Connection Strings in Firebird 3

by Helen Borrie & Vlad Khorsun

The format of the connection string for connecting to a database plays an essential role in determining the protocol of the connection request. Firebird 3 introduced a new way to express these strings—URL-style.

Connection Protocols

Firebird supports two connection protocols on POSIX and four on Windows:

- TCP/IP is the protocol of choice for connecting clients to servers on all platforms, including server-local connections using local loopback.
- Hostless connection, available on all platforms, which uses an embedded engine if the engine provider is available and the database is not already opened exclusively by a stand-alone server or another embedded engine process. Authentication is not required, so no password check is done. The user name, supplied in the connection request or acquired from the ISC_USER environment variable (if available) or from the operating system, is used at database level to check privileges and mappings.
  Where embedded is not available, a hostless connection could be established using the Loopback provider, in which case full login credentials would need to be present. On Windows, the provider will try to use XNET (see below) to make the local connection and, failing that, a localhost connection via TCP/IP or, on Windows, via WNET (see below).
- The Named Pipes protocol on Windows (a.k.a. WNET, Windows Networking or NetBEUI) is deprecated. It is still supported in Firebird 3 for connecting to databases at physical drive addresses, although it is considered too "noisy" for database traffic. Another limitation is that Windows restricts piped connections to 255. Support for WNET is likely to be removed in later Firebird versions.
- The XNET protocol, sometimes referred to as "Windows Local" is a full network protocol, most familiar to Delphi application developers, that runs in the application space of the local user, currently only on Windows, making attachments to a running server. It uses a "hostless" database file path but it is not a variant of embedded. Like TCP/IP and WNET, it requires connections to be authenticated by either trusted or native authentication, according to the configuration of AuthServer and (at the client side) AuthClient.

Providers

The unified Firebird 3 engine comes with three providers that plug in automatically according to one or two factors. The installation default configuration, in firebird.conf, makes all three available:

```
#Providers = Remote,Engine12,Loopback
```

Remote and Loopback are the providers that are built in as part of the client library (fbclient.dll on Windows, libfbclient.so on Linux) allowing clients to connect to a remote server. Engine12 is a separate stand-alone provider that contains the implementation of the Firebird database engine. Engine12 is built as a dynamic library (shared object) that is located in the plugins sub-folder of the standard installation (engine12.dll on Windows, libEngine12.so on Linux).

The Remote and Loopback providers share the implementation of the INET, WNET and XNET protocols. They differ in that the Remote provider enables connection to a Firebird server running at any host on the network (including localhost), while the Loopback provider is limited to connections only to a server running on the local host.

On Windows. the Loopback provider first tries to connect using the XNET protocol. If that fails, its next attempt is with WNET, using the host name "\\." If that fails, Loopback tries INET using "localhost" as the host name. Of course, on the other platforms, INET is the only protocol attempted.

Connection Strings

Almost all of the traditional remote and local connection string formats are still applicable. However, Firebird 3 introduced a number of URL-style formats that can be used alternatively.

Formats: Traditional and URL-Style

The URL-style connection string format has the "look" of a URL The pattern is consistent across all protocols:

```
 [<protocol>://[<hostname>[:<port>]/]]<database file path or alias>
```

The `<protocol>` element can be INET, INET4, INET6, WNET or XNET. It is case-insensitive. Watch out in the detailed notes on Page 5 for a slight exception to the pattern for the `<hostname>` element with WNET.

For databases hosted in Windows, the database file path can contain directory separators that are interchangeably blackslashes or forward slashes. Backslashes are not valid for databases hosted on a POSIX filesystem.

TCP/IP

The traditional style for connecting via TCP/IP is

```
<hostname> | <IP address>[/<port>]:<database file path or alias>
```

Remote Client

For example, to connect from a remote client to the aliased employee.fdb database on a Linux server that is listening on port 13050:

```
linuxhost/13050:employee
```

Connecting from a remote client to a database on a Windows server:

```
winserver:c:\databases\mydatabase.fdb
```

Local Client

To connect to a local database on a Windows server using local loopback with the most common localhost IP address:

```
127.0.0.1:d:\databases\mydatabase.fdb
```

On Linux, using the host name localhost instead of the IP address:

```
localhost:/opt/databases/mydatabase.fdb
```

The equivalents, using the URL-style syntax:

Remote Client

```
INET://linuxhost:13050/employee
INET://winserver:d:/databases/mydatabase.fdb
```

Local Client

Connecting to the Windows server:

```
INET://d:\databases\mydatabase.fdb
```

Connecting to the Linux server:

```
INET:///opt/databases/mydatabase.fdb
```

Notice the absence of the host name or IP address in these local strings. The Dispatcher just passes the database path to each configured provider until the connection succeeds. If the Remote provider finds the INET protocol prefix with no host name or address, it attempts the same connection procedure that the Loopback provider would do.

IPv6

INET will use IPv6 if it is configured for the connection. Use INET6 as the protocol identifier if you want to enforce the IPv6 connection.

Note also that, if the configuration parameter `IPv6V6Only = 1` (i.e., true) in firebird.conf, then the server will not accept IPv4 connections at all. By default, this parameter is false.

IPv4

Use the protocol modifier INET4 instead of INET if you want to enforce the IPv4 connection.

WNET (Windows Networking)

The Windows networking (WNET) protocol processes Named Pipes connections from Windows clients to a Windows server. Although the path format looks similar to that used on Windows for accessing shares and mapped locations, the path to a Firebird database, including that stored for an alias, must be the true path to the file's location on a drive that is physically attached to the host server.

Note, many Delphi database components refer to WNET as NetBEUI, although NetBEUI is not, strictly speaking, a network protocol.

The format of the traditional WNET connection string is:

```
<hostname>[@<service_name>]\\<database file path or alias>
```

where <service_name> is the value of the parameter RemoteServiceName in firebird.conf. By default, it is gds_db and you do not need to append it to the hostname.

Pipe Name

The name of the pipe that Windows is to use for the WNET connection is constructed by the Remote provider from the following elements, all of which are case-insensitive:

```
<remote host name>\<pipe prefix>\<remote pipe name>\<remote service name>
```

where

`<remote host name>` is the host name passed in the connection string, without the double backslash

`<pipe prefix>` is a fixed value ***pipe***

`<remote pipe name>` is set in the RemotePipeName configuration setting, default ***interbas.*** It is not possible to set RemotePipeName in the connection string.

`<remote service name>` is set in the RemoteServiceName configuration setting. The default ***gds_db*** can be overridden in the connection string

Examples of traditional WNET strings

Connect to the remote server on host "winserver" running with the default setting for RemoteServiceName and RemotePipeName:

```
\\winserver\c:\databases\mydatabase.fdb
```

Connect to the server on the local host running with the default RemoteServiceName and RemotePipeName settings:

```
\\.\c:\databases\mydatabase.fdb
```
or
```
\\localhost\c:\databases\mydatabase.fdb
```

Connect to the remote server host "winserver" running with RemoteServiceName = "MyAppInstance" and the default RemotePipeName:

```
\\winserver@MyAppInstance\c:\databases\mydatabase.fdb
```

Connect to the local server running with RemoteServiceName = "MyAppInstance" and the default RemotePipeName:

```
\\.@MyAppInstance\c:\databases\mydatabase.fdb
```

Using the URL syntax for WNET

The URL-style syntax for a WNET connection is (formally):

```
WNET://[<hostname>[:<port>]/]<database file path or alias>
```

However, owing to a bug in the implementation that was still present at Firebird 3.0.3, the `<hostname>` element requires to be prefaced by two backslashes. These particular backslashes are not interchangeable with forward slashes.

E.g.
```
WNET://\\winserver/d:/databases/mydatabase.fdb
```

If using a non-default service name:

```
WNET://\\winserver:myservice/d:/databases/mydatabase.fdb
```

XNET ("Windows Local")

XNET surfaces the old "Windows local" (shared memory) protocol, whereby a "hostless" connection is made to a server process on the same host as the user application runs on. It is not the same as running an application with an embedded engine. Indeed, it is a regular client-server connection and, as such, is subject to authentication, either trusted or native.

The traditional path format for an XNET connection to Superserver was simply the hostless path:

```
d:\databases\mydatabase.fdb
```

The same connection using the URL-style syntax:

```
XNET://d:/databases/mydatabase.fdb
```

Using an alias:

```
XNET://MyDB
```

Why XNET?

Windows Local connection, nowadays implemented in Firebird as XNET, is a legacy from the early days of Delphi in the 1990's, when the higher-end editions of Delphi shipped with a development-only edition of InterBase known as "Local InterBase". InterBase had no embedded engine. Local InterBase could not be networked but it provided a means for developers to test their Delphi/InterBase applications without purchasing the full InterBase software. All Delphi database components in those days provided an API wrapper for the Windows local connection protocol, so that it was not uncommon for Delphi developers to deploy stand-alone desktop applications that used only Windows local and (illegally) Local InterBase. However, many developers took advantage of Windows Local to connect selected users locally to databases that were running under a fully-licensed Superserver.

Firebird's XNET protocol was introduced in Firebird 2.0 to replace the really old, inferior legacy InterBase implementation of Windows local protocol. It is a full-strength client-server access protocol and is not deprecated. It is an important part of Firebird that can be used for access to the local Firebird server without any limitations. It usually performs better than INET or WNET, too.

Protocol 'Local', still an option in the latest versions of some Delphi data access components and, perhaps, in drivers for some other host languages on Windows, means XNET when accessing databases on Firebird servers from v.2.0 forward. It is often used for implementing some administrative applications on servers that run concurrently in client/server or intranet networks.

Why not use Embedded instead?

In Firebird 3, an embedded engine can happily connect to a database that is already running in Classic or Superclassic mode. If the application suite needs to be deployed on Windows with the server in Super mode, XNET provides a way past the difficulties associated with trying to connect Superserver clients and embedded applications to the same database simultaneously.

There might be a security angle, too. Unlike Embedded, XNET connections must be authenticated. The developer might want to make use of that to control access to administrative functions.

It should also be noted that, if an embedded application should crash, it could corrupt the database. If an application connected with XNET crashes, it cannot corrupt the database because the server engine and the application are separate processes.

Conclusion

In summary, Firebird 3 introduces new ways to express protocol and path syntax for client applications connecting to Firebird databases.