

The Secrets of Secure Communication

By Adam Hawthorne

In the release of [BBj® 4.0](#) and [PRO/5® 5.0](#), BASIS provides developers with the ability to encrypt communication between virtually all BASIS products, communicate to third party products, and utilize industry-standard encryption technologies. This article describes the technical details of configuring both PRO/5 and BBj to make use of encryption and provides syntax and working examples that make use of these new features.

BASIS' original products did not provide or make use of network facilities available on the supported platforms. Later, PRO/5 added support for network devices, and BASIS released the PRO/5 Data Server®, both of which utilized the networking capabilities of the underlying operating system. BBj continued on this path, supporting several network configurations to allow distinct components of BBj to run on different computers. However, when data moves between computers, new previously inapplicable risks arise. The medium on which the data travels is susceptible to eavesdropping. A program that communicates sensitive information, either to the file system or to the user's display, then puts the data at risk of compromise. To lessen the threat of a malicious attacker gaining access to a customer's sensitive information, BASIS products now include the ability to use [Secure Sockets Layer \(SSL\)](#), the leading security protocol on the Internet.



Figure 1. List of servers and their SSL status

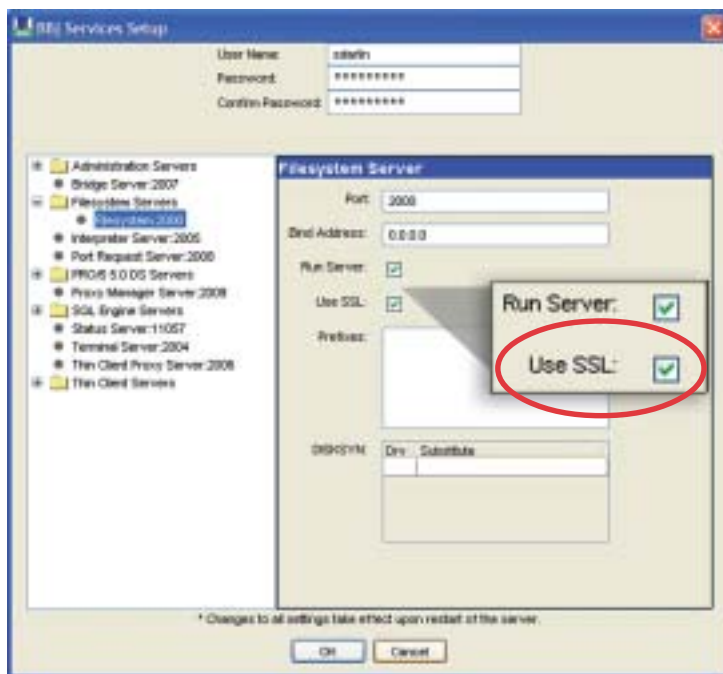


Figure 2. Enable SSL on a particular server

By connecting to a running server and clicking on the BBj Services node in the configuration tree, one can view the list of servers and their SSL status as shown in **Figure 1**. By double-clicking the **BBj Services** node, a user may configure the SSL status of the servers, which then displays the dialog shown in **Figure 2**. After selecting one of the configurable servers, simply toggle the **Use SSL** checkbox as shown. After restarting BBj Services, that server only accepts connections from SSL-enabled clients.

Configuring BBj Components to Use SSL

In BBj, the [Enterprise Manager](#) gives the user access to the configuration of the servers susceptible to eavesdropping, including the Filesystem Server, the SQL Engine Server, the PRO/5 5.0 DS Server, the Administration Server, and the Thin Client Server. Currently, the Thin Client Server allows only one non-SSL server and one SSL server. In the future, Enterprise Manager will support configuring multiple Thin Client Servers of each type.

Several of the servers shown do not have an option to enable SSL. With the exception of the Bridge Server, these servers all listen to connections from localhost only, and are not susceptible to eavesdropping. The Bridge Server does not currently support SSL, but will support it in a future release of BBj.

continued...

Configuring the PRO/5 Data Server to Use SSL

In UNIX environments, the only configuration required by the PRO/5 Data Server is to add the command line option `-e`. For example, the following command runs an encrypted PRO/5 Data Server:

```
pro5.server -e {other options}
```

In the Windows NT/2000/XP operating system, configure the NT Data Server by clicking on the PRO/5 Data Server applet in the Windows Control Panel. This displays a dialog as shown in **Figure 3**. Clicking on the **SSL** checkbox enables or disables SSL in the NT Data Server.

Secure File Access

To connect from a PRO/5 or BBJ program that opens a file on a Data Server with SSL enabled, simply add an `ssl` option to the data server syntax. For example:

```
open(channel) "/<fileserv, port=2100, ssl>/u1/data/DATA01"
```

In BBJ, the [JDBC Driver](#) may also connect to an SSL-enabled SQL Engine Server. To access via the [SELECT](#) or [SQL](#) verbs, add the option `ssl=true` to the connect string of all the databases an SSL-enabled SQL Engine serves.



Figure 3. Enable SSL in PRO/5 Data Server

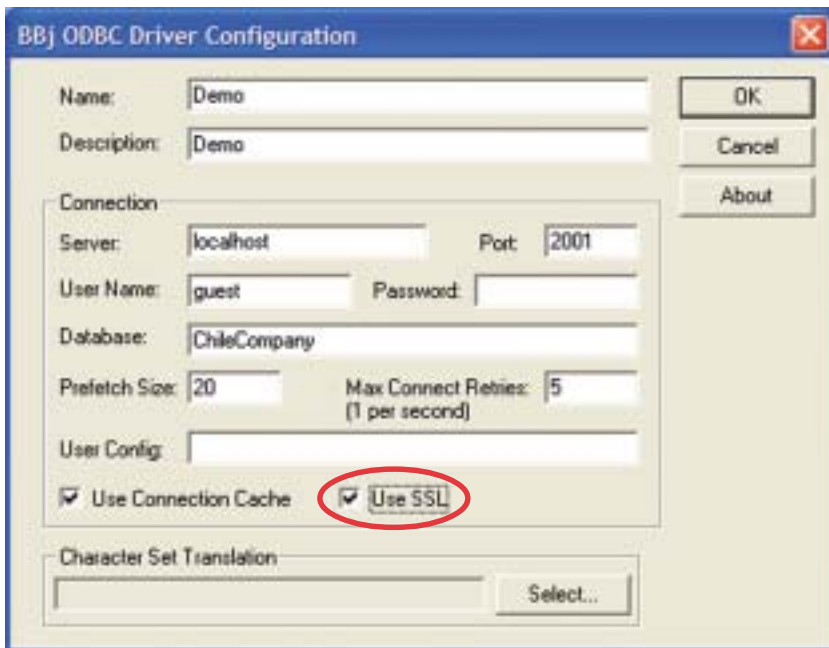


Figure 4. Enable SSL in ODBC

Connecting to an SQL Engine Server from an ODBC driver also requires configuration in order for the connection attempt to succeed. To navigate to the ODBC Data Sources in Windows 2000 and later, go to the Administrative Tools folder. In versions prior to Windows 2000, go to the Control Panel. Click on the appropriate icon to display the dialog for configuring various data sources. To configure a data source that uses the BBJ ODBC Driver, select an appropriate data source, and then click the **Configure** button. When the dialog in **Figure 4** appears, select the **Use SSL** checkbox to enable or disable SSL.

The configuration of a client must match the configuration of the server to which it connects. For instance, if a PRO/5 session connects to an SSL-enabled PRO/5 Data Server, the PRO/5 session must specify the `ssl` mode in the `OPEN` syntax. If the BBJ ODBC Driver connects to an SSL-enabled SQL engine, configure the ODBC driver to use SSL.

Secure Thin Client Access

In BBJ, the display for an interpreter may appear on a different computer than the one actually running the

program. Many times, the data displayed on a user's screen is just as sensitive as the data in a file. For this reason, BASIS also allows SSL connections from the Thin Client to the Thin Client Server. If the secure Thin Client Server is running on the default port (2103), the user may simply add the `-SC` option to the command line of BBJ as follows:

```
bbj -cconfig.bbx -SC -RHbigiron program1.bbj
```

continued...

The -RH option specifies the remote host to which BBJ Thin Client should connect. To connect to a different port, specify the -RP option:

```
bbj -cconfig.bbx -SC -RHbigiron -RP3103 program1.bbj
```

Using SSL in Programs

In addition to giving users the ability to protect the communications between BASIS components, BASIS provides SSL to the programmer. Using SSL as a client is very straightforward. The following configuration and program examples allow the administrator to test whether the SSL is functioning properly:

Configuration in config.bbx

```
ALIAS N5 SSL ""
```

Program

```
REM ' Open a connection to https://www.basis.com/
REM ' Standard https port is 443
OPEN(1,MODE="HOST=www.basis.com,PORT=443")"N5"

REM ' Write the request to the server
write(1)"GET /index.html", $$

REM ' Read server response
while 1
    read(1,end=LOOP_DONE)A$
    print A$
    continue
LOOP_DONE:
    break
wend
close(1)
```



Defining the ALIAS in the [config.bbx](#) tells BBJ or PRO/5 to open the network channel named N5 using SSL. Specify both the HOST and PORT option in the MODE string on the OPEN when making a client connection to an SSL server. Now the developer can use the open channel exactly as any other channel. This program prints the HTML source for the BASIS home page.

Creating an SSL server is slightly more involved. SSL requires the use of certificates to verify to a client that it is connecting to the correct computer. A server may have a certificate that verifies its authenticity. The client verifies that certificate by checking to see if a trusted [Certificate Authority](#) (CA) issued the certificate. If a CA owns the signature on the certificate, the client will continue accepting the connection. Otherwise, the client may choose to disallow connecting to that server.

Obtaining a certificate issued by a well-known CA is usually somewhat expensive. If a casual SSL user chooses to forego this step, the server may use a “self-signed” certificate. Keep in mind, the client only validates self-signed server certificates that appear in the client’s list of trusted certificates.

When PRO/5 acts as an SSL client, PRO/5 does not verify the server’s certificate. The advantage is that clients can connect to servers that use certificates not signed by a CA. However, PRO/5 is vulnerable to a “man in the middle attack” where a computer, pretending to be the server, participates in the SSL key negotiation, decrypts and re-encrypts the data, and forwards the re-encrypted data to the real server. A “man in the middle attack” requires tampering with components such as routers, DNS servers, DHCP servers, and wiring, and is more difficult to accomplish than a packet-sniffing or eavesdropping attack. Future versions of PRO/5 may provide a way to specify a list of trusted certificates.

When PRO/5 acts as an SSL server, it uses a built-in self-signed certificate so there is no need to generate a server certificate for PRO/5. Future versions of PRO/5 may provide a way to specify a user supplied server certificate.

For BBJ, the keytool command that ships with the [Java SDK](#) provides the requisite functionality to produce the needed keys and certificates. The Java documentation includes information to aid one in learning how to use this command. The server

continued...

requires one keystore that contains a private key, the signed certificate (which may be signed by a CA or self-signed), and the public key. The client requires another keystore that contains the same certificate and public key. This sample command generates a keystore file that contains a self-signed server certificate:

```
keytool -genkey -dname "o=BBjTest, c=US" -alias test -validity 7000 -storepass\
serverpwd -keystore serverks
```

These three commands generate a client keystore file that includes the server's certificate on a list of trusted certificates for the client:


```
keytool -export -alias test -storepass serverpwd -keystore serverks -file client.cer
keytool -import -alias test -storepass clientpwd -file client.cer -keystore clientks\
del client.cer
```

With the server keystore in place, the following syntax will open a server socket with the given certificate and key:

```
open(1,MODE="PORT=11000,KEYSTORE=serverks,PASSWORD=serverpwd") "N5"
```

On the client, similar syntax with a different keystore file (and optionally a different password) will connect to the desired port on the server:

```
open(1,MODE="HOST=sslserv,PORT=11000,KEYSTORE=clientks,PASSWORD=clientpwd") "N5"
```

With minimal effort, a developer achieves a much higher degree of confidence in the inaccessibility of their data, regardless of the use and location of the clients. The myriad of methods available to developers to tighten security of their communications provides a great deal of flexibility with the structure and design of their application. 

For more information, visit http://msdn.microsoft.com/library/en-us/security/security/c_gly.asp