SQL DM for MySQL Solutions

How do connection names resolve to IP's?

It does not answer questions about the SQL DM program as such. But we experience quite often that SQL DM user having problems in getting the connection to one ore more MySQL servers working are missing the basic understanding of the TCP protocol that is used for connection. So this is a 'background article' that supplements the more typical FAQ items here.
The TCP protocol needs an IP to connect to a host. An IP is a group of 4 2-digit hexadecimal numbers. Like '0A.1B.2D.3D'. It is common to express this in decimal format 'v.x.y.z' where v,x,y and z are numbers between 0 and 255. To get your own IP in this format just type (on windows) 'ipconfig' on a command-line and the IP is returned.
The IP returned can be a global ip (accessible directly from the internet) or - if you are behind a router - a local ip that only applies behind the router. All computers behind the router share the global ip of the router itself. The router has the functionality to ensure that communication from each machine behind it to and from the internet works.
The TCP-protocol communicates on 'ports'. There are about 65.000 ports possible. A TCP-port is nothing physical but just a number that each 'packet' of information that is sent is coded with. Various ports are reserved for various server programs and programs with sort-of server functionality. Here, you can review the complete list of Service Name and Transport Protocol Port Number Registry:
So to access a MySQL server (running on the standard MySQL port 3306) on the ip v.x.y.z the client must ask for connection to the server 'v.x.y.z:3306'. The Internet can handle this. This is the basically how the Internet and the TCP-protocol works! Pretty simple actually! And there is one basic rule more: the ip 127.0.0.1 is always any computer itself!
However you probably never enter internet addresses that way no matter if it is a website, an FTP-server or a MySQL-server that you access. You use NAMES. However these names must be resolved to the above format. If not resolved to ip's the servers on the Internet can't handle the request.
There are basically four ways of resolving a name to an IP(and in that order):

1. Using client hosts' file
2. Using (local) nameserver lookup
3. Using Domain Name Server (DNS) lookup.
4. Using Remote Network routing (using either simple routing based on ports, hosts file or nameserver) on remote network.

1. Client hosts' file
There is an excellent article on the hosts file What is hosts?. Try to find your own hosts file. You will find this line in it:

> ***127.0.0.1 localhost***

You might find more lines too - for several reasons: some viruses and spyware add items to the hosts file. Some users and Sys Admins do it on purpose. But since you are able to connect to a MySQL server running on your local machine with the host name **'localhost'** it is because of the above line in the hosts file. The hosts files resolves the name 'localhost' to ip 127.0.0.1. If you prefer to use **'cutie'** instead of **'localhost'** then just add the line

> ***127.0.0.1 cutie***

to your hosts file!
2. A local nameserver.
A local nameserver can be installed explicitly or implicitly. If you have more windows computers connected on a windows' network (using the TCP protocol - not old NETBEUI protocol etc) you already have a local nameserver that was implicitly installed. If you have two machines on your network named 'Bonnie' and 'Clyde', then you can connect to the MySQL server running on 'Bonnie' from the computer 'Clyde' with SQL DM entering 'Bonnie' as the host name. Because the nameserver that was implicitly installed as a part of the Windows network resolves the name 'Bonnie' to 'Bonnie's ip on the network
3. A Domain Name Server
Domain name Servers are part of the Internet structure. Any name/string in the format domain.topleveldomain (ie: school.edu, business.com, whisky.org etc.) can be looked up on the Domain Name Servers of the Internet. The ip returned is then used for creating the connection.

A typical example

You connect to a remote server with the host name *mydb.thisismyisp.com*. What happens here? Simply:

- **First:** local hosts file is checked if there are any matching entry. if not proceed to step 2.
- **Second:** local nameserver (if exists) is checked if there are any matching entry. if not proceed to step 3.
- **Third:** Domain Name Server lookup identifies the ip of domain-host *thisismyisp.com*
- **Fourth:** local network routing systems (router, nameserver or hosts file) on the remote network identifies the local IP mapped to name 'mydb´'

After following the previous steps *mydb.thisismyisp.com* is resolved to ip v.x.y.z on the internet and IP a.b.c.d on the remote network. Once connection is established the routers and other networking gear in the complete communications chain keeps that information until it is closed or times-out due to inactivity (or some error occur).

One more detail, when connecting to the MySQL server at an ISP it is common practice that MySQL user names and MySQL database names must be prefixed with the domain user name (the one that is used for FTP for instance). Like me_username and me_mydb. Simple because other users may have created a MySQL user **username** and a MySQL database **mydb**. But failure to do so does not result in a connection error, but an authentication error. Also note that MySQL usernames can be up to 16 characters long only, review the error "I have a very long username for the MySQL database at my ISP. SQL DM won't let me use it" that is listed below.

Nevertheless, everything can get 'messed up' if it is the first time you try to connect to a remote MySQL server! And here tunneling is not involved.

What should I enter as 'hostname' when connecting to a MySQL server at an ISP.

It depends on the systems and the network settings at your ISP. The systems that ISP's operate are too numerous to mention. Some operate almost all their server programs on one very powerful (UNIX-) machine, others operate a network of many computers that are basically PC-type hardware. Some use Linux/Unix operating systems other Windows server systems (and some mix it!). And ISP's vary in size from 200 customers to 2.000.000. They can't all operate the same systems!
But the general answer is that probably the ISP operates some name server internally. Such name server will typically be configured with the some name like 'mysqlserver' pointing to the ip of the mysqlserver on the local network.

- **Direct connection:**

  If the ISP has the domain name *thisisanisp.com* then with direct connection you will use *mysqlserver.thisisanisp.com*.
  The information about the name assigned to the mysql server and thus the exact URL to the MySQL server usually is available as part of your account info from some web-based control panel application. Most likely it is something like 'mydb1.thisisanisp.com' or 'mysql.thisisanisp.com'. If you cannot find the information yourself you will have to ask the support/help desk there.
  It also could happen that you could simply use *thisisanisp.com* or the global IP of the ISP, and traffic on port 3306 is routed to the MySQL server - but it will not always work - particularly not if it is a big ISP!. But everything here depends on the local routing systems used at the remote host - whether port-number based routing is active or not.

- **HTTP-tunneling:**

  With HTTP-tunneling 'host' can simply be 'mysqlserver' (ie: 'mydb1' or 'mysql' or whatever). *mysqlserver.thisisanisp.com* will work too, but there is no need for an extra domain lookup! If it is the same computer that runs the Web-server (with PHP-interpreter) and the MySQL server it can also be 'localhost' or the ip '127.0.0.1'. PHP will connect to MySQL through a Unix-Socket if 'localhost' is specified and TCP if '127.0.0.1'. Both will normally work. TCP may be a little bit slower on Linux machines than Socket, so you can try 'localhost' first. However there are situations where 'localhost' will return an error like error: **'Error No. 2013: Lost connection to MySQL server during query'**. This is likely because PHP looks for the socket file where it does not exist. The underlying reason could be that MySQL and PHP have been installed from different repositories using different file positions for the socket file and the mysql datadir.

- **SSH-tunneling.**

  With SSH-tunneling there are two hosts' settings:

1. The 'MySQL Host Address' on the Server Tab on the connections manager: Here the 'host' is the 'path' to the MySQL-server relative to the SSH-server on the remote local network. Most often 'localhost' will do. If it does not work and 'mysqlserver' does not either you will have to ask the support/Sys Admin for more information. There are lots of possible ways to configure SSH on a network! Also here if the MySQL host as entered here cannot be reached from the SSH-server the MySQL client error: **'Error No. 2013: Lost connection to MySQL server during query'** occurs. Thus this can be caused by an erroneous 'host' entry, but it can also be a network/configuration error on the remote network. Or the MySQL server simply could be down.
2. The 'SSH host' on the Tunnel Tab on the connections manager: This is the 'global URL' of the SHH server at the ISP. The ip will normally work fine, *thisisanisp.com* too. And maybe SSH is available via the internal name server too - then use something like *sshserver.thisisanisp.com.*

SQL DM gives you the most options for connecting to MySQL

SQL DM connects to MySQL using the native C-API from MySQL - the fastest and most effective way to manage MySQL. This API is compiled into SQL DM code itself.

Even if the TCP-port (3306) normally used by the MySQL server is blocked (as often is the case at ISP's) SQL DM still let you connect using HTTP(s)-tunneling or SSH-tunneling. And if you are behind a proxy SQL DM can handle that too. The client for SSH connection is installed with SQL DM, and a PHP-script for HTTP-tunneling is too. That script must be uploaded to your webhost. The PHP-script uses php_mysql extension that is available practically everywhere where MySQL is available.

SQL DM works with MySQL version 3.23 and upwards.

SQL DM takes long time to connect when using SSH-tunnel.

If SQL DM is taking long time (more than 5-10 seconds) to connect to your MySQL server using SSH tunneling then there could be problem with the name resolution.
The first thing that the SSH server does before forwarding the connection to the specified MySQL server is to perform a reverse DNS lookup on the client's IP. The SSH server may cause an unnecessary delay during authentication due to incorrect or absent of reverse DNS settings. So in case of any slowness you should check those settings.
You can also disable most of the server-side lookups by setting UseDNS = "no" in SSHD configuration file (/etc/ssh/sshd_config on most systems). But in that case the MySQL host must be specified with an ip and not a hostname.

What Is SSH and SSH-tunneling?

The Acronym **SSH** stands for **Secure Shell Host**. SSH was originally created to provide a secure way to access server systems at "low level", to be used instead of common (but insecure) **telnet** methods. SSH can use several different forms of encryption, anywhere from 56 to 1024 bit. SSH has been ported to Operating Systems on several platforms including Linux, Microsoft Windows and Macintosh. There are SSH servers and SSH clients available for different types of communication.

Here you may notice this: "OpenSSH includes the ability to forward remote TCP ports over a secure tunnel, allowing that way arbitrary TCP ports on the server side and on the client side to be connected through an SSH tunnel". This is excatly what we make use of.

The term **"SSH tunneling"** in relation to a database server means that in- and outgoing communications to the network that hosts the database server "passes through" the SSH-server and uses the communications port (usually port 22) and the protocol of the SSH-server. The SSH-server then "translates" and "transfers" that in-and outgoing communication to the database server. SSH is actually quite simple to use. However if you are totally unfamiliar with networking terminology you will have to study it somewhat. Actually you may even install an SSH server at your own local machine and use it for connecting to a local MySQL server. Not much use of that, but it will give you an excellent understanding of what SSH is!

There is a built-in SSH-client in SQL DM that lets you connect to a MySQL server using SSH.

Basically there are two benefits of SSH Tunneling:
* SSH can be used to encrypt communications between SQL DM and your remote MySQL server.
* It lets you access the MySQL server even if the MySQL port (3306) is blocked.

Unlike HTTP-tunneling, you cannot take it for granted that SSH-tunneling is available at your webhost. In general the "more professional" and "more expensive" hosting providers offer SSH and the cheaper ones don't.

Refer to the SQL DM help file for instructions how to set up the SQL DM Connections Manager, if you want to use SSH-tunneling with SQL DM. Each SSH-connection occupies a TCP-port at your local machine. With recent SQL DM versions this port is picked automatically from the pool of high-numbered ports not already in use.

A concluding note on the popular 'Putty' program and SQL DM SSH-tunneling.

Sometimes when people are having problems with SSH-connections, we often hear "I can connect with Putty without problems". Maybe so, but it does not tell very much (almost nothing actually!) because the type of connection with Putty or a similar program referred to here by users **is not tunneling** and does not make use of port forwarding. Putty creates a remote (and secure) shell on the client machine, and connects to the 'mysql' client program on the server. So here the MySQL client is the 'mysql' client on the remote server. It is true that Putty can be used for setting up a SSH-tunnel as well, but this is not the simple 'connect with Putty' most often referred to and compared with here. With (SQL DM) SSH-tunnel the MySQL client is the client API that is compiled into SQL DM (and SJA). That is why port forwarding is needed and must be functional with SQL DM SSH-tunneling!

You can review the SSH-related error messages.

Connection Issues
I have a very long username for the MySQL database at my ISP. SQL DM won't let me use it.

Right! Because a MySQL user name is up to 16 characters long. The MySQL docs clearly state:

> ***MySQL usernames can be up to 16 characters long. Operating system usernames might have a different maximum length. For example, Unix usernames typically are limited to eight characters.***

However, it is a bad practice with some ISP's that they generate longer usernames than that. It is typically 'cheaper' hosting providers that offer a single MySQL database as part of a 'personal' or 'small business' subscription plan. They auto-generate the username from the user's domain name and it could be somewhat like **mydb_myveryowndomain** or similar. It is also true that some of our competitors offer support for that. However it is very bad practice! To make it work you will have to:

1. ALTER the TABLE mysql.users and change the mysql.users.users column from a char(16) to some longer value
2. You can no longer GRANT user rights, but must INSERT/UPDATE the mysql system tables directly
3. You must 'patch' (or rather 'hack') the MySQL API/client code

We have had intense discussions with this the MySQL AB on this issue. From the official correspondence we quote:

> ***"This is simply a lucky fluke of sorts (****if it works****). MySQL simply does not support longer usernames .... Altering the system tables, aside from using our own mysql_fix_privilege_tables script to keep up with our changes, is simply unsupportable. There are server and client changes needed to properly handle any sort of modifications here, even though in some cases a quirk (as above) may seem to function .... This is, basically, dangerous behavior. We will attempt to curb it as well as we can.... Luckily, our manual states clearly that in both cases, MySQL will not provide support if any problems arise ... That is , it may work and it may not work, but MySQL will not ponder as to why it works or why it does not work ... We simply do not provide support for such cases."***

We won't play that game as others do! You should convince your ISP that changing the format of the user table is bad and dangerous practice!

And further: MySQL has 'stopped the game'. Again we quote from the above correspondence:

> ***"To make things even more precise, I will now send a server patch to our development management. This patch adds a code that will truncate user column at 16 chars and other columns to their nominal sizes. This will ensure that future 4.1 and 5.0 versions will not work with longer names, whatever changes some application could envisage."***

So with the most recent builds in the MySQL 4.1.x series and with MySQL 5.0 it would not work anyway. There is now code in the server binary itself that truncates any user name to 16 characters.

I get Error Nº 2002. Can't connect to local MySQL server through socket ...

This can occur when connecting using HTTP-tunneling to a MySQL server running on Unix/Linux platforms.

MySQL writes Problems with MySQL sock about this issue.

How to cope with this would depend on which webserver and which php version is used. But here is a workaround that has worked with Apache:
If the directory /var/lib/mysql doesn't exist than create it and chown to user mysql:
"mkdir /var/lib/mysql; chown mysql /var/lib/mysql"
Then edit the /etc/my.cnf file and specify:
[mysqld]
socket=/var/lib/mysql/mysql.socket
[client]
socket=/var/lib/mysql/mysql.socket
And, restart the mysql server (**"etc/init.d/mysql restart"**)
That is enough for MysSQL. However the chances are that PHP was compiled with a different default mysql socket location (e.g. **/tmp/mysql.sock**). In which case you have to edit the php.ini file and find the variable "mysql.default_socket". Set this to the above value
mysql.default_socket = /var/lib/mysql/mysql.socket
And restart Apache to re-read the php.ini file
If you do not have access to the configuration files and system command-line then you must ask your Sys Admin/support to help with this.

Error no. 1251: "Client does not support authentication..."

Error no. 1251: "Client does not support authentication protocol requested by server - consider upgrading MySQL client" occurs when the hashing-method for storing password used by the client differs from the one of the server. Typically it occurs when trying to connect to MySQL 4.1 or 5.x with a client compiled for 3.x or 4.0.

The Client as far as SQL DM goes is either the IderaSQLdmforMySQL-8.7.0-0.exe and sja.exe executable files (with its compiled-in MySQL C-API) or - in case you use HTTP-tunneling - your PHP-binary.

SQL DM itself (the IderaSQLdmforMySQL-8.7.0-0.exe -executable) handles all MySQL versions from 3.23.x and upwards automatically, and the error message should not occur with direct connection. In case you experience the error when HTTP-tunneling, you can EITHER replace the PHP-binary OR downgrade the hash-type for the user used for tunneling (and other PHP based connections) with the following command:

SET PASSWORD FOR 'some_user'@'some_host' = OLD_PASSWORD('newpwd');

The command must be written as it is.

At ISP's you should expect the Sys Admin there to have MySQL-installations and PHP-binaries that "fit". If you operate your own MySQL server the hashing method **may** or **may not** change when upgrading the server from 4.0 to a newer version. That depends on the upgrade method. This one is very similar to "Error no. 1045: Connection denied ..", which you can find it in this list. However, this case **connection** is established OK but **access to data** is denied. The user exists but most like he does not have any privilege at all. About user privileges start, please refer to the issue "I am able to connect but can't see the list of databases/tables" listed in this page.

I am able to connect using phpMyAdmin, but SQL DM will not let me connect.

PhpMyAdmin is running on the server itself so when connecting to MySQL with phpMyAdmin you are NOT connecting from a remote host! With SQL DM you are connecting from a remote host. This is a very important difference as far as user configuration with MySQL is concerned.

The user that you are connecting with maybe has no privilege to connect from remote.
By default, MySQL only gives access for users to connect from localhost. If you want to give some user access to the server from another host you must specify the hostname (an ip or a URL). You can use the SQL wildcards "%" and "_" (but not windows wildcards like "*"). Simply giving permission for a user to access the server from "%" means from everywhere. Read more about the MySQL privileges system in "Error Nº1044".
If the database is at an ISP there usually is some kind of "Control Panel" application available from where to configure users. Often user configuration is only allowed using this tool. It also is very likely that your ISP has blocked direct access to MySQL on port 3306. If that is the case SQL DM offers you the option of using HTTP-tunneling as well as SSH-tunneling.

Error Nº 1044: "Access denied..."

This one is very similar to "Error no. 1045: Connection denied .."

However this case **connection** is established OK but **access to data** is denied. The user exists but most like he does not have any privilege at all. Read more about user privileges in "I am able to connect but cannot see the list of databases/tables" issue listed.

I am able to connect but can't see the list of databases/tables.

You must have at least SELECT privilege (as a global privilege or a SCHEMA/COLUMN privilege, to be able to see any data. More information on the MySQL privilege system can be found The MySQL Access Privilege System.

If the database is hosted at an ISP/hosting provider it is likely that you must use some web-based "Control Panel" application to set up user privileges.

I am getting "Protocol Mismatch; Server Version 9; Client Version 10"

The protocol version 9 is used by MySQL 3.22 and earlier. The current version of SQL DM supports 3.23.x and above. To use it with 3.22 we have to provide you with a special build of SQL DM. Registered users can request such copy from ideramysqlsupport@idera.com. We can not guarantee that it will be the latest version of SQL DM.

The opposite error message appears if you try to connect from an application with a connector (for instance an ODBC-driver) that is too old too be used with the actual MySQL version.

I have an account with Yahoo. Can I use SQL DM…

Yes. But several users have had problems getting connection parameters right. Here is what Yahoo say themselves:

> Why can't I access my database?
>
> First, make sure that you are using the hostname "mysql" and not "localhost" in any of your PHP or Perl configuration files that require a MySQL hostname.

You will need to use HTTP Tunneling. First upload SQL DMTunnel.php (available with the SQL DM installer). Put the correct URL in the HTTP Tunneling field and use the same credentials as you use in your PHP pages. The Hostname should be "mysql" (case-sensitive).

I get error 1130 "Host is not allowed to connect ..." or "Access denied ..." or "Could not connect …"

Error 1130 is a networking error. The server cannot resolve the hostname of the client. Or the host is not allowed to connect to the MySQL server.

There are basically 2 categories of possible reasons:

- **The simple one:**

In MySQL a user a user is specified using BOTH the user name and the host from where the user may connect. If no user has been created where the host-part (using wildcards or not) mathces the host of the client trying to connect MySQL returns this error.

- **Specific for MySQL 5.7:**

When upgrading to MySQL 5.7.3 from a previous version this may occur due to changes to the user table introduced in 5.7.3. There is a good blog about it, Upgrade and Resolving ERROR 1130 Host 'localhost' is Not Allowed to Connect.

- **The tricky ones:**

1. Your hosts file is damaged or invalid. Various vira and spyware attack and alter the host file in various ways. For instance if the hosts file does not contain the line

   > ***127.0.0.1 localhost***

   'Localhost' can not be resolved as pointing to ip 127.0.0.1. On some larger corporate network it is widely used to "roll out" host files to all clients with symbolic names (like 'mysqlserver', 'mailserver' etc.) for important machines on the network and the corresponding IP's. Check with your admin that you got the right file!
2. If you use the Windows network name as hostname, it may be a network configuration problem. Try using the ip instead.
3. On Unix/Linux systems the hosts files sometimes reads
4. Is a variation of number 3. With a complex server setup (involving more IP's, domains, subdomains and/or virtual hosts) a similar issue can occur.

   In this situation HTTP-tunneling will normally work for all users, but often/sometimes direct connection and SSH will not work with the 'root' user - everything depending on the server configuration. Try another and 'ordinary' user account. You may mirror the privileges of 'root' to a 'superadmin' user.

   > ***127.0.0.1 localhost.localdomain***

   This causes a problem with MySQL. MySQL docs at Access Control, Stage 1: Connection Verification say:

   > A Host value may be a hostname or an IP number, or 'localhost' to indicate the local host.

   No mention of 'localhost.localdomain'. This means that MySQL cannot resolve that 'automatically'! This can affect SQL DM when tunneling and SJA for Linux. Workarounds for this include:

I: A workaround that has worked is to give the user access from 'localhost.%'

II: You can add localhost to ip 127.0.0.1 in host file like
a. Make sure your '/etc/hosts' file reads as follows:
127.0.0.1 localhost //localhost *MUST* be first (notice separate entries)
127.0.0.1 localhost.localdomain
127.0.0.1 . . .
b. Make sure you reference the local server in the SJA.XML file as:
127.0.0.1 or localhost.localdomain
This is known to work in some situations where the MySQL configuration file contains:
bind-address = 127.0.0.1
III: It has sometimes worked to add a 'dummy' ip to my.cnf like:
bind-address = 10.10.10.10
.. supposed that 10.10.10.x is also the ip of the local machine. Then you use this 'dummy ip' as the host specification when connecting with SQL DM and SJA.
It seems to be something special for some DEBIAN distributions to use this 'bind-address' construction with MySQL. Solution II) and III) both are solutions that users have contributed at our Forums. Both situations involved DEBIAN.
IV: You can try any host name that the host file maps to ip 127.0.0.1. There might be several! Even a SAMBA NetBIOS alias might work!
V: You can use the local ip (ie. 10.0.0.1 or whatever) of the actual machine or a name server alias for this. But in this case normal TCP-connections must be enabled in MySQL configuration - that is 'skip-networking' must be disabled/commented out and no 'bind-address' may be there. Of course then an additional DNS lookup will have to take place for the connection to be established. This is of no practical importance.

Finally you could test if this connection issue is the same with 'MySQL Administrator'. It uses the same client code (the C-API) and connects exactly as SQL DM and SJA do.

Error no. 1045: "Connection denied..."

The error message: Error No. 1045: Connection denied for 'someuser@somehost' (using password: YES/NO)

It is a user authentication error. The user details specified do not "match" the user tables of the specified MySQL server. Common situations are:

- No such user.

NOTE: MySQL does not use the OS's or domain's user's management. It operates its own user accounts. With a fresh MySQL installation the user ROOT is created with NO PASSWORD. When working with MySQL databases at ISP's an admin user most often must be activated from some web based Control Panel Application before connection to the MySQL server is possible. There could be more "rules" applying here (database and user naming conventions etc). Refer to the docs/support at the ISP for details on that. We can't give them!

- User is not allowed to connect from the actual host. Note that MySQL by default only allows connection from 'localhost'. To specify from where a user may connect SQL wildcards (% and _) can be used. Simply 'someuser@%' means that user 'someuser' may connect from everywhere.
- Wrong password, missing password or password specified where it should not
- If you are upgrading MySQL from an old version (4.0.x or lower) to a more recent (4.1.x or highere) and if you are still using a rather old PHP version you may need to execute this command from command-line client

  SET PASSWORD FOR some_user@localhost = OLD_PASSWORD('newpwd');

  (where some_user@localhost is the user used for this connection) since the format for storing passwords has changed between 4.0.x and 4.1.x versions.

Error No. 2005: Unknown MySQL server host...

The error message: Error No. 2005: Unknown MySQL server host 'some_URL_or_ip'

Simply means that connection is not possible for the following (or similar) reasons:

A protocol is specified in the "MySQL host address" field of the SQL DM Connection Manager that does not support MySQL connection. It is a common mistake among beginners to use "http://...", instead of just "www.myveryowndomain.com" or "sales.myowncompany.biz" or "localhost" (if the webserver and the MySQL server is running on the same computer). When connecting to a remote network you may need to ask the Sys Admin there for the correct URL to use for addressing the MySQL server.

However if you use HTTP-tunneling the URL-field on the Tunnel -tab of the SQL DM Connection Manager should be a complete URL with the "http://" protocol specified. This is because the tunneling script must be addressed through a webserver (that is the idea of HTTP-tunneling!).

Error no. 2003: Cannot connect...

The error message: Error No. 2003: Can't connect to MySQL server on 'localhost' (or some other host)

Simply means that connection is not possible for one of the following (or similar) reasons:

- There is no MySQL server running at the specified host
- Connection to the MySQL server is not allowed using TCP-IP. Check the 'skip-networking' setting in the MySQL configuration file (my.ini on Windows, my.cnf on Unix/Linux). It shall be commented out like '#skip-networking'. If it is not commented out, then do it and restart the MySQL server for the change to take effect. SQL DM needs to connect using TCP-IP.
- Some networking issue prevents connection. It could be a network malconfiguration or a firewall issue. We have experienced sometimes that some firewalls (ZoneAlarm in particular) is blocking TCP-IP connections even if it claims to be disabled. Most often it will help to uninstall and reinstall the firewall.
- When trying to connect to a MySQL server at an ISP this error message often indicates that direct connection to MySQL has been blocked. You must then use HTTP-tunneling or SSH-tunneling to connect.

- Also, you can find more relevant information in Error No. 1130 above. It describes some more special situations when connection to MySQL on Linux.