MQSeries® Adapter Kernel for Multiplatforms

スタートアップ・ガイド

バージョン 1 リリース 1

GC88-8887-01
(英文原典：GC34-5855-05)

MQSeries® Adapter Kernel for Multiplatforms

スタートアップ・ガイド

バージョン 1 リリース 1

GC88-8887-01
(英文原典：GC34-5855-05)

ご注意

本書の情報およびそれによってサポートされる製品を使用する前に、123ページの『特記事項』に記載する一般的情報をお読みください。

本書は、MQSeries Adapter Kernel for Multiplatforms (製品番号 5648-D75) バージョン 1 リリース 1、モディフィケーション・レベル 1 および新版において、特に断りのない限り、それ以降のすべてのリリースおよびモディフィケーションにも適用されます。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

原　典： GC34-5855-05
MQSeries® Adapter Kernel for Multiplatforms
Quick Beginnings
Version 1 Release 1

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2001.5

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この (書体*) は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

図

表

MQSeries Adapter Kernel スタートアップ・ガイドの紹介

本書では MQSeries® Adapter Kernel について解説し、その計画、インストール、および使用の方法を説明します。

MQSeries Adapter Kernel の使用に際しては、以下に示す一般的なステップに従ってください。

1. 1ページの『第1章 MQSeries Adapter Offering について』 を読みます。
2. インストールの準備をします。詳細は、40ページの『インストールの準備』を参照してください。
3. カーネルをインストールします。詳細は、41ページの『カーネルのインストール』を参照してください。
4. インストールを確認します。詳細は、46ページの『インストールの検査』を参照してください。
5. カーネルを構成します。詳細は、58ページの『カーネルの構成』を参照してください。
6. 必要であれば、カーネルとともに使用するオプション・ソフトウェアを構成します。詳細は、89ページの『MQSeries および MQSeries Integrator の構成』を参照してください。
7. MQSeries Adapter Builder を使用してアダプターを構築し、テストして配置します。詳細は、MQSeries Adapter Builder の資料を参照してください。
8. カーネルを起動します。詳細は、90ページの『カーネルの始動』を参照してください。

これらを行うためには、前提条件とオプション製品についての理解も必要になります。31ページの『第2章 カーネルのインストール計画』を参照してください。また、97ページの『参考資料』も参照してください。

本書の対象読者

本書は、MQSeries Adapter Kernel の計画、インストール、または使用を必要とするユーザーを対象としています。

関連情報

追加情報として次を参照してください。

- `readme.txt` ファイル。このファイルには、本書の完成後に追加された情報が含まれていることがあります。 `readme.txt` ファイルは、インストールを行う前は、製品 CD-ROM のルート・ディレクトリーに置かれています。インストール後の `readme.txt` ファイルは、インストールした MQSeries Adapter Kernel のルート・ディレクトリーに置かれます。
- *問題判別ガイド* (資料番号 GC88-8888)。この資料には、 MQSeries Adapter Kernel 固有の問題を解決するための、トレースなどのツールについて説明されています。「*問題判別ガイド*」は、製品とともにインストールされる MQSeries Adapter Kernel Information Center から参照できます。
- MQSeries Adapter Kernel Information Center にある、アプリケーション・プログラミング・インターフェース (API) のオンライン資料。この情報は、カーネル機能の動作を理解するため、および診断を補助するためだけに提供されています。95ページの『第5章 MQSeries Adapter Kernel API の使用』を参照してください。
- MQSeries Adapter Builder 情報 (資料およびヘルプ・システムを含む)。
- MQSeries 製品ファミリーの Web サイト、 http://www.ibm.com/software/ts/mqseries/。

  この Web サイトのリンクをたどれば、次のことを行うことができます。

  – MQSeries Adapter Offering など、MQSeries 製品ファミリーについての最新情報を入手できる。

  – MQSeries の資料 (HTML および PDF フォーマット) にアクセスできる。本書の新しい版が含まれている場合もあります。

  – MQSeries SupportPacs をダウンロードできる。

表記規則

MQSeries Adapter Kernel の資料では、以下のような書体の規則およびキーの規則を使用しています。

表 1. 本書の規則

規則	意味
太字	コマンド名を示します。グラフィカル・ユーザー・インターフェース (GUI) に関しては、メニュー、メニュー項目、ラベル、およびボタンも示します。
モノスペース	コマンド・プロンプトで入力する必要のあるテキスト、およびファイル名、パス、ならびに関数、クラス、メソッドのようなプログラム言語のエレメントなど、示されているとおりに使用しなければならない値を示します。モノスペースは画面テキストおよびコード例を示すためにも使用されます。
イタリック	指定する必要のある可変値を示します (例: *fileName* にファイルの名前を指定します)。イタリックは、強調表示および書名の表示にも使用されます。
%	root 権限を必要としないコマンドの UNIX コマンド・シェル・プロンプトを示します。
#	root 権限が必要なコマンドの UNIX コマンド・シェル・プロンプトを示します。
C:¥>	Windows® システムのコマンド・プロンプトを示します。
>	メニューの記述に使用している場合は、一連のメニューの選択を示します。たとえば、「**「File (ファイル)」>「New (新規)」**をクリックします」とある場合は、「**「File (ファイル)」**メニューから**「New (新規)」**コマンドをクリックする」という意味になります。
コマンド入力	コマンドを「入力」する、または「発行」すると指示された場合は、コマンドを入力してから Enter を押します。たとえば、「**ls** コマンドを入力してください」という指示は、コマンド・プロンプトに **ls** とタイプしてから Enter を押すという意味になります。
[ ]	構文記述内のオプション項目を囲みます。
{ }	構文記述内の、項目を選択する必要があるリストを囲みます。
|	構文記述の中で、中括弧 (**{ }**) で囲まれた選択項目リストにある項目を区切るために使用されます。
...	構文記述内の省略記号は、前の項目を 1 回以上繰り返すことができることを示します。例にある省略記号は、簡潔にするために例から情報が省略されていることを示します。

注: カーネル・ソフトウェアおよび本書において、 Epic という語が、いくつかの値や名前として出現します。 MQSeries Adapter Offering に関しては、この語自体に意味はありません。

変更の要約

第 6 版 (現行の版) では、第 5 版から次の変更が加えられています。

- ランタイム・フローの説明の更新。いくつかの変更点が反映されています。 14ページの『ランタイム・フロー』を参照してください。
- MQSeries Adapter Kernel を WebSphere® Business Integrator とともに使用する場合の情報。詳細は、28ページの『MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server とともに使用』を参照してください。
- さまざまなアダプターと共に提供される各国語サポート (NLS) のレベルに関する情報。詳細は、30ページの『各国語サポート』を参照してください。
- インストール手順の説明。41ページの『カーネルのインストール』を参照してください。
- サイレント・インストールについての情報。詳細は、51ページの『サイレント・インストールの使用法』を参照してください。
- カーネルを構成するのに役立つ構成の概要。詳細は、58ページの『構成の概要』を参照してください。
- 新しいヘッダー値についての情報。詳細は、107ページの『MQSeries Adapter Kernel メッセージ・ディスクリプター・ヘッダー』を参照してください。

第 4 版から第 5 版での変更は次のとおりです。

- Windows® 2000、OS/400®、HP-UX、および Solaris プラットフォームでのカーネルの使用についての情報。これらのプラットフォームでのサポートは、 MQSeries Adapter Kernel バージョン 1.1 から新しく採用されました。これより前は、カーネルは Windows NT® および AIX® でしか使用できませんでした。
- MQSeries AdapterKernel バージョン 1.1 を考慮した、インストール手順の説明の全面的な更新。
- `aqmconfig.xml` ファイルを使用して MQSeries Adapter Kernel を構成するための情報。これまでカーネルは、 `aqmconfig.properties` ファイルを使用して構成されていました。詳細は、64ページの『構成ファイル』を参照してください。
- 新しい MQ および JMS (Java Message Service) 通信モードについての情報。詳細は、99ページの『付録A. 通信モード』を参照してください。
- トレースについての情報は、本書から新規資料である「*問題判別ガイド*」に移行しました。詳細は、「*問題判別ガイド*」を参照してください。

第1章 MQSeries Adapter Offering について

IBM MQSeries Adapter Kernel は、まとめて IBM MQSeries Adapter Offering と呼ばれる、アプリケーション統合製品群の一部です。IBM MQSeries Adapter Offering は、MQSeries のメッセージ・サービスおよび他のメッセージ・サービスとともに機能して、ビジネス・プロセスの Point-to-Point 統合の管理のリスク、複雑さ、およびコストを低減します。

Point-to-Point 統合では、各アプリケーションは他のアプリケーションそれぞれと個々にインターフェースを持ちます。各インターフェースは異なっており、そして多くのさまざまなインターフェースがあります。あるアプリケーションに変更があると、通常は多くのインターフェースの変更が必要になります。そして、アプリケーションの数が増加すると、Point-to-Point 統合のコストは急速に増加します。通常は、新しいアプリケーションを統合するごとに、アプリケーションを統合する前よりも大きな作業負荷が必要になってしまいます。

MQSeries Adapter Offering を使用すると、Point-to-Point 統合の利用から、 *One-to-Any* 統合の利用へと進展することができます。 One-to-Any 統合には次のようなものを始め多くの利点があります。

- すべてのアプリケーションが 1 つの共通インターフェースを使うことができます。
- ソース・アプリケーション のデータは、メッセージ の形で、複数の ターゲット・アプリケーション にルーティング されます。
- あるアプリケーションに変更があっても、通常は、1 つのインターフェースに影響が出るだけです。
- どのアプリケーションにも依存しない共通インターフェース (たとえば XML のような業界標準) を使用することは、それ以外のコスト上の効果もあります。より多くのアプリケーションを、少ない労力でサポートすることができます。
- アプリケーションの数が増加するほど、One-to-Any 統合のコスト上の効果も大きくなります。通常は、新しいアプリケーションを追加するごとに、他のすべてのアプリケーションのインターフェースに大きな変更が必要になることはありません。
- 統合作業を自動化することができ、また、テンプレートに基づいて行うことができます。

MQSeries Adapter Offering は、アプリケーションやビジネス・プロセスをまったく変更することなく展開することができます。統合作業は通常、MQSeries Adapter Offering で行うので、書く必要があるカスタム・コードは減少します。

MQSeries Adapter Offering では、あるアプリケーションとのインターフェースは、アダプター によって提供されます。アプリケーション環境とメッセージ環境の間のインター

フェースを装備するには、すべてのアプリケーションに最低 1 つのアダプターが必要です。アダプターはそれぞれアプリケーションおよびメッセージ・タイプに固有のものです。

MQSeries Adapter Kernel を、オプションの MQSeries Integrator とともに展開すると、ブローカリングおよびメッセージ変換を行うことができます。MQSeries Adapter Offering の補足分は、IBM などのサービス・オファリングによって供給されます。

アダプターの使用例として次のものがあります。

- 販売オーダーの追加
- 顧客記録の同期化
- 在庫記録の同期化
- 品目の同期化
- 販売オーダーの同期化

ビルド・タイムおよびランタイム

MQSeries Adapter Offering は、 Adapter Builder (ビルダーとも呼びます) と Adapter Kernel (カーネルとも呼びます) の、2 つの主なコンポーネントで構成されています。このセクションでは、それらのコンポーネントと、Adapter Offering で構築および実行されるアダプターについて説明します。

アダプター

アプリケーションとのインターフェースを備えているソフトウェア。アダプターは、MQSeries Adapter Builder を使用して作成します。通常、アダプターはそれぞれ、アプリケーションとの間で送信される 1 メッセージ・タイプに固有のものとして作成されます。アダプターそのものは MQSeries Adapter Offering のパーツではありません。

アダプターは共用ライブラリーにコンパイルされる C または Java™ ソース・コードから成ります。アダプターと MQSeries Adapter Kernel が一緒に実行されると、それらは MQSeries Adapter Offering のランタイム機能を実行します。

アダプターの MQSeries Adapter Builder でのモデル化の方法にもよりますが、アダプターは次のようなさまざまな機能を持つことができます。制御フロー、データ・フロー、順次ナビゲーション、判別と反復を含む条件付き分岐、データ・タイプ化、データ・コンテキストの記憶、データ・エレメントの変換、トランザクション制御、論理演算、およびカスタム・コード化。

アダプターは再利用できます。

アダプターには次の 2 つの主なタイプがあります。

- ソース・アダプター (データを送信するアプリケーション用)

- ターゲット・アダプター (データを受信するアプリケーション用)

あるアプリケーションから、あるタイプのメッセージを 2 番目のアプリケーションに送る場合は、通常は、1 つのソース・アダプターと 1 つのターゲット・アダプターが必要です。その 2 番目のアプリケーションが、あるタイプのメッセージを最初のアプリケーションに送る必要がある場合は、別にソース・アダプターとターゲット・アダプターが必要です。この場合のように、最初のアプリケーションから 2 番目のアプリケーションにあるタイプのメッセージを送り、それから 2 番目のアプリケーションから最初のアプリケーションに別のタイプのメッセージを送り返すには、通常は 4 つのアダプターを配置します。

メッセージ・タイプごとに別々のアダプターが必要です。

IBM WebSphere Application Server とエンタープライズ bean がカーネルのターゲット・サイドで使用される場合、3 つ目のタイプのアダプターである、Java サービス・セッション bean アダプターが使用されます。WebSphere Application Server は、Sun Microsystems の Enterprise JavaBeans (EJB) 仕様を実装しているため、Java サービス・セッション bean アダプターおよび他のエンタープライズ bean を使用できます。詳細は、28ページの『MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server とともに使用』および MQSeries Adapter Builder の資料を参照してください。

MQSeries Adapter Builder

グラフィカル・ユーザー・インターフェース (GUI) を使用して、事実上どんなアプリケーションにもアダプターを作成することができます。ユーザー・インターフェースは MQSeries Integrator のユーザー・インターフェースと類似しています。詳細は、MQSeries Adapter Builder Information Center を参照してください。

MQSeries Adapter Kernel

アプリケーション・プログラミング・インターフェース (API)、C と Java™ で書かれたいくつかの実行可能プログラム、およびいくつかの構成ファイルのセットです。カーネルによって、アダプターの配置と実行が可能になります。カーネルはアダプターを直接サポートすることの外に、メッセージの単純ルーティングなどの関連する機能を実行します。さらに、メッセージの組み立て、トランザクションの制御、トレース、MQSeries や他のメッセージング・ソフトウェアとのインターフェースなどの基盤サービスを実行します。

カーネルは、ソース・アダプターまたはターゲット・アダプターが実行されるコンピューターごとにインストールします。

MQSeries Adapter Offering を使用することによって、ビジネス・プロセスと各アプリケーションは、ミドルウェアの特性、メッセージの詳細、および他のアプリケーションか

ら独立性を維持することができます。メッセージの共通インターフェースによって、既存のアプリケーションやビジネス・プロセスを変更することなく新規のアプリケーションを追加することができます。

MQSeries Adapter Kernel は 2 層に配置することができます。 1 つの層はランタイムのソース・サイド、もう 1 つの層はランタイムのターゲット・サイドです。 2 層配置によって、効率的なオペレーションと少ない管理オーバーヘッドを実現することができます。ルーティングおよび送達のための 3 つ目の層を、ランタイムの両サイド間に常駐させる必要はありません。ただし、複合ルーティング、データ形式変更、およびデータの調停などのブローカリングを実行するために、MQSeries Integrator をオプションで追加することができます。

特別な部分を除いて、本書ではこれ以降 MQSeries Adapter Kernel に関して述べます。MQSeries Adapter Builder についての詳細な情報は、製品の Information Center を参照してください。

カーネルについて

最も簡単に言えば、ランタイム、つまりカーネルおよび作成するアダプターには次の目的があります。

1. ソース・アプリケーションからターゲット・アプリケーションにデータを転送すること。
2. ソース・アプリケーションのデータを、MQSeries または他のメッセージング・ソフトウェアを使って、通常はアプリケーションに依存しないフォーマットのメッセージに変換すること (そのメッセージはカーネルによって経路指定される)。
3. メッセージの、ターゲット・アプリケーションへの経路を指定すること。
4. ターゲット・アプリケーションに行くデータの取得方法を判別すること。
5. データを、アダプターを通りカーネルによって経路指定されるメッセージのフォーマットから、ターゲット・アプリケーションのフォーマットに変換すること。

このセクションでは、カーネルの機能性の概要を述べます。カーネルの機能性のより詳細な記述は、14ページの『ランタイム・フロー』にあります。

カーネルには次の 2 つのサイドがあります。

- ソース・サイド : ソース・アプリケーションからメッセージを受信するときに始まり、そのメッセージがメッセージ・キューに置かれるときに終了します。
- ターゲット・サイド : メッセージ・キューからメッセージが検索されるときに始まり、そのメッセージがターゲットに送信されるときに終了します。

通常これらのサイドはそれぞれ別々のコンピューターに常駐しますが、両方のサイドが同一コンピューターに常駐することが可能です。

6ページの図1 を参照してください。そこでのシーケンスを説明します。

カーネルのソース・サイド

1. カーネルのソース・サイドでは、ソース・アプリケーションが、ソース・アプリケーションのフォーマット のデータを、*アプリケーション固有のインターフェース* を使って、MQSeries Adapter Builder で作成されたソース・アダプターに送信します。メッセージ・タイプ (たとえば「販売オーダーの追加」や「顧客記録の同期化」) ごとに別のソース・アダプターが必要です。

   アプリケーション固有のインターフェースの開発は MQSeries Adapter Offering の範囲外です。アプリケーション固有のインターフェースそのものの実態は、ソース・アプリケーションやターゲット・アプリケーションの特性に依存します。たとえば例として、API 呼び出しとユーザー出口、ファイルの読み取りと書き込み、データベース・トリガー、およびメッセージ・キューなどがあります。

   ソース・アダプターは、ソース・アプリケーションのプロセスで実行されるので注意してください。ソース・アダプターを機能させるには、ソース・アダプターを含んでいるデーモンまたはサーバーを始動する必要があります。

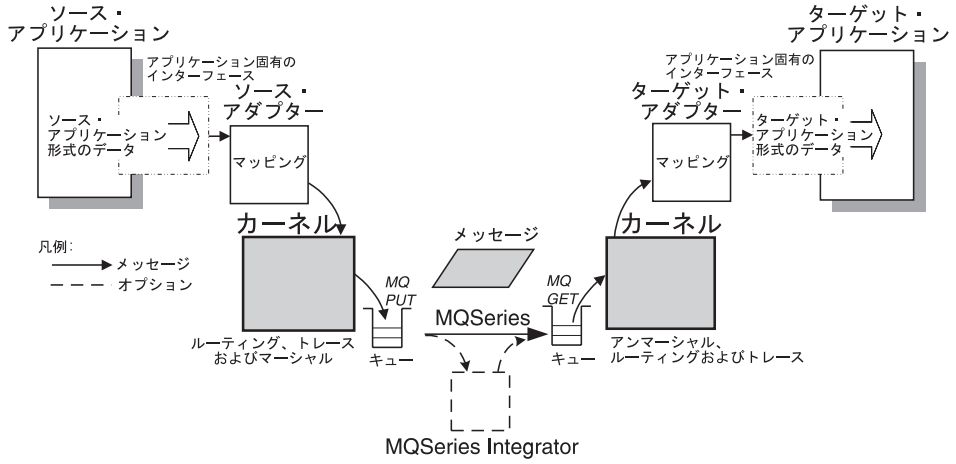
2. ソース・アダプターは、その作成仕様に従って機能を実行します。代表的な機能はデータ・エレメントの変換です。つまり、ソース・アプリケーションのフォーマットのエレメントを、本体データとして統合メッセージのフォーマット にマッピングすることです。本体データおよび制御値を示す付加的なメタデータは、カーネルの メッセージ・ホルダー・オブジェクト に置かれます。

3. ソース・アダプターが、*ネイティブ・アダプター* を使用してカーネルにメッセージ・ホルダー・オブジェクトを渡す場合、メッセージ・ホルダー・オブジェクトの制御値 (*メッセージ制御値*) がカーネルによって使用され、メッセージ・ホルダー・オブジェクトの通信メッセージ・フォーマットへのマーシャル、および通信メッセージのルーティングを制御します。

   メッセージに特定のメッセージ制御値が含まれていない場合は、カーネルは構成ファイルから得られるデフォルト値またはメッセージ制御値を使用することができます。メッセージ制御値の定義については、16ページの『メッセージ制御値』を参照してください。

4. カーネルは、メッセージ・マーシャル、単純な*ルーティング*、およびオプションで*トレース* などの機能を実行します。 12ページの『メッセージおよびメッセージ・フォーマット』、13ページの『ルーティングおよび送達』、および 28ページの『トレース』を参照してください。

図 1. MQSeries Adapter Offering の概要.



カーネルのソース・サイドからターゲット・サイドへの送達

5. カーネルは、そのネイティブ・アダプターを使用して メッセージを適切なメッセージ・キューに書き込みます。

   ソース・サイドで行われる送信には、次の 2 つのメソッドがあります。

   - `sendMsg`: メッセージを送信して即時に戻ります。 `sendMsg` メソッドは `begin`、`commit`、および `rollback` メソッドとともに使用して、 *transactionally* というメッセージを送信することもできます。これは、他のオペレーションが正常に完了したとき (だけ)、メッセージが送信されるということです。詳細は、27ページの『トランザクション機能』を参照してください。
   - `sendRequestResponse`: メッセージを送信して応答を待ちます。
     `sendRequestResponse` メソッドは、トランザクション的には使用できません。

     なお、`sendResponse` という 3 つ目のメソッドがありますが、これは送信側が応答を要求したときにカーネルのターゲット・サイドで使用されるものです。

   MQSeries または他のメッセージング・ソフトウェアはメッセージのトランスポートを行います。 8ページの『MQSeries または他のメッセージング・ソフトウェアの役割』を参照してください。なお、MQSeries Adapter Offering をサポートするには、メッセージング・ソフトウェアが構成済みであることが必要ですので注意してください。

   MQSeries Integrator が宛先としてのカーネルに構成されていると、 MQSeries Integrator はブローカリング機能を実行することができます
   (オプション)。 8ページの『MQSeries Integrator の役割』を参照してください。最終

宛先であるメッセージ・キューが MQSeries Integrator の規則 (メッセージ・フロー) で構成されている場合、 MQSeries Integrator はメッセージ・キューにメッセージを送信します。

メッセージは、適切なメッセージ・キューに送達されます。

カーネルのターゲット・サイド

6. カーネルのターゲット・サイドでは、ランタイムとターゲット・アプリケーションの間のインターフェースとして、 2 つの*送達モデル* が用意されています。
   - 最も一般的なモデルは *push* です。この場合、カーネルはターゲット・アプリケーションへのメッセージの送達の開始と管理を行います。 push モデルでは通常、 MQSeries Adapter Offering をサポートするためのターゲット・アプリケーションの変更は必要ありません。
   - *pull* モデルでは、ターゲット・アプリケーションがメッセージの受信の管理を行います。 pull モデルでは、 MQSeries Adapter Offering をサポートするためにターゲット・アプリケーションを変更する必要があります。ターゲット・アプリケーションが、カーネルのターゲット・アプリケーションへのインターフェースを管理しなければなりません。

   push モデルでのターゲット・サイドでは、メッセージの取得と送達のために、ユーザーはあらかじめカーネルのプロセスを始動しておく必要がありますので注意してください。90ページの『カーネルの始動』を参照してください。

   push モデルでは、カーネルはメッセージ・キューからメッセージを取得します。トレースが可能になっていれば、カーネルはトレースを行います。カーネルはメッセージのルーティングを継続的に行い、適切なターゲット・アダプターを選択します。一般的に、メッセージ・タイプごとに別々のターゲット・アダプターが必要です。
7. カーネルは、メッセージを適切なターゲット・アダプターに送達します。ターゲット・アダプターは、そこに作成された機能を実行します。代表的な機能は、統合メッセージのフォーマットのエレメントを、ターゲット・アプリケーションのフォーマット にマッピングすることです。

   ターゲット・アダプターは、MQSeries Adapter Kernel アダプター・デーモンまたは WebSphere Application Server のいずれかによりホスト可能です。この説明は、28ページの『MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server とともに使用』を参照してください。
8. ターゲット・アダプターは、MQSeries Adapter Offering の範囲外で開発されたアプリケーション固有のインターフェースを使用して、ターゲット・アプリケーションのフォーマットで、データをターゲット・アプリケーションに送信します。
9. ターゲット・アダプターがそのメッセージを送達したとき、そのメッセージはメッセージ・キューからコミットされます。これによって、キューからメッセージが除去されます。

10. ソース・アダプターによってメッセージ制御値が設定され肯定応答を要求している場合、カーネルはメッセージ送達の肯定応答またはターゲット・アダプター出力のいずれかを、sendResponse メソッドを使ってソース・アダプターに送ります。
11. エラーの場合は、カーネルはオリジナル・メッセージをエラー・キューに書き込みます。カーネルがオリジナル・メッセージをエラー・キューに書き込めないときは、コミットは起きません。

MQSeries または他のメッセージング・ソフトウェアの役割

MQSeries Adapter Offering の通信メッセージは、メッセージ・キュー上でトランスポートが行われます。メッセージ・キューは、 MQSeries または Java Message Service (JMS) のようなメッセージング・ソフトウェアが備えています。MQSeries Adapter Offering によってトランスポートされるメッセージは、次のタイプのキューを使用します。

- *受信*キュー : MQSeries Adapter Offering の用語です。これは、メッセージを受信するための主な入力キューとして使用されます。ターゲット・アプリケーションごとに複数の受信キューがあることがあります。
- エラー・キュー : MQSeries Adapter Offering の用語です。これは受信キューから取得したメッセージを処理できないときに使用されます。
- 応答キュー (オプション)。これは sendRequestResponse メソッドとともに使用されます。

MQSeries Adapter Offering は、MQSeries の次の特定のメッセージ・タイプの機能を使用します。

- データグラム: sendMsg メソッドで使用されます。
- 要求: sendRequestResponse メソッドで使用されます。
- 応答: sendRequestResponse メソッドおよび sendResponse メソッドで使用されます。

MQSeries は、オプションで、アプリケーション固有のインターフェースとして作用することができます。

MQSeries および MQSeries Adapter Offering の妥当性検査済みの構成のリストは、105ページの『付録B. 妥当性検査済みの構成』を参照してください。MQSeries および他のソフトウェアのサポートしているバージョンのリストは、32ページの『ソフトウェア』を参照してください。

MQSeries Integrator の役割

MQSeries Integrator は、MQSeries Adapter Kernel とともに配置することができます (オプション)。これを使用して、ブローカリングについて可能性がある次のいくつかの要求に対応することができます。

- 複合ルーティング、つまりメッセージ・ヘッダーまたはメッセージ本体の内容に基づく経路指定です。メッセージ本体の内容が変わると、ルーティングは動的に変わるこ

とができます。複合ルーティングおよび単純ルーティングについての情報は、13ページの『ルーティングおよび送達』を参照してください。

- データ形式変更、つまり別の文書タイプに変更することです。
- データ調停、つまりメッセージ本体の内容を変更することです。たとえば、ソース・アプリケーションが、あるフィールドの `each` という値を示していても、ターゲット・アプリケーションが扱えるそのフィールドの値は `ea` の場合、データ調停によって元の値は扱える値に置き換えられます。

MQSeries Integrator を使用して、ユーザーのサイトのルーティングのほとんどを行うことができます。また、MQSeries Adapter Kernel のルーティング機能の使用を減少させることもできます。

MQSeries Integrator および MQSeries Adapter Offering の妥当性検査済みの構成のリストは、105ページの『付録B. 妥当性検査済みの構成』を参照してください。MQSeries Integrator および他のソフトウェアのサポートしているバージョンのリストは、32ページの『ソフトウェア』を参照してください。

カーネルの動作

このセッションで次の項目について述べます。

- 『カーネル・ランタイムのコンポーネント』
- 12ページの『メッセージおよびメッセージ・フォーマット』
- 13ページの『ルーティングおよび送達』
- 14ページの『ランタイム・フロー』

カーネル・ランタイムのコンポーネント

作成したアダプター、開発したカスタム・コード、および MQSeries Adapter Kernel が一緒に実行される場合に、 MQSeries Adapter Offering の機能が提供されます。

カーネル・ランタイムの主要コンポーネントは次のとおりです。

ソース・アダプター
特定のアプリケーション向けに (通常は MQSeries Adapter Builder を使用して) 作成されたソフトウェアで、そのアプリケーションのデータを統合メッセージ・フォーマット (本体データ) に変換します。ソース・アダプターは通常、ソース・アプリケーションと同じマシン上で、アプリケーションのプロセス内で、または別のプロセスとして実行されます。ソース・データには、たとえば、ファイル、C 構造体、および Java オブジェクトがあります。統合メッセージ・フォーマットは、通常は OAG や RosettaNet のような業界標準の結果としてできたもので、その例として XML があります。

メッセージ・ホルダー

カーネルによって使用されるメタデータのコンテナーで、カーネルによって使用される統合メッセージおよび他の制御データをカプセル化します。メタデータの例として、ソース・アプリケーションおよびターゲット・アプリケーションのアプリケーション ID (論理 ID)、メッセージのカテゴリー (たとえば OAG)、メッセージのタイプ (たとえば「購入オーダー」)、および送信または受信される通信メッセージ (本体データ) があります。

ネイティブ・アダプター

メッセージ・ホルダー・オブジェクトの送信および受信に使用されるソフトウェアです。メッセージの送信時にネイティブ・アダプターは、単純データ・ルーティング、および 1 つまたは複数の通信トランスポート機構のサポート機能を提供します。単純データ・ルーティングは、メッセージのカテゴリーやメッセージのタイプのような、メッセージ・ホルダー・オブジェクトのメタデータに基づいて行われます。メッセージは非同期で、または同期で送信されます。基本になる通信トランスポート機構がトランザクションのメッセージをサポートする場合、メッセージは単一フェーズ・トランザクション制御のもとで送信されます。トランザクションのサポートは使用するトランスポート機構の機能に限定されます。メッセージ・ホルダー・オブジェクトの場合、トランスポート機構 によって使用される通信メッセージ・フォーマットへのマーシャルが行われます。通信メッセージが受信されると、ネイティブ・アダプターは、そのメッセージに対してメッセージ・ホルダー・オブジェクトへのアンマーシャルを行います。

アダプター・デーモン

アダプター・ワーカーをインスタンス化するプロセスです。アダプター・デーモンは一度始動すると、アクティブ状態を維持し続けます。各ターゲット・アプリケーションのアプリケーション受信キューごとに、 1 つのアダプター・デーモンが存在します。

アダプター・ワーカー

各メッセージを、適切なターゲット・アダプターに送達するプロセスです。各ワーカーはそれぞれ 1 つのネイティブ・アダプターを管理します。アダプター・デーモンがワーカーを生成し、始動します。

ワーカーが複数あるので、ターゲット・アダプターへのマルチスレッドのメッセージ送達が可能になっています。各ワーカーは自分のネイティブ・アダプターとともに、1 つのスレッドを扱います。ワーカーが 1 つだけの場合は、ターゲット・アダプターへの (その結果ターゲット・アプリケーションへの) メッセージの送達は、単一スレッドということです。

ワーカーは、ネイティブ・アダプターの管理の外に、次のタスクも行います。

- トレースが可能になっていれば、トレース・クライアントをインスタンス化します。

- 各ターゲット・アプリケーションごとに適切なログオン・クラスをインスタンス化します。
- メッセージの本体タイプと本体カテゴリーに基づいて、ターゲット・アダプターを選択します。
- 選択したターゲット・アダプターにメッセージを送信します。
- コミットを行えないときは、ロールバックを行い、自分のアダプター・デーモンのもとの他のすべてのワーカーのためにフラグを立てて、自分自身と自分のネイティブ・アダプターをシャットダウンします。このことは、そのときのメッセージに問題があることを意味します。すべてのワーカーをシャットダウンすることによって、他のワーカーが、問題がある同じメッセージを再処理して同じ結果になることを防ぎます。
- 別のワーカーによって立てられたシャットダウンのフラグを認識すると、自分自身と自分のネイティブ・アダプターをシャットダウンします。

ターゲット・アダプター

特定のアプリケーション向けに (通常は MQSeries Adapter Builder を使用して) 作成されたソフトウェアで、統合メッセージ・フォーマット (本体データ) のデータを、ターゲット・アプリケーションが必要とするネイティブ・データ・タイプに変換します。ターゲット・アダプターは、ターゲット・アプリケーションの必要な API を呼び出してメッセージを送達します。ターゲット・アダプターは、アプリケーションまたはアプリケーション・クライアントと同一マシン上で実行します。

Java サービス・セッション bean アダプター

WebSphere Application Server のような EJB サーバーでホストされる Java 言語 EJB アダプターの 1 タイプです。

構成コンポーネント

論理 ID からキュー名のようなオブジェクトを決めるために使用されるデータです。構成データは、ファイルまたは WebSphere Business Integrator 製品の LDAP 構成のいずれかで指定することができます。データはカーネル構成の次の側面を制御します。

- メッセージのマーシャルとルーティング
- インストールの検査
- 通信モード
- トレース

構成ファイルの詳述は、64ページの『構成ファイル』 を参照してください。 WebSphere Business Integrator をカーネルと動作させるように構成する場合は、WebSphere Business Integrator の資料を参照してください。

トレース・コンポーネント

トレース・メッセージを書き込むソフトウェアです。カーネルのコンポーネントの大部分はトレース・コンポーネントを使用します。トレースの概要については、28ページの『トレース』を、トレースについての詳細は「*問題判別ガイド*」を参照してください。

メッセージおよびメッセージ・フォーマット

MQSeries および MQSeries Adapter Offering では、メッセージ は、あるプログラムによって送信され、別のプログラムに与えられるデータの集合です。メッセージのある時点のフォーマットは、その時点のメッセージ・フローの中の、そのメッセージの位置に依存します。MQSeries Adapter Kernel では、次の 3 つのメッセージ・タイプを使用します。

- *統合メッセージ* - ソース・アプリケーションのデータを、ターゲット・アプリケーションに送信するために、 XML のような別のアプリケーションのフォーマットに変換したデータから成るメッセージ。統合メッセージは、メッセージの本体データとしてメッセージ・ホルダー・オブジェクトに挿入されます。データ表記の標準として XML があります。フォーマットが XML の場合、フォーマットは *Document Type Definition* (DTD) で定義されています。 DTD は、文書 (この場合はメッセージ本体) の形式の定義を含む 1 つまたは複数のファイルです。強くお勧めしますが、メッセージ本体はアプリケーション非依存のフォーマットである必要はありません。メッセージ本体のフォーマットを専用のものにしたり、そうでなくとも特殊なものにしたりすることはできますが、そのタイプのフォーマットはお勧めできません。

  MQSeries Adapter Offering では、その統合メッセージの中のメッセージ本体を定義するために、*Business Object Documents* (BOD) を使用することができます。 BOD は、組織内または組織間を流れる標準的なビジネス・プロセスを表現したものです。その中の例として、「購入オーダーの追加」、「製品の可用性の説明」、および「販売オーダーの追加」があります。BOD は、Open Applications Group (OAG) によって XML で定義されています。 BOD の使用をお勧めしますが、必須ということではありません。

- *メッセージ・ホルダー・オブジェクト* - 統合メッセージと、 MQSeries Adapter Kernel に固有の制御値を表す、付加されたヘッダー・メタデータを含むオブジェクト。ソース・アダプターはメッセージ・ホルダー・オブジェクトを作成し、適切な制御情報を設定し、そして送信される統合メッセージがあるときは本体データをセットします。ターゲット・アダプターはメッセージ・ホルダー・オブジェクトを受信し、本体データを取得し、そしてその本体データをターゲット・アプリケーション固有のデータに変換します。ソース・アダプターとターゲット・アダプターは、MQSeries Adapter Builder を使用して作成します。
- *通信メッセージ* - 通信トランスポート固有の情報にメッセージ・ホルダー・オブジェクトが加わったもので、使用される通信トランスポート固有のメッセージ・フォーマットに変換される。複数のメッセージ・フォーマットをサポートする通信トランスポートもあります。通常、通信メッセージと結合したカーネルのヘッダー・メタデータ

値は、通信トランスポートでは、アプリケーション・データとして見なされます。詳細は、99ページの『付録A. 通信モード』を参照してください。たとえば MQSeries のトランスポートは、 MQSeries 固有のメッセージ・ヘッダーに、マーシャルが行われたメッセージ・ホルダー・オブジェクトが加わったものから成ります。MQSeries 固有のフォーマットには次のものが含まれます。

– MQSeries によって追加される MQSeries メッセージ・ヘッダー
– MQSeries Integrator が使用されている場合は、バージョン固有の次のメッセージ・ヘッダー
  - MQSeries Integrator version 1 というメッセージ・ヘッダー (MQSeries Integrator バージョン 1.1 を使用の場合)
  - MQSeries Integrator version 2 というメッセージ・ヘッダー (MQSeries Integrator バージョン 2 を使用の場合)
– 制御値を表す、カーネル固有のヘッダー・メタデータ
– 統合メッセージ (本体データ)

MQSeries Adapter Offering のメッセージ・ヘッダーで使用される関係フィールドとそれらの説明のリストは、 107ページの『付録C. メッセージ・ヘッダー』を参照してください。

ルーティングおよび送達

カーネルは、ソース・アダプターのメッセージごとに経路を定めて、それを適切なターゲット・アダプターに送達します。ルーティングは次の 2 段階で行われます。

1. ソース・サイドのカーネルが、メッセージを適切なメッセージ・キューに書き込みます。
2. ターゲット・サイドのカーネルがメッセージ・キューからメッセージを取得し、適切なターゲット・アダプターを呼び出します。

ルーティングは、いくつかの要因によって決まります。

- メッセージ・キュー。最も基本的なレベルとして、メッセージ・キューは、MQSeries Adapter Offering のルーティングをサポートするように構成されていなければなりません。
- メッセージの中のメッセージ制御値。メッセージ制御値には、ソースの論理 ID、宛先の論理 ID、論理 ID への応答、本体カテゴリー、本体タイプ、トランザクション ID、メッセージ ID、肯定応答要求、およびタイム・スタンプがあります。詳細は、16ページの『メッセージ制御値』を参照してください。メッセージの中の宛先論理 ID は、カーネルの構成ファイルをオーバーライドすることができます。各メッセージのヘッダーの、これらのメッセージ制御値が変わると、ルーティングは動的に変化することがあります。ただし、メッセージの本体データ (統合メッセージ) の内容では、ルーティングは決まりません。

- カーネルの構成ファイルの中のメッセージ制御値。カーネルの構成ファイルでは、宛先論理 ID、キュー名、および関連するターゲット・アダプターを指定することができます。このファイルを編集して、構成の決定および変更を行います。追加情報は、64ページの『構成ファイル』を参照してください。
- オプションで、MQSeries Integrator を使用して、複合ルーティングなどメッセージのブローカーを行うことができます。メッセージ本体の内容が変わると、ルーティングは動的に変わることがあります。 8ページの『MQSeries Integrator の役割』を参照してください。対照的に、MQSeries Adapter Offering そのものでは、単純ルーティングのみを行うことができます。単純ルーティングは、メッセージの中のメッセージ制御値と構成ファイルの中の関連したメッセージ制御値の組み合わせに基づきます。単純ルーティングは、メッセージ本体の内容には基づきません。

カーネルがメッセージ送達への応答を要求されることがあります。これは、アプリケーション・レベルの肯定応答です。

ランタイム・フロー

このセクションでは、ランタイム・フローの詳細を述べます。典型的な実稼働環境における、カーネルによるメッセージの送信、ルーティング、トレース、および送達の方法についてです。ランタイム・フローの図は、15ページの図2 を参照してください。

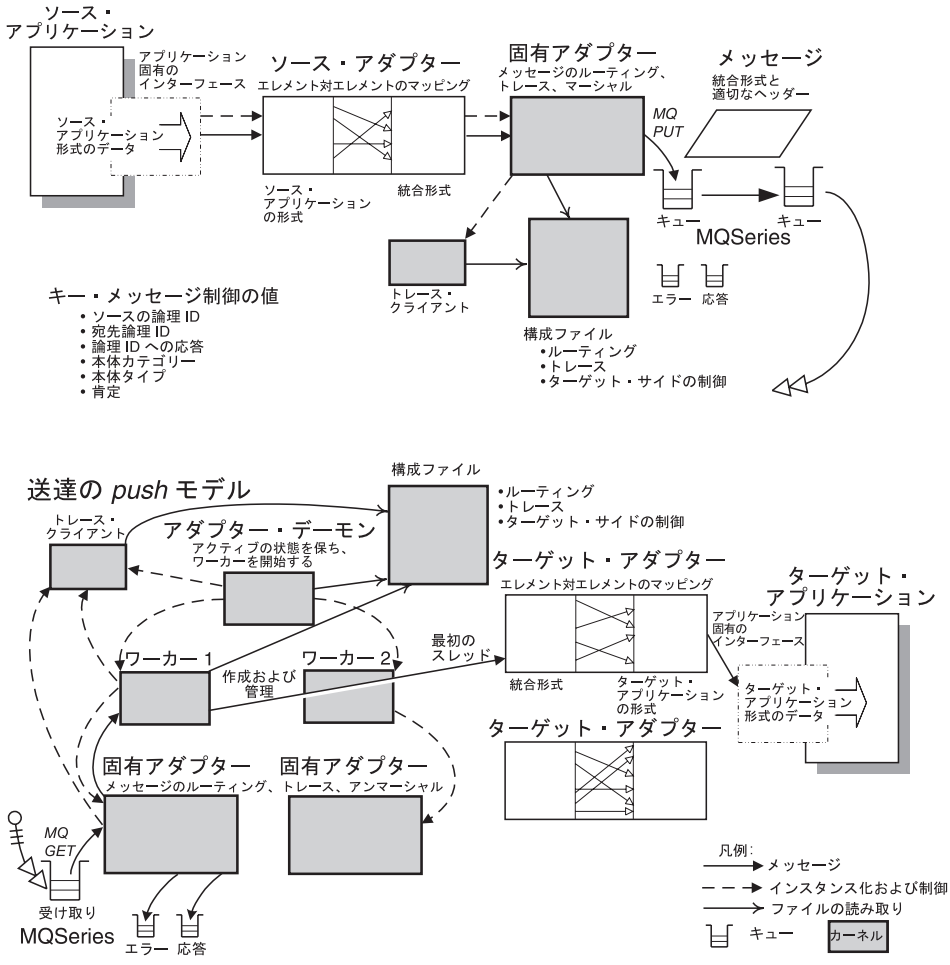


図 2. メッセージのマーシャル、送信、ルーティング、およびトレース (概要).

カーネルのソース・サイド

このセクションでは、カーネルのソース・サイドでのランタイム・フローについて説明します。つまり、ソース・アプリケーションからソース・アダプターを通って通信トランスポートに、どのようにデータが移動するかについてです。 19ページの『カーネルのターゲット・サイド』では、通信トランスポートからターゲットまで、どのようにデータが移動するかについて説明しています。

1. ソース・アダプターは、アプリケーション固有のインターフェースで、ソース・アプリケーションからメッセージを獲得します。通常は、ソース・アダプターはアプリケーション固有のインターフェースで呼び出されます。

2. ソース・アダプターは、 MQSeries Adapter Builder を使用してそこに作成された機能を実行します。ソース・アダプターは通常は、ソース・アプリケーションのフォーマットのデータを、アプリケーション非依存の統合フォーマット (メッセージ本体用) に変換します。

   ソース・アダプターは、機能の一部として、いくつかのメッセージ制御値を MQSeries Adapter Kernel のヘッダーに書き込みます。つまり、これらの値をメッセージのエンベロープに使用します。最初の 5 つのメッセージ制御値でマーシャルとルーティングが決まり、最後の値で肯定応答が決まります。

   メッセージ制御値

   ソース論理 ID
   ソース・アプリケーションの論理 ID。メッセージの中に常に必要です。

   宛先論理 ID
   ターゲット・アプリケーションの論理 ID メッセージの中にない場合は、構成ファイルにあるデフォルト値が代わりに使用されます。構成ファイルでは、メッセージに含まれない値の代わりに、複数の宛先論理 ID を使用することができます。

   論理 ID への応答
   応答が要求されている場合に、応答がそこに送信されることになるアプリケーションの論理 ID。デフォルトはメッセージの中のソース論理 ID です。

   本体カテゴリー
   メッセージのアプリケーション・タイプを表します (たとえば OAG や RosettaNet)。メッセージの中に常に必要です。

   本体タイプ
   メッセージの特定の目的を表します (たとえば「販売オーダーの追加」や「在庫の同期化」)。メッセージの中に常に必要です。

   肯定応答要求
   ソース・アプリケーションが応答を要求するかどうかを決定します。応答は次の形式のいずれかになります。

   - ターゲット・アプリケーションからの応答データ
   - OAG Confirm BOD メッセージ

     注: Confirm BOD メッセージは OAG によって定義済みです。その本体カテゴリーは `OAG` で、本体タイプは `CONFIRM_BOD_003` です。その中にデータも含めることができます。

   この応答は、アプリケーション・レベルの肯定応答です。

   カーネルがメッセージの送信に `sendRequestResponse` メソッドを使用すると、 `sendRequestResponse` メソッドが最初に受信した応答のみが使用され

ます。オリジナル・メッセージが複数の宛先に送信され、応答を要求する場合 (それは推奨できません) は、最初の応答のみがソース・アプリケーションに送信されます。

デフォルトでは肯定応答はありません。したがって、応答の要求や送信はありません。

3. ソース・アダプターはネイティブ・アダプターを初期化して次のものを渡します。
   - ソース・アダプターが実行されているアプリケーションの論理 ID。
   - メッセージ制御値とメッセージ本体データを含むメッセージ・ホルダー・オブジェクト。
4. ネイティブ・アダプターは構成ファイルを調べて、そのソース論理 ID ではトレースが可能になっているかどうかを判別します。トレースが可能になっていれば、ネイティブ・アダプターはトレース・クライアントをインスタンス化します。
5. トレース・クライアントは構成ファイルを調べて、どのトレース・レベルにするかの判別と他の値の獲得を行います。トレース・クライアントは、トレース・レベルを使用してトレース・メッセージをフィルターに掛けます。トレースの概要については、28ページの『トレース』を、トレースについての詳細は「*問題判別ガイド*」を参照してください。
6. ネイティブ・アダプターはメッセージ・ホルダー・オブジェクトの中から宛先論理 ID を探します。存在する場合は、それを使用します。
   - 宛先論理 ID がない場合、ネイティブ・アダプターは、ソース論理 ID、本体カテゴリー、本体タイプを基に、構成ファイルからデフォルトの宛先論理 ID を調べます。
   - ネイティブ・アダプターはソース論理 ID を基に、次の順序で段階的に、構成ファイルの本体カテゴリーおよび本体タイプ値を調べます。
     a. 特定の本体カテゴリー、および本体タイプの値。
     b. 特定の本体カテゴリー値、およびデフォルト本体タイプ値。
     c. デフォルト本体カテゴリー値、および特定の本体タイプ値。
     d. デフォルト本体カテゴリー、および本体タイプ値。

     注: カーネルは、構成ファイルから値を調べる際は毎回、この段階的な検索を行います。
7. ネイティブ・アダプターは、直前のステップで判別された宛先論理 ID ごとに、宛先論理 ID、本体カテゴリー、および本体タイプを基に、*通信モード* を調べます。次の通信モードがサポートされています。

   MQPP MQSeries 基本サービスを使用するカーネル・トランスポート・メッセージ。

MQRFH1 MQSeries を使用するカーネル・トランスポート・メッセージ、および MQSeries Integrator バージョン 1.1 を使用するブローカー・メッセージ。

MQRFH2 MQSeries を使用するカーネル・トランスポート・メッセージ、および MQSeries Integrator バージョン 2 を使用するブローカー・メッセージ。

MQBD MQSeries 基本サービスを使用する、カーネル・トランスポート・メッセージ。ただし、本体データのみの送信と受信を行う。

MQ MQSeries を使用するカーネル・トランスポート・メッセージ。

JMS Java Message Service (JMS) を使用して、カーネルはメッセージをトランスポートします。

FILE カーネルはメッセージをファイルに書き込み、メッセージをファイルから取得します。このモードは、診断の目的のみに使用されます。

通信モードごとに、メッセージ構造は異なります。12ページの『メッセージおよびメッセージ・フォーマット』を参照してください。通信モードについての詳細は、99ページの『付録A. 通信モード』を参照してください。

注: MQSeries Integrator を使用する場合は、MQSeries Integrator がメッセージを送信する最終宛先も、MQSeries Integrator と同じ通信モードを使用してメッセージを受信しなければなりません。

8. ネイティブ・アダプターは、通信モードを基に、自分自身の中のサブクラスをインスタンス化して、メッセージを処理します。サブクラスは*論理メッセージ・サービス* と呼ばれます。通信モードごとにサブクラスの論理メッセージ・サービスは異なります。

   ネイティブ・アダプターは、論理メッセージ・サービスに、宛先論理 ID、本体カテゴリー、および本体タイプを渡します。

9. 論理メッセージ・サービス・サブクラスは、メッセージを送信する必要があるパラメーターを検出します。たとえば通信モードが MQPP の場合、パラメーターにはフォーマットおよび受信キュー、応答キュー、エラー・キューの名前が含まれています。論理メッセージ・サービスは、渡された宛先論理 ID、本体カテゴリー、および本体タイプを基に、構成ファイルから次の段階的な検索を行います。

   a. 特定の本体カテゴリー、および本体タイプの値。

   b. 特定の本体カテゴリー値、およびデフォルト本体タイプ値。

   c. デフォルト本体カテゴリー値、および特定の本体タイプ値。

   d. デフォルト本体カテゴリー、および本体タイプ値。

論理メッセージ・サービスはこの時点で、メッセージのルーティングとマーシャルを行うために必要なすべての情報を得ています。

10. 論理メッセージ・サービスは次のタスクを実行します。
    - 通信モードとフォーマットに合わせたメッセージのマーシャル。各通信モードは、フォーマットが特に指定されていなければ、デフォルトのフォーマットを使用します。たとえば通信モードが MQRFH2 の場合、論理メッセージ・サービスは、トランスポートのために MQSeries を使用して、ブローカリングのために MQSeries Integrator バージョン 2 を使用して、適切なヘッダーの作成とメッセージの構成を行います。
    - メッセージの送信。たとえば通信モードが MQRFH2 の場合、メッセージを適切な MQSeries メッセージ・キューに書き込みます。
11. メッセージの送信には次の 2 つのメソッドが使用されます。
    - ネイティブ・アダプターが `sendMsg` メソッドを使用してメッセージを送信する場合、ネイティブ・アダプターは応答を待ちません。
    - ネイティブ・アダプターがメッセージの送信に `sendRequestResponse` メソッドを使用すると、論理メッセージ・サービスが応答を待ちます。ネイティブ・アダプターは論理メッセージ・サービスを使用して、構成ファイルに設定されている *受信タイムアウト時間* を応答キューでモニターします。

      受信タイムアウト時間は、ソース・アプリケーション ID、本体カテゴリー、および本体タイプに基づきます。
      - 肯定応答が受信されると、ネイティブ・アダプターはメッセージを戻します。
      - 受信タイムアウト時間内に肯定応答が受信されないと、ネイティブ・アダプターはメッセージを戻しません。
12. MQSeries または他のメッセージング・ソフトウェアは、メッセージのトランスポートを、構成された内容に従って行います。オプションで、MQSeries Integrator はブローカリング・サービスを行います。 8ページの『MQSeries Integrator の役割』を参照してください。
13. ソース・アダプターはネイティブ・アダプターを必要としなくなると、ネイティブ・アダプターをクローズしてリソースを解放します。

カーネルのターゲット・サイド

このセクションでは、独立の MQSeries Adapter Kernel を使用したターゲット・サイドでのメッセージの受信および処理を行う方法を説明し、WebSphere Application Server とともにカーネルを使用する方法を詳述しています。カーネルのターゲット・サイドで、JMS、WebSphere Business Integrator の JMS Listener コンポーネント、および WebSphere Application Server とカーネルを使用する際の説明は、28ページの『MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server とともに使用』を参照してください。このセクションでは、送達の push モデルを説明しますが、このモデルでは、カーネルがターゲット・アプリケーションへのメッセージの送達の開始と管理を行います。モデルの簡単な説明は、128ページの『送達モデル (delivery models)』を参照してください。

アダプター・ワーカーの概要

このセクションでは、MQSeries Adapter Kernel アダプター・ワーカーの構造および動作を説明します。 MQSeries Adapter Kernel アーキテクチャーの前提事項の 1 つは、ターゲット・アプリケーションは他のアプリケーションとの統合データ・フローに積極的に参加しないということです。つまり、アプリケーションは通常、メッセージを処理するために 積極的にポーリングしないということです。このような場合、メッセージ・データはターゲット・アプリケーションに対して積極的に push される必要があります。アダプター・ワーカーは、アプリケーションまたは他のサービスに、サービス・インターフェース・ タイプの範囲を選択し起動することで、メッセージ・データを push します。

MQSeries Adapter Kernel は、独立型デーモン (アダプター・デーモン) または Enterprise JavaBeans アプリケーション・サーバー (現在、IBM WebSphere Application Server Advanced Edition) で稼働するアダプター・ワーカーをホストすることが可能です。メッセージは、どのタイプのターゲット環境が使用されるかに従って、さまざまな方法でアダプター・ワーカーに到着します。独立型のアダプター・デーモンが使用される場合、ネイティブ・アダプターを使用してメッセージを受信する、1 つまたは複数の独立型のアダプター・ワーカーをホストします。 EJB サーバーが使用される場合、JMS Listener コンポーネントがメッセージを受信し、ワーカー・メッセージ bean (時々、message-driven-bean アダプター・ワーカーとも言いますが) にそのメッセージを渡します。

使用されるターゲット環境にかかわらず、アダプター・ワーカーはメッセージを受信した後、適切なターゲット・アダプターにそのメッセージを転送します。その後、ターゲット・アダプターは必要な作業を行い、そのメッセージをターゲット・アプリケーションに引き渡します。ターゲット・アダプターが作成され、特定のターゲット・アプリケーションと作業します。アダプター・デーモン、アプリケーション・サーバー、独立型アダプター・ワーカー、およびワーカー・メッセージ bean は、どのソースまたはターゲット・アプリケーションにも特有のものではありません。

アダプター・ワーカーは、次の 2 つのタイプのターゲット・ アダプター・インターフェースを処理します。Enterprice Access Builder (EAB) コマンド・アダプター、および EJB サービス・セッション bean です。それぞれのアダプター・タイプにハンドラーがあり、適切な環境をセットアップし、そのアダプターに必要な 付加的な構成情報にアクセスし、そのアダプターの動作に必要な他の低レベルのタスクを行います。使用されるハンドラーは、構成ファイルにリストされるアダプター・タイプによって決まります。2 つのタイプのハンドラーが、次の付加的なタスクを行います。

- EAB ハンドラーは、ログオン・クラスを取得します。ログオン・ クラスは、ターゲット・アダプターに接続情報を提供するために使用されます。また、EAB ハンドラーは、IBM Common Connector Framework (CCF) ランタイムを初期化します。ログオン・クラスは、ターゲット・アプリケーションの論理 ID を渡され、これを使用してアプリケーション固有のログオン情報を取得します。

- EJB ハンドラーは、Java Naming and Directory Interface™ (JNDI) 接続を取得します。その後、サービス・セッション bean のリモート・インターフェース、およびサービス・セッション bean にアクセスするのに必要な他の情報を取得します。

独立型アダプター・デーモン内の、アダプター・ワーカーの基本的な プロセス・フローは次の通りです。

1. アダプター・デーモンは、始動時にカーネルの構成ファイルで提供される情報に従って、1 つまたは複数の独立型アダプター・ワーカーをインスタンス化します。アプリケーションの論理 ID とオプションの本体カテゴリーおよび 本体タイプ値は、アダプター・デーモンに渡されます。本体カテゴリーおよび本体タイプ値が、付加的な構成値を取得するために使用されます。
2. それぞれの独立型アダプター・ワーカーが次のタスクを行います。
   a. アダプター・ワーカーはネイティブ・アダプターをインスタンス化し、メッセージの受信を開始します。それぞれのメッセージがトランザクション制御のもとで受信され、メッセージ・ホルダー・オブジェクトとしてアダプター・ ワーカーに戻されます。
   b. それぞれのメッセージが受信されるごとに、アダプター・ワーカーは 構成ファイルのメッセージを処理するために、アダプター・コマンド・タイプを検索し、そのコマンド・タイプに適切なハンドラーを取得します。
   c. そのハンドラーは、構成ファイルから、ターゲット・ アダプター・インスタンスをインスタンス化するのに必要な付加的な情報を取得します。ハンドラーはターゲット・アダプターをインスタンス化し、メッセージをターゲット・アダプターに渡します。
   d. メッセージが正常に (つまり、例外、エラー、または正しくないリターン・データもなく) 処理されると、そのメッセージは着信メッセージ・キューからコミットされます。メッセージが正常に処理されないと、エラー・キューに書き込まれます。メッセージが正常に処理されずエラー・キューに書き込むことができない場合、そのメッセージはロールバックされ、すべてのワーカーがシャットダウンされます。

WebSphere Application Server 内のアダプター・ワーカーの基本的なプロセス・フローは次の通りです。

1. WebSphere Application Server Advanced Edition の EJB サーバーと動作する JMS Listener プロセスが、JMS メッセージを受信します。その後、JMS Listener プロセスは ワーカー・メッセージ bean を取得しそのメッセージを処理します。アプリケーションの論理 ID とオプションの本体カテゴリーおよび 本体タイプ値は、ワーカー・メッセージ bean の環境の一部です。本体カテゴリーおよび本体タイプ値が、付加的な構成値を取得するために使用されます。
2. それぞれのワーカー・メッセージ bean が、次のタスクを行います。
   a. ワーカー・メッセージ bean は、ネイティブ・アダプターを インスタンス化し、ネイティブ・アダプターの `receiveMsg` メソッドを使用して、ネイティブ・アダ

プターに JMS メッセージを渡します。ネイティブ・アダプターは、JMS メッセージをメッセージ・オブジェクトに変換し、メッセージ・ホルダー・オブジェクトを戻します。

b. それぞれのメッセージ・ホルダー・オブジェクトが受信されるごとに、アダプター・ワーカーは 構成ファイルのメッセージ・ホルダー・オブジェクトを処理するために、ターゲット・アダプター・コマンド・タイプを検索し、そのコマンド・タイプに適切なハンドラーを取得します。

c. そのハンドラーは、構成ファイルから、ターゲット・ アダプター・インスタンスをインスタンス化するのに必要な付加的情報を取得します。ハンドラーはターゲット・アダプターをインスタンス化し、メッセージ・ホルダー・オブジェクトをターゲット・アダプターに渡します。

d. メッセージ・ホルダー・オブジェクトが正常に (つまり、例外、エラー、または正しくないリターン・データもなく) 処理されると、そのメッセージは着信メッセージ・キューからコミットされます。メッセージ・ホルダー・オブジェクトが正常に処理されないと、エラー・キューに書き込まれます。メッセージが正常に処理されずエラー・キューに書き込むことができない場合、そのメッセージはロールバックされ、すべてのワーカーがシャットダウンされます。

1 つのアダプター・デーモンおよび 1 つの独立型アダプター・ワーカーのあるターゲット・サイドでのランタイム・フローは次の通りです。

1. ターゲット・アプリケーションの受信キューごとに 1 つのアダプター・デーモンがあります。アダプター・デーモンは始動しています。

   始動時に、アプリケーション ID として動作する名前が与えられます。アダプター・デーモンそれぞれの名前は、通常は、宛先論理 ID、つまりターゲット・アプリケーションの論理 ID を基にしたものです。たとえば、アダプター・デーモンが、宛先論理 ID が `ABC` のターゲット・アプリケーションのサービスを行っている場合、そのアダプター・デーモンの名前は `ABCdaemon` です。

   始動時にアダプター・デーモンに渡されるパラメーターには、他に本体カテゴリーと本体タイプがあります。ネイティブ・アダプターは後でそれらを使って、着信するメッセージの通信モードと受信キューを判別します。

   アダプター・デーモンの始動についての説明は、90ページの『カーネルの始動』を参照してください。

2. アダプター・デーモンはその始動時に構成ファイルを調べて、そのアダプター・デーモンの名前でトレースが可能かどうかを判別します。トレースが可能になっていれば、アダプター・デーモンはトレース・クライアントをインスタンス化します。

   トレースの詳細については、「*問題判別ガイド*」を参照してください。

3. アダプター・デーモンはその始動時に最初のワーカーをインスタンス化して、そこにアダプター・デーモンの名前、メッセージの本体カテゴリー、および本体タイプを渡します。

4. 最初のワーカーは構成ファイルを調べて、そのアダプター・デーモン名でトレースが可能かどうかを判別します。トレースが可能になっていれば、最初のワーカーはトレース・クライアントをインスタンス化し、そのトレース・クライアントは構成ファイルを調べてトレース・レベルを決定します。有効なトレース・レベルのリストは、「*問題判別ガイド*」を参照してください。
5. 最初のワーカーは、アダプター・デーモンのアプリケーション ID を基に、インスタンス化されて始動されるワーカーの最小数を示す値を構成ファイルから調べます。

   最初のワーカーは、*依存関係アプリケーション ID* も検索します。依存関係アプリケーション ID は、そのワーカーがサービスを行うアプリケーションの名前です。依存関係アプリケーション ID は、後でネイティブ・アダプターに渡されます。
6. アダプター・デーモンは最初のワーカーにワーカーの最小数を照会します。
7. アダプター・デーモンは最初のワーカーを始動してから、最小数のワーカーをインスタンス化して始動します。

   ワーカーが複数ある目的は、ターゲット・アダプターへのマルチスレッドのメッセージ送達を行えるようにするためです。各ワーカーは自分のネイティブ・アダプターとともに、1 つのスレッドを扱います。ワーカーが 1 つだけの場合は、ターゲット・アダプターへの (その結果ターゲット・アプリケーションへの) メッセージの送達は、単一スレッドということです。

   AIX システムでは、スレッドのスケジューリング・ポリシーとして、プロセス・ベース・スケジューリングとシステム・ベース・スケジューリングの 2 つが使用可能です。プロセス・ベース・スケジューリング (デフォルト) では、すべてのユーザー・スレッドがオペレーティング・システム (OS) のカーネル・スレッドのプールにマップされて、仮想プロセッサーのプール上で実行されます。システム・ベース・スケジューリングでは、ユーザー・スレッドごとに単一の OS カーネル・スレッドにマップされて、単一仮想プロセッサー上で実行されます。 AIX の C の実行可能ファイルから呼び出される C のソース・アダプターを使用する場合は、システム・ベース・スケジューリングを使用してください。 AIX でのスレッド・スケジューリング・ポリシーについての情報は、45ページの6 のステップを参照してください。

   Windows システム、HP-UX、Solaris、および OS/400 でサポートされているのは、プロセス・ベース・スケジューリングのみですので注意してください。

最初のワーカーが実行する次のステップは、他のワーカーも実行します。

8. 各ワーカーがそれぞれの関連したネイティブ・アダプターをインスタンス化します。各ワーカーごとに、関連したネイティブ・アダプターが 1 つ存在します。依存関係アプリケーション ID、本体カテゴリー、および本体タイプが、ネイティブ・アダプターに渡されます。ネイティブ・アダプターはこれらの 3 つの値を使って、入ってくるメッセージの通信モードを判別し、論理メッセージ・サービスも使用してフォーマットと受信キューを判別します。このプロセスはメッセージの送信の場合のプロセスに似ています。

9. ネイティブ・アダプターはコミット制御のもとで受信キューから通信メッセージを取得し、メッセージ・ホルダー・オブジェクトに変換します。ネイティブ・アダプターは、ネイティブ・カーネル・ヘッダー以外の通信トランスポートに固有のヘッダーすべてを除去します。
10. ネイティブ・アダプターはメッセージ・ホルダー・オブジェクトをワーカーに渡し、ワーカーはメッセージのネイティブ・カーネル・ヘッダーの本体カテゴリー、本体タイプ、および肯定応答要求の値を読み取ります。

    ワーカーは、依存関係アプリケーション ID、本体カテゴリー、および本体タイプを基に、呼び出すべきターゲット・コマンドを次の順序で段階的に構成ファイルから検索します。

    a. 特定の本体カテゴリー、および本体タイプの値。
    b. 特定の本体カテゴリー値、およびデフォルト本体タイプ値。
    c. デフォルト本体カテゴリー値、および特定の本体タイプ値。
    d. デフォルト本体カテゴリー、および本体タイプ値。

    ワーカーは、ターゲット・コマンド・タイプを基に、適切なターゲット・アダプター・タイプ・ハンドラー、その特定のアダプター・タイプを処理する Java クラスを判別します。それによってその特定のターゲット・アダプターがインスタンス化されます。
11. アダプター・タイプ・ハンドラーには 2 種類あります。EAB コマンド・ターゲット・アダプター・ハンドラー、および EJB サービス・セッション bean ターゲット・アダプター・ハンドラーです。この異なる種類のアダプター・ハンドラーは次のように動作します。

    注: EJB サービス・セッション bean ターゲット・アダプター・ハンドラーは、Windows NT プラットフォームで WebSphere Application Server と稼働する WebSphere Business Integrator の場合だけサポートされます。

    - EAB コマンド・ターゲット・アダプター・ハンドラーが 呼び出されると、Common Connector Framework (CCF) 環境を開始し、構成ファイルから 取得された名前でログオン・クラスを設定し、そして構成ファイルから 取得された名前で EAB ターゲット・アダプターを起動します。
    - EJB サービス・セッション bean ターゲット・アダプターは、WebSphere Business Integrator および WebSphere Application Server と相互作用して、適切な構成情報を取得し、EJB サービス・セッション bean を起動しなければなりません。カーネルのターゲット・サイドで、JMS、WebSphere Business Integrator の JMS Listener コンポーネント、および WebSphere Application Server とカーネルを使用する際の説明は、28ページの『MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server とともに使用』を参照してください。
12. それぞれのアダプター・タイプには、次のような異なる インターフェースと必須のサポートするクラスがあります。

- EAB ターゲット・アダプター・コマンドには、3 つの 呼び出しメソッドがあります。この 3 つのメソッドは次の順番で実行します。
  a. *set message input* メソッド: プロセスへのメッセージをターゲット・アダプターにセットします。
  b. *execute* メソッド: set message input メソッドによってターゲット・アダプターに書き込まれたメッセージを処理してから待ち状態になります。
     1) ターゲット・アダプターは、 MQSeries Adapter Builder を使用してそこに作成された機能を実行します。通常、統合メッセージからのデータを、ターゲット・アプリケーションのフォーマットに変換します。エレメントとエレメントがマップされます。
     2) ターゲット・アダプターは、アプリケーション固有のインターフェースを使って、ターゲット・アプリケーションにメッセージを送信します。
     3) ターゲット・アプリケーションは、その特性によって、応答をターゲット・アダプターに送信するものとしないものがあります。
  c. *get message output* メソッド: ターゲット・アダプターから応答を取得します。応答は、単にターゲット・アプリケーションがメッセージを受信したことを示しますが、データを含むこともできます。
- EJB サービス・セッション bean は 1 つのメソッドを呼び出します。このメソッドは、`TerminalDataContainer` オブジェクトを必要とします。このメソッドによって戻されたデータは、応答データと見なされ、`TerminalDataContainer` タイプ・オブジェクトでなければなりません。

13. ターゲット・アダプターのコマンドが例外をスローしない場合、またはターゲット・アダプターのコマンドに Confirm BOD 応答がない場合 (この場合はエラーになります) は、ワーカーはネイティブ・アダプターを使って受信キューの受信メッセージをコミットします。
14. 肯定応答が要求されていた場合、ワーカーは、ネイティブ・アダプターの `sendResponse` メソッドを呼び出します。
    - ターゲット・アダプターが応答を作成する場合は、オリジナル・メッセージの論理 ID への応答を、応答メッセージの宛先論理 ID フィールドに書き込みます。
    - ターゲット・アダプターが応答を作成しないときは、ワーカーが完了状況を含む Confirm BOD 応答メッセージを作成します。
      – エラーがない場合は、完了状況は正常です。
      – エラーがある場合は、完了状況はエラー状態に設定されます。
15. 応答が送信されます。
    a. 応答メッセージが作成されていると、ワーカーがネイティブ・アダプターに応答メッセージを送信します。
    b. ネイティブ・アダプターは応答メッセージを応答キューに書きこみます。
    c. ネイティブ・アダプターは、次のような受信していたオリジナル・メッセージによっては、応答メッセージを送信します。

- それが MQSeries の要求メッセージだった場合は、ネイティブ・アダプターは応答のキュー情報を MQSeries の要求メッセージから獲得します。このキュー情報はメッセージの中の宛先論理 ID をオーバーライドします。
- それが MQSeries の要求メッセージではなかった場合は、ネイティブ・アダプターは sendMsg メソッドを使って応答を送信します。

16. 例外またはエラー状況の Confirm BOD 応答メッセージの場合、ワーカーは、アダプター・デーモンと同じディレクトリーにある EpicSystemExceptionFile*nnnnnnnn*.log という例外ファイルに例外メッセージを記録します。ここで、*nnnnnnnn* はログ・ファイルの数です。さらに、WebSphere Business Integrator クラスがインストールされると、これらのクラスは例外を WebSphere Business Integrator Solution Management コンポーネントに送信します。93ページの『例外メッセージ』を参照してください。
17. 例外またはエラー状況の Confirm BOD 応答メッセージの場合、ワーカーは、ネイティブ・アダプターにオリジナル・メッセージをエラー・キューに書き込むように指示を出します。エラー・キューの名前は、オリジナル・メッセージの依存関係論理 ID、本体カテゴリー、および本体タイプを基に、構成ファイルから取得されます。

    ワーカーは、依存関係アプリケーション ID、本体カテゴリー、および本体タイプを基に、次の順序で段階的に構成ファイルから検索を行います。

    a. 特定の本体カテゴリー、および本体タイプの値。
    b. 特定の本体カテゴリー値、およびデフォルト本体タイプ値。
    c. デフォルト本体カテゴリー値、および特定の本体タイプ値。
    d. デフォルト本体カテゴリー、および本体タイプ値。

    - ネイティブ・アダプターがエラー・キューにエラー・メッセージを書き込むことができるときは、ネイティブ・アダプターは受信キューのメッセージをコミットするように指示されます。
    - ネイティブ・アダプターがエラー・キューにエラー・メッセージを書き込むことができないときは、次のような動作になります。

      a. ワーカーはネイティブ・アダプターにロールバックを指示します。つまり、コミットは行われません。
      b. ワーカーは、そのときのアダプター・デーモンのもとのすべてのワーカーに、シャットダウンするように指示するフラグを立てます。このことは、そのときのメッセージに問題があることを意味します。すべてのワーカーをシャットダウンすることによって、他のワーカーが、問題がある同じメッセージを再処理して同じ結果になることを防ぎます。
      c. メモリー外エラーが起きると、その例外は他のすべての例外と同じように扱われます。ただし、ワーカーは自分自身のフラグを立てて、その時点のメッセージの処理を完了したときに停止します。これによって、他のワーカーがより多くのメモリーを使えるようにしています。

18. ネイティブ・アダプターがワーカーに、作業が行われたことを通知すると、ワーカーは次の 2 つのフラグをチェックします。
    - そのワーカーが停止すべきかどうか。これは、Java のメモリー外の状態に起因します。
    - すべてのワーカーが停止すべきがどうか。これは前記のステップで述べたことに起因します。
19. いずれかのフラグが立っていると、ワーカーは停止します。どちらのフラグも立っていないと、ワーカーは次のメッセージを処理します。ワーカーは、ネイティブ・アダプターにメッセージを受信するよう要求します。
20. 応答キューに応答メッセージが置かれている場合、またはエラー・キューにエラー・メッセージが置かれている場合は、次の動作になります。
    a. そのメッセージは、MQSeries または他のメッセージング・ソフトウェアによって、ソース・サイドのカーネルに送達されます。
    b. ソース・アダプターが自分のネイティブ・アダプターの `sendRequestResponse` メソッドを呼び出したときは、カーネルは応答キューのメッセージを検索してソース・アダプターにそれを戻します。ソース・アダプターが `sendMsg` メソッドを呼び出したときは、カーネルはソース・アプリケーションの受信キューにメッセージを書き込みます。

トランザクション機能

トランザクション はオペレーションの集合であり、分割できない作業単位として実行されなければなりません。トランザクションを構成するすべてのオペレーションが成功すると、そのトランザクションは*コミット済み* になります。つまり、すべてのオペレーションが完了したことになります。トランザクションを構成するオペレーションが 1 つでも失敗すると、そのトランザクションは *ロールバック* になります。つまり、オペレーションは何も行われなかったことになります。ソース・アダプターは、MQSeries Adapter Kernel のトランザクション機能を使って、一連のオペレーションを単一ユニットとして実行することができます。その際、トランザクションがコミットされてすべてのオペレーションが成功するか、またはトランザクションがロールバックされてオペレーションが行われないかを確認します。

トランザクション機能は、MQSeries Adapter Builder、またはカーネルの Java API のクラス `EpicNativeAdapter` の `begin`、`rollback`、および `commit` メソッドを使用して、アダプターに作成することができます。トランザクションのメソッドが正しくないコンテキスト (たとえば、最初に `begin` メソッドの呼び出しを行っていない中での `commit` メソッドの呼び出しや、別のトランザクションの範囲内での `begin` メソッドの呼び出し) で呼び出されると、カーネルはその呼び出しを無視してトレースに警告を発行します。API の使用についての情報は、95ページの『第5章 MQSeries Adapter Kernel API の使用』を参照してください。

制限

カーネルのトランザクション機能に関連して、次の制限があります。

- トランザクションは sendRequestResponse メソッドではサポートされていません。
- トランザクションのネスト (トランザクション内で別のトランザクションを呼び出すこと) はサポートされていません。
- すべての通信モードでトランザクションがサポートされているわけではありません。詳細は、99ページの『付録A. 通信モード』を参照してください。

トレース

トレース・メッセージには、カーネルでのある時点での処理中のメッセージの状態が含まれます。トレース・メッセージは、カーネルや作成したアダプターの問題の診断に利用することができます。 MQSeries Adapter Kernel「*問題判別ガイド*」では、カーネルでのトレースの使用について述べています。

MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server とともに使用

このセクションでは、MQSeries Adapter Kernel を WebSphere Business Integrator および WebSphere Application Server 製品とともに使用する方法を説明します。さらに詳細については、WebSphere Business Integrator 資料を参照してください。

JMS Listener

WebSphere Business Integrator では、JMS Listener というコンポーネントを備えています。このコンポーネントは、MQSeries Adapter Kernel および WebSphere Application Server Advanced Edition とともに動作し、ターゲット・アプリケーションにメッセージを送達する代替方法を提供します。JMS Listener は、WebSphere Application Server の Enterprise JavaBeans (EJB) サーバー内で動作します。このセクションでは、JMS Listener の機能を概説します。WebSphere Business Integrator および JMS Listener の構成方法の詳細を含む追加情報は、WebSphere Business Integrator の資料を参照してください。JMS Listener をターゲットとして認識するように MQSeries Adapter Kernel を構成する際の情報は、58ページの『カーネルの構成』を参照してください。JMS Listener をターゲットとして使用する方法は、メッセージをアダプター・デーモンに送信する方法と同じです。

JMS Listener を使用する前に、カーネルのターゲット・サイドに MQSeries Adapter Kernel ワーカー・メッセージ bean、および Java サービス・セッション bean アダプターまたは EAB アダプターを配置しなければなりません。MQSeries Adapter Builder を使用して、これらの作業を行います。WebSphere Business Integrator 環境では、JMS Listener がアダプター・ワーカーの代わりにメッセージを受信し適切なアダプター・ワーカーを起動するということを除いて、WebSphere Application Server 内のカーネルの動作は、独立型アダプター・デーモンと一緒の場合と同様です。独立型 MQSeries Adapter

Kernel 環境では、アダプター・デーモンがアダプター・ワーカーを始動させ、アダプター・ワーカーが順番にメッセージを直接受信します。

MQSeries Adapter Kernel が JMS Listener とともに動作する場合のイベントのシーケンスは次の通りです。

1. JMS Listener は JMS キューを監視して、EJB クライアントまたは非 EJB アプリケーションから、JMS メッセージ・オブジェクトを受信します。
2. JMS Listener は、ワーカー・メッセージ *bean* をインスタンス化して、メッセージ・オブジェクトをそれに渡します。ワーカー・メッセージ bean は、*session bean* のインスタンスで、特定の クライアントに関連した一時データをカプセル化するエンタープライズ bean の 1 タイプです。
3. ワーカー・メッセージ bean は、JMS メッセージ・オブジェクトを MQSeries Adapter Kernel メッセージ・ホルダー・オブジェクトに変換します。
4. カーネルは、メッセージのヘッダー値に基づいて、EAB アダプターまたは EJB アダプターを呼び出します。呼び出されるアダプター・タイプが EAB アダプターのとき、データ・フローは独立型の場合と同様です。呼び出されるアダプター・タイプが EJB アダプターの場合、EJB ハンドラーが呼び出されて次のタスクを実行します。
   - 起動するのに正しいサービス・セッション bean (ホーム・インターフェース)、呼び出すのに適切なメソッド、およびそのメソッドの `TerminalDataContainer` オブジェクト用の入力パラメーター・タイプを判別します。
   - メッセージ・ホルダー・オブジェクト内にあるアプリケーション・データを、`Mapper` クラスを使用して、サービス・セッション bean に適切な `TerminalDataContainer` データ構造に変換します。`TerminalDataContainer` オブジェクトには、メッセージ・ホルダー・オブジェクトのメタデータに加え アプリケーション・オブジェクトが含まれています。多くの場合、アプリケーション・オブジェクトは メッセージ・ホルダー・オブジェクトの本体データ XML 文書ストリングです。
   - サービス・セッション bean を起動し、`TerminalDataContainer` オブジェクトを、サービス・セッション bean の適切なメソッドに渡します。サービス・セッション bean は、Java サービス・アダプターの一部ですが、メッセージのターゲットです。
5. 応答が要求されると、ワーカー・メッセージ bean はその応答 `TerminalDataContainer` オブジェクトを、メッセージ・ホルダー・オブジェクトに変換し、ネイティブ・アダプターを使用して応答を送信します。
6. エラーが発生すると、ワーカー・メッセージ bean は そのメッセージ・ホルダー・オブジェクトを、ネイティブ・アダプターを使用してエラー・キューに書き込みます。

各国語サポート

MQSeries Adapter Kernel では、Java アダプターが使用される場合に各国語サポート (NLS) を提供しています。各国語サポートは、C アダプターの場合は提供されません。

第2章 カーネルのインストール計画

この章では MQSeries Adapter Kernel のための前提条件とコンポーネントをリストします。

最新の詳細情報は、次の MQSeries 製品ファミリー Web サイトを参照してください。

www.ibm.com/software/ts/mqseries/

ここに示す情報は、更新されることがあります。サポートしているソフトウェアのレベルに関しての最新の情報は、次を参照してください。

www.ibm.com/software/ts/mqseries/platforms/supported.html

ハードウェア

MQSeries Adapter Kernel は、次のハードウェアで動作します。

- Windows NT 4.0、Service Pack 5 以降、または Windows 2000、 Service Pack 1 が動作する IBM PC マシン (または互換機)。
- AIX バージョン 4.3.2 または 4.3.3 が動作する IBM RS/6000 マシン。
- HP-UX バージョン 11.0 が動作する HP Series 9000 マシン。
- Solaris バージョン 8 が動作する Sun SPARC または UltraSPARC マシン。
- OS/400 バージョン 4.4 または 4.5 が動作する IBM AS/400 または iSeries マシン。

注: OS/400 に MQSeries Adapter Kernel をインストールするには、 AS/400 マシンとのインターフェースのために Windows システムが必要です。詳細は、34ページの『OS/400 インストールの前提条件』を参照してください。

MQSeries Adapter Kernel には、製品のコードおよびデータのための最低約 25 MB のディスク・スペースが必要です。

アダプターを収容するための十分な使用可能ディスク・スペースがあることを確認してください。アダプターのサイズは、データ構造のサイズ、マッピングの複雑さ、および使用するカスタム・コードによって異なります。たとえば、Windows システムでもアダプター・サイズは次のように異なります。ユーザーの実際のアダプターに必要なディスク・スペースは、これ以上またはこれ以下になることがあります。次の例はそれぞれアダプター・ソース、コンパイルしたアダプター・コード、 API ソース、およびコンパイルした API コードを含んでいます (MB または KB 単位)。

- 「販売オーダーの追加」用ソース・アダプター: 1.89 MB
- 「顧客記録の同期化」用ターゲット・アダプター: 389 KB
- 「在庫記録の同期化」用ターゲット・アダプター: 161 KB
- 「品目の同期化」用ターゲット・アダプター: 249 KB

- 「販売オーダーの同期化」用ターゲット・アダプター: 579 KB

この外に、カーネルとアダプターのワークスペースとして最低 20 MB が必要です。ワークスペースの要件は、キューの数とサイズ、およびトレース・ファイルのサイズなどのいくつかの要素によって大きく異なります。

ソフトウェア

このセクションでは、MQSeries Adapter Kernel との使用をサポートしているソフトウェアをリストします。サポートしているレベルを示します。105ページの『付録B. 妥当性検査済みの構成』を参照してください。開発システムには C コンパイラーが必要ですが、実動システムには必要ありません。ここでリストする C コンパイラーは、MQSeries Adapter Kernel でのテストで正常だったものです。他の C コンパイラーもカーネルで正しく動作するかもしれませんが、正式にはサポートされていません。

Windows システム用:

- Microsoft Windows NT バージョン 4.0、Service Pack 5 以降; または Microsoft Windows 2000、 Service Pack 1。 Microsoft Windows のバージョンと Service Pack を判別するには、Windows Explorer をオープンして、**「ヘルプ」>「バージョン情報」**をクリックします。
- Microsoft Visual C++ 6.0 Compiler。
- SupportPac MA88 を含む MQSeries バージョン 5.2。
- IBM Java Development Kit (JDK) バージョン 1.2.2 または 1.3。

  注: 現在、Windows NT と Windows 2000 だけが、MQSeries Adapter Kernel が JDK バージョン 1.3 をサポートする場合のプラットフォームです。

AIX 用:

- AIX オペレーティング・システム、バージョン 4.3.2 または 4.3.3。
- IBM C Set++ for AIX バージョン 3.1.3。
- SupportPac MA88 を含む MQSeries バージョン 5.2。
- Java Development Kit バージョン 1.2.2。JDK 1.3 はサポートしていません。
- X Window System (X11R5 またはそれ以上)。これはインストールのために必要なもので、実行時には必要ありません。

HP-UX 用:

- HP-UX オペレーティング・システム、バージョン 11.0。
- HP-UX C/ANSI C Compiler。詳細は `readme.txt` ファイルを参照してください。
- SupportPac MA88 を含む MQSeries バージョン 5.2。
- Java Development Kit バージョン 1.2.2。JDK 1.3 はサポートしていません。

- X Window System (X11R5 またはそれ以上)。これはインストールのために必要なもので、実行時には必要ありません。

Solaris 用:

- Solaris 稼働環境、バージョン 8。
- Sun Workshop Compilers C/C++。詳細は readme.txt ファイルを参照してください。
- SupportPac MA88 を含む MQSeries バージョン 5.2。
- Java Development Kit バージョン 1.2.2。JDK 1.3 はサポートしていません。
- X Window System (X11R5 またはそれ以上)。これはインストールのために必要なもので、実行時には必要ありません。

OS/400 用:

- 次のプログラムを含む OS/400 オペレーティング・システム、バージョン 4.4 または 4.5。
  - Java Toolkit および Java Developer Kit バージョン 1.2.2。JDK 1.3 はサポートしていません。Java Toolkit と Java Developer Kit は、ライセンス・プログラム番号 5769-JV1 として出荷されています。AS/400 システムへの MQSeries Adapter Kernel のインストールに必要な Java Developer Kit のバージョンについての付加的な詳細は、34ページの『OS/400 インストールの前提条件』を参照してください。
  - Host Servers オプション。これはライセンス・プログラム番号 5769-SS1、オプション 12 として出荷されています。
  - Qshell Interpreter。これはライセンス・プログラム番号 5769-SS1、オプション 30 として出荷されています。
  - TCP/IP。これはライセンス・プログラム番号 5769-TC1 として出荷されています。
  - Integrated Language Environment C (AS/400 用)。これはプログラム番号 5769-CX2 として出荷されています。
- SupportPac MA88 を含む MQSeries バージョン 5.2。

OS/400 への MQSeries Adapter Kernel のインストールに必要な付加要件は、 34ページの『OS/400 インストールの前提条件』を参照してください。

次の製品が MQSeries Adapter Kernel でサポートされています。

- SupportPac MA88 を含む MQSeries バージョン 5.2。

  注: MQSeries を使用しない場合は、Java Message Service (JMS) などの別のメッセージング・ソフトウェア製品を実装して使用する必要があります。
- MQSeries Integrator バージョン 1.1
- MQSeries Integrator バージョン 2

MQSeries Adapter Kernel、MQSeries、および MQSeries Integrator の妥当性検査済みの構成のリストは、105ページの『付録B. 妥当性検査済みの構成』を参照してください。

OS/400 インストールの前提条件

このセクションでは、AS/400 または iSeries システムに、MQSeries Adapter Kernel をインストールするための前提条件について述べます。AS/400 システムへの MQSeries Adapter Kernel のインストールについての詳細な説明は、42ページの3 のステップを参照してください。AS/400 の端末は本来 Java グラフィックスをサポートしていないので、カーネルの Java ベースの GUI インストール・プログラムを実行するには、Windows システムのようなグラフィックスが使えるワークステーションが必要です。ワークステーションは、次のいずれかの方法で AS/400 システムとインターフェースをとることができます。

- リモート AWT 経由。この場合、すべてのグラフィックスは AS/400 システムで処理されてワークステーションに表示されます。詳細な記述は、『リモート AWT の使用』にあります。
- 接続クライアントとして。この場合、ワークステーションがグラフィックスを処理して表示します。詳細な記述は、35ページの『接続クライアントの使用』にあります。

このセクションでは、グラフィックスを使用できるワークステーションとして Windows システムを使うことを想定します。

リモート AWT の使用

リモート AWT を使用する場合は、Java グラフィックスの処理は AS/400 システム上で行われ、グラフィックスは AS/400 システムに接続されたクライアント・ワークステーションに表示されます。このセクションでは、リモート AWT を使って AS/400 に MQSeries Adapter Kernel をインストールする際に適合しなければならない要件を述べます。

次のプログラムを、OS/400 でインストールする必要があります。

- Java Toolkit および Java Developer Kit バージョン 1.2.2。Java Toolkit と Java Developer Kit は、ライセンス・プログラム番号 5769-JV1 として出荷されています。OS/400 上のリモート AWT 機能は、Java Developer Kit が備えています。
- TCP/IP。これはライセンス・プログラム番号 5769-TC1 として出荷されています。TCP/IP についての詳細は、*AS/400 TCP/IP Fastpath Setup Information* および *AS/400 TCP/IP Configuration* の資料を参照してください。これらは、http://www.ibm.com/servers/eserver/iseries/library/ の AS/400 ライブラリーで入手できます。

ワークステーションの要件は次のとおりです。

- Windows 95、Windows 98、Windows NT、または Windows 2000 が動作する IBM PC マシン (または互換機)。
- AS/400 システムとの TCP/IP 接続。
- JDK 1.2.2 またはそれ以上。

リモート AWT のセットアップと開始は、次のステップで行います。

1. ワークステーションに JDK 1.2.2 またはそれ以上がインストールされていることを確認します。
2. AS/400 システムとワークステーションが TCP/IP 接続されていることを確認します。
3. AS/400 システムの /QIBM/ProdData/Java400/jdk12 ディレクトリーのファイル RAWTGui.jar を、ワークステーションのディレクトリーにコピーします。
4. ワークステーションで、RAWTGui.jar ファイルをコピーしたディレクトリーにして、次のコマンドを入力してリモート AWT を開始します。

   ```
   java -jar RAWTGui.jar
   ```

注: AS/400 システムでの Java グラフィックスの処理は、もともとリソースを集中的に使用するため、MQSeries Adapter Kernel のインストールにリモート AWT を使用すると、接続クライアントを使用する場合よりかなり時間がかかることがあります。

リモート AWT についての詳細は、www.ibm.com/servers/eserver/iseries/library/ の AS/400 ライブラリーを参照してください。

接続クライアントの使用

AS/400 システムへの MQSeries Adapter Kernel のインストールに接続クライアントを使用する場合は、Java グラフィックスの処理は AS/400 システム上ではなく、クライアント・ワークステーション上で行われます。このセクションでは、接続クライアントを使って AS/400 に MQSeries Adapter Kernel をインストールする際に適合しなければならない要件を述べます。

次のプログラムを、OS/400 でインストールする必要があります。

- Java Toolkit および Java Developer Kit バージョン 1.2.2。Java Toolkit と Java Developer Kit は、ライセンス・プログラム番号 5769-JV1 として出荷されています。
- Host Servers オプション。これはライセンス・プログラム番号 5769-SS1、オプション 12 として出荷されています。
- TCP/IP。これはライセンス・プログラム番号 5769-TC1 として出荷されています。

ワークステーションの要件は次のとおりです。

- Windows NT 4.0、Service Pack 5、または Windows 2000、Service Pack 1 が動作する IBM PC マシン (または互換機)。
- AS/400 システムとの TCP/IP 接続。
- JDK 1.2.2 またはそれ以上。

カーネルのコンポーネント

インストール後は MQSeries Adapter Kernel はルート・ディレクトリーにあります。それには他のディレクトリーを順次に含むことができるサブディレクトリーが含まれています。インストールと構成に最も関係のあるファイルの要約とともに、ルート・ディレクトリーとそのサブディレクトリーがリストされます。

root デフォルト名は、Windows システムでは C:¥Program Files¥MQAK、AIX では /usr/lpp/mqak on、HP-UX では /MQAK、Solaris では /opt/MQAK、および OS/400 では /QIBM/ProdData/mqak です。ここには次のものが含まれています。

- MQSeries Adapter Kernel の他のすべてのディレクトリー。
- aqmsetenv.bat (Windows システム)、または aqmsetenv.sh (UNIX) のファイル。インストール後に、必要ならば、このファイルでシステム環境変数を変更します。
- readme.txt ファイル。
- aqmuninstall.bat (Windows システム)、または aqmuninstall.sh (UNIX) のファイル。

bin 次のものが含まれています。

- クラス・ライブラリーと共用ライブラリー。
- カーネルの一部として提供されるアダプター。これは検査のみに使用します。
- aqmversion.bat ファイル (Windows システム)、または aqmversion.sh ファイル (UNIX および OS/400)。実行すると、カーネルのバージョン番号を表示するスクリプトです。
- aqmcrtmsg.bat ファイル (Windows システム)、または aqmcrtmsg.sh ファイル (UNIX および OS/400)。実行すると、実動の前の構成ファイルの妥当性検査に使用する XML ファイルを作成するスクリプトです。
- aqmsndmsg.bat ファイル (Windows システム)、または aqmsndmsg.sh ファイル (UNIX および OS/400)。実行すると、実動前の構成ファイルの妥当性検査を行うスクリプトです。
- aqmstrad.bat ファイル (Windows システム)、または aqmstrad.sh ファイル (UNIX および OS/400)。実行すると、アダプター・デーモンを始動するスクリプトです。
- aqmstrtd.bat ファイル (Windows システム)、または aqmstrtd.sh ファイル (UNIX および OS/400)。実行すると、トレース・サーバーを開始するスクリプトです。

documentation
Information Center などの製品資料が含まれています。

runtimefiles
カーネルのランタイム・ファイルが含まれています。

samples

アダプターのサンプル、および関連した構成ファイルとユーティリティー・ファイルが含まれています。これらを使って実験と習得を行うことができます。

注: カーネルは、MQSeries Adapter Builder を使って作成されたアダプターと共に使用するように意図されています。カーネルは、カスタム・コードのみでカーネル API を呼び出して使用するようには意図されていません。アダプター・サンプルは、カーネル機能の動作の理解の補助、および診断の補助としてのみ提供されています。

- アダプターのサンプル。
- カーネルのセットアップ・ファイル `aqmsetup`。アダプターのサンプルをサポートする値になっています。このファイルの説明は、64ページの『セットアップ・ファイル』を参照してください。
- カーネルの構成ファイル `aqmconfig.xml`。サンプル・トレース値など、アダプターのサンプルをサポートする値になっています。このファイルの説明は、64ページの『構成ファイル』を参照してください。

toolkit
ソフトウェア開発ツールキット (SDK) が含まれており、次のものがあります。

- ヘッダー・ファイル
- Windows システムでのコンパイル時に使用するライブラリー・ファイル。

uninstall
カーネルのアンインストールに使用するファイルが含まれています。

verification
カーネルのインストールの検査をサポートする次のファイルが含まれています。

- `aqmverifyinstall.bat` ファイル (Windows システム)、または `aqmverifyinstall.sh` ファイル (UNIX および OS/400)。実行すると、1 台のコンピューターのカーネルのインストールを検査するスクリプトです。
- `aqmcreateq.bat` ファイル (Windows システム)、または `aqmcreateq.sh` ファイル (UNIX および OS/400)。検査用の

MQSeries キューを作成するスクリプトです。94ページの『MQSeries キューの作成』を参照してください。

- `aqmconfig.xml` ファイル。このファイルの説明は、64ページの『構成ファイル』を参照してください。
- `aqmsetup` ファイル。このファイルの説明は、64ページの『セットアップ・ファイル』を参照してください。
- `aqminstalltest.xml` ファイル。

第3章 カーネルのインストール

この章では、MQSeries Adapter Kernel をインストールし、確認するのに必要なステップを説明します。インストールは、通常、次のステップから成っています。

ステップ 1. インストールの準備をします。詳細は、40ページの『インストールの準備』を参照してください。

ステップ 2. カーネルをインストールします。詳細は、41ページの『カーネルのインストール』を参照してください。

ステップ 3. インストール後のステップを完了します。詳細は、44ページの『インストール後のステップを完了』を参照してください。

ステップ 4. インストールを確認します。詳細は、46ページの『インストールの検査』を参照してください。

この章では、次のトピックも取り扱っています。

- サイレント・インストールを使用して MQSeries Adapter Kernel をインストールする方法。詳細は、51ページの『サイレント・インストールの使用法』を参照してください。
- 以前のバージョンからの MQSeries Adapter Kernel のアップグレード。詳細は、53ページの『カーネルのアップグレード』を参照してください。
- インストールした MQSeries Adapter Kernel の除去。詳細は、54ページの『カーネルの除去』を参照してください。

カーネルをインストールした後、使用する準備のために次の 付加的な作業を行います。

1. カーネルを構成します。詳細は、58ページの『カーネルの構成』を参照してください。
2. メッセージング・ソフトウェアとオプション・ソフトウェアを構成します。詳細は、89ページの『MQSeries および MQSeries Integrator の構成』を参照してください。
3. アダプターを MQSeries Adapter Builder を使用して構築し、テストと配置を行います。
4. カーネルを始動します。詳細は、90ページの『カーネルの始動』を参照してください。

インストールの準備

MQSeries Adapter Kernel をインストールするには、アドミニストレーター権限またはルート権限が必要です。MQSeries Adapter Kernel をインストールする場所およびカーネル構成ファイルを 2 つ置く場所にファイルを作成して、アクセスするためのアクセス権が必要です。実行可能パスに現行ディレクトリーがなければなりません。カーネルを実行するすべてのユーザー ID に、読み取り、書き込み、および実行のアクセス権があることを確認してください。

キュー・マネージャーの作成およびキューの作成とアクセスなどの MQSeries のオペレーションを行うには、権限が必要です。これらのオペレーションの実行方法は、プラットフォームによって異なります。プラットフォームについての詳細は、 *MQSeries Administration Guide* を参照してください。

カーネルのプロセスを始動するユーザー ID は、mqm グループでなければなりません。カーネル・プロセスには、次の 2 種類があります。

- アダプター・デーモン。コンピューターによって実行されるターゲット・アプリケーションごとに 1 つ存在します。
- トレース・サーバー (オプション)

ソース・アダプターは、ソース・アプリケーションのプロセスで実行されるので注意してください。ソース・アダプターを実行させるには、ソース・アダプターを含んでいるデーモンまたはサーバーを始動する必要があります。

作成したアダプターを実行するには、カーネルをインストールして構成することが必要です。ただし、MQSeries Adapter Builder をインストールするためや、それをアダプターの作成に使用するためにカーネルをインストールする必要はありません。

インストールを始める前に、次のステップを行います。

- CD-ROM の、またはローカル・エリア・ネットワークで、`readme.txt` ファイルをお読みください。そこには、本書が完成した後に使用可能になった重要な情報が含まれていることがあります。そのファイルはインストール・プログラムのディレクトリーのルートにあります。
- 次の MQSeries Web サイトを参照してください。www.ibm.com/software/ts/mqseries/。そこには、今後の改訂版も含めて本書が発行された後に使用可能になった重要な情報が含まれていることがあります。
- MQSeries Adapter Kernel の前のバージョンからアップグレードを行う場合の手順は、53ページの『カーネルのアップグレード』を参照してください。
- ハードウェアおよびソフトウェアの前提条件に適合していることを確認します。詳細は、31ページの『ハードウェア』および 32ページの『ソフトウェア』を参照してください。MQSeries Adapter Kernel のインストールの検査は、MQSeries をインストールして実行した後に行います。MQSeries Java サポートがインストールされて構成されていることを確認します。

カーネルのインストール

MQSeries Adapter Kernel を Windows システム (Windows NT または Windows 2000)、UNIX プラットフォーム (AIX、HP-UX、または Solaris)、または OS/400 にインストールするには、オペレーティング・システム (OS) に固有の次のステップを行います。

Windows システムの場合:

ステップ 1. インストール・プログラムを次のように始動します。

- ローカル・エリア・ネットワークからインストールする場合は、ディレクトリーを MQSeries Adapter Kernel インストール・ファイルを含むディレクトリーにして、 `install.bat` ファイルを実行します。
- CD-ROM からインストールする場合は、CD-ROM ドライブに MQSeries Adapter Kernel CD-ROM を挿入します。自動実行が使用可能になっていれば、インストール・プログラムは自動的に始動します。自動実行が使用可能になっていなければ、CD-ROM のルート・ディレクトリーにある `install.bat` ファイルを実行してインストール・プログラムを始動します。

注: Windows システムでは、`install.bat` ファイルを、それを実行する前に別の場所にコピーする必要はありません。インストール処理中に、MQSeries Adapter Kernel をどこにインストールするかという選択があります。

ステップ 2. インストール・プログラムが表示するプロンプトに従って進みます。デフォルト (Windows システムでは `C:¥ProgramFiles¥MQAK`) 以外の場所に MQSeries Adapter Kernel をインストールするように選択する場合は、インストールするディレクトリーを相対パス名ではなく、完全修飾パス名で指定してください。

UNIX の場合:

ステップ 1. インストール・プログラムを次のように始動します。

- ローカル・エリア・ネットワークからインストールする場合は、ディレクトリーを MQSeries Adapter Kernel インストール・ファイルを含むディレクトリーにして、 `install.sh` スクリプトを実行します。
- CD-ROM からインストールする場合は、CD-ROM ドライブに MQSeries Adapter Kernel CD-ROM を挿入します。必要であれば、オペレーティング・システムの資料に従って CD-ROM ドライブを装着してください。CD-ROM のルート・ディレクトリーにある `install.sh` スクリプトを実行します。

ステップ 2. インストール・プログラムが表示するプロンプトに従って進みます。デフォルト以外の場所に、MQSeries Adapter Kernel をインストールすることを選択する場合は、インストールするディレクトリーを 相対パス名ではな

く、完全修飾パス名で指定しなければならないことに注意してください。UNIX でのデフォルトのインストール・ディレクトリーは次の通りです。

- AIX: `/usr/lpp/mqak`
- HP-UX: `/MQAK`
- Solaris: `/opt/MQAK`

OS/400 の場合:

ステップ 1. 31ページの『ハードウェア』、33ページのOS/400 ソフトウェア前提条件、および 34ページの『OS/400 インストールの前提条件』に書かれているすべての前提条件に適合していることを確認します。なお、OS/400 に MQSeries Adapter Kernel をインストールする場合は、InstallShield ベースのプログラムを使用してください。InstallShield ベースのプログラムには、AS/400 システムとインターフェースがとれるワークステーションを使用する必要があります。詳細は、34ページの『OS/400 インストールの前提条件』を参照してください。

ステップ 2. Control Language (CL) プロンプトで、**CRTUSRPRF** コマンドを使用して、AS/400 システムに MQAKSRV という名前のユーザー・プロファイルを作成します。

ステップ 3. インストールの実行にリモート AWT を使用するか、または接続クライアント・ワークステーションを使用するかによって、次のステップになります。

- インストールの実行にリモート AWT を使用する場合、次のステップを実行します。
  a. リモート AWT がセットアップされていて動作中であることを確認します。詳細は、34ページの『リモート AWT の使用』を参照してください。
  b. `installAS400.jar` ファイルが AS/400 システムからアクセス可能であることを確認します。そのファイルは、Integrated File System (IFS) の中、または AS/400 システムに接続されたデバイス上になければなりません。そのファイルが接続デバイス上にあるときは、Create Link (**CRTLINK**) コマンドを使用して、そのファイルへのシンボリック・リンクを作成します。
  c. インストール処理のパフォーマンスを向上させるには、`installAS400.jar` ファイルに対して、Create Java Program (**CRTJVAPGM**) コマンドを使用します。
  d. Run Java (**RUNJVA**) コマンドを次のように実行します。ここで、*n.n.n.n* は、リモート AWT を実行しているワークステーションの TCP/IP のアドレスを表します。

```
RUNJVA CLASS(run)
CLASSPATH('/installAS400.jar')
PROP((os400.class.path.rawt 1) (RmtAwtServer 'n.n.n.n ')
(java.version 1.2))
```

- インストールの実行に接続クライアント・ワークステーションを使用する場合、次のステップを実行します。

  ステップ a. 35ページの『接続クライアントの使用』で指定された要件を満たしていることを確認します。

  ステップ b. AS/400 マシンに Host Servers オプションがインストールされていて、動作中であることを確認します。CL プロンプトで (**STRHOSTSVR**) コマンドを使用して、Host Servers を始動できます。

  ステップ c. AS/400 マシンに TCP/IP がインストールされていて、動作中であることを確認します。TCP/IP を開始するには、CL プロンプトで Start TCP/IP (**STRTCP**) コマンドを使用します。

  ステップ d. ワークステーションでは、コマンド・プロンプトをオープンして、ディレクトリーを MQSeries Adapter Kernel インストール・メディア (ローカル・エリア・ネットワークまたは CD-ROM のいずれか) の AS400 ディレクトリーにします。

  ステップ e. 次のコマンドを入力します。

  ```
  java -classpath installAS400.jar; run -os400
  ```

ステップ 4. インストール・プログラムが始動し、**「Signon to AS/400 (AS/400 へのサインオン)」**パネルが表示されます。**「System: (システム:)」**フィールドに AS/400 マシンの TCP/IP アドレスを入力し、ユーザー ID とパスワードを該当するフィールドに入力します。 **Default User** チェック・ボックスはチェックしないでください。**「次へ」**をクリックします。

ステップ 5. インストール・プログラムが表示するプロンプトに従って進みます。ネットワークとマシンの速度によっては、インストール処理の完了までに最大 1 時間かかる場合があります。ワークステーションに表示される進行表示バーは、インストールの状況を示しています。

なお OS/400 では、MQSeries Adapter Kernel は常に Integrated File System (IFS) のルートの /QIBM/ProdData/mqak ディレクトリーにインストールされます。

ステップ 6. CLASSPATH、PATH、および QIBM_MULTI_THREADED 環境変数を次のように設定します。

- CLASSPATH 環境変数に /QIBM/ProdData/mqak/bin ディレクトリーを追加します。

- PATH 環境変数に /QIBM/ProdData/mqak/bin ディレクトリーを追加します。
- QIBM_MULTI_THREADED 環境変数を Y に設定します。

ステップ 7. QSYS.LIB ライブラリー・リストにライブラリー MQAK を追加します。

これで、カーネルのインストールは完了です。カーネルは、インストールされた時点では、検査をサポートするように構成されており、実際の実動をサポートするようには構成されていません。46ページの『インストールの検査』で述べられているステップを実行して、インストールの検査を行ってください。インストールを検査した後、『インストール後のステップを完了』のステップに従って環境変数を設定し、現場での実動に合うように構成ファイルをいくつか移動します。

必要に応じて他のコンピューターにカーネルをインストールします。

インストール後のステップを完了

カーネルをインストールした後、次のステップを行います。

ステップ 1. aqmsetup および aqmconfig.xml ファイルをどこに置くかを決めます。これらのファイルは、カーネルを構成するのに使用します。これらのファイルの詳細については、58ページの『カーネルの構成』 を参照してください。

注意:
構成ファイルを作成しないで、samples ディレクトリーにある構成ファイルをその代わりに実動に使用する場合、新しいバージョンのカーネルをインストールすると、それらは上書きされて実動用の構成は破棄されます。

ステップ 2. 2 つの構成ファイル用にディレクトリーを作成します。 2 つの構成ファイルは必ずしも同じディレクトリーに置く必要はありませんが、簡単にするために同じディレクトリーに置くことをお勧めします。それらの構成ファイルを MQSeries Adapter Kernel をインストールしたディレクトリー以外に置くと、後でカーネルをアンインストールしたときに、いくつかのディレクトリーが残ることになります。アンインストール処理では、オリジナルの MQSeries Adapter Kernel ファイル以外のものを含むディレクトリーは残されます。

ステップ 3. aqmsetup と aqmconfig.xml ファイルを、 samples ディレクトリーから任意の場所にコピーします。それらのファイルをネットワーク・ドライブや他の中央設置場所に置くと、多数のコンピューターからアクセスできるので、それらのファイルの更新やバックアップが容易になります。

aqmconfig.xml ファイルをリネームすると、カーネルは正常に動作しません。 aqmsetup ファイルは、Step 45ページの5の中で正しくそのファイルを指すように環境変数を設定すると、リネームすることができます。

ステップ 4. テキスト・エディターを使って aqmsetup ファイルを編集して、aqmconfig.xml ファイルを置いた任意のディレクトリーを指すようにします。ディレクトリーの場所として、相対パス名ではなく、完全修飾パス名を使用してください。パスにファイル名そのものは含めません。たとえば、次のようになります。

```
# Location of configuration file aqmconfig.xml.
AQMCONFIG=C:¥Program Files¥MQAK¥Data¥
```

aqmconfig.xml ファイルを置く場所が、 aqmsetup ファイルがあるディレクトリーと同じときでも、ここで完全修飾パス名を入力する必要があります。 aqmsetup ファイルを保存してクローズします。

ステップ 5. AQMSETUPFILE 環境変数を設定し aqmsetup ファイルの場所を指します (たとえば、Windows システムでは C:¥Program Files¥MQAK¥Data¥aqmsetup、UNIX では /MQAK/data/aqmsetup、または OS/400 では /home/*user_name*/aqmsetup です)。OS/400 では aqmsetup ファイルは常にユーザーの現行ホーム IFS ディレクトリー (つまり、/home/*user_name*) に置かなければならないことに注意してください。

カーネルをネットワーク・ドライブにインストールしている場合は、それにアクセスするコンピューターごとにこのステップを実行します。

ステップ 6. AIX を使用していて、 C プログラムから呼び出されるネイティブ C 言語ソース・アダプターを使用する計画の場合は、 AIXTHREAD_SCOPE 環境変数の値を S に設定します。この環境変数を Bourne シェルまたは Korn シェルで設定するには、次のコマンドを入力します。

```
export AIXTHREAD_SCOPE=S
```

この環境変数を C シェルで設定するには、次のコマンドを入力します。

```
setenv AIXTHREAD_SCOPE S
```

AIX にログインするときに AIXTHREAD_SCOPE 変数が自動的に設定されるようにするには、このコマンドを .profile ファイル (Bourne シェルまたは Korn シェルを使用の場合)、または .cshrc ファイル (C シェルを使用の場合) に追加します。

スケジューリング・ポリシーの追加情報は、23ページの7 を参照してください。

ステップ 7. 必要であれば、THREADS_FLAG 環境変数を設定します。この変数は、次の条件がすべて合致しているときにのみ設定する必要があります。

- 使用しているオペレーティング・システムは Solaris である。
- 使用している Java Development Kit (JDK) のバージョンは 1.2.2 である。
- メッセージのトランスポートに MQSeries を使用している。

- ソース・アダプターとターゲット・アダプターは C で書かれている。

これらの条件にすべて合致している場合は、THREADS_FLAG 環境変数を native に設定します。この環境変数を Bourne シェルまたは Korn シェルで設定するには、次のコマンドを入力します。

```
export THREADS_FLAG=native
```

この環境変数を C シェルで設定するには、次のコマンドを入力します。

```
setenv THREADS_FLAG native
```

Solaris にログインするときに THREADS_FLAG 変数が自動的に設定されるようにするには、このコマンドを .profile ファイル (Bourne シェルまたは Korn シェルを使用する場合)、または .cshrc ファイル (C シェルを使用する場合) に追加します。

インストール後のステップを完了した後、カーネルを使用する準備のために次のステップを行います。

1. 実動の準備をします。 57ページの『実動の準備』を参照してください。
2. 構成ファイルを編集します。詳細は、58ページの『カーネルの構成』を参照してください。
3. MQSeries とオプション・ソフトウェアを構成します。89ページの『MQSeries および MQSeries Integrator の構成』を参照してください。
4. 実動システムでは、89ページの『パフォーマンスの推奨事項』を考慮してください。
5. カーネルを始動します。 90ページの『カーネルの始動』を参照してください。
6. カーネルの保守計画をセットアップします。 92ページの『カーネルの保守』を参照してください。

インストールの検査

カーネルをインストールした後に、検査スクリプトを実行して、正しくインストールできたことを検査します。スクリプトは、ソース・アプリケーションからソース・アダプターを使い、その後カーネルを使って MQSeries に、テスト・メッセージを送信します。次にカーネルを使って MQSeries からメッセージを受信してから、ターゲット・アダプターを起動します。これらの処理のすべては 1 台のコンピューターで実行します。

この検査に使用するソース・アプリケーションは、TEST1 という MQSeries キューです。ターゲット・アプリケーションは、TEST2 という別の MQSeries キューです。

検査では次の作業を行います。

- カーネルが、提供されているソース・アダプターとターゲット・アダプターとともにテスト・メッセージの経路指定を、コンピューター内のエンドツーエンドで正しく行ったことを検査します。その際、メッセージング・ソフトウェアに MQSeries を使用します。
- インストール時に指定した aqmconfig.xml ファイルと aqmsetup ファイルを検査します。それらはカーネルの構成を決定するのもです。それらのファイルについての情報は、58ページの『カーネルの構成』を参照してください。

構成ファイルは、検査した後に実動で使用してください。 85ページの『構成ファイルの妥当性検査』を参照してください。

MQSeries Adapter Kernel に備わっているインストール検査スクリプトは、それを実行するマシンに MQSeries がインストールされ、構成されていることを前提としています。MQSeries 以外のメッセージング・ソフトウェアを使用する場合は、そのメッセージング・ソフトウェアをサポートするために、インストール検査スクリプトを次のように編集します。

1. ディレクトリーを、カーネル・インストールの verification ディレクトリーにします。
2. テキスト・エディターで aqmconfig.xml ファイルをオープンして、<epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr> の行を <epicmqppqueuemgr>*queue_manager_name*</epicmqppqueuemgr> に変更します。ここで *queue_manager_name* は、使用するキュー・マネージャーの名前です。
3. aqmverifyinstall ファイルを次のように編集します。
   - インストール検査を Windows システムで行う場合は、テキスト・エディターで aqmverifyinstall.bat ファイルをオープンし、aqmcreateq TEST2 の行を aqmcreateq TEST2 *queue_manager_name* に変更します。ここで *queue_manager_name* は、使用するキュー・マネージャーの名前です。
   - インストール検査を UNIX または OS/400 で行う場合は、テキスト・エディターで aqmverifyinstall.sh ファイルをオープンし、aqmcreateq.sh TEST2 の行を aqmcreateq.sh TEST2 *queue_manager_name* に変更します。ここで *queue_manager_name* は、使用するキュー・マネージャーの名前です。

この検査では、ターゲット・アダプター名 com.ibm.epic.adapters.eak.test.InstallVerificationTest のようなコンポーネントをいくつか使用しますが、これらはカーネルの一部ではありません。それらのコンポーネントは、インストール検査のみを目的としてカーネルが備えているものです。

検査を完了すると、検査のアダプター・デーモンは停止します。

検査の間はトレースは使用できません。

検査手順

ステップ 1. 検査では 3 つの MQSeries キューを作成して使用します。検査を実行する前にそれらのキューにメッセージがあると、検査は失敗します。次のキューにあるメッセージをクリアしてください。

- TEST2AIQ
- TEST2AEQ
- TEST2RPL

ステップ 2. カーネルのインストールと検査を行う権限を与えられていることを確認します。40ページの『インストールの準備』を参照してください。

ステップ 3. 次のように検査を開始します。

- Windows システムでは、 verification ディレクトリーの aqmverifyinstall.bat ファイルをダブルクリックします。または、コマンド・プロンプトをオープンし、ディレクトリーを verification に変更して、aqmverifyinstall.bat を実行します。
- UNIX では、ターミナルをオープンし、ディレクトリーを verification にして、 aqmverifyinstall.sh ファイルを実行します。
- OS/400 では、次のステップを実行します。
  a. **STRQSH** コマンドを入力して、**qsh** セッションを開始します。
  b. ホーム・ディレクトリー (/home/*user_name*) に /QIBM/ProdData/mqak/verification/aqmsetupファイルをコピーします。
  c. ディレクトリーを /QIBM/ProdData/mqak/verification にします。
  d. aqmverifyinstall.sh ファイルを実行します。

aqmverifyinstall ファイルには、その機能についての説明が含まれています。

ステップ 4. Installation Verification Test completed successfully というメッセージは、成功を表します。必要であれば検査ウィンドウをクローズします。

ステップ 5. 失敗の場合は、検査ウィンドウとログ・ファイルの EpicSystemExceptionFile*nnnnnnnn*.log を調べて、エラーを判別します。

ステップ 6. 検査時に発生することがある共通の問題および考えられる対応については、49ページの『検査の共通問題』を参照してください。

ステップ 7. 必要であれば、オプションの検査を行います。詳細は、51ページの『検査オプション』を参照してください。

ステップ 8. インストール手順に戻り、実際の現場での運用に合わせてカーネルを構成します。 44ページの1 のステップに進んでください。

検査の共通問題

このセクションでは、検査時に見つかることがある共通の問題と、考えられる対応を取り上げます。例外メッセージの中の重要な情報は、**太字**で強調しています。

問題: aqmsetup ファイルが見つからなかった。

対応: AQMSETUPFILE 環境変数が、 verification ディレクトリーの aqmsetup ファイルの場所に設定されていることを確認します。

例外メッセージ:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterDirectory::getProperties():
Received exception <com.ibm.epic.adapters.eak.common.AdapterException>
Message information: <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterCfg::readConfig(String):
Received exception <java.io.FileNotFoundException> Message information:
<C:¥aqmsetup> Additional program information <>.>
Additional program information <Error Reading Configuration File
[File or Keys in file may not exist]>.>
```

問題: aqmconfig.xml ファイルが見つからなかった。

対応: verification ディレクトリーにある aqmsetup ファイルを編集して、AQMCONFIG= エントリーが verification ディレクトリーを指していることを確認します。この場合、完全修飾パス名を使用してください。また、aqmconfig.xml ファイルが verification ディレクトリーにあることも確認してください。

例外メッセージ:

```
com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.common.AdapterDirectory::
getProperties(): Received exception
<java.io.FileNotFoundException> Message information:
<AQMCONFIG.xml> Additional program information <>.>
```

問題: メッセージを書き込むキューがなかった。

対応: MQSeries を使用して、例外メッセージの中の名前のキュー (インストールを検査中は TEST2AIQ) が存在していて、メッセージを受信できることを確認します。 94ページの『MQSeries キューの作成』を参照してください。

例外メッセージ:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message
<AQM0107: com.ibm.epic.adapters.eak.nativeadapter.LMSMQbase::
createMQOutputQueue(String):
Received MQException creating queue, QManager name <DEFAULT>
Queue name <TEST2AIQ>:
completion code <2> reason code <2085>.>
```

問題: ターゲット・アダプターが見つからなかった。

対応: メッセージで指定したターゲット・アダプター
com.ibm.epic.adapters.eak.test.InstallVerificationTest があることを確認します。 CLASSPATH 環境変数がカーネルの bin ディレクトリーを含んでいることを確認します。

例外メッセージ:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2:
Message <<TEST2> <2000.05.18.09.41.43.781> <<Processing Messages.>
<com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker:
:instantiateClass(String, Class[], Object[]): Received exception
<java.lang.ClassNotFoundException> Message information:
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>
Additional program information <[Cannot obtain Class for class name
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>]>.>>>>
```

問題: メッセージの送達の際に、ロードするアダプターが見つからなかった。キューにあるメッセージで指定される本体タイプと本体カテゴリーへの宛先論理 ID のエントリーが、 aqmconfig.xml ファイルにありません。

対応: 検査の間のこの例外メッセージの原因として最も考えられるのは、検査の前に TEST2AIQ という名前のキューにメッセージが存在していたことです。 TEST2AIQ キューにあるメッセージをすべてクリアして、検査を再試行してください。 verification ディレクトリーの aqmconfig.xml ファイルにあるアプリケーション TEST2 のコマンド・クラス名へのエントリーは、本体タイプ TESTBOD と 本体カテゴリー OAG へのエントリーだけです。

例外メッセージ:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2: Message <<TEST2> <2000.05.18.10.28.43.105>
<<Processing Messages.> <com.ibm.epic.adapters.eak.common.
AdapterException:
MessageID <AQM0401> <AQM0401: com.ibm.epic.adapters.eak.
adapterdaemon.EpicAdapterWorker::processMessage(EpicMessage):
Cannot obtain Command class name to load for a received message.>>>>
```

問題: 検査のキュー・マネージャーが始動しなかった。

対応: デフォルトの MQSeries キュー・マネージャーが正常に始動したことを確認します。

例外メッセージ:

```
com.ibm.epic.adapters.eak.common.AdapterException: Message ID <AQM0104>
<AQM0104: com.ibm.epic.adapters.eak.nativeAdapter.queueCollection::
constructor(String,String,boolean,String,String,int):
```

Received MQException creating QManager connection for QManager name <QMGRNAME>
MQ Message information: completion code <2> reason code <2059>.>

問題: MQSeries の一般的なエラーが発生した。

対応: マシンに MQSeries がインストールされ、正しく構成されて動作中であることを確認します。 MQException 理由コードを調べ、資料の *MQSeries Messages* を使用して理由コードの原因を判別してください。

例外メッセージ:

Received MQException "*ACTION ATTEMPTED*." Message information: completion code <*completion_code*> reason code <*reason_code*>

検査オプション

最初のコンピューターにカーネルが正しくインストールされたことを検査した後、オプションで次のステップを行うことができます。

1. 同じ検査手順で、 2 番目のコンピューターにカーネルが正しくインストールされていることを検査します。
2. あるコンピューターのソース・アダプターから別のコンピューターのターゲット・アダプターに、テスト・メッセージを送信できることを確認します。この検査は手操作で構成して実行します。カーネルが備えているオリジナルの検査ファイルを変更してこの検査を行う場合は、オリジナルの検査ファイルのコピーをバックアップのために保存してください。

サイレント・インストールの使用法

MQSeries Adapter Kernel は、*サイレント・インストール*を使用して、あらゆるプラットフォームにインストール可能です。サイレント・インストールにより、希望のオプションを自分で選択しなければならない MQSeries Adapter Kernel インストール・プログラムをう回することができます。複数マシンにデフォルトの構成をインストールしたい場合に、サイレント・インストールは役に立ちます。

カーネルをサイレントにインストールするためには、オペレーティング・システム (OS) 固有の次のステップを行います。

Windows システムの場合:

ステップ 1. コマンド・プロンプトをオープンして、MQSeries Adapter Kernel インストール・ファイルを含むディレクトリーにします。

ステップ 2. 次のコマンドを入力します。

```
java -cp install.jar run -P product.installLocation="install_location"
-silent
```

ここで、*install_location* は希望するインストール場所です (たとえば、D:¥mqak)。

UNIX の場合:

ステップ 1. 端末で、MQSeries Adapter Kernel インストール・ファイルを含むディレクトリーにします。CD-ROM からインストールする場合は、CD-ROM ドライブに MQSeries Adapter Kernel CD-ROM を挿入します。必要であれば、オペレーティング・システムの資料に従って CD-ROM ドライブを装着してください。

ステップ 2. 次のコマンドを入力します。

```
java -cp install.jar run -P product.installLocation="install_location"
-silent
```

ここで、*install_location* は希望するインストール場所です (たとえば、/opt/mqak)。

OS/400 の場合:

AS/400 マシンへのアクセスに接続クライアントを使用しない場合、次のステップを行います。

ステップ 1. installAS400.jar ファイルが AS/400 システムからアクセス可能であることを確認します。そのファイルは、Integrated File System (IFS) の中、または AS/400 システムに接続されたデバイス上になければなりません。そのファイルが接続デバイス上にあるときは、Create Link (**CRTLINK**) コマンドを使用して、そのファイルへのシンボリック・リンクを作成します。

ステップ 2. インストール処理のパフォーマンスを向上させるには、installAS400.jar ファイルに対して、Create Java Program (**CRTJVAPGM**) コマンドを使用します。

ステップ 3. CL プロンプトか、または **qsh** セッションを使用しているかに従って、次のいずれかのコマンドを 入力します。

- CL プロンプトを使用している場合、次のコマンドを入力します。

  ```
  RUNJVA CLASS(run)
  CLASSPATH('/installAS400.jar')
  PROP((java.version 1.2)) PARM('-silent')
  ```

- **qsh** セッションを使用している場合、次のコマンドを入力します。

  ```
  java -Djava.version=1.2 -classpath installAS400.jar run -silent
  ```

AS/400 システムとのインターフェースに接続クライアントを使用している場合、次のステップを行います。

ステップ 1. 35ページの『接続クライアントの使用』で指定された要件を満たしていることを確認します。

ステップ 2. AS/400 マシンに Host Servers オプションがインストールされていて、動作中であることを確認します。Host Servers を開始するには、Control Language (制御言語)(CL) プロンプトで、Start Host Servers (**STRHOSTSVR**) コマンドを使用します。

ステップ 3. AS/400 マシンに TCP/IP がインストールされていて、動作中であることを確認します。TCP/IP を開始するには、CL プロンプトで Start TCP/IP (**STRTCP**) コマンドを使用します。

ステップ 4. ワークステーションでは、コマンド・プロンプトをオープンして、ディレクトリーを MQSeries Adapter Kernel インストール・メディア (ローカル・エリア・ネットワークまたは CD-ROM のいずれか) の `AS400` ディレクトリーにします。

ステップ 5. 次のコマンドを入力します。

```
java -cp installAS400.jar run -silent -os400 machine_name user_ID password
```

ここで、*machine_name* は AS/400 システムの TCP/IP アドレス、*user_ID* はユーザー ID、そして *password* はパスワードです。

カーネルのアップグレード

MQSeries Adapter Kernel バージョン 1.0 のインストールに、修正サービス・ディスケット (Corrective Service Diskette)(CSD) を使った場合も、使わなかった場合も、または以前の修正レベルの MQSeries Adapter Kernel バージョン 1.1 をインストールした場合、現行修正レベルの MQSeries Adapter Kernel バージョン 1.1 をインストールする前に次のステップを行います。

ステップ 1. MQSeries Adapter Kernel インストール・ディレクトリー以外の場所に `aqmsetup` および `aqmconfig` (`aqmconfig.properties` または `aqmconfig.xml`) ファイルをバックアップします。

ステップ 2. MQSeries Adapter Kernel CSD がインストールされている場合は、それを次のようにアンインストールします。

- Windows NT では、次の方法のいずれかで行います。
  - Windows NT の「スタート」メニューから、**「プログラム」>「MQSeries Adapter Kernel」 >「Remove CSD (CSD の除去)」** をクリックします。
  - 「コントロール・パネル」の「アプリケーションの追加と削除」ユーティリティーを使用します。
  - カーネルのルート・ディレクトリーにある `aqmuninstallCSD.bat` ファイルを実行します。
  - コマンド・プロンプトをオープンし、ディレクトリーをカーネルのルート・ディレクトリーにして、次のコマンドを入力します。

```
java uninstallCSD
```

- AIX では、ディレクトリーをカーネルのルート・ディレクトリーにして、次のいずれかのコマンドを入力します。

  ```
  aqmuninstallCSD.sh
  ```

  ```
  java uninstallCSD
  ```

ステップ 3. 次のように MQSeries Adapter Kernel をアンインストールします。

- Windows NT では、次の方法のいずれかで行います。
  - Windows NT の「スタート」メニューから、**「プログラム」>「MQSeries Adapter Kernel」> 「Uninstall MQSeries Adapter Kernel (MQSeries Adapter Kernel のアンインストール)」** をクリックします。
  - 「コントロール・パネル」の「アプリケーションの追加と削除」ユーティリティーを使用します。
  - カーネルのルート・ディレクトリーにある `aqmuninstall.bat` ファイルを実行します。
  - コマンド・プロンプトをオープンし、ディレクトリーをカーネルのルート・ディレクトリーにして、次のコマンドを入力します。

    ```
    java uninstall
    ```

- AIX では、ディレクトリーをカーネルのルート・ディレクトリーにして、次のいずれかのコマンドを入力します。

  ```
  aqmuninstall.sh
  ```

  ```
  java uninstall
  ```

ステップ 4. MQSeries Adapter Kernel バージョン 1.1 をインストールします。詳細は、41ページの『カーネルのインストール』を参照してください。

ステップ 5. `aqmsetup` および `aqmconfig` ファイルを、MQSeries Adapter Kernel をインストールしたディレクトリーの以前の場所にリストアします。必要な場合、`aqmconfig.properties` ファイルを、`aqmconfig.xml` ファイルに変換します。`aqmconfig.xml` ファイルについての詳細は、64ページの『構成ファイル』を参照してください。

カーネルの除去

カーネルを除去するには、いくつかの方法があります。アンインストール処理では、カーネルをインストールした後に作成されたファイルやディレクトリーは除去されないことに注意してください。すべてのログ・ファイルおよびユーザーがコピーしたデータ・ファイルも同様です。

- Windows システムでは、次の方法のいずれかで行います。

- 「スタート」メニューから、**「プログラム」>「IBM MQSeries Adapter Kernel」> 「Uninstall MQSeries Adapter Kernel (MQSeries Adapter Kernel のアンインストール)」** をクリックします。
- 「コントロール・パネル」の「アプリケーションの追加と削除」ユーティリティーを使用します。
- カーネルのルート・ディレクトリーにある `aqmuninstall.bat` ファイルを実行します。
- カーネルをサイレントに (つまり、アンインストール・プログラムが詳細や確認のプロンプトを出すことなしに) アンインストールするには、コマンド・プロンプトをオープンしカーネルのインストール・ディレクトリーにして、次のコマンドを入力します。

  ```
  java -cp uninstall.jar run -silent
  ```

- UNIX では、ディレクトリーをカーネルのルート・ディレクトリーにして、次のコマンドを入力します。

  ```
  aqmuninstall.sh
  ```

  カーネルをサイレントに (つまり、アンインストール・プログラムが詳細や確認のプロンプトを出すことなしに) アンインストールするには、カーネルのルート・ディレクトリーにして、次のコマンドを入力します。

  ```
  java -cp uninstall.jar run -silent
  ```

- OS/400 では、次の方法のいずれかでアンインストールを行います。
  - カーネルのアンインストールにリモート AWT を使用する場合、次のステップを行います。

    ステップ 1. リモート AWT がセットアップされていて動作中であることを確認します。詳細は、34ページの『リモート AWT の使用』を参照してください。

    ステップ 2. アンインストール処理のパフォーマンスを向上させるには、`/QIBM/ProdData/mqak/uninstall/uninstall.jar` ファイルに対して、Create Java Program (**CRTJVAPGM**) コマンドを使用します。

    ステップ 3. Run Java (**RUNJVA**) コマンドを次のように実行します。ここで、*n.n.n.n* は、リモート AWT を実行しているワークステーションの TCP/IP のアドレスを表します。

    ```
    RUNJVA CLASS(run)
    CLASSPATH('/QIBM/ProdData/mqak/uninstall/uninstall.jar')
    PROP((os400.class.path.rawt 1) (RmtAwtServer 'n.n.n.n')
    (java.version 1.2))
    ```

  - カーネルのアンインストールに接続クライアント・ワークステーションを使用する場合、次のステップを行います。

    ステップ 1. 35ページの『接続クライアントの使用』で指定された要件を満たしていることを確認します。

ステップ 2. AS/400 マシンに Host Servers オプションがインストールされていて、動作中であることを確認します。Host Servers を開始するには、Control Language (制御言語)(CL) プロンプトで、Start Host Servers (**STRHOSTSVR**) コマンドを使用します。

ステップ 3. AS/400 マシンに TCP/IP がインストールされていて、動作中であることを確認します。 TCP/IP を開始するには、 CL プロンプトで Start TCP/IP (**STRTCP**) コマンドを使用します。

ステップ 4. AS/400 システムの `/QIBM/ProdData/mqak/uninstall` ディレクトリーのファイル `uninstall.jar` と `uninstall.dat` を、クライアント・ワークステーションのディレクトリーにコピーします。

ステップ 5. 次のコマンドを入力します。

```
java -classpath uninstall.jar; run -os400
```

カーネルをサイレントに (つまり、アンインストール・プログラムが詳細や確認のプロンプトを出すことなしに) アンインストールするには、次のコマンドを入力します。

```
java -cp uninstall.jar run -silent -os400 machine_name user_ID password
```

ここで、*machine_name* は AS/400 システムの TCP/IP アドレス、*user_ID* はユーザー ID、そして *password* はパスワードです。

– 直接 AS/400 システムの CL プロンプト または **qsh** セッションで作業していて、カーネルをサイレントに (つまり、アンインストール・プログラムが詳細や確認のプロンプトを出すことなしに) アンインストールしたい場合、次のいずれかのコマンドを入力します。

- CL プロンプトの場合。

```
RUNJVA CLASS(run)
CLASSPATH('/uninstall.jar')
PROP((java.version 1.2)) PARM('-silent')
```

- **qsh** セッションの場合。

```
java -Djava.version=1.2 -classpath uninstall.jar run -silent
```

第4章 カーネルの使用

この章には、カーネルの使用についての次の情報があります。

- 『実動の準備』
- 58ページの『カーネルの構成』
- 89ページの『MQSeries および MQSeries Integrator の構成』
- 90ページの『カーネルの始動』
- 92ページの『カーネルの停止』
- 92ページの『カーネルの保守』
- 92ページの『問題の診断』

実動の準備

カーネルを実動作業に使用する前に、次の作業を行います。

1. 実際の現場の要件と状態を基に、MQSeries Adapter Offering、MQSeries または他のメッセージング・ソフトウェア、およびオプションの MQSeries Integrator などの総合的なシステム体系を設計します。アーキテクチャーは、現場ごとに固有のものになるのが普通です。
2. MQSeries Adapter Builder を使用して、必要なソース・アダプターとターゲット・アダプターを作成し、それらのテストと配置を行います。
3. 次の目的のために、MQSeries Adapter Offering の範囲外のアプリケーション固有のインターフェースを開発します。
   - ソース・アダプターが、ソース・アプリケーションのアプリケーション・データを獲得できるようにする。
   - ターゲット・アプリケーションが、ターゲット・アダプターのメッセージ・データを獲得できるようにする。

   アプリケーション固有のインターフェースそのものの実態は、ソース・アプリケーションとターゲット・アプリケーションの特性に依存します。アプリケーション固有のインターフェースには、たとえば次のものがあります。
   - API 呼び出しとユーザー出口
   - ファイルの読み取りと書き込み
   - データベース・トリガー
   - メッセージ・キュー
4. ランタイム・フロー (送信、ルーティング、トレース、送達メッセージ) に合わせてカーネルを構成します。カーネルの構成についての情報は、58ページの『カーネルの構成』を参照してください。

5. 実際の総合的なシステム体系に合うように、MQSeries または他のメッセージング・ソフトウェア、およびオプションで MQSeries Integrator を構成します。89ページの『MQSeries および MQSeries Integrator の構成』を参照してください。
6. 必要であれば、メッセージ送達に合わせて Java ログオン・クラスを開発します。そのクラスは、それぞれのターゲット・アプリケーションに固有のものです。そのクラスは、ターゲット・アダプターが、ログオンおよびアプリケーションとの接続の情報を要求するときのみ必要なものです。
7. システムを実動作業に使用する前に、システム全体をテストします。システム全体とは、実際のソース・アダプターとターゲット・アダプター、実際のアプリケーション固有のインターフェース、および実際のカスタム・コードを用いた MQSeries Adapter Kernel のことです。
8. 実稼働環境にシステムを配置します。
9. 1 つまたは複数のアダプター・デーモン、およびオプションでトレース・サーバーを始動して、カーネルをオンにします。ソース・アプリケーションが始動していることを確認します。ソース・アダプターがソース・アプリケーションのプロセスで実行されている場合は、ソース・アダプターはソース・アプリケーションによって自動的に始動されます。この場合、ソース・アダプターの始動のために特別なステップは必要ありません。ソース・アダプターを含んでいるデーモンまたはサーバーを始動する必要があります。90ページの『カーネルの始動』を参照してください。

カーネルの構成

このセクションでは、ユーザー環境に合うようなカーネルの構成方法を説明します。『構成の概要』では、カーネル構成の考え方について概説しています。63ページの『始動および構成に関連するファイル』では、MQSeries Adapter Kernel 構成を一緒に定義するさまざまなファイルについて説明しています。64ページの『セットアップ・ファイル』では、カーネルの初期設定をいくつか定義する aqmsetup ファイルについて説明しています。64ページの『構成ファイル』では、aqmconfig.xml ファイルについて説明しています。このファイルはカーネルに、ソース・アプリケーションとターゲット・アプリケーションの名前、ソース・アダプターとターゲット・アダプター、キューとキュー・マネージャー、通信モード、ロギング仕様とトレース仕様、などの固有の構成情報を提供します。

構成の概要

このセクションでは、カーネル構成の考え方について概説しています。カーネルを構成する前にカーネルのランタイム・フローを理解することが大切です。このセクションでは、単純化したレベルでランタイム・フローを説明します。ランタイム・フローの詳細は、14ページの『ランタイム・フロー』を参照してください。

最も基本的なレベルでは、MQSeries Adapter Kernel の構成は、アプリケーション間を流れるデータ主導型です。構成する際、次の要因も考慮に入れなければなりません。

- データを受信するアプリケーション。
- ソース・サイドで必要なアダプター、そしてターゲット・サイドで必要なターゲット・アダプター、アダプター・デーモン、およびワーカー。
- 使用する通信モード、マーシャル・フォーマット、およびトランスポート機構。

構成内のそれぞれのアプリケーションごとに、データ構造とデータ・フォーマットは異なります。たとえば、ある構成に A と B の 2 つのアプリケーションが含まれていて、それぞれがアプリケーション C に購入オーダーのデータを送る場合、アプリケーション A のデータは、アプリケーション B のデータとは、フォーマットが異なり、タグの意味が違っていることがよくあります。アプリケーション C が 2 つの異なるアプリケーションからの 2 つの異なるデータ・フローを認識し、構文解析する必要をなくすために、それぞれのアプリケーションのデータが*統合メッセージ* に変換されます。これは、*統合中立フォーマット*のものです。通常、この統合中立フォーマットは、XML を基にした業界標準です。アプリケーション C が認識し構文解析できるようになる必要がある唯一のデータは、統合中立フォーマットです。

図3 は、アプリケーション A および B から、アプリケーション C および D へのデータのフローを示しています。この図を追ってみると、ここで示しているさまざまなデータ・フローが分かります。

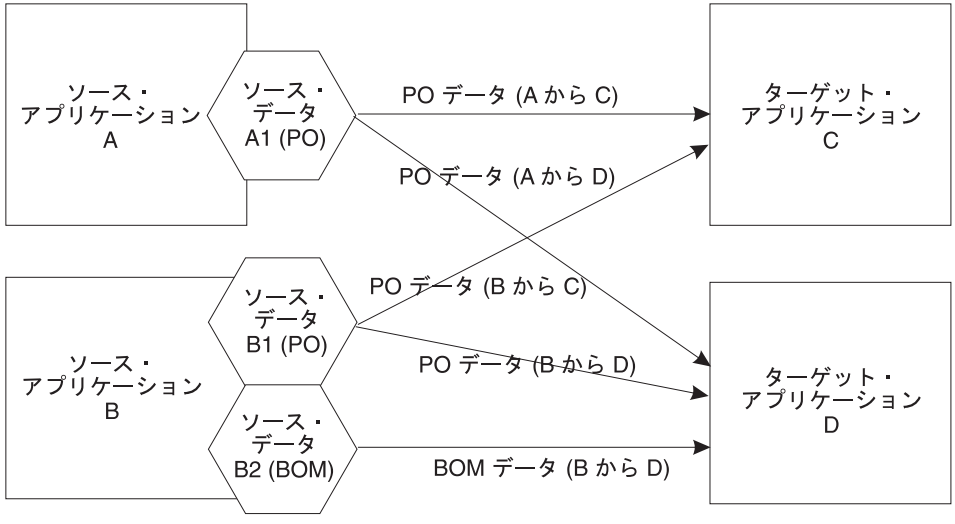


図 3. 単純構成のデータ・フローで接続されたアプリケーション

図3 では、アプリケーション A からの購入オーダー・データがアプリケーション C および D に流れ、アプリケーション B からの購入オーダー・データもアプリケーション C と D に流れています。そして、アプリケーション B からの材料の請求書データは、アプリケーション D だけに流れています。それぞれの場合、データはターゲット・アプリケーションに送信される前に、業界標準フォーマットに変換されます。アプ

リケーション A および B から来る購入オーダー・データの場合、データは購入オーダー・データを示す標準 XML フォーマットに変換されます。アプリケーション B から来る材料の請求書データの場合、データは材料の請求書データを示す標準 XML フォーマットに変換されます。

MQSeries または Java Message Service (JMS) のインプリメンテーションのような通信トランスポートが使用され、データをターゲット・アプリケーション (単数または複数) へ送信されます。統合メッセージは、特定の通信トランスポートに必要な マーシャル・フォーマット に変換され、その後、通信トランスポート (たとえば、MQSeries キュー) に送達されます。それぞれのターゲット・アプリケーションが、メッセージを受信する異なる通信トランスポートおよびマーシャル・フォーマットを使用できます。たとえば、61ページの図4 に示されるように、アプリケーション C はメッセージを受信するのに MQSeries を使用でき、アプリケーション D は JMS を使用できます。この場合、アプリケーション C に向かう統合メッセージすべて (つまり、アプリケーション A および B からの購入オーダー) は MQSeries マーシャル・フォーマットに変換され、アプリケーション D に向かう統合メッセージすべては (つまり、アプリケーション B からの材料の請求書データだけでなく、アプリケーション A および B からの購入オーダーも) JMS マーシャル・フォーマットに変換されます。 MQSeries Adapter Kernel は、次の機構を使用してこれらの変換を実行します。

- ソース・アダプター を使用して、アプリケーション・データを統合メッセージに変換します。ソース・アダプターは MQSeries Adapter Builder で作成されます。
- ネイティブ・アダプターを使用して、統合メッセージを *通信*メッセージに変換します。ネイティブ・アダプターは *論理*メッセージ・サービス (LMS) を使用して、通信トランスポートの移送用にメッセージを変換します。LMS は使用する通信トランスポートにとって固有です。その後、LMS は フォーマッター を使用してトランスポートへのメッセージのマーシャルを行います。

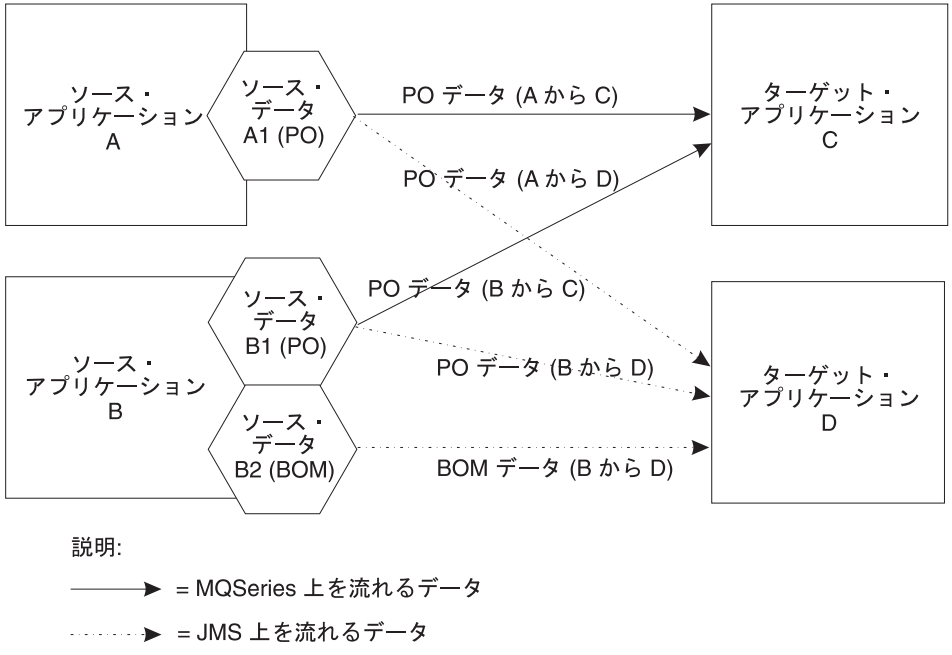


図 4. 単純構成の異なる通信トランスポートで接続されたアプリケーション

アプリケーションから統合メッセージへ、そして通信メッセージへのデータのフローが、図5 および 62ページの図6 に示されています。通信メッセージがターゲットで受信されると、変換が反転します。つまり、ネイティブ・アダプターは通信メッセージを統合メッセージに変換し戻します。その後、必要なら、ターゲット・アダプターは統合メッセージをターゲット・アプリケーションで必要なデータ・フォーマットに変換します。

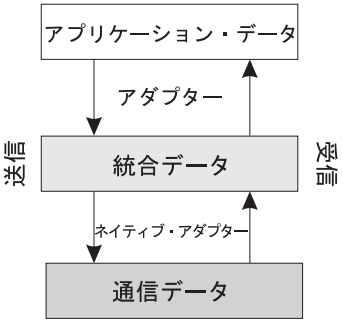


図 5. データの変換

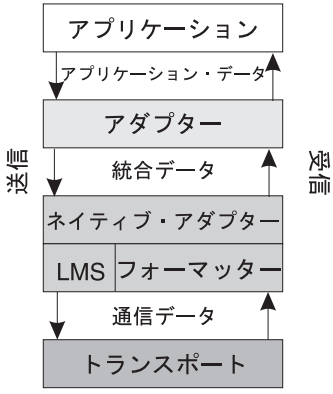


図 6. データのフロー

データが移動するそれぞれのポイントは、カーネルの構成ファイルの中で示されなければなりません。アプリケーション ID (ソースまたはターゲット) に分割される 3 つの論理構成要件があります。メッセージがアプリケーション (ターゲット) に向かっているか、またはアプリケーション (ソース) から来るかによって、アプリケーション ID は、ソース・アプリケーション用、またはターゲット・アプリケーション用になることができます。構成ファイルには、次のタイプの情報が入っていなければなりません。

- 通信
  - データはどこへ行く必要があるか
  - 使用する通信トランスポート
  - 使用する通信マーシャル・メソッド (フォーマッター)
  - MQSeries キュー・マネージャーおよび MQSeries キュー名、または JMS キュー接続ファクトリーおよび JMS キュー名のような、基本となる通信要件
- アダプター (ターゲット・サイド用のみ)
  - データを処理するのに必要なアダプター
  - 使用するタイプのアダプター (EAB または EJB)
  - 使用するアダプター・タイプ固有の追加情報
  - 独立型 MQSeries Adapter Kernel 用のアダプター・デーモンおよびワーカー情報
- その他
  - トレース仕様
  - ロギング仕様

構成の異なる部分に関連するときのデータ・フローが、63ページの図7 に示されています。

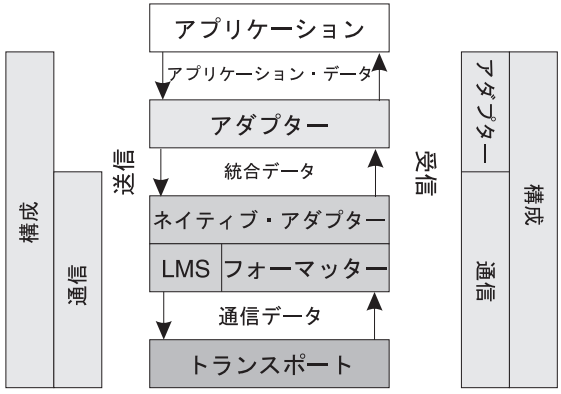


図 7. 構成に関連するデータのフロー

aqmconfig.xml ファイルの XML エレメントへのこれらの構成要件のマッピングについての詳細は、65ページの『構成ファイルの構文と編成』を参照してください。このファイルによって、カーネルの構成の外観が制御されます。76ページの『共通の構成』では、共通の構成をいくつかリストしています。

始動および構成に関連するファイル

カーネルの構成は、いくつかのカスタマイズ可能なファイルによって決まります。標準的なテキスト・エディターを使ってファイルを編集して、実際の現場に合うカーネルを構成します。カーネルを構成するには、次のファイルを使用します。

- aqmsetenv.bat ファイル (Windows システム)、または aqmsetenv.sh ファイル (UNIX)。環境変数を設定するファイルです。必要であればインストール後にこのファイルを編集して、システムの環境変数を変更します。このファイルによって設定される環境変数には PATH、CLASSPATH、および LIBPATH があります。Windows システムでは、これらの変数はインストール・プログラムによって自動的に設定されます。 UNIX にログインするときにこれらの変数を自動的に設定するには、aqmsetenv.sh ファイルで指定した値を .profile ファイル (Bourne シェルまたは Korn シェルを使用する場合) または .cshrc ファイル (C シェルを使用する場合) に追加します。

  OS/400 での適切な環境変数の設定に関する情報は、43ページの6 のステップを参照してください。
- aqmsetup ファイル。このファイルはカーネルのいくつかの初期セットアップ値を規定します。詳細は、64ページの『セットアップ・ファイル』を参照してください。
- aqmconfig.xml ファイル。このファイルはカーネルを構成します。追加情報は、64ページの『構成ファイル』を参照してください。このファイルには、カーネルを構成する値の大部分が含まれています。

- aqmcreateq.bat ファイル (Windows システム)、または aqmcreateq.sh ファイル (UNIX および OS/400) ファイル。このファイルは MQSeries キューを生成するスクリプトです。94ページの『MQSeries キューの作成』を参照してください。

これらのファイルにはすべて、編集時に役立つコメントが含まれています。

これらのファイルのバックアップをとることをお勧めします。追加情報は、92ページの『カーネルの保守』を参照してください。

セットアップ・ファイル

セットアップ・ファイルの aqmsetup で、次のようなカーネルのいくつかの初期設定を制御します。

- 構成ファイルの場所。『構成ファイル』を参照してください。
- XML DTD の場所 (現行ディレクトリーにないとき)。
- C インターフェースのための Java Native Interface (JNI) 環境変数。使用するメモリー総量を変更します。これは、C 実行可能モジュールがプロセスを始動し、Java 仮想マシンがそのプロセスによってインスタンス化される場合に適用されます。この場合は、aqmsetup ファイルにある次の行をコメントにしないで変更することにより、メモリーの使用を制御することができます。

  ```
  #AQM_JNI_NATIVESTACKSIZE=1048576
  #AQM_JNI_JAVASTACKSIZE=4194304
  #AQM_JNI_MINHEAPSIZE=16777216
  #AQM_JNI_MAXHEAPSIZE=268435426
  ```

  サイズはすべてバイト単位です。

aqmsetup ファイルのサンプルが 121ページの『付録E. サンプル・セットアップ・ファイル』にあります。また、インストールした MQSeries Adapter Kernel の samples ディレクトリーにもあります。

MQSeries Adapter Kernel を最初にインストールするときに、必要であればセットアップ・ファイルを編集します。インストール後は、このファイルは、前述の Java のメモリー外の問題がカーネルに発生するときのみ編集してください。

構成ファイル

このセクションでは、カーネルの構成を決定する aqmconfig.xml ファイルについて述べます。65ページの『構成ファイルの構文と編成』では、構成ファイルの構造に関する情報を提供しています。84ページの『構成ファイルの編集』には、構成ファイルを編集する際の実際的な推奨事項があります。

MQSeries Adapter Kernel の構成は、aqmconfig.xml という XML ファイルによって決まります。構成ファイルのサンプルが 115ページの『付録D. サンプル構成ファイル』にあります。また、インストールした MQSeries Adapter Kernel の samples ディレクトリーにもあります。

構成ファイルで指定した値によって、カーネルの次のエレメントが制御されます。

- ソース論理 ID
- 宛先論理 ID
- ターゲット・サイドでのアダプター・デーモンおよびワーカー情報
- トレース・クライアント
- トレース・サーバー
- メッセージのマーシャルとルーティング。これは次の指定によって決まります。
  - 受信キュー、エラー・キュー、および応答キューの名前
  - メッセージが送信される 1 つまたは複数のデフォルト宛先
  - メッセージの取得または送信を行う、MQSeries キュー・マネージャーまたは JMS キュー接続ファクトリーの名前
  - メッセージまたは応答を受信する場合のタイムアウト
  - 各メッセージを処理する、カーネルのターゲット・サイドのターゲット・アダプター・クラス
  - ターゲット・アダプターに固有の追加情報
  - ターゲット・サイドのワーカーの最少数 (独立型 MQSeries Adapter Kernel を動作する場合)
  - トレースの使用可能化と使用不可化、およびトレース・レベルの制御
  - 監査ログの使用可能化と使用不可化
- 通信モード

構成ファイルの構文と編成

MQSeries Adapter Kernel の構成は Lightweight Directory Access Protocol (LDAP) に基づいているため、構成ファイルの構造は LDAP を反映しています。XML の最上位エレメントの Epic は LDAP ディレクトリーの最上位を表し、従属する LDAP オブジェクトは最上位エレメントにネストされた XML エレメントで表されます。XML エレメントには、LDAP 情報を表す必須の属性を持つものがあります。値は、エレメントの内容として、またはエレメントの属性として構成に加えられます。

<epictracelevel>-1</epictracelevel> は、エレメントの内容として割り当てられる構成値の例です。これによって値 -1 (可能なすべてのメッセージ) が、epictracelevel エレメントに割り当てられます。

<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler"> は、エレメ

ントの属性として割り当てられる構成値の例です。これによって com.ibm.logging.ConsoleHandler クラスが、トレース・ハンドラーとして使用されるように割り当てられます。

次に示すのは、構成ファイルに使用する上位エレメントのリストと説明です。68ページの『構成ファイルで使用する XML エレメント』には、構成ファイルに使用するすべてのエレメントを網羅したリストと説明があります。コンテキストでのさまざまなエレメントの使い方の例として、構成ファイルのサンプルを参照してください。

- Epic--aqmconfig.xml ファイルには必須の最上位エレメント。
- ePICApplications--Epic エレメントの必須の子。
- ePICApplication--ePICApplications エレメントの必須の子。カーネルがサービスを行うアプリケーションをリストし、定義します。アプリケーションごとに完全に定義された ePICApplication エレメント (子エレメントを含む) が 1 つ必要です。
- AdapterRouting--ePICApplication エレメントのオプションの子。キュー・マネージャーおよび関連した情報を定義します。
- ePICBodyCategory--AdapterRouting エレメントの必須の子。カーネルによってルーティングされるメッセージの本体カテゴリーを設定します。
- ePICBodyType--ePICBodyCategory エレメントの必須の子。カーネルによってルーティングされるメッセージの本体タイプを設定します。ここにはメッセージの宛先、メッセージ受信の際の通信モード、およびメッセージ・フォーマッターなどの項目の定義が含まれます。
- ePICAdapterDaemonExtensions-- ePICApplication エレメントのオプションの子。アダプター・デーモンを示します。ここにはアプリケーション ID およびアダプター・ワーカーの数などの、アダプター・デーモンに関連した情報が含まれます。
- ePICTraceExtensions--ePICApplication エレメントのオプションの子。トレース・クライアント・アプリケーションまたはトレース・サーバー・エレメントを示します。これによってトレースに関連した情報を定義します。

67ページの図8 は、構成ファイルの上位構造を示したものです。これは実際に動作する構成ファイルの例ではありません。あくまで上位エレメント間の関連と依存関係を説明するための例です。完成例は、115ページの『付録D. サンプル構成ファイル』を参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<Epic o="ePIC">
  <ePICApplications o="ePICApplications">
    <!-- The following <ePICApplication> tag configures the kernel to work with
    an application named APP1. -->
    <ePICApplication epicappid="APP1">
      <!-- Tags here specify logging and trace information for the APP1
      application. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Tags here specify the queue manager and its attributes. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Tags here specify the details of transporting and processing messages
             from APP1. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- The following <ePICApplication> tag starts an adapter daemon for the
    APP1 application. -->
    <ePICApplication epicappid="APP1Daemon">
      <!-- Specifications for the APP1Daemon adapter daemon, which works with
      the APP1 application. -->
      <ePICAdapterDaemonExtensions cn="epicappextensions">
        <epicdepappid>APP1</epicdepappid>
        <epicminworkers>1</epicminworkers>
      </ePICAdapterDaemonExtensions>
    </ePICApplication>
    <!-- The following <ePICApplication> tag configures the kernel to work with
    an application named APP2. -->
    <ePICApplication epicappid="APP2">
      <!-- Tags here specify logging and trace information for the APP2
      application. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Tags here specify the queue manager and its attributes. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Tags here specify the details of transporting and processing messages
            from APP2. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- The following <ePICApplication> tag configures a trace client named
    TraceClient. -->
    <ePICApplication epicappid="TraceClient">
      <ePICTraceExtensions cn="epicappextensions">
        <!-- Tags here specify attributes of the trace client. -->
      </ePICTraceExtensions>
    </ePICApplication>
  </ePICApplications>
</Epic>
```

図 8. 構成ファイルの上位構造

構成ファイルで使用するすべての上位エレメントを網羅したリストと説明を以下に示します。エレメントがデフォルト値を持っていて、構成のエレメントが値を必要とする場合、その値を特に指定していないときは、カーネルはエレメントのデフォルト値を使用します。

構成ファイルで使用する XML エレメント

Epic 構成ファイルの最上位エレメント。

子エレメント:

- context
- ePICApplications (必須)

属性: o="ePIC" (必須)

context

Java Message Service (JMS) オブジェクトを使用する場合に、Java Naming and Directory Interface (JNDI) ファイル・システム・コンテキスト (FSContext) のルートを指定します。デフォルトは現行ディレクトリーです。JMS を使用する場合は必須です。MQSeries Adapter Kernel での JMS オブジェクトの使用についての情報は、102ページの『JMS オブジェクト記憶の使用』を参照してください。

子エレメント: なし

属性: なし

ePICApplications

カーネルによってサービスされるアプリケーションについての情報を含みます。

子エレメント: ePICApplication (必須)

属性: o="ePICApplications" (必須)

ePICApplication

カーネルによってサービスされるアプリケーションについての情報を指定します。

子エレメント:

- epiclogging
- epictrace
- epictracelevel
- epictraceclientid
- epiclogoninfoclassname
- AdapterRouting
- ePICTraceExtensions
- ePICAdapterDaemonExtensions

属性: epicappid="*application_ID*"。ここで *application_ID* は有効なアプリケーション ID です (必須)。

epiclogging

監査ログを実行するかどうかを決定します。監査ログを実行するには、WebSphere Business Integrator 製品が必要です。デフォルトは `false` です。

子エレメント: なし

属性: なし

epictrace

トレースを使用するかどうかを決定します。デフォルトは `false` です。

子エレメント: なし

属性: なし

epictracelevel

`com.ibm.logging.IRecordType` クラスによって指定される定数を使用して、トレースのレベルを設定します。デフォルトは `0` (設定なし) です。トレースについての詳細、および有効なトレース・レベルの全リストは、「*問題判別ガイド*」を参照してください。

子エレメント: なし

属性: なし

epictraceclientid

トレース・クライアント・アプリケーションの名前を指定します。デフォルトは `TraceClient` です。

子エレメント: なし

属性: なし

epiclogoninfoclassname

EAB アダプターを使用する場合に、アプリケーションとの接続に使用するログオン・クラスの名前を指定します。デフォルトは `com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault` です。

子エレメント: なし

属性: なし

AdapterRouting

メッセージ・タイプおよびメッセージのルーティングについての情報を含みます。

子エレメント:

- `epicmqppqueuemgr`
- `epicuseremotequeuemanagertosend`
- `epicmqppqueuemgrhostname`
- `epicmqppqueuemgrportnumber`

- epicmqppqueuemgrchannelname
- epicjmsconnectionfactoryname
- ePICBodyCategory (required)

属性: cn="epicadapterrouting" (必須)

epicmqppqueuemgr

トランスポート機構として MQSeries を使用する場合に、使用するキュー・マネージャーの名前を指定します。指定しない場合、または DEFAULT を指定した場合は、デフォルトのキュー・マネージャーが使用されます。

子エレメント: なし

属性: なし

epicuseremotequeuemanagertosend

トランスポート機構として MQSeries を使用している場合に、メッセージの送信にリモート・キュー・マネージャーを使用するかどうかを決定します。デフォルトは false です。

子エレメント: なし

属性: なし

epicmqppqueuemgrhostname

トランスポート機構として MQSeries を使用する場合に、キュー・マネージャーが常駐するマシンの TCP/IP ホスト名を指定します。MQSeries Client を使用する場合のみ必須です。

子エレメント: なし

属性: なし

epicmqppqueuemgrportnumber

トランスポート機構として MQSeries を使用する場合に、キュー・マネージャーのサーバー・プロセスのポート番号を指定します。デフォルトは 1414 (MQSeries のデフォルト) です。MQSeries Client を使用する場合のみ必須です。

子エレメント: なし

属性: なし

epicmqppqueuemgrchannelname

トランスポート機構として MQSeries を使用する場合に、キュー・マネージャー・サーバーのチャネル名を指定します。 MQSeries Client を使用する場合のみ必須です。

子エレメント: なし

属性: なし

epicjmsconnectionfactoryname

トランスポート機構として JMS を使用する場合に、JMS Connection のファクトリー名を指定します。値は *attribute=object* で指定します。ここで *attribute* は LDAP 属性、*object* は JMS Connection オブジェクト名です。オブジェクトは AdapterRouting エレメントのもとで保管されることになります。たとえば、JMS Connection オブジェクトの名前が QCFTEST1 で、LDAP 属性が cn の場合、このエレメントによって指定される値は cn=QCFTEST1 になります。

子エレメント: なし

属性: なし

ePICBodyCategory

送信されるメッセージの本体カテゴリーを指定します。

子エレメント: ePICBodyType (必須)

属性: epicbodycategory=*body_category*。 *body_category* で、送信されるメッセージの本体カテゴリーを指定します (必須)。

ePICBodyType

送信されるメッセージの本体タイプを指定します。

子エレメント:

- epiccommandclassname
- epiccommandtype
- epiccommandejbmapper
- epiccommandejbmethod
- epiccommandejbmethodparmtype
- epiccommandejburl
- epiccommandejbinitialcontext
- epicdestids
- epicreceivemode
- epicmessageformatter
- epicreceivetimeout
- epicreceivemqppqueue
- epicerrormqppqueue
- epicreplymqppqueue
- epicjmsreceivequeuename
- epicjmserrorqueuename
- epicjmsreplyqueuename
- epicreceivefiledir

- epiccommitfiledir
- epicerrorfiledir

属性: epicbodytype=*body_type*。 *body_type* で送信されるメッセージの本体タイプを指定します (必須)。

epiccommandclassname
メッセージの処理のために呼び出される EAB ターゲット・アダプターの名前または EJB コマンドを指定します。メッセージの受信にアダプター・デーモン、または WebSphere Application Server を使う場合は必須です。

子エレメント: なし

属性: なし

epiccommandtype
ターゲット・アダプターのタイプを指定します。次の値をとることができます。MQAKEAB および MQAKEJB。MQAKEAB は、標準 MQSeries Adapter Kernel EAB ターゲット・アダプターを指定します。MQAKEJB は、WebSphere Application Server のカーネルのターゲット・サイドでエンタープライズ bean を使用することを指定します。デフォルトは MQAKEAB です。ターゲット・アダプターがエンタープライズ bean の場合、MQAKEJB の値が必須です。

子エレメント: なし

属性: なし

epiccommandejbmapper
入力データをマップするのに使用する TDCMapper クラスの名前を指定します。デフォルトは TDCGenericMapper です。ターゲット・アダプターがエンタープライズ bean の場合に必須です。

子エレメント: なし

属性: なし

epiccommandejbmethod
エンタープライズ bean で呼び出すメソッドの名前を指定します。メソッドは TerminalDataContainer オブジェクトを入力として受け入れ、TerminalDataContainer オブジェクトを戻さなければなりません。デフォルトは execute です。ターゲット・アダプターがエンタープライズ bean の場合に必須です。

子エレメント: なし

属性: なし

epiccommandejbmethodparmtype
エンタープライズ bean で呼び出されたメソッドのパラメーターとして使用す

る、オブジェクトのクラス名を指定します。デフォルトは、TDCMapper に戻されたオブジェクトのクラス名です。ターゲット・アダプターがエンタープライズ bean の場合に必須です。

子エレメント: なし

属性: なし

epiccommandejburl

IIOP://*host_name* :*port* の形式で、配置されたエンタープライズ bean の URL を指定します。ここで、*host_name* は EJB サーバーのホスト名、*port* はネーム・サーバーが listen するポート (デフォルトでは 900) です。デフォルトは IIOP:/// です。ターゲット・アダプターがエンタープライズ bean の場合に必須です。

子エレメント: なし

属性: なし

epiccommandejbinitialcontext

エンタープライズ bean のホーム名を検索するのに使用する、初期コンテキスト・ファクトリーの名前を指定します。デフォルトは com.ibm.ejs.ns.jndi.CNInitialContextFactoryです。ターゲット・アダプターがエンタープライズ bean の場合に必須です。

子エレメント: なし

属性: なし

epicdestids

メッセージの宛先として使用する 1 つまたは複数のアプリケーションの ID を指定します。アプリケーションがメッセージを送信して、宛先論理 ID を NONE に設定している場合は必須です。

子エレメント: なし

属性: なし

epicreceivemode

使用する通信モードを指定します。有効な通信モードのリストと説明は、99ページの『付録A. 通信モード』を参照してください。アプリケーションがメッセージを受信する場合は必須です。

子エレメント: なし

属性: なし

epicmessageformatter

使用するメッセージ・フォーマッターを指定します。 epicreceivemode の値および使用するトランスポート・メソッドに依存します。メッセージ・フォーマッターおよびトランスポート・メソッドについての詳細は、 100ページの表10 および 101ページの表11 を参照してください。

子エレメント: なし

属性: なし

epicreceivetimeout

受信側がメッセージを待つ際のタイムアウトになるまでの時間を指定します (ミリ秒単位)。デフォルトは `0` です。値 `-1` は、タイムアウトなしを指定します (無制限に待ちます)。

子エレメント: なし

属性: なし

epicreceivemqppqueue

メッセージを受信するためのキューの名前を指定します。`epicreceivemode` エレメントが MQSeries 通信モードを指定する場合は必須です。MQSeries 通信モードのリストは、99ページの『付録A. 通信モード』を参照してください。

子エレメント: なし

属性: なし

epicerrormqppqueue

エラー・メッセージを書き込むキューの名前を指定します。エラー・メッセージのキューイングを使用して、`epicreceivemode` エレメントで MQSeries 通信モードが指定されている場合は必須です。MQSeries 通信モードのリストは、99ページの『付録A. 通信モード』を参照してください。

子エレメント: なし

属性: なし

epicreplymqppqueue

応答メッセージを受信するキューの名前を指定します。応答要求を使用して、`epicreceivemode` エレメントで MQSeries 通信モードを指定している場合は、必須です。MQSeries 通信モードのリストは、99ページの『付録A. 通信モード』を参照してください。

子エレメント: なし

属性: なし

epicjmsreceivequeuename

メッセージを受信するためのキューの名前を指定します。JMS 通信モードの場合は必須です。オブジェクトは `ePICBodyType` エレメントのもとで保管されることになります。値は *`attribute=object`* で指定します。ここで *attribute* は LDAP 属性、*object* は JMS キュー・オブジェクト名です。たとえば、JMS オブジェクトの名前が `TEST1AIQ` で、LDAP 属性が `cn` の場合、このエレメントによって指定される値は `cn=TEST1AIQ` になります。

子エレメント: なし

属性: なし

epicjmserrorqueuename
エラー・メッセージを書き込むキューの名前を指定します。エラー・メッセージ・キューイングを JMS 通信モードで使用している場合は必須です。オブジェクトは ePICBodyType エレメントのもとで保管されることになります。値は *attribute=object* で指定します。ここで *attribute* は LDAP 属性、*object* は JMS キュー・オブジェクト名です。たとえば、JMS オブジェクトの名前が TEST1AEQ で、LDAP 属性が cn の場合、このエレメントによって指定される値は cn=TEST1AEQ になります。

子エレメント: なし

属性: なし

epicjmsreplyqueuename
応答メッセージを受信するキューの名前を指定します。応答要求を JMS 通信モードで使用している場合は必須です。オブジェクトは ePICBodyType エレメントのもとで保管されることになります。値は *attribute=object* で指定します。ここで *attribute* は LDAP 属性、*object* は JMS キュー・オブジェクト名です。たとえば、JMS オブジェクトの名前が TEST1RPL で、LDAP 属性が cn の場合、このエレメントによって指定される値は cn=TEST1RPL になります。

子エレメント: なし

属性: なし

epicreceivefiledir
メッセージを受信するためのディレクトリーの名前を指定します。通信モードが FILE の場合は必須です。

子エレメント: なし

属性: なし

epiccommitfiledir
受信したメッセージをコミットされるまで保持するディレクトリーの名前を指定します。メッセージが受信されるときの通信モードが FILE の場合は必須です。

子エレメント: なし

属性: なし

epicerrorfiledir
エラー・メッセージを書き込むディレクトリーの名前を指定します。エラー・メッセージ・キューイングを FILE 通信モードで使用している場合は必須です。

子エレメント: なし

属性: なし

ePICAdapterDaemonExtensions
アダプター・デーモンの拡張についての情報を含みます。

子エレメント:

- epicdepappid
- epicminworkers

属性: cn="epicappextensions" (必須)

ePICTraceExtensions
トレースの拡張についての情報を含みます。このエレメントと子の詳細な説明は、「*問題判別ガイド*」を参照してください。

子エレメント:

- epicdepappid
- epictracesyncoperation
- epictracemessagefile
- epictracehandler
- ePICTraceHandler

属性: cn="epicappextensions" (必須)

epicdepappid
アダプター・デーモンがサービスを行うアプリケーションの ID を指定します。アダプター・デーモンが始動したときに使用されたアプリケーション ID がデフォルトになります。

子エレメント: なし

属性: なし

epicminworkers
アダプター・デーモンによって始動されるアダプター・ワーカーの数を指定します。デフォルトは 1 です。

子エレメント: なし

属性: なし

共通の構成

このセクションでは、さまざまな通信トランスポートを使用してメッセージの送受信を行う場合の値のような、いくつかの構成シナリオで共通の構成値をリストしています。メッセージが送信される場合、構成値はソース・サイドとターゲット・サイドの両方から取得されます。一方、メッセージが受信される場合、構成値はターゲット・サイドからしか取得されません。ソースとターゲットは、それぞれの論理 ID によって示されます。ここでの例は、ソースとターゲットが 2 台の異なるマシンにあることを前提にして

います。ターゲット・アプリケーション ID がまだ設定されていない場合、ソースの構成にある epicdestids エレメントの値から決定します。

注: これらの構成シナリオでは、適用できる、設定可能なエレメント構成値をリストしています。エレメントに適用するデフォルトについては、68ページの『構成ファイルで使用する XML エレメント』のエレメント・リストを参照してください。

MQSeries 共通構成: このセクションでは、MQSeries を通信トランスポートとして使用する場合の共通構成について説明します。 epicreceivemode エレメントで MQSeries 通信モードを指定します (たとえば、MQPP または MQRFH2)。次のシナリオをリストしています。

- 表2 で示しているのは、MQSeries サーバーから MQSeries サーバーへメッセージを送信する場合に設定する必要のある構成エレメントです。
- 78ページの表3 で示しているのは、リモート・キュー・マネージャーを使用している MQSeries サーバーから、MQSeries サーバーへメッセージを送信する場合に設定する必要のある構成エレメントです。
- 79ページの表4 で示しているのは、ホスト・サーバーを使用している MQSeries クライアントから、MQSeries サーバーへメッセージを送信する場合に設定する必要のある構成エレメントです。
- 80ページの表5 で示しているのは、MQSeries サーバーでメッセージを受信する場合に設定する必要のある構成エレメントです。
- 80ページの表6 で示しているのは、ホスト・サーバーを使用している MQSeries クライアントでメッセージを受信する場合に設定する必要のある構成エレメントです。

表 2. 共通の構成: MQSeries サーバーから別の MQSeries サーバーへメッセージを送信

ソース構成	ターゲット構成
	epicreceivemode エレメントで、MQSeries 通信モードを指定します。
epicmqpqueuemgr エレメントで、キュー・マネージャーの名前を指定します。このキュー・マネージャーはソース・アプリケーションのマシンに存在しなければなりません。	
	epicreceivemqpqueue エレメントで、受信キューの名前を指定します。このキューは、ターゲット・アプリケーションのマシンでの MQSeries リモート・キュー、または MQSeries クラスターの一部でなければなりません。

表 2. 共通の構成: MQSeries サーバーから別の MQSeries サーバーへメッセージを送信 (続き)

ソース構成	ターゲット構成
epicreplymqppqueue エレメントで、応答キューの名前を指定します。このキューは、送信側マシンの MQSeries ローカル・キュー、または MQSeries クラスターの一部でなければなりません。同期要求および応答の場合のみ使用されます。	
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。
epicreceivetimeout エレメントで、タイムアウトになる前に受信側が応答を待つ時間を指定します。	

表 3. 共通の構成: MQSeries サーバーからリモート・キュー・マネージャー経由で MQSeries サーバーへメッセージを送信

ソース構成	ターゲット構成
	epicreceivemode エレメントで、MQSeries 通信モードを指定します。
epicmqppqueuemgr エレメントで、キュー・マネージャーの名前を指定します。このキュー・マネージャーはソース・アプリケーションのマシンに存在しなければなりません。	epicmqppqueuemgr エレメントで、キュー・マネージャーの名前を指定します。このキュー・マネージャーはターゲット・アプリケーションのマシンに存在しなければなりません。名前を指定しなければなりません。デフォルト値は使用できません。
epicremotequeuemanagertosend エレメントで、メッセージを送信するのにリモート・キュー・マネージャーを使用することを指定します。	
	epicreceivemqppqueue エレメントで、受信キューの名前を指定します。このキューは、ターゲット・アプリケーションのマシンでの MQSeries ローカル・キュー、または MQSeries クラスターの一部でなければなりません。
epicreplymqppqueue エレメントで、応答キューの名前を指定します。このキューは、送信側マシンの MQSeries ローカル・キュー、または MQSeries クラスターの一部でなければなりません。同期要求および応答の場合のみ使用されます。	
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。

表 3. 共通の構成: MQSeries サーバーからリモート・キュー・マネージャー経由で MQSeries サーバーへメッセージを送信 (続き)

ソース構成	ターゲット構成
epicreceivetimeout エレメントで、タイムアウトになる前に受信側が応答を待つ時間を指定します。	

表 4. 共通の構成: ホスト・サーバーを使用している MQSeries クライアントから MQSeries サーバーへメッセージを送信

ソース構成	ターゲット構成
	epicreceivemode エレメントで、MQSeries 通信モードを指定します。
epicmqppqueuemgr エレメントで、キュー・マネージャーの名前を指定します。このキュー・マネージャーは送信側クライアントのホスト・マシンに存在しなければなりません。	
epicmqppqueuemgrhostname エレメントで、MQSeries サーバー・マシンのホスト名を指定します。	
epicmqppqueuemgrportnumber エレメントで、サーバー・マシンでのキュー・マネージャーのサーバー・プロセスのポート番号を指定します。	
epicmqppqueuemgrchannelnumber エレメントで、キュー・マネージャー・サーバーのチャネル番号を指定します。	
	epicreceivemqppqueue エレメントで、受信キューの名前を指定します。このキューは、ターゲット・アプリケーションのマシンでの MQSeries リモート・キュー、または MQSeries クラスターの一部でなければなりません。
epicreplymqppqueue エレメントで、応答キューの名前を指定します。このキューは、送信側クライアントのホスト・マシンの MQSeries ローカル・キュー、または MQSeries クラスターの一部でなければなりません。同期要求および応答の場合のみ使用されます。	
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。
epicreceivetimeout エレメントで、タイムアウトになる前に受信側が応答を待つ時間を指定します。	

表 5. 共通の構成: メッセージを受信する MQSeries サーバー

ソース構成	ターゲット構成
適用できません。	epicreceivemode エレメントで、MQSeries 通信モードを指定します。
	epicmqppqueuemgr エレメントで、キュー・マネージャーの名前を指定します。このキュー・マネージャーはターゲット・アプリケーションのマシンに存在しなければなりません。
	epicreceivemqppqueue エレメントで、受信キューの名前を指定します。このキューは、ターゲット・マシンでの MQSeries ローカル・キューでなければなりません。
	epicerrormqppqueue エレメントで、エラー・キューの名前を指定します。このキューは、ターゲット・マシンでの MQSeries ローカル・キュー、またはクラスターの一部でなければなりません。アダプター・ワーカーを使用する場合のみ必須です。
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。
	epicreceivetimeout エレメントで、タイムアウトになる前に受信側がメッセージを待つ時間を指定します。

表 6. 共通の構成: ホスト・サーバーを使用している MQSeries クライアントがメッセージを受信

ソース構成	ターゲット構成
適用できません。	epicreceivemode エレメントで、MQSeries 通信モードを指定します。
	epicmqppqueuemgr エレメントで、キュー・マネージャーの名前を指定します。このキュー・マネージャーは受信側クライアントのホスト・マシンに存在しなければなりません。
	epicmqppqueuemgrhostname エレメントで、MQSeries サーバー・マシンのホスト名を指定します。
	epicmqppqueuemgrportnumber エレメントで、サーバー・マシンでのキュー・プロセスのサーバー・プロセスのポート番号を指定します。
	epicmqppqueuemgrchannelnumber エレメントで、キュー・マネージャー・サーバーのチャネル番号を指定します。

表 6. 共通の構成: ホスト・サーバーを使用している MQSeries クライアントがメッセージを受信 (続き)

ソース構成	ターゲット構成
	epicreceivemqppqueue エレメントで、受信キューの名前を指定します。このキューは、受信側クライアント・ホスト・マシンでの MQSeries ローカル・キューでなければなりません。
	epicerrormqppqueue エレメントで、エラー・キューの名前を指定します。このキューは、受信側クライアントのホスト・マシンの MQSeries ローカル・キュー、またはクラスターの一部でなければなりません。アダプター・ワーカーを使用する場合のみ必須です。
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。
	epicreceivetimeout エレメントで、タイムアウトになる前に受信側がメッセージを待つ時間を指定します。

JMS 共通構成: このセクションでは、JMS を通信トランスポートとして使用する場合の共通構成について説明します。epicreceivemode エレメントで、JMS を指定します。

MQSeries JMS インプリメンテーションを使用する場合、適切な MQSeries オブジェクトが存在しなければなりません。たとえば、JMS キュー接続ファクトリーは、MQSeries サーバーのキュー・マネージャーに関連しなければならず、そして JMS キューは MQSeries キューに関連しなければなりません。MQSeries オブジェクトは構成にリストする必要はありませんが、サポートする MQSeries オブジェクトは存在しなければなりません。

次のシナリオをリストしています。

- 表7 で示しているのは、JMS 経由でメッセージを送信する場合に設定する必要のある構成エレメントです。
- 82ページの表8 で示しているのは、JMS 経由でメッセージを受信する場合に設定する必要のある構成エレメントです。

表 7. 共通の構成: JMS 経由でのメッセージの送信

ソース構成	ターゲット構成
	epicreceivemode エレメントで、JMS 通信モードを指定します。

表 7. 共通の構成: JMS 経由でのメッセージの送信 (続き)

ソース構成	ターゲット構成
epicjmsconnectionfactoryname エレメントで、JMS キュー接続ファクトリーの名前を指定します。参照されるオブジェクトが、構成の中に存在しなければなりません。	
	epicjmsreceivequeuename エレメントで、JMS 受信キューの名前を指定します。参照されるオブジェクトが、構成の中に存在しなければなりません。
epicjmsreplyqueuename エレメントで、JMS 応答キューの名前を指定します。参照されるオブジェクトが、構成の中に存在しなければなりません。同期要求および応答の場合のみ使用されます。	
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。
epicreceivetimeout エレメントで、タイムアウトになる前に受信側が応答を待つ時間を指定します。	

表 8. 共通の構成: JMS 経由でメッセージを受信

ソース構成	ターゲット構成
適用できません。	epicreceivemode エレメントで、JMS 通信モードを指定します。
	epicjmsconnectionfactoryname エレメントで、JMS キュー接続ファクトリーの名前を指定します。参照されるオブジェクトが、構成の中に存在しなければなりません。
	epicjmsreceivequeuename 要素で、JMS 受信キューの名前を指定します。参照されるオブジェクトが、構成の中に存在しなければなりません。
	epicjmserrorqueuename エレメントで、JMS エラー・キューの名前を指定します。参照されるオブジェクトが、構成の中に存在しなければなりません。
	epicmessageformatter エレメントで、使用するフォーマッターの名前を指定します。
	epicreceivetimeout エレメントで、タイムアウトになる前に受信側がメッセージを待つ時間を指定します。

アダプター共通構成: このセクションでは、アダプターがターゲット・サイドで呼び出される場合の共通構成について説明します。ターゲットが Enterprise Access Builder (EAB) ターゲット・アダプター (MQAKEAB の epiccommandtype 値で指定します)、または、EJB サービス・セッション bean ターゲット・アダプター (MQAKEJB の epiccommandtype 値で指定します) かによって、異なる構成値が使用されます。

注: EJB セッション bean ターゲット・アダプターは、WebSphere Application Server の中でのみサポートされます。

MQAKEAB の epiccommandtype 値の場合、次の追加エレメントに値を指定します。

- epiclogoninfoclassname
- epiccommandclassname

MQAKEJB の epiccommandtype 値の場合、次の追加エレメントに値を指定します。

- epiccommandclassname
- epiccommandejbmethod
- epiccommandejbmethodparmtype
- epiccommandejburl
- epiccommandejbinitialcontext
- epiccommandejbmapper

構成へのアダプター情報の追加

カーネルの構成に新たにアダプターを追加するときは、最小限度のいくつかの指定を構成ファイルに追加する必要があります。最小の構成ファイルの例として、aqmconfig.minimum.xml ファイルを参照してください。このファイルは 115ページの『付録D. サンプル構成ファイル』に示されてます。また、インストールした MQSeries Adapter Kernel の samples ディレクトリーにもあります。

新たにアダプターを追加するときに構成に追加しなければならない最小限度の情報として、次のことを指定します。

- **ソース・アダプター** (メッセージの送信):
  - ソース・アダプターが実行されるアプリケーションの ID。
  - デフォルトのキュー・マネージャー。MQSeries がトランスポート機構として使用され、ソース・アダプターと同じマシンにインストールされて実行されている場合は、キュー・マネージャーを特に構成する必要はありません。
  - メッセージの宛先論理 ID。メッセージがすべて同じ宛先に行く場合は、本体カテゴリーに DEFAULT、本体タイプに DEFAULT を使用します。
  - ソース・アダプターがメッセージを送信する宛先論理 ID ごとの受信キュー。
- **ターゲット・アダプター** (メッセージの受信):
  - ターゲット・アダプターが実行されるアプリケーションの ID。

- デフォルトのキュー・マネージャー。MQSeries がトランスポート機構として使用され、ソース・アダプターと同じマシンにインストールされて実行されている場合は、キュー・マネージャーを特に構成する必要はありません。
- MQSeries の受信モード。これは、通常はすべてのメッセージで同じです。その場合は、本体カテゴリーに DEFAULT、本体タイプに DEFAULT を使用します。
- 受信キュー。受信キューがすべてのメッセージで同じ場合は、本体カテゴリーに DEFAULT、本体タイプに DEFAULT を使用します。
- エラー・キュー。ターゲット・アダプターがメッセージを処理中にエラーが発生したときのためです。これは、通常はすべてのメッセージで同じです。その場合は、本体カテゴリーに DEFAULT、本体タイプに DEFAULT を使用します。
- メッセージを受信したときに呼び出すターゲット・アダプターのクラス名。これは本体カテゴリーと本体タイプに固有のものです。
- 受信タイムアウト値。CPU 使用量の増加を抑えるために適切な値を設定することをお勧めします。これは、通常はすべてのメッセージで同じです。その場合は、本体カテゴリーに DEFAULT、本体タイプに DEFAULT を使用します。

追加のターゲット・アダプターの場合、同じ受信キューを使用するときは、同じ情報でかまいません。この場合、別に指定する必要がある情報は、指定された本体カテゴリーと本体タイプに応じて呼び出すターゲット・アダプターのクラス名のみです。

- **トレースの指定**:
  - トレースのオン / オフ。
  - トレース・レベル。
  - トレースの宛先など、ソース・アダプターおよびターゲット・アダプターに応じたトレース指定の追加。デフォルトでは、トレースは、カーネルが始動した所のコマンド・プロンプト・ウィンドウまたはターミナルに表示されます。

構成ファイルの編集

構成ファイルの編集には、テキスト・エディターまたは専用の XML エディターを使用します。XML エディターのユーザーのために、インストールしたカーネルの `samples` ディレクトリーに、`aqmconfig.dtd` という DTD ファイルが用意されています。Xeena と呼ばれる XML エディターを、IBM alphaWorks Web サイト www.alphaworks.ibm.comからダウンロードすることができます。構成ファイルを編集する際には、次のことをお勧めします。

- 構成ファイルの編集を始める前に、希望する構成についての適切な情報をすべて収集します。これには、構成に含まれるアプリケーションとキューの名前、やり取りされるメッセージのタイプ、使用する通信モード (複数の場合あり)、およびトレース・プログラムと他の拡張機能についての情報があります。
- `samples` ディレクトリーのサンプル・ファイル `aqmconfig.xml` を任意の場所にコピーします。構成ファイルのコピーはリネームしないでください。編集はコピーに対して行います。

- コメントを使って構成ファイルのセクションを示し、また、構成に使用する特定の値 (たとえばアプリケーション ID、メッセージ・キュー名、タイムアウト値) の記述を記録します。XML では、コメントは <!-- の文字で始まり、--> の文字で終わります。コメントは次の例のように、複数行にわたって書くことができます。

  ```
  <!--
       Comment text
  -->
  ```

  なお、XML では、コメントの中に別のコメントを含むことはできないので注意が必要です。
- アプリケーション ID に従って構成ファイルを編成します。各アプリケーション ID へのエントリーは一緒にしておきます。
- 専用の XML エディターを使用しない場合は、ファイルが保存されるときに行の終了を保持し、行を分割しないテキスト・エディターを使用します。この種のテキスト・エディターの例として Windows システムでは「メモ帳」、 UNIX では vi エディターまたは Emacs があります。
- XML では大文字小文字の区別があるので、すべてのタグ (エレメント) 名と属性に正しい大文字小文字を使用するように十分な注意が必要です。タグ付けに大文字小文字を誤って使うと、その構成ファイルは無効になります。専用の XML エディターを使用すると、大文字小文字の誤りを防ぐことに役立ちます。
- 本体カテゴリーと本体タイプにデフォルト値を使いたいがデフォルト値としての値がない場合は、構成ファイルで各 DEFAULT の値を構成する必要があります。そうでない場合は、カーネルはデフォルト値は使用しません。
- 構成ファイルを、実動に使用する前に検査を行います。『構成ファイルの妥当性検査』を参照してください。
- 構成ファイルの変更は、カーネル・プロセスの次の始動時に実際に反映されます。構成ファイルを変更するときにプロセスが実行中の場合は、プロセスを停止してから再始動します。これによって構成ファイルの変更が実際に反映されます。現在実動中の構成ファイルを編集する場合は、十分注意してください。
- 構成ファイルは、編集するごとにバックアップをとります。

構成ファイルの妥当性検査

編集が終了した構成ファイルは、実動に使用する前に妥当性検査を行うことをお勧めします。構成ファイルの妥当性検査は、次の一般的なステップで行います。

1. 妥当性検査を行いテストをセットアップするための、構成ファイルの妥当性検査のディレクトリーを作成します。
2. 妥当性検査 XML メッセージを作成します。
3. 妥当性テストに合わせてメッセージ・キューをセットアップします。
4. 構成ファイル妥当性テストをセットアップしてから実行します。そのテストでは、メッセージの送信と受信を行います。

5. テストの結果を調べて、構成ファイルが正しいかどうかを判別します。

妥当性検査 XML メッセージの作成に利用できるユーティリティー、および構成ファイル妥当性テストは、カーネルの一部に用意されています。

構成ファイル妥当性テストでは、sendMsg メソッドを呼び出して、妥当性検査 XML メッセージを、カーネルのソース・サイドのネイティブ・アダプターからカーネルのターゲット・サイドのアダプター・デーモンに送信します。ソース・アダプターとターゲット・アダプターは必要ありません。ただし、使用できるターゲット・アダプターがあれば、そのターゲット・アプリケーションへのメッセージの送信テストも行うことができます。

次の手順で行います。

注: この手順を行う際に使用すると便利な、いくつかのスクリプトが用意されています。必要であれば、そのスクリプトをコピーしてからそのコピーを編集して、実際に使用するバージョンを作成してください。OS/400 を使用している場合は、スクリプトの UNIX バージョンが **qsh** セッションで動作することができます。**qsh** セッションを開始するには、Control Language (制御言語)(CL) プロンプトで、QSH の開始 (**STRQSH**) コマンドを入力します。

ステップ 1. コマンド・プロンプト・ウィンドウをオープンします。

ステップ 2. 構成ファイル妥当性検査ディレクトリーを作成します。そこに構成ファイルとセットアップ・ファイルをコピーします。

ステップ 3. ディレクトリーを妥当性検査ディレクトリーにします。

ステップ 4. 次のコマンドを入力して、妥当性検査 XML メッセージを作成します。

- aqmcrtmsg.bat (Windows システム)
- aqmcrtmsg.sh (UNIX および OS/400)

ステップ 5. オプションのリストが表示されます。オプションを選択して Enter を押します。それぞれの値を入力します。値を入力する順序は重要ではありません。たとえばオプションとして、set sourcelogicalid、set msgtype、および set bodycategory があります。オプション 20、21、22、および 23 には値を入力する必要があります。オプション 24 または 241 を使って、メッセージ本体データを規定することができます。他の値は必要ありません。

ステップ 6. オプション 1 を入力して、妥当性検査 XML ファイルを作成します。EpicMessage*nn*.xml という名前の妥当性検査 XML ファイルが現行ディレクトリーに作成されます。ここで *nn* は、XML ファイルの数です。

ステップ 7. オプション 0 を入力して妥当性検査ユーティリティーを終了します。

ステップ 8. 妥当性検査に合わせて適切なメッセージ・キューをセットアップします。

ステップ 9. AQMSETUPFILE 環境変数を設定して、妥当性検査ディレクトリーにあるセットアップ・ファイルを一時的に指すようにします。

- Windows システムでは、コマンド・プロンプトで次を入力します。

  ```
  set AQMSETUPFILE=E:¥run_time_files¥aqmsetup
  ```

  ここで、E:¥ は正しいドライブを表し、*run_time_files* は妥当性検査ディレクトリーです。
- UNIX および OS/400 では、次のコマンドを入力します。このコマンドの例では Korn シェルの使用を想定しています。別のシェルを使用する場合は、シェルに応じてコマンドを変更してください。

  ```
  export AQMSETUPFILE=root_directory/run_time_files/aqmsetup
  ```

  ここで *root_directory* はカーネルをインストールしたディレクトリー、そして *run_time_files* は妥当性検査ディレクトリーです。OS/400 では、aqmsetup ファイルは常にユーザーの IFS ホーム・ディレクトリー (/home/*user_name*) に置かなければなりません。

必要であれば、妥当性検査ディレクトリーにあるセットアップ・ファイルを編集して、妥当性検査を行う構成ファイルを指すようにします。

ステップ 10. 次のどのテストを行うかを選択します。

- カーネルのソース・サイドのみ
- メッセージの、ターゲット・アプリケーションへのルート指定がいろいろ行われるかどうか。このテストでは、ターゲット・アダプターをあらかじめ使えるようにしておく必要があります。
- トレース

最初にソース・サイドのテストを行い、次にターゲット・サイドのテストを行います。ソース・サイドのみのテストを行うために、アダプター・デーモンをオフにします。ターゲット・サイドのテストを同様に行うために、アダプター・デーモンをオンにします。ターゲット・アダプターがあらかじめ使えるようになっていない場合でも、アダプター・デーモンが正しいターゲット・アダプターのコマンドの呼び出しを試みているときに、アダプター・デーモンがそのポイントまでメッセージを処理しているかどうかをテストすることができます。特にターゲット・アダプターをあらかじめ使えるようにしていない場合は、トレースを使用可能にすることをお勧めします。

ステップ 11. 妥当性テストを実行します。いずれかのディレクトリーから、次のコマンドを入力します。

- Windows システムの場合:

  ```
  aqmsndmsg.bat -a source_logical_identifier -f XML_message_file
  ```
- UNIX および OS/400 の場合:

  ```
  aqmsndmsg.sh -a source_logical_identifier -f XML_message_file
  ```

  ここで:

source_logical_identifier
は、ソース論理 ID を示します。この値は、86ページの5 ステップのオプション 20 で入力したソース論理 ID の値と一致しなければなりません。

XML_message_file
は、XML メッセージ・ファイルを示します。

注: 次のコマンドを入力すると、このテストのすべてのオプションのリストが表示されます。

Windows システムの場合:

```
aqmsndmsg.bat -?
```

UNIX および OS/400 の場合:

```
aqmsndmsg.sh -?
```

なお、-? は、Korn シェルでのみ有効です。別の UNIX シェル (たとえば Bourne シェルまたは C シェル) を使用する場合は、円記号 (¥) を使用して (つまり -¥?)、疑問符 (?) をエスケープしてください。

ステップ 12. 結果を調べます。妥当性検査メッセージには正しい本体カテゴリー、本体タイプ、およびデータが含まれています。

- カーネルのソース・サイドのみをテストしている場合 (つまりアダプター・デーモンを始動していない場合) は、メッセージがルート指定されたキューを調べます。
  - 妥当性検査メッセージがそのキューにあれば、構成ファイルにあるエントリーは有効であることが確認されています。
  - 妥当性検査メッセージがそのキューにない場合は、例外ファイルをチェックします。トレースが使用可能になっていれば、トレース・メッセージをチェックします。
- カーネルのターゲット・サイドをテストしていて、ターゲット・アダプターが使えるようになっていれば、ターゲット・アプリケーションをチェックします。
  - 妥当性検査メッセージがターゲット・アプリケーションで受信されれば、構成ファイルにあるエントリーは有効であることが確認されています。
  - 妥当性検査メッセージがターゲット・アプリケーションで受信されなければ、例外ファイルをチェックします。トレースが使用可能になっていれば、トレース・メッセージをチェックします。
- カーネルのターゲット・サイドをテストしていて、使えるようになっているターゲット・アダプターがなければ、妥当性検査メッセージのエラ

ー・キューと例外メッセージの例外ファイルをチェックします。トレースが使用可能になっていれば、トレース・メッセージをチェックします。

– 妥当性検査メッセージがそのエラー・キューにあり、例外メッセージがあれば、構成ファイルにあるエントリーは有効であることが確認されています。
– 妥当性検査メッセージがそのエラー・キューにない場合は、例外ファイルをチェックします。トレースが使用可能になっていれば、トレース・メッセージをチェックします。

ステップ 13. 必要であれば、構成ファイルを変更して再び妥当性検査を行います。

MQSeries および MQSeries Integrator の構成

MQSeries および MQSeries Integrator などのオプション・ソフトウェアを次のように構成して、カーネルをサポートします。

MQSeries の場合:

- インストールの検査のためにいくつかのキューが使用されます。これらのキューをテストまたは実稼働環境で使用している場合は、インストールの検査のためにそれらをクリアします。インストールの検査に使用するキューについては、 48ページの『検査手順』 を参照してください。
- メッセージのトランスポートをサポートするために、設計したルーティング体系に従ってキューをセットアップします。
- キューを作成するとき、環境変数の MAX_QUEUE_DEPTH は可能な限りのキューの最大長に設定します。

MQSeries Integrator では、入力および出力キューを、 rules (バージョン 1.1) または message flows (バージョン 2) でセットアップします。いずれも構成ファイルで構成するキューに相当しています。

パフォーマンスの推奨事項

以下のパフォーマンスの推奨事項は、特に MQSeries Adapter Kernel についてのものです。

- XML DTD が構文解析されるときは、構文解析するプロセスと同じディレクトリーに DTD ファイルを置くようにします。これによって、そのプロセスが DTD ファイルを検出するために要する時間を低減できます。
- 大規模なメッセージが送受信されるときは、メッセージ・タイプ RFH2 を使うと、メッセージ・タイプ XML を使う場合より良いパフォーマンスが得られます。

パフォーマンスの向上のための一般的な推奨事項は、MQSeries の資料を参照してください。

カーネルの始動

カーネルを始動するには、次のものを始動します。

- 各ターゲット・アプリケーションのアダプター・デーモン
- トレース・サーバー (オプション)

ソース・アダプターがソース・アプリケーションのプロセスで実行されている場合は、ソース・アダプターはソース・アプリケーションによって自動的に始動されます。この場合、ソース・アダプターの始動のために特別なステップは必要ありません。ソース・アダプターを含んでいるデーモンまたはサーバーを始動する必要があります。ソース・アダプターは、直接始動できません。

次のステップを実行して、各アダプター・デーモンとトレース・サーバーを始動します。

注: この手順を行う際に使用すると便利な、いくつかのスクリプトが用意されています。必要であれば、そのスクリプトをコピーしてからそのコピーを編集して、実際に使用するバージョンを作成してください。OS/400 を使用している場合は、スクリプトの UNIX バージョンが **qsh** セッションで動作することができます。**qsh** セッションを開始するには、Control Language (制御言語) (CL) プロンプトで、QSH の開始 (**STRQSH**) コマンドを入力します。

ステップ 1. MQSeries または他のメッセージング・ソフトウェア、および MQSeries Integrator などのオプション・ソフトウェアを始動します。

ステップ 2. 実際の現場で必要な関連ソフトウェアを始動します。たとえば、キューのトレース・メッセージを読み取るためのアプリケーション (カーネルの範囲外) があります。

ステップ 3. コマンド・プロンプトをオープンします。アダプター・デーモンごとに、次のコマンドを入力します。

- Windows システムの場合:

  ```
  aqmstrad.bat -a application_identifier [-bc body_category
  -bt body_type] [-noretry]
  ```

- UNIX および OS/400 の場合:

  ```
  aqmstrad.sh -a application_identifier [-bc body_category
  -bt body_type] [-noretry]
  ```

  where

  -a *application_identifier*
  は、アダプター・デーモンがサービスを行う宛先論理 ID を示します。

  -bc *body_category*
  は、アダプター・デーモン・ワーカーがメッセージを受信する際

に、通信モードおよび関連情報を判別するために使用する本体カテゴリーを指定します。値がない場合は、アダプター・デーモンは値 DEFAULT を使用します。

-bt *body_type*
は、アダプター・デーモン・ワーカーがメッセージ受信の際に、通信モードおよび関連情報を判別するために使用する本体タイプを指定します。値がない場合は、アダプター・デーモンは値 DEFAULT を使用します。

-noretry
は、メッセージがすでにないときにワーカーが自動的に停止するように指定します。 -noretry が指定されない場合は、ワーカーはキューに対してメッセージのポーリングを繰り返します。また、アダプター・デーモンの停止を手操作で行う必要があります。

注: Java 始動パラメーターを変更する必要がある場合は、aqmstrad.bat ファイル (Windows システム)、または aqmstrad.sh ファイル (UNIX および OS/400) を編集します。詳細は、ファイルの中のコメントを参照してください。

ステップ 4. トレース・サーバーごとに、次のコマンドを入力します。

- Windows システムの場合:

  aqmstrtd.bat *-how* -a *source_application_identifier*

- UNIX および OS/400 の場合:

  aqmstrtd.sh *-how* -a *source_application_identifier*

  ここで:

  -how
  は、トレース・メッセージがどのように受信されるかを示します。次の値をとることができます。

  – socket

  – ena (ネイティブ・アダプター)

  -a *source_application_identifier*
  ソース・アプリケーション ID。値がない場合は、構成ファイルの中のデフォルト値の TraceServer が使用されます。

トレース・サーバーについての詳細は、「*問題判別ガイド*」を参照してください。

ステップ 5. アダプター・デーモンまたはトレース・サーバーが始動した後は、プロセス・ウィンドウはアダプター・デーモンを停止するまでオープンのままになります。プロセス・ウィンドウに例外が表示されます。93ページの『例外メッセージ』を参照してください。

カーネルの停止

カーネルを停止するには、各アダプター・デーモン、トレース・サーバーをそれぞれ停止します。それらを停止するには、次のようないくつかの方法があります。

- アダプター・デーモンを始動するときに、パラメーター `-noretry` を設定します。90ページの『カーネルの始動』を参照してください。
- アダプター・デーモンまたはトレース・サーバーを始動したコマンド・プロンプト (Windows システム)、またはターミナル (UNIX) で、**Ctrl-C** を押します。各アダプター・デーモンまたは各トレース・サーバーごとに、このステップを実行します。
- Windows システムではタスク・マネージャーを使ってプロセスを終了させることができます。
- UNIX では **ps** コマンドを使ってプロセスの数を判別してから、**kill** コマンドを使ってプロセスを終了させることができます。

カーネルの保守

カーネルの保守計画をセットアップします。定期的に次の項目のバックアップをとることをお勧めします。

- 次のファイルで指定されている構成:
  - `aqmconfig.xml`
  - `aqmsetup`
- 作成したアダプター、およびそれらの関連ファイル

カーネルによって使用されたトレースや他のファイルの内容のバックアップをとったり、それらを定期的に削除することで、カーネルそのものの処理をサポートする必要はありません。これらのファイルは、必要があればバックアップをとってください。トレース・メッセージが複数ファイルではなく、単一ファイルにルート指定されていると、単一のトレース・ファイルは非常に大きくなります。トレース・レベルがハイレベルで詳細を取り込むように設定されている (たとえばすべてのトレース・メッセージまたは情報メッセージ) 場合は、トレース・ファイルを定期的に削除するようにします。

問題の診断

問題を診断するには、例外メッセージ、トレース・メッセージ、および MQSerie エラー・キューを利用することができます。MQSeries Adapter Kernel は例外メッセージと、トレースが使用可能であればトレース・メッセージを生成します。MQSeries Adapter Kernel カーネル環境での問題の診断方法についての情報は、「*問題判別ガイド*」を参照してください。

例外メッセージとトレース・メッセージを理解するには、カーネルの動作を理解することが必要です。カーネルはエラー・キューを使用して、いくつかのエラーを扱います。9ページの『カーネルの動作』を参照してください。

例外メッセージとトレース・メッセージの原因になったメッセージは、メッセージの固有 ID とトランザクションの固有 ID の組み合わせで識別することができます。

エラー・キューとカーネルの両方の同じメッセージを、決定的に識別できる ID はありません。ただし、エラー・キューのメッセージと、それに相当する例外メッセージ、トレース・メッセージ、またはその両方との相互関係を、手操作で明確にすることができます。そのためには、次の 1 つまたは複数を比較します。

- およそのタイム・スタンプ
- ソース論理 ID 用のキュー
- 宛先論理 ID 用のキュー
- 本体カテゴリー
- 本体タイプ
- メッセージ固有 ID
- トランザクション固有 ID

それらが一致すれば、その時点で、エラー・キューのメッセージと、それに相当する例外メッセージまたはトレース・メッセージとの相互関係がおそらく明確になっているでしょう。

バージョン番号

bin ディレクトリーの aqmversion.bat (Windows システム)、または aqmversion.sh (UNIX および OS/400) を実行すると、カーネルのバージョン番号が表示されます。

例外メッセージ

カーネルは次のタイプの例外メッセージを生成します。

- カーネルのソース・サイドのネイティブ・アダプターがソース・アダプターに例外をスローします。ソース・アダプターがそれらの例外をどのように扱うかについては、MQSeries Adapter Builder の資料を参照してください。
- カーネルのターゲット・サイドのネイティブ・アダプターが、ネイティブ・アダプターを管理するワーカーに例外をスローします。
- ワーカーは例外を、ワーカーと同じディレクトリーにある EpicSystemExceptionFile*nnnnnnnn*.log ファイルに書き込みます。
- アダプター・デーモンは例外メッセージを、アダプター・デーモンと同じディレクトリーにある EpicSystemExceptionFile*nnnnnnnn*.log という例外ファイルに書き込みます。アダプター・デーモンとそのワーカーは同じディレクトリーにあるので、両方とも同じ例外ファイルに書き込みます。また、アダプター・デーモンは例外メッセージをコンソール (つまり、アダプター・デーモンをウィンドウから始動した場合は、その始動したコマンド・プロンプトまたはターミナル) にも書き込みます。

カーネルのトレース例外メッセージは MQSeries 例外メッセージとは異なります。次はカーネルの例外メッセージの例です。

```
2000.10.26 19:38:20.929 com.ibm.epic.adapters.eak.nativeadapter.LMSMQ
Thread Name=main receiveRequest(ENAService)  ePIC TEST2
TYPE_ERROR_EXC AQM5004: Received exception <com.ibm.epic.adapters.eak.common.
AdapterException> Message information: <AQM0114: com.ibm.epic.adapters.eak.
nativeadapter.MQNMRFH2Formatter::convertMessage(MQMessage): Expecting a message
with an MQHRF2 format and received a message with format <MQSTR   >.>
for <unmarshall Message()> having invalid data <(null)>
```

例外メッセージの値は、メッセージの性質によって決まりますが、次の項目が含まれる場合もあります。

- タイム・スタンプ
- ソース論理 ID
- 宛先論理 ID
- 本体カテゴリー
- 本体タイプ
- メッセージ固有 ID
- トランザクション固有 ID
- 例外情報

インストールの検査時に起こる共通の問題および考えられる対応については、49ページの『検査の共通問題』を参照してください。

トレース・メッセージ

カーネルを、トレース・メッセージを生成するように構成することができます。トレースについての情報は、「*問題判別ガイド*」を参照してください。

ユーティリティー

MQSeries キューの作成

バッチ・ファイルまたはシェル・スクリプトを使って、MQSeries キューの作成を自動化することができます。`aqmcreateq.bat` (Windows システム)、または `aqmcreateq.sh` (UNIX および OS/400) を、パラメーターにアプリケーション名を使って実行します。これらのファイルは、アプリケーションごとに次のキューを作成します。

- 受信キュー (*application_name*AIQ)
- エラー・キュー (*application_name*AEQ)
- 応答キュー (*application_ name*RPL)

第5章 MQSeries Adapter Kernel API の使用

カーネルには、メッセージの送受信、XML の作成と構文解析、およびカーネル構成の管理などの機能として使用する API があります。これらの関数は MQSeries Adapter Builder を使って作成されるアダプターによって使用されます。 MQSeries Adapter Kernel Information Center には、 API に関連したオンライン資料 (Javadoc HTML フォーマット) があります。

カーネルは、ユーザーが MQSeries Adapter Builder を使って作成したアダプターと共に使用するように意図されています。カーネルは、カスタム・コードのみでカーネル API を呼び出して使用するようには意図されていません。 API に関するオンライン資料は、カーネル機能の動作の理解の補助および診断の補助としてのみ提供されています。

カーネル API のオンライン資料は、 `documentation` ディレクトリーにあります。

第6章 追加情報の入手

MQSeries Adapter Offering を使用する際に有用な情報源がいくつかあります。MQSeries Adapter Kernel に関する追加情報は、「*問題判別ガイド*」資料を参照してください。製品とともにインストールされる MQSeries Adapter Kernel Information Center にあります。「*問題判別ガイド*」には、カーネルの使用時に起きる特定の問題の解決方法についての情報があります。 MQSeries Adapter Builder に関する情報は、その製品の Information Center およびオンライン・ヘルプを参照してください。

インターネットでの入手

MQSeries 製品ファミリーの Web サイトは、www.ibm.com/software/ts/mqseries/ です。この Web サイトのリンクをたどれば、次のことを行うことができます。

- MQSeries Adapter Offering など、MQSeries 製品ファミリーについての最新情報を入手できる。
- MQSeries の資料 (HTML および PDF フォーマット) にアクセスできる。本書の新しい版が含まれている場合もあります。MQSeries ライブラリー・ページへの直接のリンク先は、 http://www.ibm.com/software/ts/mqseries/library/manualsa/ です。
- MQSeries SupportPacs をダウンロードできる。

OS/400 上での MQSeries の使用に関する情報は、OS/400 ライブラリー http://www.ibm.com/servers/eserver/iseries/library/ を参照してください。MQSeries ライブラリー Web サイト http://www.ibm.com/software/ts/mqseries/library/manualsa/ から入手できる OS/400 の特別資料も参照してください。

参考資料

次のサイトの参考資料には、本書で触れたトピックの記述があります。

- Open Applications Group Web サイト、www.openapplications.org/
- Extensible Markup Language (XML) 1.0 W3C 勧告 www.w3.org/TR/1998/Rec-xml-19980210

これらは IBM のサイトではありません。

付録A. 通信モード

この付録では、MQSeries Adapter Kernel がサポートする通信モード、およびそれをサポートするために使用する Java クラスに関する情報を提供します。通信モードによっては、デフォルト・フォーマッターがあるコンビニエンス・モードとして提供されるものもあります。コンビニエンス・モードで使用されるデフォルト・フォーマッターについては、 100ページの表10 を参照してください。

次の通信モードをサポートしています。

MQPP MQSeries 基本サービスを使用するカーネル・トランスポート・メッセージ。これはコンビニエンス・モードです。

MQRFH1 MQSeries を使用するカーネル・トランスポート・メッセージ、および MQSeries Integrator バージョン 1.1 を使用するブローカー・メッセージ。これはコンビニエンス・モードです。

MQRFH2 MQSeries を使用するカーネル・トランスポート・メッセージ、および MQSeries Integrator バージョン 2 を使用するブローカー・メッセージ。これはコンビニエンス・モードです。

MQBD MQSeries 基本サービスを使用するカーネル・トランスポート・メッセージ。ただし、本体データのみの送信と受信を行う。これはコンビニエンス・モードです。このモードには次のような独自の特性があります。

- 本体データのみを送信することができます。メッセージ・ヘッダー値のみの送信は行いません。
- 本体データのみのメッセージを受信することができます。受信メッセージに次のデフォルト・メッセージ・ヘッダー値を使用します。
  - `SourceLogicalApplicationID`-- 受信メソッドの呼び出しに使用する `ENAService` オブジェクトにある値。
  - `BodyCategory`-- 受信メソッドの呼び出しに使用する `ENAService` オブジェクトにある値。
  - `BodyType`-- 受信メソッドの呼び出しに使用する `ENAService` オブジェクトにある値。
  - `Acknowledgment`-- 受信した MQMessage が MQSeries REQUEST の場合、 `Acknowledgment` は `1` に設定されます。
  - `BodyData`--MQSeries から受信するメッセージ・データ。

  他のヘッダー値はすべて通常のデフォルト値を使用します。

MQ MQSeries 基本サービスを使用するカーネル・トランスポート・メッセージ。

JMS Java Message Service (JMS) を使用して、カーネルはメッセージをトランスポートします。MQSeries Adapter Kernel での JMS オブジェクトの使用についての情報は、102ページの『JMS オブジェクト記憶の使用』を参照してください。

FILE カーネルはメッセージをファイルに書き込み、メッセージをファイルから取得します。このモードは、診断の目的のみに使用します。

表9 は通信モード、および それらをサポートする Java クラスのリストです。Java クラスはすべて Java パッケージ `com.ibm.epic.adapters.eak.nativeadapter` にあります。なお、論理メッセージ・サービス (LMS) をサポートする Java クラスは、いずれも通信モードとして指定することができます。この場合、クラスそのものは通信をサポートするために使用されるので注意してください。

表 9. 通信モードおよびサポートする Java クラス

通信モード	Java クラス	注
MQPP	`LMSMQBindingMQPP`	MQSeries のインストールが必要
MQRFH1	`LMSMQBindingMQRFH1`	MQSeries のインストールが必要
MQRFH2	`LMSMQBindingMQRFH2`	MQSeries のインストールが必要
MQBD	`LMSMQMQBD`	MQSeries のインストールが必要
MQ	`LMSMQBinding`	MQSeries のインストールが必要
JMS	`LMSJMS`	JMS のインストールが必要
FILE	`LMSFile`	なし

表10 は通信モード、およびそれらと関連したフォーマッター・インターフェースのリストです。101ページの表11 は、フォーマッター・インターフェース、フォーマッター・クラス名、およびそれらの使用についての相互参照リストです。フォーマッターはすべて Java パッケージ `com.ibm.epic.adapters.eak.nativeadapter` にあります。なお、いずれのフォーマッター・クラスも通信モード用に指定することができます。この場合、指定したフォーマッター・クラスはフォーマッターとして使用されるので注意してください。

表 10. 通信モードおよびフォーマッター・インターフェース

通信モード	フォーマッター・インターフェース	デフォルト・フォーマッター
MQPP	`MQFormatterInterface`	`MQNMXMLFormatter`
MQRFH1	`MQFormatterInterface`	`MQNMRFH1Formatter`
MQRFH2	`MQFormatterInterface`	`MQNMRFH2Formatter`

表 10. 通信モードおよびフォーマッター・インターフェース (続き)

通信モード	フォーマッター・ インターフェース	デフォルト・フォーマッター
MQBD	MQFormatterInterface	MQNMBDFormatter
MQ	MQFormatterInterface	MQNMXMLFormatter
JMS	JMSFormatterInterface	JMSNMRFH2Formatter
FILE	StringFormatterInterface	NMXMLFormatter

表 11. フォーマッター・インターフェース、フォーマッター・クラス名、および目的

フォーマッター・ インターフェース	フォーマッター・クラス名	目的
MQFormatterInterface	MQNMXMLFormatter	XML での EpicMessage
	MQNMRFH1Formatter	RFH1 での EpicMessage
	MQNMRFH2Formatter	RFH2 での EpicMessage
	MQNMDBFormatter	本体データのみ
JMSFormatterInterface	JMSNMXMLFormatter	XML での EpicMessage
	JMSNMRFH2Formatter	RFH2 での EpicMessage
	JMSNMBDFormatter	本体データのみ
StringFormatterInterface	NMXMLFormatter	XML での EpicMessage

表12 はサポートしている LMS クラス、およびそれらのトランザクションのサポートの度合いです。MQSeries Adapter Kernel でのトランザクションの使用についての情報は、27ページの『トランザクション機能』を参照してください。

表 12. LMS クラスおよびトランザクションのサポート

LMS クラス	トランザクションのサポート
LMSMQBindingMQPP	単一フェーズ
LMSMQBindingMQRFH1	単一フェーズ
LMSMQBindingMQRFH2	単一フェーズ
LMSMQMQBD	単一フェーズ
LMSMQBinding	単一フェーズ
LMSJMS	単一フェーズ
LMSFILE	サポートなし

JMS オブジェクト記憶の使用

JMS オブジェクトの名前は、MQSeries JMS SupportPac の一部として提供される JNDI の FSContext ファイルのインプリメンテーションを通して保管します。カーネルが FSContext に使用するコンテキスト (ディレクトリー構造) は、LDAP の階層制に従って、関連した値で区別できる属性をディレクトリー名に使用します。たとえば、LDAP の階層 o=ePIC, o=ePICApplications, epicappid=TEST1 の場合、ディレクトリー構造は o-ePIC/o-ePICApplications/epicappid-TEST1 になります。

コンテキストとオブジェクトを作成するには、JMS のインストールで提供される JMS Admin ツールを使用します。基本ステップは、コンテキストを定義すること、それからそのコンテキストを変更することです。コンテキストを変更するには、コンテキストの理解が必要です。適切な場所に JMS オブジェクトを作成します。以下に、コンテキスト構造と JMS オブジェクトを作成するコマンドの例を示します。この例では、アプリケーション ID は TEST1 です。

```
#
# aqmjmscreatesample.scp 1.00 09Mar01
# Used for MQSeries Adapter Kernel
# Sample AQM JMS Configuration.
#
# Copyright (c) 2001 International Business Machines. All Rights Reserved.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# This is a script to use with the JMS administration (JMSAdmin) tool
# which comes with MQSeries Support pac MA88.
# This tool requires the JMSAdmin.config to be set to use either
# FSCONTEXT (file) or LDAP. This script is setup to work with
# FSCONTEXT, but will work with LDAP with the following changes:
# - Change the "-" signs to "=".  Example: define ctx(o-ePIC)
#   becomes define ctx(o=ePIC)
# - In LDAP the contexts have to already be defined using the
#   LDAP administration tool.  For example you do not need
#   to "define ctx(o=ePIC) but only change into it with the
#   "change ctx(o-ePIC)" command.
# - There are some notes in the following script which highlight
#   differences when using LDAP.
#
#
# Example usage: MQSeries root¥java¥bin¥jmsadmin.bat < aqmjmscreatesample.scp
#
# Some helpful commands:
# "display ctx" will display the context's of the context you are
```

```
# currently in.
# "=UP" means return to the parent context.  Example: change ctx(=UP)
# "=INIT" means return to root context. In this example one directory level
# above o-ePIC.  Example: change ctx(=INIT)
# "define xxx" is for creating either a context or object.
# "change xxx" is for changing/moving into the context.
#
# Always required.
define ctx(o-ePIC)
change ctx(o-ePIC)
# Always required.
define ctx(o-ePICApplications)
change ctx(o-ePICApplications)
# Application id is TEST1, requires a context.
define ctx(epicappid-TEST1)
change ctx(epicappid-TEST1)
# Always required.
define ctx(cn-epicadapterrouting)
change ctx(cn-epicadapterrouting)
# This will hold the JMS QueueConnectionFactory object.
# Note: These two steps are not required for LDAP.
define ctx(cn-QCFTEST1)
change ctx(cn-QCFTEST1)
# Create the JMS QueueConnectionFactory object whose name is QCFTEST1
# Using MQSeries in server (bindings) mode.
define qcf(QCFTEST1) qmgr(yourQManagerName) tran(BIND)
change ctx(=UP)
# BodyCategory is DEFAULT
define ctx(epicbodycategory-DEFAULT)
change ctx(epicbodycategory-DEFAULT)
# BodyType is DEFAULT
define ctx(epicbodytype-DEFAULT)
change ctx(epicbodytype-DEFAULT)
# This will hold the JMS Queue object whose name is TEST1AIQ.
# Note: These two steps are not required for LDAP.
define ctx(cn-TEST1AIQ)
change ctx(cn-TEST1AIQ)
# Create the JMS Queue object whose name is TEST1AIQ
# q(JMS Q Object Name) queue(MQSeries Queue name)
define q(TEST1AIQ) queue(TEST1AIQ)
# Can move up and define other contexts and JMS objects.
# Quit the administration tool.
end
```

付録B. 妥当性検査済みの構成

MQSeries、MQSeries Adapter Offering、および MQSeries Integrator には、多くの構成と組み合わせが考えられます。MQSeries 製品ファミリーのこれらの各製品は、豊富な機能と構成を持っています。さらに MQSeries、MQSeries Adapter Offering、および MQSeries Integrator の機能性を結合することができます。MQSeries 製品ファミリーのある製品のいくつかの機能性は、ファミリーの他の製品が備えている機能性と部分的にオーバーラップしています。MQSeries、MQSeries Adapter Offering、および MQSeries Integrator の異なるメッセージ・ルーティング機能と送達機能を、どのように使い、結合させるかを判断する必要があります。

MQSeries、MQSeries Adapter Offering、および MQSeries Integrator の次に示す構成は、本書の発行時点で妥当性を検査済みです。妥当性検査済みの最新の構成は、MQSeries Web サイトを参照してください。

MQSeries アダプター・カーネル:

- 肯定応答要求あり、および肯定応答要求なしでメッセージを送信。
- MQSeries または JMS 通信モードを使用。有効な通信モードについては、99ページの『付録A. 通信モード』を参照してください。
- メッセージのルーティングおよび送達:
  - 1 つのソース・アダプターから 1 つのターゲット・アダプターにメッセージを送信
  - 1 つのソース・アダプターから複数のターゲット・アダプターにメッセージを送信
  - マルチスレッドのメッセージ送達 (複数ワーカー)
  - 本体カテゴリー、本体タイプ、およびソース論理 ID を基に、カーネルの構成ファイルを使用して宛先論理 ID を判別するために、メッセージの中の宛先論理 ID は NONE に設定
  - 送達は push モデルで
  - トレースを使用可能に

  注: 107ページの『付録C. メッセージ・ヘッダー』を参照してください。そこには、カーネルが取り込んで処理を行う MQSeries Adapter Kernel メッセージ・ヘッダー・フィールドのリストと説明があります。
- 31ページの『ハードウェア』および 32ページの『ソフトウェア』に示された前提条件。
- 構成を定義するためには LDAP ではなく、構成ファイルを使用。

MQSeries:

- MQSeries クラスターは使用しない。

注: 107ページの『付録C. メッセージ・ヘッダー』を参照してください。そこには、カーネルが取り込んで処理を行う MQSeries Adapter Kernel メッセージ・ヘッダー・フィールドのリストと説明があります。

MQSeries Integrator:

- MQSeries Adapter Kernel と MQSeries が、MQSeries Integrator へのメッセージのルーティングと送達を行う。MQSeries Integrator のメッセージのブローカー機能を判断するには、MQSeries Integrator の情報を参照してください。
- カーネルのソース・サイドから、MQSeries と MQSeries Integrator バージョン 2 を使って、直接ルーティングで、カーネルのターゲット・サイドにメッセージを送信。MQSeries Integrator のメッセージ・フローでのルーティングは、静的な経路指定として構成されます。フローの MQInput ノードに到着するメッセージはすべて、特定の MQOutput キューへの直接経路が指定されます。

  注: 107ページの『付録C. メッセージ・ヘッダー』を参照してください。そこには、カーネルが取り込んで処理を行う MQSeries Adapter Kernel メッセージ・ヘッダー・フィールドのリストと説明があります。

付録C. メッセージ・ヘッダー

MQSeries Adapter Offering では、いくつかのメッセージ・ヘッダーを使用します。どの環境でどのヘッダーが使用されるかは、12ページの『メッセージおよびメッセージ・フォーマット』を参照してください。

この付録ではメッセージ・ヘッダー・フィールドをリストし、説明します。

MQSeries Adapter Kernel メッセージ・ディスクリプター・ヘッダー

MQSeries Adapter Kernel で使用するヘッダー値。これらの値はメッセージ・ホルダー・オブジェクトに置かれます。 **応答で伝搬されるか?** カラムでは、ソース・アプリケーションが応答を要求する場合に、応答メッセージの中で値がソース・アプリケーションに伝搬し戻されるかどうかをについて、リストしています。WebSphere Business Integrator でしか使用されない値もあります。

表 13. MQSeries Adapter Kernel ヘッダー

ヘッダー名	応答で伝搬されるか?	意味または使用法
UniqueID	いいえ	それぞれのメッセージごとの固有 ID (UID)。
TransactionID	はい	それぞれのメッセージ、および応答がある場合はその応答に共有されるトランザクション ID。 Extricity PublicProcessID または DataInterchange (DI) ApplicationID と同等。
MessageType	いいえ	ゲートウェイおよび log/trace/exception メッセージに使用。
SourceLogicalID	いいえ	ソース・アプリケーションの論理 ID。 DI、Partner Agreement Manager (PAM)、および Business Flow Manager (BFM) の予約名と同等。
DestinationLogicalID	いいえ	ターゲット・アプリケーションの論理 ID。DI および PAM の場合、デフォルト値は none、しかしオーバーライド可能。

表 13. MQSeries Adapter Kernel ヘッダー (続き)

ヘッダー名	応答で伝搬されるか?	意味または使用法
RespondToLogicalID	はい	応答メッセージが送信される論理 ID。DI の場合、DestinationLogicalID にコピーされ、PAM の場合、SourceLogicalID にコピーされる。
CorrelationID	いいえ	使用予約済み。
GroupStatus	いいえ	使用予約済み。
ProcessingCategory	いいえ	PAM Public Process ID または DI Command Process ID と同等。
QosPolicy	いいえ	使用予約済み。
DeliveryCategory	いいえ	DI RequestorProfileID と同等。
AckRequested	いいえ	ソース・アプリケーションが応答メッセージを要求するかどうかを決定。
PublicationTopic	いいえ	使用予約済み。
SessionID	いいえ	DI BatchID と同等。
EncryptionStatus	いいえ	本体暗号化のタイプおよびシグニチャーを決定。
TimeStampCreated	いいえ	メッセージが作成された日時。
TimeStampExpired	いいえ	すでにメッセージの意味がなくなった日時。 -1 値は有効期限なしの意味。
Size	いいえ	使用予約済み。
BodyType	いいえ	メッセージの特定の目的を示す。
BodyCategory	いいえ	メッセージのアプリケーション・タイプを示す。
BodySecondaryType	いいえ	使用予約済み。
UserArea	いいえ	ユーザー・データのための汎用領域。
RelatedSubjectID	いいえ	プロセス間の相関に使用。
ExternalID	いいえ	アプリケーション環境外の現行所有者の ID (たとえば、ユーザーまたは取引先)。
InternalID	いいえ	アプリケーション環境内の現行所有者の ID (たとえば、ユーザーまたは取引先)。

表 13. MQSeries Adapter Kernel ヘッダー (続き)

ヘッダー名	応答で伝搬されるか?	意味または使用法
BodySignature	いいえ	使用予約済み。
TransportCorrelationID	はい	使用予約済み。

MQSeries メッセージ・ディスクリプター・ヘッダー

フィールドの内容は MQSeries によって決定します。MQSeries Adapter Offering は、メッセージ制御値で決定したとおりにキューにメッセージを書き込みます。詳細は、16ページの『メッセージ制御値』を参照してください。

表 14. MQSeries ヘッダー

セクションまたはフィールド	意味または使用法
Revision	固定。
UniqueID	各メッセージが独自 ID を所有。
TransactionID	メッセージとその応答が同一トランザクション ID を共用。
MessageType	使用予約済み。
SourceLogicalID	ソース・アプリケーションの論理 ID。
DestinationLogicalID	ターゲット・アプリケーションの論理 ID。
RespondToLogicalID	応答メッセージが送信される論理 ID。
CorrelationID	使用予約済み。
GroupStatus	使用予約済み。
ProcessingCategory	使用予約済み。
QosPolicy	使用予約済み。
DeliveryCategory	使用予約済み。
AckRequested	ソース・アプリケーションが応答を要求するかどうかを決定。
PublicationTopic	使用予約済み。
SessionID	使用予約済み。
EncryptionStatus	使用予約済み。
TimeStampCreated	メッセージが作成された日時。
TimeStampExpired	すでにメッセージの意味がなくなった日時。
Size	使用予約済み。
BodyCategory	メッセージのアプリケーション・タイプを表す (たとえば OAG や RosettaNet)。
BodyType	メッセージの特定の目的を表す (たとえば「販売オーダーの追加」や「在庫の同期化」)。
BodySecondaryType	予約済み。

表 14. *MQSeries* ヘッダー *(続き)*

UserArea	ユーザー・データのための汎用領域。
BodyData	メッセージ本体データ。

MQSeries Integrator なしの MQSeries

カーネル・ヘッダー値と本体データは XML 文書に書き込まれます。以下に XML 文書を記述した DTD の例を示します。

```
<!ELEMENT EPICHEADER (HEADER, EPICBODY,USERAREA*)>
<!ELEMENT HEADER (#PCDATA)>
<!ATTLIST HEADER Revision CDATA #FIXED "001">
<!ATTLIST HEADER UniqueID CDATA #REQUIRED>
<!ATTLIST HEADER TransactionID CDATA #REQUIRED>
<!ATTLIST HEADER MessageType CDATA #REQUIRED>
<!ATTLIST HEADER SourceLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER DestinationLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER RespondToLogicalID CDATA #IMPLIED>
<!ATTLIST HEADER CorrelationID CDATA #IMPLIED>
<!ATTLIST HEADER GroupStatus CDATA #IMPLIED>
<!ATTLIST HEADER ProcessingCategory CDATA #IMPLIED>
<!ATTLIST HEADER QosPolicy CDATA #IMPLIED>
<!ATTLIST HEADER DeliveryCategory CDATA #IMPLIED>
<!ATTLIST HEADER AckRequested CDATA #IMPLIED>
<!ATTLIST HEADER PublicationTopic CDATA #IMPLIED>
<!ATTLIST HEADER SessionID CDATA #IMPLIED>
<!ATTLIST HEADER EncryptionStatus CDATA #IMPLIED>
<!ATTLIST HEADER TimeStampCreated CDATA #REQUIRED>
<!ATTLIST HEADER TimeStampExpired CDATA #REQUIRED>
<!ATTLIST HEADER Size CDATA #IMPLIED>
<!ELEMENT EPICBODY (#PCDATA)>
<!ATTLIST EPICBODY Size CDATA #IMPLIED>
<!ATTLIST EPICBODY BodyType CDATA #REQUIRED>
<!ATTLIST EPICBODY BodyCategory CDATA #REQUIRED>
<!ATTLIST EPICBODY BodySecondaryType CDATA #IMPLIED>
<!ELEMENT USERAREA (#PCDATA) >
```

MQSeries Integrator バージョン 1 ヘッダー

MQSeries Integrator バージョン 1 ヘッダー RFH1 は、次の項目から成ります。

1. 固定部分
2. Neon ヘッダー
3. データ・セクション (カーネル・ヘッダーとメッセージ本体データを含む)

表 15. MQSeries Integrator バージョン 1 ヘッダー - RFH1

セクションまたはフィールド	意味または使用法
固定部分	MQSeries Integrator バージョン 1.1 の指定で使用。
Neon ヘッダー	Neon ヘッダー・フォーマットに従う。
OPT_APP_GRP	SourceLogicalId 値。カーネル・ヘッダーから引用。
OPT_MSG_TYPE	BodyCategory+BodyType。カーネル・ヘッダーから派生。 例: BodyCategory が OAG で BodyType が SyncItem の場合、値は OAG+SyncItem。
データ・セクション	メッセージ本体データの前のカーネル・ヘッダー値から成る。
カーネル・ヘッダー	カーネル・ヘッダーはタグで囲まれる。 `<EPICHEADER>`*`header`*`</EPICHEADER>`。 カーネル・ヘッダー値は XML 構文で表す。あるのは値を持つ属性のみ。実データの行は分離しない。値のフォーマットの例: `<MessageType>`*`value`*`</MessageType>`
MessageType	使用予約済み。
SourceLogicalID	ソース・アプリケーションの論理 ID。
DestinationLogicalID	ターゲット・アプリケーションの論理 ID。
RespondToLogicalID	応答メッセージが送信される論理 ID。
TimeStampCreated	メッセージが作成された日時。
TimeStampExpired	すでにメッセージの意味がなくなった日時。
TransactionID	メッセージとその応答が同一トランザクション ID を共用。
UniqueID	各メッセージが独自 ID を所有。
AckRequested	ソース・アプリケーションが応答を要求するかどうかを決定。
ProcessingCategory	予約済み。
BodyCategory	メッセージのアプリケーション・タイプを表す (たとえば OAG や RosettaNet)。
BodyType	メッセージの特定の目的を表す (たとえば「販売オーダーの追加」や「在庫の同期化」)。
BodySecondaryType	予約済み。
UserArea	ユーザーの統合に固有のアプリケーション・データ。

表 15. *MQSeries Integrator バージョン 1 ヘッダー - RFH1 (続き)*

MsgHeaderVersion	カーネル・ヘッダー・バージョン (予約済み)。
CorrelationID	ユーザーの統合に固有。
GroupStatus	ユーザーの統合に固有。
QosPolicy	予約済み。
DeliveryCategory	予約済み。
PublicationTopic	予約済み。
SessionID	予約済み。
EncryptionStatus	予約済み。
メッセージ本体データ	メッセージ本体データ。

MQSeries Integrator バージョン 2 ヘッダー

MQSeries Integrator バージョン 2 ヘッダー RFH2 は、次の項目から成ります。

1. 固定部分
2. <mcd> フォルダー - メッセージ内容ディスクリプター
3. <usr> フォルダー - アプリケーション (ユーザー) 定義のプロパティー
4. データ・セクション (カーネル・ヘッダーとメッセージ本体データを含む)

表 16. *MQSeries Integrator バージョン 2 ヘッダー - RFH2*

セクションまたはフィールド	意味または使用法
固定部分	MQSeries Integrator バージョン 2 の指定で使用。
<mcd>	メッセージが XML の場合は XML。 MQSeries Integrator バージョン 2 のルールに従う。
セット	カーネルでは使用しない。
タイプ	カーネルでは使用しない。
フォーマット	メッセージが XML の場合は XML。 MQSeries Integrator バージョン 2 のルールに従う。
<usr> フォルダー - アプリケーション (ユーザー) 定義のプロパティー	カーネル・ヘッダー値から成る。
カーネル・ヘッダー	あるのは値を持つ属性のみ。実データの行は分離しない。
SourceLogicalID	ソース・アプリケーションの論理 ID。
DestinationLogicalID	ターゲット・アプリケーションの論理 ID。
MessageType	使用予約済み。
RespondToLogicalID	応答メッセージが送信される論理 ID。
TimeStampCreated	メッセージが作成された日時。
TimeStampExpired	すでにメッセージの意味がなくなった日時。

表 16. *MQSeries Integrator バージョン 2 ヘッダー - RFH2 (続き)*

TransactionID	メッセージとその応答が同一トランザクション ID を共用。
UniqueID	各メッセージが独自 ID を所有。
ProcessingCategory	予約済み。
BodyCategory	メッセージのアプリケーション・タイプを表す (たとえば OAG や RosettaNet)。
BodyType	メッセージの特定の目的を表す (たとえば「販売オーダーの追加」や「在庫の同期化」)。
BodySecondaryType	予約済み。
AckRequested	ソース・アプリケーションが応答を要求するかどうかを決定。
UserArea	ユーザーの統合に固有のアプリケーション・データ。
MsgHeaderVersion	カーネル・ヘッダー・バージョン (予約済み)。
CorrelationID	ユーザーの統合に固有。
GroupStatus	ユーザーの統合に固有。
QosPolicy	予約済み。
DeliveryCategory	予約済み。
PublicationTopic	予約済み。
SessionID	予約済み。
EncryptionStatus	予約済み。
データ・セクション	メッセージ本体データ。

付録D. サンプル構成ファイル

このセクションでは、aqmconfig.xml ファイルの、本書の発行時点のバージョンをリストします。119ページの『サンプル最小構成ファイル』には、aqmconfig.minimum.xml ファイルの、本書の発行時点のバージョンをリストしています。ここに載っている例が古くなっている場合もあるので、最新のバージョンについては、インストールしたカーネルの samples ディレクトリーにあるファイル aqmconfig.xml および aqmconfig.minimum.xml を参照してください。

構成ファイルの解釈と編集に関しての情報は、64ページの『構成ファイル』を参照してください。

この構成ファイルの例には、いくつかのアプリケーション ID が含まれています。エントリーのセットは、各アプリケーション ID の下にリストしています。このサンプル構成ファイルには、次のアプリケーション ID が含まれています。

- TEST1
- TEST1Daemon
- TEST2
- TEST3
- TraceClient
- TraceServer

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.xml 1.01 09Mar01                                          -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Sample AQM Configuration.           -->
<!--  -->
<!-- Copyright (c) 2001 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CoprightVersion 1.0 -->
<!--  -->
   <Epic o="ePIC">
      <!-- If getObject is called this indicates the top level directory -->
      <!-- where the JNDI file system context will retrieve objects from. -->
      <!-- This defaults to the current directory if this key is not present. -->
      <!-- All applications share this context root. -->
      <context>file:///epic/configContext</context>
      <!-- Example using a drive letter 'c' -->
      <!--
      <context>file://c:/E/runtimefiles</context>
      -->
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1 with a  -->
         <!-- sample AdapterDaemon named TEST1Daemon  -->
         <ePICApplication epicappid="TEST1">
     <!--  Audit Logging on/off.  Requires WebSphere Business Integrator product. -->
     <!--  If no entry defaults to false. -->
           <epiclogging>false</epiclogging>
     <!--  Tracing on/off.  If no entry defaults to false. -->
           <epictrace>false</epictrace>
     <!-- Trace levels - Uses the jlog com.ibm.logging.IRecordType constants, -->
     <!-- common constants: -->
     <!-- 0=TYPE_NONE (No messages), 1=TYPE_INFO, 512=TYPE_ERROR_EXC (Exceptions), -->
```

```
<!-- 513=TYPE_INFO | TYPE_ERROR_EXC, -1=TYPE_ALL (All possible messages). -->
<!-- No entry defaults to TYPE_NONE -->
      <epictracelevel>-1</epictracelevel>
<!-- Name of the Trace application id.  Will be used for -->
<!-- trace configuration information.  Defaults to TraceClient -->
      <epictraceclientid>TraceClient</epictraceclientid>
<!-- When processing messages into the application. -->
<!-- LogonInfo class name used for connecting to an application. -->
  <!-- Will be used by the AdapterDaemon.  If no entry will default -->
<!-- to com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault. -->
<epiclogoninfoclassname>com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault
 </epiclogoninfoclassname>
       <AdapterRouting cn="epicadapterrouting">
          <!-- MQSeries Q Manager for this application use, no entry -->
          <!-- uses the default Q Manager.  A value of DEFAULT means -->
          <!-- use the default Q Manager. -->
          <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
          <!-- Use the remote Q Manager for sending messages.  Remote queue -->
          <!-- definitions are not required.  true - use remote Q Manager, -->
          <!-- false - do not use remote Q Manager.  No entry defaults to false -->
          <epicuseremotequeuemanagertosend>false</epicuseremotequeuemanagertosend>
          <!-- MQSeries Client hostname for where the MQSeries server -->
          <!-- resides for TEST1.  Required if using MQSeries Client -->
          <!--
          <epicmqppqueuemgrhostname>localhost</epicmqppqueuemgrhostname>
   -->
          <!-- MQSeries Client port to use for where the MQSeries server -->
          <!-- resides for TEST1.  No entry defaults to MQSeries default -->
          <!--
          <epicmqppqueuemgrportnumber>1414</epicmqppqueuemgrportnumber>
   -->
          <!-- MQSeries Client channel name to use for the MQSeries server, required -->
          <!--
          <epicmqppqueuemgrchannelname>xyz</epicmqppqueuemgrchannelname>
   -->
          <!-- JMS example for TEST1.  Refers to the JMS Connection factory name. -->
          <!-- Requires the attribute describing the object plus the attributes value. -->
          <!-- For JMS the attribute is 'cn'. -->
          <!--
          <epicjmsconnectionfactoryname>cn=QCFTEST1</epicjmsconnectionfactoryname>
   -->
          <ePICBodyCategory epicbodycategory="DEFAULT">
             <ePICBodyType epicbodytype="DEFAULT">
<!-- Contains the Command selection criteria when processing -->
<!-- a message into an application.  Will be used by the -->
<!-- AdapterDaemon - Command to invoke. -->
                <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                </epiccommandclassname>
<!-- Represents the type of command the "epiccommandclassname" -->
<!-- represents.  MQAKEAB is the EAB style interface.  MQAKEJB -->
<!-- is an EJB Service Session Bean.  No entry defaults to MQAKEAB -->
                <epiccommandtype>MQAKEAB</epiccommandtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- method name to invoke.  No entry defaults to "execute". -->
                <epiccommandejbmethod>execute</epiccommandejbmethod>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- parameter type for the method specified by "epiccommandejbmethod". -->
<!-- This will be the same datatype returned by the -->
<!-- TerminalDataContainerMapper.  No entry defaults -->
<!-- to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
                <epiccommandejbmethodparmtype>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
                </epiccommandejbmethodparmtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- URL where the EJB specified in "epiccommandclassname" -->
<!-- has been deployed in the form "IIOP://hostname:900/". -->
<!-- No entry defaults to "IIOP:///". -->
                <epiccommandejburl>IIOP:///</epiccommandejburl>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Initial Context Factory used to lookup the -->
<!-- home name for the EJB specified in "epiccommandclassname". -->
<!-- No entry defaults to "com.ibm.ejs.ns.jndi.CNInitialContextFactory". -->
                <epiccommandejbinitialcontext>com.ibm.ejs.ns.jndi.CNInitialContextFactory
                </epiccommandejbinitialcontext>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Mapper the Worker uses for creating the  -->
<!-- "epiccommandejbmethodparmtype" object passed in to the  -->
<!-- "epiccommandejbmethod" in to the "epiccommandclassname". -->
<!-- No entry defaults to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
                <epiccommandejbmapper>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
                </epiccommandejbmapper>
<!-- Default destinations to send messages to. -->
        <!-- Single destination. -->
                <epicdestids>TEST2</epicdestids>
        <!-- Multiple destinations. -->
                <!--
```

```
                <epicdestids>
                     <Value>TEST2</Value>
                     <Value>TEST3</Value>
                </epicdestids>
                -->
                <!-- Receive transport communications mode this application -->
                <!-- wants for receiving messages. -->
                <!-- For MQSeries normal mode use MQPP. -->
                <!-- For MQSeries using an RFH1 header format use MQRFH1, -->
                <!-- when using MQSeries Integrator V1 -->
                <!-- For MQSeries using an RFH2 header format use MQRFH2, -->
                <!-- when using MQSeries Integrator V2 -->
                <!-- For file normal mode use FILE. -->
                <epicreceivemode>MQPP</epicreceivemode>
                <!-- How to format the message for the receive mode. -->
                <!-- Entry is the class name of the formatter which -->
                <!-- must be for the receive mode -->
                <!-- Receive modes MQPP, MQRFH1, MQRFH2, FILE have -->
                <!-- default receive modes -->
                <epicmessageformatter>com.ibm.epic.adapters.eak.nativeadapter.MQNMBDFormatter
                </epicmessageformatter>
<!-- JMS formatter for mode for MQSeries provider implementation -->
<!--
                <epicmessageformatter>com.ibm.epic.adapters.eak.nativeadapter.JMSNMRFH2Formatter
                </epicmessageformatter>
-->
<!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
<!-- -1 means never ending.  No entry defaults to 0 -->
<!-- milliseconds.  Used when receiving messages. -->
                <epicreceivetimeout>30000</epicreceivetimeout>
<!-- MQSeries queue for this application to receive messages -->
<!-- from for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicreceivemqppqueue>TEST1AIQ</epicreceivemqppqueue>
<!-- MQSeries queue required by the AdapterWorker when -->
<!-- errors encountered processing a message -->
<!-- for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicerrormqppqueue>TEST1AEQ</epicerrormqppqueue>
<!-- MQSeries reply queue required for synchronous request/replies -->
<!-- for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicreplymqppqueue>TEST1RPL</epicreplymqppqueue>
<!-- JMS receive mode, refers to the JMS queue object name for -->
<!-- this application to receive messages from. -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmsreceivequeuename>cn=TEST1AIQ</epicjmsreceivequeuename>
<!-- JMS receive mode, refers to the JMS queue object name for -->
<!-- errors required by the AdapterWorker when errors -->
<!-- encountered processing a message. -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmserrorqueuename>cn=TEST1AEQ</epicjmserrorqueuename>
<!-- JMS receive mode, refers to the JMS queue object name for -->
<!-- the reply queue, required for synchronous request/replies -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmsreplyqueuename>cn=TEST1RPL</epicjmsreplyqueuename>
<!-- In FILE receive mode, directory for this application to receive messages from -->
                <epicreceivefiledir>./TEST1AID</epicreceivefiledir>
<!-- In FILE receive mode, interim directory for this application to -->
<!-- hold received messages until committed. -->
                <epiccommitfiledir>./TEST1ACD</epiccommitfiledir>
<!-- In FILE receive mode, directory for this application to put error messages -->
<!-- File receive mode, directory required by the AdapterWorker when -->
<!-- errors encountered processing a message -->
                <epicerrorfiledir>./TEST1AED</epicerrorfiledir>
            </ePICBodyType>
         </ePICBodyCategory>
      </AdapterRouting>
   </ePICApplication>
   <!-- The following is for sample AdapterDaemon 'TEST1Daemon' -->
   <!-- for the 'TEST1' application  -->
   <ePICApplication epicappid="TEST1Daemon">
      <epictrace>false</epictrace>
      <epictracelevel>-1</epictracelevel>
      <ePICAdapterDaemonExtensions cn="epicappextensions">
  <!-- Dependency appid, if no entry then will default -->
  <!-- to the application id of the daemon.     -->
         <epicdepappid>TEST1</epicdepappid>
  <!-- Minimum number of workers the AdapterDaemon will start. -->
  <!-- No entry defaults to 1. -->
        <epicminworkers>1</epicminworkers>
      </ePICAdapterDaemonExtensions>
   </ePICApplication>
   <!-- The following is for Test Application ID: TEST2 -->
   <!-- Refer to TEST1 for explanations and possible additional entries. -->
```

```
      <ePICApplication epicappid="TEST2">
         <epictrace>true</epictrace>
         <epictracelevel>512</epictracelevel>
         <AdapterRouting cn="epicadapterrouting">
            <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
            <ePICBodyCategory epicbodycategory="DEFAULT">
               <ePICBodyType epicbodytype="DEFAULT">
                  <epiccommandclassname>com.ibm.epic.adapters.eak.test.InstallVerificationTest
                  </epiccommandclassname>
                  <epicreceivemode>MQPP</epicreceivemode>
                  <epicreceivemqppqueue>TEST2AIQ</epicreceivemqppqueue>
                  <epicerrormqppqueue>TEST2AEQ</epicerrormqppqueue>
                  <epicreplymqppqueue>TEST2RPL</epicreplymqppqueue>
               </ePICBodyType>
            </ePICBodyCategory>
         </AdapterRouting>
      </ePICApplication>
      <!-- The following is for Test Application ID: TEST3 -->
      <!-- Refer to TEST1 for explanations and possible additional entries. -->
      <ePICApplication epicappid="TEST3">
         <AdapterRouting cn="epicadapterrouting">
            <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
            <ePICBodyCategory epicbodycategory="DEFAULT">
               <ePICBodyType epicbodytype="DEFAULT">
                  <epicdestids>TEST1</epicdestids>
                  <epicreceivemode>MQPP</epicreceivemode>
                  <epicreceivemqppqueue>TEST3AIQ</epicreceivemqppqueue>
               </ePICBodyType>
            </ePICBodyCategory>
         </AdapterRouting>
      </ePICApplication>
<!-- The following is for sample Trace Client Application ID: TraceClient -->
<!-- Contains the TraceClient configuration information for doing tracing. -->
<!-- This is the application id value in the 'epictraceclientid' element -->
<!-- configured for the application wanting to do tracing -->
      <ePICApplication epicappid="TraceClient">
         <ePICTraceExtensions cn="epicappextensions">
      <!-- Dependency Trace Server application id used for SocketHandler -->
      <!-- and ENAHandler (uses MQSeries), defaults to TraceServer -->
            <epicdepappid>TraceServer</epicdepappid>
            <!-- Write messages synchronously (true) or asynchronously (false), -->
            <!-- defaults to false (write messages asynchronously).  This is    -->
            <!-- used when giving the messages to the handlers. -->
            <epictracesyncoperation>false</epictracesyncoperation>
      <!-- Default Trace message file to use if none passed in to the -->
         <!-- writeTrace method call. Defaults to this file if not indicated -->
            <epictracemessagefile>com.ibm.epic.trace.client.TraceMessage</epictracemessagefile>
      <!-- Handlers to load.  Handlers do the actual processing of the -->
      <!-- Trace message.  If the default trace client id 'TraceClient' -->
      <!-- is used then the handler defaults to the -->
      <!-- com.ibm.logging.ConsoleHandler.  If the default trace client -->
      <!-- id 'TraceClient' is not used, the handler has to be specified. -->
      <!-- A Single Trace Handler -->
            <epictracehandler>com.ibm.logging.ConsoleHandler</epictracehandler>
      <!-- Multiple Trace Handlers -->
      <!--
            <epictracehandler>
               <Value>com.ibm.logging.ConsoleHandler</Value>
               <Value>com.ibm.logging.SocketHandler</Value>
            </epictracehandler>
      -->
            <!-- Handler definitions.  Available definitions depend on the -->
            <!-- handler.  Formatters are used for formatting the trace message.-->
            <ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">
               <!-- ConsoleHandler formatter to use, defaults to this formatter if none provided. -->
               <epictraceformatter>com.ibm.epic.trace.client.EpicTraceFormatter</epictraceformatter>
            </ePICTraceHandler>
            <ePICTraceHandler epictracehandler="com.ibm.logging.FileHandler">
               <!-- FileHandler formatter to use, defaults to this formatter if none provided. -->
               <epictraceformatter>com.ibm.epic.trace.client.EpicTraceFormatter</epictraceformatter>
               <!-- Trace filename to use, defaults to trc.log in the current directory. -->
               <epictracefilename>trc.log</epictracefilename>
            </ePICTraceHandler>
            <ePICTraceHandler epictracehandler="com.ibm.epic.trace.client.ENAHandler">
               <!-- ENAHandler formatter to use, defaults to this formatter if none provided. -->
               <epictraceformatter>com.ibm.epic.trace.client.EpicXMLFormatter</epictraceformatter>
            </ePICTraceHandler>
            <ePICTraceHandler epictracehandler="com.ibm.logging.SocketHandler">
               <!-- SocketHandler formatter to use, defaults to this formatter if none provided. -->
               <epictraceformatter>com.ibm.epic.trace.client.EpicXMLFormatter</epictraceformatter>
            </ePICTraceHandler>
         </ePICTraceExtensions>
      </ePICApplication>
<!-- The following is for sample Trace Server Application ID: TraceServer -->
<!-- Contains the TraceServer configuration information. -->
```

```
<!-- This is the application id pointed to by the trace client -->
<!-- epicdepappid value.  Definitions are similar to TraceClient example. -->
       <ePICApplication epicappid="TraceServer">
          <AdapterRouting cn="epicadapterrouting">
             <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqpppqueue>TraceServerAIQ</epicreceivemqpppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
          <ePICTraceExtensions cn="epicappextensions">
       <!-- Write messages synchronously/asynchronously (true/false (default)). -->
             <epictracesyncoperation>false</epictracesyncoperation>
       <!-- Trace message file.  Defaults to this file if not indicated -->
             <epictracemessagefile>com.ibm.epic.trace.server.TraceServerMessage</epictracemessagefile>
             <!-- Handlers to load, for multiple handlers see TraceClient example. -->
             <!-- If the default trace server id 'TraceServer' is used then the handler -->
             <!-- defaults to the com.ibm.logging.MultiFileHandler. -->
             <!-- Note: Do not use SocketHandler or ENAHandler for the trace server. -->
             <epictracehandler>com.ibm.logging.MultiFileHandler</epictracehandler>
       <!-- Handler definitions for com.ibm.logging.SocketHandler -->
       <!-- Formatter to use, defaults to this formatter if none provided.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.SocketHandler">
  <!-- Entries when using socket handler from the TraceClient and -->
  <!-- starting the Trace Server in socket receive mode. -->
  <!-- SocketHandler host machine, defaults to localhost -->
                <epictracesocketserverhost>localhost</epictracesocketserverhost>
          <!-- SocketHandler port number, defaults to 8181 -->
                <epictraceportnumber>8181</epictraceportnumber>
             </ePICTraceHandler>
       <!-- Formatter to use, defaults to this formatter if none provided. -->
             <ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">
          <!-- ConsoleHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.ReFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.MultiFileHandler">
  <!-- MultiFileHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.ReFormatter</epictraceformatter>
  <!-- MultiFileHandler trace base filename to use, defaults to trc.log in the -->
  <!-- current directory.  The actual filename will be for this -->
  <!-- example trcx.log, where x is a numeric number starting at -->
  <!-- 0 and going up to the number of trace files specified. -->
                <epictracefilename>trc.log</epictracefilename>
  <!-- MultiFileHandler number of trace files, defaults to 3 -->
                <epictracefilenumber>3</epictracefilenumber>
  <!-- MultiFileHandler file size in number of bytes, defaults to -->
                <epictracefilesize>1000000</epictracefilesize>
             </ePICTraceHandler>
          </ePICTraceExtensions>
       </ePICApplication>
    </ePICApplications>
```

サンプル最小構成ファイル

このセクションでは、MQSeries Adapter Kernel で使用する 最小構成ファイルの例を提示します。最小構成ファイルについての情報は、83ページの『構成へのアダプター情報の追加』を参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.minimum.xml 1.00 00/11/07                                    -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Sample AQM Configuration showing a minimum configuration for the       -->
<!-- following conditions:                                                   -->
<!-- 1) Going from applicationid TEST1 to TEST2.  TEST1 is not receiving  -->
<!--    messages. -->
<!-- 2) TEST2 has no special application requirements.  -->
<!-- 3) TEST2 is using 1 worker.                        -->
<!-- 4) Using MQSeries with the default QManager installed on each machine.    -->
<!--    and using default format.                                           -->
<!-- 5) No specific body category and body type being used.                 -->
<!-- 6) Using default tracing to the console.   -->
<!--  -->
<!--  -->
<!--  -->
<!-- Copyright (c) 2000 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
```

```
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--    -->
   <Epic o="ePIC">
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1  -->
         <ePICApplication epicappid="TEST1">
     <!--  Tracing on/off.  If no entry defaults to false. -->
           <epictrace>false</epictrace>
     <!-- Trace levels - 512=TYPE_ERROR_EXC (Exceptions),-1=TYPE_ALL (All possible messages). -->
           <epictracelevel>0</epictracelevel>
           <AdapterRouting cn="epicadapterrouting">
              <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
              <ePICBodyCategory epicbodycategory="DEFAULT">
                 <ePICBodyType epicbodytype="DEFAULT">
     <!-- Default destinations to send messages to. -->
                    <epicdestids>TEST2</epicdestids>
                 </ePICBodyType>
              </ePICBodyCategory>
           </AdapterRouting>
         </ePICApplication>
         <!-- The following is for Test Application ID: TEST2 -->
         <ePICApplication epicappid="TEST2">
           <epictrace>false</epictrace>
           <epictracelevel>512</epictracelevel>
           <AdapterRouting cn="epicadapterrouting">
              <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
              <ePICBodyCategory epicbodycategory="DEFAULT">
                 <ePICBodyType epicbodytype="DEFAULT">
      <!-- AdapterDaemon - Command to invoke. -->
                    <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                    </epiccommandclassname>
                    <epicreceivemode>MQ</epicreceivemode>
     <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
     <!-- -1 means never ending.  No entry defaults to 0. -->
     <!-- milliseconds.  Used when receiving messages. -->
                    <epicreceivetimeout>30000</epicreceivetimeout>
                    <epicreceivemqppqueue>TEST2AIQ</epicreceivemqppqueue>
                    <epicerrormqppqueue>TEST2AEQ</epicerrormqppqueue>
                    <epicreplymqppqueue>TEST2RPL</epicreplymqppqueue>
                 </ePICBodyType>
              </ePICBodyCategory>
           </AdapterRouting>
         </ePICApplication>
      </ePICApplications>
   </Epic>
```

付録E. サンプル・セットアップ・ファイル

以下に示すのは、いくつかの環境変数など、カーネルの初期構成値をいくつか定義する aqmsetup ファイルの例です。このファイルの追加情報は、64ページの『セットアップ・ファイル』を参照してください。aqmsetup ファイルは、インストールしたカーネルのルート・ディレクトリーの samples ディレクトリーにあります。

```
#
# aqmsetup 1.01 01/03/27
# Sample AQM Adapter runtime parameter configuration file entries.
#
# Copyright (c) 2001 International Business Machines. All Rights Reserved.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# Pound (#) signs are comments.
#
####################################################################
#
# Use Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services or configuration file. No entry defaults to
# true (use configuration file). To use the WSI directory service
# set the value to false.  Refer to the WSI documentation for
# specifics on using the directory service.
#AdapterDirectoryUseFileFlag=true
# When using Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services this additional entry is required. Refer to the
# WSI documentation for specifics on using the directory service.
#DirectoryServices=ChangeToDestDir/samples/DirectoryServices.properties
# Location of configuration file aqmconfig.xml when not using
# the Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services.
# No entry defaults to current directory.
#AQMConfig=ChangeToDestDir/samples
#
####################################################################
####################################################################
# XML DTD Catalogs and Directories - where to locate DTD's if not
# in the current directory.
# Format: XML_DTD_DIRECTORY_x=ddd where x is a numeric suffix to
# be incremented for each key and ddd is the directory.
# The numeric suffix's must start with 1 and be contiguous.
####################################################################
XML_DTD_DIRECTORY_1=ChangeToDestDir/runtimefiles/oag
```

```
#XML_DTD_DIRECTORY_2=ChangeToDestDir/runtimefiles
#
#####################################################################
# Java JNI Environment Variables for C Interface for increasing
# the amount of memory used.  This applies to when a C module
# is instantiating a JVM.  When a C Interface is being called
# from within JAVA the JVM is already established.
#####################################################################
# The stack memory is used for holding local function, function
# parameters, local variable references.
# Native stack is used for non-Java calls from within Java such
# as to C code.  Stack size in bytes to use.
# Default is 128 kilobytes on NT.
#AQM_JNI_NATIVESTACKSIZE=1048576
# Java stack is for Java method calls and local variables.
# Stack size in bytes to use.
# Default is 400 kilobytes on NT.
#AQM_JNI_JAVASTACKSIZE=4194304
# The heap memory is used for storing instantiated Java objects
# Minimum heap size in bytes to start with.
# Default is 1 megabyte on NT.
#AQM_JNI_MINHEAPSIZE=16777216
# Maximum heap size in bytes which can be used.
# Default is 16 megabytes on NT.
#AQM_JNI_MAXHEAPSIZE=268435426
#
#####################################################################
#  Designate end of configuration file
#####################################################################
*ENDCFG
```

特記事項

本書は米国 IBM 社が提供する製品およびサービスについて作成したものであり、米国以外の国においては本書で述べる製品、サービス、または機能を提供しない場合があります。日本で利用可能な製品、サービス、およびフィーチャーについては、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらのプログラムまたは製品に代えて、IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM 製以外の製品と組み合わせた場合、その操作の評価と検証については、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む。) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP事業所
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適確な表現や誤植など) は、本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。また、IBM 以外の製品に関するパフォーマンスの正確性、互換性、またはその他の要求は確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

商標

次のものは、IBM Corporation の米国およびその他の国における商標です。

AIX
AS/400
IBM
MQSeries
OS/400
RISC System/6000
RS/6000
WebSphere

Lotus および LotusScript は、Lotus Development Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Sun Microsystems, Inc. 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、Windows NT、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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

用語集

この用語集には、MQSeries Adapter Kernel の資料で使用される キー となる用語と、その意味を記載しています。

あるセクションにしか現れない特殊な概念や用語の中には、この用語集には含まれていないものがあります。ただしその場合でも、133ページの『索引』に記載しているものもあります。

この用語集は、MQSeries などの他の IBM 製品の用語は含んでいません。

アダプター (adapter). MQSeries Adapter Builder の所産。通常はユーザーが、アプリケーションとの間で送信される *1* メッセージ・タイプ ごとに固有のアダプターを作成する。したがって、アダプターそのものは MQSeries Adapter Offering のパーツではない。アダプターは共用ライブラリーにコンパイルされる C または Java™ ソース・コードから成ります。アダプターと MQSeries Adapter Kernel が一緒に実行されると、それらは MQSeries Adapter Offering のランタイム機能を実行する。アダプターの MQSeries Adapter Builder でのモデル化の方法にもよりますが、アダプターは次のようなさまざまな機能を持つことができます。制御フロー、データ・フロー、順次ナビゲーション、判別と反復を含む条件付き分岐、データ・タイプ化、データ・コンテキストの記憶、データ・エレメントの変換、トランザクション制御、論理演算、およびカスタム・コード化。作成済みのアダプターを再利用することができる。

130ページの『メッセージ・タイプ (message type)』、129ページの『ソース・アプリケーション (source application)』、および 129ページの『ターゲット・アプリケーション (target application)』を参照のこと。

アダプター・デーモン (adapter daemon). カーネルの一部である実行可能ソフトウェア。アダプター・デーモンが使用されるのは、送達モデルが push モデルの場合のみ。ワーカーのインスタンス化がその目的。アダプター・デーモンは一度始動すると、アクティブ状態を維持し続ける。ターゲット・アプリケーションごとに、1 つまたは複数のアダプター・デーモンが存在する。

アダプター・デーモンがターゲット・アプリケーションの役割を実行する場合がある。 E メール・メッセージの送信やファイルへのレコードの書き込みの際にターゲット・アダプターを使用するなどの、必須機能を実行する。

宛先論理 ID (destination logical identifier). ターゲット・アプリケーションを表す値。カーネルが他のメッセージ制御値と共に使用して、メッセージのルート指定とメッセージのマーシャルを行う。130ページの『メッセージ制御値 (message-control values)』を参照のこと。

アプリケーション固有インターフェース (application-specificinterface). 次のいずれかの目的で、MQSeries Adapter Offering の範囲外で開発されるインターフェース。

- ソース・アダプターがソース・アプリケーションからメッセージを獲得できるようにするため。
- ターゲット・アプリケーションがターゲット・アダプターからメッセージを獲得できるようにするため。

アプリケーション非依存フォーマット (application-neutral format). 129ページの『integration message』を参照のこと。

アプリケーション論理 ID (application logical identifier). アダプター (ソース・アダプターまたはターゲット・アダプター) が関連しているアプリケーションを表す ID。『ソース論理 ID (source logical identifier)』および 129ページの『ターゲット論理 ID (target logical identifier)』を参照のこと。

依存関係アプリケーション ID (dependency application identifier). ワーカーがサービスを行うアプリケーションの名前。ワーカーはアダプター・デーモンの名前を基に構成ファイルから、依存関係にあるアプリケーション ID を取得する。

エラー・キュー (error queue). MQSeries Adapter Offering の用語では、受信キューから取得したメッセージを処理できないときに使用されるメッセージ・キュー。

応答キュー (reply queue). 応答を受信するために使用されるメッセージ・キュー。カーネルの `sendRequestResponse` メソッドで使用される。

カーネル (kernel). MQSeries Adapter Kernel と同義。

カーネルのソース・サイド (source side of the kernel). カーネル機能の一部で、メッセージがソース・アダプターから受信されるときに始まり、そのメッセージがメッセージ・キューに書き込まれるときに終了する。

カーネルのターゲット・サイド (target side of the kernel). カーネル機能の一部で、メッセージがメッセージ・キューから取得されるときに始まり、そのメッセージがターゲット・アダプターに送信されるときに終了する。

構成ファイル (configuration file). `aqmconfig.xml` ファイル。カーネルの構成値の大部分が含まれている。詳細は 64ページの『構成ファイル』を参照のこと。

受信キュー (receive queue). MQSeries Adapter Offering の用語では、メッセージを受信する際に主な入力キューとして使用されるメッセージ・キュー。ターゲット・アプリケーションあたり複数の受信キューがあることもあるが、アプリケーション ID、本体カテゴリーおよび本体タイプの組み合わせごとの受信キューは 1 つだけである。

セットアップ・ファイル (setup file). カーネルのいくつかの初期設定値を含むファイル。このファイルのデフォルトの名前は `aqmsetup`。

送達モデル (delivery models). カーネルの、ターゲット・アプリケーションとのインターフェースには 2 つのモデルがある。2 つのモデルは次のとおり。

push ターゲット・アプリケーションへのメッセージの送達の開始と管理をカーネルが行う。このモデルでは通常、MQSeries Adapter Offering をサポートするためのターゲット・アプリケーションの変更は必要ない。

pull ターゲット・アプリケーションがメッセージの送達の管理を行う。このモデルでは、 MQSeries Adapter Offering をサポートするためにターゲット・アプリケーションを変更する必要がある。ターゲット・アプリケーションが、カーネルのターゲット・アプリケーションへとのインターフェースを管理しなければならない。

ソース論理 ID (source logical identifier). ソース・アプリケーションを表す値。カーネルが他のメッセージ制御値とともに使用して、メッセージのルート指定とメッセージのマーシャルを行う。130ページの『メッセージ制御値 (message-control values)』、127ページの『アプリケーション論理 ID (application logical identifier)』、および129ページの『ターゲット論理 ID (target logical identifier)』を参照のこと。

ソース・アダプター (source adapter). 次のタスクを実行するアダプター。

- ソース・アプリケーションの構造化されたデータを受信、または獲得する (通常はアダプターの範囲外で開発されるアプリケーション固有のインターフェースを使用して行う)。
- モデル化されたアダプターの仕様に従って構造化データを処理する。
- 構造化データを統合メッセージ・フォーマットに変換する。
- メッセージを 1 つまたは複数のターゲット・アダプターに送達し、そこからターゲット・アプリケーションに送達するために、カーネルを使用してメッセージをメッセージ・キューに書き込む。

メッセージ・タイプごとに 1 つのソース・アダプターが存在する。ソース・アプリケーションは通常、複数のメッセージ・タイプを送信することができるので、多くの場合、ソース・アプリケーションは複数のソース・アダプターでサポートされる。

127ページの『アダプター (adapter)』を参照のこと。

ソース・アプリケーション (source application). データをコンピューター・ネットワークを通して、通常は別のコンピューターにあるプログラム (いわゆるターゲット・アプリケーション) に送信するために必要なプログラム。

ターゲット論理 ID (target logical identifier). ターゲット・アダプターに関連したターゲット・アプリケーションを表す値。『ターゲット論理 ID (target logical identifier)』および 127ページの『アプリケーション論理 ID (application logical identifier)』を参照のこと。

ターゲット・アダプター (target adapter). 次のタスクを実行するアダプター。

- ソース・アダプターによって送信されたメッセージを、 (カーネルおよび MQSeries または他のメッセージング・ソフトウェアから) 受信する。
- モデル化されたアダプターの仕様に従って統合メッセージを処理する。
- 統合メッセージを、ターゲット・アプリケーションが受信できるアプリケーション固有のフォーマットのメッセージに変換する。
- アプリケーション固有のインターフェースを使って、メッセージをターゲット・アプリケーションに送信する。
- ターゲット・アプリケーションへのメッセージの送信が終了したときにワーカーに知らせて、ワーカーが肯定応答を送信できるようにする。

ターゲット・アプリケーションが統合メッセージを受信できる場合は、ターゲット・アダプターは必要ないことがある。

メッセージ・タイプごとに 1 つのターゲット・アダプターが存在する。ターゲット・アプリケーションは通常、複数のメッセージ・タイプを受信することができるので、多くの場合、ターゲット・アプリケーションは複数のターゲット・アダプターでサポートされる。127ページの『アダプター (adapter)』を参照のこと。

ターゲット・アプリケーション (target application). 通常は別のコンピューターにあるプログラム (いわゆるソース・アプリケーション) から、コンピューター・ネットワークを通してデータを受信するために必要なプログラム。

通信メッセージ (communications message). 通信トランスポート固有の情報に メッセージ・ホルダー・オブジェクトが加わったもので、使用する通信トランスポート固有のメッセージ・フォーマットに変換される。

通信モード (communications mode). メッセージのトランスポートおよびブローカー・サービスの実行のためにカーネルが使用するモード。

統合メッセージ (integration message). アプリケーション非依存のフォーマットのアプリケーション・データから成る、統合のためのメッセー

ジ。ソース・アダプターがソース・アプリケーションのフォーマットから XML に変換する XML 文書はその例。

トランザクション (transaction). オペレーションの集合であり、分割できない作業単位として実行されなければならない。トランザクションを形成するすべてのオペレーションが成功すると、そのトランザクションはコミット済みになる。つまり、すべてのオペレーションが完了したことになる。トランザクションを構成するオペレーションが 1 つでも失敗すると、そのトランザクションはロールバックされる。つまり、オペレーションは何も行われなかったことになる。

トレース (tracing). カーネルがトレース・メッセージを書き込むために使用するプロセスの集合。『トレース・メッセージ (trace messages)』を参照のこと。

トレース・クライアント (trace client). トレース・メッセージを書き込む、カーネルのコンポーネント。

トレース・メッセージ (trace messages). カーネルでのある時点でのメッセージの処理の状態を含むメッセージ。トレース・メッセージは、カーネルや作成したアダプターの問題の診断に利用することができる。

『トレース (tracing)』を参照のこと。

ネイティブ・アダプター (native adapter). メッセージ・ホルダー・オブジェクトの送信および受信に使用するソフトウェア。

本体カテゴリー (body category). メッセージに含まれるデータで、メッセージのアプリケーション・タイプ (たとえば OAG や RosettaNet) を表す。メッセージ制御値の設定に含まれる。『メッセージ制御値 (message-control values)』を参照のこと。

本体カテゴリーはメッセージ・タイプの指定にも利用できる。『メッセージ・タイプ (message type)』を参照のこと。

本体タイプ (body type). メッセージに含まれるデータで、メッセージの特定の目的 (たとえば「販売オーダーの追加」や「在庫の同期化」) を表す。メッセージ制御値の設定に含まれる。『メッセージ制御値 (message-control values)』を参照のこと。

本体タイプはメッセージ・タイプの指定にも利用できる。『メッセージ・タイプ (message type)』を参照のこと。

メッセージ (message). MQSeries Adapter Offering など MQSeries では、あるプログラムによって送信され、別のプログラムに与えられるデータの集合。

メッセージ制御値 (message-control values). カーネルがメッセージのマーシャルとルーティングを制御するために使用し、また各アダプターが一部機能の動作を制御するために使用する、メッセージ (本体とヘッダー) および構成ファイルの値の集合の総称。

メッセージ・タイプ (message type). 本体カテゴリーと本体タイプの固有の組み合わせで指定されるメッセージ。『本体カテゴリー (body category)』および 『本体タイプ (body type)』を参照のこと。

メッセージ・ホルダー・オブジェクト (message-holder object). カーネルが統合メッセージおよび他の制御データのカプセル化を行うために使用する、メタデータのコンテナー。

ログオン・クラス (logon class). ターゲット・アプリケーションごとに固有の Java クラスで、メッセージをターゲット・アプリケーションに送達する際に利用される。ログオン・クラスが必要なのは、メッセージの送達の前に、ターゲット・アダプターがターゲット・アプリケーションにログオンしなければ成らないときだけである。各ログオン・クラスはユーザーによって作成される。ワーカーがログオン・クラスをインスタンス化する。ログオン・クラスは構成ファイルを調べて、ターゲット・アダプターがターゲット・アプリケ

ーションとのアプリケーション固有インターフェースをサポートするために必要な値を検出する。それらの値は通常ログオン・パラメーターである。したがって、ターゲット・アダプターがそれらの値を使用できる。

何も行わないダミーのログオン・クラスがカーネルに用意されている。

論理メッセージ・サービス (logical message service). ネイティブ・アダプターが、通信トランスポートの移送用にメッセージを変換するために使用するコンポーネント。

論理 ID への応答 (respond-to logical identifier). 応答が要求されているときに、応答がそこに送信されることになるアプリケーションの論理 ID。デフォルトはメッセージの中のソース論理 ID。

ワーカー (worker). カーネルの一部であるソフトウェア。ワーカーが使用されるのは、送達モデルが push モデルの場合のみ。アダプター・デーモンがワーカーを生成し、始動する。各ワーカーはそれぞれ 1 つのネイティブ・アダプターを管理する。ワーカーは各メッセージを、それぞれの適切なターゲット・アダプターに送達する。

ワーカー・メッセージ bean (worker message bean). エンタープライズ bean の 1 つで、WebSphere Application Server をカーネルのターゲット・サイドで使用する場合にワーカーの機能を実行する。

aqmconfig.xml ファイル (aqmconfig.xml file). 128ページの『構成ファイル (configuration file)』を参照のこと。

aqmsetup ファイル (aqmsetup file). 128ページの『セットアップ・ファイル (setup file)』を参照のこと。

BOD. Business Object Document。組織内または組織間を流れる標準的なビジネス・プロセスを表現したもの。たとえば「購入オーダーの追加」、「製品の可用性の説明」、「販売オーダーの追加」などがある。BOD は、OAG によって XML を使って定義されている。132ページの『OAG』および 132ページの『XML』を参照のこと。

MQSeries Adapter Offering では、統合メッセージの中のメッセージ本体を定義するために BOD を使用することができる。

DTD. 文書タイプ定義 (Document Type Definition)。文書固有のタイプの形式定義を含む、通常は XML のファイル (または一緒に使用するいくつかのファイル)。DTD 内のエレメントに使用できる名前と (DTD 内でエレメントを起こせる)、エレメントが相互にどう整合するかを指定する。MQSeries Adapter Offering では、メッセージ本体の定義に DTD を使用できる。132ページの『XML』および 129ページの『integration message』を参照のこと。

Java サービス・アダプター (Java service adapter). Java アダプターの 1 タイプ。JMS Listener 環境で、アダプター・デーモン、ワーカー、およびターゲット・アダプターの 機能を実行する。

JMS Listener. WebSphere Business Integrator 製品で提供されるコンポーネント。MQSeries Adapter Kernel と WebSphere Application Server Advanced Edition 間の緊密な統合を可能にする。

MQSeries Adapter Builder. グラフィカル・ユーザー・インターフェース (GUI) を使用して、事実上どんなアプリケーションのアダプターでも作成できるソフトウェア。

MQSeries Adapter Kernel. API、C と Java で書かれたいくつかの実行可能プログラム、およびいくつかの構成ファイルのセット。カーネルはアダプターとともに動作し、またアダプターをサポートする。127ページの『アダプター (adapter)』を参照のこと。カーネルはアダプターを直接サポートすることの外に、メッセージングのルーティング、およびメッセージの組み立て、トレース、

MQSeries や他のメッセージング・ソフトウェアとのインターフェースなどの基盤サービスなど、関係する機能を実行する。

MQSeries Adapter Kernel ネイティブ・アダプター (MQSeries Adapter Kernel native adapter). ネイティブ・アダプターと同義。

MQSeries Adapter Offering. MQSeries Adapter Builder と MQSeries Adapter Kernel から成るアプリケーション統合製品のセット。

OAG. Open Applications Group。ビジネス・ソフトウェア・コンポーネントの相互運用の分野の多くの有力企業で形成される、非営利の業界コンソーシアム。OAG は Business Object Documents (BODs) を定義している。

pull モデル送達 (pull model of delivery). 128ページの『送達モデル (delivery models)』を参照のこと。

push モデル送達 (push model of delivery). 128ページの『送達モデル (delivery models)』を参照のこと。

WebSphere Application Server Advanced Edition. IBM のソフトウェア製品で、Sun Microsystems の Enterprise JavaBeans (EJB) 仕様を使用可能にする。WebSphere Application Server Advanced Edition には、エンタープライズ bean が実行できる、EJB サーバーが含まれる。エンタープライズ bean は、EJB クライアントが使用し共有するビジネス論理およびデータをカプセル化する。エンタープライズ bean には 2 つのタイプがある。セッション bean - 短期で、クライアント固有の、タスクおよびオブジェクトをカプセル化する。エンティティー bean - 永続的データをカプセル化する。ワーカー・メッセージ bean と呼ばれるタイプのセッション bean は、MQSeries Adapter Kernel のターゲット・サイドで使用できる。

XML. Extensible Markup Language。データ表記のための W3C の標準。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

［タ行］

［ナ行］

［ハ行］

［マ行］

［ラ行］

［ワ行］

［数字］

A

B

D

E

H

I

J

M

O

S

W

X

Printed in Japan

GC88-8887-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12