IBM Software Group

WebSphere Application Server - Service Integration Bus Messaging Engine Data Store Connectivity Problems and Solutions

Ty Shrake (tyshrake@us.ibm.com)
WebSphere Service Integration Bus Team
18 November 2010

WebSphere® Support Technical Exchange

Agenda

- What is a Data Store?
- Data Store Tables
- Term Definitions
- Basic Connection Process
- Data Store Exclusive Access
- Data Store Configurations
- Database Disconnections
- TCP KeepAlive
- Failure and Recovery Scenarios
- File Stores

What is a Data Store?

When using Service Integration Bus technologies there are 2 types of Message Stores: Data store and file stores. This presentation will focus on data stores. Data stores are databases, or schemas in a database. A data store is a set of tables used by a messaging engine to store persistent data, such as destination/queue/topic information and transaction information. It can also be used to store persistent messages. The data that is stored in a data store can be used for recovery operations. Each messaging engine has one and only one data store and that data store cannot be used by any other messaging engine. An overview of SIB data stores can be found here…

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/cjm0410_.html

Data Store Tables

Each data store has 7 tables. Two of these tables are used for locking purposes and the remaining 5 tables are used for actual data storage.

Table Name	Purpose
SIBOWNER	Ensures exclusive access to the data store by an active messaging engine.
SIBOWNERO	Used for locking the data store. This table stores no data in its one EMPTY_COLUMN column.
SIBCLASSMAP	Catalogs the different object types in the data store.
SIBLISTING	Catalogs the SIB*nnn* tables.
SIBXACTS	Maintains the status of active two-phase commit transactions.
SIBKEYS	Assigns unique identifiers to objects in the messaging engine.
SIB*nnn*, where *nnn* is a number	Contains persistent objects such as messages and subscription information. These tables hold both persistent and nonpersistent objects, using separate tables for the different types of data.

Term Definitions

- **Data Source**: A data source is a configured object that applications, such as messaging engines, can use to connect to databases. It is similar to a J2C connection factory that applications use to connect to messaging systems. Details about data sources can be found here…

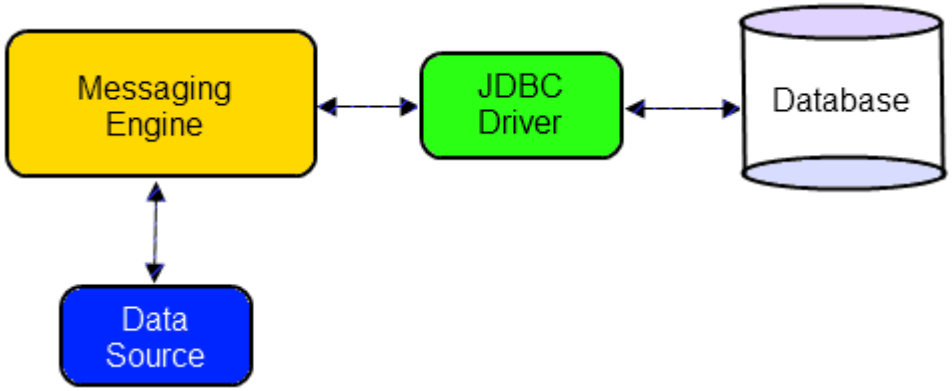
  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cdat_datasor.html

- **JDBC Driver**: A JDBC driver interacts with a database on behalf of a connected application. Messaging engines use JDBC drivers to make SQL calls to databases. More information can be found here…

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cdat_jdbcprov.html

Resource Relationship

This is the relationship between the messaging engine, the data source and the JDBC driver:



The Messaging Engine gets its connection and JDBC information from the Data Source and this is used to connect to the database through the JDBC driver. All connection threads to the database are through the driver, including the database locking thread.

Basic Connection Process

These are the basic steps that occur when a messaging engine connects to it's data store:

1) The messaging engine gets a network connection to the database.
2) We start the database locking sequence:

|------> GET Exclusive - LOCK ON SIBOWNERO TABLE
|
|----------> GET Exclusive - LOCK ON SIBOWNER TABLE
|
|--------------> UPDATE LOCK DATA IN SIBOWNER TABLE
|
|----------> RELEASE Exclusive - LOCK ON SIBOWNER TABLE
|
|----------> GET Shared - LOCK ON SIBOWNER TABLE
|
|------> RELEASE Exclusive - LOCK ON SIBOWNER0 TABLE

Connection Process Details

1. Get a connection to the database using a datasource defined in WSAS.

A> 2. On that connection execute the following SQL statement:
LOCK TABLE <SCHEMA>.SIBOWNERO IN EXCLUSIVE MODE

3. Get a second connection to the database using the same datasource.

B> 4. On the second connection execute the following SQL statement:
LOCK TABLE <SCHEMA>.SIBOWNER IN EXCLUSIVE MODE

5. On the second connection read the contents of the table:
SELECT ME_UUID,INC_UUID,VERSION,MIGRATION_VERSION
FROM <SCHEMA>.SIBOWNER

6. On the second connection update the contents of the table:
UPDATE <SCHEMA>.SIBOWNER SET INC_UUID=X,VERSION=X,
MIGRATION_VERSION=X WHERE ME_UUID=X

C> 7. Close the second connection, which commits the update and releases the exclusive table lock on <SCHEMA>.SIBOWNER.

Connection Process Details (cont'd)

8. Get a third connection to the database using the same datasource.

D> 9. On the third connection execute the following SQL statement:

```
LOCK TABLE <SCHEMA>.SIBOWNER IN SHARE MODE
```

10. On the third connection execute verify the contents of the table:

```
SELECT ME_UUID,INC_UUID,VERSION,MIGRATION_VERSION
FROM <SCHEMA>.SIBOWNER
```

E> 11. Close the first connection, which releases the exclusive table lock on <SCHEMA>.SIBOWNERO.

So the locks which the messaging engine owns at startup are:

```
Timeline:                A------  B-------  C-------  D-------  E------>

Table SIBOWNERO  [X-lock==================]
Table SIBOWNER             [X-lock]           [S-lock==========>
```

Connection Process Details (cont'd)

It's important to understand that SIB does NOT lock any other tables. We have seen customers observe that other locks can occur that appear to be related to the SIB locks, but these are usually database driven. They are not explicitly requested by SIB. If you see locks that are taken in addition to the SIBOWNER and SIBOWNERO tables they will be taken by the database itself or some other application using the database.

Data Store Exclusive Access

In versions 6.0 and 6.1 of WebSphere Application Server there were only 6 data store tables, not 7. APAR PK35226 changed the database locking model by adding an additional table (SIBOWNERO). This table was added to accommodate certain functionality for DB/2 on z/OS. Before this APAR the messaging engine would place an Exclusive lock on the SIBOWNER table. After this APAR and the addition of the SIBOWNERO table the messaging engine maintains a Shared lock on the SIBOWNER table. DB/2 on z/OS is able to make use of this shared lock to improve recovery scenarios. However, this change makes no difference whatsoever on other platforms (Unix®, Linux®, Windows®). The fact that this lock is now Shared does NOT mean that database tooling, such as runstats, will now be able to lock the table while the messaging engine still holds the lock. Database tools such as runstats should only be run when the messaging engine is shutdown and has no lock on the data store. Details about the APAR can be found here…

http://www-01.ibm.com/support/docview.wss?uid=swg1PK35226

Data Store Configurations

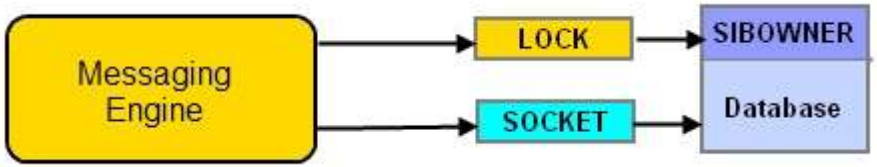
In general there are 2 basic ways to setup a data store topology: Locally or remotely.

Local data store configuration means that the data store exists on the same machine or node as a messaging cluster. Remote data store configuration means that the data store resides on a physically different server from the messaging cluster.

The preferred configuration is to configure the data store on a remote machine. This avoids a single point of failure and provides more flexibility.

Data Store Connections

When a messaging engine connects to its data store it first opens a socket connection to the database. It then goes through the locking sequence as described earlier (pages 7-9). Once the messaging engine has its data store lock it holds this lock for the lifetime of the messaging engine. Normally the lock and the socket on the database side will not be cleaned up until the messaging engine goes through a normal shut down.



Database Disconnections

What happens if the messaging engine unexpectedly loses its connection to the database?

If you are running WebSphere Application Server 6.0 or 6.1 the default behavior is for the messaging engine to start a new TCP connection to the database and attempt to regain its data store lock. It will try this for 15 minutes and, if unsuccessful, it will terminate. At this point the High Availability Manager will failover the messaging engine to another server in the same cluster and this new instance of the same messaging engine will again open a socket connection to the database and then attempt to get a lock on the data store.

Database Disconnections (cont'd)

Note: In WSAS 6 and 6.1 you can force the messaging engine to stop immediately after an unexpected disconnection from its data store rather than retry for 15 minutes by setting the following tuning parameter to 'true' in the sib.properties file:

sib.msgstore.jdbcFailoverOnDBConnectionLoss

This value is set to true by default on WebSphere Application Server version 7.0. When this tuning parameter is set to true the messaging engine will become disabled immediately after the disconnection from the database and the failover sequence will begin. You will also see a "Panic" message in the SystemOut.log. Details about this tuning parameter and why you might want to set it are documented under APAR PK72567, here…

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK72567

Database Disconnections (cont'd)

Once the messaging engine fails over to another server in the cluster it will resume its attempt to get a lock on the data store. If the attempt fails it will continue attempting for 15 minutes. If the messaging engine is unable to acquire the lock after 15 minutes it will become disabled.

In most cases the initial 15 minute period of attempts and the post failover 15 minute period of attempts will both fail. The reason for the failure is that when the disconnection originally occurred the messaging engine became aware of this fact but the database did not. So while the messaging engine is attempting to reacquire its data store lock on the SIBOWNER table the database is still holding both the lock and the socket from the previous connection before the unexpected disconnect occurred. **As long as this socket and data store lock remain in place the messaging engine will not be able to acquire a new lock and resume messaging functions.**

Database Disconnections (cont'd)

Solution:

The solution to this problem is actually fairly simple. The key is to somehow get the data store lock released so that the messaging can acquire a fresh lock in a timely manner. But the old (orphaned) data store lock will not be released by the database until the socket connection to the database is first cleaned up. Once the socket is cleaned up the lock will be released by the database and the messaging engine will be able to acquire a fresh lock and resume its messaging functions.

So how do we get the socket cleaned up?

TCP KeepAlive

TCP KeepAlive is the solution to this problem. KeepAlive is a feature of TCP that has 2 main benefits. It can detect that a connection to a peer is no longer valid and it can prevent a network connection from being terminated due to inactivity.

In WebSphere Application Server recovery scenarios KeepAlive can play an important role, In particular it's ability to detect invalid connections is useful during failover scenarios. Once an invalid connection is detected the socket for that connection will be cleaned up (destroyed) . This is vitally important in disconnection and failover situations where rapid recovery is important. KeepAlive will check for stale sockets at certain intervals.

KeepAlive has a variety of settings, but for the purposes of rapid recovery when database disconnections occur we are only interested in 1 parameter.

TCP KeepAlive

KeepAlive periodically checks for broken connections (stale or orphaned sockets). By default KeepAlive will check for stale connections if the connection has been idle (no packet traffic) for a period of no less than 2 hours. This means that if an unexpected disconnection occurs KeepAlive will, by default, wait for 2 hours before deciding to do anything with the stale socket. What is needed is an adjustment to this 2 hour period. If we can reduce the 2 hour wait to something more reasonable, such as 3 or 5 minutes, the socket will be cleaned up quickly, the old lock on the SIBOWNER table will be released by the database, and the messaging engine will be able to acquire a fresh lock and resume messaging.

TCP KeepAlive Parameters

Here are the relevant KeepAlive parameters on the major distributed operating systems:

AIX: **TCP_KEEPIDLE**

The default value is 14,400 half-seconds, which is 2 hours. This value can be changed using the following command:

no -o tcp_keepidle=600

In the example above the value is set to 600 half-seconds (35 minutes).

TCP KeepAlive Parameters (cont'd)

Linux: **tcp_keepalive_time**

The default value is 7200 seconds (2 hours). This value can be changed using the following command (procfs interface):

 # echo 600 > /proc/sys/net/ipv4/tcp_keepalive_time

The example above sets the value to 600 seconds (10 minutes).

Solaris: **TCP_KEEPALIVE_INTERVAL**

The default value is 7200000 milliseconds (2 hours). This value can be changed using the following command:

ndd -set /dev/tcp tcp_keepalive_interval 30000

TCP KeepAlive Parameters (cont'd)

The example above sets the value to 30,000 milliseconds (30 seconds).

Windows: **KeepAliveTime**

The default value is 7200000 milliseconds (2 hours). You can change this value in the Windows Registry here:

\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Tcpip\Parameters

Failure and Recovery Scenarios

It's important to remember that a messaging engine will try to reacquire its lock on its data store for a maximum of 15 minutes. Therefore you want to be sure the old, orphaned lock is released before that 15 minute period of retries completes. This can be done by setting the KeepAlive parameter to a value less than 15 minutes. Most people find that anywhere from 2 to 5 minutes works well for them. Once KeepAlive cleans up the socket the database will release the lock and the next retry from the messaging engine should succeed in getting a new lock on its data store.

Once the KeepAlive setting has been changed to a shorter time value data store locks on the SIBOWNER table will be released much faster. This will allow much faster failover recovery times. Here are some typical scenarios:

Failure and Recovery Scenarios

1. JVM shuts down normally. In this case the messaging engine will properly disconnect from its data store. The socket and data store lock will be properly cleaned up.

2. JVM terminates abnormally. In this case the messaging engine will unexpectedly lose its connection to the data store, leaving an orphaned lock on the data store. The messaging engine will failover to another server in the cluster and reconnect to its data store as long as the KeepAlive timeout is set to less than 15 minutes.

Failure and Recovery Scenarios

3. JVM completely hangs. If the JVM hangs the messaging engine will still have a lock on its data store. If the JVM is killed then you effectively create scenario number 2 above. If you do not manually kill the JVM then the Node Agent will detect that the JVM is hung and will automatically kill it and restart it for you. This will also create scenario 2. You will see messages in the Node Agent log indicating that the JVM is hung (unresponsive) and that it is being restarted. You will see ADML0064I or ADML0063W log entries. But as in scenario 2, if KeepAlive is set to a value less than 15 minutes the messaging engine should be able to acquire a fresh lock on its data store and resume messaging functions.

Failure and Recovery Scenarios

4. The JVM is not hung, but the messaging engine loses its connection to the data store (perhaps due to a network interruption). In this scenario the behavior depends on which version of WebSphere Application Server you are running. In version 6.1 the default behavior is for the messaging engine to run for another 15 minutes while it attempts to reacquire a lock on its data store. If it fails to acquire the lock it will failover to another server in the cluster and again attempt to acquire the lock. In version 7 the default behavior when an messaging engine loses its connection to the data store is for the JVM to be automatically killed (even though it isn't hung… see page 15 of this presentation) . The Node Agent will then restart the JVM. This will create scenario 2 above.

Failure and Recovery Scenarios

The lesson in all of this is that if your messaging engine loses its connection to its data store it will not be able to acquire a new lock on the data store in a timely manner unless the appropriate TCP KeepAlive parameter is set to a value less than 15 minutes.

There are at least 2 other ways to release the orphaned lock: Manually release the lock using database tools, although some databases do not support this, or simply restart the database. Restarting the database is usually not feasible for most people. Further, both of these require manual intervention. Using KeepAlive avoids the need for manual intervention and ensures a quick, fully automatic recovery.

File Stores

If you are using a file store you do not need to worry about using KeepAlive. If a messaging engine loses its connection to its file store it should be able to re-establish a new connection and new lock on the file store immediately. It is worth noting though that this only works properly with NFS 4, not NFS 3. If you are using NFS and want best results use NFS 4.

Summary

- What is a Data Store?
- Data Store Tables
- Term Definitions
- Basic Connection Process
- Data Store Exclusive Access
- Data Store Configurations
- Database Disconnections
- TCP KeepAlive
- Failure and Recovery Scenarios
- File Stores

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:
  http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:
  http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community:
  http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM® Education Assistant:
  http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:
  http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails:
  http://www.ibm.com/software/support/einfo.html

We Want to Hear From You!

Tell us about what you want to learn

Suggestions for future topics
Improvements and comments about our webcasts
We want to hear everything you have to say!

Please send your suggestions and comments to:
wsehelp@us.ibm.com

Questions and Answers