Business Usecases JMSME

Piotr Anders, Paulina Kania, Łukasz Osipiuk, Paweł Ostrowski

June 15, 2003

1 Introduction

1.1 Goal

1.2 Scope

2 Glossary

Once-and-only-once delivery - message is for sure delivered exactly once

At-most-once delivery - message is either delivered once or is not delivered

Application - JAVA program using JMSME client-side

Queue - message queue in JMS system

Topic - message topic in JMS system

User - Can be identified with programmer or just an application using JMSME api

Client-side - JMS implementation classes running on J2ME device

Server-side - JMS proxy running on J2EE middleware machine

JMS provider - third party JMS implementation

JMS - Java Message Service

JMS Session - single connection session in which messages can be sent or received to/from different Queues/Topics. Can be transacted.

2.1 Description of the rest of the Document

The rest of the document contains specifications of the following Use-Cases:

- Message sending
- Message receiving
- Message conveying from client-side to JMS provider
- Message conveying from JMS provider to client-side

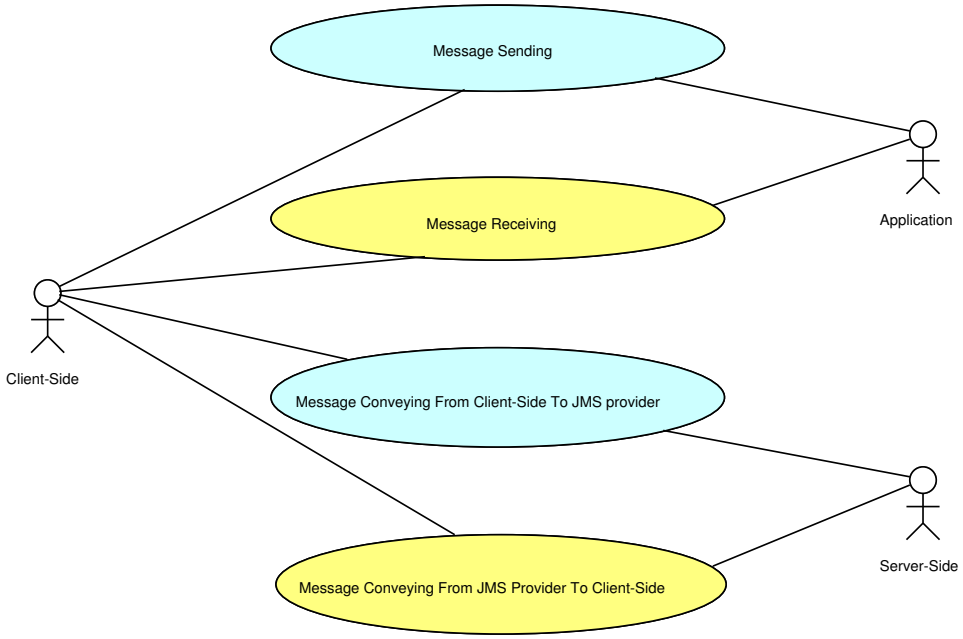

Figure 1: Use Case associations overview

3 Message sending

3.1 Brief Description

This Use-Case describes an action of sending JMS message on client-side by user using JMSME.

3.2 Goals

Messages sent as persistent must apply to once-and-only-once delivery scheme. Messages sent as non-persistent must apply to at-most-once delivery scheme.

3.3 Metrics

3.4 Entry Conditions

- Application uses JMSME client-side.
- Client-side has network connection to server-side. The connection may be unstable and not durable.

3.5 Workflow

3.5.1 Basic Workflow Steps

POINT-TO-POINT MODEL:

1. User chooses Queue name.
2. User creates JMS Session choosing whether session is transacted [1] or not.
3. User creates environment required to establish JMSME queue sending connection.
4. User establishes a connection with selected Queue. [2]
5. System checks if selected Queue name exists in the system and if not, an exception is thrown.
6. Repeatably:
   - User creates Message.
   - User chooses whether the Message is persistent or non-persistent.
   - Optionaly User chooses whether a message is sent by client-side synchronously[3] or asynchronously[4].

[1] In transacted sessions all sent/received but uncommited messages are considered as a whole. And as such they can only be either commited or rollbacked together. If such session breaks/closes, all uncommited messages are withdrawn.

[2] Jeszcze nie jesteśmy zdecydowani czy tu będzie wymagane połączenie - być może ustalimy że sprawdzenie poprawności i wogole połączenie będzie przesunięte do chwili gdy będzie to niezbędne (np w przypadku subskrybcji topica lub odbioru wiadomości z topica/kolejki
PS. podobny przypis powinien pojawić się jeszcze w innych miejscach

[3] Connection to JMS Provider must be established. Message is never enqueued. **Jak się uda to będzie ;) Na razie nie wiemy jak to pogodzić z interfejsem Message Producera.**

[4] Message is enqueued in client-side and is sent when connection to JMS Provider is established.

- Optionaly User chooses wait time after which sending in synchronous mode is abandoned.
- User sends the Message.
- The client-side takes over sending of the Message.
- User application continues execution.

7. User can any time check how many messages are pending for sending.

If the Message is *PERSISTENT*:

8. User can at any time check whether the Message has reached JMS provider.
9. The client-side enqueues the Message.
10. If network connection is available the client-side sends the Message to server-side and waits for acknowledgment from server-side.
11. If network connection is not available the client-side may (User choice):
    (a) try to reconnect in every fixed period of time
    (b) try to reconnect when the next message to send appears
    (c) try to reconnect when application starts or before application is closed (this can be timeouted).
    (d) try to reconnect od users demand.
12. User closes JMS Session.

If the Message is *NON-PERSISTENT*:

8. User can at any time check whether the Message has left the client-side, but he can not be entirely sure if the Message has reached JMS Provider or not.
9. If network connection is available the client-side sends the Message to server-side.
10. If network connection is not available the client-side enqueues the Message and then may (User choice):
    (a) try to reconnect in every fixed time period
    (b) try to reconnect when the next message to send appears
    (c) try to reconnect when application starts or before application is closed (this can be timeouted).
    (d) try to reconnect od users demand.
11. User closes JMS Session.

PUBLISH/SUBSCRIBE MODEL:

1. User chooses Topic name.
2. User creates JMS Session choosing whether session is transacted or not.
3. User creates environment required to establish JMSME topic publishing connection.
4. User establishes a connection with selected topic.
5. System checks if selected topic name exist in the system and if not an exception is thrown.
6. Repeatably:
   - User creates Message.
   - User chooses whether the Message is persistent or non-persistent.
   - Optionaly User chooses whether a message is published by client-side synchronously[3] or asynchronously[4]
   - Optionaly User chooses wait time after which sending in synchronous mode is abandoned.
   - User publishes the Message.
   - The client-side takes over publishing of the Message.
   - User application continues execution.
7. User can any time check how many messages are pending for sending.

If the Message is *PERSISTENT*:

8. User can at any time check whether the Message has reached JMS provider.
9. The client-side enqueue the Message.
10. If network connection is available the client-side publishes the Message and waits for acknowledgment from server-side.
11. If network connection is not available the client-side may (User choice):
    (a) try to reconnect in every fixed time period
    (b) try to reconnect when the next message to publish appears
    (c) try to reconnect when application starts or before application is closed (this can be timeouted).
    (d) try to reconnect od users demand.
12. User closes JMS Session.

If the Message is *NON-PERSISTENT*:

8. User can at any time check whether the Message has left the client-side, but he can not be entirely sure if the Message has reached JMS Provider.

9. If network connection is available the client-side publishes the Message.

10. If network connection is not available the client-side enqueues the Message and then may (User choice):

    (a) try to reconnect in every fixed period of time

    (b) try to reconnect when the next message to publish appears

    (c) try to reconnect when application starts or before application is closed (this can be timeouted).

    (d) try to reconnect od users demand.

11. User closes JMS Session.

3.5.2 Alternative Workflow Steps

Alternative workflow bases on three additional possibilities. JMSMEuser can:

1. Cancel sending/publishing of the Message before it leaves client-side in asynchronous[4] mode.

2. Commit or rollback transacted JMS Session, thus confirm or cancel all uncommited messages.

3.6 Category

This Use-Case is of the core category.

3.7 Risk

No risk.

3.8 Special Requirements

Server-side must be correctly configured to access JMS provider.

4 Message receiving

4.1 Brief Description

This Use-Case describes an action of receiving JMS message on client-side by user using JMSME.

4.2 Goals

Messages sent as persistent must apply to once-and-only-once delivery scheme. Messages sent as non-persistent must apply to at-most-once delivery scheme.

4.3 Metrics

4.4 Entry Conditions

- Application uses JMSME client-side.
- Client-side has network connection to server-side. The connection may be unstable and not durable.

4.5 Workflow

4.5.1 Basic Workflow Steps

POINT-TO-POINT MODEL:

1. User chooses Queue name.
2. User creates JMS Session choosing whether session is transacted or not.
3. User creates environment required to establish JMSME queue receiving connection.
4. User establishes a connection with selected Queue.
5. System checks if selected Queue name exists in the system and if not an exception is thrown.
6. Repeatably:
   - User sends request to the client-side.
   - The client-side gets the message from the server-side (if one is available and if there is network connection)
   - User receives the message and continues execution.
7. User closes JMS Session.

PUBLISH/SUBSCRIBE MODEL:

1. User chooses Topic name.
2. User creates JMS Session choosing whether session is transacted or not.
3. User creates environment required to establish JMSME topic receiveing connection.
4. User establishes a connection with selected Topic and subscribes it. (user must be subscribed to a topic to get messages from it)
5. System checks if selected Topic name exists in the system and if not an exception is thrown.
6. Repeatably:
   - User sends request to the client-side.
   - The client-side gets the message from the server-side (if one is available and if there is network connection)
   - User receives the message and continues execution.
7. User closes JMS Session.

4.5.2 Alternative Workflow Steps

Alternative workflow bases on a few additional possibilities.

1. User can try to create durable subscription [5] to the Topic.
2. User can an any moment commit or rollback transacted JMS Session, thus confirm or cancel all uncommited messages.

4.6 Category

This Use-Case is of the core category.

4.7 Risk

No risk.

4.8 Special Requirements

Server-side must be correctly configured to access JMS provider.

[5] With durable subscription all messages published to the Topic are retained until they are acknowledged by this durable subscriber or they have expired.

5 Message conveying from client-side to JMS provider

5.1 Brief Description

This Use-Case describes an action of conveying JMS message from client-side to JMS provider by JMSMEserver-side.

5.2 Goals

Messages sent as persistent must apply to once-and-only-once delivery scheme. Messages sent as non-persistent must apply to at-most-once delivery scheme.

5.3 Metrics

5.4 Entry Conditions

- Application uses JMSME client-side.
- Client-side has network connection to server-side. The connection may be unstable and not durable.
- Server-side has network connection to JMS provider.
- Application works with open JMS Session

5.5 Workflow

5.5.1 Basic Workflow Steps

POINT-TO-POINT MODEL:

1. The server-side receives the Message from a client-side.
2. The server-side enqueues the Message.
3. The server-side creates environment required to establish JMS queue sending connection (if required. The server-side can already be connected).

If network connection is available:

4. The server-side sends the Message to JMS provider.
5. **If the Message is *PERSISTENT*:** The server-side waits for acknowledgment from JMS provider.
6. The server-side dequeues the Message.

7. **If the Message is *PERSISTENT*:** The server-side sends acknowledgment to the client-side.

If network connection is not available:

4. The server-side returns an error message to the client-side

PUBLISH/SUBSCRIBE MODEL:

If the Message is *PERSISTENT*:

1. The server-side receives the Message from a client-side.
2. The server-side enqueues the Message.
3. The server-side creates environment required to establish JMS topic publishing connection (if required. The server-side can already be connected).

If network connection is available:

4. The server-side publishes the Message.
5. **If the Message is *PERSISTENT*:** The server-side waits for acknowledgment from JMS provider.
6. The server-side dequeues the Message.
7. **If the Message is *PERSISTENT*:** The server-side sends acknowledgment to the client-side.

If network connection is not available:

4. The server-side returns an error message to the client-side

5.5.2 Alternative Workflow Steps

Alternative workflow bases on a few additional possibilities.

1. The server-side crashes.
   (a) The server-side is restarted.
   (b) Persistent messages are kept on a client-side (they were not acknowledged)
   (c) The Client-side resends messages.
   (d) The server-side sends/publishes all the messages which are not acknowledged by JMS provider.

5.6 Category

This Use-Case is of the core category.

5.7 Risk

No risk.

5.8 Special Requirements

Server-side must be correctly configured to access JMS provider.

6 Message conveying from JMS provider to client-side

6.1 Brief Description

This Use-Case describes an action of conveying JMS message from JMS provider to client-side by JMSMEserver-side.

6.2 Goals

Messages sent as persistent must apply to once-and-only-once delivery scheme. Messages sent as non-persistent must apply to at-most-once delivery scheme.

6.3 Metrics

6.4 Entry Conditions

- Application uses JMSME client-side.
- Client-side has network connection to server-side. The connection may be unstable and not durable.
- Server-side has network connection to JMS provider.
- Application works with open JMS Session

ADDITIONALLY IN PUBLISH/SUBSCRIBE MODEL:

- The server-side is subscribed to selected Topic (on behalf of the client-side)

6.5 Workflow

6.5.1 Basic Workflow Steps

BOTH POINT-TO-POINT AND PUBLISH/SUBSCRIBE MODELS:

1. The client-side asks server-side to receive a Message.
2. The server-side receives the Message from JMS provider without acknowledging it.
3. The server-side enqueues the Message in memory.
4. If network connection is available the server-side sends the Message to the client side.
5. If network connection is not available the server-side may:
   (a) try to resend message in every fixed period of time
   (b) try to resend message when the next message to send appears
6. **If the Message is *PERSISTENT*:** The server-side waits for acknowledgment from the client-side.
7. The server-side acknowledges JMS provider.
8. The server-side dequeues the Message.

6.5.2 Alternative Workflow Steps

Alternative workflow bases on a few additional possibilities.

1. The server-side may crash.
   (a) The server-side is restarted.
   (b) Persistent messages are resent by JMS provider (they were not acknowledged).

6.6 Category

This Use-Case is of the core category.

6.7 Risk

No risk.

6.8 Special Requirements

Server-side must be correctly configured to access JMS provider.

7 Revision History

```
$Log: buc.tex,v $
Revision 1.13  2002/11/27 10:09:34  pasza
Zmienilem typ obrazka na pdf, wrzucam tez zrodlo obrazka

Revision 1.12  2002/11/27 07:56:27  lucash
wstawilem obrazek w figure

Revision 1.11  2002/11/26 23:37:00  pasza
Dodałem diagramik (na razie w postaci eps), napisałem ’Description of
the rest of the document’

Revision 1.10  2002/11/26 13:56:57  lucash
Wprowadziłe poprawki w miejscach o których Grześ mówił
(w paru miejscach zlepilem non-pers z pers) - teraz jest chyba czytalniej

Revision 1.9  2002/11/13 23:56:50  pasza
Przeczytałem i drobne poprawki

Revision 1.8  2002/11/13 16:31:15  lucash
przeczyściłem źródło, zakomentowałem polskie pierdoły

Revision 1.7  2002/11/13 14:51:40  piotrek
Rozne zmiany, poprawki, troche uwag, podkonczenie Message Receiving.

Revision 1.6  2002/11/12 16:59:42  lucash
poprawki

Revision 1.5  2002/11/12 14:16:31  lucash
Dopisalem troche ale mam watpliwosci. Sa na grubo
w tekscie.

Revision 1.4  2002/11/11 22:53:31  ofca
wysylanie jest zrobione i czesc odbierania (ale pewno jest sporo
niescislosci) !!!! niech ktos to przejzy. !!!!

Revision 1.3  2002/11/11 13:12:58  lucash
Commands-pliczek z makrami, polowka pierwszego usecasa;)

Revision 1.2  2002/11/07 16:07:14  piotrek
Dodanie ’Message sending’ i innych
```

```
Revision 1.1.1.1  2001/10/30 17:49:15  robmar
zajecia, etc.
```