

Java EE 7 アプリケーション設計ガイド

JMS on Liberty 編

日本アイ・ビー・エム システムズ・エンジニアリング株式会社



Disclaimer

- この資料は日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム システムズ・エンジニアリング株式会社の正式なレビューを受けておりません。
- 当資料は、資料内で説明されている製品の仕様を保証するものではありません。
- 資料の内容には正確を期するよう注意しておりますが、この資料の内容は 2016 年 12 月現在の情報であり、製品の新しいリリース、PTF などによって動作、仕様が変わる可能性があるのでご注意ください。
- 今後国内で提供されるリリース情報は、対応する発表レターなどでご確認ください。
- IBM、IBM ロゴおよび IBM.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。
- 当資料をコピー等で複製することは、日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム システムズ・エンジニアリング株式会社の承諾なしではできません。
- 当資料に記載された製品名または会社名はそれぞれの各社の商標または登録商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- Microsoft, Windows および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。
- Linux は、Linus Torvalds の米国およびその他の国における登録商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。

目次

1. はじめに

- 1.1. JMS とは？
- 1.2. Liberty での JMS 対応

2. どのように使う？

- 2.1. JMS プロバイダーとして使う
- 2.2. JMS クライアントとして使う

3. パターン1：Liberty 組み込みメッセージング・エンジン

- 3.1. 構成例
- 3.2. プロバイダー設定
- 3.3. クライアント設定
- 3.4. サンプル・コード

4. パターン2：IBM MQ メッセージング・プロバイダー

- 4.1. 構成例
- 4.2. プロバイダー設定
- 4.3. クライアント設定
- 4.4. 考慮点

5. MDB (Message Driven Bean)

- 5.1. MDB とは
- 5.2. 構成例-1
- 5.3. 構成例-1：プロバイダー設定/クライアント設定
- 5.4. 構成例-1：MDB設定
- 5.5. 構成例-2
- 5.6. 構成例-2：プロバイダー設定/クライアント設定
- 5.7. 構成例-2：MDB設定

6. その他の設定項目

- 6.1. Liberty 組み込みメッセージングの JMS トレースの有効化
- 6.2. 接続数の調整

1. はじめに

1.1. JMS とは？

1.2. Liberty での JMS 対応

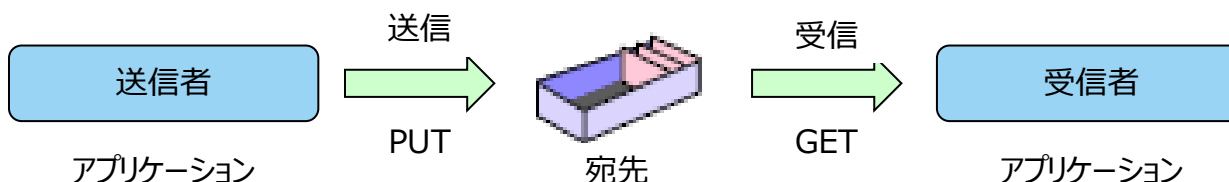
1.1. JMS とは？

■ Java Message Service (JMS)

– Java EE 標準のメッセージング・インターフェイス

- メッセージングとは

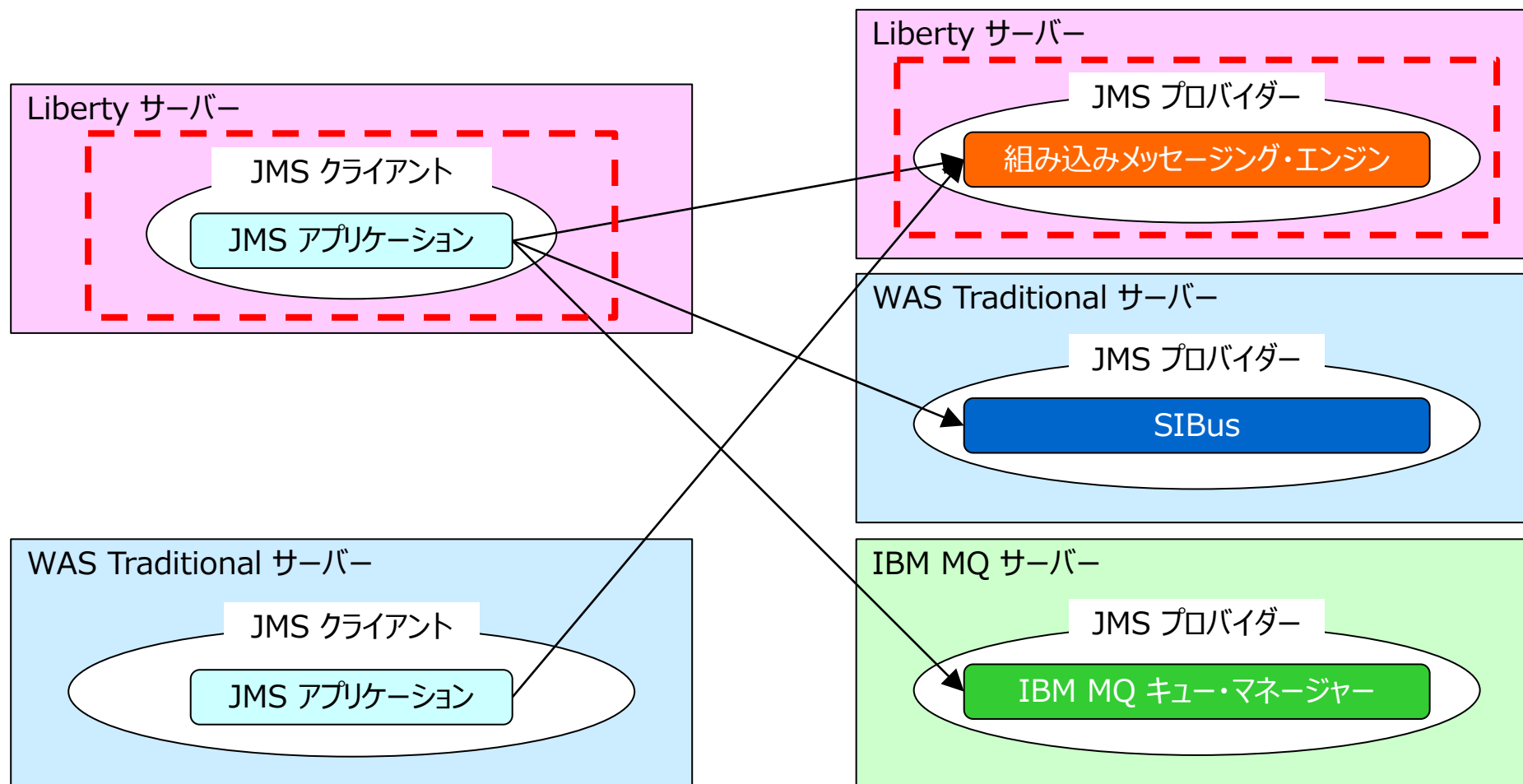
- メッセージを介したプログラム間のコミュニケーション方式
- データをメッセージと呼ばれる塊にまとめて送付
- 人間同士のコミュニケーションで電子メールの送受信に例えられる



- 標準のインターフェイスでコーディングすることにより、メッセージング・サービスに依存しないアプリケーションの開発が可能
- JMS プロバイダーがメッセージの送受信機能を提供
- アプリケーションは JMS クライアントとして動作

1.2. Liberty での JMS 対応 (1/3)

- JMS クライアントとしても、JMS プロバイダーとしても構成可能



1.2. Liberty での JMS 対応 (1/4)

- JMS クライアントとして使用する場合、以下の 3 つの JMS プロバイダーに接続可能
 - Liberty の組み込みメッセージング・エンジン
 - ローカル、リモート両方接続可能
 - WAS traditional のサービス統合バス (SIBus)
 - tWAS のメッセージング機能
 - IBM MQ
 - IBM のメッセージング製品

1.2. Liberty での JMS 対応 (2/4)

- JMS クライアントとして使用する際、各 JMS プロバイダーに接続するために必要なフィーチャー

接続先/用途	JMS2.0	JMS1.1
Liberty 組み込みメッセージング・エンジン、SIBus	wasJmsClient-2.0	wasJmsClient-1.1
IBM MQ	wmqJmsClient-2.0 (※)	wmqJmsClient-1.1 (※)
MDB(Message-Driven Bean)の利用	mdb-3.2	mdb-3.1

(※) wmqJmsClient フィーチャーを使用して MQ のリソース・アダプターを Liberty で使用する場合は、以下の制約事項については、以下を参照。

IBM Knowledge Center : IBM MQ, Version 9.0 「WebSphere Application Server Liberty および IBM MQ リソース・アダプター」

http://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.dev.doc/q120040_.htm

IBM Knowledge Center : WebSphere Application Server Liberty 16.0.0.x 「Liberty: ランタイム環境での既知の問題および制約事項」

http://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_restrict.html

また、汎用 JCA サポートを使用して IBM MQ リソース・アダプターを実行することも可能。詳細については、以下を参照。

IBM Knowledge Center : IBM MQ, Version 9.0 「WebSphere Application Server Liberty および IBM MQ リソース・アダプター」

http://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.dev.doc/q120040_.htm

1.2. Liberty での JMS対応 (3/4)

- JMS プロバイダーとして使用する場合、Liberty 組み込みメッセージング・エンジンを使用
 - 必要なフィーチャー : wasJmsServer-1.0 または javaee-7.0
 - メッセージング・エンジンは Liberty 内で singleton インスタンスとして実行されるため、Liberty サーバーでは 1 つのメッセージング・エンジンのみ稼働可能
 - サポートする JMS 仕様
 - JMS 1.1
 - JMS 2.0
 - サポートするメッセージング・モデル
 - point-to-point 方式
 - publish/subscribe 方式
 - 接続可能なアプリケーション
 - ローカルで稼働するアプリケーション : in-process 接続
 - JMS アプリケーションが、メッセージング・エンジンが実行されているのと同じ JVM 内にデプロイされている場合、アプリケーションは、TCP/IP 層を介することなく、in-process メッセージング・エンジンと通信が可能。
 - リモートの Liberty で稼働するアプリケーション : TCP/IP 接続
 - WAS traditional で稼働するアプリケーション : TCP/IP 接続

1.2. Liberty での JMS対応 (4/4)

- 本ドキュメントでは、Liberty サーバーを以下の用途で使用する際に必要な設定ファイルおよび設定について解説
 - JMS プロバイダーとして使用する場合
 - JMS クライアントとして使用する場合
 - Liberty サーバーで MDB を稼働する場合
- 本ドキュメントでは、Liberty サーバーを JMS プロバイダー / JMS クライアントとして構成、または MDB を稼働するために必要な設定を対象とする
 - 稼働するアプリケーション等についての記述は本ドキュメントでは割愛しているため、実構成では必要に応じて記載

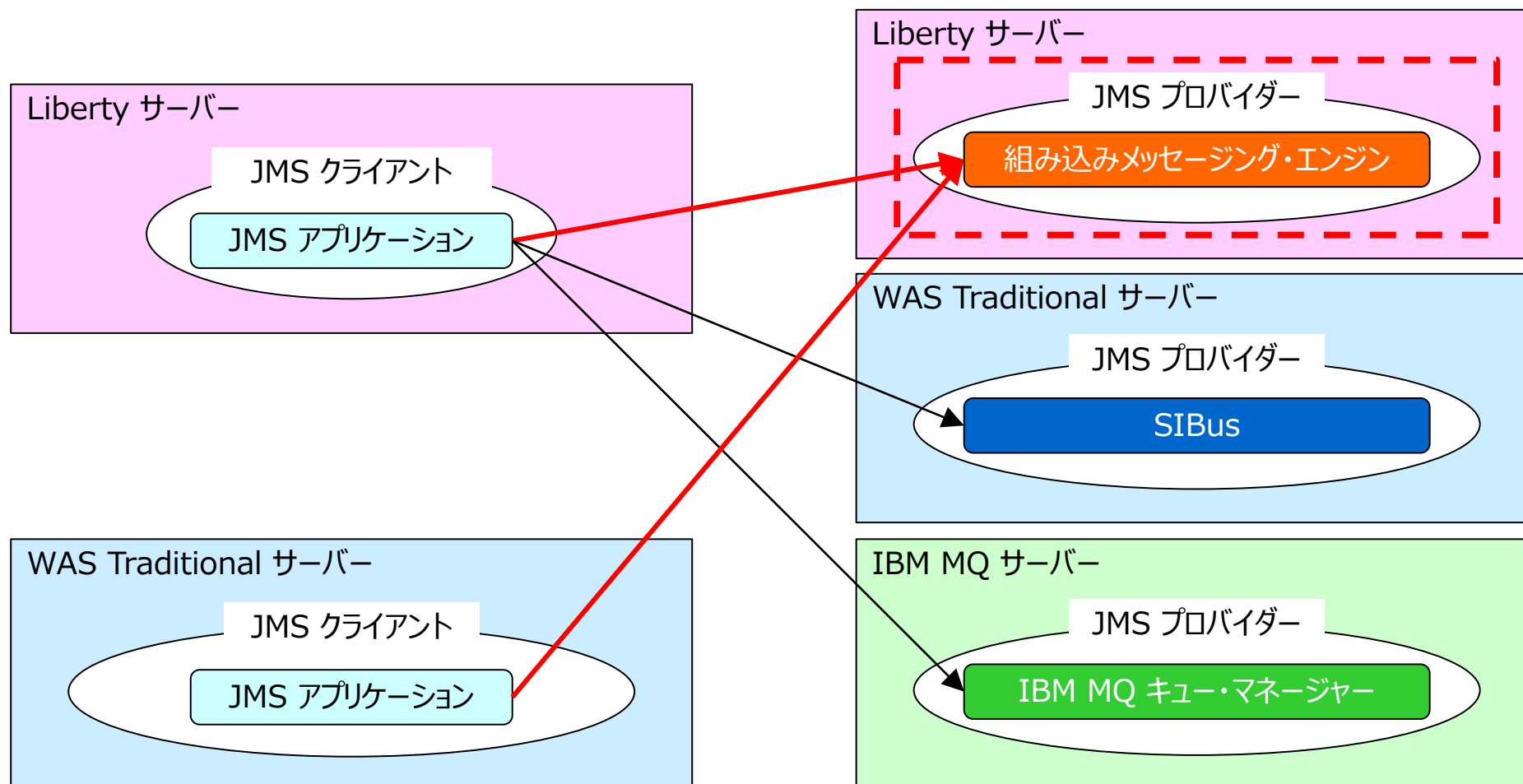
2. どのように使う？

2.1. JMSプロバイダーとして使う

2.2. JMSクライアントとして使う

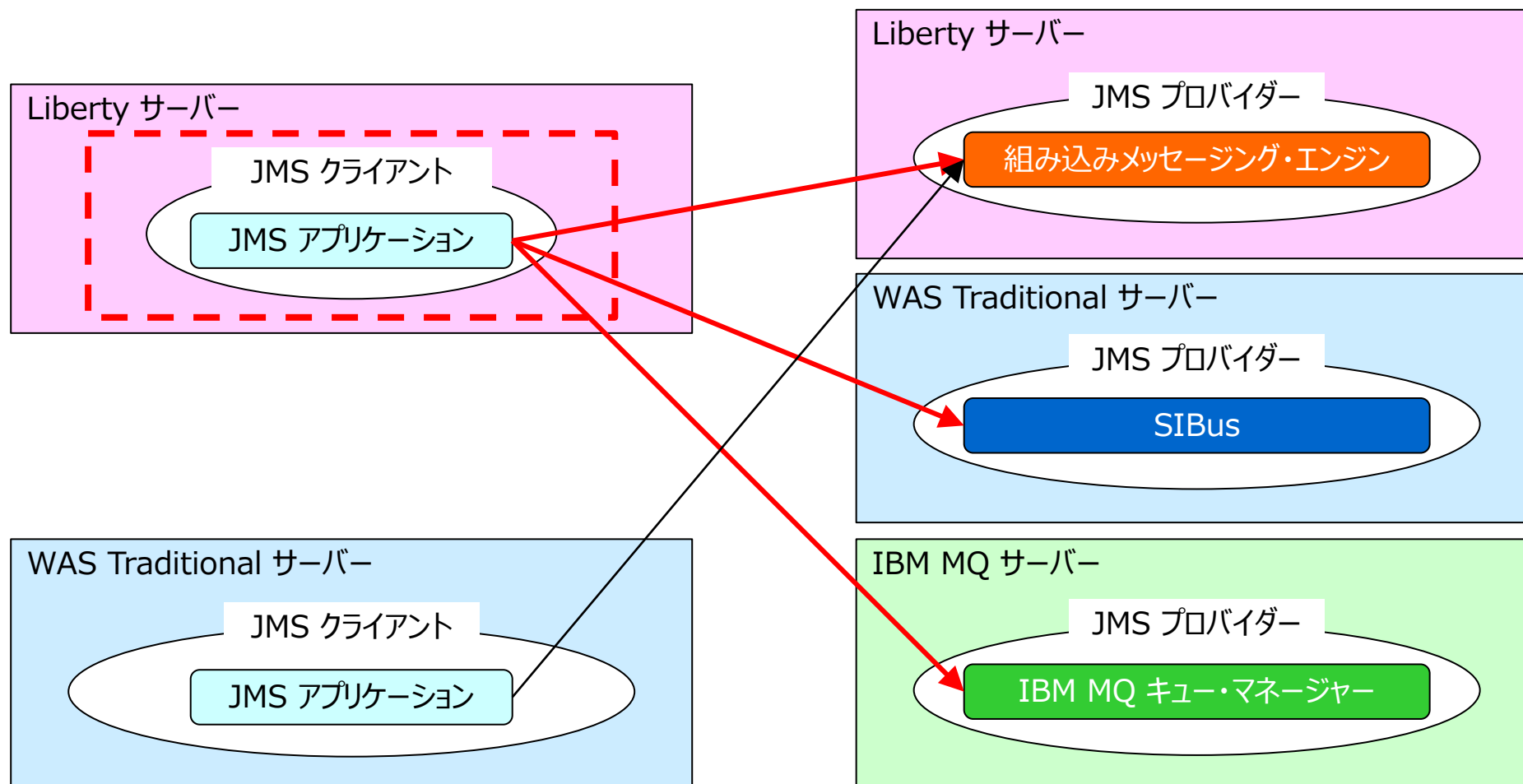
2.1. JMSプロバイダーとして使う

- Libertyサーバーで組み込みメッセージング・エンジンを構成
- JMS クライアントから接続して使用



2.2. JMSクライアントとして使う

- Libertyサーバーで JMS アプリケーションを稼働
- JMS プロバイダーに接続して使用



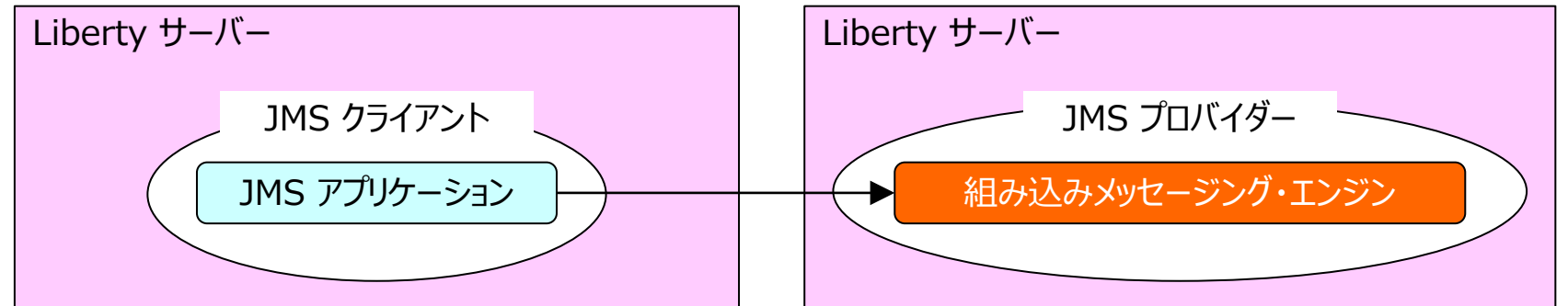
3. パターン1 : Liberty 組み込みメッセージング・エンジン

3.1. 構成例

3.2. プロバイダー設定

3.3. クライアント設定

3.4. サンプル・コード

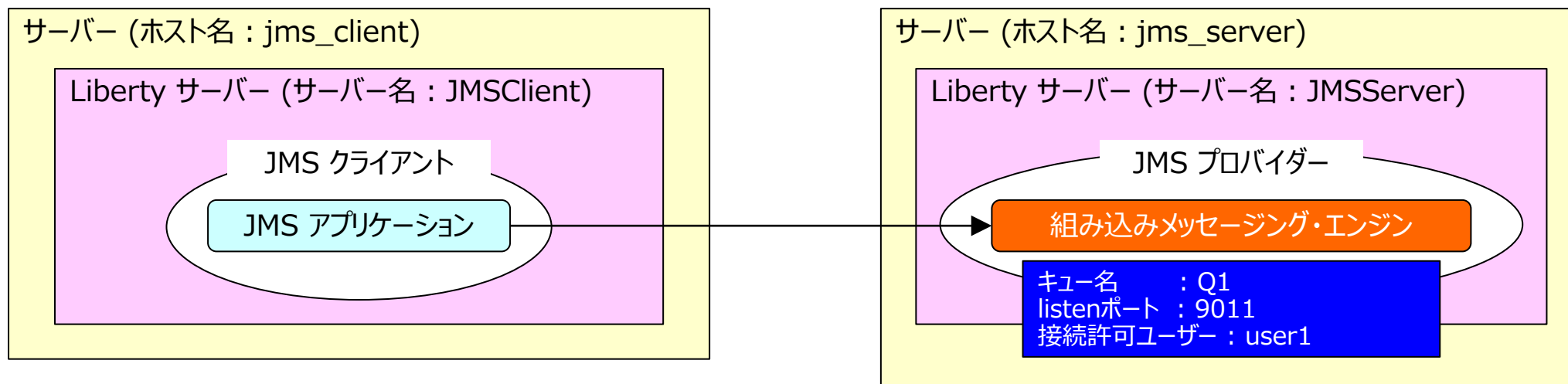


3.1. 構成例

- 以下構成にて Liberty サーバーを JMS プロバイダー/ JMS クライアントとして稼働するために必要な設定ファイルおよび設定を解説

－構成

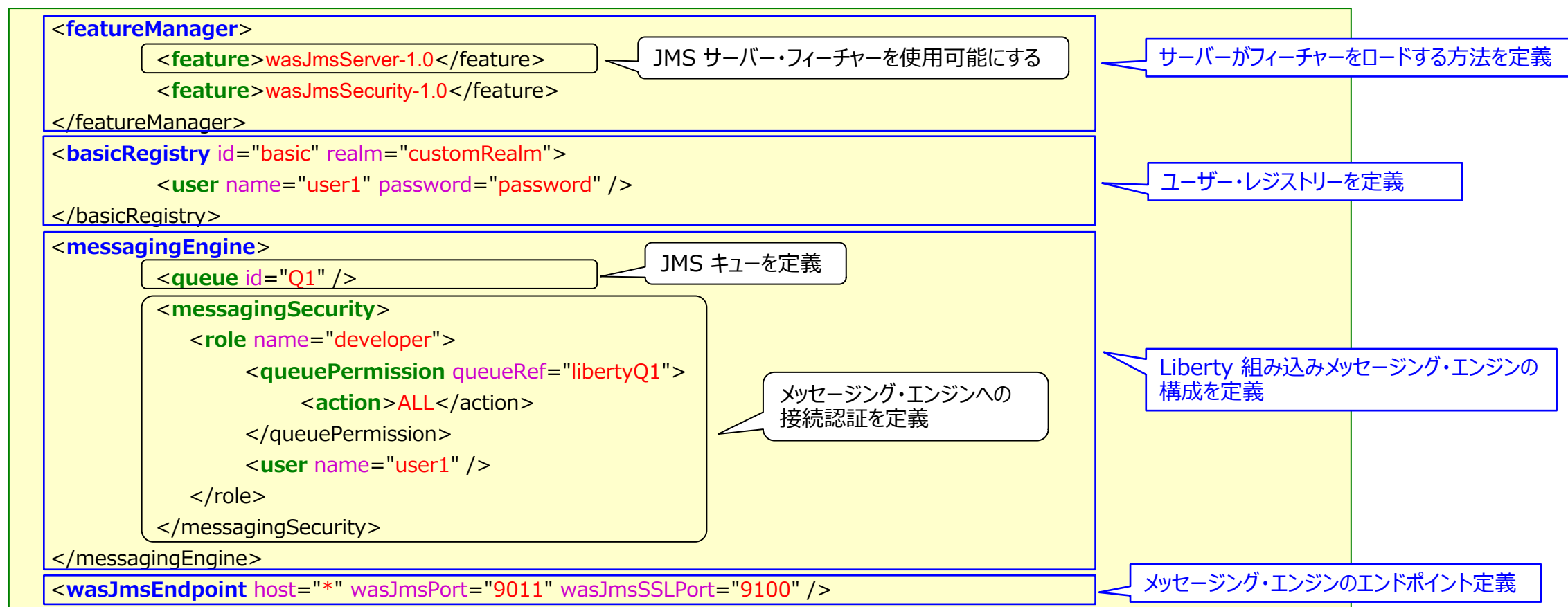
- Liberty サーバー (サーバー名 : JMSClient) を JMS クライアントとして使用
- Liberty サーバー (サーバー名 : JMSServer) で Liberty 組み込みメッセージング・エンジンを構成し、JMS プロバイダーとして使用
- JMS クライアントと JMS プロバイダー(組み込みメッセージング・エンジン)は異なるサーバーで稼働するものとする
 - － 本構成例では異なるサーバーで稼働する構成を取っているが、同一サーバーでの稼働も可能



3.2. プロバイダー設定 (1/4)

■ 構成例の Liberty サーバー「JMSServer」の server.xml サンプル

- Liberty サーバーで Liberty 組み込みメッセージング・エンジンを構成するために、server.xml ファイルに以下を記述
- 各記述の説明については次ページ参照



3.2. プロバイダー設定 (2/4)

■ server.xml に定義する基本要素 (1/3)

– **featureManager** : サーバーがフィーチャーをロードする方法を定義

- **feature** : 使用するフィーチャーを定義

- JMS サーバー・フィーチャー (wasJmsServer-1.0) を使用可能にする
- JNDI 検索を実行する場合は、jndi-1.0 フィーチャーも追加
- メッセージング・エンジンに接続しようとするユーザー/グループの認証を行う場合は、wasJmsSecurity-1.0 フィーチャーも追加
 - wasJmsSecurity-1.0 フィーチャーを追加した場合、ユーザー・レジストリーの構成および **messagingSecurity** エLEMENTの定義が必要

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「フィーチャー・マネージャー (featureManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_featureManager.html

– **basicRegistry** : ユーザー・レジストリーを定義

- ユーザー・レジストリーの定義については以下を参照

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「メッセージング・エンジンに接続するユーザーの認証」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.doc/ae/twlp_msg_sec_authenticate.html

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「基本ユーザー・レジストリー (basicRegistry)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_basicRegistry.html

3.2. プロバイダー設定 (3/4)

■ server.xml に定義する基本要素 (2/3)

– **messagingEngine** : Liberty 組み込みメッセージング・エンジンの構成を定義

- **queue** : JMS キューを定義
 - **id** : キュー定義名
- **messagingSecurity** : アクセス権限を定義 (メッセージング・エンジンへの接続ユーザー/グループ認証を行う場合に設定)
 - デフォルトでは、どのユーザーからでもメッセージング・エンジンへの接続が可能
 - **role** : ユーザーおよびグループにマップされる許可のセット
 - **name** : ロールの名前
 - **queuePermission** : ユーザーおよびグループのセットを対象に、特定のキューに関して定義される許可
 - **queueRef** : 参照する queue の id (メッセージング・エンジンに定義されているキュー)
 - **action** : 許可される操作 (SEND / RECEIVE / BROWSE / ALL)
 - **user** : ロールに割り当てられるユーザー
 - **name** : ユーザー名

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「メッセージング・エンジン (messagingEngine)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_messagingEngine.html

3.2. プロバイダー設定 (4/4)

■ server.xml に定義する基本要素 (3/3)

– **wasJmsEndpoint** : メッセージング・エンジンのエンドポイント定義

- メッセージング・エンジンが listen するホスト名 (IP アドレス) とポートを指定
 - **host** : クライアントがリソースの要求に使用するホスト名 (IP アドレス)
 - ・すべてのネットワーク・インターフェースを使用可能とする場合には「*」を使用
 - **wasJmsPort** : メッセージング・エンジンが listen するポート (非セキュア)
 - **wasJmsSSLPort** : メッセージング・エンジンが listen するポート (セキュア)
- デフォルトとは異なるポートにメッセージング・エンジンをバインドしたい場合に定義
 - 定義しない場合、デフォルトでポート 7276 (非セキュア) および 7286 (セキュア) で listen

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「WAS JMS エンドポイント (wasJmsEndpoint)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_wasJmsEndpoint.html

■ 参考 : その他の基本要素

– **fileStore** : 永続メッセージを設定する場合に定義

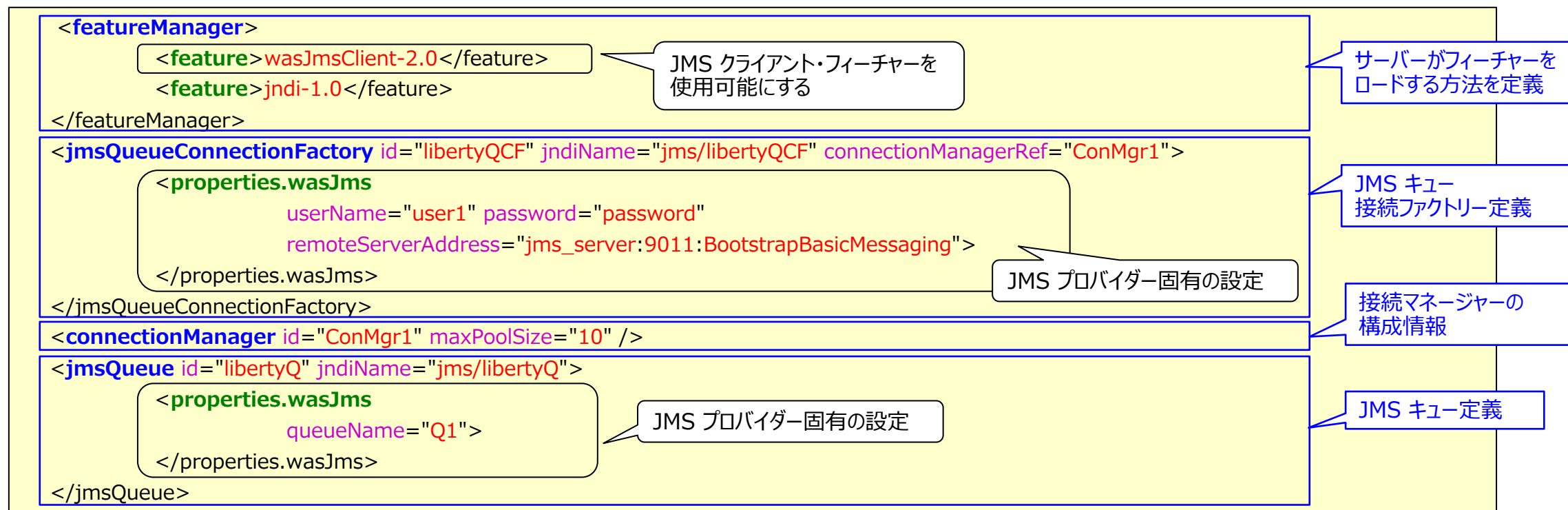
- 詳細については、以下を参照

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「メッセージング・エンジン (messagingEngine)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_messagingEngine.html

3.3. クライアント設定 (1/4)

- 構成例の Liberty サーバー「JMSClient」の server.xml サンプル
 - Liberty サーバーを JMS クライアントとして使用するために、server.xml ファイルに以下を記述
 - 各記述の説明については次ページ参照



3.3. クライアント設定 (2/4)

■ server.xml に定義する基本要素 (1/3)

– **featureManager** : サーバーがフィーチャーをロードする方法を定義

- **feature** : 使用するフィーチャーを定義
 - JMS クライアント・フィーチャー (wasJmsClient-2.0) を使用可能にする
 - JNDI 検索を実行する場合は、jndi-1.0 フィーチャーも追加

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「フィーチャー・マネージャー (featureManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_featureManager.html

3.3. クライアント設定 (3/4)

■ server.xml に定義する基本要素 (2/3)

– **jmsQueueConnectionFactory** : JMS キュー接続ファクトリー定義

- Liberty 組み込みメッセージング・エンジンに接続するための JMS キュー接続ファクトリーを定義
 - **id** : JMS キュー接続ファクトリー定義名
 - **jndiName** : JMS キュー接続ファクトリーの JNDI 名
 - **connectionManagerRef** : 参照する connectionManager の id
- **properties.wasJms** : 組み込みメッセージング・エンジンに接続する場合の JMS プロバイダー固有の設定を定義
 - **userName** : メッセージング・エンジンに接続するユーザー名
 - **password** : ユーザーのパスワード

メッセージング・エンジンに接続するユーザー/パスワードの設定については、以下を参照

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「メッセージング・エンジンに接続するユーザーの認証」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.doc/ae/twlp_msg_sec_authenticate.html

- **remoteServerAddress** : メッセージング・エンジンが稼働するリモートのサーバーに TCP/IP 接続するための設定

`remoteServerAddress="JMS_Server_hostname:JMS_Port:mode"`

`JMS_Server_hostname` : メッセージング・エンジンが稼働するサーバーのホスト名

`JMS_Port` : サーバー側で構成されている JMS インバウンド・ポート

セキュリティが無効な場合 - サーバー側の「wasJmsPort」エレメントで指定されているポート

セキュリティが有効な場合 - サーバー側の「wasJmsSSLPort」エレメントで指定されているポート

`mode` : SSL が使用されない場合 - BootstrapBasicMessaging / SSL が使用される場合 - BootstrapSecureMessaging

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS キュー接続ファクトリー (jmsQueueConnectionFactory)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_jmsQueueConnectionFactory.html

3.3. クライアント設定 (4/4)

■ server.xml に定義する基本要素 (3/3)

– **connectionManager** : 接続マネージャーの構成情報を定義

- デフォルトと異なる値を設定したい場合に定義 (connectionManager エLEMENTで定義されていない接続マネージャー設定については、デフォルトの値が使用される)
 - **id** : 接続マネージャー定義名
 - **maxPoolSize** : プールの物理接続の最大数

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「接続マネージャー (connectionManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_connectionManager.html

– **jmsQueue** : JMS キュー定義

- **properties.wasJms** : 組み込みメッセージング・エンジンに接続する場合の JMS プロバイダー固有の設定を定義
 - **id** : JMS キュー定義名
 - **jndiName** : JMS キューの JNDI 名
 - **queueName** : 使用する JMS キュー名

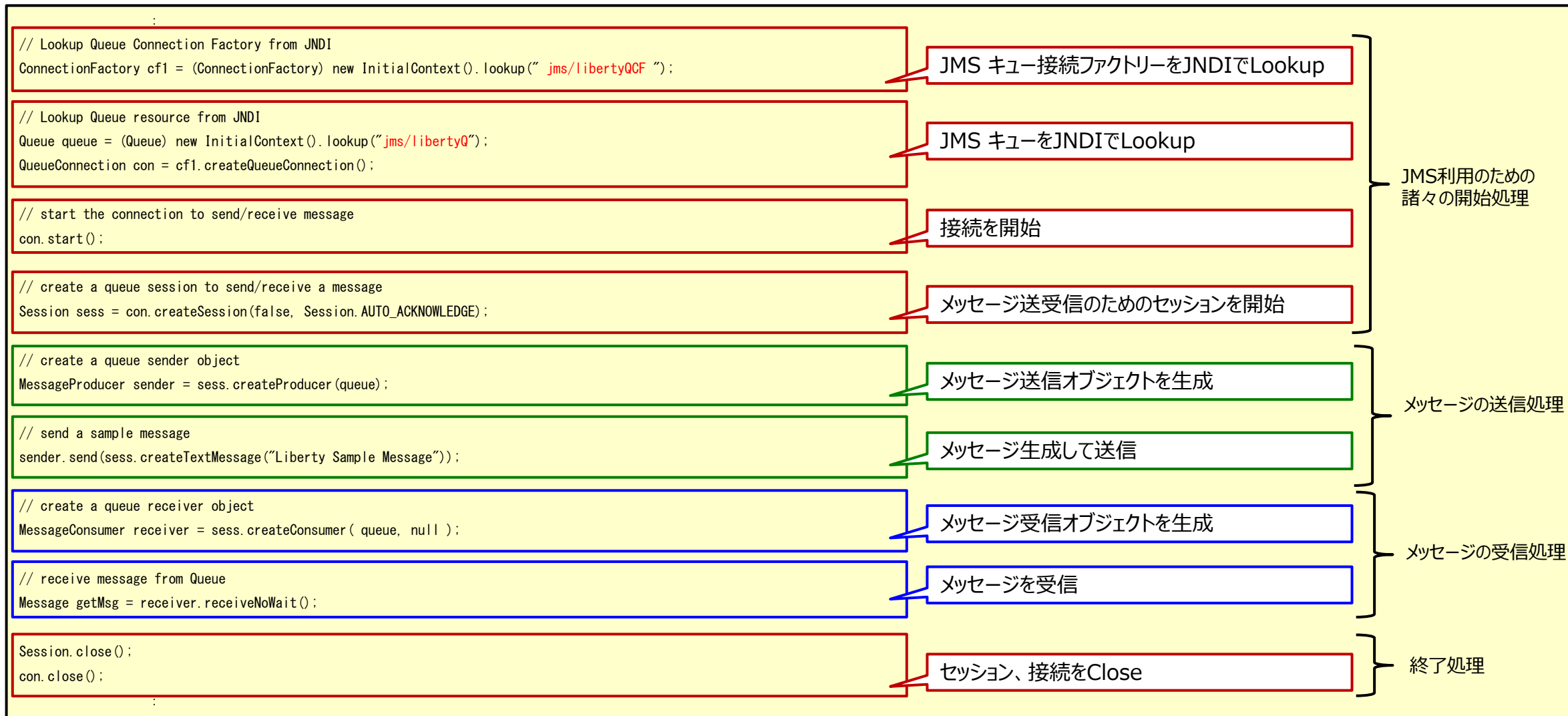
【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS キュー (jmsQueue)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_jmsQueue.html

3.4. サンプル・コード

- 前頁までの設定例を前提とした環境で動作するクライアント・コードのサンプル（メッセージを送信してすぐに受信するというシンプルな処理）
- 関連箇所を抜粋して記述



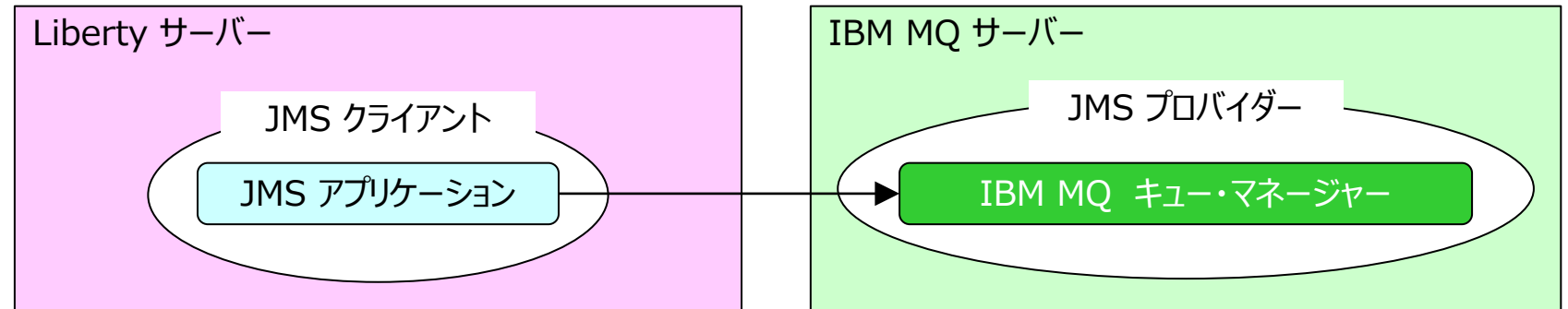
4. パターン2 : IBM MQ メッセージング・プロバイダー

4.1. 構成例

4.2. プロバイダー設定

4.3. クライアント設定

4.4. 考慮点

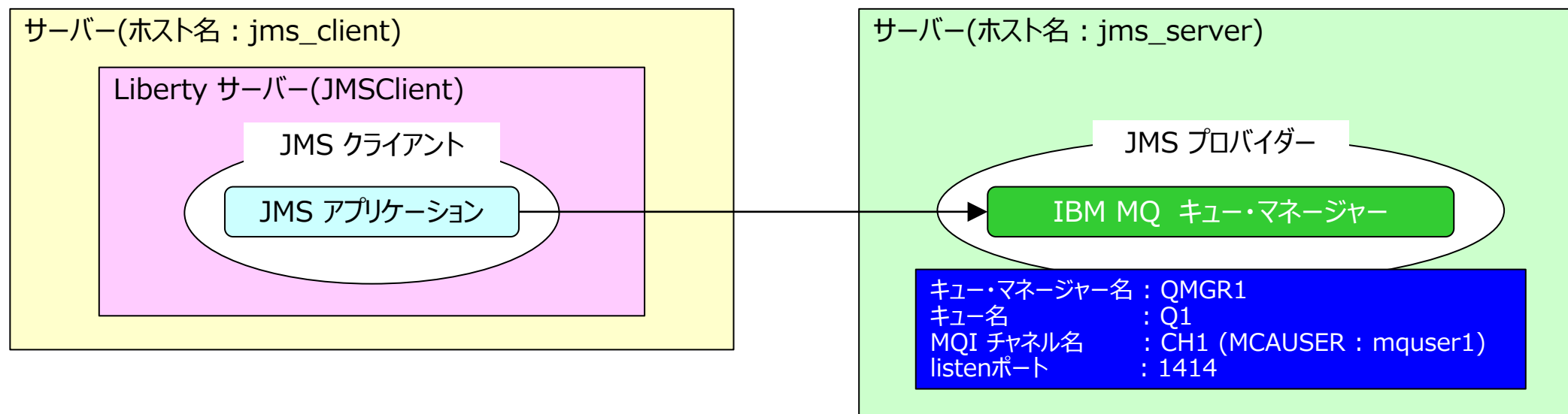


4.1. 構成例

- 以下構成にて Liberty サーバーを IBM MQ サーバーに接続し JMS クライアントとして稼働するために必要な設定ファイルおよび設定を解説

ー構成

- Liberty サーバー (サーバー名 : JMSClient) を JMS クライアントとして使用
- IBM MQ サーバーを JMS プロバイダーとして使用
- JMS クライアントと JMS プロバイダー (IBM MQ) は異なるサーバーで稼働するものとする
- MQI チャンネルの MCAUSER 属性に設定しているユーザー「mquser1」は、適切な MQ 権限を持つユーザーとする



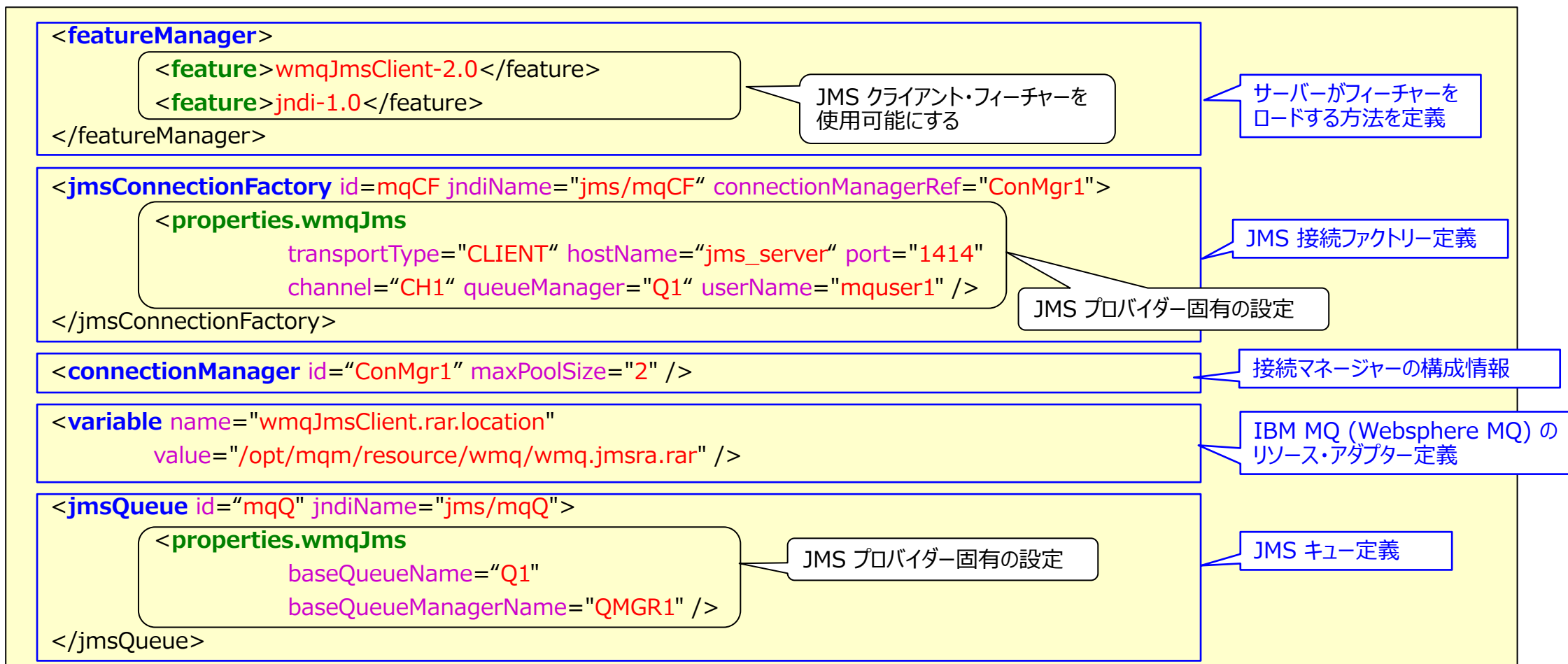
4.2. プロバイダー設定

- IBM MQ サーバーで、JMS プロバイダーとして必要な定義がされている前提とする
(本ドキュメントでは、MQ サーバー側の設定については割愛する)

4.3. クライアント設定 (1/5)

■ 構成例の Liberty サーバー「JMSSClient」の server.xml サンプル

- Liberty サーバーを JMS クライアントとして使用するために、server.xml ファイルに以下を記述
- 各記述の説明については次ページ参照



4.3. クライアント設定 (2/5)

■ server.xml に定義する基本要素 (1/4)

– **featureManager** : サーバーがフィーチャーをロードする方法を定義

- **feature** : 使用するフィーチャーを定義

- JMS クライアント・フィーチャー (wmqJmsClient-2.0) を使用可能にする

- JNDI 検索を実行する場合は、jndi-1.0 フィーチャーも追加

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「フィーチャー・マネージャー (featureManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_featureManager.html

4.3. クライアント設定 (3/5)

■ server.xml に定義する基本要素 (2/4)

– **jmsConnectionFactory** : JMS 接続ファクトリー定義

- IBM MQ(WebSphere MQ)に接続するための JMS 接続ファクトリーを定義
 - **id** : JMS 接続ファクトリー定義名
 - **jndiName** : JMS 接続ファクトリーの JNDI 名
 - **connectionManagerRef** : 参照する connectionManager の id
- **properties.wmqJms** : IBM MQ(WebSphere MQ)に接続する場合の JMS プロバイダー固有の設定を定義
 - **transportType** : キュー・マネージャーへの接続モード (CLIENT / BINDINGS)
 - ローカル構成の場合は BINDINGS 接続 (Java Native Interface (JNI) を使用して直接キュー・マネージャーに接続)、リモート構成の場合は CLIENT 接続 (TCP/IP を介してキュー・マネージャーに接続) を利用 (デフォルトは CLIENT)
 - 接続方式については、以下を参照
 - WAS虎の巻: 第4回「JMSとメッセージング・サービス接続」 5. メッセージング・トポロジ
 - <http://www.ibm.com/developerworks/jp/websphere/library/was/toranomaki/4.html>
 - BINDINGS 接続を利用する場合 : 「4.3. クライアント設定 (5/5)」の「BINDINGS 接続を利用する場合」参照
 - **hostName** : キュー・マネージャーが存在しているサーバーのホスト名またはIPアドレス
 - **port** : キュー・マネージャーの listen ポート
 - **channel** : 使用する MQI チャンネルの名前
 - **queueManager** : 接続するキュー・マネージャー名
 - **userName** : 接続するユーザー名

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS 接続ファクトリー (jmsConnectionFactory)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_jmsConnectionFactory.html

4.3. クライアント設定 (4/5)

■ server.xml に定義する基本要素 (3/4)

– **connectionManager** : 接続マネージャーの構成情報を定義

- デフォルトと異なる値を設定したい場合に定義 (connectionManager エLEMENTで定義されていない接続マネージャー設定については、デフォルトの値が使用される)
 - **id** : 接続マネージャー定義名
 - **maxPoolSize** : プールの物理接続の最大数

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「接続マネージャー (connectionManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_connectionManager.html

– **variable** : IBM MQ (Websphere MQ) のリソース・アダプター定義

- IBM MQ (Websphere MQ) のリソース・アダプターのロケーションを定義
 - **name** : wmqJmsClient.rar.location
 - **value** : IBM MQ リソース・アダプター・ファイル wmq.jmsra.rar への絶対パス
- ※ IBM MQ リソース・アダプター・ファイルは事前にダウンロードしてサーバー(本構成例ではサーバーjms_client)上に置いておく必要がある。ダウンロードについては以下を参照。

Obtaining the IBM MQ Resource Adapter for the WebSphere Application Server Liberty Profile

<http://www-01.ibm.com/support/docview.wss?uid=swg21633761>

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「変数宣言 (variable)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multipatform.doc/ae/rwlp_config_variable.html

4.3. クライアント設定 (5/5)

■ server.xml に定義する基本要素 (4/4)

– **jmsQueue** : JMS キュー定義

- JMS キューを定義
 - **id** : JMS キュー定義名
 - **jndiName** : JMS キューの JNDI 名
- **properties.wasJms** : IBM MQ (Websphere MA) に接続する場合の JMS プロバイダー固有の設定を定義
 - **baseQueueName** : キュー・マネージャー上のキュー名
 - **baseQueueManagerName** : キューが定義されているキュー・マネージャー名

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS キュー (jmsQueue)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_jmsQueue.html

– BINDINGS 接続を利用する場合

- バインディング・モードで使用する、ネットワーク経由で通信するのではなく、Java Native Interface (JNI) を使用して、既存のキュー・マネージャー API を直接呼び出します。
- **wmqJmsClient** にて **nativeLibraryPath** 属性を設定
 - **nativeLibraryPath** : IBM MQ Java JNI ライブラリー (mqjbnd.dll またはそれに相当するもの) のロケーションへの絶対パス

```
<wmqJmsClient nativeLibraryPath="/opt/mqm/java/lib64"/>
```

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「WebSphere MQ リソース・アダプター (wmqJmsClient)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_wmqJmsClient.html

4.4. 考慮点 (1/3)

■ JMS アプリケーションで MQ クラスを利用する場合の考慮点

– JMS クライアントでフィーチャー `wmqJmsClient` を使用した場合、リソース・アダプターの JCA 処理はアプリケーションから参照するクラスローダーとは別の参照で呼ばれる。

そのため、アプリケーション側で MQ クラスの利用が必要な場合、現状は MQ class for JMS のライブラリとして、別途共有ライブラリを関連付ける必要がある。

- リソース・アダプター `wmq.jmsra.rar` を解凍し、含まれているクラスファイル (`com.ibm.mq.xxx.jar`) をアプリケーションが参照可能な共有ライブラリとして定義

– server.xml 記述例

- server.xml にて、アプリケーションが参照可能な共有ライブラリとしてクラスファイル(`com.ibm.mq.xxx.jar`) を明示的に定義

```
<webApplication
    id="JMSApp" location="JMSApp.war" name="JMSApp" contextRoot="TestForJMSApp">
    <classloader privateLibraryRef="MQLIB"/>
</webApplication>
<library id="MQLIB" name="MQLIB" filesetRef="mqlibf">
    <fileset dir="${shared.resource.dir}/wmq/mqlib" id="mqlibf" includes="com.ibm.mq.connector.jar,
        com.ibm.mq.headers.jar, com.ibm.mq.jar, com.ibm.mq.jmqi.jar, com.ibm.mq.pcf.jar,
        com.ibm.mqjms.jar">
    </fileset>
</library>
```

4.4. 考慮点 (2/3)

- JMS アプリケーションから MQ の MsgId、CorrelID を指定/取得する場合の考慮点 (1/2)
 - Liberty 上の JMS アプリケーションから JMS_IBM_MQMD_* プロパティを使用して MsgId、CorrelID を指定/取得する場合
 - JMS キュー定義の properties.wasJms にて、arbitraryProperties に MDWRITE="YES", MDREAD="YES" を指定
 - 設定値にダブルクォートが含まれるため、server.xmlでは「arbitraryProperties='MDWRITE="YES"」のように、シングルクォートで囲んで指定
 - コンテキスト情報のレベルを指定したい場合は、arbitraryProperties に MDMSGCTX="..." を追加
 - 【 参考 】
 - IBM Knowledge Center : IBM MQ バージョン 9.0 「IBM MQ classes for JMS オブジェクトのプロパティ : MDWRITE」
[https://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.ref.dev.doc/q112160 .htm](https://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.ref.dev.doc/q112160.htm)
 - IBM Knowledge Center : IBM MQ バージョン 9.0 「IBM MQ classes for JMS オブジェクトのプロパティ : MDREAD」
[https://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.ref.dev.doc/q112150 .htm](https://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.ref.dev.doc/q112150.htm)
 - IBM Knowledge Center : IBM MQ バージョン 9.0 「IBM MQ classes for JMS オブジェクトのプロパティ : MDMSGCTX」
[https://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.ref.dev.doc/q112170 .htm](https://www.ibm.com/support/knowledgecenter/ja/SSFKSJ_9.0.0/com.ibm.mq.ref.dev.doc/q112170.htm)
 - targetClient=MQ を指定することで、不要な JMS ヘッダーを削除することが可能

4.4. 考慮点 (3/3)

- JMS アプリケーションから MQ の MsgId、CorrelID を指定/取得する場合の考慮点 (2/2)
 - MsgID、CorrelID を指定する場合の sserver.xml 記述例

```
<jmsQueue id="mqQ" jndiName="jms/mqQ">  
  <properties.wmqJms  
    baseQueueName="Q1"  
    baseQueueManagerName="QMGR1"  
    arbitraryProperties='MDWRITE="YES"'  
    targetClient="MQ" />  
</jmsQueue>
```

5. MDB (Message Driven Bean)

5.1. MDB とは

5.2. 構成例-1

5.3. 構成例-1 : プロバイダー設定/クライアント設定

5.4. 構成例-1 : MDB設定

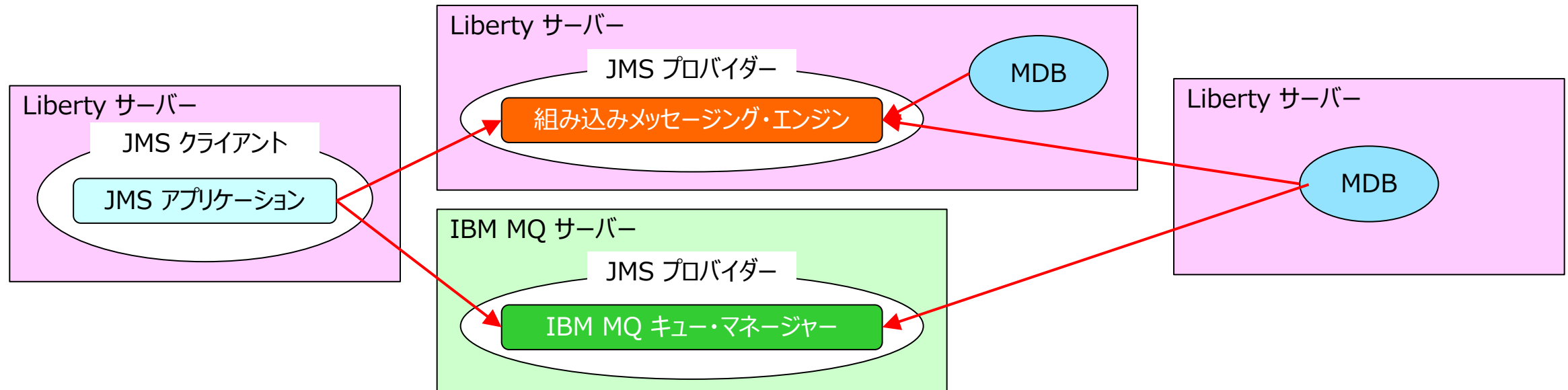
5.5. 構成例-2

5.6. 構成例-2 : プロバイダー設定/クライアント設定

5.7. 構成例-2 : MDB設定

5.1. MDB とは

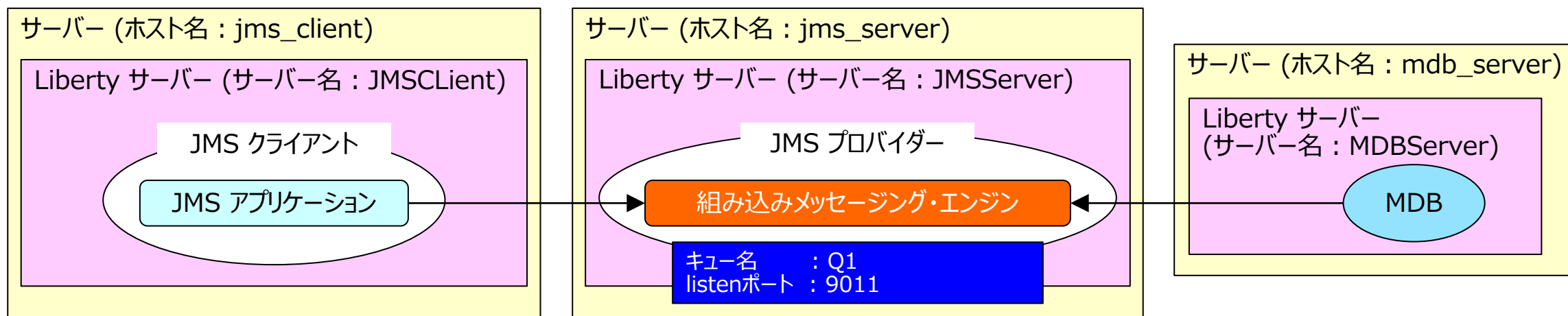
- JMS API を利用したメッセージ駆動型処理を行う EJB
- MDB はメッセージを受信した時に EJB コンテナによって呼び出され、メッセージを EJB アプリケーションで処理することが可能
- 並列にメッセージ受信およびビジネスロジックの実行が可能
- MDB では、どのサーバーに接続して、どの宛先を監視するかといった設定をアクティベーション・スペック (Activation Specification : 活動化仕様) に記述
- EJB のデプロイメント時に、どのアクティベーション・スペックを利用するかを指定



5.2. 構成例-1

- 以下構成にて Liberty サーバーにて MDB を使用可能とするために必要な設定ファイルおよび設定を解説
ー構成

- Liberty サーバー (サーバー名 : JMSClient) を JMS クライアントとして使用
- Liberty サーバー (サーバー名 : JMSServer) で Liberty 組み込みメッセージング・エンジンを構成し、JMS プロバイダーとして使用
- Liberty サーバー (サーバー名 : MDBServer) で MDB を稼働
- JMS クライアント、JMS プロバイダー(組み込みメッセージング・エンジン)、MDB は異なるサーバーで稼働するものとする

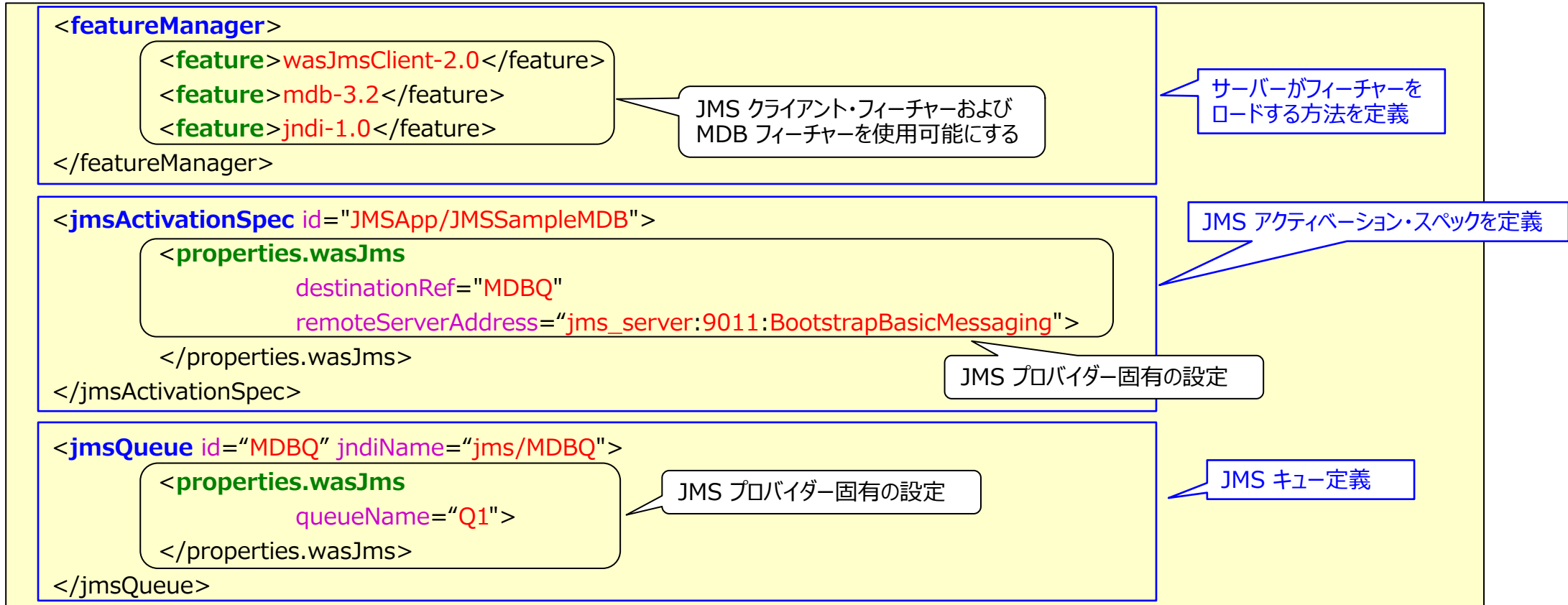


5.3. 構成例-1 : プロバイダー設定 / クライアント設定

- JMS プロバイダーおよび JMS クライアント構成は「3.1. 構成例」と同じ構成のため、「3.2. プロバイダー設定」および「3.3. クライアント設定」を参照

5.4. 構成例-1 : MDB設定 (1/3)

- 構成例-1 の Liberty サーバー「MDBServer」の server.xml サンプル
 - Liberty サーバーで MDB を稼働するために、server.xml ファイルに以下を記述
 - 各記述の説明については次ページ参照



5.4. 構成例-1 : MDB設定 (2/3)

■ server.xml に定義する基本要素 (1/2)

– **featureManager** : サーバーがフィーチャーをロードする方法を定義

- **feature** : 使用するフィーチャーを定義
 - JMS クライアント・フィーチャー (wasJmsClient-2.0) を使用可能にする
 - MDB フィーチャー (mdb-3.2) を使用可能にする
 - JNDI 検索を実行する場合は、jndi-1.0 フィーチャーも追加

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「フィーチャー・マネージャー (featureManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_featureManager.html

– **jmsActivationSpec** : JMS アクティベーション・スペックを定義

- Liberty 組み込みメッセージング・エンジンに接続するための JMS アクティベーション・スペックを定義
 - **id** : JMS アクティベーション・スペック定義名
 - ※ 形式 : デプロイされるアプリケーションの名前/Bean がパッケージされているモジュールの名前/エンタープライズ Bean の ejb-name
- **properties.wasJms** : 組み込みメッセージング・エンジンに接続する場合の JMS プロバイダー固有の設定を定義
 - **destinationRef** : 参照する jmsQueue の id
 - **remoteServerAddress** : メッセージング・エンジンが稼働するリモートのサーバーに TCP/IP 接続するための設定

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS アクティベーション・スペック (jmsActivationSpec)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_jmsActivationSpec.html

5.4. 構成例-1 : MDB設定 (3/3)

■ server.xml に定義する基本要素 (2/2)

– **jmsQueue** : JMS キュー定義

- **properties.wasJms** : 組み込みメッセージング・エンジンに接続する場合の JMS プロバイダー固有の設定を定義
 - **id** : JMS キュー定義名
 - **jndiName** : JMS キューの JNDI 名
 - **queueName** : 使用する JMS キュー名

【 参考 】

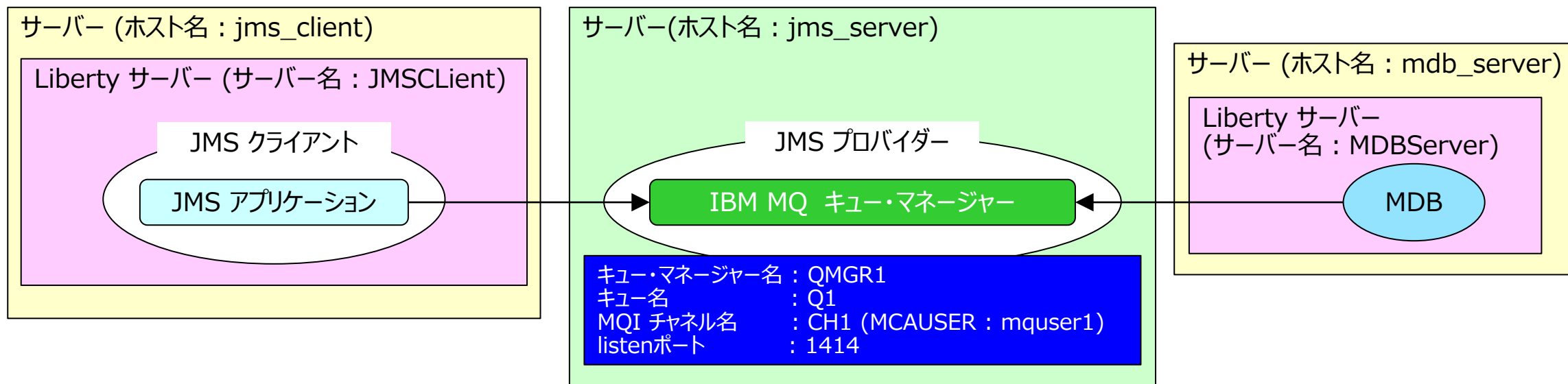
IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS キュー (jmsQueue)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_jmsQueue.html

5.5. 構成例-2

- 以下構成にて Liberty サーバーにて MDB を使用可能とするために必要な設定ファイルおよび設定を解説
ー構成

- Liberty サーバー (サーバー名 : JMSClient) を JMS クライアントとして使用
- IBM MQ サーバーを JMS プロバイダーとして使用
- Liberty サーバー (サーバー名 : MDBServer) で MDB を稼働
- JMS クライアント、JMS プロバイダー(組み込みメッセージング・エンジン)、MDB は異なるサーバーで稼働するものとする

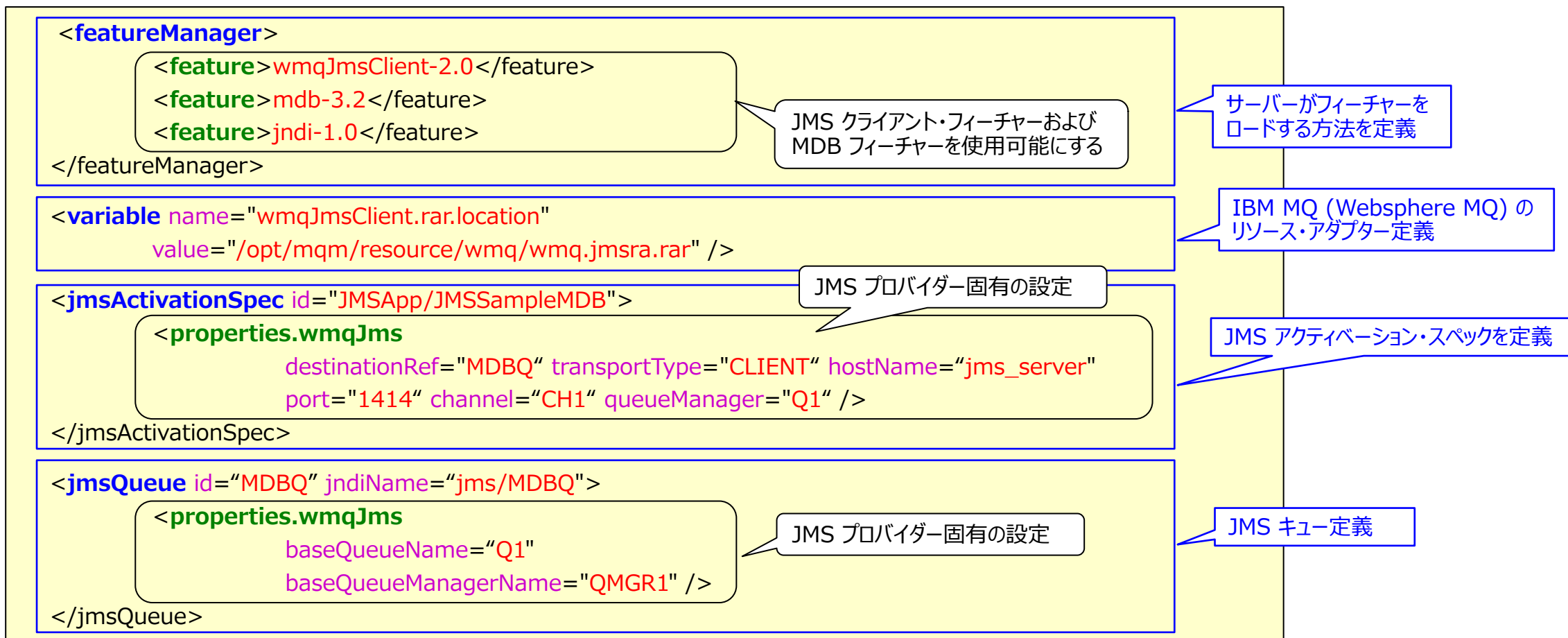


5.6. 構成例-2 : プロバイダー設定 / クライアント設定

- JMS プロバイダーおよび JMS クライアント構成は「4.1. 構成例」と同じ構成のため、「4.2. プロバイダー設定」および「4.3. クライアント設定」を参照

5.7. 構成例-2 : MDB設定 (1/4)

- 構成例-2 の Liberty サーバー「MDBServer」の server.xml サンプル
 - Liberty サーバーで MDB を稼働するために、server.xml ファイルに以下を記述
 - 各記述の説明については次ページ参照



5.7. 構成例-2 : MDB設定 (2/4)

■ server.xml に定義する基本要素 (1/3)

– **featureManager** : サーバーがフィーチャーをロードする方法を定義

- **feature** : 使用するフィーチャーを定義
 - JMS クライアント・フィーチャー (wmqJmsClient-2.0) を使用可能にする
 - MDB フィーチャー (mdb-3.2) を使用可能にする
 - JNDI 検索を実行する場合は、jndi-1.0 フィーチャーも追加

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「フィーチャー・マネージャー (featureManager)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_featureManager.html

– **variable** : IBM MQ (Websphere MQ) のリソース・アダプター定義

- IBM MQ (Websphere MQ) のリソース・アダプターのロケーションを定義
 - **name** : wmqJmsClient.rar.location
 - **value** : IBM MQ リソース・アダプター・ファイル wmq.jmsra.rar への絶対パス

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「変数宣言 (variable)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_variable.html

5.7. 構成例-2 : MDB設定 (3/4)

■ server.xml に定義する基本要素 (2/3)

– **jmsActivationSpec** : JMS アクティベーション・スペックを定義

- IBM MQ(WebSphere MQ)に接続するための JMS アクティベーション・スペックを定義
 - **id** : JMS アクティベーション・スペック定義名
 - ※ 形式 : デプロイされるアプリケーションの名前/Bean がパッケージされているモジュールの名前/エンタープライズ Bean の ejb-name
- **properties.wmqJms** : IBM MQ(WebSphere MQ)に接続する場合の JMS プロバイダー固有の設定を定義
 - **destinationRef** : 参照する jmsQueue の id
 - **transportType** : キュー・マネージャーへの接続モード (CLIENT 接続または BINDINGS 接続)
 - **hostName** : キュー・マネージャーが存在しているサーバーのホスト名またはIPアドレス
 - **port** : キュー・マネージャーの listen ポート
 - **channel** : 使用する MQI チャンネルの名前
 - **queueManager** : 接続するキュー・マネージャー名

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS アクティベーション・スペック (jmsActivationSpec)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_jmsActivationSpec.html

5.7. 構成例-2 : MDB設定 (4/4)

■ server.xml に定義する基本要素 (3/3)

– **jmsQueue** : JMS キュー定義

- JMS キューを定義
 - **id** : JMS キュー定義名
 - **jndiName** : JMS キューの JNDI 名
- **properties.wasJms** : IBM MQ (Websphere MA) に接続する場合の JMS プロバイダー固有の設定を定義
 - **baseQueueName** : キュー・マネージャー上のキュー名
 - **baseQueueManagerName** : キューが定義されているキュー・マネージャー名

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「JMS キュー (jmsQueue)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_jmsQueue.html

6. その他の設定項目

6.1. Liberty 組み込みメッセージングの JMS トレースの有効化

6.2. 接続数の調整

6.1. Liberty 組み込みメッセージングの JMS トレースの有効化 (1/4)

- Liberty では、問題判別のために組み込みメッセージングの JMS トレースを使用可能
– server.xml ファイルで以下のトレース・ストリングを設定 (1/3)

- 接続の問題：接続の問題に関する情報を収集

```
SIBTrm=all
```

- 通信および TCP/IP：アプリケーション・サーバー・チャネル・フレームワークおよび TCP/IP ネットワーク通信に関する情報を収集

```
SIBCommunications=all:SIBJFapChannel=all:TCPChannel=fine:com.ibm.io.async.*=all
```

- JMS クライアント・アプリケーション：JMS アプリケーションからの要求に関する情報を収集

```
SIBJms*=all:SIBCommunications=all:SIBJFapChannel=all:SIBMessageTrace=all:SIBTrm=all:SIBJmsRa=all:SIBRa=all
```

- ロックされたメッセージ：MDB およびアプリケーションに送信中のメッセージが予期しているよりも長くロックされている場合

```
SIBProcessor=all:SIBMessageTrace=all
```

- メッセージ駆動型 Bean：メッセージの宛先に対してアクティベーション・スペックを使用して構成されている MDB に関する情報を収集

```
SIBMessageTrace=all:SIBJmsRa=all:SIBRa=all
```

6.1. Liberty 組み込みメッセージングの JMS トレースの有効化 (2/4)

– server.xml ファイルで以下のトレース・ストリングを設定 (2/3)

- メッセージのフォーマットおよびスキーマ：メッセージ・データの構文解析または処理で問題が発生した場合

```
SIBMfp=all:SIBCommunications=all
```

- メッセージ・プロセッサ：メッセージング・エンジンのコア機能に関する情報を収集 (データのサイズが大きくなる可能性あり)

```
SIBProcessor=all:SIBMessageTrace=all
```

- メッセージ・ストア：障害の発生時にリカバリーできるようにストレージに書き込まれるパーシスタント・メッセージ・データに関する情報を収集

```
SIBMessageStore=all
```

- パフォーマンスおよびメッセージのトラッキング：メッセージの引き渡しの遅延を判別する場合、またはメッセージの宛先を確認する必要がある場合

```
SIBMessageTrace=all
```

- パブリッシュ/サブスクライブ：サブスクライバーがトピックの適切なパブリケーションを取得していない場合

```
SIBMatchSpace=all:SIBProcessor=all
```

6.1. Liberty 組み込みメッセージングの JMS トレースの有効化 (3/4)

–server.xml ファイルで以下のトレース・ストリングを設定 (3/3)

- セキュリティー：ユーザーが誤って認証される場合、あるいはリソースへのアクセスが誤って許可または却下される場合

```
SIBSecurity=all
```

【 参考 】

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「Liberty 組み込みメッセージングの JMS トレースの有効化」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.doc/ae/rwlp_dep_jms_trace.html

6.1. Liberty 組み込みメッセージングの JMS トレースの有効化 (4/4)

■ 例) 通信および TCP/IP のトレースを有効化する場合

- Liberty サーバーの server.xml で以下を記載すると、<LIBERTY_HOME>/usr/servers/<serverName>/logs 配下に trace.log が作成され、通信および TCP/IP のトレースが収集される

```
<logging traceSpecification="SIBCommunications=all:SIBJFapChannel=all:TCPChannel=fine:com.ibm.io.async.*=all"
        traceFileName="trace.log"
        maxFileSize="20"
        maxFiles="10"
        traceFormat="BASIC" />
```

– **logging** エLEMENTの各属性については、以下を参照

IBM Knowledge Center : WebSphere Application Server Liberty 16.0.0.x 「ロギングおよびトレース」

http://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.core.doc/ae/rwlp_logging.html

Setting up trace and getting a full dump in the WebSphere Application Server Liberty profile

<http://www-304.ibm.com/support/docview.wss?uid=swg21596714>

IBM Knowledge Center : WebSphere Application Server traditional 9.0.0.x 「ロギング (logging)」

https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_9.0.0/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/rwlp_config_logging.html

6.2. 接続数の調整

- JMS のチューニング項目の一つとして接続プールに保持する最大接続数/最小接続数があるが、Liberty でも JMS 接続ファクトリーからの接続数のカスタマイズが可能
- 特定の JMS 接続ファクトリーの接続プーリングの構成は connectionManager エlementを定義して行い、次にそのElementを server.xml ファイル内の JMS 接続ファクトリーに関連付ける
- server.xml サンプル
 - **connectionManager** : 接続マネージャーの構成情報を定義
 - **id** : 接続マネージャー定義名
 - **maxPoolSize** : プールの物理接続の最大数 (デフォルト : 50)
 - **minPoolSize** : プールで保持する物理接続の最小数

```
<jmsQueueConnectionFactory id="libertyQCF" jndiName="jms/libertyQCF" connectionManagerRef="ConMgr1">  
  <properties>  
    <property name="wasJms" value="true" />  
    <property name="remoteServerAddress" value="jms_server:9011:BootstrapBasicMessaging" />  
  </properties>  
</jmsQueueConnectionFactory>  
  
<connectionManager id="ConMgr1" maxPoolSize="10" minPoolSize="2" />
```

The diagram illustrates a reference from the `connectionManagerRef` attribute in the `<jmsQueueConnectionFactory>` element to the `id` attribute of the `<connectionManager>` element. A blue arrow points from the `connectionManagerRef="ConMgr1"` box to the `id="ConMgr1"` box, with a blue box labeled "参照" (Reference) next to the arrow.

参考資料

- WASdev : JMS 2.0 Client for Message Server
 - <https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.websphere.appserver.wasJmsClient-2.0>
- WASdev : WebSphere MQ Sample using Liberty profile
 - <https://developer.ibm.com/wasdev/blog/2013/07/24/websphere-mq-sample-using-liberty-profile/>
- Knowledge center : WebSphere Application Server Liberty 16.0.0.x 「Liberty: JMS メッセージング」
 - http://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp_messaging.html
- Knowledge center : WebSphere Application Server Liberty 16.0.0.x 「Liberty へのメッセージング・アプリケーションのデプロイ」
 - http://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/twlp_dep_messaging.html
- Redbook : IBM WebSphere Application Server V8.5 Administration and Configuration Guide for Liberty Profile
 - <http://www.redbooks.ibm.com/abstracts/sg248170.html?Open>
- Bluemix でサポートされる Liberty フィーチャー
 - <https://console.ng.bluemix.net/docs/runtimes/liberty/libertyFeatures.html>
- Qiita : LibertyとActiveMQを連携する備忘録
 - <http://qiita.com/tikuase/items/23bbf026a84b7cfe4f11>
- Connect a JMS adapter to a Liberty profile server
 - http://setgetweb.com/p/MobileFirst/com.ibm.worklight.dev.doc/devref/t_connecting_jms_adapters_to_liberty.html
- WAS V9 Liberty基盤設計セミナー資料「6. 外部連携」
 - https://www.ibm.com/developerworks/jp/websphere/library/was/liberty_v9_infradesign/

END