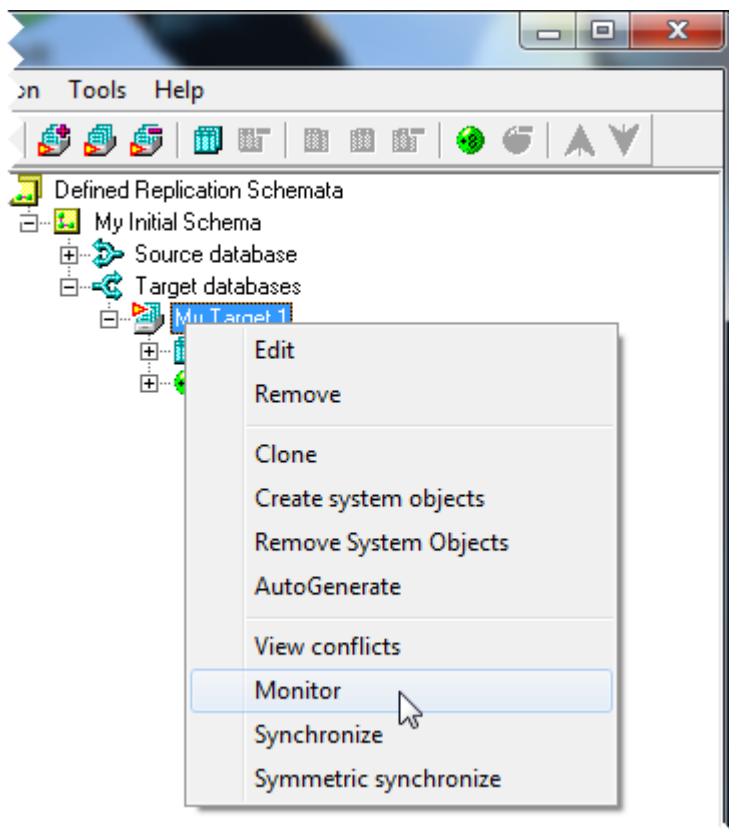
[Schema View](#)[Notify Server](#)[View Conflicts](#)[License Manager](#)

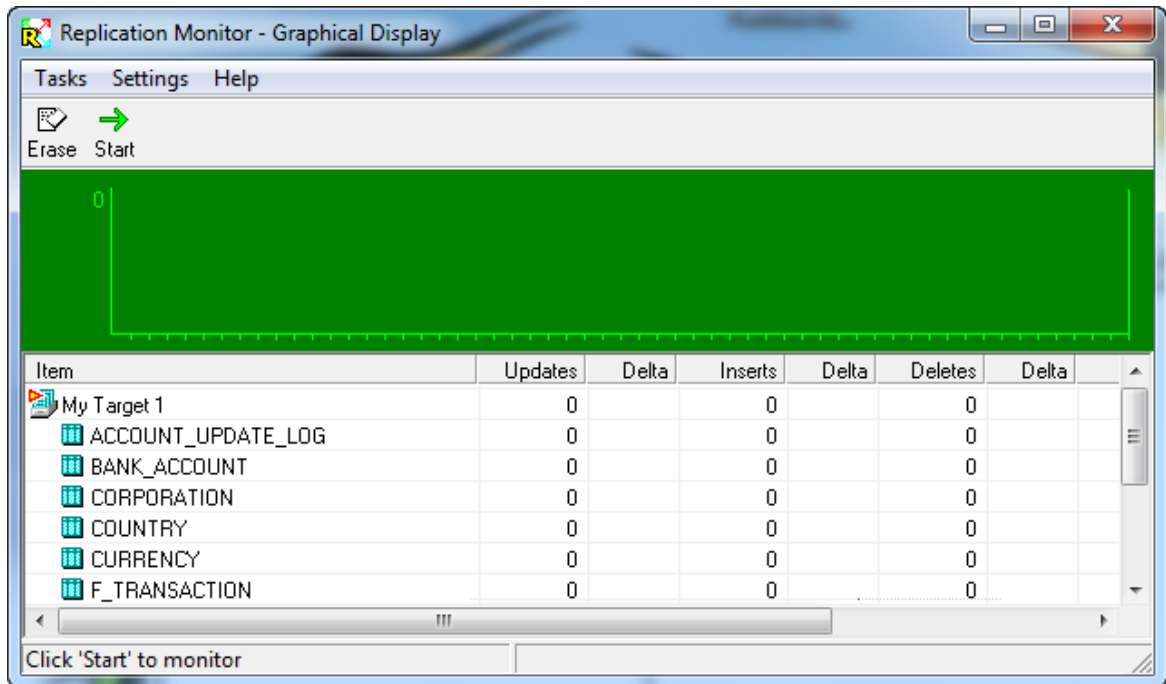
### 8.3.1 Replication Monitor

The Replication monitor displays a graph of replication activity for the replication schema that is currently selected in its tree.

By default, from this version forward, monitoring is disabled by default to reduce performance overhead. Specified tables can be tagged and untagged for monitoring by setting the [Keep Statistics](#) flag in the Table Settings area of configuration.

If monitoring is enabled, open the monitor by right-clicking on the Target database you want to monitor and selecting the Monitor option from the context menu:





### Activating and Stopping the Monitor

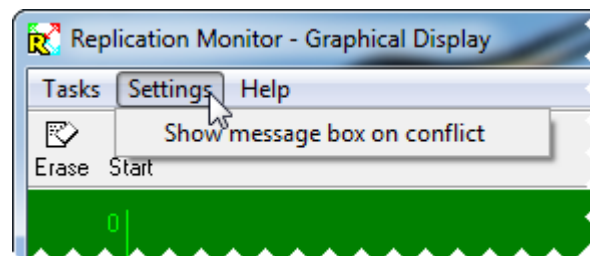
Click the Start button to begin monitoring the selected types. While monitoring is going on, the button caption changes to Stop. Click this button again to stop the graphing process.

### Keeping Things Rolling Along

You can graph just one type of replication statistic or any combination of types. While the monitoring and graphing is going on, you can

- clear the display by clicking the Erase button
- select schemata in the tree at any time
- set or unset a flag to receive a message box when a conflict occurs:

[Menu | Settings | Show message box on conflict](#)



Click the button to set the message box flag on:

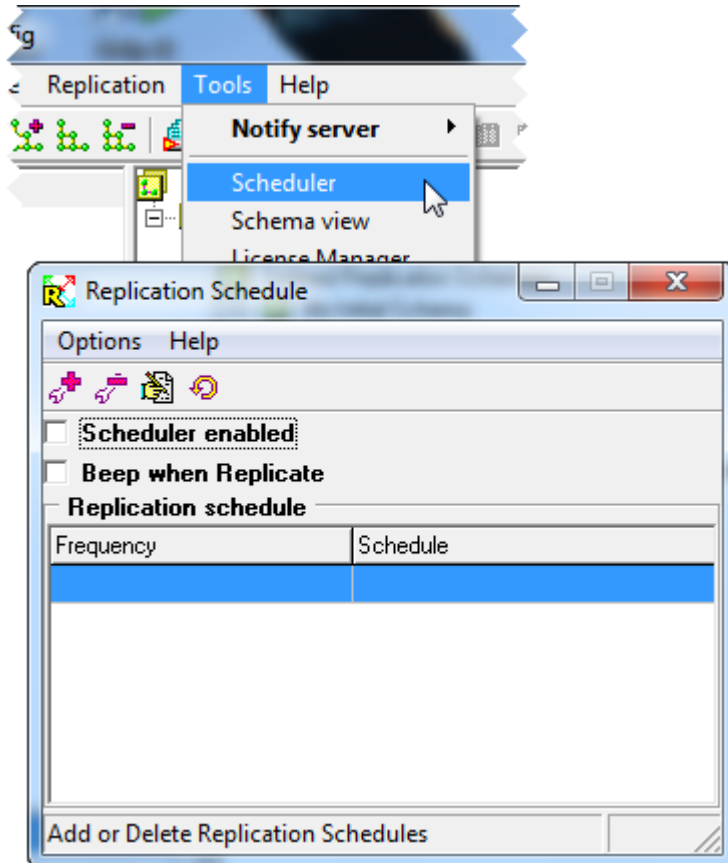
Click it again to toggle the flag off.

### 8.3.2 Replication Scheduler

The Scheduler provides the ability to manage replication at custom periods by setting a predefined interval (in minutes) or by frequency—Once Only, Periodically, Hourly, Daily, Week days, Weekly or Monthly.

#### Setting up a Schedule

To set up a replication schedule, select **Tools | Scheduler**:

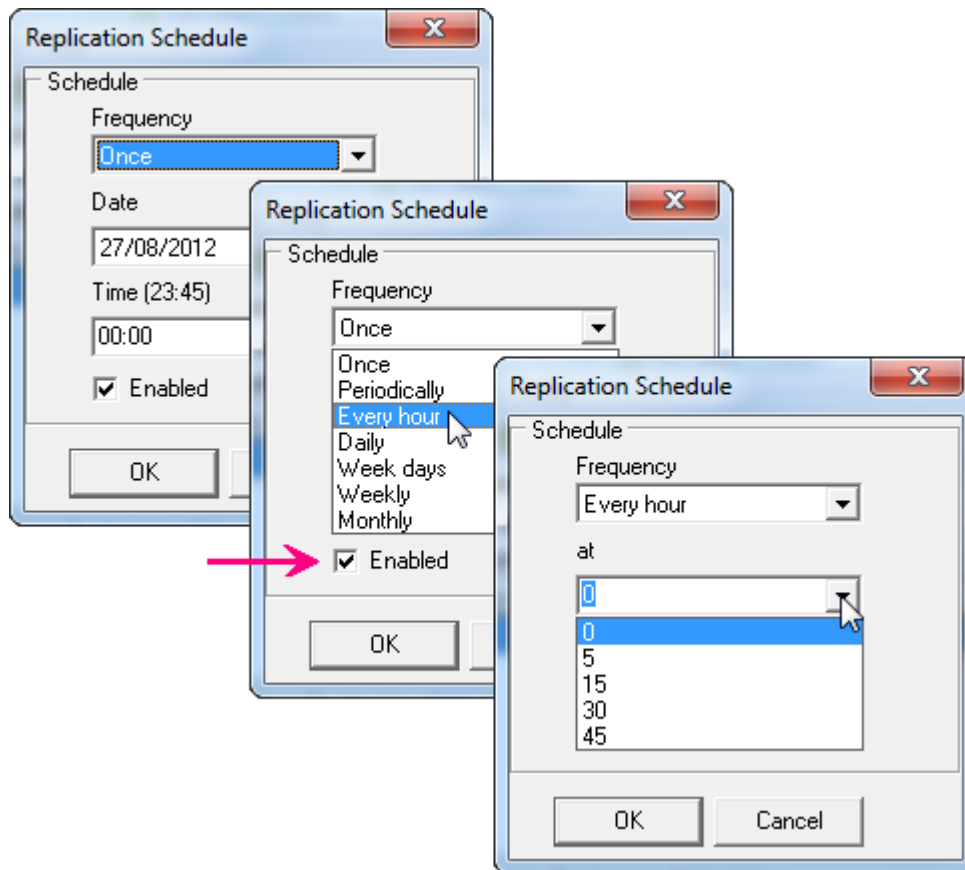


#### Adding a Schedule

To add a schedule, either

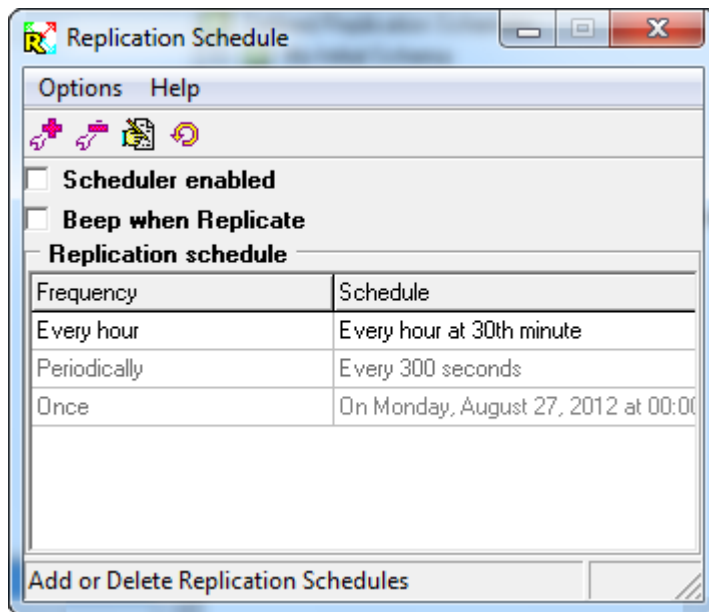
- drop down the Options menu and select Add, or
- click the Add button on the toolbar, or
- simply press the Ins[ert] key on your keyboard



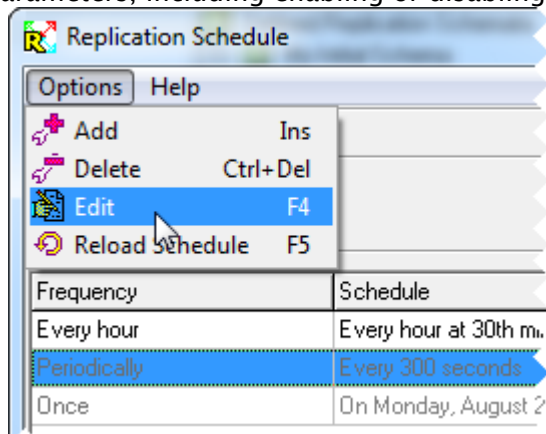


Initially, the add dialog shows the default "schedule", which is to replicate once, at midnight on the current day—which won't happen today, of course! In this example, we are choosing an hourly replication, which we can choose to be on the hour or at a selected number of minutes after the hour. Of course, you can drop down the options and pick the ones that you need. Click OK to add the schedule.

You can add more schedules. Any schedules you create are enabled by default, meaning that they will all be scheduled. For example, you might want to include an hourly and a daily schedule and keep both enabled. Any that you want to keep for alternative or occasional use can be disabled by deselecting Enabled on the final window. Disabled schedules appear as greyed-out in the list:

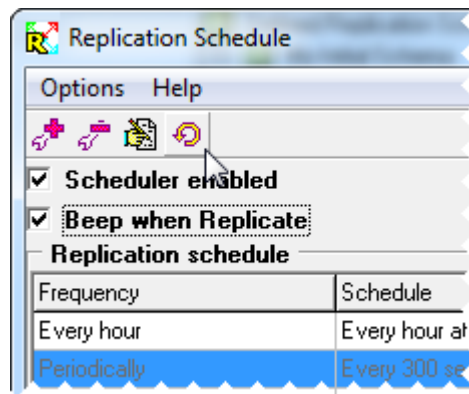


In future, you can select a schedule item in the list and use Options>Edit to change the parameters, including enabling or disabling the item:



### Loading and reloading the Scheduler

- Check on Scheduler enabled
- If you want to hear an audible signal when a replication starts, also check on Beep when Replicate
- To update the Scheduler and replace any schedule that is currently in effect, select Reload Schedule from the Options menu, press F5, or click the Reload Schedule button on the toolbar:



### Altering the Scheduling

Schedules can be added, edited or deleted as required. Always remember to Reload Schedule if an existing schedule is in effect.

**TIP** If you need only occasional replications, you need not schedule replications at all: simply use the menu option Tools | [Notify Server](#) | Replicate Now to replicate whenever it suits you.

### 8.3.3 Notify Server

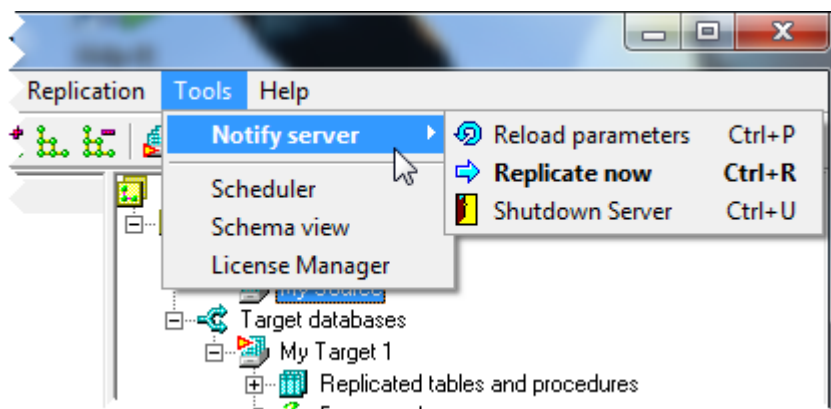
It is possible to intervene in the running Replication Server's activity and instruct it to perform certain actions. You can do this intervention via the Notify Server utility, from the main menu of the Replication Manager.

On Firebird and InterBase, this utility uses events to alert the server. It is essential to open ports in your firewall on both the server and the target database hosts for the event traffic.

On Oracle, it uses Publish-Subscribe notification in anonymous namespace.

### The Notify Server Options

Tools | Notify Server offers three options:



### Reload Parameters (Ctrl-P)

Instructs the server to reload all its settings from the current configuration database, usually because the configuration has been altered and there is no need to shut the server down and restart it.

### Replicate now (Ctrl-R)

Instructs the server to replicate immediately, without waiting for its next scheduled replication. If you need only occasional replications, you need not schedule replications at all: simply use this command to replicate whenever you like.

Notifications have no effect if the replication server is not running!

### Shutdown server (Ctrl-U)

Instructs the Replication Server to shut down immediately.

## 8.3.4 Conflict Resolution

When something goes wrong because of a conflict between the source data and the target data, you want to know about and fix it.

If you are checking the log, you will be aware of any conflicts encountered, because they are summarised there after each replication. There are other places in Replication Manager where conflict information can be viewed in more detail.

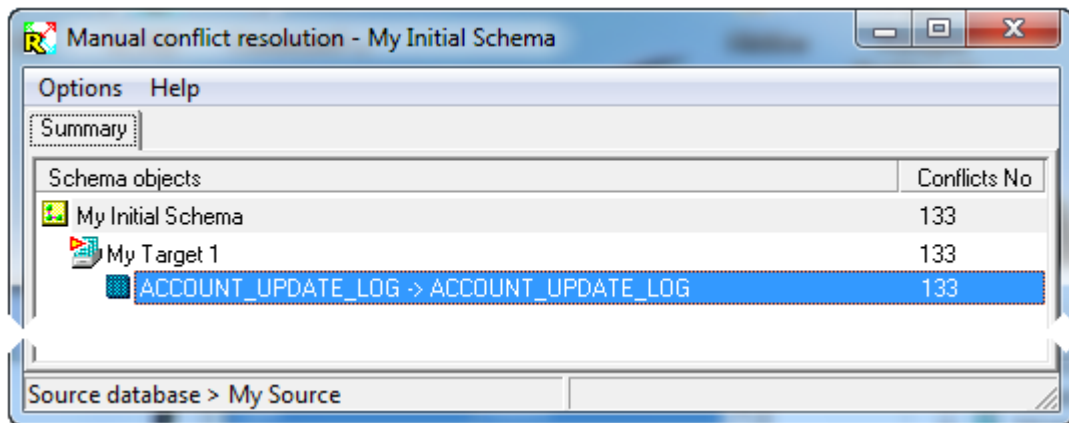
### Viewing Conflicts

With the replication schema you are interested in selected, either

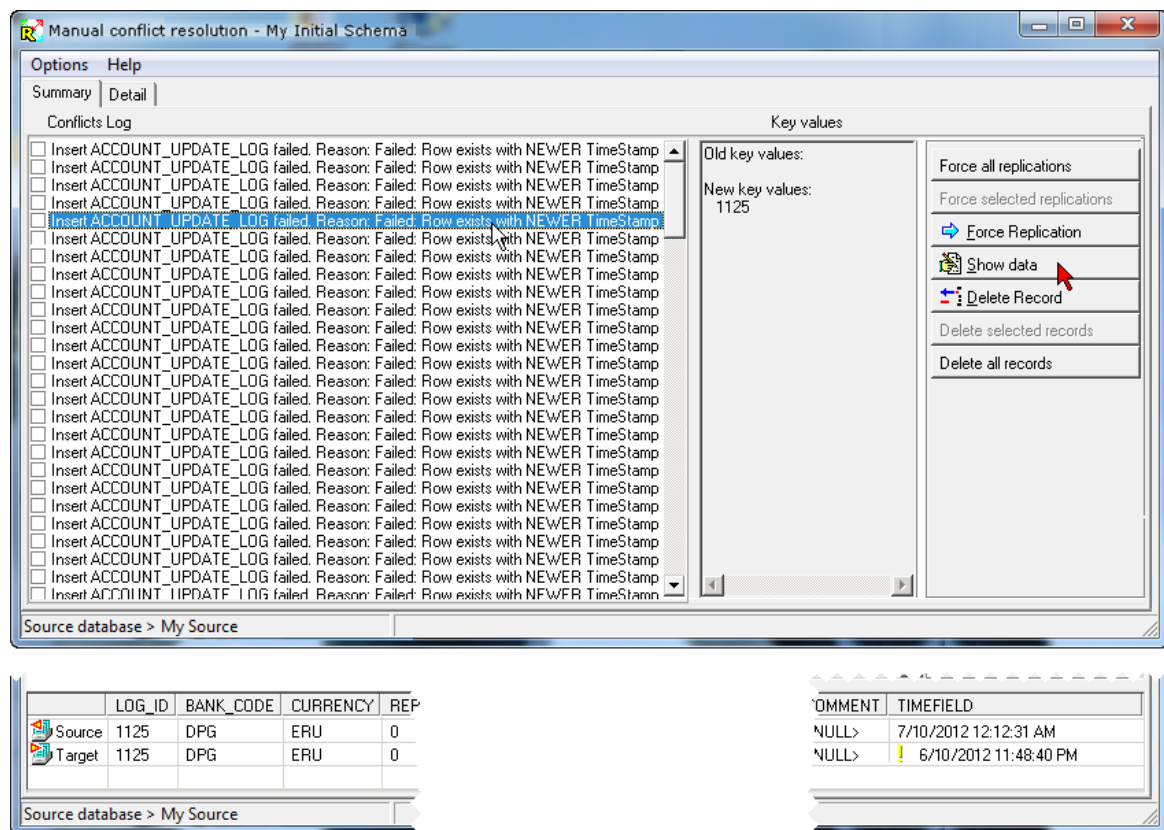
- click [Replication | Tools | View Conflicts](#); or
- right-click on the schema node of a defined source or target database in the Replication tree, or anywhere else within that schema branch, and choose [View conflicts](#) from the context menu

**NOTE** to focus on all conflicts that may have occurred when replicating from a particular source database or to a particular target database, have that database node selected.

If there are no conflicts, the conflict viewer simply says so. If there are unresolved conflicts, the viewer displays them:



Here we see that the target database had 133 conflicts. We want to open the Detail tab to see what those conflicts were:



Every conflict for this table is the same—so we select a record and click the Show Data button at the right to find out the offending field. A new panel opens beneath the Detail display and shows the conflict for this record. Ah—yes! we previously used a synchronization to get all the records from the source table over to this table, but the server clock at the target was 20 minutes fast!

## Resolution

If any conflicts are listed, you can click on one to highlight it and select a button from the Options menu to resolve it:

Force All Replications	IBP Replicator will try to replicate all the problem rows during its next scheduled replication
Force Selected Replications	If you want to force replication on some but not all of the conflicting records, use the checkboxes at the left of the detail display to select/deselect. Then, the button will become enabled and you can use it to tell IBP Replicator to try to replicate those rows during its next scheduled replication
Force Replication	IBP Replicator will try to replicate the problem row during its next scheduled replication
Delete record	Delete the selected conflict. Data in source and target tables will stay as they are
Delete selected records	If you want to delete groups of conflict records, select those you want deleted. This will enable the button, ready for you to click it to have the conflict records abandoned.
Delete all records	Delete all the conflicts. Data in source and target tables will stay as they are

After an option is executed, the record in the manual log will be deleted and the conflict will disappear from the display, considered to have been resolved.

Finally, the view reports the "all clear" and you can close it.

### Off-line Replication

It is possible to perform manual conflict resolution following an off-line upward replication from a Files.. type "database". Instead of being written to the manual log table, the conflicts are written to .conflict files in the inbound directory that was used as the "source database". The resolution options are limited to Force All Replications and Delete all records. For more information, refer to the topic [Manual Conflict Resolution After Off-line Replication](#) in the next chapter.

## 8.3.5 License Manager

### About Licensing

#### Evaluation Licence

No specific restrictions are placed on the distribution of the Evaluation version, except for the following:

- Once entered, the evaluation licence lasts for a period of 14 days
- The evaluation licence is limited to one server and four replicant licences

#### Normal Licensing

On each host machine that acts as a replication server, a server licence must be installed. Each database—whether source or target—needs a replicant license. The minimum server licence needs two replicant licences. One is bundled with the server licence automatically and issued separately. You should order your initial Server licence with enough Replicant licences attached to cover all of the databases that will be involved in a

single replication cycle.

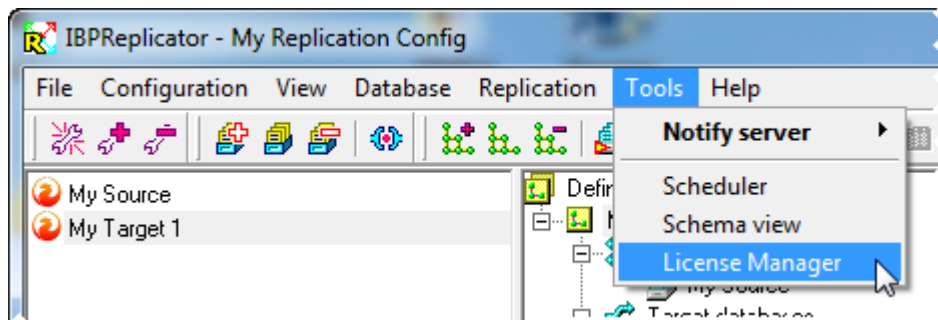
So, for example, a machine that acts as the replication server and has a single Source database, and which is replicating to one remote Target database, will need only a Server licence with the bundled Replicant licence plus one more replicant licence purchased separately.

#### Same database used as multiple replicants

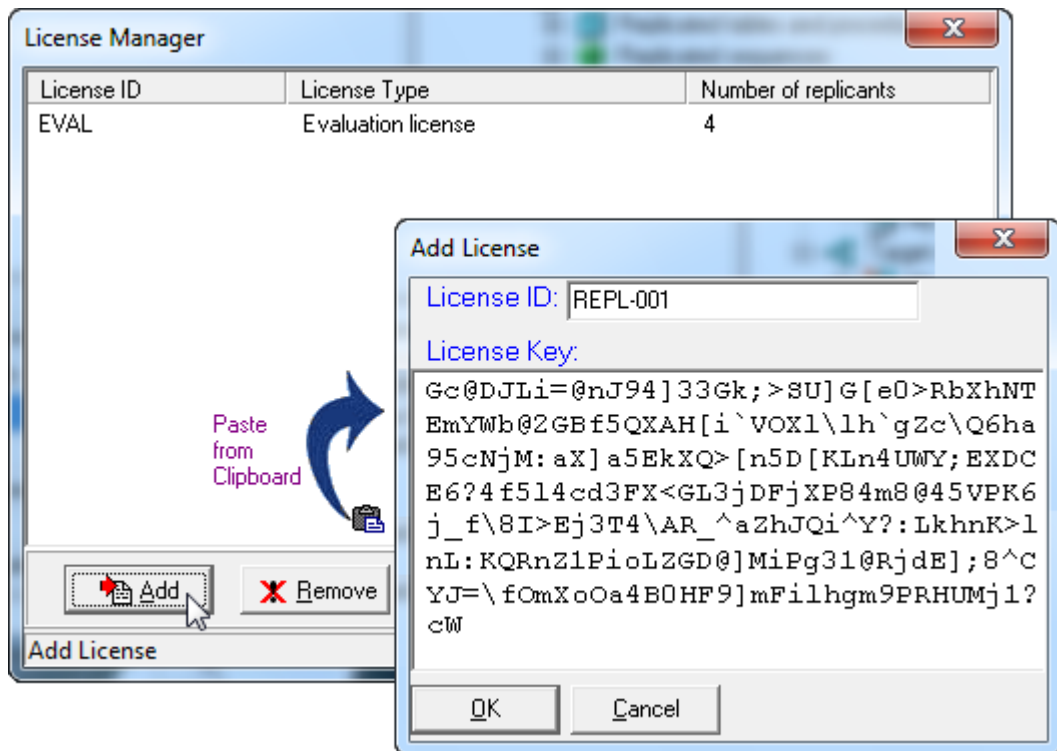
No database needs more than one Replicant licence. If the same Source or Target database is being controlled by more than one instance of the Replication server then you need to account for one database only once as a replicant when calculating the replicant denominations for your server licences.

#### Installing Licences

Once you have created your first configuration database, the Tools menu of the Replication Manager becomes visible. It contains the License Manager applet for installing your licences:



Click the License Manager item to start the applet, then click the Add button to add your first licence.

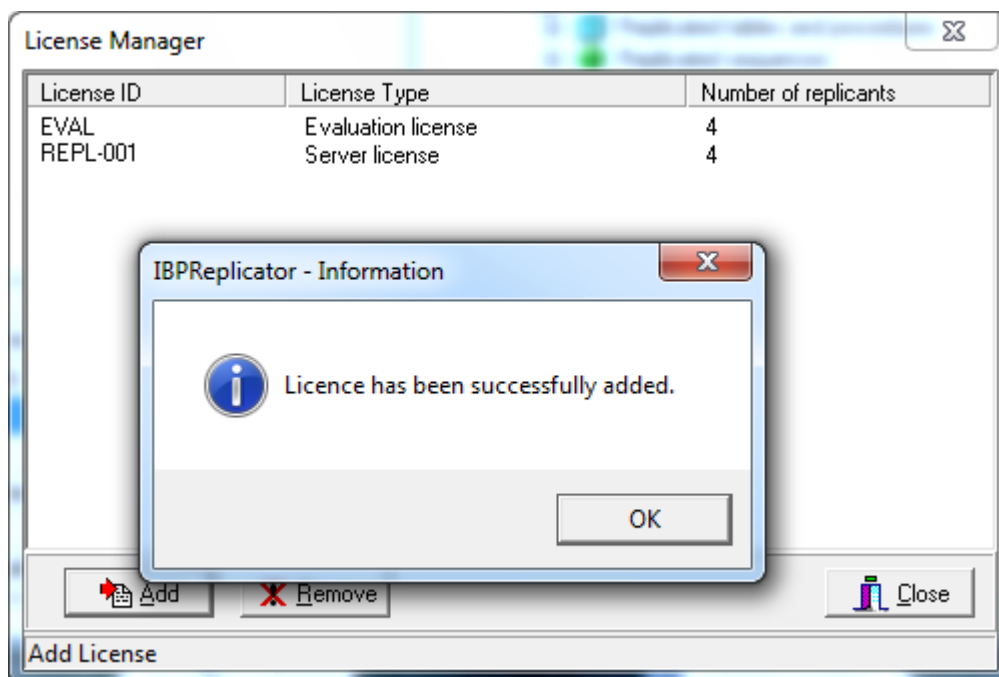


### Evaluation License

Enter EVAL as the License ID and leave the License Key field to be filled automatically.

### Normal Licensing

Type in or copy/paste the first of the License ID and License Key value pairs from your licensing document and press OK. The first licence registration appears in the display with a "success" message:





If you get an error message, try again. The codes are case-sensitive.

Continue to add each of your licences in the same fashion, until you have registered all of them. They should appear on the display as a list.

Don't worry about the order in which the licences are registered. When the time comes for action, each server is only concerned that there are enough available licences of each type to perform the tasks it is asked to do.

When you are finished, click the Close button to terminate the applet. You can access the License Manager any time by returning to the Tools menu or running it from the Start Menu.

## Loading Licences Without Windows

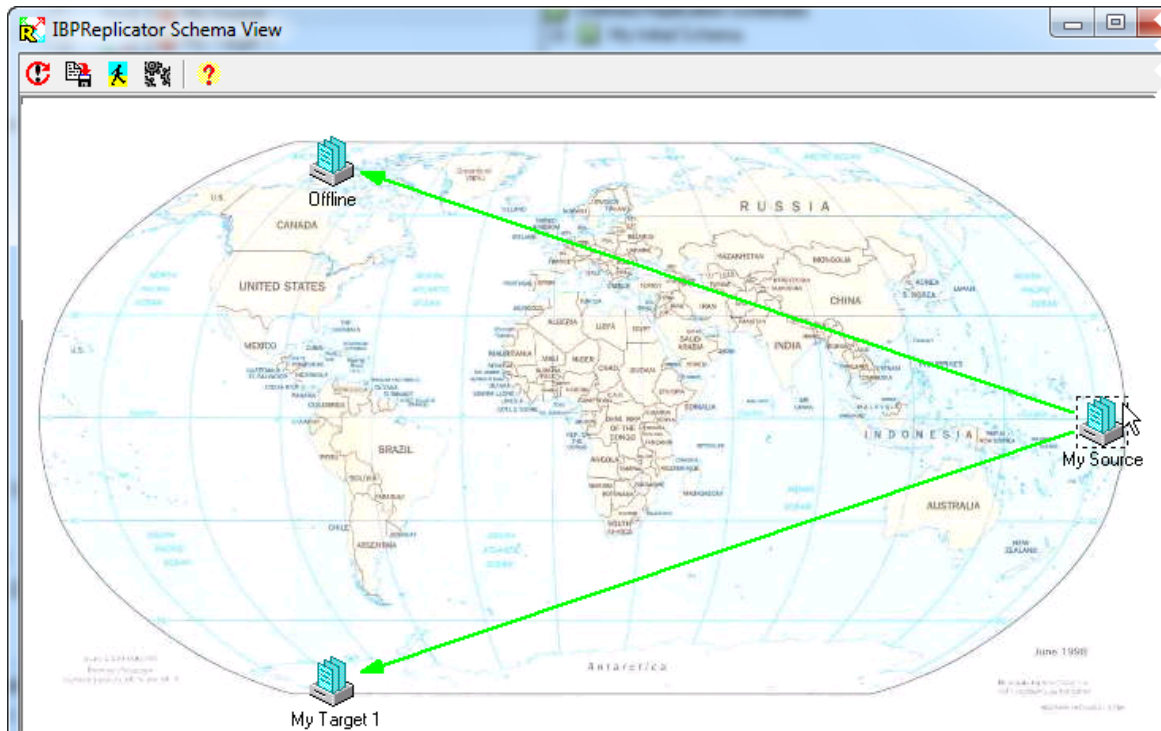
If you have completed your configuration for running IBP Replicator on a non-Windows host and no longer have Windows or Wine available to run the License Manager, you can load your licences manually from a script. For more details, please refer to the topic [Loading Licences Manually](#).

### 8.3.6 Schema View

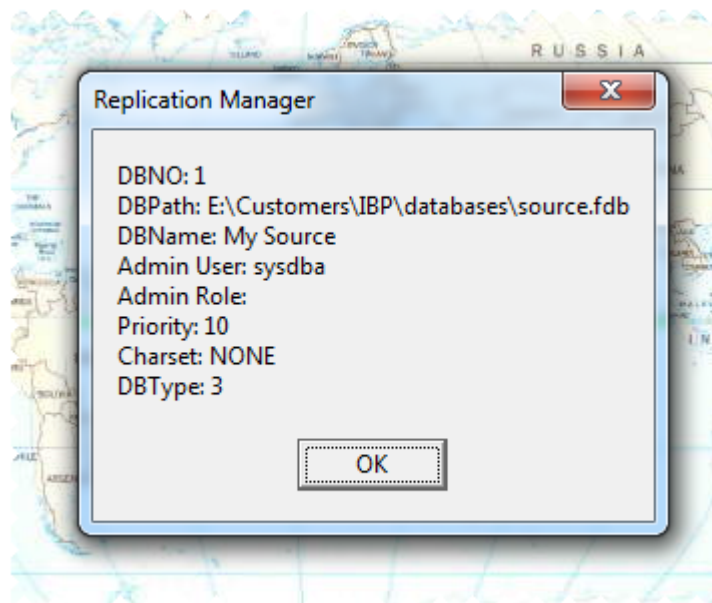
Schema View, accessed from the Tools menu, provides a way of storing a visual "map" of the physical layout of your replication servers.

Although this tool is aware of how the sources and targets in a configuration relate to one another, at this point it is not a visual doorway to your replication schemas. However, it can be useful as a quick way to identify information about the databases in your replication system and to store it diagrammatically.

Click on a database icon:



Then right-click to see the database information:

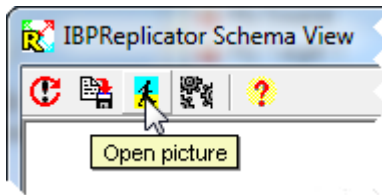


You can pick up a database icon and drag it to somewhere meaningful on your background image and then drop it where you want it. Of course, this would be so much more meaningful to most of us if the map were on a smaller scale! So the tool allows you to use any image you like as your background: perhaps a map of your region, country, plant, server room—whatever works for you!

### To Use Your Own Picture as Background

Create or copy the image (.bmp, .jpg, etc.) into a convenient location and scale it to suit.

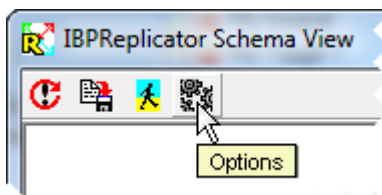
Click the Image button in the toolbar:



This will open a file browser dialog, enabling you to locate and select your image.

### Other Layout Options

To open a dialog where you can customise your display, click the Options button:



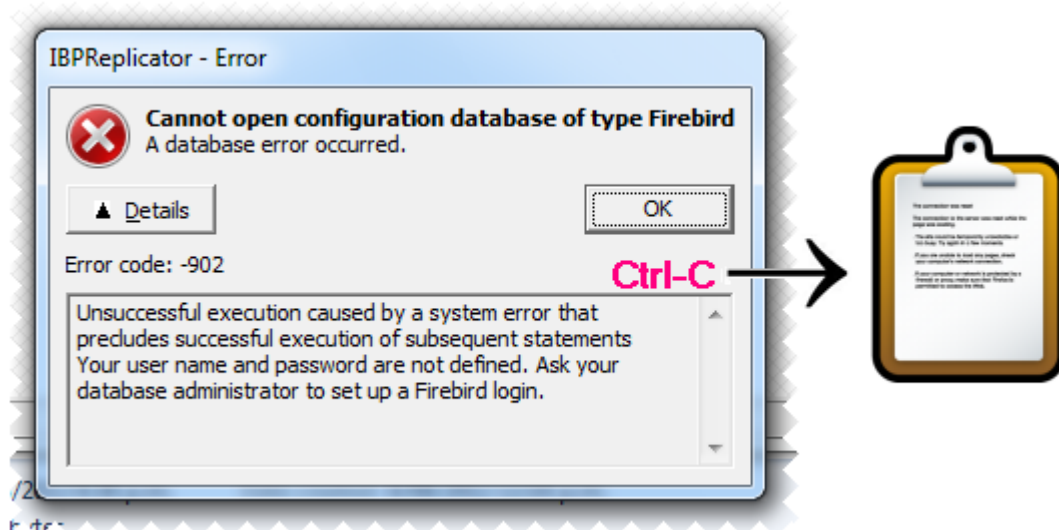
### Remember to Save Your Settings

If you wish to save your layout, simply click on the Save button.

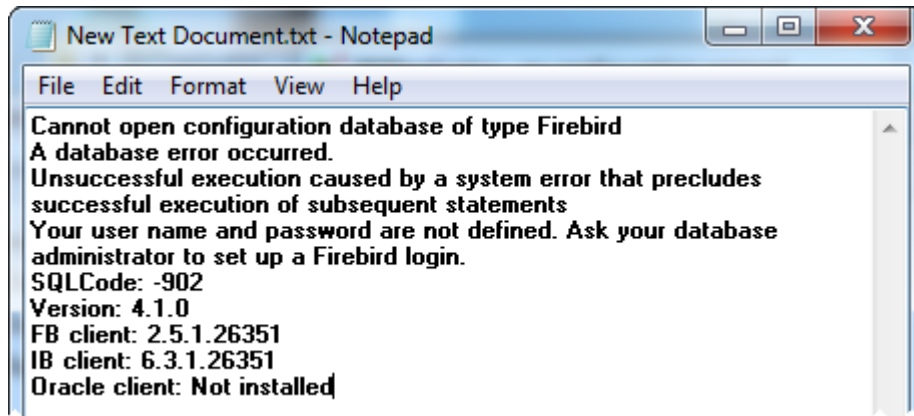
To restore the original (default) layout simply click the Reset button. You can also restore the original layout from the Options dialog, by checking Use Defaults there.

## 8.3.7 Reporting Errors

Whenever an error occurs during interactive activity (configuration using Replication Manager or Replicator execution as an application) a dialog box should appear. If details are available, a Details button will be visible to display any accompanying error message. You can use Ctrl-C to copy the entire message, including its details, to the Windows clipboard:



The message can then be pasted into a text file or directly into an email using Ctrl-V. The screenshot below shows the content of the message above, pasted from the clipboard into Notepad:



## Reporting Errors to Support

When you report errors to the support list [ibp\\_replicator@lists.ibphoenix.com](mailto:ibp_replicator@lists.ibphoenix.com), please use the method described here to post error details in your problem description. Please do not post screenshots of error dialogs into the list.

## Subscribing to the Support List

Support questions will be answered in the list free of charge, even if you are just at the stage of evaluating IBPhoenix Replicator. To subscribe, click the link below and send the email that is generated into your default email client:

[ibp\\_replicator-request@lists.ibphoenix.com?subject=subscribe](mailto:ibp_replicator-request@lists.ibphoenix.com?subject=subscribe)

Alternatively, [visit the web interface](#) to fill in and submit a subscription form.



# Chapter 9

## Chapter 9 - Advanced Topics

## Chapter 9 - Advanced Topics

This chapter examines an assortment of topics that will be of interest to the replication designer when addressing more complex replication requirements. As time goes by, more articles and "How-tos" will be added.

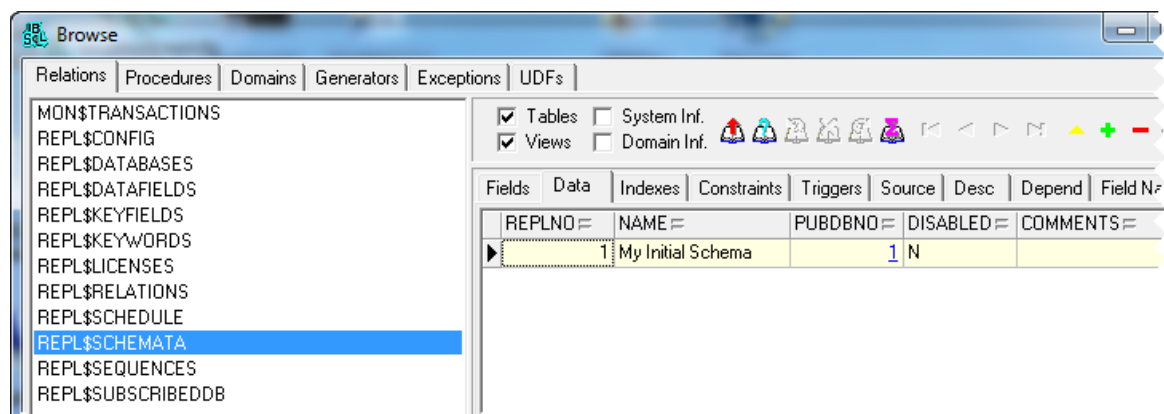
- [Schema Numbering](#) explains why the internal numbers assigned to configurations by the Replication Manager might be interesting to the replication designer
- [Metadata Changes](#) describes what happens when metadata changes occur in the source database
- [Adding Timefields](#) provides a way to generate a script that you can run to add Timefields to all of the tables in a database
- [Advanced Mapping Techniques](#) provides a simple example of writing a stored procedure in a target database to accept replication parameters from the source database
- [Complex Schemas](#) summarises some more complex replication models and works through an example of n-way replication using the Hub-and-Spoke model

### 9.1 Schema Numbering

In many replication environments, it is possible to have the same database being referred to as source database in more than one schema. Each schema configuration will cause its own system objects to be created in the source database. Unless precautions are taken, the identifiers of SOs for one schema could conflict with those for another.

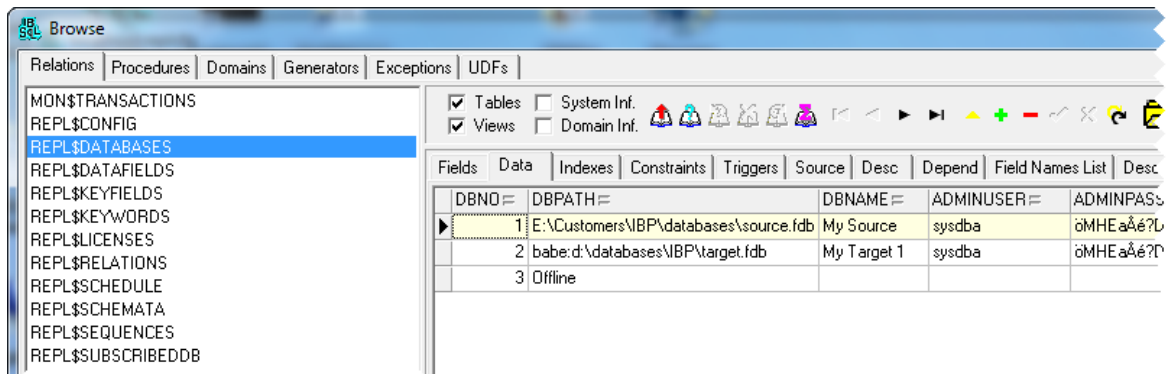
Each schema that is created in a configuration database is referred to by an internal number. This number is used in the naming of triggers, etc., when the System Objects are created. The following screenshots from a graphical DB browser tool illustrate how the numbering is set automatically.

In this shot, we are looking at the configuration database for the replication used in this manual.



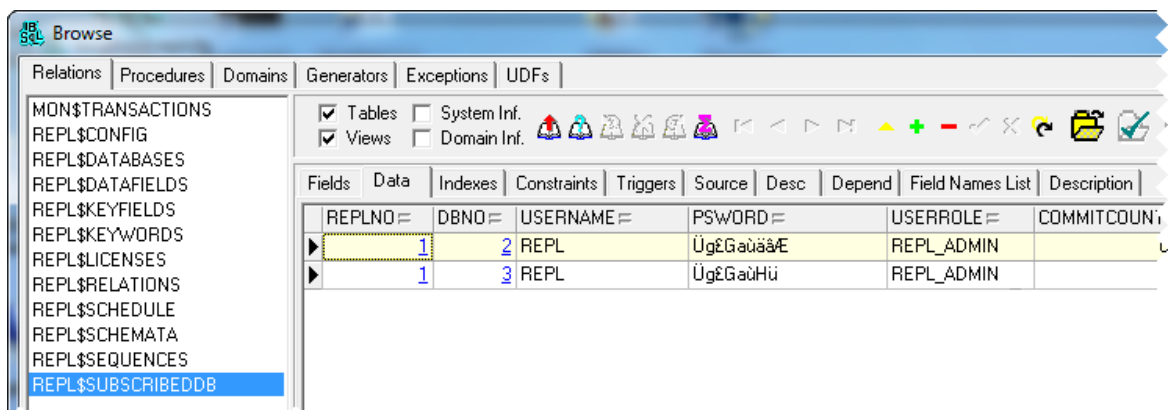
So far, we have only one schema in the schemata table. Its replication number is 1.

Next, we look at the databases defined:



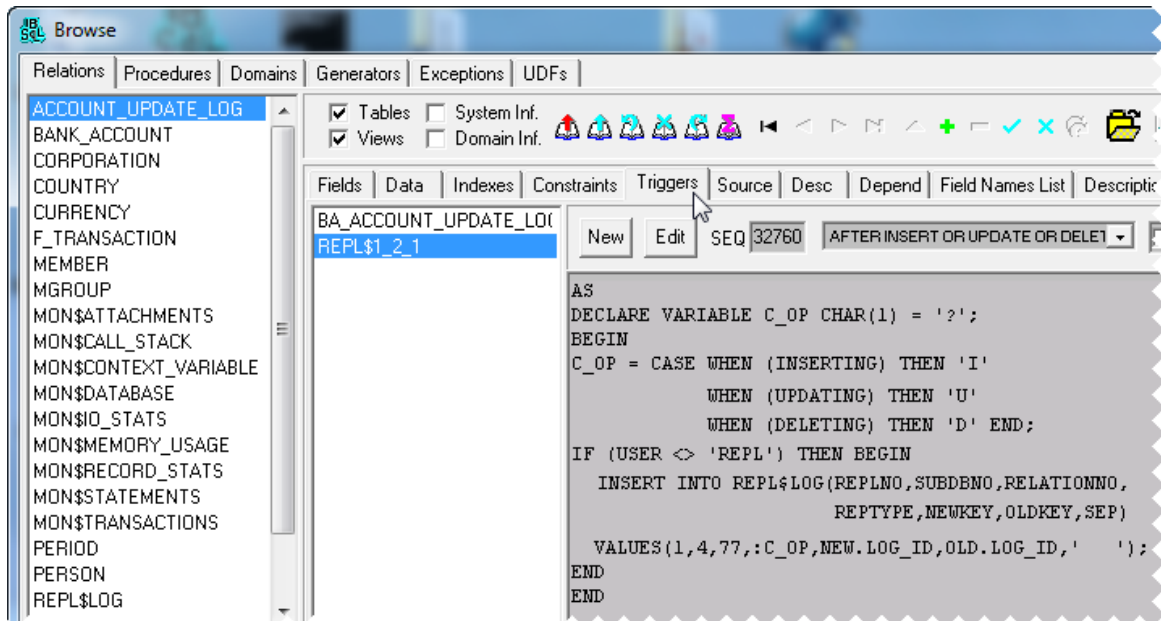
Here we see two databases and our offline target, in the order that they were defined.

Now, we look at the "subscribed" databases—databases in the configuration that have actually been defined as targets:



Again, no surprises. We don't see our Source database here because it is the "published" database for this configuration, i.e., the one that is replicating its data to the targets.

In the next shot we look at some system objects created in our source database:

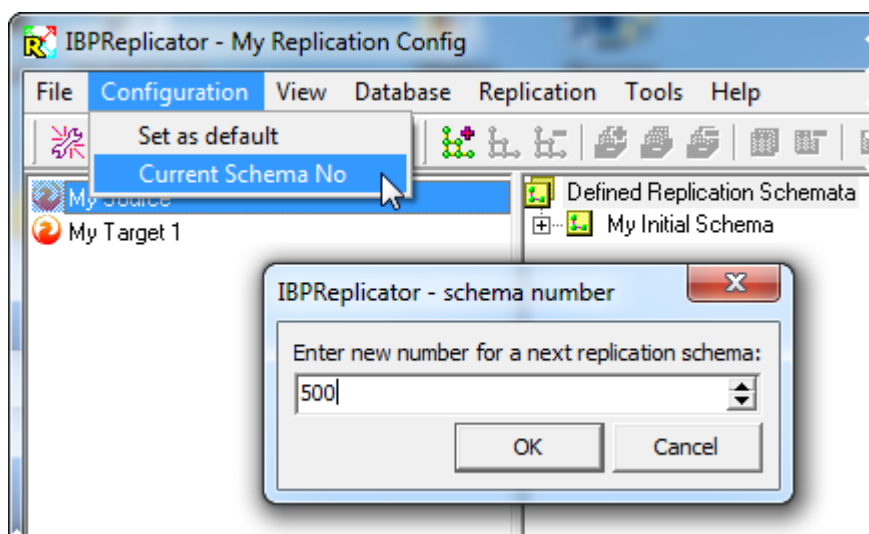


We find some triggers have been created for our replicated tables. The prefix is REPL\$1 and the number following it is the replication number of the table—it was the fourth table we defined in the schema. Each configuration database numbers its objects independently. As long as the internal Schema Numbers are unique across all configuration databases, no conflicts can occur.

## Setting Your Own Schema Number

When creating a new Configuration Database, you can ensure that the Current Schema Number is set to a number that will prevent two configuration databases using the same numbers. Do this task before you save the new schema record. It is necessary only if one or more source databases are included as sources in more than one configuration database.

Use [Configuration | Current Schema No.](#) from the main menu:



For example, ConfigOne.fdb could start at 1, whilst ConfigTwo.fdb could start at 500.



## Multiple Schemas within One Configuration Database

Each new Schema created within any configuration database will cause the internal Schema number to be incremented by one—the default Current Schema No. that you see when this dialog pops up is generated internally by a generator called REPL\$GEN, so there is no risk of conflict with multiple schemas inside the same configuration database. Most configuration databases have fewer than 10 replication schemas defined, so an easy rule would be to start each configuration database with values separated by 100.

**IMPORTANT** When you do have multiple configuration databases, use the command-line switches with ReplServer.exe/replserver to set the one to use for a specified instance.

## 9.2 Metadata Changes

Metadata changes performed after the creation of a replication schema are not propagated between the source and target databases by IBP Replicator: you must take care of this by your own devices.

Replication schemata are not automatically updated to reflect changes in metadata.

### Tips for Smooth Changes

- Prepare a DDL script to implement metadata changes in the source database and use the same script, with modifications if necessary, to update the target database[s]. Ad hoc changes performed with interactive tools are prone to human error when duplicated in multiple locations.
- Plan a down-time "window" for the IBP Replicator service. Enter the "window" by disabling the IBP Replicator service before you run the DDL script on the source. Allow enough time in this window to apply the script to all affected databases, test their effects and modify or replace all affected replication schemata.
- It is strongly recommended that you use a tidy shutdown to remove all users, including other administrative users, from the affected systems while databases and replication configurations are being altered.
- You might wish to archive the replication log file[s] before resuming services.
- Put the database online and restart IBP Replicator once you are happy that the revision is complete.

## 9.3 Adding Timefields

If you plan to use timestamping for as your replication conflict strategy, you will need to add a Timefield column to every source and target table involved in the replication and write Before Insert or Update triggers for each table to perform the actual timestamping.

For large databases, this could be a very tedious and error-prone task involving an extended period of downtime for the users at all of the replication sites. However, you can prepare a script ahead of time to speed up the task. You can reduce the effort involved by defining and executing a stored procedure to generate the DDL and export the required statements into an external table.

## Generating a Script

Start by defining an external table in the database that the timefield value is going to be applied to:

```
create table ext_data external file 'inject_timestamps.sql' (
  data char(90) character set NONE);
commit;
```

Our sample procedure takes as input the timefield identifier that we have set up for the configuration.

The generated script depends on all table names having lengths of < 28 characters. If you have any table names that are longer, you will need to edit the script later to replace the trigger names in the generated script.

```
create procedure inject_timestamps (
  timefield_id varchar(31))
as
  declare variable tablename varchar(31);
  declare variable crlf char(2);
  declare variable data varchar(90);
  declare variable date_today varchar(24);
begin
  crlf = '
';
  /* literally hit the Enter/Return key between 2 apostrophes. No spaces! */

  date_today = CAST (CAST ('NOW' AS TIMESTAMP) AS VARCHAR(24));
  data = '/* One-time script for applying Timefield metadata to tables';
  insert into ext_data (data) values (:data);

  data = crlf || ' generated by procedure INJECT_TIMESTAMP ' || date_today || ' */';
  insert into ext_data (data) values (:data);
  data = crlf;
  insert into ext_data (data) values (:data);
  data = crlf || 'SET TERM ^';
  insert into ext_data (data) values (:data);

  for
    select RDB$RELATION_NAME from RDB$RELATIONS
    where RDB$RELATION_NAME not starting with 'RDB$'
    and RDB$VIEW_SOURCE IS NULL
    and RDB$RELATION_NAME <> 'EXT_DATA'
  into :tablename do
  begin
    data = crlf || 'ALTER TABLE ' || tablename || ' ADD ' || timefield_id || ' TIMESTAMP ^';
    insert into ext_data (data) values (:data);
    data = crlf || 'COMMIT ^';
    insert into ext_data (data) values (:data);
    data = crlf || '/*      */';
    insert into ext_data (data) values (:data);

    data = crlf || 'CREATE TRIGGER BI5_' || tablename || ' FOR ' || tablename ;
    insert into ext_data (data) values (:data);
    data = crlf || 'ACTIVE BEFORE INSERT OR UPDATE POSITION 5';
    insert into ext_data (data) values (:data);
    data = crlf || 'AS BEGIN';
    insert into ext_data (data) values (:data);
    data = crlf || 'NEW.' || timefield_id || ' = CAST (''NOW'' AS TIMESTAMP);';
    insert into ext_data (data) values (:data);
    data = crlf || 'END ^';
    insert into ext_data (data) values (:data);
    data = crlf;
    insert into ext_data (data) values (:data);
    data = crlf || 'COMMIT ^';
```

```

insert into ext_data (data) values (:data);
data = crlf || '/' * ===== */';
insert into ext_data (data) values (:data);
end
data = crlf || 'SET TERM ^';
insert into ext_data (data) values (:data);
data = crlf || '/' * End script */';
insert into ext_data (data) values (:data);
data = crlf;
insert into ext_data (data) values (:data);
end

```

## 9.4 Advanced Mapping Techniques

### Replicating to Stored Procedures

Sophisticated users can customise IBP Replicator to handle specialised replication conditions by defining stored procedures to manipulate the data that the replication will insert, update and delete data in the target database.

#### Requirements

In designing a Stored Procedure that IBP Replicator can use, you need to make provision for everything that will be wanted for the operations.

#### Input Arguments

Provide one input argument for

- each key column
- each data column that will be replicated
- an OLD value for each key column if an update might be involved
- a final parameter that defines the type of action required: I (insert), U (update), D (delete)

#### Output Argument

Provide an integer RETURNS argument that indicates the result of the action. Possible return values are:

- 0: Success. If the stored procedure handles conflicts itself, e.g., by converting an insert into an update on a key violation, the procedure should also return 0.
- 1: Unresolvable conflict. The record is moved to MANUAL\_LOG.
- 2: Fatal error. The replication will not proceed.

#### A Simplistic Example

The example here is simplistic and should not be used as a boilerplate that would cover your requirements. Refer to the [comments](#) at the end of this topic if the reasons for that are not obvious.

The source database contains the following table definition:

```

CREATE TABLE T (
K1 INTEGER NOT NULL,
K2 VARCHAR(10) NOT NULL,

```

```

K3 DATE NOT NULL,
F1 VARCHAR(20),
F2 DOUBLE PRECISION,
F3 INTEGER,
F4 VARCHAR(100),
F5 NUMERIC(4,1),
F6 NUMERIC(9,2),
F7 NUMERIC(15,2),
PRIMARY KEY (K1,K2,K3)
);

```

A stored procedure can then be defined on a target database with the same structure as the source table as follows:

```

CREATE PROCEDURE REPLICATE_T (
K1_NEW INTEGER,
K2_NEW VARCHAR(10),
K3_NEW DATE,
F1 VARCHAR(20),
F2 DOUBLE PRECISION,
F3 INTEGER,
F4 VARCHAR(100),
F5 NUMERIC(4,1),
F6 NUMERIC(9,2),
F7 NUMERIC(15,2),
K1 INTEGER,
K2 VARCHAR(10),
K3 DATE,
ACTION_TYPE CHAR(1)
) RETURNS (RESULT SMALLINT)
AS
DECLARE VARIABLE ROW_EXISTS SMALLINT;
BEGIN
    RESULT = 0; /*default return value*/
    ROW_EXISTS = 0; /* initialize */
    IF (EXISTS ( SELECT 1 FROM T
        WHERE K1 = :K1 AND K2 = :K2 AND K3 = :K3))
    THEN
        ROW_EXISTS = 1;

    /* Inserts: If the row already exists, then exit with result=1. This
    causes the Replication Server to apply conflict rules, which will
    probably cause the procedure to be called again, but now with ACTION_TYPE of "U".
    An alternative approach would be to simply change the ACTION_TYPE to "U" and proceed
    to update the row instead. */

    IF (ACTION_TYPE = 'I'
        AND ROW_EXISTS = 1  ))
    THEN
        BEGIN
            RESULT = 1;
            EXIT;
        END

    /* Updates: If the row does not exist, then exit with result=1. This
    causes the Replication Server to apply conflict rules, which will
    probably cause the procedure to be called again, but with ACTION_TYPE of "I".
    An alternative approach would be to simply change ACTION_TYPE to "I" and proceed
    to insert the row instead. */

    IF (ACTION_TYPE = 'U' AND ROW_EXISTS = 0) THEN
        BEGIN
            RESULT = 1;
            EXIT;
        END

    /* Deletes: If the row does not exist then exit with result=1. The
    Replication Server will log the error, but otherwise ignore it. */

    IF (ACTION_TYPE = 'D' AND ROW_EXISTS = 0) THEN
        BEGIN
            RESULT = 1;
            EXIT;
        END

    IF (ACTION_TYPE = 'I') THEN
        INSERT INTO T(K1,K2,K3,F1,F2,F3,F4,F5,F6,F7)
        VALUES (:K1_NEW,:K2_NEW,:K3_NEW,:F1,:F2,:F3,:F4,:F5,:F6,:F7);

    IF (ACTION_TYPE = 'U') THEN

```

```
UPDATE T SET
  K1 = :K1_NEW, K2 = K2_NEW,
  K3 = K3_NEW,
  F1 = :F1, F2 = :F2,
  F3 = :F3, F4 = :F4,
  F5 = :F5, F6 = :F6, F7 = :F7
WHERE K1 = :K1
AND K2 = :K2
AND K3 = :K3;

IF (ACTION_TYPE = 'D') THEN
  DELETE FROM T
  WHERE K1 = :K1
  AND K2 = :K2
  AND K3 = :K3;
EXIT;
END
```

### Comments

In real life, a less simplistic rendering of example above would need to test the parameters K1, K2, K3, K1\_NEW, K2\_NEW and K3\_NEW for null and ensure that the local variables were initialized and reset in a timely way, to avoid the constraint exceptions and unpredictable effects that null-matching and inserts into non-nullable table columns could cause.

## 9.5 Complex Schemas

Schema scenarios are usually a bit more complex than the simple one used to illustrate the general usage in the earlier topics. In this topic are some tips about approaching the issues when schemas involve n-way replications and heterogeneous scenarios.

### Peer to Peer or Bi-Directional Replication

Bi-directional replication is used when source databases need to send changes to target databases and vice-versa. (The target databases become source databases to the original sources, which become targets). Although the concept is straightforward enough, care needs to be taken to prevent changes from “bouncing” backwards and forwards in a continuous loop.

A change made to a source database gets replicated to a target. If the target also acts as a source database to the original source (now a target), this change will get picked up on the target and be replicated back, and so on.

### The REPL User

The solution to the looping problem is to have the user REPL defined on the servers.

REPL is a “special” user within the IBP Replicator application, as IBP Replicator recognises REPL as “myself”. Any change made by a REPL user will not get replicated to the target database.

**NOTE** To use the REPL user you do need to set the user up in the server's authentication system with the appropriate privileges it needs to do its job.

## Replicating Subsets of Data

Some scenarios require the rows that are to be replicated from one database to another to be restricted. For example, in a Head Office (London) to Branch Office scenario (Leeds), the branch office needs only to receive replication of data applicable to Leeds—not London specific data, nor data specific to other branches.

There are two ways to implement this in IBP Replicator:

- The simplest is to place a row-level replication condition on each table to exclude all but the applicable rows.
- The other method—usually used for very complex situations—is to replicate all data to a Stored Procedure on the target database and let the logic of the rules implemented in the procedure target database decide whether it wants to receive a row. Refer to the previous topic, [Advanced Mapping Techniques](#), for some guidelines for writing such procedures.

### Example—Row-level Replication Condition

To illustrate the 'row level replication condition', we will use the following scenario as an example.

The head office is in London, and there are two branch offices in Leeds and Manchester. Sales data needs to be replicated from the branch offices to the head office, and certain other specific data needs to be replicated from the head office to the branch offices.

The Leeds branch office is designated as DeptNo=10, whilst Manchester is Deptno=20.

Manchester (source) -> All Data -> London (target) <- All Data <- Leeds (source)

Manchester (target) <- DeptNo= 20 <- London (source) -> DeptNo=10 -> Leeds (target)

On the Leeds target database, we need to define a row level replication condition for each table, based on the DeptNo. This is done via in the table settings in the Table mapping properties dialog.

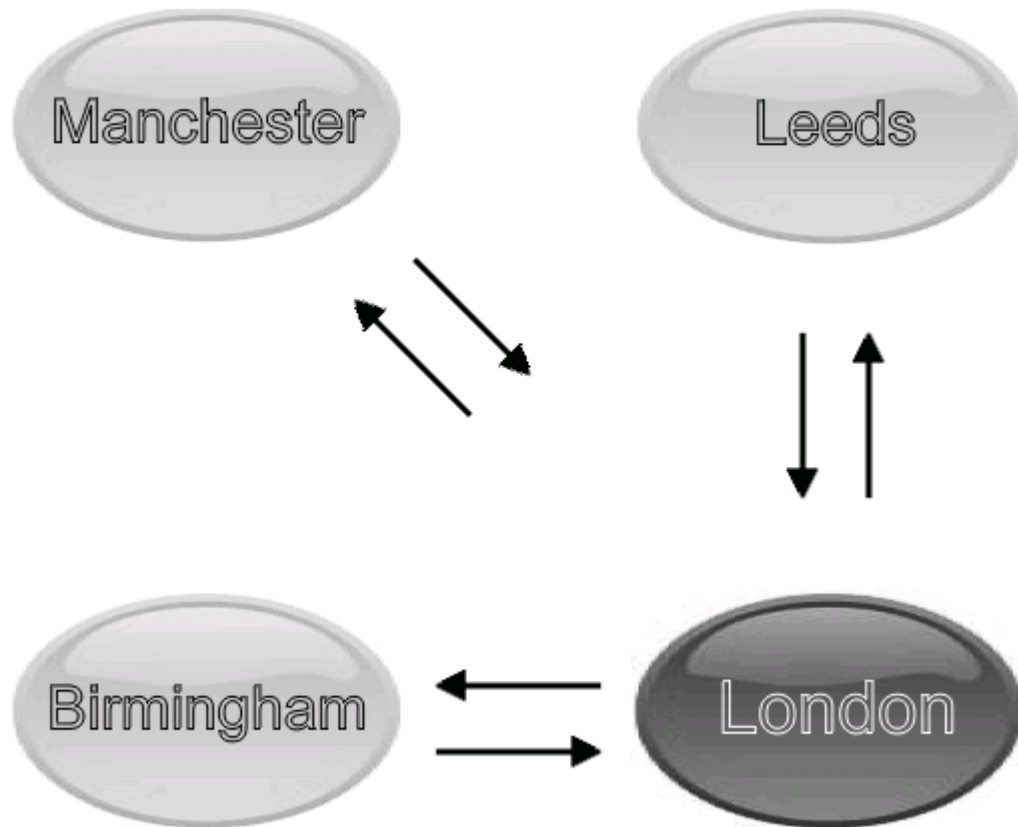
:DeptNo=10

Note the colon(:) in front of the field name.

:DeptNo=20 would be applied in the same way to the Manchester target database.

### Example—Hub and Spoke Replication

In this scenario a central database controls the replication, fetching and sending data from spoke databases to and from the hub.



This means that London acts both as a source and as a target for Leeds, Manchester and Birmingham.

In this illustration, London (the central database) fetches changes made by Manchester and forwards those changes onto Leeds and Birmingham.

The same is true for Leeds and Birmingham.

To prevent changes from Manchester/Leeds/Birmingham bouncing back to Manchester/Leeds/Birmingham, respectively, through London, you would normally use the special user REPL in a simple Peer to Peer schema. However that generates a problem, because we do want to forward Manchester's changes to Leeds and Birmingham, Leeds changes to Manchester and Birmingham and Birmingham's changes to Leeds and Manchester.

In this case we cannot use the username REPL, since data would (for example) replicate from Leeds to London, but would not get forwarded to the other branches. On the other hand, we do not want data to be replicated from Leeds to London and then from London back to Leeds.

Our solution is to have more than one dedicated "replication user" in our systems, e.g. REPL\_BIR, REPL\_LEE and REPL\_MAN for our three spokes, and use the applicable alternative replication user as our username.

### The Source Database Triggers

CAVEAT :: These examples are for guidance toward a custom solution to a complex business problem. They are in no way intended to be templates or a recommendation to meddle with the generated System Objects.

Use [row-level replication conditions](#) to achieve the effects indicated in the examples.

Trigger sets created in the source database for London (for each target) would be modified by [row-level conditions](#), to take the localised username into account when determining what to replicate:

```
CREATE TRIGGER REPL$_XYZ FOR ATABLE
ACTIVE AFTER INSERT POSITION 32760 AS
BEGIN
  IF (USER<>'REPL') THEN /* row-level condition :USER<>'REPL' */
  BEGIN
    IF (USER<>'REPL_MAN') THEN /* row-level condition :USER<>'REPL_MAN' */
    BEGIN
      INSERT INTO REPL$LOG (....)
      VALUES(....);
    END
  END
END
END
```

This logic can be tabulated as follows:

Source	Username	Target	Username	Condition
Manchester	REPL	London	REPL_MAN	
Birmingham	REPL	London	REPL_BIR	
Leeds	REPL	London	REPL_LEE	
London	REPL	Manchester	REPL	USER <> 'REPL_MAN'
London	REPL	Birmingham	REPL	USER <> 'REPL_BIR'
London	REPL	Leeds	REPL	USER <> 'LEE'

This ensures that data is replicated from one branch into the hub in London. The hub then replicates this data to the other branches, but not back to the originating branch.

### Replication Schema Definition

The defined replication schema for such a scenario would look something like this:

#### Manchester To London

Source Database in Manchester – UserName REPL

Target Database in London – UserName REPL\_MAN

#### Birmingham To London

Source Database in Birmingham – UserName REPL

Target Database in London – UserName REPL\_BIR

#### Leeds To London



Source Database in Leeds– UserName REPL

Target Database in London – UserName REPL\_LEE

London to the Rest

Source Database in London – UserName REPL

Target Databases

Manchester – UserName REPL, Row Level Replication Condition USER<>'REPL\_MAN'

Birmingham– UserName REPL, Row Level Replication Condition USER<>'REPL\_BIR'

Leeds – UserName REPL, Row Level Replication Condition USER<>'REPL\_LEE'

## 9.6 Off-line Replication and Synchronization

It is possible to perform what is termed "off-line replication", where one party in the replication is a database and the other party is a file. It enables you to do a form of asynchronous replication between two databases that are not in a connected client/server transaction.

This feature will be useful where it is not possible to maintain a connection between the source server and the replicant database that is sufficiently fast or reliable. At the source server you replicate "outbound" to a target file, transport the file, or a batch of files, to the target server by your preferred means (SFTP, HTTP or some transportable medium such as DVD or flash card). At the destination, you have a file defined as the "inbound" source and the replication server will pull the replications into the target database.

On each server, you will need replicant licences for both the source or target database and the outbound or inbound file, respectively.

### Setup Steps

#### Configure Directory Locations for the Files

On the servers, create the appropriate directories to which your offline replications are to be written (e.g. outbound) and/or from which files are to be read (e.g., inbound). If your replications are going to be one-way, you will not need both.

The Outbound and Inbound directories can be any file system location to which the replserver process is allowed to write directly, including a mountable network directory that is on-line. You can write to a removable storage device such as a flash drive or a media card. When writing to mountable or removable storage devices, it is your responsibility to ensure that the device is on-line whenever it needs to be accessed—which includes any time when you are configuring the "pseudo" database—and that the device has enough available capacity for the file[s] that will be written.

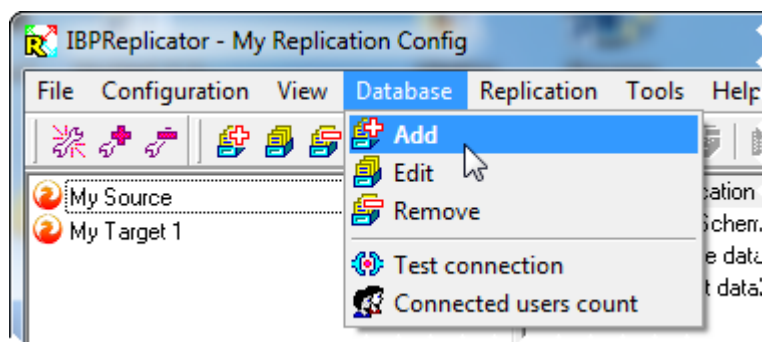
## Archiving

From Replicator 4.2 onward, it is possible to preserve the files after replication completes, rather than have them deleted, for archiving or other purposes. If you plan to use this feature, it would be a good idea to set up an archive location as well. Preserved files are renamed with the extension ".done".

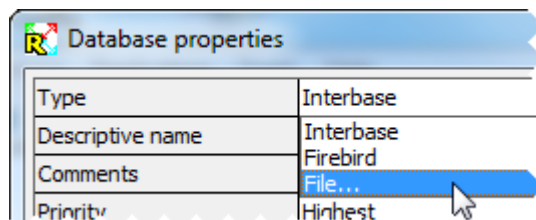
## Register Directory as a "Database"

In the examples that follow, we will set up an Outbound offline replication (Source database to file). For setting up and Inbound replication (file to Target database) the procedure is similar.

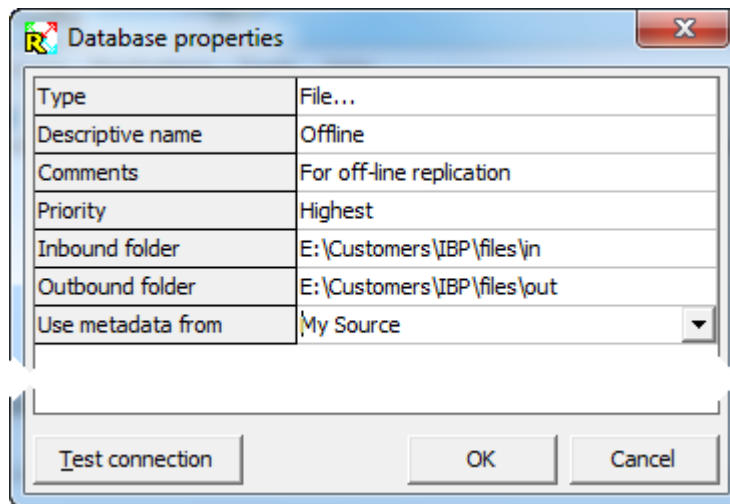
Use a selection method to invoke the database registration tool:



In the Database Type list, choose File...



Decide on a friendly descriptive name for this pseudo database, e.g. "Offline", "Outbound" or whatever you like. Fill in the fields, defining the full paths to the "inbound" and/or "outbound" folders and selecting the database from whose metadata you want to "borrow" metadata definitions:



Click OK to save the configuration so that the pseudo database is registered.

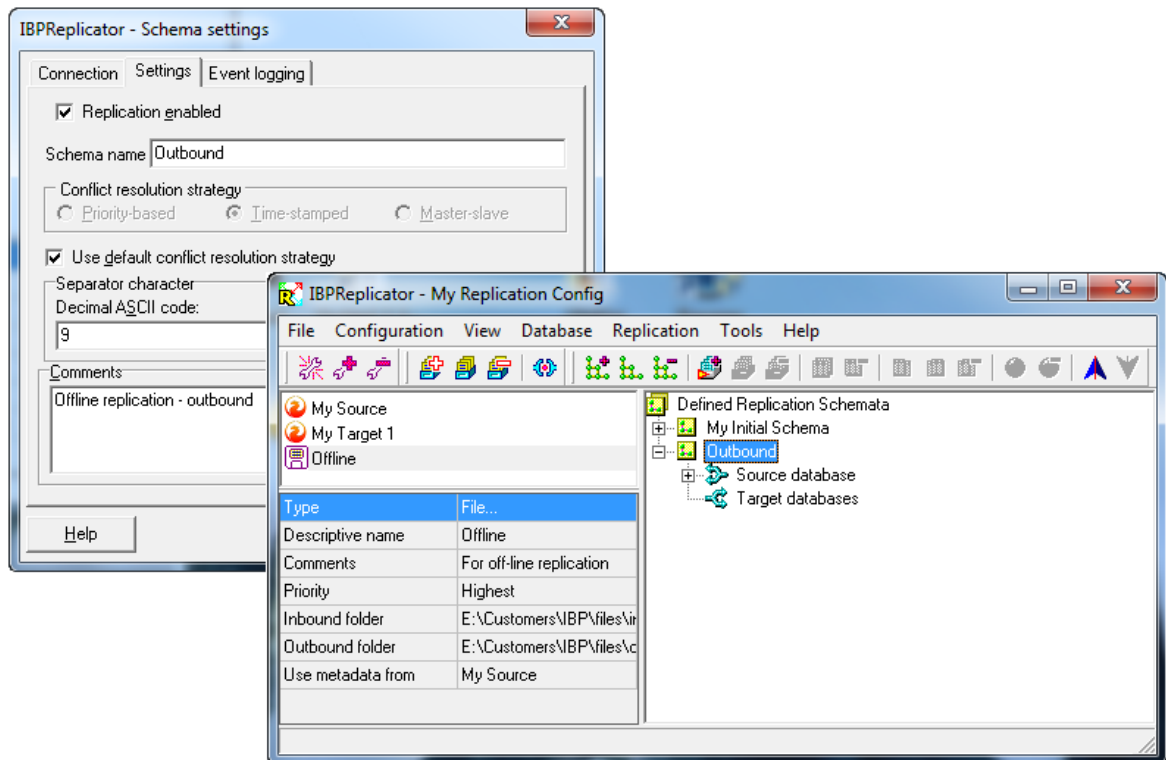
Now you are ready to begin defining your offline replication schema.

### 9.6.1 Configuring the Offline Schema

Configuring an offline replication schema is virtually identical to configuring an online one.

For simplicity, we'll assume here that your purpose in using offline replication is to substitute the 2-tier database-to-database schema entirely. It is possible to include an offline replication in an existing schema to meet a specific need. You can ask in the users' support list about any aspects of your specific case that might not be obvious.

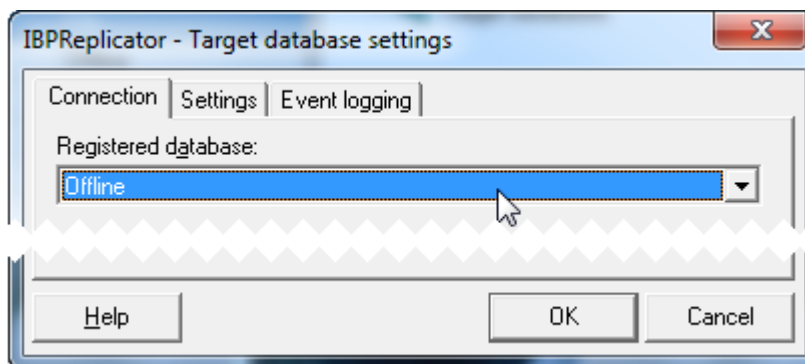
Create a new schema and assign the database that you want to be the Source.



Now, you are ready to set up the file that you prepared as a pseudo database previously to be the target of this outbound replication. Select the Target node and choose Add.

### Setting Up the Outbound Target

Choose the Offline target and complete the dialog:



You may click the Test connection button to confirm that the database you named as the one from which to borrow definitions is online.

### Set Up Tables, Keys and Fields

Proceed to set up keys and fields just as you would for the target of a database-to-database replication. When you are done, you are ready to create the system objects.

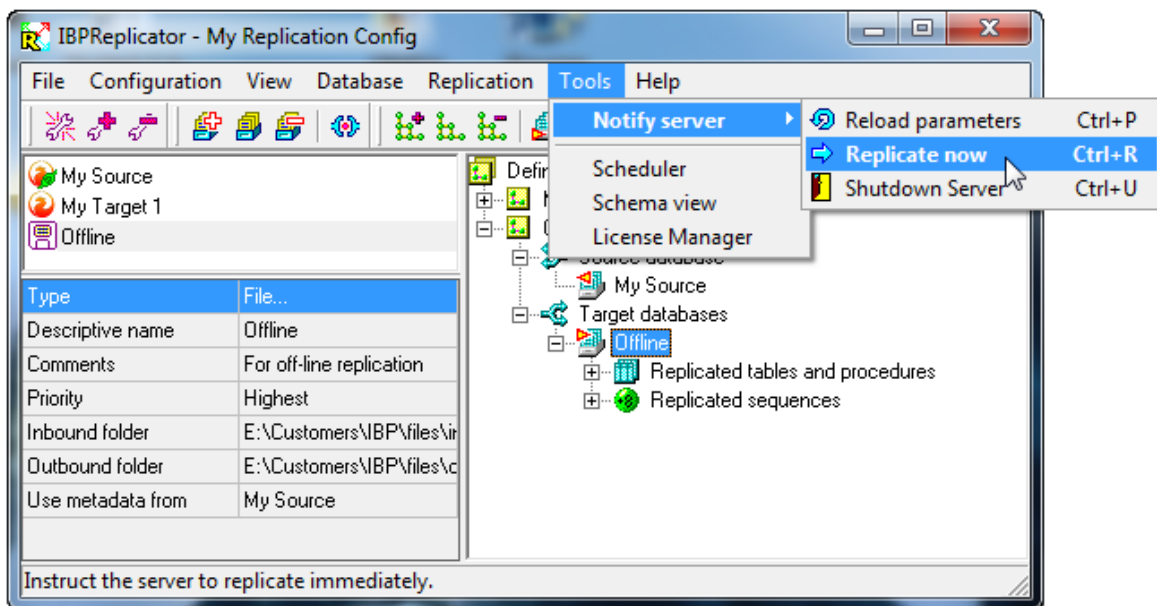
### Create System Objects

Again, this step behaves just as you would expect if you were setting up a database-to-database schema. Creating the system objects in the Source database defines the procedures and permissions needed to write records to the REPL\$LOG table and identifies the mappings that will be rendered to the destination file.

### 9.6.2 Running an Offline Replication

An offline replication can be run in any of the normal ways, according to your requirements:

- under the replication service or application by a repeating or Once-only schedule
- as a one-off in a Replication Manager session, through the Tools menu:



- in Windows, as a one-off or scheduled operation from the Replicator application interface (replserver.exe):



## Running the Replication

Now, the replication is ready to go, just like a normal 2-tier replication. You can verify it by checking the Outbound directory and the replication log.

In the Outbound directory, a fresh file is created for each replication cycle. The file names are of the form Rxxxxxxx.dat, where the x's are an alphanumeric representation of a number. The initial value of the number is calculated from the CURRENT\_TIMESTAMP value at the launch of the Replication server and is incremented by 1 for each successive file.

During replication, the file name has the suffix '.tmp'. If the transaction is rolled back for some reason, this temporary version of the file is deleted. Should something untoward happen that leaves the file in place with this .tmp suffix, do not use it.

## What Comes Next?

Once the file has been renamed with the '.dat' suffix, it is ready for its journey to the target server. That file must be used to replicate to a database in a schema that is within a different configuration database. IBP Replicator does not provide any tools for porting the offline files between the servers: it is your responsibility to design and set up a transport mechanism to suit your conditions.

The next section contains brief instructions for configuring the target server to receive the inbound file.

### 9.6.3 Inbound Offline Replication

Setting up and running the inbound side of offline replication is similar to what you did in setting up the outbound side. You configure it in a configuration database at the target side. You must, of course, have a replication server licence with two replicant licences available there.

It is strongly recommended that you read the whole of the Offline Replication topic. The outbound side of offline replication is described in detail. Since the procedure at each

side is almost identical, this section is merely a brief overview of the inbound side.

## Databases

At this side, the on-line database is configured as the Target, while the 'pseudo-database' inbound directory will be configured with Files... as the Source.

## Schema Configuration

Configuring the schema for the inbound replication will be the reverse of what you did when configuring the outbound one. No system objects are created in the on-line Target for this replication.

## Preparing for an Inbound Replication

The IBP Replicator package does not provide any tools for porting the offline files between the servers. It is up to you to set up a mechanism for picking up a file from the source server and delivering it to the configure Inbound directory.

Do not overlook the fact that the inserts, updates and deletes replicated from the file will be timestamped with the source server's timestamps, which may have a bearing on their age relative to the timestamps on any records affected by the inbound replication and/or on any REPL\$LOG records awaiting a downstream replication.

## Replicating from an Offline Synchronization Set

Offline synchronization generates a set of files containing all of the data from tables specified for an offline replication. After such a synchronization, the specified outbound "database" directory will contain one file for each table specified. The off-loaded files are named as "STableName.synch", where TableName is the name of a table. The binary format of each .synch file is the same as that used for the replication ".dat" file. If the ".synch" extension on these files is renamed to ".dat", the Replicator can process them as an off-line replication.

## Gotchas

An original "Rnnnnn.dat" file is the output from processing all of the specified tables, in the order specified when the replication was defined. However, offline replication does not respect the order in which tables were defined: the replicator service simply starts at the beginning of the directory and works its way through it in alphabetical order, deleting each file as it finishes with it.

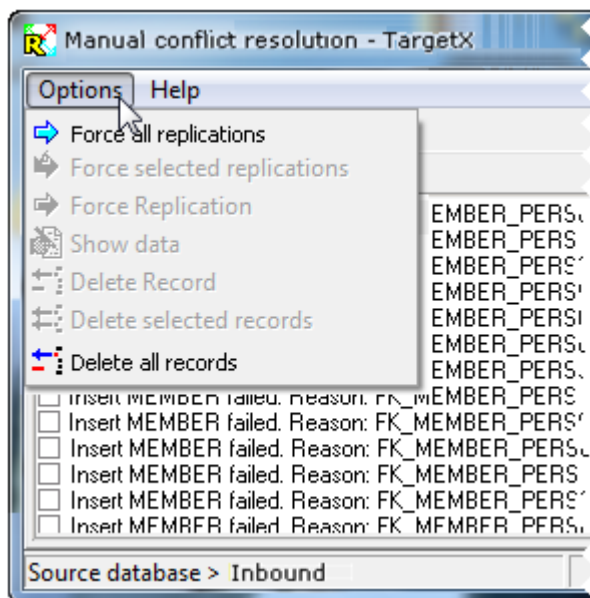
The "architectural" layout of a set of renamed .synch files in a filesystem directory is obviously quite different to the internal architecture of an offline replication file. When the replication service processes a set of files, it has no way to know either which tables it is processing or what is the correct dependency order. If you need the files to be processed in a specific order, you also need to rename the main part of each file name so that their alphabetical order reflects their dependency order.

## Manual Conflict Resolution After Off-line Replication

It is possible to perform a manual conflict resolution following an inbound replication from a Files.. type "database". Conflicts are stored in files with the suffix ".conflict" in the

inbound directory. There will be one such file for each conflict, which will be listed by the Conflict Resolution tool if it is invoked on the "database" that is defined as the offline replication source.

Conflict files are written during the up-replication, as each conflict occurs. Since the data in the conflict files is not under transaction control, a rollback of the replication does not delete any conflict files that were created before the rollback. Thus, careful management of these files is needed, to avoid accidentally processing the same conflicts more than once.



#### 9.6.4 Off-line Synchronization

Synchronization allows the IBP Replicator to get the source and target tables into the state where data matches in both, i.e. a one-time "global" replication. It is possible to perform an asymmetric synchronization off-line.

#### How Off-line Synchronization Works

The target table is synchronized using data off-loaded to files from the source database, one table per file. A "pull" synchronization from the files at the target server results in target tables that are exact copies of the source tables. IBP Replicator performs any necessary inserts and deletes. It is similar to a data pump operation, except that the target table need not be empty. Any records in the target table that do not exist in the source are deleted.

These "\*.synch" files can be saved for use—or re-use—as backups for point-in-time recovery for a whole database or just selected tables. Each time a file is re-used it will reset the target table to the state of the source table when the file was off-loaded.

#### Synchronization vs Replication



The synchronization files have the same structure as files created for off-line replication. The off-loaded files are named as "STableName.synch", where TableName is the name of a table. By renaming the ".synch" extension to ".dat" and passing them to the Replicator process as an off-line replication, you can preserve the pre-existing records in the target table.

For more information about replicating from .synch files, refer to the topic [Replicating from an Offline Synchronization Set](#) in the previous section.

### Synchronizing Multiple Tables

You can synchronize all of the tables configured for the target database ("whole database") or just one table. If you have more than one table to synchronize, although not the entire database, the synchronizations need to be performed one by one. There is no way to "multi-select" a sub-set of tables.

### "Missing" Tables

If the whole database is selected for synchronization and there is a mismatch between the source table names available and those configured for the target database, a stop error will occur and the synchronization will be rolled back.

### Operating System Environment

Since off-line synchronization is a routine of Replication Manager, you will need a Windows machine as the client. For Linux users with no Windows machine available, it should be possible to run a Replication Manager instance in the Wine emulator.

### File Names on Non-Windows Servers

Suppose you have a table created with CREATE TABLE "Holy Mackerel"... The Replication Manager, running on Windows, will off-load a synchronization file named "SHoly Mackerel.synch". Windows does not care about case-sensitivity or the presence of spaces in a name. You will run the up-load part of the operation from Replication Manager as well; so the application will not be bothered by either issue.

Where there might be an issue with such file names is in a server environment that requires them to be single-quoted in entirety, e.g. 'SHoly Mackerel.synch' or to have spaces and other special characters escaped, e.g., SHoly\ Mackerel.synch. You might bump into such an issue if you are on-loading and moving files around at the command line on some POSIX platforms.

### Non-ANSI Table Names

If non-ANSI characters were used in defining tables in either database, there is the possibility of problems occurring with the names of the OLS files at filesystem level if the target platform is not configured to interpret those characters correctly. This is not Replicator's problem, per se, but it will need to be checked to ensure that your OLS can proceed smoothly.

### Character Sets

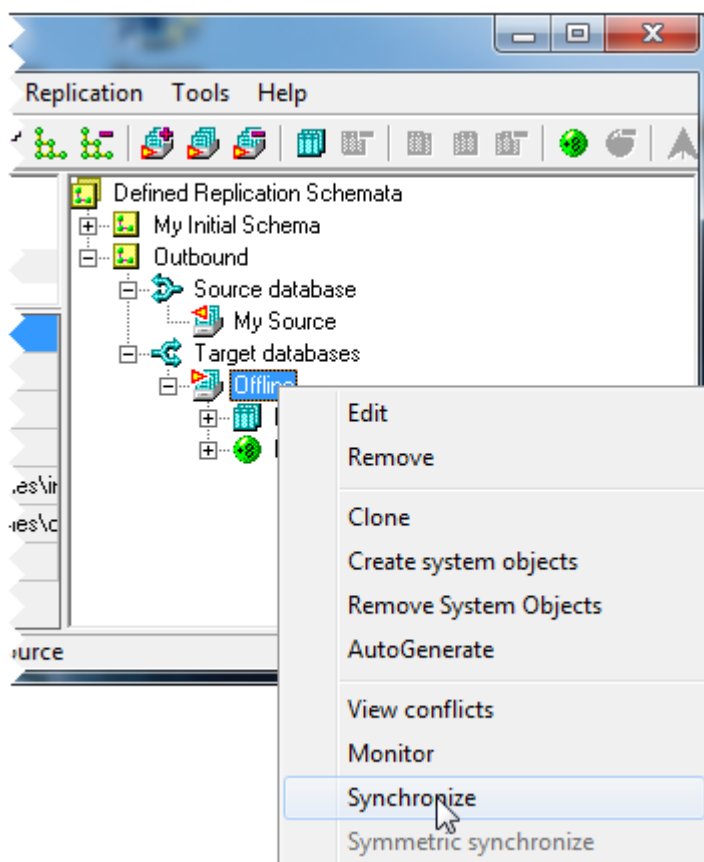
The off-loaded files are in a binary format so there should not be any character set issues

at application level. If you are using OLS to populate a new, empty database, make sure that the character set and collation attributes of the target database and all relevant objects in it are the same as those in the source database.

## Running an Off-load

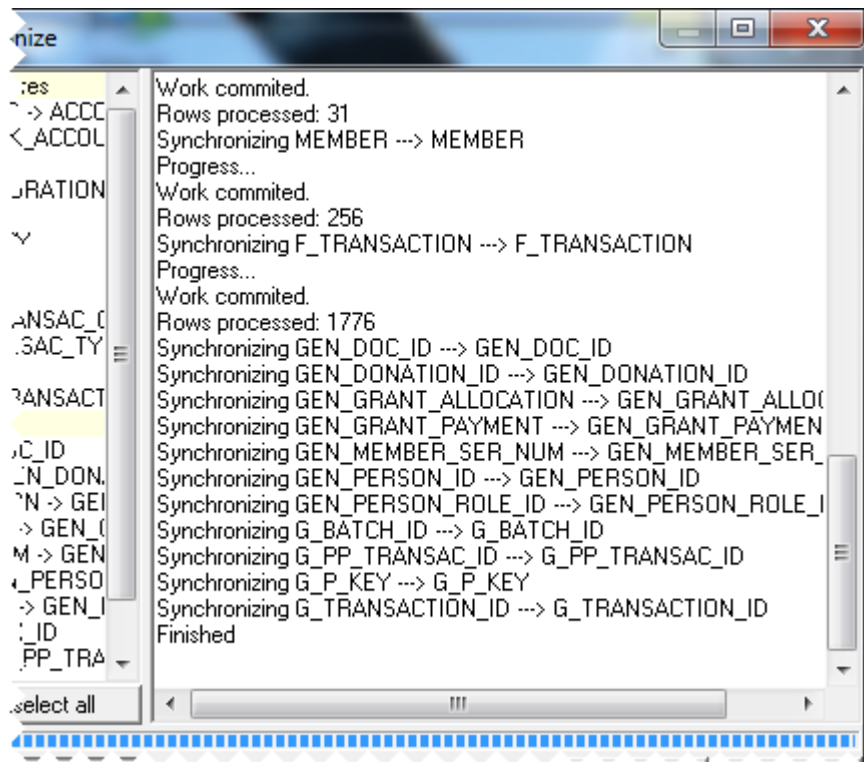
As with off-line replication, you need to define a target "database" that is a folder somewhere in your filesystem. If you have already set one up for off-line replication, it is possible to use that. Otherwise, go to the start of this section for instructions on how to do this setup.

Once this target structure exists, the approach for an off-line synchronization is no different from that for an on-line one. In the Replication tab, select the offline "database" and right-click on it. Select Synchronize.



Notice that the Symmetric synchronize option is disabled, for obvious reasons.

As with the online synchronization, A monitor window will pop up and display the synchronization as it happens. The window will fill up very fast but you can scroll back to check what has been done:



When it is done, you can check the contents of your "outbound" directory. There should be one ".synch" file for each table defined in the replication.

## Synchronizing from an Off-load

For synchronizing from an off-load, just as with replicating from one, you need a replication server licence and two replicant licences available on the server where the database to be synchronized is located. Define a replication in the config file at the target replication server.

To use the off-loaded files to synchronize a database, set up a File.. type of database as the source, pointing to the folder where the Replication Manager can find your off-loaded files. If you already have a schema set up for an offline up-replication, simply use the same schema. It won't interfere with any waiting replication, since the file extensions are different.

When you are ready, just right-click on it in the tree panel and choose Synchronize. The monitor list will pop up and show you what it is doing.

The up-synchronization will stop and roll back if dependency problems are encountered. Make sure that the table order defined in the target is correct for the referential integrity dependencies in your database: parent tables must precede their children. Use the Up and Down buttons in the toolbar to rearrange the table order.





# Chapter 10

Appendices

## Appendices

### 10.1 Appendix I: Supported Data Types and Compatibilites

Table 1: Interbase/Firebird

		Destination types													
		ARRAY	BOOLEAN	BLOB	CHAR	DATE	DECIMAL	DOUBLE PRECISION	FLOAT	INTEGER	NUMERIC	SMALLINT	TIME	TIMESTAMP	VARCHAR
Source types	ARRAY	X <sup>1</sup>	-	-	-	-	-	-	-	-	-	-	-	-	-
	BOOLEAN <sup>2</sup>	-	X	-	-	-	-	-	-	-	-	-	-	-	-
	BLOB <sup>3</sup>	-	-	X	X	-	-	-	-	-	-	-	-	-	X
	CHAR	-	-	X	X	X	X	X	X	X	X	X	X	X	X
	DATE	-	-	-	-	X	-	-	-	-	-	-	-	X	-
	DECIMAL	-	-	-	-	-	X	X	X	X	X	X	-	-	-
	DOUBLE PRECISION	-	-	-	-	-	X	X	X	X	X	X	-	-	-
	FLOAT	-	-	-	-	-	X	X	X	X	X	X	-	-	-
	INTEGER	-	-	-	-	-	X	X	X	X	X	X	-	-	-
	NUMERIC	-	-	-	-	-	X	X	X	X	X	X	-	-	-
	SMALLINT	-	-	-	-	-	X	X	X	X	X	X	-	-	-
	TIME	-	-	-	-	-	-	-	-	-	-	-	X	X	-
	TIMESTAMP	-	-	-	-	X	-	-	-	-	-	-	X	X	-
	VARCHAR	-	-	X	X	X	X	X	X	X	X	X	X	X	X

1 No datatype or dimension transformation. Total size of data must fit available RAM.

2 In Interbase only

3 Segmented BLOBs only. Streamed BLOBs are transformed to segmented.

Table 2: Oracle

		Destination types													
		BINARY_DOUBLE	BINARY_FLOAT	BLOB	CHAR	CLOB	DATE	INTERVAL DAY TO SECOND	INTERVAL YEAR TO MONTH	NUMBER	RAW	TIMESTAMP	TIMESTAMP WITH LOCAL TZ	TIMESTAMP WITH TIME ZONE	VARCHAR2
Source types	BINARY_DOUBLE	X	X	-	-	-	-	X	X	X	-	-	-	-	X
	BINARY_FLOAT	X	X	-	-	-	-	X	X	X	-	-	-	-	X
	BLOB	-	-	X	-	X	-	-	-	-	-	-	-	-	-
	CHAR	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	CLOB	-	-	X	-	X	-	-	-	-	-	-	-	-	-
	DATE	-	-	-	-	-	X	-	-	-	-	X	X	X	-
	INTERVAL DAY TO SECOND	-	-	-	-	-	-	X	-	X	-	-	-	-	X
	INTERVAL YEAR TO MONTH	-	-	-	-	-	-	-	X	X	-	-	-	-	X
	NUMBER	X	X	-	-	-	-	X	X	X	-	-	-	-	X
	RAW	-	-	X	-	-	-	-	-	-	X	-	-	-	X
	TIMESTAMP	-	-	-	-	-	X	-	-	-	-	X	X	X	X
	TIMESTAMP WITH LOCAL TZ	-	-	-	-	-	X	-	-	-	-	X	X	X	X
	TIMESTAMP WITH TIME ZONE	-	-	-	-	-	X	-	-	-	-	X	X	X	X
	VARCHAR2	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 3: Interbase/Firebird – Oracle

		Destination types													
		BINARY_DOUBLE	BINARY_FLOAT	BLOB	CHAR	CLOB	DATE	INTERVAL DAY TO SECOND	INTERVAL YEAR TO MONTH	NUMBER	RAW	TIMESTAMP	TIMESTAMP WITH LOCAL TZ	TIMESTAMP WITH TIME ZONE	VARCHAR2
Source types	ARRAY	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	BOOLEAN	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	BLOB	-	-	X	-	X	-	-	-	-	-	-	-	-	-
	CHAR	X	X	X	X	X	X	X	X	X	X <sup>1</sup>	X	X	X	X
	DATE	-	-	-	-	-	X	-	-	-	-	X	X	X	-
	DECIMAL	X	X	-	-	-	-	X	X	X	-	-	-	-	-
	DOUBLE PRECISION	X	X	-	-	-	-	X	X	X	-	-	-	-	X
	FLOAT	X	X	-	-	-	-	X	X	X	-	-	-	-	X
	INTEGER	X	X	-	-	-	-	X	X	X	-	-	-	-	X
	NUMERIC	X	X	-	-	-	-	X	X	X	-	-	-	-	-
	SMALLINT	X	X	-	-	-	-	X	X	X	-	-	-	-	-
	TIME	-	-	-	-	-	X	-	-	-	-	X	X	X	-
	TIMESTAMP	-	-	-	-	-	X	-	-	-	-	X	X	X	-
	VARCHAR	X	X	X	X	X	X <sup>2</sup>	X <sup>3</sup>	X	X	X	X	X	X	X

1 Fields with character set OCTETS are copied as-is, others must be in hexadecimal form.

2 For DATE and TIMESTAMP types string is expected in ANSI format, not current NLS.

3 For INTERVAL types string must fit current client NLS format.



Table 4: Oracle – Interbase/Firebird

		Destination types													
		ARRAY	BOOLEAN	BLOB	CHAR	DATE	DECIMAL	DOUBLE PRECISION	FLOAT	INTEGER	NUMERIC	SMALLINT	TIME	TIMESTAMP	VARCHAR
Source types	BINARY_DOUBLE	–	–	–	–	–	X	X	X	X	X	X	–	–	–
	BINARY_FLOAT	–	–	–	–	–	X	X	X	X	X	X	–	–	–
	BLOB	–	–	X	X	–	–	–	–	–	–	–	–	–	X
	CHAR	–	–	X	X	X	X	X	X	X	X	X	X	X	X
	CLOB	–	–	X	X	–	–	–	–	–	–	–	–	–	X
	DATE	–	–	–	–	X	–	–	–	–	–	–	–	X	–
	INTERVAL DAY TO SECOND	–	–	–	X	–	X	X	X	X	X	X	–	–	X
	INTERVAL YEAR TO MONTH	–	–	–	X	–	X	X	X	X	X	X	–	–	X
	NUMBER	–	–	–	X	–	X	X	X	X	X	X	–	–	X
	RAW	–	–	X	X	–	–	–	–	–	–	–	–	–	X <sup>4</sup>
	TIMESTAMP	–	–	–	X	X	–	–	–	–	–	–	X	X	X
	TIMESTAMP WITH LOCAL TZ	–	–	–	X	X	–	–	–	–	–	–	X	X	X
	TIMESTAMP WITH TIME ZONE	–	–	–	X	X	–	–	–	–	–	–	X	X	X
	VARCHAR2	–	–	X	X	X	X	X	X	X	X	X	X	X	X

4 If a destination field has character set OCTETS, its content is copied as-is; otherwise it is converted into hexadecimal representation.



Native data types in ODBC targets are not supported

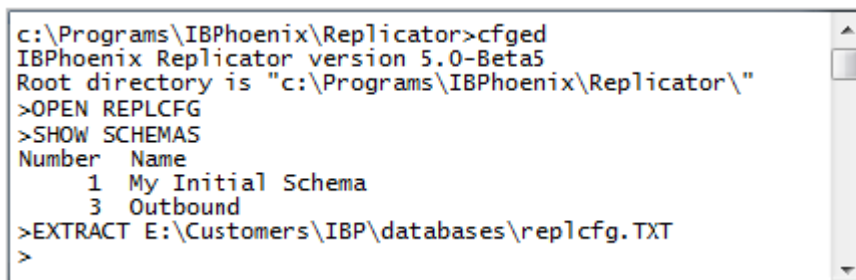
## 10.2 Appendix II: Command-Line Editor: cfged

The command-line application cfged (cfged.exe on Windows) enables the Replicator-savvy administrator to create and manage a replication configuration without using Replication Manager. It can be used interactively from a simple text console or in batch or cron jobs. It has obvious benefits if you need to set up the same configuration on multiple sites.

For Linux installations, cfged provides a Windows-free way to configure replication. For all platforms, it provides a dedicated toolset for making ad hoc modification of configurations.

### Starting cfged

Using a graphical window or a shell on Linux, or a command window on Windows, change to the directory where the Replicator executables are located and enter the command cfged:



```
c:\Programs\IBPhoenix\Replicator>cfged
IBPhoenix Replicator version 5.0-Beta5
Root directory is "c:\Programs\IBPhoenix\Replicator\"
>OPEN REPLCFG
>SHOW SCHEMAS
Number  Name
      1  My Initial Schema
      3  Outbound
>EXTRACT E:\Customers\IBP\databases\replcfg.TXT
>
```

### The OPEN command

The OPEN command opens a configuration for editing. The first parameter, which is the only one not accompanied by a keyword, is the absolute path or (for Firebird, the alias) for the location of the configuration database. In this screenshot, a Firebird configuration database with the alias REPLCFG has been opened.

Any parameter that is omitted, other than ROLE, will be taken from the default configuration, if it is set.

Other parameters can include

TYPE – with either INTERBASE | ORACLE | FIREBIRD as the argument, as appropriate.

USER – either the database owner or the user name of a user with SYSDBA privileges. If you need this, you will also need to include PASSWORD along with the password as its argument.

ROLE role\_name if the user credentials are tied to a role for access to configuration structures.

SILENTLY – can be included to stop cfged from showing warnings about a missing configuration or a wrong version.

If a configuration is already open, it will be closed before the requested one is opened.

In the screenshot, two cfged commands have been submitted:

- SHOW SCHEMAS lists the schemas that have been defined in this configuration database
- EXTRACT file-destination requests that the entire configuration be extracted to the file path in the argument

TIP EXTRACT produces a full extract of the configuration as it stands currently, as a batch of cfged commands. It is handy not just as a quick review of the whole configuration but also as a quick tool for cloning a configuration.

## Common Rules

Although there is variation in the keywords, syntax and parameters for different commands (discussed later), a small set of rules is common to all commands.

1. Keywords are case insensitive. Parameters may be case sensitive or not, according to purpose and usage.

Boolean parameters are case-insensitive and may be in any of the following forms:

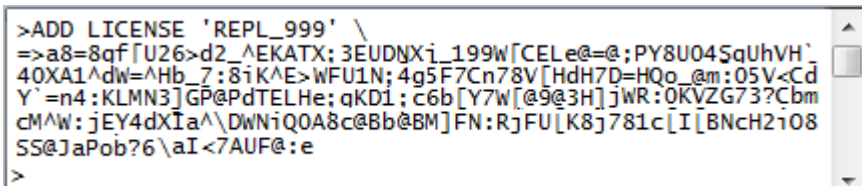
true|false, yes|no, on|off, y|n or 1|0.

2. Multi-line commands are allowed, as long as a backslash is included as a new line signal at the end of each line. For example, when adding a licence, which requires the licence ID and the licence key to be separated, you would include the backslash before pressing Enter:



```
>ADD LICENSE 'REPL_999' \
=>
```

The continuation sign => that appears is ready to receive the pasted licence key from the clipboard:



```
>ADD LICENSE 'REPL_999' \
=>a8=8qf[U26>d2_^EKATX;3EUDNXi_199w[CELe@=@;PY8U04SqUhVH`
40XA1^dw=^Hb_7:8iK^E>wFU1N;4g5F7Cn78V[HdH7D=HQo_@m:05V<Cd
Y`=n4:KLMN3]GP@PdTELHe;qKD1;c6b[Y7W[@9@3H]jWR:OKVZG73?Cbm
cM^w:jEY4dXIa^DWNiQ0A8c@Bb@BM]FN:RjFU[K8]781c[I[BNcH2i08
SS@JaPob?6\ai<7AUF@:e
>
```

3. Words must be separated by white space; for example

OPEN REPLCFG – good

OPENREPLCFG – rejected

4. Parameters that contain spaces, such as file paths, must be delimited by pairs of quotes (single or double) or by square brackets. For example:

```
INIT [C:\Customer Data\SMITHDB.FDB]
```

or

```
INIT "/usr/Customer Data/SMITHDB.FDB"
```



White space between quoted or bracketed segments is ignored.

If a closing quote or bracket at the end of an otherwise valid command is omitted, cfged will allow it.

## Command-line Switches

When you start cfged, in either interactive or direct mode, or within a batch script, some switches are available to control the way the program runs.

- b [ON|OFF] bail on error causes cfged to terminate if an error occurs. It is equivalent to the SET BAIL command in interactive mode. By default, bail on error is OFF.
- e [ON|OFF] controls command echo. It is equivalent to SET ECHO in interactive mode. By default, command echo is ON.
- h, -? [<command>|ALL] displays terse or detailed documentation. It is equivalent to HELP in interactive mode.
- q "be quiet" – instructs cfged to execute without returning output or errors to the console.

## Commands

The cfged application has a command vocabulary consisting of keywords for a number of verbs and operands and parameters. All commands start with a verb. Some verbs are self-contained while others require objects, parameters, or both.



The [ ] and < > symbols that appear in the Syntax sections are not syntactic elements.

## On-line Help

### HELP

Use HELP to display the cfged on-line documentation.

### Syntax

```
HELP [<command>|ALL]
```

Used alone, HELP displays some usage notes and a list of the available commands.

HELP <command> gives detailed help for the requested command.

HELP ALL gives detailed help for all of the commands, in alphabetical order.

### Equivalent switches -h & -?

The equivalent switches when passed from the command shell are -h and ? .

## Where Am I?

### SHOW CURRENT

Shows the numbers of the current database, schema, target database and table. It takes no parameters.

#### Syntax

```
SHOW CURRENT
```

### SET CURRENT

Set the specified object (DATABASE | SCHEMA | TARGET | TABLE) to be the current one of its type.

#### Syntax

```
SET CURRENT <object> [TO] <object number or name>  
<object> ::= DATABASE | SCHEMA | TARGET | TABLE
```

Take care with this command if specifying the object number as the argument, as no existence check is performed.



Neither is there any check to verify whether a specified table or target database belongs to the current schema.

## Program Settings

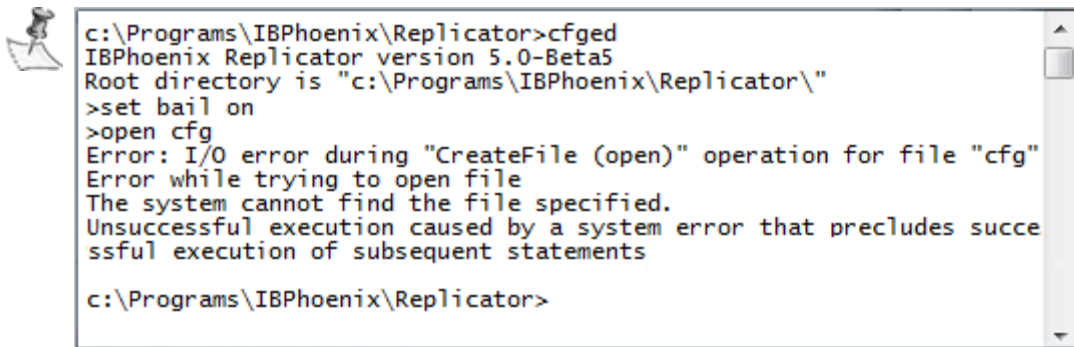
### SET BAIL

Controls whether cfged should exit immediately after an error occurs. Useful during script debugging, it is alternatively available as a command switch.

#### Syntax

```
SET BAIL <boolean value>
```

Use of SET BAIL ON in interactive mode is probably not very useful!



```
c:\Programs\IBPhoenix\Replicator>cfged
IBPhoenix Replicator version 5.0-Beta5
Root directory is "c:\Programs\IBPhoenix\Replicator\"
>set bail on
>open cfg
Error: I/O error during "CreateFile (open)" operation for file "cfg"
Error while trying to open file
The system cannot find the file specified.
Unsuccessful execution caused by a system error that precludes successful
execution of subsequent statements

c:\Programs\IBPhoenix\Replicator>
```

### Equivalent switch -b

The equivalent switch when passed from the command shell is -b.

## SET ECHO

Controls whether cfged displays executed commands. ECHO ON is useful in batch mode when standard input has been redirected.

### Syntax

```
SET ECHO <boolean value>
```

### Equivalent switch -e

The equivalent switch when passed from the command shell is -e.

## SET PROMPT

Controls whether cfged should display the input prompt. PROMPT OFF is useful in batch mode.

### Syntax

```
SET PROMPT <boolean value>
```

## Working with Configurations

### CREATE CONFIGURATION

Use to create a new configuration. If a configuration is already open, it will be closed.

For the Oracle driver, the command CREATE CONFIGURATION is equivalent to OPEN+INIT.

### Syntax

```
CREATE [CONFIGURATION] <cfg connection string>
[TYPE <cfg type>]
```

```
[USER <user name>] [PASSWORD <user password>]  
[ROLE <user role>]
```

The first parameter is the absolute path or (for Firebird, the alias) for the location of the configuration database. The keyword CONFIGURATION is optional.

The remaining parameters can be in any order. Any that is omitted will be taken from the default configuration. The default configuration, if it is set, is taken from the Windows registry or the Linux config file (usually /etc/IBPReplicator.conf).

### Other parameters

TYPE – with either INTERBASE | ORACLE | FIREBIRD as the argument, as appropriate.

USER – either the database owner or the user name of a user with SYSDBA privileges. If you need this, you will also need to include PASSWORD along with the password as its argument.

ROLE role\_name if the user credentials are to be tied to a role for access to configuration structures.

### DROP

Drop the configuration database. It takes no parameters or arguments.

### Syntax

```
DROP
```

**ORACLE** For the Oracle driver the command DROP is equivalent to CLEAR.

### EXTRACT

Export the current configuration into a file or a console. The resulting output is a script that can be used for cfged.

### Syntax

```
EXTRACT [[INTO] <file name>]
```

The keyword INTO is optional.

If the file name is not set, output will go to the console.

### INIT

Initialise configuration structures in a currently opened user database that is not a dedicated configuration database. This command takes no parameters.

### Syntax

INIT

## CLEAR

Use to clear a replication configuration from a user database without dropping the database. It takes no parameters or arguments.

### Syntax

CLEAR

## SET AS DEFAULT

Set the currently opened configuration as the default one.

### Syntax

SET [AS] DEFAULT



On Windows versions higher than XP and Server2000, cfged.exe must be running with Run As Administrator privileges for this command to work.

## UPGRADE

Upgrades the configuration structures in the currently opened database to the current version. Use this command after installing a new version of IBPReplicator over an older version (Replicator 3 or higher). It takes no parameters.

### Syntax

UPGRADE

## ADD LICENSE

Add a license to a configuration.

### Syntax

ADD LICENSE <license id> <license key>



Separate the license id from the license key with a backslash followed by Enter and paste the license key into the new line.

## REMOVE LICENSE

Use to remove a licence from a configuration.

### Syntax



```
REMOVE LICENSE license Id
```



You can use SHOW LICENSES to identify the licence you want to remove. You cannot remove an EVAL licence.

## SHOW LICENSES

Lists the licences installed in the configuration database. No parameters.

## Database Registration

### ADD DATABASE

Register a database in a configuration. The set of allowed parameters depends on the database TYPE (INTERBASE | ORACLE | FIREBIRD).

#### Syntax

```
ADD DATABASE [OF] TYPE <db type> <db params>
```

```
db type ::= INTERBASE | ORACLE | FIREBIRD | ODBC | FILE (Required).
```

```
db params ::= <param name> <param value> [<db params>]
```

#### Common parameters

This group of parameters is common to all database types:

NUMBER – database number. If not provided, it will be set from a sequence (generator).

NAME – descriptive name for the database. Required.



If you want to look up the numbers and descriptive names of existing registered databases, use SHOW DATABASES.

CHARSET – connection character set.

**ORACLE** If this parameter is not used, or is empty, the default character set from the environment variable NLS\_LANG is used.

COMMENT – Quoted or bracketed passage of text containing anything you like.

PRIORITY – database priority from 0 to 10. A higher number means higher priority.

KEEP\_CONNECTION – whether to keep RAS connection after a database disconnect.

KEY\_SIZE – the size of the field in the log tables to store the values of the key field (s).

USER – administrative user name. For a database used as a source for any schema the user must be one of following, to enable cfged to create the logging triggers on replicated tables:

- a) SYSDBA
- b) The Owner of all replicated tables
- c) A user that has been granted the RDB\$ADMIN role. In this case, the Role [below] must be registered as RDB\$ADMIN.

PASSWORD – administrative user password.

RAS\_NAME – the name of the RAS connection to dial before connect.

RAS\_PASSWORD – the password for the RAS connection.

RAS\_USER – the RAS connection user name.

SERVER – the entire connection string, including the host name and port number if applicable to the connection protocol. Required.

or

DSN – the name of the system DSN that has been configured for an ODBC target

TIME\_FIELD – the name of the timestamp field for timestamp-based conflict resolution strategy.

Parameter applicable to InterBase and Firebird only:

ROLE – administrative role. If the USER parameter is a regular user that has been granted the RDB\$ADMIN role, then RDB\$ADMIN is required here.

InterBase only:

DIALECT – SQL dialect for the connection. Must be AUTO (the default), 3 or 1. Don't use this parameter unless there is a real reason to do so.



Don't use this parameter for Firebird databases: Firebird selects the correct client dialect automatically.

Parameters specific to file:

INBOUND – the name of the inbound directory. The directory must exist. For an Alaverion server with embedded replication enabled, this should be the same directory that is configured as OutputDirectory in ibpr\_cdc.conf.

OUTBOUND – the name of the outbound directory. The directory must exist.

STRUCTURE – the number or descriptive name of the database to use as a source of metadata information.

AFTER APPLY Delete | Rename – Used for optionally preserving files after successful replication. The default is Delete. If Rename is supplied, files will be renamed with the extension ".done".

AUTO [REGISTRATION] – set table auto-registration option to ENABLED or DISABLED.



If you want to look up the numbers and descriptive names of existing databases, use SHOW DATABASES or SHOW SCHEMAS.

## MODIFY DATABASE

Modify database registration information. Rules and parameters follow those for ADD DATABASE (above). If the database number or name is omitted, the current database is modified.

### Syntax

```
MODIFY DATABASE [<db number or name>] SET <db params>

<db params> ::= <param name> <param value> [<db params>]
```

## REMOVE DATABASE

Remove the database registration info from a configuration. All replication schemas that use this database will also be removed.

### Syntax

```
REMOVE DATABASE [<db number or name>]
```



System objects associated with the removed schemas will remain.

## SHOW DATABASE

Shows the registration information about databases.

### Syntax

```
SHOW DATABASE [<db number or name>]
```

SHOW DATABASE without a parameter displays information about the current database.

SHOW DATABASE <db number or name> shows information about the specified database.

## SHOW DATABASES

Lists the names, numbers and types of all registered databases. It takes no parameters.

### Syntax

```
SHOW DATABASES
```

## Working with Schemas

### ADD SCHEMA

Add a new schema with a given set of parameters. Some parameters may be omitted. In that case default values will be used.

### Syntax

```
ADD SCHEMA <schema name> [<schema parameters>]
```

### Schema parameters

[NAME] <schema name> – your own user-friendly name for a new schema. It is required. Make sure it is different from other schema names in the configuration. Use of the keyword NAME is optional if the schema name is the first parameter.

NUMBER <schema number> – if you are managing your own schema-numbering system, use this parameter to provide a unique number. If this parameter is omitted, a new number will be autogenerated.

SOURCE <source database number or name> – provide either the number or the friendly name of a registered database. If this parameter is omitted, the current database will be used.



Although cfged will allow you to set an ODBC DSN as the source database for a schema, do not do it. The underlying DBMS engine is inaccessible to the application, making it impossible to create system objects.

ENABLED | DISABLED – By default, the new schema will become the active one. Use this parameter to control this option.

COMMENT <schema comments> – Option comments you can include with the schema.

USER <user name>, PASSWORD <user password>, ROLE <user role> – User name, password and (if required) the name of the role that is authorised to read the source database and write to the target databases to be included in this schema.

CRS <strategy number | DEFAULT> – Conflict resolution strategy to be used under this schema.

If DEFAULT is provided (or implied by being omitted), the global default strategy will be used when replication is performed under this schema.

If you provide the strategy number explicitly, then possible strategies are:

- 0 – priority-based
- 1 – timestamp-based
- 2 – master-slave (default)

LOG <log levels> | DEFAULT – one of two possible LOG parameters available. This one sets the logging levels and takes two integer arguments, for logging to the window (first argument) and logging to the log file (second argument). For example,

```
ADD SCHEMA 'My Initial Schema' SOURCE 1 USER REPL PASSWORD xxxxxxxx \
ROLE REPL_ADMIN CRS 1 LOG 3 4 LOG INISHEMA.LOG ORDER 0
```

specifies level 3 logging to the window and level 4 to the log file.



Log levels are from 0 (no log) to 5 for ordinary builds of the Replicator server. If you are running a debug build, the highest logging level is 7.

If DEFAULT is provided (or implied by being omitted), the global default level will be used for logging under this schema.

LOG <log file name> – allows the name of the log file to be customised. By default, it will be REPLICATE.LOG and you do not have to specify that.



You can use both LOG parameters in a single command, if need be.

SEPARATOR <separator character code> – Use this if you need to change the character that will be used in the replication log table to separate the segments of multi-segment keys in the text representation of records changed. By default, the value is 9 (indicating ASCII decimal 9, TAB). Don't change this unless there is a real reason to do so.

ORDER <number> – This parameter enables you to set the order of multiple schemas in your configuration, corresponding to moving them up and down in the Replication Manager program ([view topic](#)).

## MODIFY SCHEMA

Use to modify a replication schema. Parameters and arguments follow the same rules as ADD SCHEMA (above).

### Syntax

```
MODIFY SCHEMA [<schema name or number>] SET <schema parameters>
```



If the schema number or name is omitted the current schema is modified.

## REMOVE SCHEMA

Use to remove a replication schema from a configuration.

### Syntax

```
REMOVE SCHEMA [<schema number or name>] [SILENTLY]
```

If SILENTLY is used, any warnings about possible failures during the removal of the schema are suppressed.

## SHOW SCHEMA

Shows information about the schema. If the schema number or name is omitted, the information about the current schema is displayed.

### Syntax

```
SHOW SCHEMA [<schema number or name>]
```

## SHOW SCHEMAS

Lists all of the schemas in the configuration. No parameters.

## Working with Target Databases

### ADD TARGET

Add a new target database to the schema. If a schema or a database name is omitted current values are used.

### Syntax

```
ADD TARGET DATABASE <database name or number> <target DB parameters>
```

<database name or number> is mandatory and must be either the (case-sensitive) friendly name you used when registering the database or the database number that was auto-generated. You can check these using SHOW DATABASES.

### Target database parameters

[TO] [SCHEMA] <schema name or number> – if you have multiple schemas, use

this to designate the schema to which the new target is to be added. The keywords TO and SCHEMA are optional.

The three parameters USER <user name>, PASSWORD <user password> and (if required) ROLE <user role> are for the credentials of the replication user (normally REPL).

COMMIT <commit count> sets the number of records that constitute a batch that is to be committed, if and only if you need to use the "periodic commit" provision. Don't use this parameter unless it is actually necessary. It can impact the general performance of the database over time and breaking a replication into numerous separate transactions breaks the overall atomicity of a replication. For more information, see the topic [Periodic Commit](#).



If the conditions are sufficiently poor to necessitate resorting to periodic commit, it is recommended that you use offline replication instead.

CONDITION <row-level replication condition> – use for a row-level replication condition that needs to be applied to all relations in the target database. For more information about this parameter, refer to the topic [Row-level replication condition](#).

LOG <log levels> | DEFAULT – one of two possible LOG parameters available if you need to override the schema setting for this target. One sets the logging levels and takes two integer arguments, for logging to the window (first argument) and logging to the log file (second argument).



Log levels are from 0 (no log) to 5 for ordinary builds of the Replicator server. If you are running a debug build, the highest logging level is 7.

If DEFAULT is provided (or implied by being omitted), the level set for the schema will be used for logging replication to this target.

LOG <log file name> – allows the name of the log file for replications to this target to be customised.



You can use both LOG parameters in a single command, if need be.

ORDER <number> – This parameter enables you to set the order of replication when this target is not the only one. Use SHOW TARGETS to view the current order of replication per schema.

## MODIFY TARGET

Use to modify a specified target database in a specified replication schema. Parameters follow the same rules as for ADD TARGET (above).

### Syntax

```
MODIFY TARGET [DATABASE <db number or name>]  
[[OF] [SCHEMA] <schema number or name>]  
SET <target DB parameters>
```

### Target DB parameters

If the parameter DATABASE <db number or name> is omitted, the current database will be used.

If the OF SCHEMA parameter is omitted, the current schema will be used. The keyword OF is optional.

## REMOVE TARGET

Use to remove a target database from a replication schema.

### Syntax

```
REMOVE TARGET [[DATABASE] <db name or number>]  
[[FROM] [SCHEMA] <schema number or name>] [SILENTLY]
```



If the number or name properties of the TARGET or FROM SCHEMA parameters are omitted, the current ones are used. Be certain you are directing the changes to the right target! Use of the keyword FROM is optional.

If SILENTLY is used, there will be no warnings about any failures during the removal of the target.

## SHOW TARGET

Shows information about a target database.

### Syntax

```
SHOW TARGET [[DATABASE] <db number or name>]  
[[OF] [SCHEMA] <schema number or name>]
```

### Description:

If a database number or name is omitted, then the current target database is used.

If the schema number or name is omitted, then the current schema is used.



When specifying the schema explicitly, either of the keywords OF and SCHEMA may be omitted, but it is recommended to use one or both.

## SHOW TARGETS

Lists the target databases for one or all schemas.

### Syntax



SHOW TARGETS [[FOR [SCHEMA]] <schema number or name>]



In contrast with other commands, if schema number or name is omitted, target databases for all schemas will be shown, not just the current one.

## Working with Data Fields and Keys

### ADD DATA

Add a data field definition for a table.

#### Syntax

ADD DATA <field parameters>

#### Field parameters

**FIELD** source field name – The name of the column to add. '\*' means 'All fields'. The keyword FIELD is optional.

**NUMBER** field number – A number for the field that is unique amongst the fields in the replicated record. Use a number that is the same as the field number used by the database engine in the metadata of the target database.



It is not mandatory to have these field numbers matching but it does reduce the risk of errors when later editing the field definitions in the Replication Manager program.

**TO** target field name – The name of the field in the target record. If the target field name is not set, it is assumed to be the same as the source field name. Omit this parameter if the FIELD argument is '\*'.

**SCHEMA** schema number or name – required if the configuration has more than one schema

**TARGET** target database number or name – required if the configuration has more than one target database

**TABLE** table number or name – required if more than one table is replicated in the schema.

### MODIFY DATA FIELD

Use to modify data field definitions.

#### Syntax

```
MODIFY DATA [FIELD] target field name or number
[[OF] [SCHEMA] schema number or name]
[[TARGET] target db number or name]
```

```
[[TABLE] table number or target table name]
SET <field parameters>
```

The parameters for SET in this command follow the same rules as for ADD DATA (above).

## REMOVE DATA FIELD

Use to remove a non-key data field from a list of replicated fields in a record. The parameters and arguments follow the same rules as for ADD DATA and MODIFY DATA FIELD (above).

### Syntax

```
REMOVE DATA [FIELD] field number or name
[[OF] [SCHEMA] schema number or name]
[[TARGET] target db number or name]
[[TABLE] target table number or name]
```



Key fields cannot be removed with this command. The entire table must be removed to undefine a key field.

## SHOW FIELDS

Lists the replicated fields for the specified target database and table in the specified schema.

### Syntax

```
SHOW FIELDS [[OF] [SCHEMA]] <schema number or name>
[TARGET] <target db number or name>
[TABLE] <target table/SP number or name>
```



All parameters must be specified. The keyword OF is optional.

The keywords SCHEMA, TARGET and TABLE are optional but, if any keyword is omitted, the parameters must be in strict order.

## ADD KEY

Add a key field definition for a table.

### Syntax

```
ADD KEY <field parameters>
```

**NUMBER** field number – A number for the field that is unique amongst the fields in the replicated record. Use a number that is the same as the field number used by the database engine in the metadata of the target database.



It is not mandatory to have these field numbers matching but it does reduce the risk of errors and confusion when later editing the field definitions in the Replication Manager program.

FIELD source field name – The name of the key column segment to add.

TO target field name – The name of the corresponding key field in the target record. If the target field name is not set, it is assumed to be the same as the source field name.

SCHEMA schema number or name – required if the configuration has more than one schema

TARGET target database number or name – required if the configuration has more than one target database

TABLE table number or name – required if more than one table is replicated in the schema.

## MODIFY KEY

Use to modify a key field definition. Field parameters follow the same rules as ADD KEY (above).

### Syntax

```
MODIFY KEY [FIELD] target field name or number
  [[OF] [SCHEMA] schema number or name]
  [[TARGET] target db number or name]
  [[TABLE] table number or target table name]
SET <field parameters>
```

## Working with Sequences (Generators)

### ADD SEQUENCE

Add a replicated sequence to a replication schema. By default the same sequence name is added to the current schema and the current target database.

### Syntax

```
ADD SEQUENCE [FROM] <sequence name> [<sequence parameters>]
```

### Sequence parameters

[FROM] <source sequence name> – the name of the generator in the source database that you want added to the schema. It is mandatory, although the FROM keyword is optional.

For Oracle databases, the sequence name can include the database schema in the format SCHEMA.SEQUENCE. Both parts of the sequence name are case-sensitive.

TO <target sequence name> – the name of the corresponding sequence in the target database. This parameter can be omitted if the target sequence name matches that of the source sequence.

SCHEMA <schema number or name> – Use this parameter if your command is directed at a schema other than the current one (or if you are not sure!)

TARGET <target db number or name> – Use this parameter if the target database is not the current target database (or if you are not sure!)

NUMBER <sequence number> – each sequence in a schema is assigned an integer ordinal number automatically. If you want the sequences to fit in with a customised order instead, use this parameter to pass a unique new number.



You can modify the numbering later, with MODIFY SEQUENCE.

## MODIFY SEQUENCE

Use to modify the properties of a replicated sequence. Parameters follow the same rules as for ADD SEQUENCE (above).

### Syntax

```
MODIFY SEQUENCE <sequence number>
<target sequence name> [[OF] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]]
SET <sequence parameters>
```



If the number or name properties of the SCHEMA or TARGET parameters are omitted, the current ones are used. Be certain you are directing the changes to the right target!

## REMOVE SEQUENCE

Use to remove a replicated sequence from a replication schema.

### Syntax

```
REMOVE SEQUENCE <sequence number>
<target sequence name> [[FROM] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]]
```



If the number or name properties of the SCHEMA or TARGET parameters are omitted, the current ones are used. Be certain you are directing the changes to the right target!

## SHOW SEQUENCES

List the replicated sequences for a given or current schema and target database.

## Syntax

```
SHOW SEQUENCES [[FOR] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]
```



In contrast with the behaviour of other commands, if the target database number or name is omitted, sequences for all target databases will be shown, not just the current one.

## Working with Replicated Tables

### ADD TABLE

Add a replicated table to a replication schema. By default a table is added to the current schema and the current target database.

Most of the settings are effective only during the creation of system objects. If you alter them afterwards, it will be necessary to remove and recreate the system objects.

### Syntax

```
ADD TABLE [FROM] <source table name>
TO <target table parameter>
<other table parameters>
```

### Table parameters



For Oracle databases, table or procedure names can include the database schema in the format SCHEMA.TABLE. Both parts of the table name are case-sensitive.

The FROM parameter, with the name of the source table as its argument, comes first and is obligatory. The keyword FROM is optional.

TO [TABLE|PROCEDURE|VIEW] <target table parameter> – The table, stored procedure or view that is to receive the replicated data. Use of the respective keyword TABLE, PROCEDURE or VIEW, is optional.

This parameter can be omitted if it matches the source.

NUMBER <table number> – each table in a schema is assigned an integer ordinal number automatically. If you want the table to fit in with a customised order instead, use this parameter to pass a unique new number.



Use SHOW TABLES n (where n is the schema number) to list the currently defined tables with their ordinal numbers.

SCHEMA <schema number or name> – Use this parameter if your command is directed at a schema other than the current one (or if you are not sure!)

COMMENT <table comment> – Optional text comment: use it for any notes you

want to keep about this addition. Enclose text in quotes or square brackets.

**DISABLED | ENABLED** – Signals whether this table is to be included in any batch synchronization. Normally, there is no need to set it explicitly at ADD time, as it is enabled by default. It can be modified as required, using MODIFY TABLE.

**TIME\_FIELD** <TS field name> – the name of the time field column that is used for timestamp-based conflict resolution. It must be the same in both the source and target tables, or it won't work. Only needed if this table pair uses a different time field name than the one set for the schema.

**CONDITION** <row-level replication condition> – use for applying a condition to the replication of this table pair. For more information, see the topic [Row-level Condition](#).

**TARGET** <target db number or name> – Use this parameter if the target database is not the current target database (or if you are not sure!)

**SEPARATOR** <separator character code> – Use this if you need to change the character that will be used in the replication log table to separate the segments of multi-segment keys in the text representation of records changed. Needed only if a different character has to be used for the key of this table than the one defined for the schema, for some reason.

**KEEP\_STATS** <boolean value> – Set True to enable the GUI performance monitor tool. By default, it is disabled for all tables, to reduce performance overhead by allowing replserver to work with read-only access to the tables in the configuration database. Omit this parameter unless you want to set it True immediately. It can be enabled later, using MODIFY TABLE.

**ORDER** <number> – Use this parameter if you want the table to occupy a specific position within the replication order, e.g., to precede other tables that have foreign key dependencies on it. You can use SHOW TABLES SCHEMA n to view the order of tables already defined.



You may wish to omit this parameter and defer changing the order of the tables until you have finished defining them all. A unique number will be auto-generated if you don't set it here explicitly.

**OPERATIONS** <replicated operations list> – A string value, containing the letters 'U', 'I' and 'D' for inserts, updates and deletes, respectively. By default all operations are replicated so, if this is the behaviour you want, you can ignore this parameter.

Use this parameter to disable replication of a specific operation by providing the list with only the desired operations. You can use MODIFY TABLE later to alter the set of desired operations.

**CRS** <strategy number >| DEFAULT – Use this parameter if you need to override the conflict resolution strategy that is set for the schema, for this table only. Available strategies are:

- 0 – priority-based
- 1 – timestamp-based
- 2 – master-slave

FK [BEHAVIOR] DEFAULT|ERROR|CONFLICT|IGNORE – The global default for dealing with foreign key violations is selected by default. You don't need to use this parameter unless you want a foreign key violation to be treated differently for this relation than the specified global method. Choose an alternative behavior as follows:

Error :: causes an error to be written to the log and the record will not be replicated.

Conflict :: treats the violation as a conflict, that will be resolved according to the resolution strategy applicable to the table.

Ignore :: the offending record will not be replicated.

Default :: reverts the setting to the default global behavior.

## MODIFY TABLE

Use to modify properties for replicated table. Parameters follow the same rules as for ADD TABLE (above).

### Syntax

```
MODIFY TABLE [<table number or target table name>]
[[OF] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]
SET <table parameters>
```



If the number or name properties of the SCHEMA or TARGET parameters are omitted, the current ones are used. Be certain you are directing the changes to the right target!

## REMOVE TABLE

Use to remove a table object from a replication schema.

### Syntax

```
REMOVE TABLE [<target table name or number>]
[[FROM] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>] [SILENTLY]
```



If the number or name properties of the SCHEMA or TABLE parameters are omitted, the current ones are used. Be certain you are directing the changes to the right target!

If SILENTLY is used, there will be no warnings about any failures during the removal of the table.

## SHOW TABLE

Shows information about the specified replicated table.

### Syntax

```
SHOW TABLE [<target table name or number>]
[[OF] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]
```

## SHOW TABLES

Lists the replicated tables for a given or current schema and target database.

### Syntax

```
SHOW TABLES [[FOR] [SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]
```



In contrast with other commands, if target database number or name is omitted, tables for all target databases will be shown, not just the current one.

## Working with System Objects

### CREATE SYSTEM OBJECTS

Create system objects such as log tables and triggers for a given configuration. If a schema number or name is omitted, the current schema is used. No defaults for the target database and table.

### Syntax

```
CREATE [SYSTEM] OBJECTS
[FOR] [[SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]
[[TABLE] <target table number or name>]
[SILENTLY]
```

### Parameters

**FOR SCHEMA** <schema number or name> – Use this parameter if the schema is not the current one (or if you are not sure!) The keyword FOR is optional.

**TARGET** <target db number or name> – Use this parameter if the target database is not the only target database. The keyword TARGET is optional.



TABLE <target table number or name> – optionally create system objects for just a specified table. If this parameter is omitted, system objects will be created for all tables. If TABLE is specified with no argument, system objects will not be created.

If SILENTLY is used, pending replications and conflicts are removed instead of showing an error.

## REMOVE SYSTEM OBJECTS

Use to remove a system object or all system objects for a given configuration. On completion, all pending replications and unresolved conflicts for the specified part of the schema are dropped.

### Syntax

```
REMOVE [SYSTEM] OBJECTS
[FOR] [[SCHEMA] <schema number or name>]
[[TARGET] <target db number or name>]
[[TABLE] <target table number or name>]
```

### Parameters

FOR SCHEMA <schema number or name> – Use this parameter if the schema is not the current one (or if you are not sure!) The keyword FOR is optional.

TARGET <target db number or name> – Use this parameter if the target database is not the only target database. The keyword TARGET is optional.

TABLE <target table number or name> – optionally remove system objects for just a specified table. If this parameter is omitted, all system objects for the schema will be removed. If TABLE is specified with no argument, no system objects will be removed.

## Working with the Replication Schedule

### ADD SCHEDULE

Add a schedule item.

### Syntax

```
ADD SCHEDULE [NUMBER <item number>] <schedule item> [ENABLED|DISABLED]
```

```
<schedule item> ::=
  ONCE <date> <time>
  | PERIODICALLY [EVERY] <period> [SECONDS]
  | HOURLY [AT] <minute number> [MINUTE]
  | DAILY <time>
  | WEEKDAYS <time>
  | WEEKLY <week day> <time>
  | MONTHLY <day of month> <time>
```



A new schedule is enabled by default. Include DISABLED if you just want to add the schedule item without enabling it.

### Argument Formats

- Date – US format MM/DD/YYYY
- Time – HH:MM:SS
- Week day may be in numeric (1 – Monday, 7 – Sunday) or in literal form.

## MODIFY SCHEDULE

Use to modify an item in the replication schedule. Parameters and arguments follow the same rules as ADD SCHEDULE (above).

### Syntax

```
MODIFY SCHEDULE <schedule item number> SET <schedule params>
```



Use SHOW SCHEDULE to list all the current schedule items.

## REMOVE SCHEDULE

Use to remove a replication schedule item.

### Syntax

```
REMOVE SCHEDULE <schedule id>
```

## SHOW SCHEDULE

Displays the replication schedule. No parameters.

### Syntax

```
SHOW SCHEDULE
```

## Customising Global Settings

### MODIFY GLOBAL

Use to customise the global settings.

### Syntax

```
MODIFY GLOBAL  
SET <global parameters>
```

## Global parameters

CRS <strategy number | DEFAULT> – Conflict resolution strategy to be used unless overridden at schema or database level.

If you provide the strategy number explicitly, then possible strategies are:

- 0 – priority-based
- 1 – timestamp-based
- 2 – master-slave (default)

LOG <log levels> | DEFAULT – one of two possible LOG parameters available. This one sets the logging levels and takes two integer arguments, for logging to the window (first argument) and logging to the log file (second argument).



Log levels are from 0 (no log) to 5 for ordinary builds of the Replicator server. If you are running a debug build, the highest logging level is 7.

LOG <log file name> – allows the name of the log file to be customised. By default, it will be REPLICATE.LOG and you do not have to specify that.



You can use both LOG parameters in a single command, if need be.

SCHEDULER ON|OFF – Enable or disable the replication scheduler.

MAIL ENABLED|DISABLED – Enable or disable email notification from the server to an administrator.

[MAIL] TO <e-mail address> – An email address for the administrator who wants the notifications. The keyword MAIL is optional if this is not the first MAIL parameter contained in the command.

[MAIL] HOST <SMTP server host name> – Host name of the sending email server.

[MAIL] FROM <e-mail address> – A Sender email address that has been set up for IBPReplicator's use.

[MAIL] NAME <visible name> – The Sender name that is to appear in the headers of emails sent by IBPReplicator.

[MAIL] PERIOD <minimum time between mails> – To prevent your mailbox from overflow, you can set a minimum interval (in minutes) between email notifications. If it is set, IBPReplicator will swallow mail messages for the specified number of minutes after successfully sending one.



The default setting of 0 minutes disables this protection.

BEEP <boolean value> – Set to True if you want the server to give an audible signal when it begins replicating.

FK [BEHAVIOR] ERROR|CONFLICT|IGNORE – Use to set the global default for dealing with foreign key violations. Valid settings are:

Error :: causes an error to be written to the log and the record will not be replicated.

Conflict :: treats the violation as a conflict, that will be resolved according to the resolution strategy applicable to the table.

Ignore :: the offending record will not be replicated.

## SHOW GLOBAL

Shows the global settings. It takes no parameters.

### Syntax

```
SHOW GLOBAL
```

## SIGNAL SERVER

Sends a notification to the replication server to perform the specified action. The action parameter (REPLICATION|RELOAD|EXIT) is mandatory.

### Syntax

```
SIGNAL [SERVER] [TO] {REPLICATE|RELOAD|EXIT}
```

The keywords SERVER and TO are optional.

## 10.3 Appendix III: Version Notes

### Replication Server 5.0

#### IMPORTANT!

The configuration database structure in version 4 and higher releases of this software is not compatible with those of previous releases.

Please take copies of your configuration databases before installing.

The newer Replication Manager will automatically upgrade old configuration databases on opening. If the configuration editor cfged is used, executing the command UPGRADE will achieve the same thing.

### Changes to Note When Upgrading

Existing IBPhoenix Replicator V.3 and V.4 licences are valid for V.5 and will be preserved. Licences from versions prior to V.3 are not valid for Replicator 5 and will be removed from the configuration altogether.

If you are upgrading from 2.1 or lower versions, you will find that the Replication scheduler is now built into the Replication Server and no longer runs as a separate service. Configuring and applying a schedule are now done using the Scheduler tool or the configuration editor. Upgrading from a pre-2.5 configuration will convert a "regular interval" Replication into a Schedule.

If you are currently using Replicator 2.1 or older, you must uninstall the old Replicator Scheduler service before upgrading. It was superseded at v.2.5 and is no longer a component of the product.

### New in Version 5.0

Main changes in this version:

Embedded Replication under Avalerion server

Replicator 5.0 can be "plugged in" to the IBPhoenix "Avalerion" builds of Firebird 3.0 to run internally, i.e., the replicator engine runs as plug-in engine code, not as a client application.

Two-phase commit in Firebird and InterBase are no longer supported

# Index

## - . -

.conflict files, 108, 134  
 .synch files, 136

## - A -

Access existing configurations, 100  
 ADD DATA (cfged), 161  
 ADD DATABASE (cfged), 153  
 ADD KEY (cfged), 162  
 ADD LICENCE (cfged), 152  
 ADD SCHEDULE (cfged), 169  
 ADD SCHEMA (cfged), 156  
 ADD SEQUENCE (cfged), 163  
 ADD TABLE (cfged), 165  
 ADD TARGET (cfged), 158  
 Administrative Role, 32, 39  
 Architecture of IBReplicator, 12  
 Asynchronous replication, 13  
 Autogenerate mappings, 66

## - B -

Benefits of IBReplicator, 9  
 Bi-directional, 125  
 Bulk mapping of tables, 66

## - C -

Cascade, 19  
 Central-to-branch, 19  
 Central-to-standby, 19  
 cfged configuration editor, 146  
 cfged: Commands, 148  
 cfged: Common Rules, 147  
 cfged: Customising Global Settings, 170  
 cfged: Database Registration, 153  
 cfged: On-Line Help, 148  
 cfged: Program Settings, 149  
 cfged: Starting, 146

cfged: Where Am I?, 149  
 cfged: Working with Configurations, 150  
 cfged: Working with Data Fields and Keys, 161  
 cfged: Working with Replicated Tables, 165  
 cfged: Working with Schemas, 156  
 cfged: Working with Sequences (Generators), 163  
 cfged: Working with System Objects, 168  
 cfged: Working with Target Databases, 158  
 cfged: Working with the Schedule, 169  
 Choosing, replicated tables, 61  
 Choosing, source database, 51  
 Choosing, target database, 57  
 CLEAR (cfged), 152  
 Command-line configuration editor, 146  
 Command-line operation, Linux, 93  
 Command-line operation, Windows, 91  
 Command-line switches (cfged), 148  
 Command-line switches (Linux), 96  
 Command-line switches, Linux, 93  
 Command-line switches, Windows, 91  
 Configuration database, 29  
 Configuration database, default, 46  
 Configuration editor, 146  
 Configuration, default, 46  
 Configure Foreign Key violation behaviour, 82  
 Configuring Linux, 94  
 Conflict resolution, 26  
 Conflict resolution after offline replication, 135  
 Conflict resolution, master-slave, 26  
 Conflict resolution, priority-based, 26  
 Conflict resolution, setting default strategy, 82  
 Conflict resolution, timestamping, 26  
 Conflict resolution, timestamps for, 23  
 Conflict viewer tool, 108  
 Connection, for source database, 51  
 Connection, for target database, 57  
 Context menus, 47  
 Controlling replication, 87  
 CREATE CONFIGURATION (cfged), 150  
 Create schema, 47  
 CREATE SYSTEM OBJECTS (cfged), 168  
 Customising table order, 76

## - D -

Database features, support for, 9

Database priority, 35  
 Database,choosing a source, 51  
 Database,choosing a target, 57  
 Database,design issues, 23  
 Databases,registering, 13, 32  
 Default (global) settings, 82  
 Default configuration database, 46  
 Definitions, ways to work on, 48  
 Definitions,ways to work on, 47  
 Distributed primary keys, 23  
 DROP (cfged), 151

## - E -

Email notification, 82, 84  
 Enable/Disable Monitor, 74  
 Event logging, 83  
 event-triggered replication, 29  
 Exclude an operation, 75  
 Executables,Windows, 87  
 Existing configuration, to access, 100  
 EXTRACT (cfged), 151

## - F -

Foreign Key Violation behaviour, 82, 83  
 Foreign key violation behaviour, table level, 75

## - G -

Generators, 71  
 Getting email notification, 82  
 Global settings, 82  
 Global settings,customising, 82

## - H -

Handling conflicts, 26  
 HELP (cfged), 148  
 Heterogeneous replication, 25  
 Hub-and-spoke, 19

## - I -

IBReplicator,architecture, 12  
 IBReplicator,benefits, 9

IBReplicator,security, 9  
 IBReplicator,terminology, 13  
 Inbound offline replication, 134  
 INIT (cfged), 151  
 Installation, 28  
 Installation ODBC, 29  
 Installation,Start Menu shortcuts, 28  
 Installation,Windows installer, 28  
 Installer switches, 91  
 Installing licences, 110  
 Installing on Linux, 94  
 Intervals,replication, 82  
 IPX/SPX protocol, 29

## - K -

Keep Statistics, 74  
 Key mappings, 70  
 kill parameters, 93  
 killall parameters, 93

## - L -

Licences, loading manually, 99  
 Licences, loading without Windows, 99  
 Licences,installing, 110  
 License Manager tool, 110  
 Licenses,about, 42  
 Licensing,about, 42  
 Linux operation, 93  
 Linux, controlling replserver processes, 95  
 Linux, installing and configuring, 94  
 Local protocol, 29  
 Log file override, 59  
 Logging, 82

## - M -

Managing replication, 87  
 Manual conflict resolution after offline replication, 134  
 Mapping tables, bulk, 61  
 Mapping tables,heterogeneous, 61  
 Mapping tables,removing mappings, 61  
 Mapping,with stored procedures, 123  
 Metadata changes, 121

MODIFY DATA FIELD (cfged), 161  
 MODIFY DATABASE (cfged), 155  
 MODIFY GLOBAL (cfged), 170  
 MODIFY KEY (cfged), 163  
 MODIFY SCHEDULE (cfged), 170  
 MODIFY SCHEMA (cfged), 157  
 MODIFY SEQUENCE (cfged), 164  
 MODIFY TABLE (cfged), 167  
 MODIFY TARGET (cfged), 159  
 Monitor tool, 102  
 Monitoring, 74  
 MySQL ODBC issues, 29

## - N -

Named Pipes protocol, 29  
 NetBEUI protocol, 29  
 Network protocol,IPX/SPX, 29  
 Network protocol,local, 29  
 Network protocol,Named Pipes, 29  
 Network protocol,NetBEUI, 29  
 Network protocol,TCP/IP, 29  
 Notify Server utility, 107  
 N-way replication, 25

## - O -

ODBC, 28  
 ODBC and sequences, 71  
 ODBC as Source Database, 56  
 ODBC Datasource as a Source Database, 51  
 ODBC known issues, 29  
 ODBC requirements, 29  
 Offline replication, 129  
 Offline replication from inbound files, 134  
 Offline replication using .synch file sets, 135  
 Offline replication, running, 133  
 Offline replication, schema, 131  
 Off-line synchronization, 136  
 One-way replication, 25  
 OPEN (cfged), 146  
 Operations to replicate, 75  
 Operations,Linux, 93  
 Operations,Windows, 87  
 Ordering of replications, 50  
 Override log file per target, 59

## - P -

Password,for source database, 51  
 Password,for target database, 57  
 Peer-to-peer, 19, 125  
 Planning for replication, 19  
 PostgreSQL ODBC issues, 29  
 Primary keys, 23, 70  
 Primary keys,distributed, 23  
 Priority, 35

## - R -

RAS parameters, 39  
 Registered database, 51, 57  
 Registering a database, 12  
 Registering databases, 13, 32  
 Remote Access (RAS), 39  
 Remote access parameters, 32  
 Remote configuration, to access, 100  
 REMOVE DATA FIELD (cfged), 162  
 REMOVE DATABASE (cfged), 155  
 REMOVE LICENSE (cfged), 152  
 REMOVE SCHEDULE (cfged), 170  
 REMOVE SCHEMA (cfged), 158  
 REMOVE SEQUENCE (cfged), 164  
 REMOVE SYSTEM OBJECTS (cfged), 169  
 REMOVE TABLE (cfged), 167  
 REMOVE TARGET (cfged), 160  
 REPL user, 125  
 Replicant, 12  
 Replicant, definition of, 13  
 Replicated generators, 71  
 Replicated sequences, 71  
 Replication, 12  
 Replication Monitor tool, 102  
 Replication order, 50  
 Replication Scheduler tool, 104  
 Replication schema,defining, 16  
 Replication scheme, 19  
 Replication scheme,Cascade, 19  
 Replication scheme,Central-to-branch, 19  
 Replication scheme,Central-to-standby, 19  
 Replication scheme,Hub-and-spoke, 19  
 Replication scheme,Peer-to-peer, 19



- Replication tree, 47
- Replication,ad hoc, 104
- Replication,asynchronous, 13
- Replication,bi-directional, 125
- Replication,controlling, 87
- Replication,executing, 16
- Replication,heterogeneous, 25
- Replication,intervals, 82
- Replication,managed, 104
- Replication,managing, 87
- Replication,n-way, 25
- Replication,one-way, 25
- Replication,planning for, 19
- Replication,sequence of events, 16
- Replication,synchronous, 13
- Replication,timestamps for, 23
- Replication,tree, 47
- Replicator, 12
- Replicator,definition of, 13
- replserver switches (Linux), 96
- ReplServer.exe, 87
- Requirements, 6
- Requirements,database engine, 6
- Requirements,disk space, 6
- Requirements,licensing, 6
- Requirements,minimum database installation, 6
- Requirements,operating system, 6
- Role, 32
- Role, Administrative, 39
- Role,for source database, 51
- Role,for target database, 57
- Row-level condition, 73
- Row-level conditions, 71
- Row-level replication condition, 73
- Row-level replication condition (global), 59

## - S -

- Scheduler tool, 104
- Schema View tool, 113
- Schema, creating, 47
- Schema, numbering, 118
- Schema,complex, 125
- Schema,defining, 16
- Security, 9
- Separator character, 55, 75

- Separator character (global), 55
- Separator character (table-level), 75
- Sequence of events, 16
- Sequences, 71
- Sequences and ODBC, 71
- SET AS DEFAULT (cfged), 152
- SET BAIL (cfged), 149
- SET CURRENT (cfged), 149
- SET ECHO (cfged), 150
- SET PROMPT (cfged), 150
- SHOW CURRENT (cfged), 149
- SHOW DATABASE (cfged), 155
- SHOW DATABASES (cfged), 156
- SHOW FIELDS (cfged), 162
- SHOW GLOBAL (cfged), 172
- SHOW LICENSES (cfged), 153
- SHOW SCHEDULE (cfged), 170
- SHOW SCHEMA (cfged), 158
- SHOW SCHEMAS (cfged), 158
- SHOW SEQUENCES (cfged), 164
- SHOW TABLE (cfged), 168
- SHOW TABLES (cfged), 168
- SHOW TARGET (cfged), 160
- SHOW TARGETS (cfged), 160
- SIGNAL SERVER (cfged), 172
- Source database, 12
- Source database,choosing, 51
- Source database,event logging, 51
- Source database,settings, 51
- Stored procedures, 123
- Synchronization, 12
- Synchronization of tables, 76
- Synchronization order, 76
- Synchronization, off-line, 136
- Synchronous replication, 13
- System Objects, 16
- System Objects,creating, 80
- System Objects,removing, 80

## - T -

- Tables,mapping source and target, 61
- Tables,synchronization, 76
- Target database, 12
- Target database,choosing, 57
- Target database,General tab, 57

Target database,periodic commit, 57  
Target database,settings, 57  
TCP/IP protocol, 29  
Timefields,for timestamping, 121  
Timefields,generating a script for, 121  
Timestamps for conflict resolution, 23  
Tools, 100  
Tools,Conflict viewer, 108  
Tools,License Manager, 110  
Tools,Notify server utility, 107  
Tools,Replication Monitor, 102  
Tools,Replication Scheduler, 104  
Tools,Schema Viewer, 113  
Tree nodes, 47

## - U -

Unique keys, 23  
UPGRADE (cfged), 152  
Username,for source database, 51  
Username,for target database, 57  
Using .synch files for Replication, 135  
Utilities, 100

## - V -

View Conflicts tool, 108

## - W -

Working on definitions, 48

IBPhoenix Publications



© 2016 IBPhoenix Editors