TIBCO EMS 概述

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2013-04-24 17:23:59)

|  |  |
| --- | --- |
| 标签：  [it](http://search.sina.com.cn/?c=blog&q=it&by=tag) |  |

**1**      **概述**

**1.1**  **JMS概述**

Java Message Service 1.1 (JMS)是用于应用程序间消息传输的Java框架规范。它是由SUN联合其他厂商如IBM，TIBCO等一起制定的，实现了统一的企业应用消息传输接口。

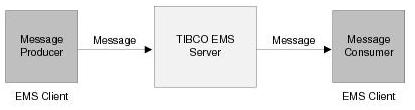
使用消息服务，我们可以更容易的进行企业级应用集成。例如，我们有几个系统，一个是客户管理系统，一个是产品管理系统，还有一个是原材料采购系统。这三个系统，每一个都很重要，但更重要是的，三个系统如何协作，毕竟这才是生产活动的基础。

面向消息的中间件(Message-oriented-middleware，MOM)使我们能够很轻松的将现存的或者新投产的系统集成起来。JMS框架不是指某一类中间件，更不是某一款MOM中间件产品，而仅仅是一个框架，一个规范，它的存在只是用于指导MOM系统的设计和开发。

TIBCO 的Enterprise Message Service是一款实现了JMS的MOM（类似的，IBM的MQ也是一款实现了JMS的MOM），EMS同时还可以与TIBCO的其他消息服务产品进行集成，比如TIBCO Rendezvous（RV，另外一款强大的消息传输中间件，是TIBCO的起家产品，国外运用很广，国内有上交所和新华社在用，为了达到性能最大化，使用的是UDP协议）。这里只介绍JMS的概念以及TIBCO EMS

**1.2**  **JMS消息模型**

JMS的基础是消息的创建和传输。消息一种应用程序间传输信息的结构化数据单元。JMS中，将消息的创建者称为producer，消息的接受者，称为consumer。TIBCO EMS服务器，充当消息传输的媒介，保证消息传输到正确的目的地。同时，TIBCO EMS也提供一些企业级应用的功能，例如fault-tolerance（故障切换，有的人叫失效备援），消息路由，与其他消息系统互联（如TIBCO Rendezvous和TIBCO SmartSockets），下图描述了消息的发送、传输和接收。

**

JMS支持三种消息模型：

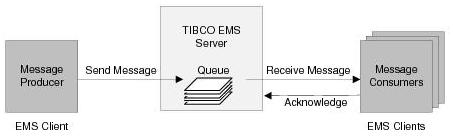
  点对点（队列）

  发布/订阅（主题）

  组播（主题）

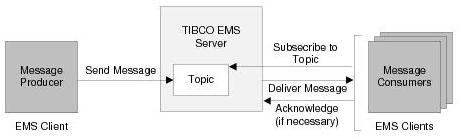
**1.2.1 点对点（队列）**

点对点传输消息传输模型中，每个消息只有一个producer和一个consumer。这种方式的传输，使用queue（队列）来存储消息。Producer将消息发送到queue，consumer从queue中将消息收走，然后向queue发一个acknowledgement（确认），表示消息已经收到了。多个Producer可以向同一个queue发消息，多个consumer也可以从同一个queue收消息。Queue可以被配置为exclusive（互斥或者排外）模式，这意味着，如果某一个consumer正在从某个队列收消息，则此时其他consumer不能从该队列收消息，只有等第一个consumer断开之后，后来的consumer才能从该队列收消息。Exclusive模式的队列有时比较有用，例如我们同一时间只允许一个接受者接收消息的场景下。如果队列设置为Non-Exclusive模式，那么任意一个consumer都能从该队列收消息。无论队列是Exclusive还是Non-Exclusive模式，某一个消息，只能被一个consumer收走。下图描述了一个non-exclusive模式队列的消息传输。每个consumer接收一个消息，然后向服务器发送一个acknowledgement，然后这个消息从队列里被删除，如此，这个消息就不会再被其他consumer收走了。

**

**1.2.2 发布/订阅（主题）**

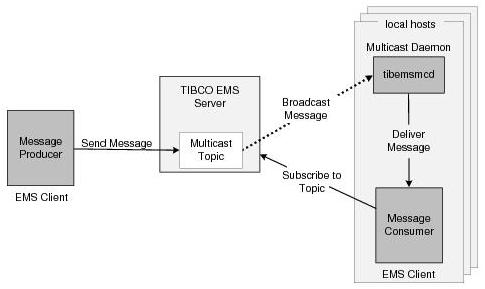
在发布/订阅消息模型中，producer被叫做publisher，consumer被叫做subscriber。多个publisher可向同一个topic发消息，publisher发送的某一个消息，可被多个subscriber收走。这种消息模型类似于广播，消息通过网络发布出去，感兴趣的接受者会把消息收走。下图描述了发布/订阅消息模型。

**

默认情况下，subscriber只有连到topic上的时候，才会接受消息。如果一个消息被publisher发送到topic上，而此时，没有subscriber与topic，则该消息会被自动丢弃。EMS服务器上可以创建durable subscriber（持久化订阅者），以保证即使subscriber没有连接到topic上，消息仍然会保存在topic里，等到subscriber连到topic之后，消息再被收走。durable subscriber与subscriber相比，可以认为每个durable subscriber在服务器上，有一块内存，用于暂存消息（当然，实际上并非如此）。

**1.2.3 多播**

多播消息模型允许一个producer将一个消息同时发个多个consumers。在发布/订阅消息模型中，topic会通过TCP协议给每一个subscriber发送一份消息的拷贝。而多播模型中，topic通过Pragmatic General Multicast (PGM)协议发送消息。subscriber所在的机器上，有一个daemon，该守护进程会从topic接收消息，再传输给subscriber。多播节省了带宽，但是不能确保把消息发送给所有subscriber。下图描述了多播传输模型的过程。

**

**1.3**  **EMS Destination特点**

什么是Destination？EMS中，服务器上的queue和topic都是destination，翻译为目的地，意思是要向哪个queue或者topic建立会话。EMS允许我们对每个destination的属性进行配置，以增强传输功能。

配置每个destination的访问权限。

限制每个destination可用来保存数据的存储空间。

对发往某个destination的流量进行控制。

不同服务器的队列之间配置路由（route）。

同一个服务器上的队列之间配置桥（bridge）。

将queue设置为Non-Exclusive或者Exclusive模式。

对进出某个destination的消息记录日志。

对每个destination的持久化存储进行设置。

配置queue是否允许consumer批量接收消息。

**1.4**  **Client APIs**

Java程序可以使用javax.jms来发收消息。这是JMS规范的标准接口，可以用来创建连接，设置消息类型，创建destination，收发消息等。由于EMS实现了JMS的标准规范，因此可以在www. java.sun.com/products/jms/index.html.查阅资料

EMS包含几个例子程序，这些例子演示了EMS的不同特性，例子代码位于安装目录中samples文件夹下，其中包含TIBCO Rendezvous连接TIBCO EMS服务器的例子。

**1.5**  **Administration**

EMS提供了一整套机制来管理服务器以及服务器中的对象，例如ConnectionFactories、Destinations等。管理工作可以通过两种方式进行，一种是通过命令行模式的管理控制台，另一种是调用管理API。

**1.6**  **安全**

为了保证服务器和客户端之间，服务器与服务器之间通信的安全，我们可以在服务器中配置SSL认证方式。SSL是一个在互联网或者局域网上传输加密数据的协议。SSL使用公钥和私钥来加密网络上传输的数据。EMS支持以下方式使用SSL：

  客户端与服务器之间

  管理端和服务器之间

  管理API和服务器之间

  存在路由的服务器之间

  主备服务器之间

**1.7**  **Fault Tolerance**

可以将EMS配置为主备模式，来实现故障切换。主机和备机成对的工作，主机接受客户端连接，负责处理消息，当主机发生故障，备机会接管主机的操作变成主机的角色。要使用EMS的Fault Tolerance特性，必须安装集群文件系统，这是一个限制。

**1.8**  **路由**

EMS提供了在服务器之间路由消息的功能。topic的路由可以跨越多跳（并且不能存在回路），queue的路由只能跨越一跳。EMS的路由存在单点故障，因此高可用方面要求较高的系统，不建议采用路由，此外ITBCO号称通过路由实现负载均衡，也是不可靠的，这方面的内容后面会论述。

**1.9**  **Transaction**

EMS提供的事务。类似于Oracle的事务，收发多个消息，必须进行提交。

**1**      **消息**

**1.1**  **EMS对JMS消息进行的扩展**

JMS规范定义的消息包含消息头，属性，消息体三部分。对消息的消息头和消息体格式JMS有详细的规定。而属性是用来存放用户定制的信息，以增强JMS的功能。EMS对JMS进行的一些扩展：

JMS规定了两种传输模式，持久化传输（PERSISTENT）和非持久化传输（NON\_PERSISTENT），EMS又增加了一种叫做可靠传输（RELIABLE\_DELIVERY）。

通常Consumer接收消息之后必须向服务器发送针对消息的确认。我们可以在创建会话时指定回话使用NO\_ACKNOWLEDGE模式，这样Consumer接收消息之后不需要向服务器发送针对消息的确认。此外，EMS还提供EXPLICIT\_CLIENT\_ACKNOWLEDGE 和EXPLICIT\_CLIENT\_DUPS\_OK\_ACKNOWLEDGE两种消息确认模式，后续章节会有描述。

EMS扩展了JMS中的[MapMessage](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "JMS Message Structure) 和[StreamMessage](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "JMS Message Structure)格式的消息体，以使EMS可以和TIBCO Rendezvous 和ActiveEnterprise进行通信。具体请参见相应章节。

**1.2**  **JMS消息结构**

JMS规定，消息包含三部分，分别是消息头、属性、消息体。其中消息头是必须的，属性和消息体是可选的。

**1.2.1 消息头**

消息头包含JMS预定义了10个项，用以消息的传输。

|  |  |
| --- | --- |
| JMSDestination | 消息目的地。 |
| JMSDeliveryMode | 传输模式（NON\_PERSISTENT， PERSISTENT等）。 |
| JMSExpiration | 消息生存时间。单位为毫秒。该值如果为0，则消息永不过期。客户端向服务器创建连接时，会与服务器进行时间同步，以此来保证消息的生存时间在客户端和服务器上是一致的。这里的时间同步，不是指二者的时间一致，二是二者的时间差保持恒定不变。生存时间涉及的问题较多，除非特别需要，否则不建议使用。 |
| JMSPriority | 消息优先级。范围是0-9，值越高，优先级越高。 |
| JMSMessageID | 消息提供者为每个消息赋予唯一的消息id。 |
| JMSTimestamp | 一条消息被创建时会有一个时间戳，之后消息被交给provider发送，消息最终发出的时间，可能会晚于该时间戳。 |
| JMSCorrelationID | 通过这个id可以从业务层面，将几条消息关联起来。该项是可选的。例如A系统之前曾经发送过一条消息给B系统，其ID为xxx，该消息被B系统处理之后，B系统将JMSCorrelationID 设置为xxx发回A系统的队列，此时，A系统可以通过筛选器指定只接受JMSCorrelationID为xxx的消息。 |
| JMSReplyTo | 消息的回复被发往的目的地。该域是可选的。 |
| JMSType | 消息类型标识。 |
| JMSRedelivered | 如果本域有值，则表示该消息因为某种原因，被多次接收。 |

**1.2.2 属性**

在属性部分，我们可以存在自己的信息。JMS规范规定了一些属性项，都是以JMS\_TIBCO开头。无论是JMS预定义的属性，还是我们自己定义的属性，所有的属性都是选填的，可以空着。

|  |  |
| --- | --- |
| JMS\_TIBCO\_CM\_PUBLISHER | RVCM sender的名字，该RVCM sender从TIBCO Rendezvous导入消息。 |
| JMS\_TIBCO\_CM\_SEQUENCE | RVCM消息的队列号。 |
| JMS\_TIBCO\_COMPRESS | 消息在发送时是否允许被压缩。 |
| JMS\_TIBCO\_DISABLE\_SENDER | 消息发送者名字是否出现在消息中。 |
| JMS\_TIBCO\_IMPORTED | 从Rendezvous or SmartSockets导入消息时，本域由服务器赋值。 |
| JMS\_TIBCO\_MSG\_EXT | 扩展了MapMessage和StreamMessage消息体的类型以包含自消息和数组。 |
| JMS\_TIBCO\_MSG\_TRACE | 消息从producer到consumer的过程中，是否被跟踪。 |
| JMS\_TIBCO\_PRESERVE\_UNDELIVERED | 如果消息必须被移除，该消息是否被放在未传输队列里。 |
| JMS\_TIBCO\_SENDER | 消息发送者名称。 |
| JMS\_TIBCO\_SS\_SENDER | 当ems服务器从TIBCO SmartSockets导入消息时，服务器把SmartSockets消息发送者消息头的值放在本域。 |

**Undelivered** **消息队列的说明**

如果某个消息的生存时间到了，或者该消息的重传次数超过了队列的[maxRedelivery](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "Destination Properties)设置，则该消息就要从队列里删除了。此时，服务器会检查消息的JMS\_TIBCO\_PRESERVE\_UNDELIVERED属性，如果属性被设置为true，则服务器把这个消息转移到Undelivered 消息队列$sys.undelivered中（这个队列是系统队列，不能被删除）。如果JMS\_TIBCO\_PRESERVE\_UNDELIVERED被设置为false，那么消息会直接从队列里删除。

使用undelivered队列，必须将JMS\_TIBCO\_PRESERVE\_UNDELIVERED设置为true。这个属性对某一个消息进行设置，不能对某个队列进行设置。我们可以创建一个consumer专门接收undelivered队列的消息。如果想删除undelivered队列的消息，在控制台中执行purge queue命令或者调用administrative API即可。undelivered队列不能创建路由。

**Message Sender**

当客户端创建一个到服务器的连接时，客户端必须使用一个用户名。队列中sender\_name 和sender\_name\_enforced属性决定消息中[JMS\_TIBCO\_SENDER](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "JMS Message Structure)属性是否必须存放用户名。当队列的sender\_name属性为true，而ender\_name\_enforced为false，则producers可以不提供用户名。Producers可以将[JMS\_TIBCO\_DISABLE\_SENDER](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "JMS Message Structure)设置为true，以是消息中不包含用户名，但是如果队列的sender\_name\_enforced为true，则服务器会忽略[JMS\_TIBCO\_DISABLE\_SENDER](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "JMS Message Structure)，用户名必须出现在属性中。

**1.2.3 消息体**

JMS的消息，可以有消息体，也可以没有。消息体有下面几种格式：

|  |  |
| --- | --- |
| Message | Message格式的消息，没有消息体。 |
| TextMessage | 字符串格式的消息体 |
| MapMessage | A set of name/value pairs. Name是字符串对象，value是java的内置类型。 |
| BytesMessage | 字节类型消息体。可以用传输二进制数据。 |
| StreamMessage | 流类型消息体。 |
| ObjectMessage | 可序列化的对象消息体。 |

**消息体的最大长度**

EMS最大支持512M的消息体。不过最好别这么干。

## 1.1   消息优先级

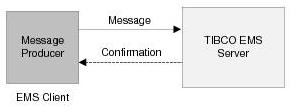
JMS规范在消息头中定义了个优先级的域，消息发送者可以以此来设置消息的优先级，优先级在0-9中取值。EMS支持消息优先级（这个域可选），其他厂商的中间件没有都实现。当consumer有多个消息需要接收时，先接收优先级高的消息。

## 1.2   消息传输模式

消息头中的[JMSDeliveryMode](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "JMS Message Structure)域指定了消息传输模式。对主题和队列这两种destination，JMS支持PERSISTENT和NON\_PERSISTENT两种消息传输模式。EMS扩展出第三种传输模式，[RELIABLE\_DELIVERY](file:///E:\write\%E6%B6%88%E6%81%AF.docx)模式。在创建producer的时候，可以指定该producer发送消息使用的默认的传输模式，也可以在每个消息的消息头中指定该消息通过哪种传输模式发送，此时将忽略producer默认的传输模式。

## 1.2.1 PERSISTENT模式

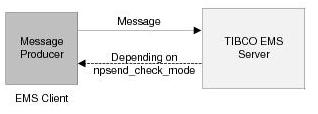
当producer发送一个PERSISTENT消息时，producer必须等待服务器返回一个确认。消息被持久化到服务器的磁盘上。这种传输模式保证消息能成功的保存到服务器上。然而这种方式也是有代价的，首先消息producer到server，要经过一个双向的网络传输，其次，server要先将消息写入磁盘，才向producer返回确认，这导致每秒传输的速率降低。

**

## 1.2.2  NON\_PERSISTENT模式

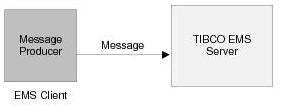
当producer发送一个NON\_PERSISTENT消息时，没有将消息写入磁盘的过程，这样可以提高性能。

如果主配置文件中的[authorization](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "tibemsd.conf)参数设置为disable，服务器不会给producer返回确认；如果该参数设置为enable，默认情况下，producer会像发送PERSISTENT消息时一样等待确认，此时，可以通过npsend\_check\_mode来制定服务器是否给producer发送确认。

**

## 1.2.3 RELIABLE\_DELIVERY模式

EMS扩展了一种RELIABLE\_DELIVERY模式，此时，无论[authorization](file:///E:\write\%E6%B6%88%E6%81%AF.docx)是否enable，服务器都返回确认，以此提高性能。但这种模式有个缺点，如果[authorization](file:///E:\write\%E6%B6%88%E6%81%AF.docx)被设置为enable，而producer向某个destination发送消息失败（也许是因为destination写错了，些许访问destination的权限不足，或者其他原因），但由于producer不会收到任何返回，所以producer总是认为发送成功了。所以除非特别需要，一般不建议使用这种方式。

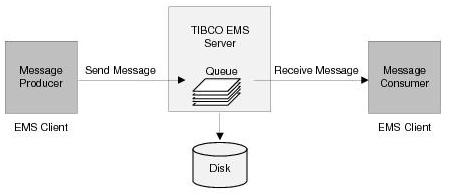
**

## 1.1   EMS如何持久化消息

NON\_PERSISTENT和RELIABLE\_DELIVERY 两种传输模式的消息是不会持久化到磁盘的，只有PERSISTENT模式消息才会持久化到磁盘。

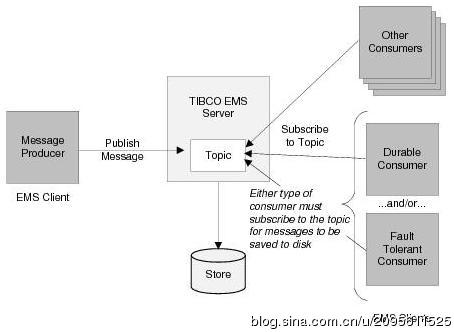
## 1.1.1 队列的持久化

发送到队列的持久化消息，总是被写入磁盘。consumer接收到消息之前，一旦服务器挂掉，不用担心，待服务器重启之后，consumer重新建立连接，仍然能收到消息。

**

## 1.1.2 主题的持久化

发送到主题的持久化消息，只有当主题拥有durable subscriber或者某个subscriber有fault-tolerant方式的连接时，消息才会被持久化到磁盘，否则，无论消息本身是否持久化的，服务器都不会将消息写入磁盘。这种方式与JMS规范是一致的。一个没有fault-tolerant连接的非持久化订阅者，在服务器挂掉之后重新连到topic上，它被认为是一个新的订阅者，因此它不会受到服务器挂掉之前的消息。

**

## 1.1.3 持久化消息和同步文件存储

当进行持久化时，消息是被异步写入磁盘的。其过程是producer向server发送持久化消息，server并不等待写入磁盘的动作结束，就向producer返回确认。这样是存在问题的，例如，producer收到了确认，而此时某个消息正在被写入硬盘，就在这时，服务器挂掉了，这就造成了producer认为消息发送成功，而server实际上没有保存数据的不一致状态。我们可以强制指定服务器使用同步方式来持久化数据，即只有在数据写入硬盘之后，再向producer返回确认。具体如何使用同步方式，后续会有描述。

## 1.1   多种存储方式

发送到服务器上的消息，被存储到磁盘里，EMS支持三种store方式，分别是文件、数据库和mstores方式。默认情况下，EMS使用文件存储消息，创建EMS服务器时，系统有三个默认的用以存储消息的持久化文件。我们可以创建自己的store（EMS中我把store称为持久化方式），并且可以将这个store与一个数据文件关联，这个数据文件可以放到我们想放的任何位置。关于store的配置信息，都保存在[stores.conf](file:///E:\write\%E6%B6%88%E6%81%AF.docx)配置文件中。在创建队列是，我们可以为每个队列都指定一个store。每种store都有自己的属性，例如，文件store具有下面三个属性，其他的属性没有列出来，后面会有描述。

 预分配磁盘空间

 定时truncate磁盘空间

 持久化是同步的还是异步的

在实际的部署过程中，我们可以按不同需求，为不同的队列指定不同的store，例如，为A队列指定一个叫做SA的store，这个store采用同步持久化；为B队列指定一个叫做SB的store，这个store采用异步持久化；为C队列指定一个叫做SC的store，这个store采用数据库store。这三个store都位于一个EMS服务器中，对用户是透明的。

## 1.1.1 文件store

EMS可以使用文件保存消息，我们可以使用服务器默认的三个文件，也可以创建自己的文件。EMS服务器会直接将需要持久化的消息，保存到与队列关联的store所对应的的文件中去。

## 1.1.2 mstores

mstores这种store，比较特殊，它的设计，是为了应对发生故障切换时，备机能够立刻启动而准备的。但是这种方式虽然可以是的故障切换可以立即完成，但是在正常使用时，它的性能是非常差的，其传输消息的速度几乎只有文件store的百分之一，因此一般不建议使用。后面会有专门内容描述mstores的原理，这里不再赘述。

## 1.1.3 数据库store

EMS也可以将消息持久化到数据库中。但是这种方式的store性能也非常滴，仅仅比mstores好一点，因此实际生产中也不建议使用。后面会有专门的内容描述如何配置数据库store。

## 1.1.4 默认的store

EMS定义了三个默认的store：

 $sys.nonfailsafe，异步持久化方式的store

 $sys.failsafe，同步持久化方式的store

 $sys.meta，EMS将durable subscribers、fault-tolerant connections、以及其他的一些元数据的信息，写入到这个store中

这三个store以及他们关联的数据文件，都是在服务器启动时自动创建的，不需要任何配置的步骤。如果我们将这三个store对应文件删掉了，服务器在下次启动时，还是会自动创建。我们可以修改这三个store的属性已经更改三个store关联的数据文件的位置。

## 1.1.5 配置store

下面介绍配置文件store和mstores的步骤，数据库store后面会有专门的介绍。

 打开[stores.conf](file:///E:\write\%E6%B6%88%E6%81%AF.docx)文件，每个store都有唯一的名字。

 修改参数。文件store有type和file两个参数。type指定store的方式必，file参数指定与该store关联的数据文件。另外，store还有些可选的参数，例如消息是同步持久化还是异步持久化，文件的最小大小，EMS服务器是否定期的truncate数据文件以释放空间。Mstores同样也有两个参数type和file。可选的参数包括scan interval等。

 为每个队列指定关联的store，队列的store参数，可以在[topics.conf](file:///E:\write\%E6%B6%88%E6%81%AF.docx) 和 [queues.conf](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "Using Other Configuration Files)文件中设置，多个destination可以关联到同一个store中。

 也可以在控制台，通过命令行的方式修改文件store的属性。

## 1.1.6 针对mstores的特别说明

如果某个队列的store属性被关联到了一个mstores，那么它是不能在控制台通过命令行的方式来修改的。必须先停止服务器，然后把mstores关联的数据文件删掉，再到topics.conf 和queues.conf文件中，修改队列的store属性，最后重启服务器。

Mstores的方式，其设计初衷，是为了在发生故障切换时，使备机能够立刻启动，而不用像普通的store那样，先把持久化到磁盘的数据恢复到备机，然后再使备机处于启动状态。

为了实现这种性能，EMS服务器必须持续的对已经保存的消息进行镜像，EMS服务器采用渐进式的方式进行数据镜像，对于已经被接受走的消息、或者已经过期的消息、或者被从控制台purge的消息，也采用渐进的方式进行删除。

采用渐进式的方式检查和删除数据，为了防止影响服务器的性能。每次更新数据的数据量，受两个参数的控制，这两个参数都可以在stores.conf中进行设置。

默认的参数已经是经过优化的了。然后如果保存在磁盘中的数据量如果很大，这时每次镜像读写数据就会影响服务器的性能了。为了减缓这种影响，可以适当镜像读写的间隔时间。

[scan\_target\_interval](file:///E:\write\%E6%B6%88%E6%81%AF.docx)参数，这个参数表示所有的数据被镜像读写一次，所允许的最大间隔时间，也就是说，要求服务器，必须在这个总的时间内，把所有的数据都处理一遍。例如，如果scan\_target\_interval设置为24小时，服务器会在最多24小时的时间内，处理服务器中的每一条消息。由于被purge的或者已经过期的消息，只有被镜像一遍之后，才会真正的被purge掉或者被当做真正过期的消息，这意味，即使某个时刻我在命令行中运行了purge命令，但有可能最多要过24小时的时间，这种消息才会被从服务器中删除掉。

[scan\_iter\_interval](file:///E:\write\%E6%B6%88%E6%81%AF.docx)参数，每次渐进式检查所间隔的时间。比如将[scan\_iter\_interval](file:///E:\write\%E6%B6%88%E6%81%AF.docx)设定为10秒，那么每隔10秒钟，EMS服务器就要执行以此渐进式检查。每次检查的数据量，取决于总的数据量，同时也与scan\_target\_interval与[scan\_iter\_interval](file:///E:\\write\\%E6%B6%88%E6%81%AF.docx" \o "Using Other Configuration Files)的比值有关。总的原则是服务器必须在scan\_target\_interval设定的时间内，检查完所有的数据。

比如，scan\_iter\_interval设定为10秒，scan\_target\_interval 设定为 1 天，也就是86,400 秒，同时数据总量为9G。那么每间隔十秒，EMS服务器要一次性检查1M左右的数据，这样比每秒检查100k的数据，对性能影响小些。如果即便如此，数据库的性能还是降低了，我们可以调大scan\_target\_interval。

渐进式的镜像和清理数据会影响一些关键数据统计的准确性。在某一个完整的扫描检查完成之前，某些统计数据不一定是真实的，因为purge的过期消息并有真的从服务器中删除。例如，运行info命令，会报告Pending Messages" 和 "Pending Message Size"，但此时的显示数据是不准的，因为此时的统计，只包含在运行info命令之前所有扫描到的数据，并不是所有的数据。同样的，show store命令会显示"Message Count" 和"Message Size"的数据，而这个数据，也许是要比实际的保存在store中的消息量。直到扫描完整的执行，上面的统计数量才是准确的，此时执行show store命令时，返回的结果中有一项"First scan finished"，当这一项的值为true是，所得的统计数据才是准确的。如果我们想尽快的得到准确的统计数据，可以将scan\_target\_interval.参数的值降低，而这又会影响性能。矛盾吧。

总之，mstores的方式，本人不建议使用。