

i-net **OPTA**TM

Februar 2003

Installation Guide

i-net **///**software

Preliminaries

This guide describes the installation of i-net OPTA™ from a ZIP-file.

Please read the contents of the file [readme.txt](#) that comes with i-net OPTA™ for new information that may not be included in this Installation Guide first.

i-net OPTA™ is a JDBC type 4 driver completely written in the Java programming language. This means that it can be used for any platform with a given JVM to connect to Microsoft SQL Server 2000, 7.0, 6.5 and MSDE.

i-net OPTA™ implements JDBC 2.0 with the so called Optional Package as described in SUN's JDBC documentation. i-net OPTA™ also implements true XA-support for MS SQL Server 2000.

i-net OPTA™ is part of the i-net GATE™, i-net software's JDBC type 4 suite for Microsoft SQL Server, Oracle, Sybase.



No part of this publication may be copied, photocopied, reproduced, transmitted, transcribed, or reduced to any electronic medium or machine-readable form without the prior written consent of i-net software.

© 2003 i-net software

i-net OPTA 2000 Version 5.01 for MS SQL Server
Last Modified: 5-Februar-2003

Table Of Contents

Table Of Contents	3
Installation.....	5
Using the driver in an java application	5
Using the driver in an applet.....	5
Install the Resource Manager Proxy for the DTCDataSource.....	5
On windows you can double click on the file ServerSetup.jar or execute	5
Getting Started	6
SQL Server, Java and JDBC Versions	6
Implemented Features.....	6
Check Host Name and Port Number of Your Server	7
Login Types	7
Driver and DataSource Class Name	8
JDBC URL Subprotocols	8
JDBC URL Syntax of the driver.....	8
JDBC URL Syntax of the pooled driver.....	9
Implemented Driver Properties.....	9
Connection Examples	10
ClassicDriver.java	10
PooledDriver.java	11
WithDataSource.java	11
WithPoolDataSourece.java	11
WithPoolManager.java.....	11
WithJDBCRowSet.java.....	11
WithJNDI.java.....	11
WithJndiAndDataSources.java	11
WithJndiAndPoolManager.java.....	11
Package sample.xa.....	11
Named Pipes.....	11
Escape Clauses.....	12
Date and Time.....	12
Stored Procedures	12
Functions.....	12
Numeric Functions.....	12
Time und Date Functions.....	12
Outer Join.....	13
Encrypt.....	13
Character Converting	13
New Datatypes with SQL Server 7.0 and 2000	13
Scrolling Cursor Types.....	14
Locking and concurrency control.....	14
Failover	14
Failover on getConnection()	14

Failover with Clusters.....	14
Debugging / Logging	14
Support	16
Documentation	16
Known Problems	16
Copyright and Support.....	17

Installation

Using the driver in an java application

- Add the file Opta2000.jar to your classpath or extract the jar file in the directory of the application.

Using the driver in an applet

- Add the file Opta2000.jar to the Archive attribute of the applet tag, e.g.:

```
<APPLET CODE="..." Codebase="..." ARCHIVE="Opta2000.jar" WIDTH="100%"  
HEIGHT="100%">
```

```
...  
</APPLET>
```

or extract the jar in the directory of the applet. You need a browser compatible with JDK 1.2.x or higher. Most browsers support JDK 1.1.x, only. A solution is to use the Java Plugin. In this case you need to use the OBJECT tag (IE) or the EMBED tag (NC).

Please note: If you want to employ:

- connection pooling:
You need to download the JDBC 2.0 Optional Package interface (javax.sql.*)
from Sun's website at:
<http://java.sun.com/products/jdbc/download.html#binary>
- the DataSource interface:
You need to download the JNDI 1.2.1 class libraries (javax.naming.*) from Sun's website at:
<http://java.sun.com/products/jndi/#download>
if you do not use the JDK 1.3.x. (J2SE version 1.3).

Install the Resource Manager Proxy for the DTCDDataSource

If you want to use Distributed Transactions with the MS DTC then you need to install the Resource Manager Proxy once on the SQL Server. The DTCDDataSource works with SQL Server 7.0 or higher, only.

Installation with setup:

On windows you can double click on the file ServerSetup.jar or execute

java -jar SetupServer.jar

Manual Installation:

- Copy the file xp_xaprx.dll in the BINN directory of the SQL Server
- execute the following batch in the Query Analyzer

```
use master
exec sp_dropextendedproc 'xp_xa_start'
exec sp_dropextendedproc 'xp_xa_end'
exec sp_dropextendedproc 'xp_xa_forget'
exec sp_dropextendedproc 'xp_xa_recover'
exec sp_dropextendedproc ,xp_xa_prepare'
exec sp_dropextendedproc ,xp_xa_commit'
exec sp_dropextendedproc ,xp_xa_rollback'
exec sp_dropextendedproc 'xp_xa_proxyversion'
```

```
exec sp_addextendedproc ,xp_xa_start', ,xp_xaprx.dll'
exec sp_addextendedproc ,xp_xa_end', ,xp_xaprx.dll'
exec sp_addextendedproc ,xp_xa_forget', ,xp_xaprx.dll'
exec sp_addextendedproc ,xp_xa_recover', ,xp_xaprx.dll'
exec sp_addextendedproc ,xp_xa_prepare', ,xp_xaprx.dll'
exec sp_addextendedproc ,xp_xa_commit', ,xp_xaprx.dll'
exec sp_addextendedproc ,xp_xa_rollback', ,xp_xaprx.dll'
exec sp_addextendedproc 'xp_xa_proxyversion', 'xp_xaprx.dll'
```

Getting Started

SQL Server, Java and JDBC Versions

Java Versions: 1.2.x or higher

JDBC Version: 2.0

SQL Server Version:

Microsoft SQL Server 2000

Microsoft SQL Server 7.0

Microsoft SQL Server 6.5

MSDE

Implemented Features

JDBC 2.0 core API

DataSource

Connection Pooling

A pooled driver (JDBC 2.0) with internal pooling.

A pool manager (i-net PLEXA™)
JDBCRowSet
CachedRowSet
Distributed Transactions (XA) (with and without MSDTC)

Check Host Name and Port Number of Your Server

This driver works with Microsoft SQL Servers that are configured to use the TCP/IP networking protocol. Please verify that your server is currently listening on a TCP/IP port.

If you know that your SQL Server is listening on a TCP/IP port and you know the host name of your SQL Server and the port number you can go to the next chapter.

To check or enable the TCP/IP Sockets protocol follow these steps:

For Microsoft SQL Server 6.5:

Click -SQL Setup- in the MS SQL Server program group. If not selected select -Change Network Support-, select -TCP/IP- and enter the port number you want to use (default port: 1433).

If -Change Network Support- is selected, then cancel the setup.

For Microsoft SQL Server 7.0 and 2000:

Click -SQL Server Network Utility- in the Microsoft SQL Server program group. On the general property sheet, click -Add- and select -TCP/IP- under Network libraries.

Enter the port number and the proxy address (if necessary) and click OK. The default port number for the Microsoft SQL Server is usually 1433.

However servers can be configured to listen on any port number.

To make sure that the RDBMS server is listening on the machine name and port number you specified use:

telnet <hostname or ip address> <port number>

If the connection is refused then the hostname or the port number is incorrect.

Login Types

The SQL Server supports three authentication types. These are

SQL Server authentication
Windows NT authentication
Mixed (or both authentications)

i-net OPTA™ is completely written in Java and therefore platform independent (type 4 driver), that is why it can't

read the current NT session account and thus can only connect with a SQL Server login. If you want to employ the i-net driver you need to enable "SQL Server authentication" or "Mixed authentication".

To verify the authentication type:

select the SQL Server in the Enterprise Manager
open the properties dialogue
select the security tab

You also need to create a SQL Server login (the default is "sa" with "").

Driver and DataSource Class Name

The class name of the:

driver is:	com.inet.tds.TdsDriver
pooled driver is:	com.inet.pool.PoolDriver
simple DataSource is:	com.inet.tds.TdsDataSource
pooled DataSource is:	com.inet.tds.PDataSource
XADataSource with DTC is:	com.inet.tds.DTCDataSource
simple XADataSource is:	com.inet.tds.XDataSource

For a description of the properties and methods of the DataSources see in the API doc of TdsDataSource because all other DataSource extends from it.

JDBC URL Subprotocols

jdbc:inetdae7:	support of the SQL Server 7.0 (or higher) features with unicode data types nText, nVarchar and nChar (the same as ...&sql7=true)
jdbc:inetdae7a:....	support of the SQL Server 7.0 (or higher) features with ASCII data types Text, Varchar and Char

The differences between inetdae7 and inetdae7a are only on save data. With both subprotocols you can read all data types. On save data the driver does not know if you want save ASCII or unicode. If you want save both data types together then you can use 2 connections. Or you use the subprotocol inetdae7a and use the convert() function in the SQL expression with varchar/char larger 4000 and text data type.

jdbc:inetdae6:	SQL Server 6.5 compatible mode (passwords are transferred plain)
jdbc:inetdae:	only for compatibility with older versions of the driver
jdbc:inetpool:	pooled driver, see the section "JDBC URL Syntax of the pooled driver"

JDBC URL Syntax of the driver

jdbc:inetdae7:hostname:portnumber	
jdbc:inetdae7:hostname	-> with default port 1433

jdbc:inetdae7:hostname:portnumber?database=MyDb&language=us_engli
sh

> with properties

jdbc:inetdae7:hostname/instancename ⇨ with named instance of MS SQL Server 2000

jdbc:inetdae7://servername/pipe/pipeName ⇨ with named pipes

jdbc:inetdae7://servername/pipe/MSSQL\$instancename/sql/query ⇨ named instance of MS SQL Server 2000

JDBC URL Syntax of the pooled driver

If you would like to use the pooled driver then you can use one of the following JDBC url's.

There is no difference in the functionality of these both url's.

jdbc:inetpool:<subprotocol with parameter>

jdbc:inetpool;jdbc:<subprotocol with parameter>

Examples

jdbc:inetpool:inetdae7:www.inetsoftware.de:1433

jdbc:inetpool:inetdae7:localhost:1433

jdbc:inetpool:inetdae7://MyServer/pipe/sql/query

jdbc:inetpool;jdbc:inetdae7:www.inetsoftware.de:1433

jdbc:inetpool;jdbc:inetdae7:localhost:1433

jdbc:inetpool;jdbc:inetdae7://MyServer/pipe/sql/query

Implemented Driver Properties

database	default is the user default database
language	default is "" "" -> SQL Server default language
user	
password	
initSQL	This expression is executed once per connection.
charset	See chapter Character Converting
nowarnings	"true" getWarnings() returns null
logging	default is "false" If the value is true and the JDBC logging is not enabled then the driver will set it to System.out.
sql7	default is "false" with "inetdae" "true" the new datatypes are supported this property only works with the classic subprotocol "inetdae"
prepare	default is "true", PreparedStatements are prepared on the SQL Server, "false" the driver sends for every call of execute() all data.
prepareLiveTime	default is 60 Set the time in seconds that an unused PreparedStatement handle is live before it is unprepared. A value of 0 means an unlimited live time.

loginTimeout	override the value from DriverManager.getLoginTimeout()
queryTimeout	default is 5 * loginTimeout Set the default query timeout for all Statements.
JAVA_OBJECT	default is "true" The method getObject() convert valid Java serialized data in IMAGE columns to a Java Object. This is required to implement the type JAVA_OBJECT but it can be fatal if you implement your own Object serialisation. The best solution if you use your own Java serialisation is to use getBytes() and not getObject().
appname	Application name (only Application and Enterprise Licenses)
useCursorsAlways	default is "false" "true" the method executeQuery() always uses server cursors.
useInsteadOfTrigger	default is "false" "true" a query with multiple tables are updated in one step with updateRow(). You need a INSTEAD OF trigger.
impltran	default is "true" for compatibility with the JDBC-ODBC bridge. If the value is true then the method setTransactionIsolation() will call setAutoCommit(false) implicitly and start a transaction.
fulltran	default is "true", creates always a transaction with setAutoCommit(false); "false" only if needed
host	overrides the value in the JDBC URL
instance	overrides the value in the JDBC URL
port	overrides the value in the JDBC URL
failover	see chapter Failover on getConnection()

There are three ways to put the properties to the driver:

1. append the properties to the URL like this
jdbc:inetdae7:hostname:portnumber?database=MyDb&language=deutsch
2. call method getConnection(String url, Properties info) from the driver manager.
3. create a DataSource and set the properties with the set methods.

Connection Examples

There are eight examples included in the download zip file.

ClassicDriver.java

An example to demonstrate how to connect and use the driver with the JDBC 1.22 interface.

(This driver requires the JDBC 2.0 interface and a JDK 1.2 or higher)

PooledDriver.java

This example shows the use of the PoolDriver. A driver with internal connection pooling.

WithDataSource.java

How to create a connection with a datasource.

WithPoolDataSource.java

How to use the PooledConnection of the driver.

WithPoolManager.java

This demonstrates the use of the PoolManager i-net PLEXA™. i-net PLEXA™ is an example of a pool manager. i-net PLEXA™ comes with i-net OPTA™.

WithJDBCRowSet.java

How to use the JDBCRowSet from i-net software (i-net OPTA™).

WithJNDI.java

This demonstrates the use of a DataSource with a JNDI directory.

WithJndiAndDataSources.java

This sample demonstrates the use of a DataSource with pooling and a JNDI directory.

WithJndiAndPoolManager.java

This demonstrates the use of a ConnectionPoolDataSource with a JNDI directory and the PoolManager (i-net PLEXA™).

Package sample.xa

A sample with EJB and a XADataSource.

Named Pipes

Another solution to connect to the sql server are named pipes. Named pipes work only in the Java VM 1.1.7 or higher and the Java VM 1.2Beta 4 or higher. Named pipes are equal to files with UNC path. We have tested named pipes only with the Win32 VM from Sun. If you want to use named pipes from another platform, you have to install SMB (server message block) on the client or you must install NFS (network file system) on the sql server.

The default pipe of the sql server is "/sql/query", but you can change this pipe name in the server manager.

Escape Clauses

The driver implements escape clauses as follows:

Date and Time

```
{d 'yyyy-mm-dd'}
{t 'hh:mm:ss[.fff]'}
{ts 'yyyy-mm-dd hh:mm:ss[.fff]'}
```

Stored Procedures

```
{call storedProcedures('Param1'[, 'Param2'][, ?][...])}
{? = call storedProcedures('Param1'[, 'Param2'][, ?][...])}
```

examples:

```
st = con.createStatement(); st.execute("{call MyProc('value')}");
pr = con.prepareStatement("{call MyProc('value')}");
pr.execute();
pr = con.prepareStatement("{call MyProc( ? )}");
pr.setString("value");
pr.execute();
```

Functions

Numeric Functions

{fn ABS(number)}	{fn LOG(float)}
{fn ACOS(float)}	{fn LOG10(float)}
{fn ASIN(float)}	{fn PI() }
{fn ATAN(float)}	{fn POWER(number, power)}
{fn ATAN2(float1, float2)}	{fn RADIANS(number)}
{fn CEILING(number)}	{fn RAND(integer)}
{fn COS(float)}	{fn ROUND(number, places)}
{fn COT(float)}	{fn SIGN(number)}
{fn DEGREES(number)}	{fn SIN(float)}
{fn EXP(float)}	{fn SQRT(float)}
{fn FLOOR(number)}	{fn TAN(float)}

Time und Date Functions

{fn now() }	{fn HOUR(datetime)}
{fn curdate() }	{fn MINUTE(datetime)}
{fn curtime() }	{fn MONTH(datetime)}
	{fn MONTHNAME(datetime)}
{fn DAYNAME(datetime)}	{fn QUARTER(datetime)}
	{fn SECOND(datetime)}
{fn DAYOFMONTH(datetime)}	{fn WEEK(datetime)}
{fn DAYOFWEEK(datetime)}	{fn YEAR(datetime)}
{fn DAYOFYEAR(datetime)}	

```
{fn TIMESTAMPDIFF(interval, count, datetime)}
{fn TIMESTAMPADD(interval, datetime, datetime2)}
```

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
```

Outer Join

```
{oj table1 LEFT OUTER JOIN table2 ON table1.id = table2.id}
{oj table1 RIGHT OUTER JOIN table2 ON table1.id = table2.id}
```

Encrypt

```
{encrypt N'password'}
```

Character Converting

By default character converting is disabled. To use character converting in the driver you need to append "charset=YourCharSet" to the url.

for example:

```
"jdbc:inetdae7:localhost:1433?charset=Cp1250"
or
"jdbc:inetdae7:localhost:1433?charset=" +
    sun.io.ByteToCharConverter.getDefault().getCharacterEncoding();
or
"jdbc:inetdae7:localhost:1433?charset=" + System.getProperty("file.encoding");
or
"jdbc:inetdae7:localhost:1433?charset=" + (new
    java.io.InputStreamReader(in)).getEncoding();
```

TIP: The property charset is case-sensitive in JAVA. The name of the character set is also case-sensitive.

You can test the availability of a character set with:

```
"test string".getBytes( charset );
```

If this line executes correctly the character set is available in the current VM.

The parameter charset is ignored on columns with nchar, nvarchar, and ntext. To save national characters with this columns you need to use the character N. Example: "INSERT INTO myTable(ntext field) VALUES(N'national text')"

New Datatypes with SQL Server 7.0 and 2000

The SQL server 7.0 and 2000 supports new datatypes, i.e. nchar, ntext, nvarchar, bigint, variant, varchar larger than 255 character.

You can use the new datatypes if you use the jdbc url "jdbc:inetdae7:..." or set the property sql7=true.

If you set the property sql7=true you will not be able to connect to the SQL Server 6.5.

If you want use the ASCII data types Text, Varchar(8000) and Char(8000) then you should use the subprotocol inetdae7a.

Scrolling Cursor Types

Id	Description	
TYPE_FORWARD_ONLY	Forward-only Dynamic Cursor	The fetch functions will allow only a fetchtype of FIRST, NEXT, or RELATIVE with a positive rownum
TYPE_SCROLL_INSENSITIVE	Insensitive Keyset Cursor	Use a concurrency of CONCUR_READ_ONLY. SQL Server will generate a temporary table, so changes made to the rows by others will not be visible through the cursor. The fetch functions will allow all fetchtype values.
TYPE_SCROLL_SENSITIVE	Keyset Cursor	The fetch functions will allow all fetchtype values.
TYPE_SCROLL_SENSITIVE+1	Dynamic Cursor	The fetch functions will allow all fetchtype values except ABSOLUTE. All position-reporting methods returns always false.

Locking and concurrency control

Concurrency type is one of the following concurrency control options.

Concurrency Type	Description
CONCUR_READ_ONLY	Read -only cursor. You cannot modify rows in the cursor result set.
CONCUR_UPDATABLE	Optimistic concurrency control using timestamp or values. Changes to a row that are initiated through the cursor succeed only if the row remains unchanged since the last fetch. Changes are detected by timestamps or by comparing all nontext, nonimage values comparing if timestamps are not available.
CONCUR_UPDATABLE+1	Intent to update locking. Places an update intent lock on the data page that contains each row as it is fetched. If not inside an open transaction, the locks are released when the next fetch is performed. If inside an open transaction, the locks are released when the transaction is closed.
CONCUR_UPDATABLE+2	Optimistic concurrency control using values. Changes to a row through the cursor succeed only if the row

	remains unchanged since the last fetch. Changes are detected by comparing all nontext, nonimage values.
--	---

Failover

Failover on getConnection()

If you set the property failover=true in the JDBC URL or in a DataSource then after an error has occurred the driver tries to connect to a failover server.

All properties in the JDBC url or DataSource that start with "failover" overwrite the properties from the first connection. All other properties will still be in effect. Therefore you can set all or only some properties

(e.g. host, port, instance user, password) for the failover connection.

For example:

```
jdbc:inedae7:YourHost?database=northwind&failover=true&failoverhost=YourHost2  
jdbc:inedae7:YourHost?failover=true&failoverport=1500
```

Failover with Clusters

With the interface FailoverRunnable you have the possibility to write program code that will be executed correctly in the case of a failure of the cluster server, also. You can configure it with the methods setFailoverTimeout() and setFailoverCount().

For more information please refer to the API documentation of the PoolManager.

Debugging / Logging

If you have problems with the driver then you can enable the logging of the driver with:

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

You can also use any other PrintStream to make this output. This can be used

- if you cannot see the default PrintStreams
- If you use the driver in an servlet, .jsp script or in an Application Server then you need to set an PrintStream on an log file:

```
PrintStream ps = new PrintStream( new FileOutputStream( "c:\\driver.log" ) );  
DriverManager.setLogStream( ps );
```

Now the driver prints the logging output in the file that you have specified.

Please note: Enable the logging of the driver only when you need it to find a problem because the driver performance will be better without the logging.

Support

Please read this file and our FAQ at:

<http://www.inetsoftware.de/English/Produkte/JDBCTreiber/Faq.htm>

If you have not found your problem in the FAQ then post your problem to our newsgroup providing as much information of what occurred or happened as possible. You can find more info about our support at:

<http://www.inetsoftware.de/English/Produkte/Opta/Support.htm>

Documentation

You can find the documentation at:

<http://www.inetsoftware.de/English/Produkte/OPTA/documentation.htm>

Known Problems

- The method `getTableName()` requires a server cursor or the "for browse" statement.
- Calling stored procedures with output parameters from type binary or char produce output with value 0x00 and spaces cut. (e.g., you have a sp with the parameter "char(10) output" and the output value is "abc " you receive the value "abc").

The following methods are not supported by this release:

- `PreparedStatement.setArray (int i, Array x)`
- `PreparedStatement.setRef (int i, Ref x)`
- `CallableStatement.getObject (int i, java.util.Map map)`
- `CallableStatement.getRef (int i)`
- `CallableStatement.getArray (int i)`
- `DatabaseMetaData.getUDTs (String catalog, String schemaPattern, String typeNamePattern, int[] types)`
- `ResultSet.getObject (int i, java.util.Map map)`
- `ResultSet.getRef (int i)`
- `ResultSet.getArray (int i)`
- `Connection.getTypeMap ()`
- `Connection.setTypeMap (java.util.Map map)`
- The `recover()` methods works only in one JVM. After a crash of the JVM the SQL Server will rollback all transactions.

Copyright and Support

© 2000-2002 by i-net software

More info and updates are located at

<http://www.inetsoftware.de>

<news://news.inetsoftware.de>