1.1.サンプル2

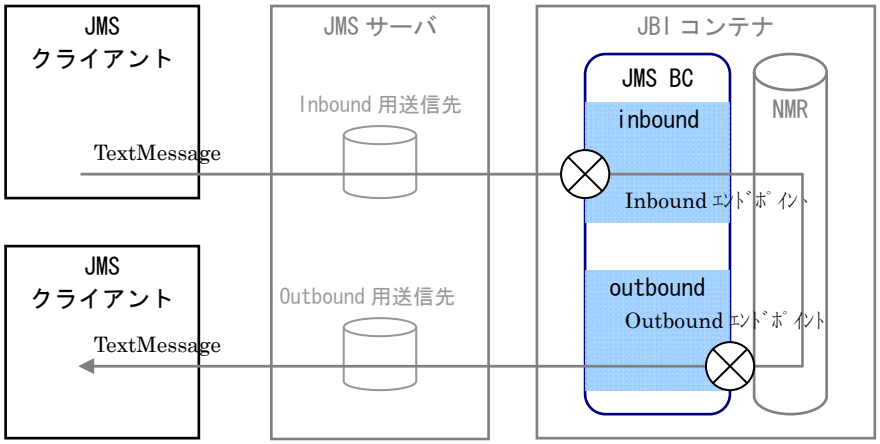
WebOTX ESB が公開するサービスと連携する JMS クライアントアプリケーションのサンプルについて説明します。

本サンプルは、単純な２つの JMS クライアント間でメッセージ送信を行うプログラムを、ESB の JMS BC を介して転送する構成にしたものです。

使用するサービスアセンブリでは、Inbound と Outbound の２つのエンドポイントを作成し、XML データの TextMessage を扱う In-Only タイプの MEP を定義しています。

そのイメージを以下に示します。

【図 1.1.1a】プログラム構成イメージ



MEMO

本サンプルでは Windows 環境でのバッチファイルのみを提供しています。Unix 環境においては、これらのバッチファイルの中で実行しているコマンドを参考にして実行してください。

1.1.1.サンプルのインストール

JMS BC のサンプルアプリケーションの zip ファイルを展開してください。

JMSBC_Sample.zip

ファイル構成

- JMS_Sample_SA.zip

  サービスアセンブリです。

- jmssample.jar

  JMS クライアントアプリケーションの jar ファイルです。

- makeenv.bat／clearenv.bat

  環境設定用(登録／削除)のバッチファイルです。

- runclient_Sender.bat／runclient_Receiver.bat

  JMS クライアントアプリケーション実行用(送信側／受信側)のバッチファイルです。

サービスアセンブリやクライアントアプリケーションの内容をカスタマイズする場合は、添付のプロジェクトファイルを WebOTX 開発環境(Developer)にインポートしてご利用ください。

- project_JMSBC_Sample_SA.zip
- project_JMSBC_Sample_AP.zip

利用の手順については「アプリケーション開発ガイド」の「５章　SOA」-「5.1. ESB」を参照してください。

本ファイルは WebOTX 開発環境(Developer)以外では利用できません。

1.1.2.実行前の作業

送信先の作成

JMS BC を利用するにあたって、利用する送信先とコネクションファクトリの準備を行う必要があります。

送信先に関しては、物理的送信先とそれに対応する JMS リソースを作成します。本サンプルを動作させるためには次のものが必要です。(名称は、JMS リソース作成時に指定する JNDI 名です。)

・ 本サンプルのエンドポイントが利用する要求用送信先と応答用送信先

jms/InOnlyInboundRequestQueue

jms/InOnlyOutboundRequestQueue

コネクションファクトリの作成

コネクションファクトリに関しては、JMS リソースを作成します。本サンプルを動作させるためには次のものが必要です。

・ 本サンプルのエンドポイントが共通に利用するコネクションファクトリ

jms/SampleQueueConnectionFactory

以下のバッチファイルを実行してください。

```
>makeenv.bat
ConnectionFactory creating ...
Command create-jms-resource executed successfully.
server.resources.connector-resource.jms/SampleQueueConnectionFactory.use-jmx-age
nt = true
Logical destination jms/InOnlyOutboundRequestQueue creating ...
Command create-jms-resource executed successfully.
Logical destination jms/InOnlyInboundRequestQueue creating ...
Command create-jms-resource executed successfully.
Physical destination InOnlyOutboundRequestQueue creating ...
Command create-jmsdest executed successfully.
Physical destination InOnlyInboundRequestQueue creating ...
Command create-jmsdest executed successfully.
```

利用するドメインのポート番号が異なる場合は、適宜バッチファイルの中身を修正してください。

1.1.3.JMS BC の起動

JMS BC が起動していることを確認してください。

1.1.4.サービスアセンブリの配備／起動

JMS BC へのサービスアセンブリの配備／起動を行います。

起動すると、JMS BC はエンドポイントを認識し、エンドポイント定義に従って JMS サーバとの接続を開始します。

サービスアセンブリの配備

以下のコマンドを実行します。

```
>jbiadmin deploy-service-assembly --user admin --password adminadmin --port 6212
```

```
JMSBC_Sample_SA.zip APG1 PG1
Deployed Service Assembly JMSBC_Sample_SA
```

（*）--port には該当ドメインのポート番号を指定してください。

Standard/Enterprise Edition の場合のみ、配備対象の ESB が動作しているアプリケーショングループとプロセスグループの指定が必要です。

サービスアセンブリの起動

以下のコマンドを実行します。

```
>jbiadmin start-service-assembly --user admin --password adminadmin --port 6212
JMSBC_Sample_SA
Started Service Assembly JMSBC_Sample_SA
```

（*）--port には該当ドメインのポート番号を指定してください。

1.1.5.JMS クライアントアプリケーションの実行

最初に受信側の JMS クライアントアプリケーションを起動します。

以下のバッチファイルを実行します。

```
>runclient_Receiver.bat
```

以下のような結果が表示されると、受信側の準備が完了です。

```
looking up jms/SampleQueueConnectionFactory ...done!!
creating QueueConnection ...done!!
creating QueueSession ...done!!
looking up jms/InOnlyOutboundRequestQueue ...done!!
InOnly(Outbound) Sample Start.
creating Receiver ...done!!
Receiver ready!!
```

次に送信側の JMS クライアントアプリケーションを起動します。

以下のバッチファイルを実行します。

```
>runclient_Sender.bat
```

以下のような結果が表示され、JMS メッセージの送信が行われます。

```
looking up jms/SampleQueueConnectionFactory ...done!!
creating QueueConnection ...done!!
creating QueueSession ...done!!
looking up jms/InOnlyInboundRequestQueue ...done!!

InOnly(Inbound) Sample Start.
creating Sender ...done!!

Sending request message : <?xml version="1.0" encoding="UTF-8"?><jbi:message
xmlns:msgns="http://www.nec.com/WebOTX/ns/jmssample" type="msgns:transformInput"
version="1.0" xmlns:jbi="http://www.nec.com/WebOTX/ns/jmssample/jbi/wsdl-11-
wrapper"><jbi:part><request>sample request</request></jbi:part></jbi:message>
InOnly(Inbound) Sample done.
```

その結果、受信側に以下のような結果が表示されると、JMS メッセージの受信が完了します。

```
Received request message : <?xml version="1.0" encoding="UTF-8"?><jbi:message
xmlns:msgns="http://www.nec.com/WebOTX/ns/jmssample" type="msgns:transformInput"
```

MEMO

JMS BC との連携時の流れについては、この実行結果とプログラムソースを比較してください。

```
version="1.0"
xmlns:jbi="http://www.nec.com/WebOTX/ns/jmssample/jbi/wsdl-11-wrapper"><jbi:part>
<request>sample request</request></jbi:part></jbi:message>
InOnly(Outbound) Sample done.
```

WebOTX のインストールディレクトリが異なる場合は、適宜バッチファイルの中身を修正してください。

1.1.6.サービスアセンブリの停止／シャットダウン／配備解除

サンプル実行後は、サービスアセンブリの停止／シャットダウン／配備解除を行います。

サービスアセンブリの停止

以下のコマンドを実行します。

```
>jbiadmin stop-service-assembly --user admin --password adminadmin --port 6212
JMSBC_Sample_SA
Stopped Service Assembly JMSBC_Sample_SA
```

（＊）--port には該当ドメインのポート番号を指定してください。

サービスアセンブリのシャットダウン

以下のコマンドを実行します。

```
>jbiadmin shut-down-service-assembly --user admin --password adminadmin --port 6212
JMSBC_Sample_SA
Shut Down Service Assembly JMSBC_Sample_SA
```

（＊）--port には該当ドメインのポート番号を指定してください。

サービスアセンブリの配備解除

以下のコマンドを実行します。

```
>jbiadmin undeploy-service-assembly --user admin --password adminadmin --port 6212
JMSBC_Sample_SA
Undeployed Service Assembly JMSBC_Sample_SA
```

（＊）--port には該当ドメインのポート番号を指定してください。

1.1.7.実行後の作業

利用した送信先とコネクションファクトリを削除します。

以下のバッチファイルを実行してください。

```
>clearenv.bat
ConnectionFactory deleting ...
Command delete-jms-resource executed successfully.
Logical destination jms/InOnlyOutboundRequestQueue deleting ...
Command delete-jms-resource executed successfully.
Logical destination jms/InOnlyInboundRequestQueue deleting ...
Command delete-jms-resource executed successfully.
Physical destination InOnlyOutboundRequestQueue deleting ...
Command delete-jmsdest executed successfully.
Physical destination InOnlyInboundRequestQueue deleting ...
Command delete-jmsdest executed successfully.
続行するには何かキーを押してください . . .
```