IBM Software Group

WebSphere Application Server Message Store Overview

Ty Shrake

WebSphere® Support Technical Exchange

ON DEMAND BUSINESS™

Agenda

- What is a Message Store?
- Two Types of Message Stores
- Common Message Store Configurations
- How Message Stores Work
- Message Store Configuration
- Tuning Message Stores
- Backing Up Message Stores
- Viewing Data in Message Stores
- Common Problems and Solutions
- Message Store Tools
- Important APARs

What is a Message Store?

Basically, a message store is a database. A message store enables a messaging engine to preserve operating information and to retain those objects that messaging engines need for recovery in the event of a failure.

A messaging engine preserves both volatile and durable data in its message store. Volatile data includes data about non-persistent messages. Durable data includes persistent messages, meta data about persistent messages, transactions and any objects owned by the messaging engines (such as queues and topics).

Two Types of Message Stores

There are 2 basic types of message stores: Data Stores and File Stores.

Data Stores: Originally introduced in WAS 6.0. Uses most standard databases (DB2, Oracle, etc...) for persistent message storage.

Advantages: Since this is a standard database, all of the usual DBA tools and SQL commands can be used to manage the tables/database instance. You can also utilize your existing resources more effectively. One technical advantage of using data store is that some J2EE application can share JDBC connections to benefit from one-phase commit optimization. Improved performance over using a file store.

Disadvantages: Creates more opportunity for problems because the SIB tables are exposed to so many tools and SQL commands. Running certain tools or commands at inopportune times can create major problems.

Two Types of Message Stores

File Stores: Introduced in WAS 6.1 and is now the default persistence layer. The message store stores messages in flat files rather than database tables. It's also a file based transactional system that is used to provide reliable and recoverable storage of message data in concert with 1 and 2 phase transactional applications in WAS.

Advantages: Simpler administration, except in the HA case where administration could be more complex (involving advanced files systems). Lower deployment costs.

Disadvantages: Configuring ME's to fail over to another application server requires File store's files to be available at the new application server.

Common Message Store Configurations

Local: All message store data is stored on the system local to the Messaging Engine using it. This is the least desirable configuration because it provides only limited support for High Availability (HA).

Remote/Shared: The message store is located on a node (machine) that is physically separate from all messaging engines. This is the most desirable configuration because it provides the best support for High Availability (HA).

How Message Stores Work

When a messaging engine is created it is assigned an unique value, the ME_UUID, and you will need to configure a message store for the messaging engine. When the message store is created the messaging engine ME_UUID is written into the message store. This value associates a specific messaging engine with a specific message store. They effectively become a matched set. Each messaging engine owns it own, unique message store. No other active (started) messaging engine can use that data store.

Data Stores: When an ME starts it gets a lock on its data store in order to prevent any other ME or thread from writing to the data store tables. Once the ME has a lock on its data store it uses a combination of SQL statements and internal interfaces to store messages in the data store tables.

How Message Stores Work

The data store tables:

SIBOWNER Ensure exclusive access to the data store by an active messaging engine. This table contains the ME_UUID value.
SIBCLASSMAP Catalogs the different object types in the data store
SIBLISTING Catalogs the SIBnnn tables
SIBXACTS Maintains the status of active two-phase commit transactions
SIBKEYS Assigns unique identifiers to objects in the messaging engine
SIBnnn, 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, according to the following convention:

SIB000 contains information about the structure of the data in the other two tables
SIB001 contains persistent objects
SIB002 contains nonpersistent objects that have been saved to the data store to reduce the messaging engine memory requirement

How Message Stores Work

Persistent messages received on queues or topics are written to the SIB001 table of the data store. Each message occupies a single row in the table. When using Assured Persistent messages the message is written to the data store BEFORE it becomes available to any applications for consumption from a queue or topic.

Information about the message, such as transaction info, the type of object the message is, etc… is written to other tables. Simply deleting a row will delete the message but it will not delete meta data about the message that exists in other tables. This will not necessarily cause any problems but it does illustrate the point that deleting a row from the SIB001 table does not completely clean up a message and its associated data. Deleting rows to delete messages is NEVER recommended.

How Message Stores Work

File Stores: File stores are flat files. File stores are created and configured during ME creation. In version 7 of WAS the file store is the default message store option.

There are 3 files used by File store:

The Log - is used to keep track of in flight transactions. Data is written to this log serially and the log file itself is circular. The default size is 100 MB.

The PermanentStore - is used to store ReliablePersistent and AssuredPersistent messages and queues. The default size is 200 MB.

The TemporaryStore - is used to store ExpressNonPersistent and ReliableNonPersistent messages. The default size is 200 MB.

How Message Stores Work

Data is first written to the Log file sequentially, that is, new records are added to the end of the file. When the end of the log file is reached, old records at the beginning of the log file are overwritten by new records and this process repeats. Subsequently, data is written to the Permanent store file and Temporary store file. The exception is extremely short-lived data, which is only written to the Log file.

The TemporaryStore is only used when a large number of nonpersistent messages accumulate in memory. If a sufficient number nonpersistent messages accumulate in memory the messaging engine spill dispatcher will temporarily write some of these messages to the TemporaryStore to free up memory resources.

How Message Stores Work

Checkpointing - Checkpointing is an internal process that compares data in the queues and topics with data in the data store tables or file store files. It ensures this data is internally consistent and removes old data that is no longer needed, which frees up space for more messages. A Checkpoint occurs every 5000 transactions (not every 5000 messages) and, in the case of a file store, when the Log is nearly full. During a file store Checkpoint the following happens:

Recovery data is written to the Log. This data is used to reconstitute the state of the Log at restart or recovery. The Log file is truncated to free up space used by completed transactions. This is analogous to moving the "tail" along to the oldest in flight transaction. Any long-lived message data in the "active" portion of the Log is moved into the PermanentStore file, thus freeing up more space in the Log.

Message Store Configuration

Data Store settings: Each data store has a number of configurable attributes. Perhaps the most important of these are the schema name and the "Create tables" option. When the data store tables are created they get a schema name. The name can be the default value (IBMWSSIB) or you can provide one of your own. This name can be used to identify exactly which database instance belongs to a specific messaging engine.

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/sibresources/SIBDatastore_DetailForm.html

Setting up the Data store for a messaging engine:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjk0001_.html

Data store HA considerations:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjk0001_.html

Message Store Configuration

File Store settings:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/sibresources/SIBFilestore_DetailForm.html

File Store High Availability Considerations:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjk0001_.html

Many customers use NFS for file store HA. Due to locking mechanism changes between NFS 3 and NFS 4, NFS 4 should always be used.

Message Store Configuration

In order for the ME to connect to a data store it requires 2 components which must be configured: The data source and the JDBC driver.

A data source is similar to a JCA connection factory because it allows applications to connect to a relational database. Details about data sources can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cdat_datasor.html

The data source is also where you configure the Component-managed Authentication Alias, which is the user ID and password required by the database. If these values are incorrectly specified an "insufficient privileges" exception will be thrown and printed in the SystemOut.log. Details about database privileges can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.web20fep.multiplatform.doc/ref/rjm0650_.html

Message Store Configuration

Details about data source configuration and settings can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/udat_jdbcdatsordet.html

Configuring a messaging engine to use a data source:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjk0001_.html

The data source is associated with a JDBC provider where you can configure which JDBC driver and helper class you will use. The JDBC driver provides an interface into the database and the helper class extends the functionality of the driver. It is critical that the latest driver version and correct helper class are selected. An incorrect configuration here may allow the ME to initially connect to the data store but will soon lead to erratic behavior and various exceptions. JDBC resource configuration is detailed here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tdat_ccrtprov.html

Message Store Configuration

Details about each JDBC driver can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rdat_minreq.html

Message Store Configuration

File stores - File stores are generally simpler than data stores. The only things that can be configured are the physical location of the file store (which cannot be changed without deleting and recreating the file store) and the sizes of the files themselves.

File store file sizes can be configured to be larger (but not smaller) than the default sizes. Details can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/sibresources/SIBFilestore_DetailForm.html

Tuning Message Stores

In general, most tuning is done to prevent production stoppages due to lack of file space. The tuning of a message store should always be done from the 'worst case scenario' perspective. An approximation of the maximum number of messages to be processed during heaviest loads, the size of each message and a contingency where messages accumulate on queues should all be taken into consideration. A general formula for the amount of space needed would be...

Maximum possible number of messages in the system (in a worst case scenario)

multiplied by

The size of the largest possible message

equals

The total amount of file space needed.

Tuning Message Stores

Data Stores: The number of data store tables can be increased if there is an apparent message-to-data store bottleneck in messaging performance. This is rarely done, and can only be done when the tables are initially created (you cannot add more later). Instructions and details can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjm0020_.html

Here is an example of what the tables might look like with this modification...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjm0240_.html

Tuning Message Stores

File Stores: File store tuning can be accomplished in 2 ways. One way is to modify the default file sizes due to heavy message loads, large message sizes, etc... The question we often hear is "What sizes should the files be?"

In most cases, the default file sizes are adequate. Using static analysis to determine the right file sizes is difficult. By default the minimum size of the Store files is 200Mb, 2x the size of The Log file default size. The size of these files should as a rule of thumb remain 2x the size of The Log file, and it is recommended that if The Log file size is increased, so are the minimum size values of the store files. Obviously if lots of messages will be sent and not consumed for a long period, this also needs to be taken into consideration when setting the sizes of the store files.

The best means to determine file size requirements is to run load tests on the applications in a as near real environment as possible for a significant duration. Testing should focus on the boundary cases of long running transactions, message size and the duration for which a message is on a queue and not consumed. If Log Full errors occur then the file sizes are too small. The right file sizes are probably the sizes that eliminate Log Full errors plus a percentage to cater for the extreme cases, for example, during failover.

Tuning Message Stores

Best practices include keeping transaction size to a minimum (i.e. using more transactions instead of less), using small messages and not letting messages build up on queues or topics.

A second way to tune the file store is to modify the sib.msgstore.cachedDataBufferSize, sib.msgstore.cachedDataItemMaximumSize and sib.msgstore.transactionSendLimit properties. These properties tune the memory buffers used to store messages. It is rare for customers to modify these properties and really should only be done if a specific problem is encountered that warrants changing them. Details about these properties can be found here...

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.express.doc/tasks/tjm0220_.html

Backing Up Message Stores

Data stores: Backing up messaging engine schemas (data stores) should be done using the standard database tools that come with the database. However, do NOT attempt to backup a data store while a messaging engine is running and using the data store. Attempting to do so can result in lost or corrupt data. **Always shutdown the messaging engine using the data store before backing it up.**

File stores: Backing up file stores should be done using the standard file system tools that come with your operating system. However, like data stores, do NOT attempt to backup a file store while a messaging engine is running and using the file store. Attempting to do so can result in lost or corrupt data. **Always shutdown the messaging engine using the file store before backing it up.**

Viewing Data in Message Stores

Data stores: Your DBA can use the standard database tools to view tables and rows in a data store.

Note: NEVER alter any data in any data store table for any reason. Altering data may cause serious problems and is an unsupported user action. If you alter any data and encounter erratic or unexpected behavior afterward the only solution is to drop the tables and recreate them.

Filestore: There is no specific tool that allows you to examine the data inside of a file store log file.

One item that should be mentioned here is the Message Store Dump. This is a dump of meta data in either a data store or a file store, but it does not dump message data itself. It is normally used to examine the state of messages on queues or topics.

Common Problems and Solutions

The following scenario is the most common data store problem scenario we see in SIB support:

When a messaging engine starts it makes a socket connection to its data store and then puts a lock on the SIBOWNER table of the data store. If the JVM a Messaging Engine is running in is terminated for any reason or there is a TCP interruption to the connection to the data store the ME will leave behind a lock on its data store. Termination of the JVM or a connection interruption (such as a Reset packet) leaves an orphaned socket behind at the database end of the connection. As long as this socket remains the lock will also remain. The database itself owns both the socket and the lock.

.

If the JVM is then restarted the ME will attempt to reconnect to its data store. However, because of the orphaned lock (socket) on the data store the ME will not be able to connect. The ME will attempt to connect for 15 minutes. If, after 15 minutes, it has not been able to connect then the data source will be disabled and no further attempts to connect will be made.

The solution to this problem is to ensure that the data store lock is cleared before the 15 minute retry period has expired. This can be done in 2 ways: One way is to simply restart the database (usually not a viable option).

continued…

Common Problems and Solutions

- The second way is to set the TCP KeepAlive idle timeout to a value less than 15 minutes. The default KeepAlive timeout value is 2 hours, which is usually much too long for most environments. KeepAlive will clean up the orphaned socket after the period specified in the timeout parameter. Once KeepAlive cleans up the orphaned socket then the database will release the lock and, if the ME is still within its 15 minute retry period, it will be able to reconnect to its data store. So the key to success is to ensure the KeepAlive timeout is set to a value less than 15 minutes (perhaps 2-5 minutes is a good setting, depending on how long you are willing to wait.) Details can be found about this here…

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/tasks/tjm_tunedbconn.html

Common Problems and Solutions

Other problems we see include:

- SQL exceptions thrown in the JDBC driver. This often occurs because of a mismatch between the JDBC driver being used and the HelperClass that is being used. Another common cause is the use of the Oracle version 9 driver, which is known to be problematic. The version 10 driver should be used instead.
- Another common cause of SQL exceptions in SystemOut logs is TCP interruptions that disconnect a messaging engine from its data store. If this happens the previous discussion on TCP KeepAlive would apply, which would allow you to recover from TCP interruptions more or less seamlessly. However, if these interruptions are frequent a deeper examination of the source of the interruptions should be conducted.

Common Problems and Solutions

- A messaging engine was deleted and recreated, but its message store was not and as a result when the new instance of the messaging engine starts you will see the following exception...

  CWSIS1535E: The messaging engine's unique id does not match that found in the data store. ME_UUID={0}, ME_UUID(DB)={1}

  The solution to this problem is to delete and recreate the message store.

- File store logs get full:

  Conceptually, when the "head" catches the "tail" (or more accurately when an attempt to write to the log would mean that the "head" catches the "tail") the log will be full. At this point, Check Pointing has been unable to free enough space in The Log for the new message or transaction. This can happen in a number of scenarios…

  continued…

Common Problems and Solutions

Use of large messages causing the PermanentStore and the Log to fill up.
Lots of work is done in a single transaction.
Transactions are long running.
Messages are put but not consumed.
High throughput.

When this happens the Log will simply timeout the oldest transaction and continue. This does solve problems that relate to long running transactions that are long running due to some other application error. However, when the system is under too much load, the Log Full problems will continue with log full and transaction timed out messages in the SystemOut.log. The solution in this case is either to increase the size of the Filestore files, or to modify the workload and or message characteristics.

Important APARs

PK75931: SERVICE INTEGRATION BUS REQUIRES DROP-ANY-TABLE AUTHORITY WHEN USING AN ORACLE DATABASE FILESTORE

This APAR introduces a new custom property:

sib.msgstore.jdbcUseDeleteInsteadOfTruncateAtStartup

that, when set to true, allows the messaging engine to use DELETE instead of TRUNCATE when it starts up. The use of TRUNCATE was a common complaint from our Oracle users and this APAR addresses that complaint. Details can be found here...

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK75931

Important APARs

PK72567: A MESSAGING ENGINE CONTINUES TO RUN WHILE THE DATABASE PROVIDING ITS MESSAGE STORE IS UNAVAILABLE WITH UNPREDICTABLE RESULTS.

This APAR introduces a new custom property:

sib.msgstore.jdbcFailoverOnDBConnectionLoss

that, when set to true, forces a server shutdown if the messaging engine loses its connection to its message store. This is done to avoid inconsistencies between the messages in queues and the data in the message store. Details can be found here...

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

Important APARs

PK74286: A MESSAGING ENGINE THAT IS USING DB2 9 FOR Z/OS FAILS TO START AND PRODUCES A CWSIS1530E ERROR.

This APAR corrects the following exception:

CWSIS1530E: The data type, 12, was found instead of the expected type, -1, for column, URI, in table, <Schema name>.SIBCLASSMAP.

Details can be found here...

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK74286

Important APARs

PK68367: MESSAGES MAY REMAIN IN LOCKED STATE AFTER DATABASE TRANSACTION LOGS BECOME FULL. A MESSAGING ENGINE RESTART IS REQUIRED.

When a system is under load and the transaction logs of the database used for the data store of the messaging engine become full, messages may become stuck in LOCKED status on queues.

The following error is seen in the JVM logs of the application server hosting the messaging engine, for each message which becomes stuck in LOCKED status:

CWSIP0002E: An internal messaging error occurred in com.ibm.ws.sib.processor.impl.store.items.
MessageItem.registerMessageEventListener, 1:1561:1.210.1.2, 10

This APAR resolves this problem. Details can be found here...

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK68367

Important APARs

PK35226: CHANGE THE DATABASE LOCKING MODEL TO IMPROVE FAILOVER TIMES

The database locking model has been changed so that an ME no longer holds a permanent exclusive lock when it is active. Instead it holds a shared lock. DB2 for z/OS can recover shared locks when an instance of DB2 fails in a data sharing group. Details can be found here...

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK35226

The APAR description at the link above suggests that this is unique to DB2 on z/OS, but the code change it discusses applies to all operating systems.

Important APARs

PK35116: SPECIFYING A SIB FILE STORE DIRECTORY PATH IN UNC FORMAT CAUSES THE FILE STORE TO BE CREATED IN THE WRONG PLACE.

The fix completes the UNC format support so that file stores are created correctly when this format is used to specify their location.

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK35116

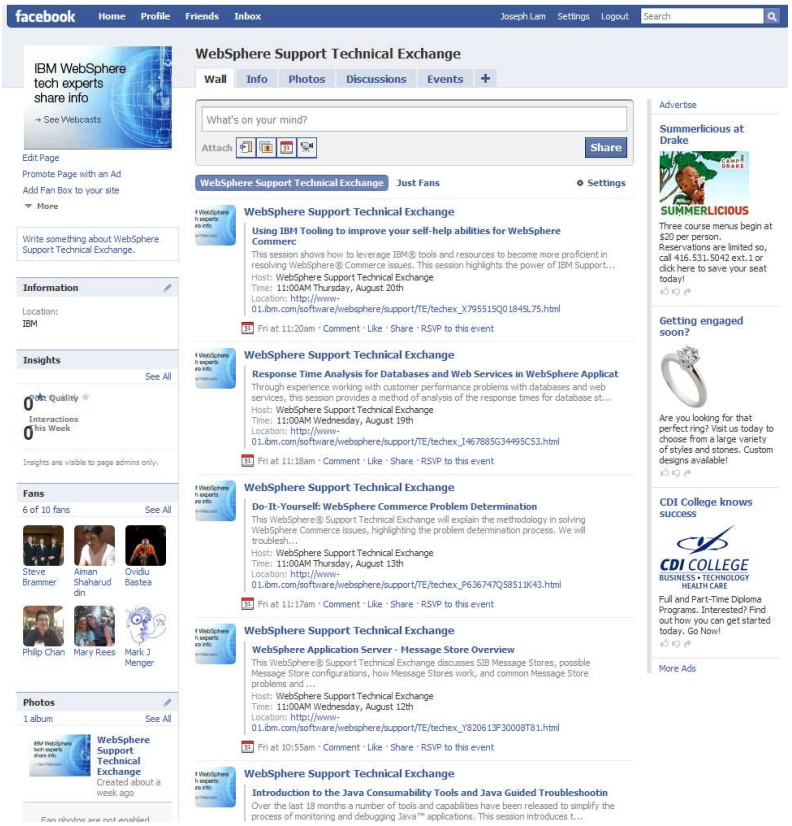
Summary

- What is a Message Store?
- Two Types of Message Stores
- Common Message Store Configurations
- How Message Stores Work
- Message Store Configuration
- Tuning Message Stores
- Backing Up Message Stores
- Viewing Data in Message Stores
- Common Problems and Solutions
- Message Store Tools
- Important APARs

Additional WebSphere Product Resources

- 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/
- Learn about other upcoming webcasts, conferences and events: http://www.ibm.com/software/websphere/events_1.html
- Join the Global WebSphere User Group Community: http://www.websphere.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

Join WebSphere Support Technical Exchange on Facebook!



- Stay up-to-update on upcoming webcast sessions
- Suggest future topics
- Suggest program improvements
- Network with other product users
- And More…

Become a fan now!

http://www.facebook.com/pages/WebSphere-Support-Technical-Exchange/121293581419

Questions and Answers