



# JTurbo™ 2005

*for Microsoft SQL Server 2000 & 2005*

## User Guide

NEW ATLANTA COMMUNICATIONS, LLC

# JTurbo™ 2005 User Guide

---

November 9, 2005

Version 4.0.0.0



Copyright © 1997-2005 New Atlanta Communications, LLC

100 Prospect Place • Alpharetta, Georgia 30005-5445

Phone 678.256.3011 • Fax 678.256.3012

<http://www.newatlanta.com>

JTurbo is a trademark of New Atlanta Communications, LLC

All other trademarks and registered trademarks herein are the property of their respective owners.

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise without the prior written consent of New Atlanta Communications, LLC.

New Atlanta Communications, LLC makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, New Atlanta Communications, LLC reserves the right to revise this document and to make changes from time to time in its content without being obligated to notify any person of such revisions or changes.

The Software described in this document is furnished under a Software License Agreement ("SLA"). The Software may be used or copied only in accordance with the terms of the SLA. It is against the law to copy the Software on tape, disk, or any other medium for any purpose other than that described in the SLA.

# Contents

<b>1. GETTING STARTED</b> .....	<b>1</b>
1.1 JTURBO 2005 PRODUCT VERSIONS .....	1
1.2 SYSTEM REQUIREMENTS .....	2
1.3 SETTING UP SQL SERVER .....	2
1.3.1 <i>Verify/Enable TCP/IP networking protocol</i> .....	2
1.3.2 <i>Verify/Enable SQL Server Authentication</i> .....	3
1.4 CONNECTING TO SQL SERVER .....	5
1.4.1 <i>Setting the Classpath</i> .....	5
1.4.2 <i>Loading the JTurbo JDBC driver</i> .....	7
1.4.3 <i>Establishing a connection</i> .....	7
1.4.4 <i>JTurbo Properties</i> .....	8
1.5 ENABLING DEBUG MESSAGES.....	10
1.6 TECHNICAL SUPPORT .....	10
<b>2. DATASOURCES</b> .....	<b>11</b>
2.1 JTURBO DATASOURCE TYPES .....	13
2.2 ENABLING DEBUG MESSAGES.....	13
2.3 BASIC DATASOURCE .....	14
2.4 CONNECTION POOL DATASOURCE .....	14
2.4.1 <i>Restarting SQL Server</i> .....	14
2.5 POOL MANAGER DATASOURCE.....	16
2.5.1 <i>Closing the Pool Manager DataSource</i> .....	17
2.5.2 <i>Restarting SQL Server</i> .....	17
2.6 DISTRIBUTED TRANSACTION DATASOURCE .....	17
<b>3. ROWSETS</b> .....	<b>19</b>
3.1 JDBC ROWSET .....	19
3.2 CACHED ROWSET .....	19
3.3 WEB ROWSET .....	20
<b>4. DATA TYPES</b> .....	<b>22</b>
<b>5. ESCAPE SYNTAX</b> .....	<b>23</b>
5.1 SCALAR FUNCTIONS .....	23
5.2 DATE AND TIME LITERALS .....	23
5.3 OUTER JOINS.....	24
5.4 STORED PROCEDURES .....	24
5.5 LIKE ESCAPE CHARACTERS .....	24
<b>6. TRANSACTION ISOLATION LEVELS</b> .....	<b>29</b>
<b>7. RESULTSET TYPES</b> .....	<b>30</b>
<b>8. RESULTSET CONCURRENCY</b> .....	<b>31</b>
<b>9. INTERNATIONALIZATION</b> .....	<b>32</b>
<b>10. KNOWN ISSUES AND LIMITATIONS</b> .....	<b>33</b>
10.1 RESULTSETMETADATA.GETTABLENAME() .....	33
10.2 BLOBS AND CLOBS .....	34
<b>11. PERFORMANCE OPTIMIZATION</b> .....	<b>35</b>



# 1. Getting Started

*The basics of using JTurbo 2005*

**N**ew Atlanta JTurbo 2005 is a Type 4 (“pure Java”) driver that implements the JDBC™ specification for database access as defined by Sun Microsystems, Inc. *JTurbo is certified by Sun for use with Java 2 Platform, Enterprise Edition (J2EE™) branded products.* Additional information about JDBC, including the J2EE certification program, can be found on Sun’s web site:

<http://java.sun.com/products/jdbc/>

JTurbo 2005 delivers high-performance access to Microsoft® SQL Server 2000 and 2005 from any Java-enabled applet, application, or application server. Because Type 4 JDBC drivers are written completely in Java they can run on any operating system that supports the standard Java platform (JDK 1.1 or higher). JTurbo can be used with most application servers, including New Atlanta ServletExec, Macromedia JRun, BEA WebLogic, IBM WebSphere, Apache Tomcat, and others.

## 1.1 JTurbo 2005 Product Versions

JTurbo 2005 comes in three versions; the versions vary based on JDBC specification level supported and JDK version requirements. Each version of JTurbo 2005 is packaged as a separate installer, so be sure to download the appropriate installer.

- **JTurbo 2005 for JDBC 3.0** - this version of JTurbo 2005 implements the JDBC 3.0 specification; it requires JDK 1.4 or higher.
- **JTurbo 2005 for JDBC 2.1** - this version of JTurbo 2005 implements the JDBC 2.1 Core API and the JDBC 2.0 Optional Package (previously referred to as the JDBC 2.0 Standard Extension); it requires JDK 1.2 or higher.
- **JTurbo 2005 for JDBC 1.2** - this version of JTurbo 2005 implements the JDBC 1.2 specification; it is supported on JDK 1.1 or higher, including the Microsoft VM.

After installing JTurbo 2005, you can verify which version was installed by referring to the `Version.txt` file that can be found within the `JTurbo.jar` archive.

## 1.2 System Requirements

All versions of JTurbo 2005 support Microsoft SQL Server 2000 and 2005 running on Windows NT 4.0, Windows 2000, Windows XP and Windows 2003. `JTurbo.jar` can be used on any operating system that supports a compliant Java VM.

## 1.3 Setting up SQL Server

Microsoft SQL Server must have the TCP/IP networking protocol and SQL Server Authentication enabled as described in the following sections to be used with JTurbo.

### 1.3.1 Verify/Enable TCP/IP networking protocol

To enable the TCP/IP networking protocol follow the steps described in the following sections based on your SQL Server version.

#### 1.3.1.1 Microsoft SQL Server 2005

For Microsoft SQL Server 2005, perform the following steps to enable TCP/IP networking (see Figure 1, below):

1. From the Microsoft SQL Server 2005 program group select “Configuration Tools” and then select “SQL Server Configuration Manager”.
2. In the left panel, expand SQL Server 2005 Network Configuration.
3. In the left panel, select the Protocols for each SQL Server instance. If TCP/IP appears “Disabled” for an instance then right click on TCP/IP and select “Enable”.

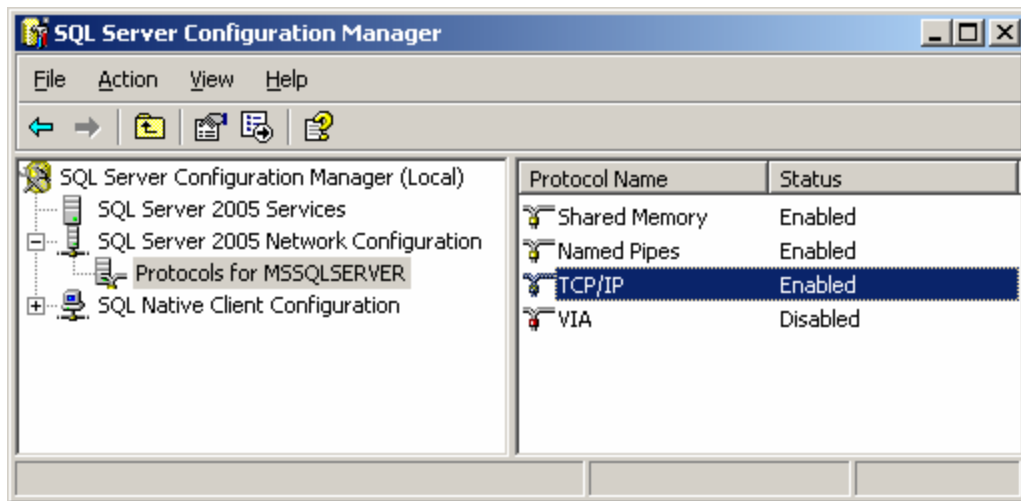
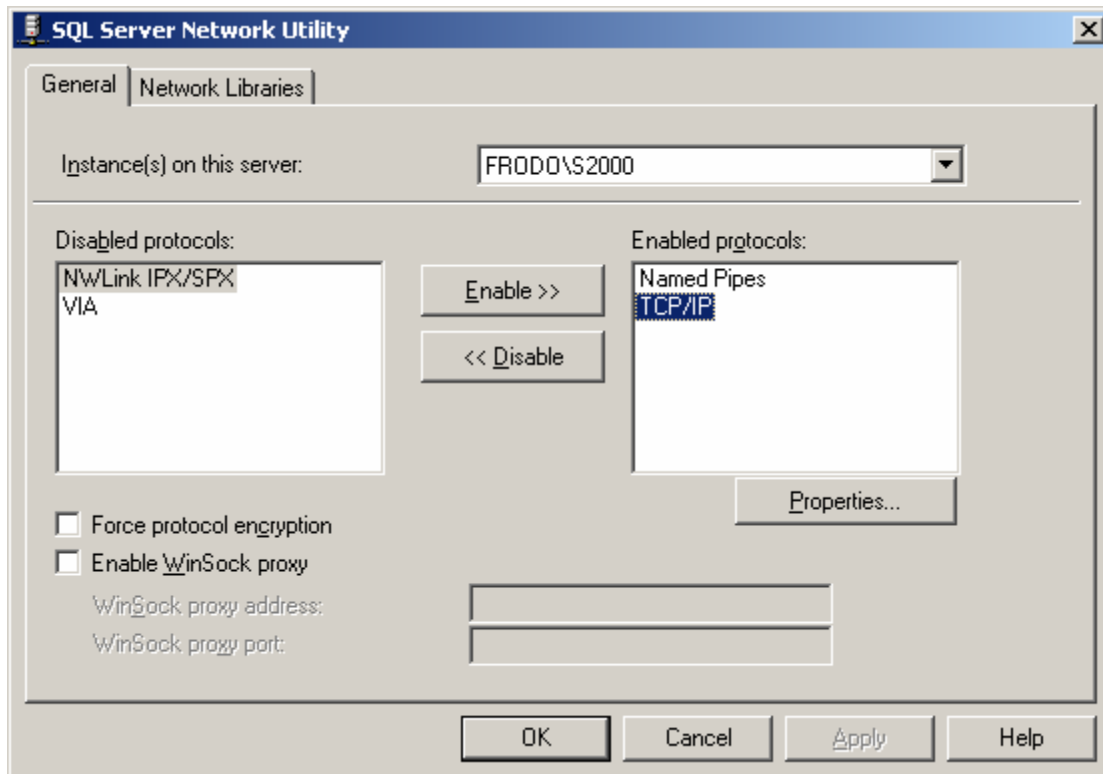


Figure 1. SQL Server 2005 Network Configuration

### 1.3.1.2 Microsoft SQL Server 2000

For Microsoft SQL Server 2000, perform the following steps to enable TCP/IP networking (refer to Figure 2, below):

4. From the Microsoft SQL Server program group select “Server Network Utility”.
5. If TCP/IP appears under the list of disabled protocols then select it, click “Enable”, then click “OK”.



**Figure 2. SQL Server 2000 Network Utility**

### 1.3.2 Verify/Enable SQL Server Authentication

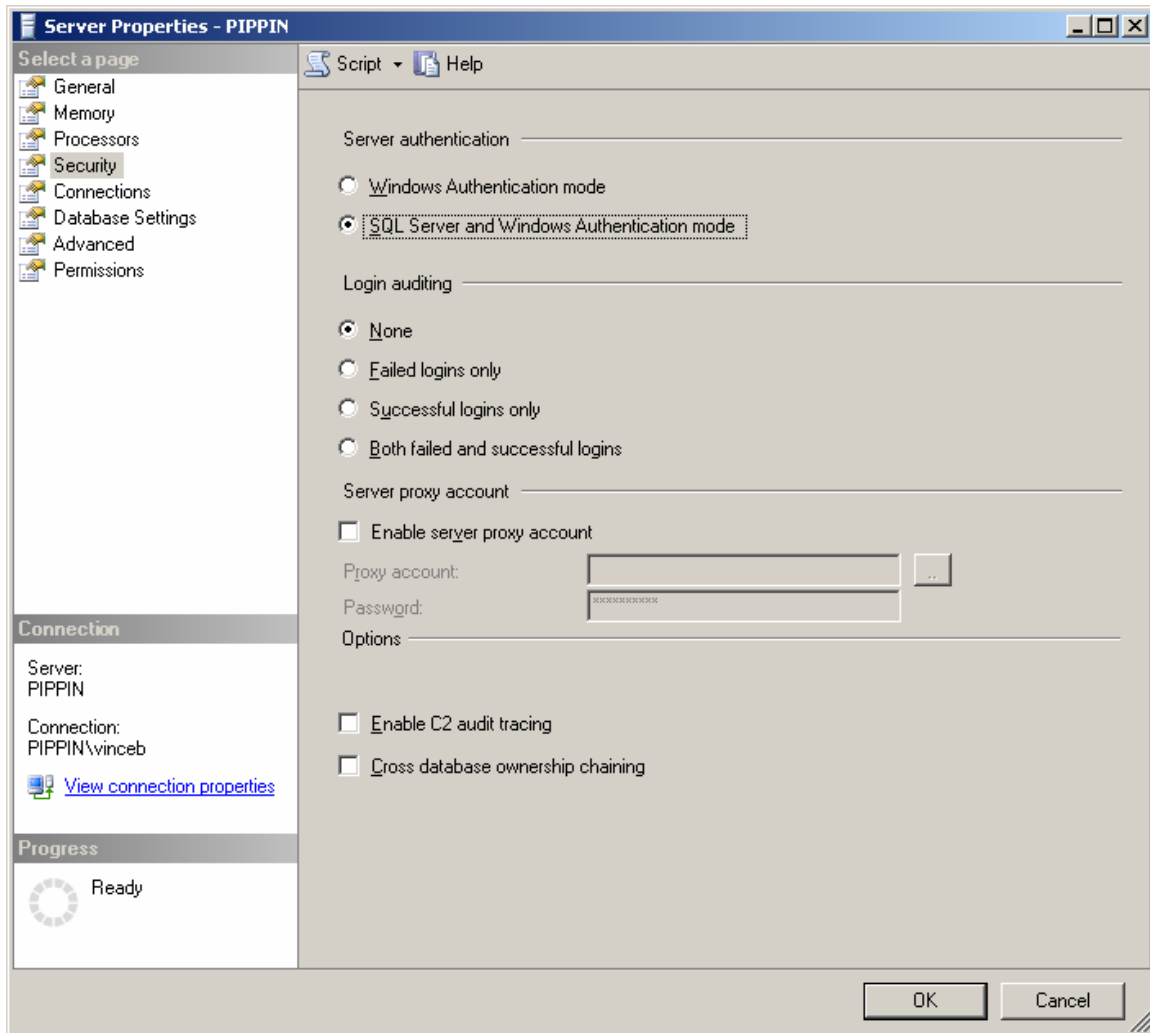
To enable SQL Server Authentication follow the steps described in the following sections based on your SQL Server version.

#### 1.3.2.1 Microsoft SQL Server 2005

For Microsoft SQL Server 2005, perform the following steps to enable SQL Server Authentication:

1. From the Microsoft SQL Server 2005 program group select “SQL Server Management Studio”.
2. Right click on the server and select “Properties”.

3. Select the “Security” page.
4. If “SQL Server and Windows Authentication mode” is not selected then select it and click OK.



**Figure 3. SQL Server 2005 Security**

#### 1.3.2.2 Microsoft SQL Server 2000

For Microsoft SQL Server 2000, perform the following steps to enable SQL Server Authentication (refer to Figure 4, below):

5. From the Microsoft SQL Server program group select “Enterprise Manager”.
6. Right click on the server and select “Properties”.
7. Select the “Security” tab (see Figure 4).
8. If “SQL Server and Windows” is not selected then select it and click OK.

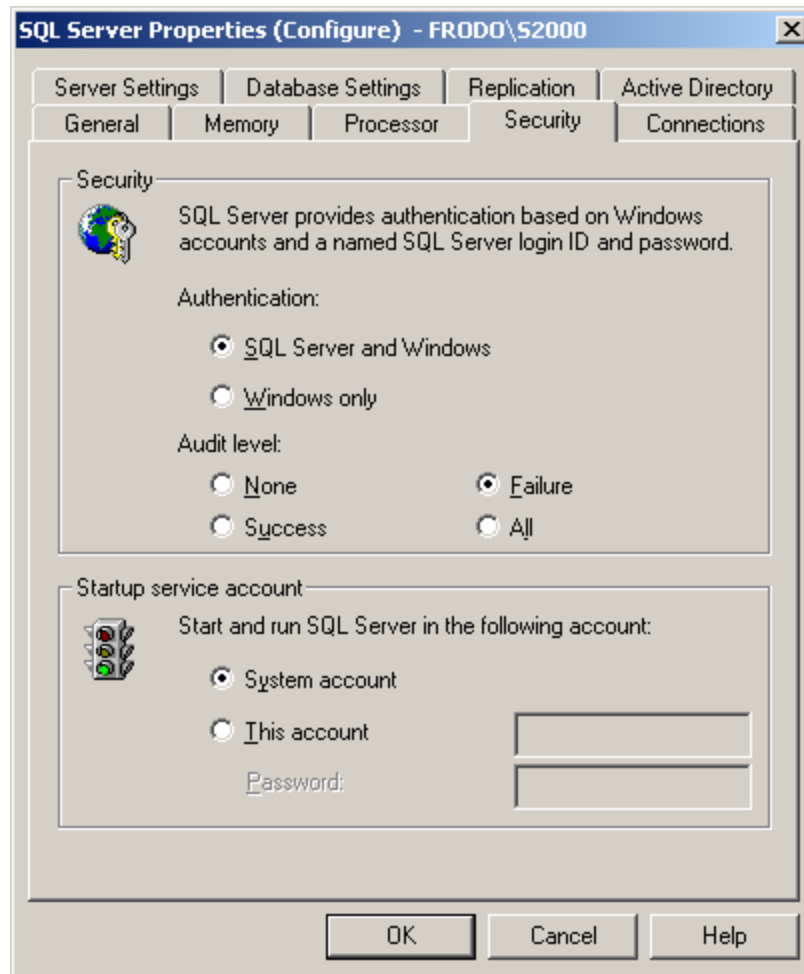


Figure 4. SQL Server 2000 Security

## 1.4 Connecting to SQL Server

This section describes how to make a connection to Microsoft SQL Server using the `JDBC DriverManager` class. **Chapter 2** of this document describes how to make connections using the `JTurbo 2005 DataSource` classes. Typically, the `DriverManager` class will be used within applets and Java applications, and the `DataSource` classes will be used within application servers; however, this may not always be the case.

### 1.4.1 Setting the Classpath

The `JTurbo.jar` archive must be included in the classpath of your applet, application, or application server in order for your code to use the `JTurbo 2005` driver. `JTurbo.jar` can be found in the `lib` subdirectory of the `JTurbo 2005` installation directory (see the `JTurbo 2005 Installation Guide` for additional information).

For applets, include `JTurbo.jar` in the `archive` attribute of the `<APPLET>` tag:

```
<APPLET archive="myApplet.jar,JTurbo.jar"  
        code="myApplet.class" width="100" height="100">
```

Additional information on the <APPLET> tag can be found on Sun's web site:

<http://java.sun.com/j2se/1.3/docs/guide/misc/applet.html>

For applications, JTurbo.jar can be added to the classpath using the `-classpath` option for the `java` command or in the `CLASSPATH` environment variable. Additional information on either of these methods for setting the classpath can be found on Sun's web site:

<http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/classpath.html>

<http://java.sun.com/j2se/1.3/docs/tooldocs/win32/classpath.html>

For application servers, JTurbo.jar must be added to the classpath for the application server. The method of doing this varies based on the application server; refer to your vendor's documentation for additional information. Instructions for adding JTurbo.jar to the classpath for several popular application servers can be found on New Atlanta's web site:

[http://www.newatlanta.com/products/jturbo/self\\_help/docs/third\\_party.jsp](http://www.newatlanta.com/products/jturbo/self_help/docs/third_party.jsp)

#### 1.4.1.1 Additional Archives

If you're using Windows Authentication using NTLM then you'll need to download `gnu-crypto.jar` from the following web site and add it to the classpath:

<http://www.gnu.org/software/gnu-crypto/>

If you're using JTurbo 2005 for JDBC 1.2, you do not need to add any additional archives other than JTurbo.jar to the classpath.

If you're using JTurbo 2005 for JDBC 3.0, you do not need to add any additional archives other than JTurbo.jar to the classpath (note that JTurbo 2005 for JDBC 3.0 requires JDK 1.4 or greater).

If you're using JTurbo 2005 for JDBC 2.1 on a JDK 1.4-compliant VM, you do not need to add any additional archives other than JTurbo.jar to the classpath.

If you're using JTurbo 2005 for JDBC 2.1 on a JDK 1.3-compliant VM and you wish to use the JDBC 2.0 Optional Package features (such as `DataSources` or `RowSets`), then in addition to JTurbo.jar the following archives must be included in the classpath:

```
crimson.jar  
xalan.jar  
jaxp.jar  
jta.jar  
jdbc2_0-stdext.jar
```

If you're using JTurbo 2005 for JDBC 2.1 on a JDK 1.2-compliant VM and you wish to use the JDBC 2.0 Optional Package features, then in addition to `JTurbo.jar` and the four archive files listed above, the following archive must be included in the classpath:

```
jndi.jar
```

All of the additional archive files listed above can be found in the `lib` subdirectory of the JTurbo 2005 for JDBC 2.1 installation directory; see the JTurbo 2005 Installation Guide for additional information.

Note that some application servers, such as New Atlanta ServletExec 4.1, already include several of the additional archives listed above in the classpath as part of the default installation. Refer to your application server vendor's documentation for more information.

### 1.4.2 Loading the JTurbo JDBC driver

Before making a connection to Microsoft SQL Server an application must first load the JTurbo JDBC driver using the following code:

```
Class.forName( "com.newatlanta.jturbo.driver.Driver" );
```

Note that a "Driver not found" error message can result if: (1) `JTurbo.jar` has not been added to the classpath correctly; or, (2) the driver class name has been mistyped.

### 1.4.3 Establishing a connection

After the JTurbo JDBC driver has been loaded, a connection can be made to Microsoft SQL Server using the following code:

```
Connection con = DriverManager.getConnection( "JTurbo URL" );
```

Or:

```
Connection con = DriverManager.getConnection( "JTurbo URL", "username",  
                                             "password" );
```

In the above examples, *JTurbo URL* has the following format:

```
jdbc:JTurbo://<server>:<port>/<database>/<property=value>
```

Please note the following:

1. The `<server>` name is required and can be either the Windows computer name, the server IP address, or the DNS domain name.
2. The `<port>` field is optional and defaults to 1433.
3. The JTurbo URL may specify 0 or more properties (see Section 1.4.4, below).

Here are some examples using the standard URL format:

```
jdbc:JTurbo://myserver:1433/pubs/  
jdbc:JTurbo://myserver/pubs/  
jdbc:JTurbo://myserver/pubs/sp=true
```

#### 1.4.3.1 SQL Server Named Instances

If you're running named instances of SQL Server, do not use the instance name in the *JTurbo URL*. Instead, use the <server> name as described above, and specify the instance to connect to using the <port> field (SQL Server named instances run on unique port numbers).

#### 1.4.3.2 Clustered Environments

An alternate format for the *JTurbo URL* should be used in clustered environments where the server name contains a '/' (such as "mycluster/myserver"):

```
jdbc:JTurbo:alt://<server>:<port>?database=<database>&<property=value>
```

Here are some examples using the alternate URL format:

```
jdbc:JTurbo:alt://mycluster/myserver:1433?database=pubs  
jdbc:JTurbo:alt://mycluster/myserver?database=pubs&sp=true
```

#### 1.4.3.3 Windows Authentication Using NTLM

In order to have JTurbo connect to SQL Server with Windows Authentication using NTLM, you must download the gnu-crypto.jar from the following web site and add it to the classpath:

<http://www.gnu.org/software/gnu-crypto/>

You must also set the JTurbo "authentication" property to "windows", the "clientDomain" property to the domain in which JTurbo is running and the "clientHost" property to the host on which JTurbo is running.

#### 1.4.4 JTurbo Properties

A JTurbo URL may contain one or more properties as defined in Table 1 on the following page. All properties are optional. Note that if the user and password properties are not specified in the JTurbo URL, then they **must** be specified as `DriverManager.getConnection()` parameters (see the examples, above).

Property	Description
user	The user name for logging in to SQL Server.
password	The user's database password for logging in to SQL Server.
language	The language to be used by SQL Server when generating error messages. The languages supported by SQL Server can be determined by executing the SQL statement "select * from syslanguages". The default value is us_english.
charset	The character encoding to be used by the driver when working with char, varchar and text SQL Server data types. If not specified, JTurbo uses the default charset of the platform on which JTurbo is running.
sp	Set to true to create temporary stored procedures for PreparedStatements. This property is obsolete and is only present to allow JTurbo 2005 to behave as JTurbo 3.0.2 . The default value is false.
prepareInline	Set to true to execute PreparedStatements as Statements. This was the default behavior with JTurbo 3.0.2 and earlier. This property has been added to allow JTurbo 2005 to behave as JTurbo 3.0.2 when executing PreparedStatements. The default value is false.
fetchSize	The default fetch size for result sets created by this connection. The default value is 100.
appname	Defines the application name that appears in sysprocesses for connections from JTurbo. This property is ignored when JTurbo is in trial mode. The default value is "JTurbo 4.0.0.0 JDBC X.X Driver" where X.X is 3.0 or 2.1.
authentication	Defines the type of authentication ("sql" or "windows") to be used by JTurbo to connect to SQL Server. The default value is "sql".
clientDomain	When "windows" authentication is used, this property must be set to the domain in which JTurbo is running.
clientHost	Defines the host name that appears in sysprocesses for connections from JTurbo. The default value is the client's IP address. When "windows" authentication is used, this property must be set to the host on which JTurbo is running.

**Table 1. JTurbo URL Properties**

## 1.5 Enabling Debug Messages

An application may enable the logging of debug messages to a `PrintStream` or `PrintWriter` for all loaded JDBC drivers (including JTurbo 2005) by using the following code:

```
PrintStream printStream = new PrintStream( ... );  
DriverManager.setLogStream( printStream );
```

or:

```
PrintStream printWriter = new PrintWriter( ... );  
DriverManager.setLogWriter( printWriter );
```

For example, to have the debug messages logged to the console:

```
DriverManager.setLogStream( System.out );
```

or:

```
DriverManager.setLogStream( System.err );
```

Note that enabling debug messages in this manner only effects JDBC connections that are created using the `DriverManager.getConnection()` method. That is, it does not enable logging for JDBC connections that are created via a `DataSource` (see the next chapter for a discussion of `DataSources`).

## 1.6 Technical Support

If you're having difficulty installing or using JTurbo, check the online Technical Support FAQ:

[http://www.newatlanta.com/products/jturbo/self\\_help/faq\\_list.jsp](http://www.newatlanta.com/products/jturbo/self_help/faq_list.jsp)

You may also want to consider subscribing to the JTurbo-Interest mailing list, a user-supported discussion forum for JTurbo developers:

[http://www.newatlanta.com/products/jturbo/self\\_help/mailling\\_list.jsp](http://www.newatlanta.com/products/jturbo/self_help/mailling_list.jsp)

Details regarding free and paid support options, including online, telephone, and pager based support are available from the New Atlanta web site:

<http://www.newatlanta.com/support/jturbo/index.jsp>

# 2

## 2. DataSources

### *JTurbo DataSources*

**D**ataSources are defined by the JDBC 2.0 Optional Package and the JDBC 3.0 specification. Therefore, data sources are implemented by JTurbo 2005 for JDBC 2.1 and JTurbo 2005 for JDBC 3.0. Data sources are **not** implemented by JTurbo 2005 for JDBC 1.2.

A data source must be deployed before it can be used. Generally, data sources are used within J2EE-based application servers that provide a mechanism for configuring and deploying them (such as New Atlanta ServletExec, BEA WebLogic, or IBM WebSphere); see your application server documentation for detailed instructions. Figure 8, below, illustrates configuring a data source via the ServletExec 4.1 administration user interface.

If you're using data sources within an environment that does not support deploying them, you can deploy a data source in your initialization code as follows:

```
com.newatlanta.jturbo.driver.DataSource ds;  
ds = new com.newatlanta.jturbo.driver.DataSource();  
ds.setServerName( "<server name>" );  
ds.setDatabaseName( "<database name>" );  
  
Context ctx = new InitialContext();  
ctx.bind( "jdbc/<data source name>", ds );
```

After a data source is deployed, it can be used in your application code as follows:

```
Context ctx = new InitialContext();  
DataSource ds = (DataSource)ctx.lookup("jdbc/<data source name>");  
Connection con = ds.getConnection( "<username>", "<password>" );
```

If you are developing an application that does not have access to JNDI then you can still use a data source by creating it and calling its setter methods. For example:

```
com.newatlanta.jturbo.driver.DataSource ds;  
ds = new com.newatlanta.jturbo.driver.DataSource();  
ds.setServerName( "<server name>" );
```

```
ds.setDatabaseName( "<database name>" );  
Connection con = ds.getConnection( "<username>", "<password>" );
```

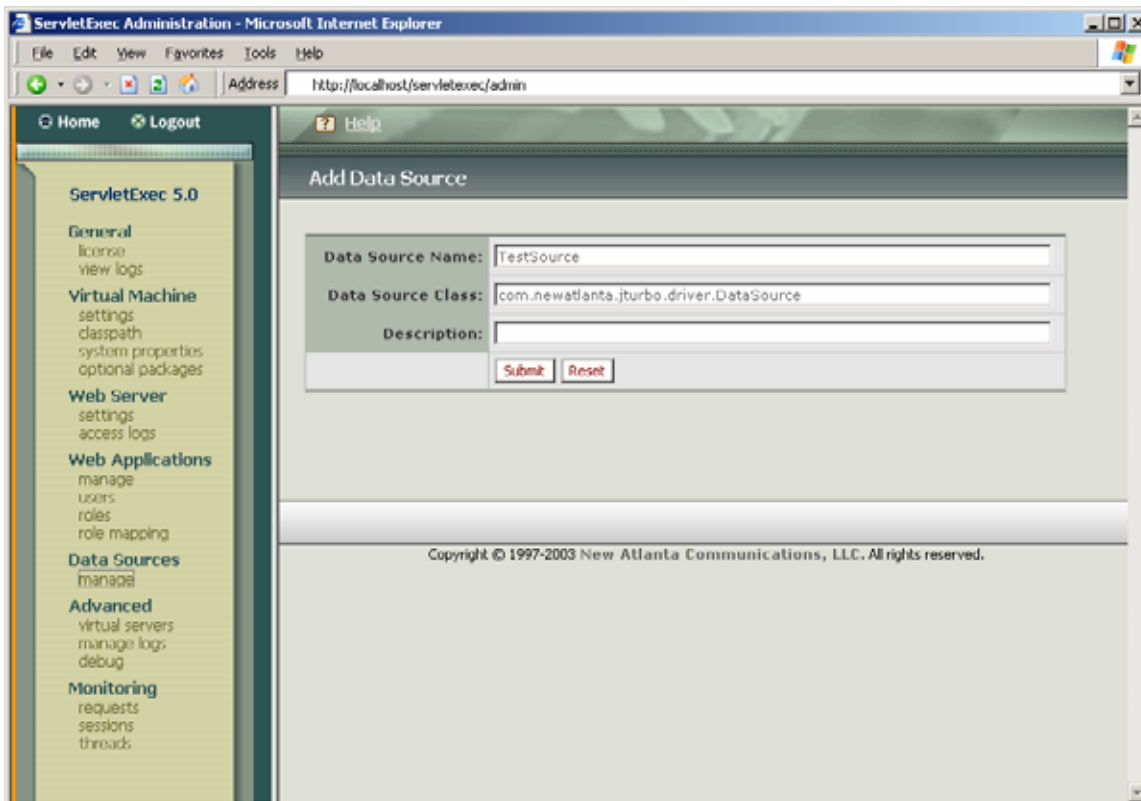


Figure 5. Configuring DataSources in ServletExec 5.0

## 2.1 JTurbo DataSource Types

JTurbo 2005 implements four different data source types, which are listed in Table 2, below. Detailed descriptions of these data sources are provided in the following sections.

DataSource Class	Description
com.newatlanta.jturbo.driver.DataSource	Used when connection pooling is not required.
com.newatlanta.jturbo.driver.ConnectionPoolDataSource	Used when connection pooling is <b>not</b> provided by JTurbo, but by some third-party implementation, such as an application server.
com.newatlanta.jturbo.driver.PoolManagerDataSource	Used when connection pooling is provided by JTurbo.
com.newatlanta.jturbo.driver.JTXADDataSource	Used when connections participate in distributed transactions.

Table 2. JTurbo DataSources

## 2.2 Enabling Debug Messages

Logging of debug messages to a `PrintWriter` for any of the JTurbo data sources can be enabled by using the following code:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/<data source name>");

PrintStream printWriter = new PrintWriter( ... );
ds.setLogWriter( printWriter );
```

For example, to have the debug messages logged to the console:

```
ds.setLogWriter( new PrintWriter( System.out ) );
```

or:

```
ds.setLogWriter( new PrintWriter( System.err ) );
```

## 2.3 Basic DataSource

This data source can be used to get a connection that is not pooled. When deploying this data source the following class should be used:

```
com.newatlanta.jturbo.driver.DataSource
```

This class implements the `javax.sql.DataSource` interface. Table 3 on the next page describes the properties that can be set when deploying this data source, and all other JTurbo data sources.

Note that if the `user` and `password` properties are not configured when the data source is deployed, then they **must** be specified as `DataSource.getConnection()` parameters.

## 2.4 Connection Pool DataSource

This data source can be used in an application or application server that implements connection pooling. This data source will not do the actual connection pooling but will allow the application or application server to do the pooling.

When deploying this data source the following class should be used:

```
com.newatlanta.jturbo.driver.ConnectionPoolDataSource
```

This class implements the `javax.sql.ConnectionPoolDataSource` interface and supports the same properties as the JTurbo basic data source described in Table 3 on the next page.

### 2.4.1 Restarting SQL Server

If SQL Server is restarted and the application or application server that is using the JTurbo Connection Pool DataSource is not also restarted, then connection failure exceptions may occur when an attempt is made to use the connections in the connection pool. These exceptions will cause the connections to be removed from the pool; after all of the “old” connections are removed from the pool, new connections will be created and the application or application server will be able to connect to SQL Server.

Note that this behavior may vary based on the connection pooling implementation of your application or application server.

Property	Description
description	A description of the data source. Optional.
serverName	The database server name. Required; can be either the Windows computer name, the server IP address, or the DNS domain name.
portNumber	The port number where a server is listening for requests. Optional; the default value is 1433.
databaseName	The name of a particular database on a server. Required.
user	The user name for logging in to SQL Server. Optional.
password	The user's database password for logging in to SQL Server. Optional.
language	The language to be used by SQL Server when generating error messages. The languages supported by SQL Server can be determined by executing the SQL statement "select * from syslanguages". Optional; the default value is us_english.
encoding	The character encoding to be used by the driver when working with char, varchar and text SQL Server data types. Optional; if not specified, JTurbo uses the default encoding of the platform on which JTurbo is running.
sp	Set to true to create temporary stored procedures for PreparedStatements. This property is obsolete and is only present to allow JTurbo 2005 to behave as JTurbo 3.0.2 . The default value is false.
prepareInline	Set to true to execute PreparedStatements as Statements. This was the default behavior with JTurbo 3.0.2 and earlier. This property has been added to allow JTurbo 2005 to behave as JTurbo 3.0.2 when executing PreparedStatements. The default value is false.
fetchSize	The default fetch size for result sets created by this connection. Optional; the default value is 100.
appName	Defines the application name which appears in sysprocesses for connections from JTurbo. This property is ignored when JTurbo is in trial mode. Optional; the default value is "JTurbo 3.0 JDBC X.X Driver", where X.X is 3.0, 2.1, or 1.2.
authentication	Defines the type of authentication ("sql" or "windows") to be used by JTurbo to connect to SQL Server. The default value is "sql".
clientDomain	When "windows" authentication is used, this property must be set to the domain in which JTurbo is running.
clientHost	Defines the host name that appears in sysprocesses for connections from JTurbo. Optional; the default value is the client's IP address. When "windows" authentication is used, this property must be set to the host on which JTurbo is running.

**Table 3. JTurbo Basic DataSource Properties**

## 2.5 Pool Manager DataSource

This data source can be used to get pooled connections. This data source performs connection pooling internally, and should **not** be used in applications or application servers that provide a connection pooling implementation. When deploying this data source the following class should be used:

```
com.newatlanta.jturbo.driver.PoolManagerDataSource
```

This class implements the `javax.sql.DataSource` interface and supports the same properties as the JTurbo basic data source described in Table 3, above. This data source also supports the additional optional properties described in Table 4, below.

Property	Description
<code>initialPoolSize</code>	The number of physical connections the pool should contain when it is created. The default value is 0.
<code>minPoolSize</code>	The number of physical connections the pool should keep available at all times. 0 (zero) indicates that connections should be created as needed. The default value is 0.
<code>maxPoolSize</code>	The maximum number of physical connections that the pool should contain. 0 (zero) indicates no maximum size. The default value is 1.
<code>maxIdleTime</code>	The number of seconds that a physical connection should remain unused in the pool before the connection is closed. 0 (zero) indicates no limit. The default value is 0.
<code>propertyCycle</code>	The interval, in seconds, that the pool should wait before enforcing the current policy defined by the values of the above connection pool properties. The default value is 60.
<code>maxStatements</code>	The total number of statements that the pool should keep open. 0 (zero) indicates that caching of statements is disabled. The default value is 0.
<code>testConnection</code>	Set to true to have the Pool Manager DataSource test a pooled connection to verify it is still good before returning it. The default value is false.

**Table 4. JTurbo Pool Manager DataSource Additional Properties**

### 2.5.1 Closing the Pool Manager DataSource

To insure that all resources it uses are released properly, the JTurbo Pool Manager DataSource should be closed before your application or application server shuts down. This is done by invoking the `close()` method:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/<data source name>");
.
.
(com.newatlanta.jturbo.driver.PoolManagerDataSource)ds.close();
```

Note that the `close()` method is not part of the JDBC DataSource API, but is proprietary to the JTurbo Pool Manager DataSource, therefore you must cast the DataSource variable before invoking the `close()` method, as illustrated above.

### 2.5.2 Restarting SQL Server

If SQL Server is restarted and the application or application server that is using the JTurbo Pool Manager DataSource is not also restarted, then connection failure exceptions will occur when an attempt is made to use the connections in the connection pool. These exceptions will cause the connections to be removed from the pool; after all of the “old” connections are removed from the pool, new connections will be created and the application or application server will be able to connect to SQL Server.

## 2.6 Distributed Transaction DataSource

This data source can be used to get connections that can participate in connection pooling and distributed transactions. This data source should be used in an application or application server that implements connection pooling and distributed transactions.

When deploying this data source the following class should be used:

```
com.newatlanta.jturbo.driver.JTXADDataSource
```

This class implements the `javax.sql.XADataSource` interface and supports the same properties as the JTurbo basic data source described in Table 3, above.

This data source has the following known limitations:

- Connection pooling must be used since the connection cannot be closed before commit or rollback is called. This data source does not implement connection pooling internally, therefore, it must be deployed within an application or application server that provides a connection pooling implementation.
- Doesn't support multiple transaction branches (that is, a transaction branch must be committed, rolled back or forgotten before a new one is started.)

- All of the `XAResource` methods must be called on the same `JTXADaSource` instance.
- After a crash the `recover` method will not return the correct value, so recoveries cannot be done properly after a crash.

# 3

## 3. RowSets

### *JTurbo Supported RowSets*

**R**owSets are defined by the JDBC 2.0 Optional Package and the JDBC 3.0 specification. Therefore, RowSets are implemented by JTurbo 2005 for JDBC 2.1 and JTurbo 2005 for JDBC 3.0. RowSets are **not** implemented by 2005 for JDBC 1.2.

There are three different RowSet implementations provided by JTurbo 2005, as described in the following sections. All implement the `javax.sql.RowSet` interface.

### 3.1 JDBC RowSet

The JDBC RowSet implementation provided by JTurbo is a connected RowSet that simply makes the JTurbo driver look like a JavaBeans component. This facilitates their use within GUI design tools that support setting JavaBeans properties. The class you need to instantiate for a JDBC RowSet is:

```
com.newatlanta.jturbo.rowset.JdbcRowSet
```

Here's an example of populating a JDBC RowSet:

```
JdbcRowSet rowset = new com.newatlanta.jturbo.rowset.JdbcRowSet();
rowset.setUrl( "jdbc:JTurbo://sqlserver/pubs" );
rowset.setUsername( user );
rowset.setPassword( pass );
rowset.setCommand( "select * from table" );
rowset.execute();
```

### 3.2 Cached RowSet

The Cached RowSet implementation provided by JTurbo is a disconnected RowSet that caches the data in memory and makes the JTurbo driver look like a JavaBeans component. The class you need to instantiate for a Cached RowSet is:

```
com.newatlanta.jturbo.rowset.CachedRowSet
```

A Cached RowSet can be populated in two ways as shown below:

```
CachedRowSet rowset = new com.newatlanta.jturbo.rowset.CachedRowSet();
rowset.setUrl( "jdbc:JTurbo://sqlserver/pubs" );
rowset.setUsername( user );
rowset.setPassword( pass );
rowset.setCommand( "select * from table" );
rowset.execute();
```

or:

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( "select * from table" );
CachedRowSet rowset = new com.newatlanta.jturbo.rowset.CachedRowSet();
rowset.populate( rs );
```

After a Cached RowSet has been populated, rows can be inserted, deleted and updated as shown below:

```
// INSERT a row
rowset.moveToInsertRow();
rowset.updateInt( 1, 4 );
rowset.insertRow();
rowset.moveToCurrentRow();
rowset.acceptChanges();

// DELETE a row
// move to the row to be deleted
rowset.deleteRow();
rowset.acceptChanges();

// UPDATE a row
// move to the row to be updated
rowset.updateInt( 1, 4 );
rowset.acceptChanges();
```

Use of a Cached RowSet within JSP pages is discussed in the following article on the JavaWorld web site:

<http://www.javaworld.com/javaworld/jw-02-2001/jw-0202-cachedrow.html>

### 3.3 Web RowSet

The Web RowSet implementation provided by JTurbo is a disconnected RowSet that caches the data in memory, communicates with a servlet (included with JTurbo) to provide data access, and makes the JTurbo driver look like a JavaBeans component. The Web RowSet is particularly well-suited for use within Java applets.

The class you need to instantiate for a Web RowSet is:

```
com.newatlanta.jturbo.rowset.WebRowSet
```

Before using a Web RowSet you will need to install `JTurbo.jar` in a web application server (such as New Atlanta ServletExec) and configure the following servlet (refer to your web application server documentation for instructions on configuring servlets):

```
com.newatlanta.jturbo.rowset.WebRowSetServlet
```

You will also need to have `WebRowSetClient.jar` present in the applet or client application's classpath.

The following code examples, which would be implemented in the applet or client, assume that you have configured this servlet with a name of `WRSServlet`.

A Web RowSet can be populated in two manners as shown below:

```
WebRowSet rowset = new com.newatlanta.jturbo.rowset.WebRowSet();
rowset.setServletURL( "http://<server name>/servlet/WRSServlet" );
rowset.setUrl( "jdbc:JTurbo://sqlserver/pubs" );
rowset.setUsername( user );
rowset.setPassword( password );
rowset.setCommand( "select * from table" );
rowset.execute();
```

or:

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( "select * from table" );
WebRowSet rowset = new com.newatlanta.jturbo.rowset.WebRowSet();
rowset.setServletURL( "http://<server name>/servlet/WRSServlet" );
rowset.populate( rs );
```

After a Web RowSet has been populated, rows can be inserted, deleted and updated as shown in the previous section for a Cached RowSet.

## 4. Data Types

The following table lists the supported SQL Server Data Types and their mapping to JDBC Data Types.

SQL Server Data Type	JDBC Data Type
bigint	BIGINT
binary	BINARY
bit	BIT
char	CHAR
datetime	TIMESTAMP
decimal	DECIMAL
float	FLOAT
image	LONGVARBINARY
int	INTEGER
money	NUMERIC
nchar	CHAR
ntext	LONGVARCHAR
numeric	NUMERIC
nvarchar	VARCHAR
nvarchar(max) (SQL Server 2005 only)	LONGVARCHAR
real	REAL
smalldatetime	TIMESTAMP
smallint	SMALLINT
smallmoney	NUMERIC
text	LONGVARCHAR
timestamp	BINARY
tinyint	TINYINT
uniqueidentifier	CHAR
varbinary	VARBINARY
varbinary(max) (SQL Server 2005 only)	LONGVARBINARY
varchar	VARCHAR
varchar(max) (SQL Server 2005 only)	LONGVARCHAR
xml (SQL Server 2005 only)	LONGVARCHAR

**Table 5. SQL Server Data Types Supported by JTurbo**

## 5. Escape Syntax

### *Supported Escape Syntax*

Escape syntax is defined by JDBC for commonly used features that don't have a standard SQL syntax defined. By using JDBC escape syntax instead of vendor specific SQL, an application will be more portable. The following sections describe the JDBC escape syntax that is supported by JTurbo.

### 5.1 Scalar Functions

JTurbo supports the following escape syntax for scalar functions:

```
{fn <function-name> (argument list)}
```

Here's an example of an escape syntax for a scalar function which resolves to the current date and time:

```
{fn NOW() }
```

Tables 6 through 10 starting on page 26, below, list the JDBC scalar functions and indicate which are supported by JTurbo.

### 5.2 Date and Time Literals

JTurbo supports the following escape syntax for date, time and timestamp literals:

```
{d 'yyyy-mm-dd' }
{t 'hh:mm:ss' }
{ts 'yyyy-mm-dd hh:mm:ss.f . . .'} }
```

Note that the fractional seconds (.f . . .) portion of the timestamp literal can be omitted.

Here's an example of escape syntax for a date, time and timestamp literals:

```
{d '2001-12-10' }
{t '10:36:14' }
{ts '2001-12-10 10:36:14.125' }
```

## 5.3 Outer Joins

JTurbo supports the following escape syntax for outer joins:

```
{oj <outer-join>}
```

where <outer-join> has the form:

```
table {LEFT|RIGHT|FULL} OUTER JOIN {table | <outer-join>}  
ON <search-condition>
```

Here's an example of escape syntax for an outer join:

```
{oj employees LEFT OUTER JOIN payroll ON id=45324}
```

## 5.4 Stored Procedures

JTurbo supports the following escape syntax for stored procedures which do not have a return value:

```
{call <procedure_name> [( <argument-list> )]}
```

and the following escape syntax for stored procedures which do have a return value:

```
{? = call <procedure_name> [( <argument-list> )]}
```

Here's an example of an escape syntax for a stored procedure which has a return value:

```
{? = call insertEmployee(?, ?, ?, ?)}
```

## 5.5 LIKE Escape Characters

In a LIKE clause the percent sign and underscore characters are wildcard characters which may be escaped by preceding them with an escape character. The default escape character is a backslash. JTurbo supports the following escape syntax to specify a different escape character in a LIKE clause:

```
{escape '<escape-character>'}
```

Here's an example of an escape syntax which specifies an ampersand as the escape character in a LIKE clause:

```
{escape '&'}
```

<b>Numeric Functions</b>	<b>Description</b>	<b>Supported</b>
ABS(number)	Absolute value of number	Y
ACOS(float)	Arccosine, in radians, of float	Y
ASIN(float)	Arcsine, in radians, of float	Y
ATAN(float)	Arctangent, in radians, of float	Y
ATAN2(float1, float2)	Arctangent, in radians, of float2 / float1	Y
CEILING(number)	Smallest integer $\geq$ number	Y
COS(float)	Cosine of float radians	Y
COT(float)	Cotangent of float radians	Y
DEGREES(number)	Degrees in number radians	Y
EXP(float)	Exponential function of float	Y
FLOOR(number)	Largest integer $\leq$ number	Y
LOG(float)	Base e logarithm of float	Y
LOG10(float)	Base 10 logarithm of float	Y
MOD(integer1, integer2)	Remainder for integer1 / integer2	N
PI()	The constant pi	Y
POWER(number, power)	number raised to (integer) power	Y
RADIANS(number)	Radians in number degrees	Y
RAND(integer)	Random floating point for seed integer	Y
ROUND(number, places)	number rounded to places places	Y
SIGN(number)	-1 to indicate number is $< 0$ ; 0 to indicate number is $= 0$ ; 1 to indicate number is $> 0$	Y
SIN(float)	Sine of float radians	Y
SQRT(float)	Square root of float	Y
TAN(float)	Tangent of float radians	Y
TRUNCATE(number, places)	number truncated to places places	N

**Table 6. JDBC Numeric Functions**

String Functions	Description	Supported
ASCII(string)	Integer representing the ASCII code value of the leftmost character in string	Y
CHAR(code)	Character with ASCII code value code, where code is between 0 and 255	Y
CONCAT(string1, string2)	Character string formed by appending string2 to string1; if a string is null, the result is DBMS-dependent	Y
DIFFERENCE(string1, string2)	Integer indicating the difference between the values returned by the function SOUNDEX for string1 and string2	Y
INSERT(string1, start, length, string2)	A character string formed by deleting length characters from string1 beginning at start, and inserting string2 into string1 at start	N
LCASE(string)	Converts all uppercase characters in string to lowercase	Y
LEFT(string, count)	The count leftmost characters from string	Y
LENGTH(string)	Number of characters in string, excluding trailing blanks	Y
LOCATE(string1, string2 [, start])	Position in string2 of the first occurrence of string1, searching from the beginning of string2; if start is specified, the search begins from position start. 0 is returned if string2 does not contain string1. Position 1 is the first character in string2.	Y
LTRIM(string)	Characters of string with leading blank spaces removed	Y
REPEAT(string, count)	A character string formed by repeating string count times	Y
REPLACE(string1, string2, string3)	Replaces all occurrences of string2 in string1 with string3	Y
RIGHT(string, count)	The count rightmost characters in string	Y
RTRIM(string)	The characters of string with no trailing blanks	Y
SOUNDEX(string)	A character string, which is data source-dependent, representing the sound of the words in string; this could be a four-digit SOUNDEX code, a phonetic representation of each word, etc.	Y
SPACE(count)	A character string consisting of count spaces	Y
SUBSTRING(string, start, length)	A character string formed by extracting length characters from string beginning at start	Y
UCASE(string)	Converts all lowercase characters in string to uppercase	Y

Table 7. JDBC String Functions

Time and Date Functions	Description	Supported
CURDATE()	The current date as a date value	N
CURTIME()	The current local time as a time value	N
DAYNAME(date)	A character string representing the day component of date; the name for the day is specific to the data source	Y
DAYOFMONTH(date)	An integer from 1 to 31 representing the day of the month in date	Y
DAYOFWEEK(date)	An integer from 1 to 7 representing the day of the week in date; 1 represents Sunday	Y
DAYOFYEAR(date)	An integer from 1 to 366 representing the day of the year in date	Y
HOUR(time)	An integer from 0 to 23 representing the hour component of time	Y
MINUTE(time)	An integer from 0 to 59 representing the minute component of time	Y
MONTH(date)	An integer from 1 to 12 representing the month component of date	Y
MONTHNAME(date)	A character string representing the month component of date; the name for the month is specific to the data source	Y
NOW()	A timestamp value representing the current date and time	Y
QUARTER(date)	An integer from 1 to 4 representing the quarter in date; 1 represents January 1 through March 31	Y
SECOND(time)	An integer from 0 to 59 representing the second component of time	Y
TIMESTAMPADD(interval, count, timestamp)	A timestamp calculated by adding count number of interval(s) to timestamp; interval may be one of the following:  SQL_TSI_FRAC_SECOND, SQL_TSI_SECOND, SQL_TSI_MINUTE, SQL_TSI_HOUR, SQL_TSI_DAY, SQL_TSI_WEEK, SQL_TSI_MONTH, SQL_TSI_QUARTER, or SQL_TSI_YEAR	Y
TIMESTAMPDIFF(interval, count, timestamp)	An integer representing the number of interval by which timestamp2 is greater than timestamp1; interval may be one of the following:  SQL_TSI_FRAC_SECOND, SQL_TSI_SECOND, SQL_TSI_MINUTE, SQL_TSI_HOUR, SQL_TSI_DAY, SQL_TSI_WEEK, SQL_TSI_MONTH, SQL_TSI_QUARTER, or SQL_TSI_YEAR	Y
WEEK(date)	An integer from 1 to 53 representing the week of the year in date	Y
YEAR(date)	An integer representing the year component of date	Y

Table 8. JDBC Time and Date Functions

System Functions	Description	Supported
DATABASE()	Name of the database	Y
IFNULL(expression, value)	Value if expression is null; expression if expression is not null	Y
USER()	User name in the DBMS	Y

**Table 9. JDBC System Functions**

Conversion Functions	Description	Supported
CONVERT(value, SQLtype)	Value converted to SQLtype where SQLtype may be one of the following SQL types: BIGINT, BINARY, BIT, CHAR, DATE, DECIMAL, DOUBLE, FLOAT, INTEGER, LONGVARBINARY, LONGVARCHAR, REAL, SMALLINT, TIME, TIMESTAMP, TINYINT, VARBINARY, or VARCHAR.	Y

**Table 10. JDBC Conversion Functions**

## 6. Transaction Isolation Levels

### *JTurbo Supported Transaction Isolation Levels*

The transaction isolation level indicates the level of interaction possible between concurrent transactions on the same data source. An application can specify the transaction isolation level by calling the `setTransactionIsolation()` method on the `Connection` object. The transaction isolation levels are described in Table 11, below, from least to most restrictive with an indication of whether or not they are supported by JTurbo.

Isolation Level	Description	Supported
TRANSACTION_NONE	Transactions are not supported	N
TRANSACTION_READ_UNCOMMITTED	Allows transactions to see uncommitted changes. This means that dirty, non-repeatable, and phantom reads are possible.	Y
TRANSACTION_READ_COMMITTED	Allows transactions to see only committed changes. This means that non-repeatable and phantom reads are possible.	Y (default level)
TRANSACTION_REPEATABLE_READ	Disallows dirty and non-repeatable reads. This means that phantom reads are possible.	Y
TRANSACTION_SERIALIZABLE	Disallows dirty, non-repeatable and phantom reads.	Y

**Table 11. Transaction Isolation Levels**

## 7. ResultSet Types

### *JTurbo Supported ResultSet Types*

The ResultSet type determines how a ResultSet's cursor can be manipulated and how changes to the database are reflected by the ResultSet. The JDBC ResultSet types are described in the table below with an indication of whether or not they are supported by JTurbo.

ResultSet Type	Description	Supported
TYPE_FORWARD_ONLY	Cursor moves forward only. Changes are reflected in the ResultSet. Maps to a SQL server cursor type of forward-only. (NOTE: this is different from the JDBC API definition which states changes are not reflected.)	Y (default type)
TYPE_SCROLL_INSENSITIVE	Cursor is scrollable. Changes are not reflected in the ResultSet. Maps to a SQL server cursor type of static. (NOTE: must be used with a concurrency level of read-only.)	Y
TYPE_SCROLL_SENSITIVE	Cursor is scrollable. Most changes are reflected in the ResultSet. Maps to a SQL Server cursor type of keyset-driven.	Y
TYPE_SCROLL_SENSITIVE+1  (NOTE: this type is not part of the JDBC API, but is proprietary to JTurbo.)	Cursor is scrollable. All changes are reflected in the ResultSet. Maps to a SQL Server cursor type of dynamic.  Because the cursor is dynamic, this ResultSet type has the following limitations: <ul style="list-style-type: none"> <li>• <code>ResultSet.getRow()</code> always returns -1</li> <li>• The following ResultSet methods always return false: <ul style="list-style-type: none"> <li>○ <code>isFirst()</code></li> <li>○ <code>isLast()</code></li> <li>○ <code>isBeforeFirst()</code></li> <li>○ <code>isAfterLast()</code></li> </ul> </li> </ul>	Y

**Table 12. ResultSet Types**

## 8. ResultSet Concurrency

### *JTurbo Supported ResultSet Concurrency Levels*

The ResultSet concurrency level determines the update functionality supported by a ResultSet. The JDBC ResultSet concurrency levels are described in the table below with an indication of whether or not they are supported by JTurbo.

ResultSet Type	Description	Supported
CONCUR_READ_ONLY	ResultSet cannot be updated. Maps to SQL Server READ_ONLY concurrency option.	Y (default type)
CONCUR_UPDATABLE	Specifies optimistic concurrency control based on a timestamp column (if available) or all nontext, nonimage columns. Maps to SQL Server OPTIMISTIC WITH ROW VERSIONING concurrency option.	Y
CONCUR_UPDATABLE+1 (NOTE: not part of the JDBC API.)	Specifies intent to update locking. If a FETCH statement is issued within a user-defined transaction, an exclusive lock is placed on the data before it is fetched. The exclusive lock prevents others from viewing or changing the data until the lock is released when the transaction closes. Maps to SQL Server SCROLL LOCKS concurrency option.	Y
CONCUR_UPDATABLE+2 (NOTE: not part of the JDBC API.)	Specifies optimistic concurrency control based on all nontext, nonimage columns. Maps to SQL Server OPTIMISTIC WITH VALUES concurrency option.	Y

**Table 13. ResultSet Concurrency**

## 9. Internationalization

### *JTurbo Support for Internationalization*

**J**Turbo provides support for internationalization through use of the `charset` property. If a database contains `char`, `varchar`, `varchar(max)` or `text` SQL Server data types which are not stored using the ISO Latin-1 (8859\_1) character set then the `charset` property should be set to indicate which character set is being used. In your application you should only use Unicode since JTurbo and SQL Server will handle the conversions back and forth between Unicode and the specified character set.

Text within an SQL statement for `nchar`, `nvarchar`, `nvarchar(max)` and `ntext` fields must be specified in the following format:

```
N'<text>'
```

For example:

```
INSERT INTO MyTable ( ncharColumn )  
VALUES ( N'some data for the nchar column' )
```

## 10. Known Issues and Limitations

### *Things to be Aware of*

This chapter contains a list of known issues and limitations with JTurbo 3.0. This list may not be complete because this document is updated infrequently. Please visit New Atlanta's web site for the most recent information:

[http://www.newatlanta.com/products/jturbo/self\\_help/index.jsp](http://www.newatlanta.com/products/jturbo/self_help/index.jsp)

### 10.1 ResultSetMetaData.getTableName()

The `ResultSetMetaData.getTableName()` method returns incorrect values for result sets for which all of the following conditions are true:

- The type is `ResultSet.TYPE_FORWARD_ONLY`
- The concurrency is `ResultSet.CONCUR_READ_ONLY`
- The Statement cursor name is `null`; that is, it has not been set via the `Statement.setCursorName()` method

Note that these conditions are true for `Statements` created using the following methods (that is, for the versions that are not passed the result set type or concurrency parameters):

- `Connection.createStatement()`
- `Connection.prepareCall( String sql )`
- `Connection.prepareStatement( String sql )`

To get correct values for `ResultSetMetaData.getTableName()` from results sets for which the above conditions are true, you can add the qualifier `FOR BROWSE` to the end of your `SELECT` statement. Note that use of the `FOR BROWSE` qualifier for Microsoft SQL Server is not standard or portable to other databases.

## 10.2 BLOBs and CLOBs

JTurbo 3.0 supports the JDBC API methods related to BLOBs and CLOBs, such as `ResultSet.getBlob()` and `ResultSet.getClob()`. However, users of these methods should be aware that the entire data is being returned to the client with every query. That is, JTurbo does not use references the way they are intended for standard BLOBs and CLOBs. This is a limitation of Microsoft SQL Server and is true for all JDBC drivers that claim support for BLOBs and CLOBs with Microsoft SQL Server.

# 11. Performance Optimization

## *Performance Tips*

The following items provide tips for improving performance of the JTurbo JDBC driver.

1. If you are using the version of JTurbo which is limited to the JDBC 1.2 API then you should use the `fetchSize` property along with the `setCursorName()` method in order to retrieve the rows from the database in chunks. This can be extremely useful in cases where a query returns a large number of rows. Without this flag the driver would read all of the rows into memory.
2. Make sure your SQL statements use a more narrow search criteria where ever possible. For example, if you want to select the "name" column from the "employee" table you should use "SELECT name FROM employee" instead of "SELECT \* FROM employee". Also make the query conditions more specific to avoid retrieval of unnecessary data. Having the Columns referred to in the WHERE clause indexed increases the speed of your query.
3. It is good practice to always define the parameter as the same data type as the column it refers to in the table. This includes user-defined data types: if the `member_no` column was defined as type "mem\_type", and `mem_type` was defined as "int", the parameter to the procedure should be defined as type "mem\_type", not "int". By doing this, the SQL Server Optimizer will know that the values it is comparing are the same type and will not mistakenly instruct SQL Server to convert the parameters when the query is executed.

There is one exception to the above guideline. When a column is defined as type "char(n) NULL", SQL Server internally represents that column as though it were defined as "varchar(n) NULL". Thus, there is a distinct difference between the types "char(n)" and "char(n) NULL". Given this fact, when a table has a column defined as type "char(n) NULL", a stored procedure parameter that will refer to that column should be defined as "varchar(n)" to ensure that the optimizer recognizes the two items as having the same data type.

4. Statements that return scrollable result sets are slower than ones which return FORWARD only result sets. Use Scrollable result sets only when necessary.
5. Use Batch updates where ever possible. This definitely boosts the performance.

6. Always reuse the same connection wherever possible. Only close a connection, if you think you are done with that session. Often closing and opening new connections slows down the performance.
7. Avoid the use of DatabaseMetaData methods which return a ResultSet. These methods cause a request to be sent to SQL Server and are costly.