Connect your apps to DB2 with high-security Kerberos

A programmer's setup guide

Smitha Pambalath (smitha.pambalath@in.ibm.com)  
Software Engineer  
IBM

Anuruddha Kumar Pandey (anurudpa@in.ibm.com)  
Software Engineer  
IBM

This tutorial is a primer to help programmers using IBM Data Server Drivers get applications quickly running in a Kerberos environment. We will be setting up a simple Kerberos environment on Windows, configuring DB2 to use Kerberos authentication, and enabling the client drivers to securely authenticate using Kerberos.

Introduction

In today’s world of online data proliferation, data on the cloud is a major concern when it comes to data privacy and security. More than ever, there is a need to minimize, if not eliminate, security breaches and make data sharing more protected over the wire. Kerberos is a network authentication protocol designed to provide strong authentication for client-server applications. Here, we will focus on securing authentication using Kerberos for applications that use IBM Data Server Driver for JDBC and SQLJ, CLI, and .NET drivers.

What is Kerberos?

Kerberos is a network authentication protocol that provides a strong means of authentication for client-server applications.

The term Kerberos is derived from Cerberus, the three-headed watchdog of the Greek god Hades. The three heads of this mythological creature are used as an analogy for the three crucial entities that make up a Kerberos environment:

- The client user-client machine
- The Key Distribution Center (KDC)
With Kerberos, a client machine can prove its identity to a server, even when the two are on an insecure network. Instead of flowing user IDs and passwords over a network, encrypted tickets are used here. These tickets are issued by an entity called the Kerberos authentication server.

Kerberos uses two types of credentials for authenticating: tickets and authenticators:

- A ticket is used to pass the identity of the client securely to the server for whom the ticket was issued. It also contains information the server can use to ensure that the client using the ticket is the same client to whom the ticket was issued. A ticket is generally ideal for a single server and a single client. It contains the client's name and network address, the server's name, a timestamp, and a session key. This information is encrypted with the server's secret key. The client can use the ticket multiple times to access the server until the key expires. The client can't decrypt the ticket, but it can present the ticket to the server in the encrypted form. No one listening on the network can read or modify the ticket as it passes through the network.
- An authenticator is the additional credential presented with the ticket. The authenticator is generated by the client every time the client wishes to access the server. The authenticator contains the client's name, a timestamp, and an optional additional session key, all encrypted with the session key shared between the client and the server. Unlike the ticket, the authenticator can only be used once. However, since the client can generate authenticators as needed (it knows the shared secret key), this is not a problem.

This tutorial will help readers set up a Kerberos environment on Windows, configure a DB2® server, and configure client drivers to use Kerberos to authenticate users. Throughout, there will be references to "client drivers." These are the IBM Data Server Drivers for JDBC and SQLJ (JCC Driver); IBM Data Server Driver for ODBC and CLI (CLI Driver); the .NET driver; and an interactive client tool called CLPPlus, which is capable of executing SQL statements, scripts, and generating custom reports. These come packaged with the IBM Data Server Driver Package (DS Driver). The DS Driver package contains drivers and libraries for various programming language environments. The JCC driver also comes with the DB2 installation.

The Figure 1 below shows the end-to-end Kerberos authentication technique and how the various components work together.
The Kerberos protocol flow shown above is as follows:

1. Applications can use Kerberos technology by using any one of the DB2 Connect™ drivers, such as the JCC driver, CLI driver, .NET driver, or CLPPlus tool. CLPPlus uses the JCC API internally to authenticate via Kerberos.
2. The first exchange takes place between the client driver and authentication server. The authentication server authenticates the user (by validating the user ID and password, for example). After successful authentication, the authentication server obtains the user's secret keys and returns a ticket-granting ticket (TGT), which is used to get the credentials to grant access to the DB2 server.
3. Upon receiving the TGT, the client sends a request (containing the TGT) for a service ticket to the ticket-granting server (TGS). The TGS authenticates the TGT and returns a service ticket to the client.
4. The client now has the service ticket that allows the client to communicate with the DB2 server providing a service that application wants to use from the DB2 server. The DB2 server can verify the service ticket without contacting the KDC.

Configuring Kerberos for use with IBM Data Server Driver for JDBC and SQLJ on Windows

Our Kerberos setup has a Windows Active Directory Domain Controller acting as the Key Distribution Center (KDC). In order for Java™ to recognize this environment, extra configurations are needed on the client and the server side, which we will deal with later.

Three basic areas of configuration

- Configure a Kerberos KDC or Kerberos server
• Set up a DB2 database to authenticate with Kerberos
• Configure the client drivers to use Kerberos authentication

Steps to set up the KDC

Kerberos KDC is the network service that supplies session tickets and temporary session keys to users and computers within an Active Directory domain. The KDC runs on the domain controller as part of Active Directory domain services (AD DS). So on setting up an AD DS, we can exploit the capability of Kerberos to authenticate users that request access to the DB2 database. The DB2 database can be installed on any machine that is part of this domain. In this setup, we use Windows server 2008 R2 to host the KDC, while the DB2 server is installed on a Windows server 2012 R2 machine.

Common environment setup

The first two steps of setting up the KDC and the DB2 server are common across the client drivers. Users looking to set up specific drivers can skip to the respective driver sections below.

Create a KDC

1. **Install the Active Directory Domain Controller.**
2. **Configure the DNS server.**
3. **Set domain security policy.**
4. **Create a new domain user account.**
5. **Add a machine to a domain.**

Setting up the DB2 server

DB2 will always look at the local setup first, then the domain. So while installing DB2, please make sure you don’t have the local user and domain user the same name (if you intend to create a domain user by name *kerbuser*, please make sure that this user does not already exist in the local directory as a local user).

1. **Install DB2 on any of the member machines connected to the domain.**
2. **Restart this database server machine under the new domain name.**
3. **Log in to the machine using the new user.**
4. **On the DB2 server, start DB2 under the domain user account created.**
5. **Update the DB2 server to Kerberos:**
   
   ```bash
   db2 update dbm cfg using authentication Kerberos.
   ```
6. **Restart the DB2 server.**

Configure the client drivers to use Kerberos authentication

Readers looking for setup steps with non-Java drivers can skip ahead to IBM Data Server CLI Driver setup.

Setting up the IBM Data Server Driver for JDBC and SQLJ

When a client application requests a connection, it should do so using a TGT. The default credentials cache will be inspected for a TGT if the user ID/password are not provided. Otherwise, the TGT will be acquired using the user ID and password specified in the connection string.
Obtaining the TGT

In most cases, the TGT is obtained prior to and separate from the database connection. Normally, this is obtained by issuing the command `kinit` from the command line. If the user ID and password are supplied in the connection string, the assumption will be that DB2 will explicitly obtain the TGT to obtain the Kerberos ticket.

If the TGT is returned successfully, a Kerberos 5 ticket will be obtained and sent to DB2 to perform authentication. If the authentication is passed, a connection will be returned to the client.

Before setting up the JDBC driver, there are a few terms to explain:

1. **Java Authentication and Authorization Service (JAAS)** — JAAS is a Kerberos login interface used by DB2 to get the TGT when the client username and password are provided. This login interface requires a JAAS configuration file that specifies `com.ibm.security.jgss.mech.krb5.Krb5JAASLoginModule` as the login module to be used.

2. **IBM JGSS Provider** — IBM JGSS is a Java Generic Security Service Application Programming Interface (GSS-API) framework with Kerberos 5 as the underlying default security mechanism. JGSS API has a Java `kinit` tool. The GSS-API enables programs to access security services by passing credentials between principals. The syntax of `kinit` to connect to the DB2 tool:

   ```
   ```

   The meaning of the different options are:
   - `-f`: forwardable
   - `-F`: not forwardable
   - `-p`: proxiable
   - `-P`: not proxiable
   - `-c`: cache name (that is FILE:D:\temp\mykrb5cc)
   - `[principal]`: the principal name (qwedf qwedf@DB2TEST.com)
   - `[password]`: the principal's Kerberos password

   The GSS-API alone does not provide any security. Instead, security service providers provide GSS-API implementations, typically in the form of libraries installed with their security software. Sensitive application messages can be wrapped or encrypted by the GSS-API to provide secure communication between client and server. Typical protections that GSS-API wrapping provides include confidentiality (secrecy) and integrity (authenticity). The GSS-API can also provide local authentication about the identity of a remote user or remote host.

3. **Kerberos realm** — A Kerberos realm is where the Kerberos database is stored. Each member in a Kerberos realm has a unique identifier called a principal. The Kerberos realm is made up of the KDC and all of its principals. The realm resides securely on one master computer, although a read-only version of the Kerberos database can reside on other Kerberos machines. The Kerberos realm is usually represented with the domain name in uppercase letters.

4. **Kerberos principal** — A Kerberos principal is a service or user known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name in the following
Connect your apps to DB2 with high-security Kerberos

form: primary-name.instance-name@realm-name. The primary name can be the user's name, the host, or the name of the service. An example of a principal in the DB2TEST.com realm is db2kuser@DB2TEST.com.

Configuring IBM Data Server Driver for JDBC and SQLJ driver to use Kerberos

A couple of configuration and policy files need to be set up properly for Kerberos authentication:

1. **Modify Kerberos configuration file (such as krb5.ini)**— This will be used during the `kinit` process. IBM JGSS requires the Kerberos configuration file. The default name and location of the Kerberos configuration file depends on the operating system being used. The default configuration file is searched for in the following order:
   - The file referenced by the Java property java.security.krb5.conf
   - /lib/security/krb5.conf
   - c:\winnt\krb5.ini on Microsoft Windows platforms
   - /etc/krb5/krb5.conf on Solaris platforms
   - /etc/krb5.conf on other Unix platforms

   In the Kerberos configuration file, the user needs to set up the realms, domain_realm, and several features such as default_keytab_name, default_realm, kdc_timeout, etc.

   **Listing 1. Sample krb.ini file for Windows**

   ```
   ***change this*** means to change this to fit your own setting.
   [libdefaults]
   default_keytab_name  = c:\ibmjgss\config\krb5kt
   default_realm = DB2TEST.COM -> change this
   default_tkt_enctypes  = rc4-hmac
   default_checksum  = rsa-md5
   kdc_timesync            = 0
   kdc_default_options     = 0x40000010
   clockskew               = 300
   check_delegate          = 0
   ccache_type             = 3
   kdc_timeout             = 10000000
   [realms]
   DB2TEST.COM= { -> change this
     kdc = yourmachinewithKDC.db2test.com:88 -> change this
   }
   [domain_realm]
   .db2test.com=DB2TEST.COM -> change this
   .db2test.com=DB2TEST.COM -> change this
   ```

2. **Update JAAS configuration file**— The use of the JAAS login feature requires a JAAS configuration file. This file may be specified as the value of the Java property java.security.auth.login.config or as the value of the property login.config.url. in the /jre/lib/security/java.security file. Modify java_home\jre\lib\security\java.security as follows:
   - Create jaas.conf in D:\kerberos (can choose any directory)
Listing 2. Sample jaas.conf file content

```
JaasClient{
    com.ibm.security.auth.module.Krb5LoginModule optional
    debug=true
    useDefaultCcache=false;
};
```

Configure jaas.conf in the java.security file using the following syntax:
```
login.config.url.1=file:D:\Kerberos\jaas.conf (use / instead of \\).
```
- If not present, add or update `security.provider.1=com.ibm.crypto.provider.IBMJCE`.
- If not present, add or update `security.provider.2=com.ibm.security.jgss.IBMJGSSProvider`.

3. Connect to DB2:
   a. **For connection using a username and password** — Set up the data source that will be referenced in your application to access the DB2 database.
      - For authentication using a username and password, you need:
        - **KerberosServerPrincipal** For example `user@DB2TEST.com` (DB2TEST.com) is the domain name, replace it with the domain you just created)
        - **SecurityMechanism** For example `com.ibm.db2.jcc.DB2BaseDataSource.KERBEROS_SECURITY` (value is 11)

Listing 3. Sample code

```
DriverManager interface:
java.util.Properties properties = new java.util.Properties();
properties.put("user", USER);
properties.put("password", PASSWORD);
properties.put("securityMechanism", new String("11")); // 11 is the integer value for kerberos
Class.forName("com.ibm.db2.jcc.DB2Driver");
String url = "jdbc:db2://serverName/STLEC1";
Connection con = DriverManager.getConnection(url, properties);
```

```
Datasource interface:
com.ibm.db2.jcc.DB2SimpleDataSource dataSource = new com.ibm.db2.jcc.DB2SimpleDataSource();
dataSource.setDatabaseName ("STLEC1");
dataSource.setServerName (serverName);
dataSource.setDriverType (4);
dataSource.setPortNumber (50000);
dataSource.setSecurityMechanism(11); // 11 is for kerberos
Connection con = dataSource.getConnection (USER, PASSWORD);
```

b. **For connection without a username and password** — For connection without a username and password, the ker.util process is used, where the application does not have to mention the username and password. A user principal keeps its Kerberos credentials in a credentials cache. The user credentials cache is located in the following order:
   1. The file referenced by the Java property KRB5CCNAME.
   2. The file referenced by the environment variable KRB5CCNAME.
   3. /tmp/krb5cc_uid on Unix systems.
   4. (user.home)/krb5cc_(user.name).
   5. (user.home)/krb5cc [if user.name cannot be obtained].
Set up the data source that will be referenced in your application to access the DB2 databases. For authenticating without a user name and password (using tickets) you only need:

- **KerberosServerPrincipal** *(user)@DB2TEST.com* (Where DB2TEST.com is the domain name, replace it with the domain you just created)

In this case, you must use the cache and populate it using `kinit`. You don't have to use jaas.conf in the java.security file.

1. Start kinit by executing the command:
   ```
   C:\SQLLIB\BIN>java com.ibm.security.krb5.internal.tools.Kinit
   Password for db2kuser@DB2TEST.com:
   ```
2. Enter the password and you will see a message like this:
   ```
   Done!
   New ticket is stored in cache file C:\Users\db2kuser\krb5cc_db2kuser.
   ```
3. Once this is no longer needed, delete the cache that was generated by going to the location 'C:\Users\db2kuser\krb5cc_db2kuser'.

**Listing 4. Sample code for connecting to DB2 without username and password:**

```java
DriverManager interface:
java.util.Properties properties = new java.util.Properties();
properties.put("securityMechanism", new String("11")); //11 is the integer value for kerberos
properties.put("KerberosServerPrincipal", "machinename.db2test.com@DB2TEST.COM"); //the name DB2 registers with KDC.
Class.forName("com.ibm.db2.jcc.DB2Driver");
String url = "jdbc:db2://HOSTNAME/STLEC1";
Connection con = DriverManager.getConnection(url, properties);

Datasource interface:
com.ibm.db2.jcc.DB2SimpleDataSource dataSource = new com.ibm.db2.jcc.DB2SimpleDataSource();
dataSource.setDatabaseName ("DBNAME");
dataSource.setServerName (HOSTNAME);
dataSource.setDriverType (4);
dataSource.setPortNumber (50000);
dataSource.setSecurityMechanism(11); //11 is for kerberos
dataSource.setKerberosServerPrincipal("machinename.db2test.com@DB2TEST.COM"); //the name DB2 registers with KDC.
Connection con = dataSource.getConnection ();
```

c. **For connection using delegation**— In a multi-tier situation, a client may connect to a server, which in turn, needs to connect to a back-end server. In order to gain access to the end server, either the client will need to obtain credentials to access the end server then pass them on to the intermediate server, or the intermediate server will need to obtain the credentials to access the end server itself. Preferably, the credentials should be obtained using the client's authority. In this case, the JCC T4 driver uses a delegated credential from another principal. The user needs to set `securityMechanism` and `KerberosServerPrincipal` properties and use the delegated credential to set the GSSCredential in DB2Configuration.
Listing 5. Sample code for connecting using delegated credentials

**DriverManager interface:**
```java
java.util.Properties properties = new java.util.Properties();
properties.put("securityMechanism", new String("11")); // 11 is the integer value for kerberos
properties.put("KerberosServerPrincipal", KerberosServerPrincipalName);
DB2Configuration.setGSSCredential(delegatedCredential);
Class.forName("com.ibm.db2.jcc.DB2Driver");
String url = "jdbc:db2://machineName/DBName";
Connection con = DriverManager.getConnection(url, properties);
```

**Datasource interface:**
```java
com.ibm.db2.jcc.DB2SimpleDataSource dataSource = new com.ibm.db2.jcc.DB2SimpleDataSource();
dataSource.setDatabaseName (DBName);
dataSource.setServerName (ServerName);
dataSource.setDriverType (4);
dataSource.setPortNumber (50000);
dataSource.setSecurityMechanism(11); // 11 is for kerberos
dataSource.setKerberosServerPrincipal(KerberosServerPrincipalNam);
DB2Configuration.setGSSCredential(delegatedCredential);
Connection con = dataSource.getConnection (username, password);
```

Enabling CLPLUS to connect with Kerberos

1. Start the CLPPlus command window: **Windows > Run > -clplus.**
2. At the prompt, type **conn** and click **Enter.**
3. Give all the details at the prompt.
4. When prompted for a user, give the user in this format: **db2kuser@DB2TEST.com.**
5. When prompted for a hostname, give the host name in this format: `machineName.domainName`— in our case, it is `xxxmachine.db2test.com`.

If a SQL prompt is returned, the connection to the database was successful using Kerberos.

Setting up the IBM Data Server Driver for ODBC and CLI (CLI Driver), the .NET driver, and DB2 CLPlus

IBM Data Server Driver for ODBC and CLI setup

To use the CLI driver with Kerberos, there are changes needed to the db2cli.ini file.

In CLI, for the username, specify the fully qualified domain user or DB2 will go looking for the user in localSystem and fail with a "Local System Authority can't be located" error:

```sql
sqldriverconnect 1 0
"Authentication=KERBEROS;database=testKerb;hostname=9.xx.xx.xx;port=50000;uid=db2kuser@DB2TEST.COM;pwd=******;protocol=tcpip;" -3 235 SQL_DRIVER_NOPROMPT
```

Listing 6. sample db2cli.ini file

```ini
; Comment lines start with a semi-colon.
[tstcli1x]
uid=user1d
pwd=password
```
autocommit=0
TableType="'TABLE','VIEW','SYSTEM TABLE'"

[tstcli2x]
; Assuming dbalias2 is a database in DB2 for MVS.
SchemaList="'OWNER1','OWNER2',CURRENT SQLID"

[MyVeryLongDBALIASName]
dbalias=dbalias3
SysSchema=MYSCHEMA

[Common]
TargetPrincipal =HOST/machineName.domainName@DOMAINNAME
trace=1
tracefilename=traceFileName

IBM Data Server .NET driver setup
The Kerberos parameter in db2dsdriver.cfg has to be set for testconn40 to be successful.

Listing 7. Sample db2dsdriver.cfg file

```
<configuration>
    <dsncollection>
        <dsn alias="S" name="test1" host="9.xx.xx.xx" port="50000">
        </dsn>
    </dsncollection>
    <databases>
        <database name="test1" host="9.184.112.161" port="50000">
            <parameter name="UserID" value="db2kuser"/>
        </database>
    </databases>
    <parameters>
        <parameter name="Authentication" value="KERBEROS"/>
    </parameters>
</configuration>
```

Listing 8. sample testconn40 output

```
C:\ProgramData\IBM\DB2\DB2COPY1\cfg>testconn40 Database=test1

Step 1: Printing version info
.NET Framework version: 4.0.30319.33440
64-bit
DB2 .NET provider version: 9.7.4.4
DB2 .NET file version: 10.5.4.4
Capability bits: ALLDEFINED
Build: s140325
Factory for invariant name IBM.Data.DB2 verified
Elapsed: 0.2500056

Step 2: DB2DSDRIVER_CFG_PATH env var: unset
Validating db2dsdriver.cfg against db2dsdriver.xsd schema file
C:\ProgramData\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg against C:\SQLLIB\cfg\db2dsdriver.xsd
Elapsed: 0.0156191

Step 3: Connecting using "Database=test1"
Server type and version: DB2/NT64 10.05.0004
Elapsed: 5.4218871

Step 4: Selecting rows from SYSIBM.SYSTABLES to validate existence of packages
SELECT * FROM SYSIBM.SYSTABLES FETCH FIRST 5 rows only
Elapsed: 0.5625038
```
Step 5: Calling GetSchema for tables to validate existence of schema functions

Troubleshooting and tips

1. **Message**: "[JGSS_DBG_CRED] main Credentials have expired"
   This may occur because of using credentials that are no longer valid.
   **Resolution**: While using a username and password to authenticate and for a message like this with debug set to true on JAAS, try with `useDefaultCcache=false` in the jaas.conf file.

2. **Message**: "com.ibm.security.jgss.i18n.exception.RenewTGTIncompatible"
   This may occur while reusing tickets is enabled and the ticket is not reused for a while. There could be a possibility that you have set `reuseTGT=true` and JAAS is reusing the ticket, which would have now become obsolete.
   **Resolution**: Remove `reuseTGT=true` from the jaas.conf file.

   This is usually a configuration issue.
   **Resolution**: Use the information from the messages to compare with the Kerberos configuration and check that the SPN is as expected:
   - Use fully qualified host name of the system and DNS.
   - Beware of case sensitivity (DB2TEST.com is not the same as db2test.com).

4. **Message**: Error calling function Protocol status: 1312
   A specified logon session does not exist. It may have already been terminated.
   **Resolution**: Java tries to get the credentials cache (ccache) from Windows Local Security Authority (LSA) if you specify `useTicketCache=true` in the JAAS config file. Without the ccache, Java will try the Kerberos login itself. Provide the username and password in the application and try again.

5. **Message**: Caught javax.security.auth.login.LoginException while using JAASLogin. No LoginModules configured for JaasClient.
   This means that there is a problem in the jaas.conf file.
   **Resolution**:
   - Correct jaas.conf file contents.
   - Check for the position of the semicolon (;).
   - Use the JaasClient keyword in jaas.conf file. JaasClient is a case-sensitive keyword.
   - Update the java.security file to point to the correct location of the jaas.conf file.

6. **Slow connection using domain ID**: Experiencing a slow connection after logging in with the administrator or other ID.
   This is caused by a delay in Windows enumerating the groups to which the user ID belongs.
   **Resolution**: Run the following commands from the DB2 command line:
   1. `db2set -g DB2_GRP_LOOKUP=LOCAL`
   2. `db2stop`
   3. `db2start`
   Try the connection after this.

7. **Synchronizing of clocks**: Kerberos tickets are time-bound, so the KDC and the client machine where the application is running need to have their clocks synchronized.

8. **Password length, 0, is not allowed. ERRORCODE=-4461, SQLSTATE=42815 exception**
If the application sends an empty password to the JCC driver for Kerberos authentication, the JCC driver throws this exception: **Password length, 0, is not allowed. ERRORCODE=-4461, SQLSTATE=42815.** No user account can be created using an empty password. An empty password doesn't meet the standard password criteria, so the JCC driver doesn't allow the empty (""") — that is, zero-length password.

**Resolution:** The application can pass either a valid or null password to the JCC driver. If the client application doesn't have a valid password, we suggest you pass a 'NULL' in place of empty (""") string to circumvent this.

**Conclusion**

Kerberos is one of the leading industry standards for identity management and security. It's a reliable solution for many network security problems. This tutorial has shown how DB2 Connect drivers can be used with Kerberos to authenticate securely with the DB2 server.
Resources

Learn

- Read more about Kerberos security under the IBM Data Server Driver for JDBC and SQLJ. JDBC support for Kerberos security is available for IBM Data Server Driver for JDBC and SQLJ type 4 connectivity.
- Read more about Properties for the IBM Data Server Driver for JDBC and SQLJ. IBM Data Server Driver for JDBC and SQLJ properties define how the connection to a particular data source should be made.
- Read more in "Understand the DB2 UDB JDBC Universal Driver."
- Learn more about DB2 ODBC CLI driver download and installation information.
- Stay current with developer technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Follow developerWorks on Twitter.
- Watch developerWorks demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.

Get products and technologies

- Download initial Version 10.5 clients and drivers.

Discuss

- Get involved in the developerWorks Community. Connect with other developerWorks users while you explore developer-driven blogs, forums, groups, and wikis.
About the authors

Smitha Pambalath

Smitha Pambalath is a software engineer with IBM India Software Labs, working to improve the quality of the IBM Data Server Driver for JDBC and SQLJ on Windows, z/OS, and Linux platforms for DB2 and also for JDBC drivers for Informix.

Anuruddha Kumar Pandey

Anuruddha Kumar Pandey works for IBM India software Labs as a Software Engineer. He has been working on the DB2 Tools Continuing Engineering team. He has completed a Master Of Technology degree from Indian Institute of Information Technology and Management, Gwalior.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)