MQSeries® Adapter Kernel 多平台版

快速入门

版本 1 发行版 1

GB84-0445-01

MQSeries® Adapter Kernel 多平台版

快速入门

版本 1 发行版 1

GB84-0445-01

注: 在使用本资料及它所支持的产品之前，请阅读第111页的『声明』中的信息。

第 6 版（2001 年 4 月）

本版本适用于 MQSeries Adapter Kernel 多平台版，版本 1 发行版 1 修订版 1（产品号 5648-D75）及其所有后续发行版和修订版，除非在新版本中另有声明为止。

IBM 欢迎您提出宝贵意见。您可以发送电子邮件到 idrcf@hursley.ibm.com，提供对本资料的意见。

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 对于您所提供的任何信息，有权利以它认为适当的方式使用或分发，而不必对您负任何责任。

目录

图

表

欢迎使用 MQSeries Adapter Kernel 快速入门

本书描述了 MQSeries® Adapter Kernel，并说明了如何规划、安装和使用它。

要准备使用内核，请执行下列常规步骤:

1. 阅读第1页的『第1章 关于 MQSeries Adapter Offering』。
2. 准备安装。有关详细信息，请参阅第33页的『准备安装』。
3. 安装内核。有关详细信息，请参阅第34页的『安装内核』。
4. 验证安装。有关详细信息，请参阅第39页的『验证安装』。
5. 配置内核。有关详细信息，请参阅第50页的『配置内核』。
6. 根据需要，配置可选软件以与内核一起工作。有关详细信息，请参阅第77页的『配置 MQSeries 和 MQSeries Integrator』。
7. 使用 MQSeries Adapter Builder 构建适配器，然后测试并部署它们。有关详细信息，请参阅 MQSeries Adapter Builder 文档。
8. 启动内核。有关详细信息，请参阅第78页的『启动内核』。

要使用本信息，还需要了解先决产品和可选产品。请参阅第25页的『第2章 规划安装内核』。还可参阅第85页的『参考』。

使用本信息的对象

本信息供那些需要规划、安装或使用 MQSeries Adapter Kernel 的人员使用。

相关信息

有关附加信息，请参阅下列各项:

- `readme.txt` 文件。此文件可能包含本书完成后才获得的信息。安装之前，`readme.txt` 文件位于产品 CD-ROM 的根目录中。安装之后，`readme.txt` 文件位于 MQSeries Adapter Kernel 安装的根目录中。
- *Problem Determination Guide*，书号 GC34-5897，它描述了用来解决 MQSeries Adapter Kernel 中特定问题的工具，包括跟踪。*Problem Determination Guide* 可以在 “MQSeries Adapter Kernel 信息中心” 中获得，它与产品一起安装。
- “MQSeries Adapter Kernel 信息中心” 中提供的联机应用程序编程接口 (API) 文档。提供此信息仅帮助理解内核如何工作并帮助进行诊断。请参阅第83页的『第5章 使用 MQSeries Adapter Kernel API』。

- MQSeries Adapter Builder 信息，包括书籍和帮助系统。
- MQSeries 产品系列的 Web 站点：www.ibm.com/software/ts/mqseries/。
  通过该 Web 站点的链接，可以：
  - 获得 MQSeries 产品系列的最新信息，包括 MQSeries Adapter Offering。
  - 访问 HTML 和 PDF 格式的 MQSeries 书籍，可能包括本书的更新版本。
  - 下载 MQSeries SupportPac。

约定

MQSeries Adapter Kernel 文档使用下列印刷和输入约定。

表 1. 本书使用的约定

约定	含义
粗体字	表示命令名。当指图形用户界面 (GUI) 时，表示菜单、菜单项、标签和按钮。
等宽字体	表示在命令提示处必须输入的文本和必须按原样使用的值，如文件名、路径以及诸如函数、类和方法等程序设计语言的元素。等宽字体还表示屏幕文本和代码示例。
斜体字	表示必须提供的变量值（例如，为 *fileName* 提供的文件名称）。斜体字还表示强调和书名。
%	表示用于不需要 root 特权的命令的 UNIX 命令外壳提示符。
#	表示用于必需 root 特权的命令的 UNIX 命令外壳提示符。
C:\>	表示 Windows® 系统上的命令提示符。
>	用来描述菜单时，显示一系列菜单选择。例如，“单击**文件 > 新建**”表示“从**文件**菜单中，单击**新建**命令。”
输入命令	当指示“输入”或“发出”时，输入命令，然后按回车键。例如，指示“输入 **ls** 命令”表示在命令提示处输入 **ls**，然后按回车键。
[ ]	将可选项在语法说明中括起来。
{ }	将必须从中选择一项的列表在语法说明中括起来。
|	分隔在语法说明中用花括号 (**{ }**) 括起的选项列表中的各项。
...	语法说明中的省略号表示可以重复先前项一次或多次。示例中的省略号表示为简练而从示例中忽略的信息。

注: 术语 Epic 出现在内核软件和本书的一些值和名称中。就 MQSeries Adapter Offering 而言，此术语没有实际意义。

更改摘要

第六版（当前版本）包括自第五版以来的下列添加和更改:

- 更新了对运行时流的论述，反映出一些更改。请参阅第11页的『运行时流』。
- 有关将 MQSeries Adapter Kernel 与 WebSphere® Business Integrator 一起使用的信息。有关详细信息，请参阅第22页的『MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 一起使用』。
- 与各种不同适配器一起提供的国家语言支持的级别信息。有关详细信息，请参阅第24页的『国家语言支持』。
- 安装指令的介绍。请参阅第34页的『安装内核』。
- 安静方式安装的信息。有关详细信息，请参阅第43页的『使用安静安装』。
- 配置的概念性介绍，帮助您配置内核。有关详细信息，请参阅第50页的『配置概述』。
- 新的头值的有关信息。有关详细信息，请参阅第95页的『MQSeries Adapter Kernel 消息描述符头』。

第五版包括了自第四版以来的下列添加和更改:

- 在 Windows® 2000、OS/400®、HP-UX 和 Solaris 平台上使用内核的有关信息。对这些平台的支持在 MQSeries Adapter Kernel 版本 1.1 中是新增的。内核以前只可用于 Windows NT® 和 AIX®。
- 所有安装指令的更新，以反映 MQSeries Adapter Kernel 版本 1.1。
- 有关使用 `aqmconfig.xml` 文件配置 MQSeries Adapter Kernel 的信息。以前用 `aqmconfig.properties` 文件配置内核。有关详细信息，请参阅第55页的『配置文件』。
- 有关新的 MQ 和 JMS（Java 消息服务）通信方式的信息。有关详细信息，请参阅第87页的『附录A. 通信方式』。
- 有关跟踪信息从本文档移至新的 *Problem Determination Guide* 文档。有关详细信息，请参阅 *Problem Determination Guide*。

第1章 关于 MQSeries Adapter Offering

IBM MQSeries Adapter Kernel 是一组应用程序集成产品中的一部分，这些产品统称为 IBM MQSeries Adapter Offering。MQSeries Adapter Offering 使用 MQSeries 消息传递或其它消息传递服务使您能够降低管理点对点集成的商业处理的风险、复杂程度和成本。

在*点对点*集成中，每个应用程序分别与其它各个应用程序通信。每个接口都不同，就有许多不同的接口。通常某个应用程序中发生更改，就需要对许多接口进行更改。随着应用程序数量的增加，点对点集成的成本就迅速增长。通常，每集成一个新的应用程序所需做的工作都比集成上一个应用程序要多。

有了 MQSeries Adapter Offering，就可以从使用点对点集成迁移到使用*一对任意*集成。一对任意集成有许多优点，它们包括:

- 所有应用程序可以使用一个公共接口。
- 一个*源应用程序*中格式为*消息*的数据可以*路由*至一个或多个*目标应用程序*。
- 对某个应用程序做更改通常只影响一个接口。
- 使用与应用程序无关的公共接口（例如，诸如可扩展标记语言 (XML) 等行业标准）甚至可以更好地符合成本效益。可以用较少的投入支持更多的应用程序。
- 随着应用程序数量的增加，“一对任意”集成会更好地符合成本效益。每添加一个新应用程序通常不需要对其它所有应用程序的接口做显著的更改。
- 可以自动化集成工作，并且集成工作可以基于模板。

可以根本不更改应用程序或商业处理，就部署 MQSeries Adapter Offering。通常，在 MQSeries Adapter Offering 中执行所有集成工作，从而减少编写定制代码的需求。

在 MQSeries Adapter Offering 中，由*适配器*提供到或来自应用程序的接口。所有应用程序都需要至少一个适配器提供应用程序环境和消息传递环境之间的接口。每个适配器都特定于一个应用程序和一种消息类型。

可以选择使用 MQSeries Integrator 来部署 MQSeries Adapter Kernel，以执行代理和消息变换。可以通过 IBM 和其它公司提供的服务软件束补充 MQSeries Adapter Offering。

使用适配器的示例包括:

- 添加销售订单。

- 同步化客户记录。
- 同步化库存记录。
- 同步化某一项。
- 同步化销售订单。

构建时和运行时

MQSeries Adapter Offering 由两个主要组件组成: Adapter Builder （也称为构建器）和 Adapter Kernel（也称为内核）。本节描述了这两个组件，以及 Adapter Offering 构建并运行的适配器。

适配器

提供到或来自应用程序接口的软件。适配器是使用 MQSeries Adapter Builder 构建的。通常，构建的每个适配器都特定于从应用程序发出或发送到应用程序的一种消息类型。适配器本身并不是 MQSeries Adapter Offering 的一部分。

适配器由编译成共享程序库的 C 或 Java™ 源代码组成。当适配器和 MQSeries Adapter Kernel 一起运行时，它们执行 MQSeries Adapter Offering 的运行时功能。

根据适配器在 MQSeries Adapter Builder 中建模的方式，它可以包含非常广泛的功能，如控制流、数据流、顺序导航、包含判定和迭代的条件转移、数据输入、数据上下文存储、变换数据元素、事务性控制、逻辑运算和定制代码等。

适配器可以再使用。

有两种主要的适配器类型:

- 源适配器，用于发送数据的应用程序。
- 目标适配器，用于接收数据的应用程序。

将一种消息从一个应用程序发送到第二个应用程序通常需要一个源适配器和一个目标适配器。如果第二个应用程序必须将一种消息发送到第一个应用程序，则需要另一个源适配器和另一个目标适配器。因此，要将一种消息从第一个应用程序发送到第二个应用程序，然后将另一种消息从第二个应用程序发送回第一个应用程序，通常要部署四个适配器。

对每个消息类型需要一个独立的适配器。

第三种类型适配器（Java 服务会话 bean 适配器）是在内核的目标方上使用 IBM WebSphere Application Server 和 enterprise bean 时使用的。WebSphere Application Server 实现了 Sun Microsystems Enterprise JavaBeans

(EJB) 的规范，这就允许使用 Java 服务会话 bean 适配器和其它 enterprise bean。有关详细信息，请参阅第22页的『MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 一起使用』和 MQSeries Adapter Builder 文档。

MQSeries Adapter Builder

一个图形用户界面 (GUI)，使您能够构建用于任何实际应用程序的适配器。用户界面与 MQSeries Integrator 的用户界面类似。有关详细信息，请参阅 MQSeries Adapter Builder 信息中心。

MQSeries Adapter Kernel

一组应用程序编程接口 (API)，一些用 C 和 Java 语言编写的可执行程序和一些配置文件。内核支持适配器的部署和执行。除了直接支持适配器外，内核还执行相关的功能，包括简单的路由消息。它还提供基础结构服务，例如消息构造、事务性控制、跟踪和与 MQSeries 或其它消息传递软件交互。

内核安装在每个运行源适配器或目标适配器的计算机上。

有了 MQSeries Adapter Offering，商业处理和每个应用程序可以保持与特定的中间件、消息细节和其它应用程序隔离。用于消息传递的公共接口允许添加新应用程序，而不必更改现有应用程序或商业处理。

MQSeries Adapter Kernel 可以部署为两层。一层是运行时的源方；另一层是运行时的目标方。两层部署提供高效操作和较低的管理系统开销。用于路由和传递的第三层不需要驻留在运行时的两方之间。但是，可以任选地添加 MQSeries Integrator 来执行代理，例如复杂的路由、数据变换和数据调和。

除了特别指出外，本文档的其余部分仅指 MQSeries Adapter Kernel。有关 MQSeries Adapter Builder 的详细信息，请参阅产品的 “信息中心”。

关于内核

简单来说，运行时（即内核和构建的适配器）有下列用途:

1. 将数据从一个源应用程序传送到一个目标应用程序。
2. 将源应用程序的数据转换为通常与应用程序无关的格式消息，然后使用 MQSeries 或其它消息传递软件通过内核路由消息。
3. 将消息路由至目标应用程序。
4. 确定如何取得数据给目标应用程序。
5. 通过适配器将数据从用内核路由的消息格式转换成目标应用程序的格式。

本节中，在较高层次论述了内核的功能。第11页的『运行时流』中更详细论述了功能。

内核有两方:

- *源方*，它在从源应用程序接收消息时开始，在消息放入消息队列时结束。
- *目标方*，它在从消息队列中检索到消息时开始，在消息发送至目标时结束。

通常，每一方位于不同的计算机中，但它们也可同在一台计算机上。

请参阅第5页的图1。它描述了以下序列。

内核的源方

1. 在内核的源方上，通过使用*应用程序特定接口*，源应用程序将数据以它的*源应用程序格式*发送到在 MQSeries Adapter Builder 中构建的源适配器中。每个消息类型需要一个不同的源适配器，例如，用于“添加销售订单”或用于“同步化客户记录”。

   必须在 MQSeries Adapter Offering 之外开发应用程序特定接口。应用程序特定接口的确切特性取决于源应用程序或目标应用程序的特性。示例包括 API 调用和用户出口、文件读写、数据库触发器和消息队列。

   注意: 源适配器在源应用程序的进程中运行。任何包含源适配器的守护程序或服务器都需要运行以便源适配器正常工作。

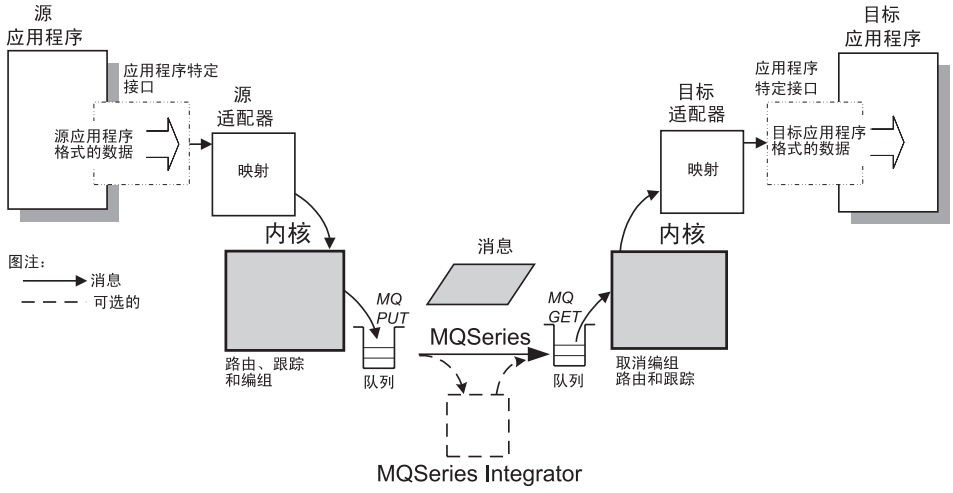
2. 根据源适配器是如何构建的，它执行相应的功能。典型功能是数据元素的变换，即，将元素从源应用程序格式映射为主体数据的*集成消息传递数据格式*。主体数据和表示控制值的附加元数据放置在内核的*消息存放器对象*中。

3. 当源适配器通过使用*本机适配器* 将消息存放器对象传送至内核时，内核使用消息存放器对象中的控制值（*消息控制值*）来控制将消息存放器对象编组为通信消息格式，并控制通信消息的路由。

   如果消息不包含某些消息控制值，则内核可以使用缺省值或从配置文件中获得的消息控制值。有关消息控制值的定义，请参阅第13页的『消息控制值』。

4. 内核执行的功能包括*消息编组*、简单*路由*和可选*跟踪*。请参阅第10页的『消息和消息格式』、第11页的『路由和传递』和第22页的『跟踪』。

图 1. MQSeries Adapter Offering 的概述.



从源方传递到内核的目标方

5. 内核通过使用它的本机适配器将消息放在适当的消息队列中。

   源方上使用的发送方法有两个:

   - `sendMsg`，发送消息并立即返回。`sendMsg` 方法还可以与 `begin`、 `commit` 和 `rollback` 方法一起使用，*事务性地*发送消息；即，当（且仅当）其它操作成功完成时，才发送消息。有关详细信息，请参阅第22页的『事务性能力』。
   - `sendRequestResponse`，发送消息并等待响应。`sendRequestResponse` 方法不能事务性地发出。

     注意: 第三个方法 `sendResponse` 是在发送方请求响应时在内核的目标方上使用的。

   MQSeries 或其它消息传递软件传送消息。请参阅第6页的『MQSeries 或其它消息传递软件的角色』。注意: 必须已配置了消息传递软件使它支持 MQSeries Adapter Offering。

   或者，如果 MQSeries Integrator 已在内核中配置为目的地，则 MQSeries Integrator 可以执行代理功能。请参阅第7页的『MQSeries Integrator 的角色』。如果已在 MQSeries Integrator 的规则或消息流中配置了最终目的地 -- 消息队列，则 MQSeries Integrator 将消息发送至消息队列。

   消息到达适当的消息队列。

内核的目标方

6. 在内核的目标方上，有两个可能的*传递模型*用于运行时和目标应用程序之间的接口。

- 最常用的模型是*推送*，在这个模型中，内核负责启动并管理将消息传递到目标应用程序。推送模型通常不需要更改目标应用程序以支持 MQSeries Adapter Offering。
- 在*拉入*模型中，目标应用程序负责管理消息的接收。拉入模型需要更改目标应用程序以支持 MQSeries Adapter Offering。目标应用程序必须管理内核至目标应用程序的接口。

在推送模型中，请注意：在目标方上，内核的进程必须在获取消息和传递消息前由用户启动。请参阅第78页的『启动内核』。

在推送模型中，内核从消息队列中取出消息。如果启用跟踪，则执行跟踪。通过选择适当的目标适配器，它继续路由消息。一般说来，每个消息类型需要一个不同的目标适配器。

7. 内核将消息传递给适当的目标适配器。目标适配器执行构建在其中的功能。典型功能是将元素从集成消息传递格式映射成*目标应用程序格式*的元素。

   目标适配器可以由 MQSeries Adapter Kernel 适配器守护程序或 WebSphere Application Server 主管。有关后者如何管理的论述，请参阅第22页的『MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 一起使用』。
8. 通过使用在 MQSeries Adapter Offering 之外开发的应用程序特定接口，目标适配器将数据以目标应用程序格式发送到目标应用程序中。
9. 当目标适配器传递了它的消息后，消息就从消息队列中提交。这将消息从队列中除去。
10. 如果源适配器已设置了一个消息控制值来请求确认，则内核会通过使用 `sendResponse` 方法，将消息传递的确认或目标输出传递至源适配器。
11. 如果发生错误，内核会将原始消息放入错误队列。如果内核不能将原始消息放入错误队列，则不会发生提交。

MQSeries 或其它消息传递软件的角色

MQSeries Adapter Offering 的通信消息是通过消息队列传送的。消息队列由诸如 MQSeries 或 “Java 消息服务 (JMS)” 等消息传递软件提供。MQSeries Adapter Offering 传送的消息使用下列队列类型：

- *接收队列*，MQSeries Adapter Offering 的术语。它们作为主输入队列用来接收消息。每个目标应用程序可以有多个接收队列。
- *错误队列*，MQSeries Adapter Offering 的术语。当无法处理从接收队列中获得的消息时，使用它们。
- 作为一个选项的*应答队列*。它们与 `sendRequestResponse` 方法一起使用。

MQSeries Adapter Offering 使用某些 MQSeries 功能，如下列消息类型：

- 数据报，由 sendMsg 方法使用。
- 请求，由 sendRequestResponse 方法使用。
- 应答，由 sendRequestResponse 方法和 sendResponse 方法使用。

可以选择使用 MQSeries 充当应用程序特定接口。

有关 MQSeries 和 MQSeries Adapter Offering 的已确认配置列表，请参阅第93页的『附录B. 已确认的配置』。有关 MQSeries 和其它软件的受支持版本的列表，请参阅第26页的『软件』。

MQSeries Integrator 的角色

可以选择将 MQSeries Integrator 与 MQSeries Adapter Kernel 一起部署。它用来满足关于代理的几个可能需求:

- 复杂路由，即根据消息头或消息主体的内容进行路由。路由可以随消息主体的改变而动态地改变。有关复杂路由和简单路由的信息，请参阅第11页的『路由和传递』。
- 数据变换，即对不同文档类型的更改。
- 数据调和，即更改消息主体的内容。例如，如果源应用程序在某个字段中提供值 each，但目标应用程序期望该字段的值是 ea，则数据调和用期望值替换所提供的值。

可以使用 MQSeries Integrator 来执行所在位置中的大多数路由；还可以少量使用 MQSeries Adapter Kernel 的路由功能。

有关已确认的 MQSeries Integrator 和 MQSeries Adapter Offering 的配置列表，请参阅第93页的『附录B. 已确认的配置』。有关 MQSeries Integrator 和其它软件的受支持版本的列表，请参阅第26页的『软件』。

内核如何工作

本节将论述下列内容:

- 第8页的『内核运行时的组件』
- 第10页的『消息和消息格式』
- 第11页的『路由和传递』
- 第11页的『运行时流』

内核运行时的组件

当构建的适配器、开发的定制代码和 MQSeries Adapter Kernel 一起运行时，它们提供 MQSeries Adapter Offering 的功能。

内核运行时的主要组件有:

源适配器

为特定应用程序构建的软件（通常使用 MQSeries Adapter Builder 来构建），用于将数据从该应用程序格式转换到集成消息传递格式（主体数据）。通常，源适配器与源应用程序在同一机器上运行，它可以在应用程序的进程中运行，也可以作为单独进程运行。源数据的示例包括文件、C 结构和 Java 对象。集成消息传递格式的示例是 XML，通常遵循诸如 OAG 或 RosettaNet 等行业标准。

消息存放器

内核使用的元数据容器，用来封装内核使用的集成消息和其它控制数据。元数据示例包括源和目标应用程序的应用程序标识（逻辑标识）、消息类别（例如，OAG）、消息类型（例如，“采购订单”）以及发送或接收的通信消息（主体数据）。

本机适配器

用于发送和接收消息存放器对象的软件。发送消息时，本机适配器提供简单数据路由及支持一个或多个通信传送机制的功能。简单数据路由是基于消息存放器对象中的元数据，如消息类别和消息类型。消息可以异步发送，也可以同步发送。如果基本的通信传送机制支持事务性消息传递，那么可以在单阶段事务性控制下发送消息。事务性支持受到所用传送机制能力的限制。消息存放器对象被编组成传送机制使用的通信消息格式。当接收到通信消息时，本机适配器将消息取消编组为消息存放器对象。

适配器守护程序

实例化适配器工作程序的进程。启动它之后，适配器守护程序保持活动。对于每个目标应用程序，每个应用程序接收队列可以有一个适配器守护程序。

适配器工作程序

将每条消息传递至适当的目标适配器的进程。每个工作程序管理一个本机适配器。适配器守护程序创建并启动工作程序。

拥有多个工作程序可启用到目标适配器的*多线程消息传递*。每个工作程序和它的本机适配器可以处理一个线程。如果只有一个工作程序，则消息传递到目标适配器再到目标应用程序是单线程的。

除了管理本机适配器之外，工作程序还执行下列任务:

- 如果启用跟踪，则它实例化跟踪客户机。
- 它实例化适用于每个目标应用程序的登录类。
- 它根据消息的主体类型和主体类别来选择目标适配器。
- 它将消息发送至选定的目标适配器。
- 如果它不能执行提交，则会执行回滚，为所有在该适配器守护程序下的其它工作程序设置一个标志，并关闭它本身及其本机适配器。这表示消息有问题。关闭所有工作程序防止其它工作程序重新处理有相同结果的同一个问题消息。
- 当它识别由另一个要关闭的工作程序设置的标志时，它关闭其本身以及它的本机适配器。

目标适配器

为特定应用程序构建的软件（通常使用 MQSeries Adapter Builder 来构建），用于将数据从集成消息传递格式（主体数据）转换到目标应用程序所需的数据类型。目标适配器调用目标应用程序上必需的 API 来传递消息。目标适配器与应用程序或应用程序客户机运行在同一台机器上。

Java 服务会话 bean 适配器

一种 Java 语言的 EJB 适配器，在 EJB 服务器中（例如 WebSphere Application Server）中托管。

配置组件

用于将逻辑标识解析成诸如队列名等对象的数据。配置数据可以在文件或 WebSphere Business Integrator 产品的 LDAP 结构中指定。数据控制内核配置的下列方面:

- 编组和路由消息
- 验证安装
- 通信方式
- 跟踪

有关配置文件的完整描述，请参阅第55页的『配置文件』。有关配置该产品与内核一起工作的信息，请参阅 WebSphere Business Integrator 文档。

跟踪组件

写跟踪消息的软件。大多数内核组件使用跟踪组件。有关跟踪的概述，请参阅第22页的『跟踪』；有关跟踪的详细信息，请参阅 *Problem Determination Guide*。

消息和消息格式

在 MQSeries 和 MQSeries Adapter Offering 中，*消息*是一个数据集合，它由一个程序发送并传给另一个程序。任何时刻的消息格式取决于在该特定时刻消息在消息流中的位置。MQSeries Adapter Kernel 指定三种消息，如下所示:

- *集成消息* -- 一条由数据构成的消息，数据来自源应用程序并转换成另一种格式（例如 XML）以便发送至目标应用程序。集成消息作为消息的主体数据插入消息存放器对象。XML 是一种数据表示法的标准。当格式是 XML 时，格式由文*档类型定义* (DTD) 定义。 DTD 是一个或多个文件，它们包含文档（在这种情况下，是消息主体）的形式定义。尽管强烈建议对于消息主体的格式使用与应用程序无关的格式，但也不是必须使用。消息主体的格式可以是专用的或特殊的，但是不建议使用这种格式。

  商业对象文档 (BOD) 可以由 MQSeries Adapter Offering 用来在它的集成消息中定义消息主体。BOD 是标准商业处理的表示法，该处理在组织内或组织之间流动。示例有: 『添加采购订单』、『显示产品可用性』和『添加销售订单』。BOD 由“开放式应用程序组”(OAG) 定义在 XML 中。建议使用 BOD，但非必需的。

- *消息存放器对象* -- 包含集成消息和附加头元数据的对象，这些头元数据表示特定于 MQSeries Adapter Kernel 的控制值。源适配器创建消息存放器对象，设置适当的控制信息，并在有要发送的集成消息时设置主体数据。目标适配器接收消息存放器对象，取出主体数据并将主体数据转换成特定于目标应用程序的数据。通过使用 MQSeries Adapter Builder 来创建源适配器和目标适配器。

- *通信消息* -- 任何通信传送工具特定的信息加上消息存放器对象，转换成特定于所使用的通信传送工具的消息传递格式。一些通信传送工具支持多个消息传递格式。通常，通信传送工具将与通信消息组合的内核头元数据值当作应用程序数据。有关详细信息，请参阅第87页的『附录A. 通信方式』。用于 MQSeries 传送的示例由 MQSeries 特定的消息头和经过编组的消息存放器对象组成。特定的 MQSeries 格式包括:
  - MQSeries 添加的 MQSeries 消息头
  - 如果使用 MQSeries Integrator，则存在版本特定的消息头:
    - 如果使用 MQSeries Integrator 版本 1.1，则为 MQSeries Integrator 版本 1 的消息头
    - 如果使用 MQSeries Integrator 版本 2，则为 MQSeries Integrator 版本 2 的消息头
  - 表示控制值的内核特定的头元数据
  - 集成消息（主体数据）

有关在 MQSeries Adapter Offering 的消息头中使用的相关字段及其描述的列表，请参阅第95页的『附录C. 消息头』。

路由和传递

内核从源适配器路由每条消息，并将它传递至适当的目标适配器。路由的执行有两个阶段:

1. 内核的源方将消息放到适当的消息队列中。
2. 内核的目标方从消息队列中取得消息并调用适当的目标适配器。

路由是由以下几个因素确定的:

- 消息队列。在最基本级别上，必须配置消息队列以支持 MQSeries Adapter Offering 的路由。
- 消息中的消息控制值。它们包括源逻辑标识、目的地逻辑标识、响应逻辑标识、主体类别、主体类型、事务标识、消息标识、请求的确认和时间戳记。有关详细信息，请参阅第13页的『消息控制值』。消息中的目的地逻辑标识可以覆盖内核的配置文件。路由可以随每个消息头中这些消息控制值的改变而动态地改变。但是，消息主体数据（集成消息）的内容不能确定路由。
- 内核配置文件中的消息控制值。文件可以指定目的地逻辑标识、队列名和相关的目标适配器。通过编辑这个文件，可确定并修改配置。有关附加信息，请参阅第55页的『配置文件』。
- 还可以选择使用 MQSeries Integrator 来代理消息，包括复杂的路由。路由可以随消息主体的改变而动态地改变。请参阅第7页的『MQSeries Integrator 的角色』。相反地，MQSeries Adapter Offering 本身只可以执行简单路由。简单路由是基于消息中的消息控制值和配置文件中相关消息控制值的组合。它不是基于消息主体的内容。

可以请求内核确认消息的传递。这是一个应用程序级的确认。

运行时流

本节详细论述了运行时流--内核如何在一个典型的产品环境中发送、路由、跟踪和传递消息。有关运行时流的图表，请参阅第12页的图2。

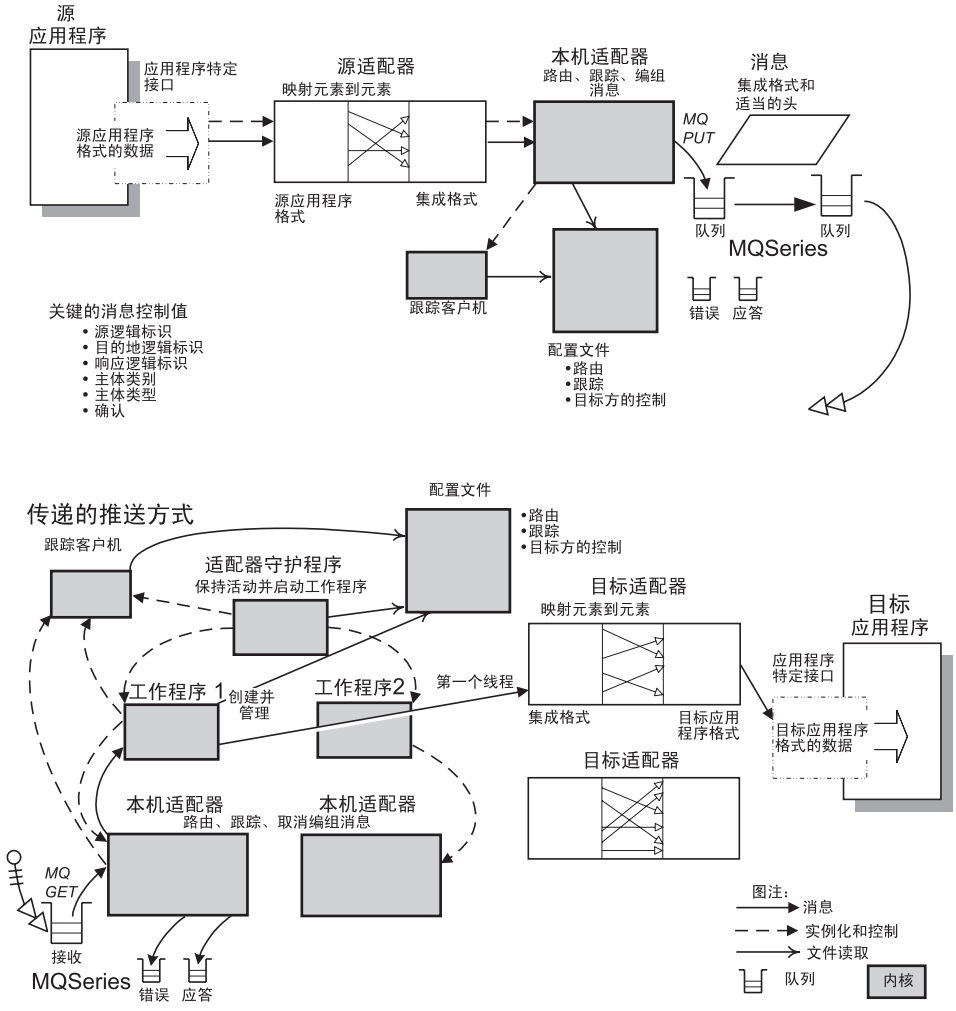


图 2. 编组、发送、路由和跟踪消息 -- 概述.

内核的源方

这部分讨论了在内核源方上的运行时流；即，数据是如何通过源适配器从源应用程序移动到通信传送工具中的。第16页的『内核的目标方』讨论了数据如何从通信传送工具中移动到目标。

1. 通过使用应用程序特定接口，源适配器从源应用程序中获取消息。通常，由应用程序特定接口来调用源适配器。

2. 源适配器执行在 MQSeries Adapter Builder 中对其构建的功能。通常，它将源应用程序格式的数据变换成与应用程序无关的集成格式（对于消息主体）。

   作为源适配器功能的一部分，源适配器将几个消息控制值放到 MQSeries Adapter Kernel 头中；它使用这些值来封装消息。前五个消息控制值决定编组和路由，最后一个值决定确认。

   消息控制值

   源逻辑标识

   源应用程序的逻辑标识。在消息中，它总是必需的。

   目的地逻辑标识

   目标应用程序的逻辑标识。如果它没有在消息中出现，则使用配置文件中的缺省值来替代。在配置文件中，可以使用多个目的地逻辑标识来代替消息中没有包含的值。

   响应逻辑标识

   应用程序的逻辑标识，该应用程序是在请求应答的情况下向其发送应答的应用程序。它的缺省值是消息中的源逻辑标识。

   主体类别

   表示消息的应用程序类型 -- 例如，OAG 或 RosettaNet。在消息中，它总是必需的。

   主体类型

   表示消息的特定目的 -- 例如，“添加销售订单”或“同步化库存”。在消息中，它总是必需的。

   请求确认

   确定源应用程序是否请求应答。应答可以是下列格式之一:

   - 来自目标应用程序的应答数据
   - 一条“OAG 确认 BOD”消息

     注: “确认 BOD”消息由 OAG 预先定义。它的主体类别是 `OAG`，主体类型是 `CONFIRM_BOD_003`。它还可以包含数据。

   这种应答是一种应用程序级的确认。

   当内核使用 `sendRequestResponse` 方法发送消息时，仅使用 `sendRequestResponse` 方法接收到的第一个应答。如果原始消息发送至多个目的地，并请求应答（不建议这样做），则只有第一个应答发送回源应用程序。

   缺省值是不确认；因此，不请求或发送应答。

3. 源适配器初始化本机适配器并向它传送:

- 运行源适配器的应用程序的逻辑标识。
- 消息存放器对象，它包含消息控制值和消息主体数据。

4. 本机适配器在配置文件中查找，以确定是否对源逻辑标识启用跟踪。如果启用跟踪，则本机适配器实例化跟踪客户机。
5. 跟踪客户机在配置文件中查找，以确定要使用的跟踪级别并获取其它值。跟踪客户机使用跟踪级别来过滤跟踪消息。有关跟踪的概述，请参阅第22页的『跟踪』；有关跟踪的详细信息，请参阅 *Problem Determination Guide*。
6. 本机适配器在消息存放器对象中查找目的地逻辑标识。如果有，则使用它。
   - 如果没有目的地逻辑标识，则本机适配器根据源逻辑标识、主体类别和主体类型，在配置文件中查找缺省的目的地逻辑标识。
   - 根据源逻辑标识，本机适配器在配置文件中按下列次序执行多阶段查找主体类别和主体类型:
     a. 特定主体类别和主体类型值。
     b. 特定主体类别值和缺省主体类型值。
     c. 缺省主体类别值和特定主体类型值。
     d. 缺省主体类别和主体类型值。

   注: 每次内核在配置文件中查找值时，都使用这个多阶段查找方式。
7. 对于上一步中确定的每个目的地逻辑标识，本机适配器根据目的地逻辑标识、主体类别和主体类型，查找*通信方式*。支持下列通信方式:

   MQPP 内核使用 MQSeries 基本服务传送消息。

   MQRFH1 内核使用 MQSeries 传送消息，使用 MQSeries Integrator 版本 1.1 代理消息。

   MQRFH2 内核使用 MQSeries 传送消息，使用 MQSeries Integrator 版本 2 代理消息。

   MQBD 内核使用 MQSeries 基本服务传送消息，但仅发送和接收主体数据。

   MQ 内核使用 MQSeries 传送消息。

   JMS 内核使用 “Java 消息服务 (JMS)” 传送消息。

   FILE 内核将消息放入文件并从文件中取出消息。这个方式仅为诊断目的提供。

   在每个通信方式中，消息结构各不相同。请参阅第10页的『消息和消息格式』。有关通信方式的详细信息，请参阅第87页的『附录A. 通信方式』。

注: 如果使用 MQSeries Integrator，则 MQSeries Integrator 发送消息的最终目的地必须使用与 MQSeries Integrator 接收消息使用的相同的通信方式。

8. 根据通信方式，本机适配器实例化其自身的一个子类来处理消息。该子类称为*逻辑消息服务*。每个通信方式有一个不同的逻辑消息服务子类。

   本机适配器将目的地逻辑标识、主体类别和主体类型传送至逻辑消息服务。

9. 逻辑消息服务子类找到发送消息所需的参数。例如，如果通信方式是 MQPP，则参数包括格式以及接收队列、应答队列和错误队列名。根据传送给它的目的地逻辑标识、主体类别和主体类型，逻辑消息服务在配置文件中执行多阶段查找:

   a. 特定主体类别和主体类型值。

   b. 特定主体类别值和缺省主体类型值。

   c. 缺省主体类别值和特定主体类型值。

   d. 缺省主体类别和主体类型值。

此时，逻辑消息服务具有路由和编组消息所需的全部信息。

10. 逻辑消息服务执行下列任务:

    - 根据通信方式和格式适当地编组消息。如果没有另外指定格式，则每个通信方式使用一个缺省格式。例如，如果通信方式是 MQRFH2，则逻辑消息服务为了使用 MQSeries 进行传送和使用 MQSeries Integrator 版本 2 进行代理而创建适当的头并构造消息。
    - 发送消息。例如，如果通信方式是 MQRFH2，则它将消息放在适当的 MQSeries 消息队列上。

11. 有两种方法可以用来发送消息:

    - 如果本机适配器使用了 `sendMsg` 方法发送消息，那么本机适配器不等待响应。
    - 如果本机适配器使用 `sendRequestResponse` 方法发送消息，则逻辑消息服务等待应答。通过使用逻辑消息服务，本机适配器会监控应答队列一段时间，这段时间是在配置文件中设置的*接收超时周期*。

      接收超时周期基于源应用程序标识、主体类别和主体类型。

      – 如果接收到确认，则本机适配器返回消息。

      – 如果在接收超时周期内没有接收到确认，则本机适配器不返回消息。

12. MQSeries 或其它消息传递软件根据它的配置情况来传送消息。或者，可以使用 MQSeries Integrator 执行代理服务。请参阅第7页的『MQSeries Integrator 的角色』。

13. 当源适配器不再需要本机适配器时，会关闭本机适配器以释放资源。

内核的目标方

这部分讨论了如何在目标方上使用独立 MQSeries Adapter Kernel 接收和处理消息，并且提供了将内核与 WebSphere Application Server 一起使用的高级描述。请参阅第22页的『MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 一起使用』获得有关将内核与 JMS、WebSphere Business Integrator 的 JMS Listener 组件，以及内核目标方上的 WebSphere Application Server 一起使用的论述。这部分描述了传递的推送模型，在此模型中，内核负责启动并管理将消息传递到目标应用程序。有关模型的简要描述，请参阅第116页的『传递模型』。

适配器工作程序概述

这部分描述了 MQSeries Adapter Kernel 适配器工作程序的结构和行为。MQSeries Adapter Kernel 体系结构中的一个假设是目标应用程序不主动参与与其它应用程序之间的数据流集成；即应用程序不主动轮询消息进行处理。在这种情况下，消息数据需要被主动推送到目标应用程序。适配器工作程序将消息数据推送到应用程序，或通过选择和调用一定范围的服务接口类型来将消息数据推送到其它服务。

MQSeries Adapter Kernel 可以主管在独立守护程序（适配器守护程序）或 Enterprise JavaBean 应用程序服务器（当前是 IBM WebSphere Application Server 高级版）上运行的适配器工作程序。根据使用的目标环境类型，消息通过各种不同途径到达适配器。如果使用独立适配器守护程序，那么它可以主管一个或多个使用本机适配器接收消息的独立适配器工作程序。如果使用 EJB 服务器，那么 JMS Listener 组件接收消息并将它们传送到一个工作程序消息 bean（有时也称为消息驱动 bean 适配器工作程序）。

不论使用何种目标环境，适配器工作程序在接收到消息后，都会将消息转发到适当的目标适配器。然后目标适配器执行必需的操作将消息传递到目标应用程序。目标适配器是为了与特定目标应用程序一起工作而创建的。适配器守护程序、应用程序服务器、独立适配器工作程序和工作程序消息 bean 不特定于任何给定的源或目标应用程序。

适配器工作程序处理两种类型的目标适配器接口：Enterprise Access Builder (EAB) 命令适配器和 EJB 服务会话 bean。每个适配器类型都包含一个处理程序，它设置适当的环境、访问适配器必需的任何附加配置信息，并执行适配器操作必需的一些其它低级任务。使用的处理程序取决于配置文件中列出的适配器类型。这两种类型的处理程序执行下列附加任务:

- EAB 处理程序获取一个登录类（它用于提供到目标适配器的连接信息），并初始化 IBM Common Connector Framework (CCF) 运行时。目标应用程序的逻辑标识被传递给登录类，登录类使用这个标识来获得应用程序特定的登录信息。

- EJB 处理程序获取一个 ″Java Naming and Directory Interface™ (JNDI)″ 连接，然后获取服务会话 bean 的远程接口和访问服务会话 bean 所必需的其它信息。

独立适配器守护程序内的适配器工作程序的基本处理流程如下:

1. 在启动时，适配器守护程序根据内核配置文件提供的信息实例化一个或多个独立适配器工作程序。应用程序的逻辑标识以及可选主体类别和主体类型值被传递给适配器守护程序。主体类别和主体类型值用来获取附加配置值。
2. 每个独立适配器工作程序执行下列任务:
   a. 适配器工作程序实例化本机适配器并开始接收消息。每个消息都是在事务性控制下接收的，并且作为消息存放器对象返回给适配器工作程序。
   b. 对于每个接收到的消息，适配器工作程序从配置文件中检索用于处理消息的目标适配器命令类型，并且为该命令类型获得适当的处理程序。
   c. 处理程序从配置文件中获取任何实例化目标适配器实例所需的任何附加信息。它实例化目标适配器并将消息传递到目标适配器。
   d. 如果消息处理成功（即没有任何异常、错误或错误的返回数据），那么消息就从进入消息队列中提交。如果消息处理不成功，那么就将它放入一个错误队列。如果消息处理不成功并且无法放入错误队列，那么就回滚消息并且关闭所有工作程序。

WebSphere Application Server 内的适配器工作程序的基本处理流程如下:

1. JMS Listener 进程（与 WebSphere Application Server 高级版的 EJB 服务器一起运作）接收 JMS 消息。然后它获取处理消息用的工作程序消息 bean。应用程序的逻辑标识以及可选主体类别和主体类型值是工作程序 bean 的环境中的一部分。主体类别和主体类型值用来获取附加配置值。
2. 每个工作程序消息 bean 执行下列任务:
   a. 工作程序消息 bean 实例化一个本机适配器并在本机适配器上使用 `receiveMsg` 方法，然后将 JMS 消息传递给它。本机适配器将 JMS 消息转换成消息对象并返回消息存放器对象。
   b. 对于每个接收到的消息存放器对象，适配器工作程序从配置文件中检索用于处理消息存放器对象的目标适配器命令类型，并且为该命令类型获得适当的处理程序。
   c. 处理程序从配置文件中获取任何实例化目标适配器实例所需的任何附加信息。它实例化目标适配器并将消息存放器对象传递到目标适配器。
   d. 如果消息存放器对象处理成功（即没有任何异常、错误或错误的返回数据），那么消息就从进入消息队列中提交。如果消息存放器对象处理不成功，那么就将它放入一个错误队列。如果消息处理不成功并且无法放入错误队列，那么就回滚消息并且关闭所有工作程序。

具有适配器守护程序和独立适配器工作程序的目标方上的运行时流如下所示:

1. 每个目标应用程序的接收队列都有一个适配器守护程序。要启动适配器守护程序。

   在启动时，向它提供一个名称作为应用程序标识。通常，每个适配器守护程序的名称是基于目的地逻辑标识（即目标应用程序的逻辑标识）。例如，如果适配器守护程序正为之服务的目标应用程序的目的地逻辑标识是 ABC，则适配器守护程序的名称是 ABCdaemon。

   启动时可以传送到适配器守护程序的其它参数包括主体类别和主体类型。本机适配器稍后使用它们来确定通信方式和进入消息的接收队列。

   有关启动适配器守护程序的指令，请参阅第78页的『启动内核』。

2. 启动时，适配器守护程序在配置文件中查找，以确定是否对该适配器守护程序的名称启用跟踪。如果启用跟踪，则适配器守护程序实例化跟踪客户机。

   有关跟踪的详细信息，请参阅 *Problem Determination Guide*。

3. 启动时，适配器守护程序实例化第一个工作程序，并向它传送适配器守护程序的名称、主体类别和主体类型。

4. 第一个工作程序在配置文件中查找，以确定是否对该适配器守护程序名称启用跟踪。如果启用跟踪，则第一个工作程序实例化跟踪客户机，并且跟踪客户机在配置文件中查找以确定跟踪级别。有关有效跟踪级别的列表，请参阅 *Problem Determination Guide*。

5. 第一个工作程序根据适配器守护程序的应用程序标识，在配置文件中查找指示要实例化和启动的最小工作程序数的值。

   第一个工作程序还查找*相关应用程序标识*。相关应用程序标识是工作程序提供服务的应用程序名称。它稍后传送给本机适配器。

6. 适配器守护程序在第一个工作程序中查询最小工作程序数。

7. 适配器守护程序启动第一个工作程序，然后实例化并启动最小数目的工作程序。

   具有多个工作程序的目的是启用到目标适配器的多线程消息传递。每个工作程序和它的本机适配器可以处理一个线程。如果只有一个工作程序，则消息传递到目标适配器再到目标应用程序是单线程的。

   在 AIX 系统上，有两个调度策略可用于线程: 基于进程的调度和基于系统的调度。在基于进程的调度（缺省值）中，所有用户线程都映射到一个操作系统 (OS) 内核线程池中，并运行在一个虚拟处理器的池上。在基于系统的调度中，每个用户线程映射到单个 OS 内核线程，并运行在单个虚拟处理器上。如果在 AIX 上使用从 C 可执行文件中调用的 C 源适配器，则必须使用基于系统的调度。有关在 AIX 上设置线程调度策略的信息，请参阅第38页的6步。

注意: Windows 系统、HP-UX、Solaris 和 OS/400 上只支持基于进程的调度。其它工作程序还执行第一个工作程序执行的下列步骤:

8. 每个工作程序实例化与它相关的本机适配器。每个工作程序与一个本机适配器相关联。相关应用程序标识、主体类别和主体类型传送到本机适配器中。本机适配器使用这三个值来确定通信方式，并利用逻辑消息服务来确定格式和进入消息的接收队列。这一进程与用于发送消息的进程类似。
9. 本机适配器在提交控制下从接收队列处获得通信消息，并将它转换为消息存放器对象。它除去所有特定于通信传送工具的头部分（本机内核头部分除外）。
10. 本机适配器将消息存放器对象传送到工作程序，该工作程序从消息的本机内核头中读取主体类别、主体类型和请求的确认值。

    根据相关应用程序标识、主体类别和主体类型，工作程序在配置文件中按下列次序执行多阶段查找，以获得调用的目标命令类型:

    a. 特定主体类别和主体类型值。
    b. 特定主体类别值和缺省主体类型值。
    c. 缺省主体类别值和特定主体类型值。
    d. 缺省主体类别和主体类型值。

    工作程序基于目标命令类型确定适当的目标适配器类型处理程序（一个处理特定适配器类型的 Java 类）。它实例化特定的目标适配器。
11. 有两种类型的适配器类型处理程序: EAB 命令目标适配器处理程序和 EJB 服务会话 bean 目标适配器处理程序。两种适配器处理程序工作方式之间的不同之处是:

    注: 仅在 Windows NT 平台上，WebSphere Business Integrator 与 WebSphere Application Server 一起运行时才支持 EJB 服务会话 bean 目标适配器处理程序。

    - 如果调用了一个 EAB 命令目标适配器处理程序，那么它会启动 Common Connector Framework (CCF) 环境、使用从配置文件中获得的名称设置一个登录类，并使用从配置文件中获得的名称调用 EAB 目标适配器。
    - EJB 服务会话 bean 目标适配器必须与 WebSphere Business Integrator 和 WebSphere Application Server 交互，以获得适当的配置信息并调用 EJB 服务会话 bean。请参阅第22页的『MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 一起使用』获得有关将内核与 JMS、WebSphere Business Integrator 的 JMS Listener 组件，以及内核目标方上的 WebSphere Application Server 一起使用的论述。
12. 每个适配器类型都有一个不同接口和必需的支持类，如下所示:
    - 一个 EAB 目标适配器命令有三个调用方法；这些方法按以下次序运行:

a. *设置消息输入方法*，将要处理的消息设置到目标适配器中。

b. *执行方法*，它处理使用设置消息输入方法放入目标适配器的消息，然后等待。

1) 目标适配器执行使用 MQSeries Adapter Builder 构建在其中的功能。通常，它将数据从集成消息变换成目标应用程序格式。它将元素与元素映射。

2) 通过使用应用程序特定接口，目标适配器将消息发送到目标应用程序。

3) 根据目标应用程序的特性，目标应用程序将应答发送回目标适配器或不发送应答。

c. *获得消息输出方法*，从目标适配器中获得应答。应答可以简单地指示目标应用程序接收到了消息；也可以包含数据。

- EJB 服务会话 bean 调用一个方法，它需要一个 `TerminalDataContainer` 对象。由方法返回的数据被当作应答数据并且必须是 `TerminalDataContainer` 类型对象。

13. 如果目标适配器命令没有传出异常或它没有“确认 BOD”应答（可以指示错误），则工作程序通过使用本机适配器提交接收队列中已接收的消息。

14. 如果请求了确认，则工作程序在本机适配器上调用 `sendResponse` 方法。

- 如果目标适配器创建了应答，那么它将原始消息的响应逻辑标识放在应答消息的目的地逻辑标识字段中。
- 如果目标适配器没有创建应答，则工作程序创建一条包含完成状态的“确认 BOD”应答消息。
  - 如果没有错误，则完成状态是成功。
  - 如果有错误，则完成状态设置为一个错误情况。

15. 发送应答。

a. 如果创建了应答消息，则工作程序将它发送到本机适配器。

b. 本机适配器将应答消息放入应答队列。

c. 本机适配器根据接收的原始消息发送应答消息:

- 如果它是一条 MQSeries 请求消息，则本机适配器从 MQSeries 请求消息中获得应答的队列信息。该队列信息覆盖消息中的目的地逻辑标识。
- 如果它不是 MQSeries 请求消息，则本机适配器使用 `sendMsg` 方法发送应答。

16. 在有异常或“确认 BOD”应答消息有错误状态的情况下，工作程序将异常消息记录在名为 `EpicSystemExceptionFile`*`nnnnnnnn`*`.log` 的异常文件中，这个文件与适配器守护程序在同一个目录中，其中 *nnnnnnnn* 是日志文件的编号。

另外，如果安装了 WebSphere Business Integrator 类，这些类会发送一个异常到 WebSphere Business Integrator Solution Management 组件。请参阅第81页的『异常消息』。

17. 在有异常或“确认 BOD”应答消息有错误状态的情况下，工作程序引导本机适配器将原始消息放在错误队列上。根据原始消息的相关逻辑标识、主体类别和主体类型，从配置文件中获得错误队列的名称。

    根据相关应用程序标识、主体类别和主体类型，工作程序在配置文件中按下列次序执行多阶段查找:

    a. 特定主体类别和主体类型值。

    b. 特定主体类别值和缺省主体类型值。

    c. 缺省主体类别值和特定主体类型值。

    d. 缺省主体类别和主体类型值。

    - 如果本机适配器能够将错误消息放在错误队列上，则引导本机适配器提交接收队列中的消息。
    - 如果本机适配器不能将错误消息放在错误队列上，会发生下列情况:

      a. 工作程序引导本机适配器回滚，即不提交。

      b. 工作程序设置一个标志，以引导关闭该适配器守护程序下的所有工作程序。这表示消息有问题。关闭所有工作程序防止其它工作程序重新处理将产生相同结果的同一个问题消息。

      c. 如果发生内存不足的错误，对待这一异常的方法与其它所有异常相同，不同之处仅在于工作程序为它自己设置一个标志，以便在完成处理当前消息后停止。这样使更多内存可用于其它工作程序。

18. 当本机适配器通知工作程序已执行完操作时，工作程序检查两个标志:
    - 是否要停止这个工作程序。这可以因 Java 内存不足的情况引起。
    - 是否要停止所有工作程序，因上一步中描述的原因引起。
19. 如果设置了其中一个标志，工作程序就停止。如果没有设置其中任何一个标志，则工作程序处理下一个消息。工作程序请求本机适配器接收消息。
20. 如果应答消息放在应答队列上，或如果错误消息放在错误队列上，会发生下列情况:

    a. MQSeries 或其它消息传递软件将它传回内核的源方。

    b. 如果源适配器调用了它本机适配器的 `sendRequestResponse` 方法，则内核从应答队列中检索到消息，并将它返回至源适配器。如果源适配器调用了 `sendMsg` 方法，则内核将消息放入源应用程序的接收队列。

事务性能力

*事务*是一组必须作为不可分的工作单元执行的操作。如果构成某一事务的所有操作都成功，则*提交*事务；即执行所有操作。如果构成某一事务的一个或多个操作失败，则*回滚*事务；即不执行任何操作。通过使用 MQSeries Adapter Kernel 的事务性能力，源适配器可以将一系列操作作为单个单元执行，并确保如果提交了事务则所有操作都成功，如果回滚事务则没有任何操作发生。

通过使用 MQSeries Adapter Builder 或在内核的 Java API 的 `EpicNativeAdapter` 类上使用 `begin`、`rollback` 和 `commit` 方法，可以将事务性能力构建在适配器中。如果在非法的上下文中调用事务性方法（例如，在尚未调用 `begin` 方法之前就调用 `commit` 方法，或在另一个事务范围内调用 `begin` 方法），则内核会忽略该调用并向跟踪发出警告。有关使用 API 的信息，请参阅第83页的『第5章 使用 MQSeries Adapter Kernel API』。

限制

下列限制与内核的事务性能力有关:

- 用 `sendRequestResponse` 方法不支持事务。
- 不支持嵌套事务（即，在其它事务中调用事务）。
- 所有的通信方式都不支持事务；有关详细信息，请参阅第87页的『附录A. 通信方式』。

跟踪

跟踪消息包含内核中某一时刻处理消息的状态。可以使用跟踪消息帮助诊断内核或适配器的问题。MQSeries Adapter Kernel *Problem Determination Guide* 论述了有关对内核使用跟踪的信息。

MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 一起使用

这部分讨论了如何将 MQSeries Adapter Kernel 与 WebSphere Business Integrator 和 WebSphere Application Server 产品一起使用。有关其它详细信息，请参阅 WebSphere Business Integrator 文档。

JMS Listener

WebSphere Business Integrator 提供了一个名为 JMS Listener 的组件，它与 MQSeries Adapter Kernel 和 WebSphere Application Server 高级版一起工作，提供了一种将消息传递到目标应用程序的替代方式。 JMS Listener 在 WebSphere Application Server 的 Enterprise JavaBean (EJB) 服务器内部运行。这部分提供了 JMS Listener 的功能概述。有关附加信息（包括配置 WebSphere Business Integrator

和 JMS Listener 的详细信息），请参阅 WebSphere Business Integrator 文档。请参阅第50页的『配置内核』获得有关配置 MQSeries Adapter Kernel 以将 JMS Listener 识别为目标的信息。使用 JMS Listener 作为目标等价于将消息发送到适配器守护程序。

在可以使用 JMS Listener 之前，必须为内核目标方部署一个 MQSeries Adapter Kernel 消息 bean 适配器工作程序和 Java 服务会话 bean 适配器（或 EAB 适配器）。使用 MQSeries Adapter Builder 执行这些任务。在 WebSphere Business Integrator 环境中， WebSphere Application Server 内的内核操作跟它与独立适配器守护程序的操作类似，不同之处是 JMS Listener 代表适配器工作程序接收消息并调用适当的适配器工作程序。在独立 MQSeries Adapter Kernel 环境中，适配器守护程序启动适配器工作程序，然后由它直接接收消息。

当 MQSeries Adapter Kernel 与 JMS Listener 一起工作时的事件序列如下所示:

1. 监控 JMS 队列的 JMS Listener 从 EJB 客户机或非 EJB 应用程序接收 JMS 消息对象。
2. JMS Listener 实例化一个*工作程序消息 bean* 并将消息对象传递给它。工作程序消息 bean 是*会话 bean* 的一个实例，它是一种 enterprise bean，将与特定客户机相关的临时数据封装起来。
3. 工作程序消息 bean 将 JMS 消息对象转换为 MQSeries Adapter Kernel 消息存放器对象。
4. 内核基于消息的头值调用 EAB 适配器或 EJB 适配器。如果要调用的适配器类型是 EAB 适配器，那么数据流是与使用独立适配器的情况相同。如果要调用的适配器类型是 EJB 适配器，那么会调用 EJB 处理程序并执行下列任务:
   - 它确定要调用的正确服务会话 bean（主接口）、要调用的适当方法以及 `TerminalDataContainer` 对象的方法输入参数类型。
   - 它通过使用一个 `Mapper` 类，将消息存放器对象中包含的应用程序数据转换成适当的服务会话 bean 的 `TerminalDataContainer` 数据结构。`TerminalDataContainer` 对象包含消息存放器对象的元数据以及应用程序对象。在许多情况下，应用程序对象是消息存放器对象的主体数据 XML 文档字符串。
   - 它调用服务会话 bean，将 `TerminalDataContainer` 对象传递给服务会话 bean 上的适当方法。作为 Java 服务适配器一部分的服务会话 bean 是消息的目标方。
5. 如果请求了应答，工作程序消息 bean 会将应答 `TerminalDataContainer` 对象转换为消息存放器对象，并使用本机适配器发送应答。
6. 如果发生错误，工作程序消息 bean 会使用本机适配器将消息存放器对象放入一个错误队列。

国家语言支持

MQSeries Adapter Kernel 提供了在使用 Java 适配器时的国家语言支持。没有为 C 适配器提供国家语言支持。

第2章 规划安装内核

本章列出了 MQSeries Adapter Kernel 的先决条件和组件。

有关最新的详细信息，请参阅 MQSeries 产品系列 Web 站点:

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

IBM 保留更新这里显示信息的权利。有关所支持的软件级别的最新信息，请参考:

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

硬件

MQSeries Adapter Kernel 在下列硬件上运行:

- 一台运行 Windows NT 4.0, Service Pack 5（或更新版本）或 Windows 2000, Service Pack 1 的 IBM PC 机（或兼容机）。
- 一台运行 AIX 版本 4.3.2 或 4.3.3 的 IBM RS/6000 机器。
- 一台运行 HP-UX 版本 11.0 的 HP 系列 9000 机器。
- 一台运行 Solaris 版本 8 的 Sun SPARC 或 UltraSPARC 机器。
- 一台运行 OS/400 版本 4.4 或 4.5 的 IBM AS/400 或 iSeries 机器。

注: 在 OS/400 上安装 MQSeries Adapter Kernel 需要一个 Windows 系统与 AS/400 机器交互。有关详细信息，请参阅第28页的『OS/400 安装的先决条件』。

MQSeries Adapter Kernel 最少需要大约 25 MB 的磁盘空间用于产品代码和数据。

请确保有足够的磁盘空间可用于存放适配器。它们的大小取决于数据结构的大小、映射的复杂程度和所用的定制代码。下面列出了 Windows 系统上不同适配器大小的一些示例。您所在位置的适配器可能会需要更多或更少的磁盘空间。每个示例表示以 MB 或 KB 为单位的适配器源、已编译的适配器码、API 源和已编译的 API 代码。

- 添加销售订单的源适配器: 1.89 MB
- 同步化客户记录的目标适配器: 389 KB
- 同步化库存记录的目标适配器: 161 KB
- 同步化某个项的目标适配器: 249 KB
- 同步化销售订单的目标适配器: 579 KB

另外，允许最少 20 MB 的工作空间用于内核和适配器。工作空间的需求会根据一些因素（如队列数、队列大小和跟踪文件的大小）而变化。

软件

本节列出了支持与 MQSeries Adapter Kernel 一起使用的软件。显示了支持级别。请参阅第93页的『附录B. 已确认的配置』。注意：在开发系统上需要 C 编译器，而产品系统上不需要。这里列出的 C 编译器已经与 MQSeries Adapter Kernel 一起成功测试过了；其它 C 编译器可能也可以与内核一起正常工作，但并没有正式支持。

对于 Windows 系统:

- Microsoft Windows NT 版本 4.0, Service Pack 5 或更新版；或 Microsoft Windows 2000, Service Pack 1。要确定 Microsoft Windows 的版本和 service pack，请打开 Windows 资源管理器，然后单击**帮助 > 关于 Windows**。
- Microsoft Visual C++ 6.0 编译器。
- MQSeries 版本 5.2，带有 SupportPac MA88。
- IBM Java 开发工具箱 (JDK) 版本 1.2.2 或 1.3。

  注: Windows NT 和 Windows 2000 是当前 MQSeries Adapter Kernel 支持 JDK 版本 1.3 的仅有平台。

对于 AIX:

- AIX 操作系统版本 4.3.2 或 4.3.3.
- IBM C Set++ for AIX 版本 3.1.3。
- MQSeries 版本 5.2，带有 SupportPac MA88。
- Java 开发工具箱版本 1.2.2。不支持 JDK 1.3。
- X Window System （X11R5 或更高版本）。这是安装所必需的，但不是运行时所需的。

对于 HP-UX:

- HP-UX 操作系统版本 11.0。
- HP-UX C/ANSI C 编译器。有关详细信息，请参阅 `readme.txt` 文件。
- MQSeries 版本 5.2，带有 SupportPac MA88。
- Java 开发工具箱版本 1.2.2。不支持 JDK 1.3。
- X Window System（X11R5 或更高）。这是安装所必需的，但不是运行时所需的。

对于 Solaris:

- Solaris 操作环境版本 8。
- Sun Workshop Compilers C/C++。有关详细信息，请参阅 `readme.txt` 文件。
- MQSeries 版本 5.2，带有 SupportPac MA88。
- Java 开发工具箱版本 1.2.2。不支持 JDK 1.3。
- X Window System（X11R5 或更高）。这是安装所必需的，但不是运行时所需的。

对于 OS/400:

- OS/400 操作系统版本 4.4 或 4.5，包括下列程序:
  - Java 工具箱和 Java 开发者工具箱版本 1.2.2。不支持 JDK 1.3。 Java Toolkit 和 Java Developer Kit 作为特许程序号 5769-JV1 交付使用。有关在 AS/400 系统上安装 MQSeries Adapter Kernel 必需的 Java Developer Kit 的版本的附加细节，请参阅第28页的『OS/400 安装的先决条件』。
  - Host Servers 选项，作为特许程序号 5769-SS1 交付使用，选项 12。
  - Qshell Interpreter，作为特许程序号 5769-SS1 交付使用，选项 30。
  - TCP/IP，作为特许程序号 5769-TC1 交付使用。
  - Integrated Language Environment C for AS/400，作为特许程序号 5769-CX2 交付使用。
- MQSeries 版本 5.2，带有 SupportPac MA88。

有关在 AS/400 上安装 MQSeries Adapter Kernel 的附加需求，请参阅第28页的『OS/400 安装的先决条件』。

MQSeries Adapter Kernel 支持下列产品:

- MQSeries 版本 5.2，带有 SupportPac MA88

  注: 如果不使用 MQSeries，则必须使用诸如实现了 “Java 消息服务 (JMS)” 的消息传递软件。
- MQSeries Integrator 版本 1.1
- MQSeries Integrator 版本 2

有关已确认的 MQSeries Adapter Kernel、MQSeries 和 MQSeries Integrator 配置的列表，请参阅第93页的『附录B. 已确认的配置』。

OS/400 安装的先决条件

本节描述了在 AS/400 或 iSeries 系统上安装 MQSeries Adapter Kernel 的先决条件。有关在 AS/400 系统上安装 MQSeries Adapter Kernel 的详细指令，请参阅第35页的3步。因为 AS/400 终端在本机不支持 Java 图形，因此必需一个支持图形的工作站，如 Windows 系统，来运行内核的基于 Java 的 GUI 安装程序。工作站可以用下列一种方法与 AS/400 系统交互:

- 通过远程 AWT，用此方法，所有图形都在 AS/400 系统上处理并在工作站上显示。这在『使用远程 AWT』中有详细描述。
- 作为连接的客户机，用此方法，工作站处理并显示图形。这在第29页的『使用连接的客户机』中有详细描述。

本节假设将 Windows 系统用作支持图形的工作站。

使用远程 AWT

使用远程 AWT 时，在 AS/400 系统上执行 Java 图形的处理，在与 AS/400 系统相连的客户机工作站上显示图形。本节描述了通过使用远程 AWT 在 AS/400 系统上安装 MQSeries Adapter Kernel 必须满足的需求。

下列程序必须与 OS/400 一起安装:

- Java 工具箱和 Java 开发者工具箱版本 1.2.2。Java Toolkit 和 Java Developer Kit 作为特许程序号 5769-JV1 交付使用。 OS/400 上的远程 AWT 功能由 Java Developer Kit 提供。
- TCP/IP，作为特许程序号 5769-TC1 交付使用。有关 TCP/IP 的详细信息，请参阅 *AS/400 TCP/IP Fastpath Setup Information* 和 *AS/400 TCP/IP Configuration* 文档，它们可从 AS/400 书库: www.ibm.com/servers/eserver/iseries/library/ 中获得。

工作站的需求如下:

- 一台运行 Windows 95、Windows 98、Windows NT 或 Windows 2000 的 IBM PC 机（或兼容机）。
- 一个至 AS/400 系统的 TCP/IP 连接。
- JDK 1.2.2 或更高版本。

要设置并启动远程 AWT，执行下列步骤:

1. 确保工作站上安装了 JDK 1.2.2 或更高版本。
2. 确保 AS/400 系统和工作站之间存在 TCP/IP 连接。
3. 将 `RAWTGui.jar` 文件从 AS/400 系统上的 `/QIBM/ProdData/Java400/jdk12` 目录复制到工作站上的一个目录。

4. 在工作站上，将目录改到复制 RAWTGui.jar 文件的目录处，并通过输入下列命令启动远程 AWT:

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

注: 由于在 AS/400 系统上处理 Java 图形的资源集中的特性，使用远程 AWT 安装 MQSeries Adapter Kernel 所花费的时间可能会比使用连接的客户机的要长。

请参阅 AS/400 书库: www.ibm.com/servers/eserver/iseries/library/，获得有关远程 AWT 的详细信息。

使用连接的客户机

当使用连接的客户机在 AS/400 系统上安装 MQSeries Adapter Kernel 时，Java 图形的处理在客户机工作站上执行，而不是在 AS/400 系统上执行。本节描述了使用连接的客户机在 AS/400 系统上安装 MQSeries Adapter Kernel 必须满足的需求。

下列程序必须与 OS/400 一起安装:

- Java 工具箱和 Java 开发者工具箱版本 1.2.2。Java Toolkit 和 Java Developer Kit 作为特许程序号 5769-JV1 交付使用。
- Host Servers 选项，作为特许程序号 5769-SS1 交付使用，选项 12。
- TCP/IP，作为特许程序号 5769-TC1 交付使用。

工作站的需求如下:

- 一台运行 Windows NT 4.0，Service Pack 5 或 Windows 2000，Service Pack 1 的 IBM PC 机（或兼容机）。
- 一个至 AS/400 系统的 TCP/IP 连接。
- JDK 1.2.2 或更高版本。

内核的组件

安装后，MQSeries Adapter Kernel 驻留在它的根目录中。它包含的子目录依次可以包含其它目录。以下列出了根目录和子目录，以及与安装和配置密切相关的文件的摘要。

root 在 Windows 系统上，缺省名称是 C:\Program Files\MQAK，在 AIX 上是 /usr/lpp/mqak，在 HP-UX 上是 /MQAK，在 Solaris 上是 /opt/MQAK，在 OS/400 上是 /QIBM/ProdData/mqak。它包含下列内容:

- 所有其它的 MQSeries Adapter Kernel 目录。
- aqmsetenv.bat（Windows 系统）或 aqmsetenv.sh (UNIX) 文件，根据需要，它在安装后可以更改系统环境变量。

- readme.txt 文件。
- aqmuninstall.bat（Windows 系统）或 aqmuninstall.sh (UNIX) 文件。

bin 包含下列内容:

- 类库和共享程序库。
- 作为内核一部分提供的适配器，仅用于验证。
- aqmversion.bat（Windows 系统）或 aqmversion.sh（UNIX 和 OS/400）文件，运行该脚本以显示内核的版本号。
- aqmcrtmsg.bat（Windows 系统）或 aqmcrtmsg.sh（UNIX 和 OS/400）文件，运行该脚本，以创建在将配置文件放入产品之前用来确认该文件的 XML 文件。
- aqmsndmsg.bat（Windows 系统）或 aqmsndmsg.sh（UNIX 和 OS/400）文件，在将配置文件放入产品之前运行该脚本以确认它。
- aqmstrad.bat（Windows 系统）或 aqmstrad.sh（UNIX 和 OS/400）文件，运行该脚本以启动适配器守护程序。
- aqmstrtd.bat（Windows 系统）或 aqmstrtd.sh（UNIX 和 OS/400）文件，运行该脚本以启动跟踪服务器。

documentation

包含产品文档，包括 “信息中心”。

runtimefiles

包含内核运行时文件。

samples

包含适配器的样本及相关配置文件和实用程序文件。可以练习使用它们并从中学习。

注: 设计内核的目的是将其与使用 MQSeries Adapter Builder 构建的适配器一起使用。内核不可以在仅从定制代码调用内核 API 时使用。提供适配器样本仅为帮助理解内核是如何工作的，并帮助诊断。

- 适配器样本。
- 内核的设置文件，aqmsetup，带有支持适配器样本的值。有关该文件的论述，请参阅第55页的『设置文件』。
- 内核的配置文件，aqmconfig.xml，带有支持适配器样本的值，包括样本跟踪值。有关该文件的论述，请参阅第55页的『配置文件』。

toolkit

包含由下列各项组成的软件开发工具箱 (SDK):

- 头文件。
- 在 Windows 系统下编译期间所用的库文件。

uninstall

包含用来卸载内核的文件。

verification

包含支持验证内核安装的下列文件:

- aqmverifyinstall.bat（Windows 系统）或 aqmverifyinstall.sh（UNIX 和 OS/400）文件，运行该脚本以验证一台计算机上的内核安装。
- aqmcreateq.bat（Windows 系统）或 aqmcreateq.sh（UNIX 和 OS/400）文件，该脚本创建用于验证的 MQSeries 队列。请参阅第82页的『创建 MQSeries 队列』。
- aqmconfig.xml 文件。有关该文件的论述，请参阅第55页的『配置文件』。
- aqmsetup 文件。有关该文件的论述，请参阅第55页的『设置文件』。
- aqminstalltest.xml 文件。

第3章 安装内核

本章讨论安装和验证 MQSeries Adapter Kernel 所必需的步骤。安装包括下列几个常规步骤:

步骤 1. 准备安装。有关详细信息，请参阅『准备安装』。

步骤 2. 安装内核。有关详细信息，请参阅第34页的『安装内核』。

步骤 3. 完成几个后安装步骤。有关详细信息，请参阅第37页的『完成后安装』。

步骤 4. 验证安装。有关详细信息，请参阅第39页的『验证安装』。

本章还讨论了下列主题:

- 使用安静安装方式安装 MQSeries Adapter Kernel。有关详细信息，请参阅第43页的『使用安静安装』。
- 从早期版本升级 MQSeries Adapter Kernel。有关详细信息，请参阅第45页的『升级内核』。
- 除去 MQSeries Adapter Kernel 的安装。有关详细信息，请参阅第46页的『除去内核』。

安装内核后，执行下列附加任务以准备使用:

1. 配置内核。有关详细信息，请参阅第50页的『配置内核』。
2. 配置消息传递软件和可选软件。有关详细信息，请参阅第77页的『配置 MQSeries 和 MQSeries Integrator』。
3. 使用 MQSeries Adapter Builder 构建适配器，然后测试并部署它们。
4. 启动内核。有关详细信息，请参阅第78页的『启动内核』。

准备安装

必须具有管理员或 root 权限才能安装 MQSeries Adapter Kernel。必须有权在安装 MQSeries Adapter Kernel 的位置和放置两个内核配置文件的位置中创建并访问文件。在可执行路径中必须有当前目录。请确保运行内核的所有用户标识都具有读写和执行许可权。

必须有权执行诸如创建队列管理器和创建并访问队列等 MQSeries 操作。这些操作在不同的平台上以不同方法执行。有关详细信息，请参考针对平台的 *MQSeries Administration Guide*。

启动内核进程的用户标识必须属于 mqm 组。有两种内核进程:

- 适配器守护程序，计算机提供服务的每个目标应用程序有一个守护程序。
- 跟踪服务器（可选的）

注意: 源适配器在源应用程序的进程中运行。任何包含源适配器的守护程序或服务器都需要启动以便源适配器正常运行。

必须安装并配置内核以运行构建的适配器。但是，不必为安装 MQSeries Adapter Builder 而安装内核，或用它来构建适配器。

在开始安装之前执行下列步骤:

- 阅读 CD-ROM 或局域网中的 `readme.txt` 文件。它可能包含本书完成后才获得的重要信息。它在安装根目录中。
- 访问 MQSeries Web 站点: www.ibm.com/software/ts/mqseries/。它可能包含本书出版后才获得的重要信息，可能包括本书的新版本。
- 如果要从 MQSeries Adapter Kernel 的先前版本升级，请参阅第45页的『升级内核』，获取指令。
- 请确保满足硬件和软件先决条件。有关详细信息，请参阅第25页的『硬件』和第26页的『软件』。在可以验证 MQSeries Adapter Kernel 的安装前，必须安装并运行 MQSeries。请确保安装并配置了 MQSeries Java 支持。

安装内核

要将 MQSeries Adapter Kernel 安装在 Windows 系统（Windows NT 或 Windows 2000）、 UNIX 平台（AIX、HP-UX 或 Solaris）或 OS/400 上，请执行下列操作系统特定的步骤:

在 Windows 系统上:

步骤 1. 如下启动安装程序:

- 如果要从局域网安装，将目录改到包含 MQSeries Adapter Kernel 安装文件的目录，并运行 `install.bat` 文件。
- 如果要从 CD-ROM 安装，将 MQSeries Adapter Kernel CD-ROM 插入 CD-ROM 驱动器。如果启用自动运行，会自动启动安装程序；如果未启用自动运行，在 CD-ROM 的根目录运行 `install.bat` 文件，以启动安装程序。

注: 在 Windows 系统上，在运行 `install.bat` 文件之前不必将它复制到另一个位置。安装过程期间，会要求您选择安装 MQSeries Adapter Kernel 的位置。

步骤 2. 按照安装程序提供的提示执行。注意: 如果选择将 MQSeries Adapter Kernel 安装在非缺省的位置（在 Windows 系统上，缺省位置是 C:\Program Files\MQAK），必须将安装目录作为全限定路径名指定，而不能作为相对路径名指定。

在 UNIX 上:

步骤 1. 如下启动安装程序:

- 如果要从局域网安装，将目录改到包含 MQSeries Adapter Kernel 安装文件的目录，并运行 install.sh 脚本。
- 如果要从 CD-ROM 安装，将 MQSeries Adapter Kernel CD-ROM 插入 CD-ROM 驱动器，并且需要的话，根据操作系统文档，安装 CD-ROM 驱动器。运行 CD-ROM 根目录中的 install.sh 脚本。

步骤 2. 按照安装程序提供的提示执行。注意: 如果选择将 MQSeries Adapter Kernel 安装在非缺省位置，必须将安装目录作为全限定路径名指定，而不能作为相对路径名指定。UNIX 上的缺省安装目录如下:

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

在 OS/400 上:

步骤 1. 确保已满足第25页的『硬件』、第27页的OS/400 软件先决条件和第28页的『OS/400 安装的先决条件』中列出的所有先决条件。注意: 在 OS/400 上安装 MQSeries Adapter Kernel 使用基于 InstallShield 的程序，它需要使用工作站与 AS/400 系统交互；有关详细信息，请参阅第28页的『OS/400 安装的先决条件』。

步骤 2. 对于 AS/400 系统，在“控制语言 (CL)”提示处使用 **CRTUSRPRF** 命名创建一个名为 MQAKSRV 的用户简要表。

步骤 3. 根据是使用远程 AWT 还是使用连接的客户机工作站来执行安装，执行下列步骤:

- 如果使用远程 AWT 来执行卸载，执行下列步骤:
  a. 确保设置了远程 AWT 并在运行中。有关详细信息，请参阅第28页的『使用远程 AWT』。
  b. 确保 AS/400 系统可以访问 installAS400.jar 文件。该文件必须在集成文件系统 (IFS) 中或在一个与 AS/400 系统连接的设备上。如果文件在一个连接的设备上，使用“创建链接”(**CRTLINK**) 命令来创建至该文件的符号链接。

c. 要改进安装进程的性能，对 installAS400.jar 文件运行“创建 Java 程序”(**CRTJVAPGM**) 命令。

d. 如下运行“运行 Java”(**RUNJVA**) 命令，其中 *n.n.n.n* 表示运行远程 AWT 的工作站的 TCP/IP 地址:

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

- 如果使用连接的客户机工作站来执行卸载，执行下列步骤:

  步骤 a. 确保满足了第29页的『使用连接的客户机』中指定的需求。

  步骤 b. 确保 AS/400 机器上安装了 Host Servers 选项，并在运行中。通过在 CL 提示处使用“启动 Host Servers”(**STRHOSTSVR**) 命令，可以启动 Host Servers。

  步骤 c. 确保 AS/400 机器上安装了 TCP/IP，并在运行中。通过在 CL 提示处使用“启动 TCP/IP”(**STRTCP**) 命令，可以启动 TCP/IP。

  步骤 d. 在工作站上，打开命令提示，并将目录改为 MQSeries Adapter Kernel 安装媒体（局域网或 CD-ROM）的 AS400 目录。

  步骤 e. 输入下列命令:

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

步骤 4. 安装程序开始并显示**注册到 AS/400** 面板。在**系统**: 字段中输入 AS/400 机器的 TCP/IP 地址，并在相应的字段中输入用户标识和密码。不要选中**缺省用户**复选框。单击**下一步**。

步骤 5. 按照安装程序提供的提示执行。根据网络和机器的速度，安装进程至多会花一个小时的时间完成。工作站上显示的进展条指示安装的状态。

注意: 在 OS/400 上，MQSeries Adapter Kernel 总是安装在集成文件系统 (IFS) 根目录中的 /QIBM/ProdData/mqak 目录中。

步骤 6. 如下所示设置 CLASSPATH、PATH 和 QIBM_MULTI_THREADED 环境变量:

- 将 /QIBM/ProdData/mqak/bin 目录添加到 CLASSPATH 环境变量。
- 将 /QIBM/ProdData/mqak/bin 目录添加到 PATH 环境变量。
- 将 QIBM_MULTI_THREADED 环境变量设置为 Y。

步骤 7. 将库 MQAK 添加到 QSYS.LIB 库列表中。

内核安装完成。到目前为止的安装，所配置的内核支持验证，但不支持在特定位置处的产品。通过执行第39页的『验证安装』中列出的步骤，验证安装。验证安装后，请遵循『完成后安装』中的步骤来设置环境变量以及移动一些配置文件以支持您所在位置上的产品。

按需要，在其它计算机上安装内核。

完成后安装

安装完内核后，请执行下列步骤:

步骤 1. 确定放置 aqmsetup 和 aqmconfig.xml 文件的位置，它们用于配置内核。有关这两个文件的详细信息，请参阅第50页的『配置内核』。

注意:
如果没有创建自己的配置文件，而是使用产品的 samples 目录中提供的配置文件，那么安装新版本的内核会覆盖它们并破坏产品配置。

步骤 2. 为两个配置文件创建目录。它们不需要位于同一个目录中，但建议这样做更简单。如果将它们放在安装 MQSeries Adapter Kernel 的目录之外，则在以后卸载内核时会留下更少的目录。卸载进程保留所有不包含原始 MQSeries Adapter Kernel 文件的目录。

步骤 3. 将 aqmsetup 和 aqmconfig.xml 文件从 samples 目录复制到期望的位置。可以将它们放在网络驱动器或其它许多计算机可以访问的中央位置，以更方便地更新它们或备份它们。

如果重命名 aqmconfig.xml 文件，则内核不能正确操作。可以重命名 aqmsetup 文件，只要在步骤 5 中正确设置了环境变量来指向它。

步骤 4. 利用文本编辑器，编辑 aqmsetup 文件，以指向 aqmconfig.xml 文件的期望目录。将全限定路径名（非相对路径名）用作目录的位置。路径中不要包括本身文件名。示例如下:

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

即使 aqmconfig.xml 文件的期望位置与 aqmsetup 文件所在的位置是同一个目录，仍必须在此输入全限定路径名。保存并关闭 aqmsetup 文件。

步骤 5. 设置 AQMSETUPFILE 环境变量以指向 aqmsetup 文件的位置（例如，在 Windows 系统上是 C:\Program Files\MQAK\Data\aqmsetup，在 UNIX 上是 /MQAK/data/aqmsetup，在 OS/400 上是 /home/*user_name*/aqmsetup）。注意: 在 OS/400 上，aqmsetup 文件必须总是位于当前用户的主 IFS 目录中（即 /home/*user_name*）。

如果内核安装在网络驱动器上，对每个访问它的计算机执行此步骤。

步骤 6. 如果使用 AIX 并计划使用从 C 程序调用的本机 C 语言源适配器，那么将 AIXTHREAD_SCOPE 环境变量设置为值 S。要在 Bourne shell 或 Korn shell 中设置此环境变量，输入下列命令:

```
export AIXTHREAD_SCOPE=S
```

要在 C shell 中设置此环境变量，输入下列命令:

```
setenv AIXTHREAD_SCOPE S
```

要在登录到 AIX 时自动设置 AIXTHREAD_SCOPE 变量，将此命令添加到 `.profile` 文件（如果使用 Bourne shell 或 Korn shell）或 `.cshrc` 文件中（如果使用 C shell）。

有关调度策略的附加信息，请参阅第18页的7步。

步骤 7. 需要的话，设置 THREADS_FLAG 环境变量。仅当下列条件*都*为真时，才必须设置该变量。

- 使用的操作系统是 Solaris。
- 使用的 Java Development Kit (JDK) 版本是 1.2.2。
- 使用 MQSeries 传送消息。
- 源和目标适配器用 C 编写。

如果这些条件都为真，则将 THREADS_FLAG 环境变量设置为 `native`。要在 Bourne shell 或 Korn shell 中设置此环境变量，输入下列命令:

```
export THREADS_FLAG=native
```

要在 C shell 中设置此环境变量，输入下列命令:

```
setenv THREADS_FLAG native
```

要在登录到 Solaris 时自动设置 THREADS_FLAG 变量，将此命令添加到 `.profile` 文件（如果使用 Bourne shell 或 Korn shell）或 `.cshrc` 文件中（如果使用 C shell）。

完成后安装步骤后，请执行下列任务以准备使用内核:

1. 准备产品。请参阅第49页的『准备产品』。
2. 编辑配置文件。有关详细信息，请参阅第50页的『配置内核』。
3. 配置 MQSeries 和可选软件。请参阅第77页的『配置 MQSeries 和 MQSeries Integrator』。
4. 对于产品系统，考虑第78页的『性能建议』。
5. 启动内核。请参阅第78页的『启动内核』。
6. 建立内核维护计划。请参阅第80页的『维护内核』。

验证安装

安装内核后，通过运行验证脚本来验证安装是否正确。脚本使用源适配器从源应用程序处发送一条测试消息，然后使用内核将消息送往 MQSeries。然后，使用内核从 MQSeries 接收消息，并调用目标适配器。所有这些进程都在单个计算机上运行。

在此验证中，源应用程序是名为 TEST1 的 MQSeries 队列。目标应用程序是另一个名为 TEST2 的 MQSeries 队列。

验证执行下列任务:

- 通过使用 MQSeries 作为消息传递软件，在计算机内端对端地验证内核利用提供的源适配器和目标适配器正确地编组并传递了测试消息。
- 验证安装时提供的 aqmconfig.xml 和 aqmsetup 文件。它们确定内核配置。有关这两个文件的信息，请参阅第50页的『配置内核』。

在将配置文件放到产品之前，可以确认配置文件。请参阅第75页的『确认配置文件』。

与 MQSeries Adapter Kernel 一起提供的安装验证脚本假设已在脚本要运行的机器上安装并配置了 MQSeries。如果要使用非 MQSeries 的消息传递软件，可以编辑安装验证脚本以支持该消息传递软件，如下所示:

1. 将目录改到内核安装的 verification 目录。
2. 在文本编辑器中打开 aqmconfig.xml 文件，并将行 <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr> 更改为 <epicmqppqueuemgr>*queue_manager_name*</epicmqppqueuemgr>，其中 *queue_manager_name* 是队列管理器的名称。
3. 如下编辑 aqmverifyinstall 文件:
   - 如果要在 Windows 系统上执行安装验证，在文本编辑器中打开 aqmverifyinstall.bat 文件，并将行 aqmcreateq TEST2 更改为 aqmcreateq TEST2 *queue_manager_name*，其中 *queue_manager_name* 是队列管理器的名称。
   - 如果要在 UNIX 或 OS/400 上执行安装验证，在文本编辑器中打开 aqmverifyinstall.sh 文件，并将行 aqmcreateq.sh TEST2 更改为 aqmcreateq.sh TEST2 *queue_manager_name*，其中 *queue_manager_name* 是队列管理器的名称。

这一验证使用的一些组件（如目标适配器名称 com.ibm.epic.adapters.eak.test.InstallVerificationTest）不是内核的一部分。它们仅为验证安装而与内核一起提供。

验证完成时，停止验证适配器守护程序。

验证期间不启用跟踪。

验证过程

步骤 1. 验证创建并使用三个 MQSeries 队列。如果在执行验证前这些队列中有消息，验证会失败。清除下列队列中的消息:
- TEST2AIQ
- TEST2AEQ
- TEST2RPL

步骤 2. 请确保有权安装并验证内核。请参阅第33页的『准备安装』。

步骤 3. 如下开始验证:
- 在 Windows 系统上，双击 verification 目录中的 aqmverifyinstall.bat 文件。或者，打开命令提示，更改到 verification 目录中，运行 aqmverifyinstall.bat。
- 在 UNIX 上，打开一个终端，将目录改到 verification 目录，并运行 aqmverifyinstall.sh 文件。
- 在 OS/400 上，执行下列步骤:
  a. 通过输入 **STRQSH** 命令，启动 **qsh** 会话。
  b. 将 /QIBM/ProdData/mqak/verification/aqmsetup 文件复制到主目录 (/home/*user_name*)。
  c. 将目录改到 /QIBM/ProdData/mqak/verification 目录。
  d. 运行 aqmverifyinstall.sh 文件。

  aqmverifyinstall 文件包含有关它如何操作的注解。

步骤 4. 消息“安装验证测试已成功完成”指示成功。需要的话，关闭验证窗口。

步骤 5. 如果失败，请检查验证窗口和日志文件 EpicSystemExceptionFile*nnnnnnnn*.log，以确定错误。

步骤 6. 有关验证期间会遇到的常见问题和可能响应，请参阅第41页的『常见验证问题』。

步骤 7. 根据需要，执行可选验证。有关详细信息，请参阅第43页的『可选验证』。

步骤 8. 返回安装过程，并将内核配置为支持在特定位置中操作。转至第37页的 1。

常见验证问题

本节列出了验证期间会发现的常见问题及可能的解决方案。异常消息中的重要信息用**粗体字**突出显示。

问题: 未找到 aqmsetup 文件。

响应: 确保 AQMSETUPFILE 环境变量设置为 verification 目录中 aqmsetup 文件的位置。

异常消息:

```
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]>.>
```

问题: 未找到 aqmconfig.xml 文件。

响应: 编辑 verification 目录中的 aqmsetup 文件，并确保 AQMCONFIG= 项指向 verification 目录。使用全限定路径名。还要确保 aqmconfig.xml 文件在 verification 目录中。

异常消息:

```
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 <>.>
```

问题: 要放置消息的队列不存在。

响应: 使用 MQSeries 确保异常消息中命名的队列（当验证安装时是 TEST2AIQ）存在并且可以接受消息。请参阅第82页的『创建 MQSeries 队列』。

异常消息:

```
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>.>
```

问题: 未找到目标适配器。

响应: 确保消息中指定的目标适配器存在: com.ibm.epic.adapters.eak.test.InstallVerificationTest。确保 CLASSPATH 环境变量包括内核的 bin 目录。

异常消息:

```
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>]>.>>>>
```

问题: 未找到要装入的用于消息传递的适配器。目的地逻辑标识在 aqmconfig.xml 中没有一项用于在队列的消息中指定的主体类型和主体类别。

响应: 验证期间，出现这一异常消息的最可能原因是: 验证之前，名为 TEST2AIQ 的队列中已存在消息。从 TEST2AIQ 队列中清除所有消息并重试验证。verification 目录中的 aqmconfig.xml 文件中只有一项用于应用程序 TEST2 的命令类名，即针对主体类型 TESTBOD 和主体类别 OAG 的一项。

异常消息:

```
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.>>>>
```

问题: 未启动验证队列管理器。

响应: 确保成功启动了缺省 MQSeries 队列管理器。

异常消息:

```
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>.>

问题: 发生一般 MQSeries 错误。

响应: 确保在机器上正确安装并配置了 MQSeries，并且在运行中。检查 MQException 原因码，并使用 *MQSeries Messages* 文档来确定原因码的原因。

异常消息:

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

可选验证

验证了第一台计算机上的内核安装正确后，可以选择执行下列步骤:

1. 通过使用相同的验证，验证第二台计算机上的内核安装是否正确。
2. 验证可以从一台计算机上的源适配器将一条测试消息发送到另一台计算机的目标适配器。手工配置并执行这个验证。如果选择通过修改与内核一起提供的原始验证文件来开发此验证，则为备份而保留原始验证文件。

使用安静安装

可以使用*安静安装*在所有平台上安装 MQSeries Adapter Kernel。安静安装可让您绕过 MQSeries Adapter Kernel 安装程序，如果使用安装程序则必须手工选择需要的安装选项。如果要在多个机器上安装缺省配置，那么安静安装十分有用。

要安静安装内核，请执行下列操作系统特定的步骤:

在 Windows 系统上:

步骤 1. 打开命令提示并更改到包含 MQSeries Adapter Kernel 安装文件的目录。

步骤 2. 输入下列命令:

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

其中，*install_location* 是期望的安装位置（例如，`D:\mqak`）。

在 UNIX 上:

步骤 1. 在终端上，更改到包含 MQSeries Adapter Kernel 安装文件的目录。如果要从 CD-ROM 安装，将 MQSeries Adapter Kernel CD-ROM 插入 CD-ROM 驱动器，并且需要的话，根据操作系统文档，安装 CD-ROM 驱动器。

步骤 2. 输入下列命令:

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

其中，*install_location* 是期望的安装位置（例如，`/opt/mqak`）。

在 OS/400 上:

如果不是使用连接的客户机来访问 AS/400 机器，则执行下列步骤:

步骤 1. 确保 AS/400 系统可以访问 `installAS400.jar` 文件。该文件必须在集成文件系统 (IFS) 中或在一个与 AS/400 系统连接的设备上。如果文件在一个连接的设备上，使用“创建链接”(**CRTLINK**) 命令来创建至该文件的符号链接。

步骤 2. 要改进安装进程的性能，对 `installAS400.jar` 文件运行“创建 Java 程序”(**CRTJVAPGM**) 命令。

步骤 3. 根据是使用 CL 提示还是使用 **qsh** 会话，输入下列命令之一:

- 如果使用 CL 提示，请输入下列命令:

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

- 如果使用 **qsh** 会话，请输入下列命令:

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

如果是使用连接的客户机来与 AS/400 机器交互，则执行下列步骤:

步骤 1. 确保满足了第29页的『使用连接的客户机』中指定的需求。

步骤 2. 确保 AS/400 机器上安装了 Host Servers 选项，并在运行中。通过在“控制语言”(CL) 提示处使用“启动 Host Servers”(**STRHOSTSVR**) 命令，可以启动 Host Servers。

步骤 3. 确保 AS/400 机器上安装了 TCP/IP，并在运行中。通过在 CL 提示处使用“启动 TCP/IP”(**STRTCP**) 命令，可以启动 TCP/IP。

步骤 4. 在工作站上，打开命令提示，并将目录改为 MQSeries Adapter Kernel 安装媒体（局域网或 CD-ROM）的 `AS400` 目录。

步骤 5. 输入下列命令:

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

其中，*machine_name* 是 AS/400 系统的 TCP/IP 地址，*user_ID* 是您的用户标识，*password* 是您的密码。

升级内核

如果已经安装了 MQSeries Adapter Kernel 版本 1.0（不论是否带有校正服务软盘 (CSD)）或带有早期修订级别的 MQSeries Adapter Kernel 版本 1.1，那么在安装带有当前修订级别的 MQSeries Adapter Kernel 版本 1.1 之前请先执行下列步骤:

步骤 1. 将 aqmsetup 和 aqmconfig（aqmconfig.properties 或 aqmconfig.xml）文件备份到 MQSeries Adapter Kernel 安装目录之外的位置。

步骤 2. 如果安装了 MQSeries Adapter Kernel CSD，如下卸载它:

- 在 Windows NT 上使用下列方法之一:
  - 从 Windows NT 的“开始”菜单，单击**程序 > MQSeries Adapter Kernel > 除去 CSD**。
  - 使用“控制面板”中的“添加／删除程序”实用程序。
  - 在内核的根目录中运行 aqmuninstallCSD.bat 文件。
  - 打开命令提示，将目录改到内核的根目录，并输入下列命令:

    ```
    java uninstallCSD
    ```
- 在 AIX 上，将目录改到内核的根目录，并输入下列命令:

  ```
  aqmuninstallCSD.sh

  java uninstallCSD
  ```

步骤 3. 如下卸载 MQSeries Adapter Kernel:

- 在 Windows NT 上使用下列方法之一:
  - 从 Windows NT 的“开始”菜单，单击**程序 > MQSeries Adapter Kernel > 卸载 MQSeries Adapter Kernel**。
  - 使用“控制面板”中的“添加／删除程序”实用程序。
  - 在内核的根目录中运行 aqmuninstall.bat 文件。
  - 打开命令提示，将目录改到内核的根目录，并输入下列命令:

    ```
    java uninstall
    ```
- 在 AIX 上，将目录改到内核的根目录，并输入下列命令:

  ```
  aqmuninstall.sh

  java uninstall
  ```

步骤 4. 安装 MQSeries Adapter Kernel 版本 1.1。有关详细信息，请参阅第34页的『安装内核』。

步骤 5. 将 aqmsetup 和 aqmconfig 文件复原到 MQSeries Adapter Kernel 安装目录中它们以前的位置。如果需要，将 aqmconfig.properties 文件转换为 aqmconfig.xml 文件。有关 aqmconfig.xml 文件的详细信息，请参阅第55页的『配置文件』。

除去内核

有几种除去内核的方法。注意，卸载进程并不除去内核安装后创建的任何文件或目录。包括所有用户复制的日志文件和数据文件。

- 在 Windows 系统上，使用下列方法之一:
  - 从“开始”菜单，单击**程序 > IBM MQSeries Adapter Kernel > 卸载 MQSeries Adapter Kernel**。
  - 使用“控制面板”中的“添加／删除程序”实用程序。
  - 在内核的根目录中运行 `aqmuninstall.bat` 文件。
  - 要安静卸载内核（即，不让卸载程序提示您详细信息或要求确认），可打开命令提示，更改到内核的安装目录，并输入下列命令:

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

- 在 UNIX 上，将目录改到内核的根目录，并输入下列命令:

  ```
  aqmuninstall.sh
  ```

  要安静卸载内核（即，不让卸载程序提示您详细信息或要求确认），更改到内核的根目录，并输入下列命令:

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

- 在 OS/400 上，使用下列卸载内核的方法之一:
  - 如果使用远程 AWT 卸载内核，请执行下列步骤:

    步骤 1. 确保设置了远程 AWT 并在运行中。有关详细信息，请参阅第28页的『使用远程 AWT』。

    步骤 2. 要改进卸载进程的性能，对 `/QIBM/ProdData/mqak/uninstall/uninstall.jar` 文件运行“创建 Java 程序”(**CRTJVAPGM**) 命令。

    步骤 3. 如下运行“运行 Java”(**RUNJVA**) 命令，其中 *n.n.n.n* 表示运行远程 AWT 的工作站的 TCP/IP 地址:

    ```
    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))
    ```

  - 如果使用连接的客户机工作站来卸载内核，请执行下列步骤:

    步骤 1. 确保满足了第29页的『使用连接的客户机』中指定的需求。

    步骤 2. 确保 AS/400 机器上安装了 Host Servers 选项，并在运行中。通过在“控制语言”(CL) 提示处使用“启动 Host Servers”(**STRHOSTSVR**) 命令，可以启动 Host Servers。

步骤 3. 确保 AS/400 机器上安装了 TCP/IP，并在运行中。通过在 CL 提示处使用“启动 TCP/IP”（**STRTCP** ）命令，可以启动 TCP/IP。

步骤 4. 将 uninstall.jar 和 uninstall.dat 文件从 AS/400 系统上的 /QIBM/ProdData/mqak/uninstall 目录复制到客户机工作站上的一个目录。

步骤 5. 输入下列命令:

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

要安静卸载内核（即，不让卸载程序提示您详细信息或要求确认），请输入下列命令:

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

其中，*machine_name* 是 AS/400 系统的 TCP/IP 地址， *user_ID* 是您的用户标识，*password* 是您的密码。

– 如果您是直接在 AS/400 系统的 CL 提示或 **qsh** 会话中操作，并且希望安静卸载内核（即，不让卸载程序提示您详细信息或要求确认），请输入下列命令:

  - 在 CL 提示上:

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

  - 在 **qsh** 会话中:

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

第4章 使用内核

本章包含有关使用内核的以下信息:

准备产品

在将内核放入产品之前，请执行以下任务:

1. 根据站点的需求和条件来设计整个系统体系结构，包括 MQSeries Adapter Offering、MQSeries 或其它消息传递软件，还可以选择使用 MQSeries Integrator。通常，体系结构对于每个站点来说都是唯一的。
2. 使用 MQSeries Adapter Builder 来构建必需的源适配器和目标适配器，然后测试并部署它们。
3. 出于以下目的开发 MQSeries Adapter Offering 之外的应用程序特定接口:
   - 允许源适配器从源应用程序获取应用程序数据
   - 允许目标应用程序从目标适配器获取消息数据

   应用程序特定接口的确切特性取决于源应用程序和目标应用程序的特性。应用程序特定接口的一些示例包括:
   - API 调用和用户出口
   - 文件读写
   - 数据库触发器
   - 消息队列
4. 配置内核以支持运行时流: 发送、路由、跟踪和传递消息。有关配置内核的信息，请参阅第50页的『配置内核』。
5. 配置 MQSeries 或其它消息传递软件（可以选择 MQSeries Integrator），以支持整个系统体系结构。请参阅第77页的『配置 MQSeries 和 MQSeries Integrator』。

6. 如果需要，可以开发 Java 登录类来支持消息传递。它们对于每个目标应用程序来说都是特定的。仅在目标应用程序需要登录以及连接到应用程序的信息时，才需要它们。
7. 在将系统放入产品之前，先测试整个系统 -- 即，MQSeries Adapter Kernel 以及源适配器和目标适配器、应用程序特定接口和定制代码。
8. 在产品环境中部署系统。
9. 通过启动一个或多个适配器守护程序（还可以选择启动跟踪服务器）来打开内核。确保源应用程序已启动。如果源适配器是在源应用程序的进程中运行的，那么源适配器是与源应用程序一起自动启动的；不需要额外的步骤来启动源适配器。任何包含源适配器的守护程序或服务器都需要启动。请参阅第78页的『启动内核』。

配置内核

本节讨论如何在您的环境中配置使用的内核。『配置概述』提供了内核配置的概念性概述。 第54页的『涉及启动和配置的文件』讨论共同定义一个 MQSeries Adapter Kernel 配置的各种文件。第55页的『设置文件』讨论 `aqmsetup` 文件，它定义内核的几个初始设置。第55页的『配置文件』讨论 `aqmconfig.xml` 文件，它为内核提供特定配置信息，如源和目标应用程序的名称、源和目标适配器、队列和队列管理器、通信方式，以及记录日志和跟踪规范。

配置概述

本节提供内核配置的概念性概述。在配置内核之前，理解内核的运行时流是很重要的。本节在一种简化层次上讨论运行时流。有关运行时流的详细信息，请参阅第11页的『运行时流』。

在最基础层上，MQSeries Adapter Kernel 的配置受流动于应用程序之间的数据驱动。配置还必须考虑下列因素:

- 接收数据的应用程序。
- 源方所需的适配器，以及目标方所需的目标适配器、适配器守护程序和工作程序。
- 使用的通信方式、编组格式和传送机制。

对于配置中的每个应用程序，数据结构和数据格式是不同的。例如，如果配置包括两个应用程序 A 和 B，每个应用程序都发送购买订单数据到应用程序 C，来自应用程序 A 的数据格式很可能不同于应用程序 B 的数据格式，并且它们的标记含义也可能不同。要使应用程序 C 无需识别和语法分析来自两个不同应用程序的两个不同数据流，可将每个应用程序的数据转换成*集成中性格式* 的*集成消息*。通常，

集成中性格式是一种基于 XML 的工业标准。应用程序 C 需要进行识别和语法分析的仅有的数据格式就是集成中性格式。

图3显示应用程序 A 和 B 的数据至应用程序 C 和 D 的流动。在该图的下面说明了它所描述的各种数据流。

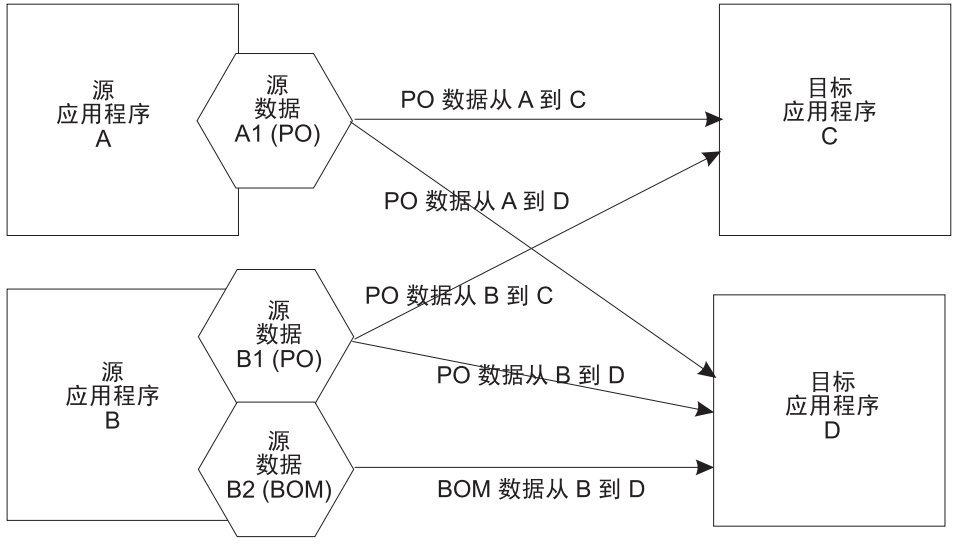


图 3. 简单配置中数据流连接的应用程序

在图3 中，应用程序 A 的购买订单数据流动到应用程序 C 和 D，应用程序 B 的购买订单数据也流动到应用程序 C 和 D，应用程序 B 的用料预计单数据只流动到应用程序 D。在每种情况中，数据在发送到目标应用程序之前都被转换成工业标准格式。在购买订单数据来自应用程序 A 和 B 的情况下，数据转换成表示购买订单数据的标准 XML 格式。在用料预计单数据来自应用程序 B 的情况下，数据转换成表示用料预计单数据的标准 XML 格式。

通信传送工具（如 MQSeries 或 “Java 消息服务 (JMS)” 的实现）用于将数据发送到目标应用程序或应用程序。集成消息转换成特定通信传送工具所需的*编组格式*，然后发送到通信传送工具（例如，MQSeries 队列）。每个目标应用程序都可以使用不同的通信传送工具和编组格式来接收消息。例如，应用程序 C 可以使用 MQSeries 来接收消息，应用程序 D 可以使用 JMS 接收消息，如第52页的图4 中所示。在这种情况中，流动到应用程序 C 的所有集成消息（即来自应用程序 A 和 B 的购买订单数据）都转换成 MQSeries 编组格式，流动到应用程序 D 的所有集成消息（即来自应用程序 A 和 B 的购买订单数据以及来自应用程序 B 的用料预计单数据）都转换成 JMS 编组格式。MQSeries Adapter Kernel 使用下列机制执行这些转换:

- *源适配器*用于将应用程序数据转换成集成消息。源适配器在 MQSeries Adapter Builder 中创建。
- *本机适配器*用于将集成消息转换成*通信消息*。本机适配器使用*逻辑消息服务* (LMS) 来转换通信传送工具传送的消息；LMS 特定于正在使用的通信传送工具。然后，LMS 使用一种*格式化器*，将消息在传送工具上进行编组。

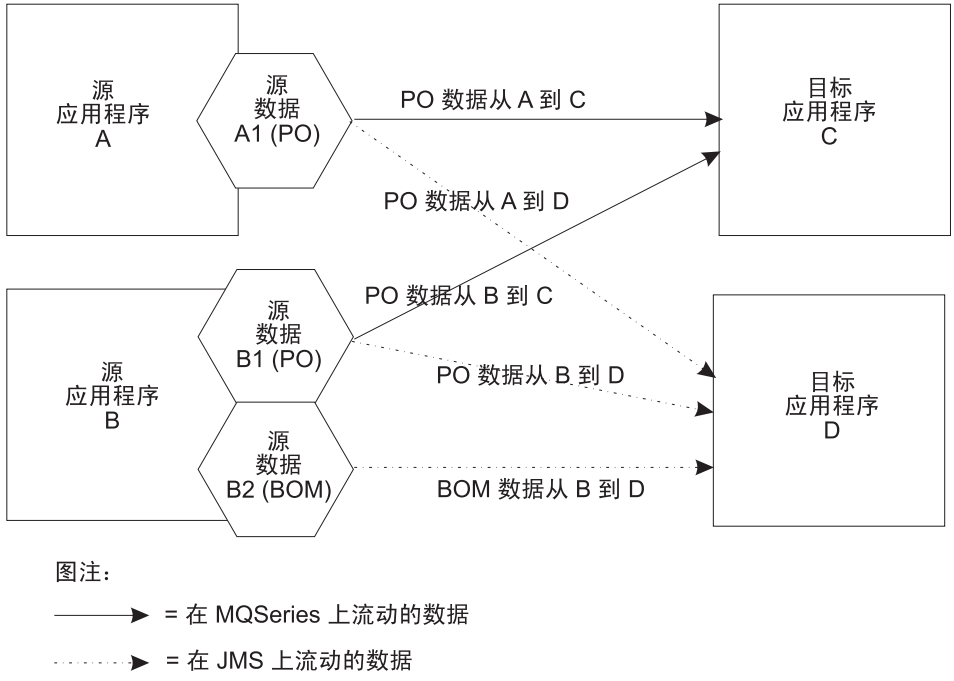


图 4. 简单配置中不同通信传送工具连接的应用程序

第53页的图5 和第53页的图6 显示数据从应用程序到集成消息再到通信消息的流动。当在目标方接收到通信消息时，转换是逆向的：本机适配器将通信消息转换回集成消息；然后，如果必需，目标适配器将集成消息转换成目标应用程序所需的数据格式。

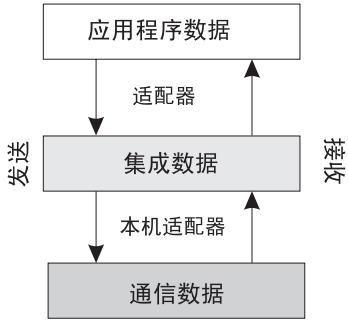


图 5. 数据的转换

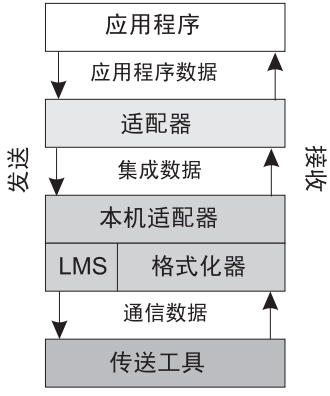


图 6. 数据的流动

必须在内核的配置文件中表示数据发送所经过的每个点。有三种由应用程序标识（源或目标）划分的逻辑配置需求。根据消息是正流入应用程序（目标）还是正从应用程序（源）流出，应用程序标识符可用于源应用程序或目标应用程序。配置文件必须包含下列类型的信息:

- 通信
  - 数据需要到达的地方
  - 要使用的通信传送工具
  - 要使用的通信编组方法（格式化器）
  - 基本通信需求，如 MQSeries 队列管理器和 MQSeries 队列名，或 JMS 队列连接工厂和 JMS 队列名
- 适配器（仅用于目标方）
  - 处理数据所需的适配器

- 使用的适配器类型（EAB 或 EJB）
- 使用的特定适配器类型的附加信息
- 对于独立 MQSeries Adapter Kernel，适配器守护程序和工作程序信息

• 其它
- 跟踪规范
- 记录日志规范

图7 显示了当数据与不同的配置部分相关时的流动。

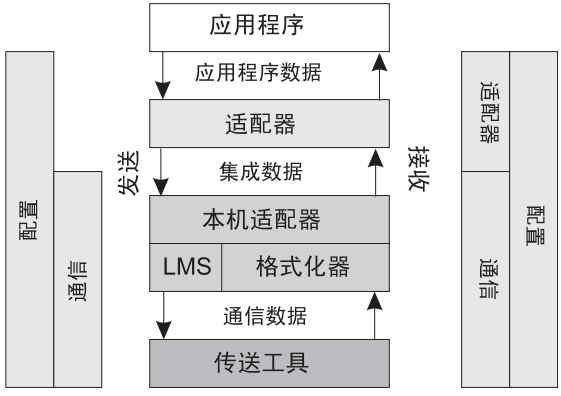


图 7. 与配置相关的数据的流动

有关将这些配置需求映射到 aqmconfig.xml 文件（该文件控制这些方面的内核配置）中的 XML 元素的详细信息，请参阅第56页的『配置文件的语法和组织』。第67页的『公共配置』列出几种公共配置。

涉及启动和配置的文件

内核配置是由几个可定制文件确定的。通过使用标准文本编辑器来编辑文件以便为您的站点配置内核。在配置内核时涉及下列文件:

• aqmsetenv.bat（Windows 系统）或 aqmsetenv.sh (UNIX) 文件，它们设置环境变量。如果需要，可以编辑这个文件以便在安装后更改系统环境变量。这个文件设置的环境变量包括 PATH、CLASSPATH 和 LIBPATH。在 Windows 系统上，这些变量是由安装程序自动设置的。要在登录到 UNIX 时自动设置这些变量，将在 aqmsetenv.sh 文件中指定的值添加到您的 .profile 文件（如果使用 Bourne shell 或 Korn shell）或 .cshrc 文件（如果使用 C shell）。

  有关在 OS/400 上设置适当环境变量的信息，请参阅第36页的6。

• aqmsetup 文件，它为内核提供一些初始设置值。有关详细信息，请参阅第55页的『设置文件』。

- aqmconfig.xml 文件，它配置内核。有关附加信息，请参阅『配置文件』。这个文件中包含了配置内核的大部分值。
- aqmcreateq.bat（Windows 系统）或 aqmcreateq.sh（UNIX 和 OS/400）文件，它们是创建 MQSeries 队列的脚本。请参阅第82页的『创建 MQSeries 队列』。

所有这些文件都包括了可帮助您编辑文件的注释。

建议您备份这些文件。有关附加信息，请参阅第80页的『维护内核』。

设置文件

设置文件 aqmsetup 控制一些内核的初始设置，包括以下内容:

- 配置文件的位置。请参阅『配置文件』。
- XML DTD 的位置（如果不在当前目录）。
- C 接口的 “Java 本机接口 (JNI)” 环境变量，用于更改使用的内存量。这适用于当 C 可执行模块启动了一个进程并且由该进程实例化了一个 Java 虚拟机时。在这种情况下可以控制内存的使用，方法是在 aqmsetup 文件中取消注释并修改下列行:

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

所有大小都以字节计。

在第109页的『附录E. 设置文件样本』中提供了一个 aqmsetup 文件样本，并且它还包括在 MQSeries Adapter Kernel 安装的 samples 目录中。

如果需要，可在首次安装 MQSeries Adapter Kernel 时编辑设置文件。在安装后，仅当内核遇到 Java 内存不足问题时才编辑该文件，就如在以上列表中所讨论的那样。

配置文件

本节讨论确定内核配置的 aqmconfig.xml 文件。 第56页的『配置文件的语法和组织』提供有关配置文件的结构信息。第74页的『编辑配置文件』提供了用于编辑配置文件的最佳方式。

MQSeries Adapter Kernel 的配置是由名为 aqmconfig.xml 的 XML 文件确定的。在第103页的『附录D. 配置文件样本』中包括了一个样本配置文件，并且它还包括在 MQSeries Adapter Kernel 安装的 samples 目录中。

配置文件中指定的值控制内核的下列元素:

- 源逻辑标识
- 目的地逻辑标识
- 目标方的适配器守护程序和工作程序信息
- 跟踪客户机
- 跟踪服务器
- 编组和路由消息，它们是由以下规范确定的:
  - 接收队列、错误队列和应答队列的名称
  - 消息发送到的一个或多个缺省目的地
  - 获取或发送消息的 MQSeries 队列管理器或 JMS 队列连接工厂的名称
  - 用于接收消息或应答的超时
  - 处理每个消息的目标方内核上的目标适配器类
  - 特定于目标适配器的附加信息
  - 目标方的最小工作程序数（如果正在运行独立 MQSeries Adapter Kernel）
  - 启用和禁用跟踪，控制跟踪级别
  - 启用和禁用审计日志
- 通信方式

配置文件的语法和组织

由于 MQSeries Adapter Kernel 的配置是基于“轻量级目录访问协议 (LDAP)”，所以配置文件的结构就反映了 LDAP。顶级 XML 元素 Epic 代表顶级 LDAP 目录，附属的 LDAP 对象由嵌套在顶级元素内的 XML 元素表示。一些 XML 元素已经需要表示 LDAP 信息的属性。值被作为元素内容或元素属性添加到配置中。作为元素内容赋值的配置值示例是 `<epictracelevel>-1</epictracelevel>`，它将值 -1（所有可能消息）赋值到 epictracelevel 元素。作为元素属性赋值的示例是配置值 `<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">`，它指定 com.ibm.logging.ConsoleHandler 类用作跟踪句柄。

以下是在配置文件中使用的高级元素的列表和描述。第59页的『在配置文件中使用的 XML 元素』列出并描述了在配置文件中使用的元素的完整集合。请参阅样本配置文件以获得有关如何在上下文中使用不同元素的样本。

- Epic -- aqmconfig.xml 文件必需的顶级元素。
- ePICApplications -- Epic 元素必需的子元素。
- ePICApplication -- ePICApplications 元素必需的子元素。它列出并定义了内核服务的应用程序；对于每个应用程序都必需一个完整定义的 ePICApplication 元素（包括子元素）。

- AdapterRouting -- ePICApplication 元素的可选子元素。它定义了队列管理器和相关信息。
- ePICBodyCategory -- AdapterRouting 元素必需的子元素。它为由内核路由的消息设置主体类别。
- ePICBodyType -- ePICBodyCategory 元素必需的子元素。它为由内核路由的消息设置主体类型。它包含对项的定义，例如消息目的地、接收消息的通信方式和消息格式化器。
- ePICAdapterDaemonExtensions -- 表示适配器守护程序的 ePICApplication 元素的可选子元素。它包含与适配器守护程序相关的信息，包括应用程序标识以及适配器工作程序数。
- ePICTraceExtensions -- 表示跟踪客户机应用程序的 ePICApplication 元素或跟踪服务器元素的可选子元素。它定义了与跟踪相关的信息。

第58页的图8 显示了配置文件的高级结构。它并不是配置文件的工作示例；只是简单地演示高级元素间的关系和相关性。如需完整示例，请参阅第103页的『附录D. 配置文件样本』。

```
<?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>
```

图 8. 配置文件的高级结构

以下是在配置文件中使用的完整元素集合的列表和描述。如果元素被注明为具有缺省值，那么在配置元素需要的值没有明确指定时，内核将使用该值。

在配置文件中使用的 XML 元素

Epic 配置文件的顶级元素

子元素:

- context
- ePICApplications（必需的）

属性: o="ePIC"（必需的）

context

当使用 “Java 消息服务 (JMS)” 对象时，指定 “Java 命名和目录接口 (JNDI)” 文件系统上下文 (FSContext) 的根目录。缺省值是当前目录。如果使用了 JMS，该元素是必需的。请参阅第89页的『使用 JMS 对象存储器』，获得有关在 MQSeries Adapter Kernel 中使用 JMS 对象的信息。

子元素: 无

属性: 无

ePICApplications

包含有关内核服务的应用程序的信息。

子元素: ePICApplication（必需的）

属性: o="ePICApplications"（必需的）

ePICApplication

指定有关内核服务的一个应用程序的信息。

子元素:

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

属性: epicappid="*application_ID*"，其中 *application_ID* 是有效的应用程序标识（必需的）

epiclogging

确定是否执行审计日志。审计日志需要 WebSphere Business Integrator 产品。缺省值是 false。

子元素: 无

属性: 无

epictrace

确定是否使用跟踪。缺省值是 false。

子元素: 无

属性: 无

epictracelevel

使用由 com.ibm.logging.IRecordType 类指定的常数来设置跟踪级别。缺省值是 0（没有消息）。请参阅 *Problem Determination Guide*，获得有关跟踪的详细信息和有效跟踪级别的完整列表。

子元素: 无

属性: 无

epictraceclientid

指定跟踪客户机应用程序的名称。缺省值是 TraceClient。

子元素: 无

属性: 无

epiclogoninfoclassname

指定当使用 EAB 适配器时用于连接到应用程序的登录类的名称。缺省值是 com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault。

子元素: 无

属性: 无

AdapterRouting

包含有关消息类型和消息路由的信息。

子元素:

- epicmqppqueuemgr
- epicuseremotequeuemanagertosend
- epicmqppqueuemgrhostname
- epicmqppqueuemgrportnumber
- epicmqppqueuemgrchannelname

- epicjmsconnectionfactoryname
- ePICBodyCategory（必需的）

属性：cn="epicadapterrouting"（必需的）

epicmqppqueuemgr

如果将 MQSeries 用作传送机制，则指定将使用的队列管理器名称。如果没有指定或指定为 DEFAULT，那么将使用缺省队列管理器。

子元素：无

属性：无

epicuseremotequeuemanagertosend

如果将 MQSeries 用作传送机制，则指定是否使用远程队列管理器来发送消息。缺省值是 false。

子元素：无

属性：无

epicmqppqueuemgrhostname

如果将 MQSeries 用作传送机制，则指定队列管理器位于的机器的 TCP/IP 主机名。仅当使用“MQSeries 客户机”时，该元素才是必需的。

子元素：无

属性：无

epicmqppqueuemgrportnumber

如果将 MQSeries 用作传送机制，则指定队列管理器的服务器进程的端口号。缺省值是 1414（MQSeries 缺省值）。仅当使用 MQSeries 客户机时，才需要该元素。

子元素：无

属性：无

epicmqppqueuemgrchannelname

如果将 MQSeries 用作传送机制，则指定队列管理器服务器的通道名称。仅当使用 MQSeries 客户机时，才需要该元素。

子元素：无

属性：无

epicjmsconnectionfactoryname

如果将 JMS 用作传送机制，则指定“JMS 连接”工厂名称。该值必须指定为 *attribute=object*，其中 *attribute* 是 LDAP 属性，*object* 是“JMS

连接”对象。对象应该存储在 AdapterRouting 元素下。例如，对于名为 QCFTEST1 的具有 LDAP 属性 cn 的 JMS 连接对象，该元素指定的值是 cn=QCFTEST1。

子元素：无

属性：无

ePICBodyCategory

指定发送消息的主体类别。

子元素：ePICBodyType（必需的）

属性：epicbodycategory=*body_category*，其中 *body_category* 指定发送消息的主体类别（必需的）

ePICBodyType

指定发送消息的主体类型。

子元素：

- 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*，其中 *body_type* 指定发送消息的主体类型（必需的）

epiccommandclassname

指定 EAB 目标适配器的名称或调用的处理消息的 EJB 命令名。如果使用适配器守护程序或 WebSphere Application Server 接收消息，则该元素是必需的。

子元素：无

属性：无

epiccommandtype

指定目标适配器的类型。可能的值包含 MQAKEAB 和 MQAKEJB。MQAKEAB 指定标准 MQSeries Adapter Kernel EAB 目标适配器； MQAKEJB 指定 Enterprise bean 用于 WebSphere Application Server 中内核的目标方。缺省值是 MQAKEAB。当目标适配器是 Enterprise Bean 时，需要值 MQAKEJB。

子元素：无

属性：无

epiccommandejbmapper

指定用于映射输入数据的 TDCMapper 类的名称。缺省值是 TDCGenericMapper。当目标适配器是 Enterprise Bean 时，需要它。

子元素：无

属性：无

epiccommandejbmethod

指定调用 Enterprise Bean 的方法名。该方法必须接受 TerminalDataContainer 对象作为输入，并返回 TerminalDataContainer 对象。缺省值是 execute。当目标适配器是 Enterprise Bean 时，需要它。

子元素：无

属性：无

epiccommandejbmethodparmtype

指定对象的类名称，它正用作在 Enterprise Bean 上调用的方法的参数。缺省值是 TDCMapper 返回的对象的类名称。当目标适配器是 Enterprise Bean 时，需要它。

子元素：无

属性：无

epiccommandejburl

以格式 IIOP://*host_name*:*port* 指定已部署的 Enterprise Bean 的统一资

源定位器 (URL)，其中，*host_name* 是 EJB 服务器的主机名，*port* 是用于名称服务器侦听的端口（缺省情况下是 900）。缺省值是 IIOP:///。当目标适配器是 Enterprise Bean 时，需要它。

子元素：无

属性：无

epiccommandejbinitialcontext

指定用于查找 Enterprise Bean 的主机名的初始上下文工厂的名称。缺省值是 com.ibm.ejs.ns.jndi.CNInitialContextFactory。当目标适配器是 Enterprise Bean 时，需要它。

子元素：无

属性：无

epicdestids

指定将用作消息目的地的一个或多个应用程序的标识。如果应用程序要发送消息并且目的地逻辑标识设置为 NONE，则该元素是必需的。

子元素：无

属性：无

epicreceivemode

指定要使用的通信方式。请参阅第87页的『附录A. 通信方式』，获得有效通信方式的列表和说明。如果应用程序要接收消息，则该元素是必需的。

子元素：无

属性：无

epicmessageformatter

指定要使用的消息格式化器，它取决于 epicreceivemode 的值和使用的传送方法。请参阅第88页的表10 和第88页的表11，获得有关消息格式化器和传送方法的详细信息。

子元素：无

属性：无

epicreceivetimeout

指定在超时之前接收方等待消息的时间量（以毫秒计）。缺省值是 0。值 -1 指定没有超时（永远等待）。

子元素：无

属性：无

epicreceivemqppqueue

指定从中接收消息的队列名称。当 epicreceivemode 元素指定了 MQSeries 通信方式时，它是必需的。请参阅第87页的『附录A. 通信方式』，获得 MQSeries 通信方式的列表。

子元素：无

属性：无

epicerrormqppqueue

指定放置错误消息的队列名称。如果使用错误消息排队并且 epicreceivemode 元素指定了 MQSeries 通信方式，那么它是必需的。请参阅第87页的『附录A. 通信方式』，获得 MQSeries 通信方式的列表。

子元素：无

属性：无

epicreplymqppqueue

指定从中接收应答消息的队列名称。如果使用应答请求并且 epicreceivemode 元素指定了 MQSeries 通信方式，那么该元素是必需的。请参阅第87页的『附录A. 通信方式』，获得 MQSeries 通信方式的列表。

子元素：无

属性：无

epicjmsreceivequeuename

指定从中接收消息的队列名称。对于 JMS 通信方式，它是必需的。对象应该存储在 ePICBodyType 元素之下。该值必须指定为 *attribute=object*，其中 *attribute* 是 LDAP 属性，*object* 是 JMS 队列对象的名称。例如，对于名为 TEST1AIQ 的具有 LDAP 属性 cn 的 JMS 对象，该元素指定的值是 cn=TEST1AIQ。

子元素：无

属性：无

epicjmserrorqueuename

指定放置错误消息的队列名称。如果使用了错误消息排队以及 JMS 通信方式，那么该元素是必需的。对象应该存储在 ePICBodyType 元素之下。该值必须指定为 *attribute=object*，其中 *attribute* 是 LDAP 属性，*object* 是 JMS 队列对象的名称。例如，对于名为 TEST1AEQ 的，具有 LDAP 属性 cn 的 JMS 对象，该元素指定的值是 cn=TEST1AEQ。

子元素：无

属性：无

epicjmsreplyqueuename

指定从中接收应答消息的队列名称。如果使用了应答请求以及 JMS 通信方式，那么该元素是必需的。对象应该存储在 ePICBodyType 元素之下。该值必须指定为 *attribute=object*，其中 *attribute* 是 LDAP 属性，*object* 是 JMS 队列对象的名称。例如，对于名为 TEST1RPL 的，具有 LDAP 属性 cn 的 JMS 对象，该元素指定的值是 cn=TEST1RPL。

子元素：无

属性：无

epicreceivefiledir

指定从中接收消息的目录名称。对于 FILE 通信方式，它是必需的。

子元素：无

属性：无

epiccommitfiledir

指定保存接收到的消息直到它们被提交为止的目录名称。当接收消息时，对于 FILE 通信方式，它是必需的。

子元素：无

属性：无

epicerrorfiledir

指定放置错误消息的目录名称。如果使用了错误消息排队以及 FILE 通信方式，那么该元素是必需的。

子元素：无

属性：无

ePICAdapterDaemonExtensions

包含有关适配器守护程序扩展的信息。

子元素：

- epicdepappid
- epicminworkers

属性：cn="epicappextensions"（必需的）

ePICTraceExtensions

包含有关跟踪扩展的信息。请参阅 *Problem Determination Guide*，获得有关该元素及其子元素的完整论述。

子元素：

- epicdepappid

- epictracesyncoperation
- epictracemessagefile
- epictracehandler
- ePICTraceHandler

属性: cn="epicappextensions"（必需的）

epicdepappid

指定适配器守护程序正在服务的应用程序标识。缺省值是启动适配器守护程序时使用的应用程序标识。

子元素: 无

属性: 无

epicminworkers

指定由适配器守护程序启动的适配器工作程序数。缺省值是 1。

子元素: 无

属性: 无

公共配置

本节列出几个公共配置方案的配置值，包括通过使用各种通信传送工具来发送和接收消息的值。当发送消息时，从源方和目标方获得配置值；当接收消息时，仅从目标方获得配置值。源和目标由它们各自的逻辑标识表示。这些示例假设源和目标在两个不同的机器上。如果还没有设置目标应用程序标识，则根据源的配置中的 epicdestids 元素值来确定它。

注: 配置方案列出可应用和可设置的元素配置值。请参考第59页的『在配置文件中使用的 XML 元素』中的元素列表，获取适用于该元素的任何缺省值。

MQSeries 公共配置: 本节提供当 MQSeries 用作通信传送工具时的公共配置。epicreceivemode 元素指定 MQSeries 通信方式（例如，MQPP 或 MQRFH2）。列出了下列方案:

- 第68页的表2 显示了当将消息从 MQSeries 服务器发送到另一个 MQSeries 服务器时需要设置的配置元素。
- 第68页的表3 显示了当将消息从使用远程队列管理器的 MQSeries 服务器发送到另一个 MQSeries 服务器时需要设置的配置元素。
- 第69页的表4 显示了当将消息从使用主机服务器的 MQSeries 客户机发送到 MQSeries 服务器时需要设置的配置元素。
- 第70页的表5 显示了当在 MQSeries 服务器上接收消息时需要设置的配置元素。

- 第70页的表6 显示了当在使用主机服务器的 MQSeries 客户机上接收消息时需要设置的配置元素。

表 2. 公共配置: 将消息从 MQSeries 服务器发送到另一个 MQSeries 服务器

源配置	目标配置
	epicreceivemode 元素指定 MQSeries 通信方式。
epicmqppqueuemgr 元素指定队列管理器的名称。该队列管理器必须存在于源应用程序的机器上。	
	epicreceivemqppqueue 元素指定接收队列的名称。该队列必须是目标应用程序机器上的 MQSeries 远程队列或 MQSeries 群集的一部分。
epicreplymqppqueue 元素指定应答队列的名称。该队列必须是发送方机器上的 MQSeries 本地队列或 MQSeries 群集的一部分。仅用于同步请求和应答。	
	epicmessageformatter 元素指定要使用的格式化器的名称。
epicreceivetimeout 元素指定在超时前接收方等待应答的时间。	

表 3. 公共配置: 将来自 MQSeries 服务器的消息经由远程队列管理器发送到 MQSeries 服务器

源配置	目标配置
	epicreceivemode 元素指定 MQSeries 通信方式。
epicmqppqueuemgr 元素指定队列管理器的名称。该队列管理器必须存在于源应用程序的机器上。	epicmqppqueuemgr 元素指定队列管理器的名称。该队列管理器必须存在于目标应用程序的机器上。必须指定名称; 不能使用缺省值。
epicremotequeuemanagertosend 元素指定远程队列管理器正用于发送消息。	
	epicreceivemqppqueue 元素指定接收队列的名称。该队列必须是目标应用程序机器上的 MQSeries 本地队列或 MQSeries 群集的一部分。

表 3. 公共配置: 将来自 MQSeries 服务器的消息经由远程队列管理器发送到 MQSeries 服务器 (续)

源配置	目标配置
epicreplymqppqueue 元素指定应答队列的名称。该队列必须是发送方的机器上的 MQSeries 本地队列或 MQSeries 群集的一部分。仅用于同步请求和应答。	
	epicmessageformatter 元素指定要使用的格式化器的名称。
epicreceivetimeout 元素指定在超时前接收方等待应答的时间。	

表 4. 公共配置: 将来自正在使用主机服务器的 MQSeries 客户机的消息发送到 MQSeries 服务器

源配置	目标配置
	epicreceivemode 元素指定 MQSeries 通信方式。
epicmqppqueuemgr 元素指定队列管理器的名称。该队列管理器必须存在于发送方客户机的主机机器上。	
epicmqppqueuemgrhostname 元素指定 MQSeries 服务器机器的主机名。	
epicmqppqueuemgrportnumber 元素指定服务器机器上队列管理器的服务器进程的端口号。	
epicmqppqueuemgrchannelnumber 元素指定队列管理器服务器的通道号。	
	epicreceivemqppqueue 元素指定接收队列的名称。该队列必须是目标应用程序机器上的 MQSeries 远程队列或 MQSeries 群集的一部分。
epicreplymqppqueue 元素指定应答队列的名称。该队列必须是发送方客户机的主机机器上的 MQSeries 本地队列或 MQSeries 群集的一部分。仅用于同步请求和应答。	
	epicmessageformatter 元素指定要使用的格式化器的名称。
epicreceivetimeout 元素指定在超时前接收方等待应答的时间。	

表 5. 公共配置: 接收消息的 MQSeries 服务器

源配置	目标配置
不适用。	epicreceivemode 元素指定 MQSeries 通信方式。
	epicmqppqueuemgr 元素指定队列管理器的名称。该队列管理器必须存在于目标应用程序的机器上。
	epicreceivemqppqueue 元素指定接收队列的名称。该队列必须是目标机器上的 MQSeries 本地队列。
	epicerrormqppqueue 元素指定错误队列的名称。该队列必须是目标机器上的 MQSeries 本地队列或群集的一部分。仅当使用适配器工作程序时，才需要该元素。
	epicmessageformatter 元素指定要使用的格式化器的名称。
	epicreceivetimeout 元素指定在超时前接收方等待接收消息的时间。

表 6. 公共配置: 使用主机服务器接收消息的 MQSeries 客户机

源配置	目标配置
不适用。	epicreceivemode 元素指定 MQSeries 通信方式。
	epicmqppqueuemgr 元素指定队列管理器的名称。该队列管理器必须存在于接收方的客户机主机机器上。
	epicmqppqueuemgrhostname 元素指定 MQSeries 服务器机器的主机名。
	epicmqppqueuemgrportnumber 元素指定服务器机器上队列进程的服务器进程的端口号。
	epicmqppqueuemgrchannelnumber 元素指定队列管理器服务器的通道号。
	epicreceivemqppqueue 元素指定接收队列的名称。该队列必须是接收方的客户机主机机器上的 MQSeries 本地队列。

表 6. 公共配置: 使用主机服务器接收消息的 MQSeries 客户机 (续)

源配置	目标配置
	epicerrormqppqueue 元素指定错误队列的名称。该队列必须是接收方的客户机主机机器上的 MQSeries 本地队列或群集的一部分。仅当使用适配器工作程序时，才需要该元素。
	epicmessageformatter 元素指定要使用的格式化器的名称。
	epicreceivetimeout 元素指定在超时前接收方等待接收消息的时间。

JMS 公共配置: 本节提供当 JMS 用作通信传送工具时的公共配置。epicreceivemode 元素指定 JMS。

如果正在使用 MQSeries JMS 实现，必须存在适当的 MQSeries 对象。例如，JMS 队列连接工厂必须与 MQSeries 服务器上的队列管理器相关， JMS 队列必须与 MQSeries 队列相关。 MQSeries 对象不必在配置中列出，但支持的 MQSeries 对象必须存在。

列出了下列方案:

- 表7 显示当经由 JMS 发送消息时需要设置的配置元素。
- 第72页的表8 显示当经由 JMS 接收消息时需要设置的配置元素。

表 7. 公共配置: 经由 JMS 发送消息

源配置	目标配置
	epicreceivemode 元素指定 JMS 通信方式。
epicjmsconnectionfactoryname 元素指定 JMS 队列连接工厂的名称。引用对象必须存在于配置中。	
	epicjmsreceivequeuename 元素指定 JMS 接收队列的名称。引用对象必须存在于配置中。
epicjmsreplyqueuename 元素指定 JMS 应答队列的名称。引用对象必须存在于配置中。仅用于同步请求和应答。	
	epicmessageformatter 元素指定要使用的格式化器的名称。

表 7. 公共配置: 经由 JMS 发送消息 (续)

源配置	目标配置
epicreceivetimeout 元素指定在超时前接收方等待应答的时间。	

表 8. 公共配置: 经由 JMS 接收消息

源配置	目标配置
不适用。	epicreceivemode 元素指定 JMS 通信方式。
	epicjmsconnectionfactoryname 元素指定 JMS 队列连接工厂的名称。引用对象必须存在于配置中。
	epicjmsreceivequeuename 指定 JMS 接收队列的名称。引用对象必须存在于配置中。
	epicjmserrorqueuename 元素指定 JMS 错误队列的名称。引用对象必须存在于配置中。
	epicmessageformatter 元素指定要使用的格式化器的名称。
	epicreceivetimeout 元素指定在超时前接收方等待接收消息的时间。

适配器公共配置: 本节提供当在目标方上调用适配器时的公共配置。根据目标是 Enterprise Access Builder (EAB) 目标适配器（由 MQAKEAB 的 epiccommandtype 值指定）还是 EJB 服务会话 bean 目标适配器（由 MQAKEJB 的 epiccommandtype 值指定）来使用不同的配置值。

注: EJB 服务会话 bean 目标适配器仅在 WebSphere Application Server 中受支持。

对于 MQAKEAB 的 epiccommandtype 值，为下列附加元素指定值:

- epiclogoninfoclassname
- epiccommandclassname

对于 MQAKEJB 的 epiccommandtype 值，为下列附加元素指定值:

- epiccommandclassname
- epiccommandejbmethod
- epiccommandejbmethodparmtype
- epiccommandejburl
- epiccommandejbinitialcontext

- epiccommandejbmapper

向配置添加适配器信息

当向内核配置添加新适配器时，必须在配置文件中至少添加几个规范。如需最小配置文件的示例，请参阅 aqmconfig.minimum.xml 文件。 第103页的『附录D. 配置文件样本』中显示了这个文件，并且该文件还包括在 MQSeries Adapter Kernel 安装的 samples 目录中。

以下规范表示了当添加新的适配器时必须在配置中添加的最小信息量:

- **源适配器**（发送消息）:
  - 运行源适配器的应用程序标识。
  - 缺省队列管理器。如果使用 MQSeries 作为传送机制，并且它安装和运行在与源适配器相同的机器上，那么就不需要具体地配置队列管理器。
  - 消息的目的地逻辑标识。如果所有消息都转至同一个目的地，那么使用主体类别 DEFAULT 和主体类型 DEFAULT。
  - 源适配器将消息发送至的每个目的地逻辑标识的接收队列。
- **目标适配器**（接收消息）:
  - 运行目标适配器的应用程序标识。
  - 缺省队列管理器。如果使用 MQSeries 作为传送机制，并且它安装和运行在与源适配器相同的机器上，那么就不需要具体地配置队列管理器。
  - MQSeries 的接收方式。通常这对于所有消息都是相同的；如果是这样，可使用主体类别 DEFAULT 和主体类型 DEFAULT。
  - 接收队列。如果这对于所有消息都是相同的，那么可使用主体类别 DEFAULT 和主体类型 DEFAULT。
  - 错误队列，如果目标适配器处理消息时发生错误，则使用它。通常这对于所有消息都是相同的；如果是这样，可使用主体类别 DEFAULT 和主体类型 DEFAULT。
  - 当接收到消息时调用的目标适配器类名。这对于主体类别和主体类型来说是特定的。
  - 接收超时值。建议设置适当的值来防止 CPU 的高度使用。通常这对于所有消息都是相同的；如果是这样，可使用主体类别 DEFAULT 和主体类型 DEFAULT。

  对于附加的目标适配器，如果使用了相同的接收队列，那么相同的信息是足够了。在这种情况下，唯一需要指定的不同信息是对于特定主体类别和主体类型要调用的目标适配器类名。
- **跟踪规范**:

– 跟踪是打开还是关闭。
– 跟踪级别。
– 源适配器和目标适配器的附加跟踪规范，包括跟踪目的地。缺省情况下，跟踪显示在启动内核的命令提示窗口或终端中。

编辑配置文件

使用文本编辑器或专用的 XML 编辑器编辑配置文件。在内核安装的 samples 目录中为 XML 编辑器的用户提供了名为 aqmconfig.dtd 的 DTD 文件。可以从 IBM alphaWorks Web 站点 www.alphaworks.ibm.com 下载名为 Xeena 的 XML 编辑器。以下建议适用于编辑配置文件:

- 在开始编辑配置文件之前，收集所有期望配置的相关信息。这包括在配置中涉及的应用程序名和队列名、交换的消息类型、使用的通信方式以及跟踪程序和其它扩展的有关信息。
- 将样本文件 aqmconfig.xml 从 samples 目录复制到期望的位置。不要重命名配置文件的副本。编辑这个副本。
- 使用注释来标识出配置文件中的不同部分，并使用注释为配置中使用的特定值（例如应用程序标识、消息队列名称和超时值）编写文档。在 XML 中，注释以字符 <!-- 开始，以字符 --> 结束。注释可以跨越多行，例如:

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

  注意，XML 不允许在其它注释中嵌套注释。
- 根据应用程序标识组织配置文件。将每个应用程序标识的各项放在一起。
- 如果没有使用专用的 XML 编辑器，那么请使用一个在保存文件时会保留行结束符并且不折断行的文本编辑器。这种文本编辑器的示例有 Windows 系统上的文本编辑器以及 UNIX 上的 vi 或 Emacs。
- 记住 XML 是区分大小写的；在对于所有标记（元素）名称和属性使用大小写时要特别仔细。在标记中使用错误的大小写会使配置文件无效。使用专用的 XML 编辑器可以帮助防止产生大小写错误。
- 如果要对主体类别和主体类型以及还没有使用缺省值的参数使用缺省值，则必须对配置文件中的每个值都配置值 DEFAULT。如果没有这样做，则内核不使用缺省值。
- 在将配置文件放入产品之前先进行确认。请参阅第75页的『确认配置文件』。
- 对配置文件的更改在下次内核进程启动时生效。如果在更改配置文件时进程正在运行，则必须停止进程，然后重新启动以使更改生效。如果是编辑当前在产品中的配置文件，那么需要特别小心。
- 每次编辑配置文件时都进行备份。

确认配置文件

在编辑配置文件之后，要将它放入产品之前，建议您先确认它。要确认配置文件，可执行下列常规步骤:

1. 创建一个配置文件确认目录，将在其中确认并设置测试。
2. 创建一个确认 XML 消息。
3. 设置消息队列以支持确认测试。
4. 设置并执行一个配置文件确认测试，它发送一条消息并接收一条消息。
5. 检查测试结果以确定配置文件是否正确。

帮助创建确认 XML 消息的实用程序以及配置文件确认测试都是作为内核的一部分提供的。

配置文件确认测试调用 `sendMsg` 方法，并从内核源方的本机适配器发送一条确认 XML 消息至内核目标方的适配器守护程序。源适配器和目标适配器并不是必需的。但是，如果有目标适配器，那么还可以测试发送消息至目标应用程序。

请遵循以下过程。

注: 提供了几个脚本以方便您在执行过程中的使用。如果需要，可复制脚本，然后编辑副本以建立您自己的版本。如果使用 OS/400，请注意脚本的 UNIX 版本可在 **qsh** 会话中运行。可以通过在“控制语言 (CL)”提示中输入“启动 QSH (**STRQSH**)”命令来启动 **qsh** 会话。

步骤 1. 打开命令提示窗口。

步骤 2. 创建一个配置文件确认目录。将配置文件和设置文件复制到目录中。

步骤 3. 更改到确认目录。

步骤 4. 输入下列命令以创建确认 XML 消息:

- `aqmcrtmsg.bat`（Windows 系统）
- `aqmcrtmsg.sh`（UNIX 和 OS/400）

步骤 5. 显示一个选项列表。选择一个选项并按 Enter 键。为每个选项输入一个值。输入值的顺序不重要。选项的示例有 `set sourcelogicalid`、`set msgtype` 和 `set bodycategory`。必须为选项 20、21、22 和 23 输入值。可以使用选项 24 或 241 来提供消息主体数据。其它值不是必需的。

步骤 6. 输入选项 1 来创建确认 XML 文件。确认 XML 文件是在当前目录中创建的，并且名为 `EpicMessage`*nn*`.xml`，其中 *nn* 是 XML 文件的编号。

步骤 7. 输入选项 0 从确认实用程序中退出。

步骤 8. 设置适当的消息队列以支持确认。

步骤 9. 设置 AQMSETUPFILE 环境变量以临时指向确认目录中的设置文件:

- 在 Windows 系统的命令提示处，输入:

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

  其中，E:\ 代表正确的驱动器，*run_time_files* 是确认目录。

- 在 UNIX 和 OS/400 上，输入下列命令。命令示例假设您使用 Korn 外壳；如果使用另一种外壳，则相应地更改命令。

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

  其中，*root_directory* 是内核的安装目录，*run_time_files* 是确认目录。在 OS/400 上，aqmsetup 文件必须总是位于您的 IFS 主目录 (/home/*user_name*) 中。

如果需要，可编辑确认目录中的设置文件以指向要确认的配置文件。

步骤 10. 选择要测试以下哪几个方面:

- 仅内核的源方。
- 是否可以将消息一直路由到目标应用程序。这个测试需要已经有一个目标适配器。
- 跟踪。

首先测试源方，然后测试目标方。关闭适配器守护程序，仅测试源方。打开适配器守护程序，还可以测试目标方。如果当前还没有目标适配器，那么仍然可以测试适配器守护程序是否处理消息直到它试图为适当的目标适配器调用命令那一刻为止。建议您启用跟踪，特别是在还没有目标适配器时。

步骤 11. 执行确认测试。从任何目录，输入以下命令:

- 在 Windows 系统上:

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

- 在 UNIX 和 OS/400 上:

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

  其中:

  source_logical_identifier
  表示源逻辑标识。该值必须匹配在步骤第75页的5中为选项 20 输入的源逻辑标识的值。

  XML_message_file
  表示 XML 消息文件。

注: 输入以下命令可以显示用于该测试的所有选项的列表:

在 Windows 系统上:

```
aqmsndmsg.bat -?
```

在 UNIX 和 OS/400 上:

```
aqmsndmsg.sh -?
```

注意，-? 仅在 Korn shell 中有效；如果使用另一个 UNIX 外壳（如 Bourne shell 或 C shell），则在问号前使用反斜杠字符（即 -\?）。

步骤 12. 检查结果。确认消息包含正确的主体类别、主体类型和数据。

- 如果仅测试内核的源方（也就是，如果还没有启动适配器守护程序），那么检查路由消息至的队列。
  - 如果看见确认消息在该队列中，那么就确认了配置文件中的那些项。
  - 如果在该队列中没有看到确认消息，则检查异常文件。如果启用了跟踪，请检查跟踪消息。
- 如果是在测试内核的目标方并且有目标适配器，那么请检查目标应用程序。
  - 如果目标应用程序接收到了确认消息，那么就确认了在配置文件中的那些项。
  - 如果目标应用程序没有接收到确认消息，请检查异常文件。如果启用了跟踪，请检查跟踪消息。
- 如果是在测试内核的目标方并且没有目标适配器，那么请检查错误队列中的确认消息和异常文件中的异常消息。如果启用了跟踪，请检查跟踪消息。
  - 如果在错误队列中看见确认消息，并且看到了异常消息，那么就确认了配置文件中的这些项。
  - 如果没有在错误队列中看到确认消息，则检查异常文件。如果启用了跟踪，请检查跟踪消息。

步骤 13. 如果需要，可修改配置文件并再次确认。

配置 MQSeries 和 MQSeries Integrator

如下所示配置 MQSeries 和可选软件（如 MQSeries Integrator）来支持内核:

在 MQSeries 中:

- 使用几个队列来验证安装。如果将这些队列用于测试或产品环境，则清空它们以便验证安装。请参阅第40页的『验证过程』，了解在验证安装时使用的队列信息。
- 根据设计的路由方案来设置队列以支持消息传送。

- 当创建队列时，将 MAX_QUEUE_DEPTH 环境变量设置为允许的最大队列深度。

对于 MQSeries Integrator，在规则（版本 1.1）或消息流（版本 2）中设置输入和输出队列，这些队列对应于在配置文件中配置的队列。

性能建议

以下性能建议特别适用于 MQSeries Adapter Kernel:

- 当语法分析 XML DTD 时，确保 DTD 文件驻留在与语法分析它们的进程相同的目录中。这就减少了进程查找 DTD 所需的操作。
- 当发送和接收大消息时，使用消息类型 RFH2 将获得比使用消息类型 XML 更佳的性能。

请参阅 MQSeries 文档，获得有关改进性能的常规建议。

启动内核

要启动内核，需启动下列项:

- 每个目标应用程序的适配器守护程序
- 跟踪服务器（可选的）

注意，如果源适配器是在源应用程序的进程中运行的，那么源适配器是与源应用程序一起自动启动的；不需要额外的步骤来启动源适配器。任何包含源适配器的守护程序或服务器都需要启动。不直接启动源适配器。

执行下列步骤启动每个适配器守护程序和跟踪服务器:

注: 提供了几个脚本以方便您在执行过程中的使用。如果需要，可复制脚本，然后编辑副本以建立您自己的版本。如果使用 OS/400，请注意脚本的 UNIX 版本可在 **qsh** 会话中运行。可以通过在“控制语言 (CL)”提示中输入“启动 QSH (**STRQSH**)”命令来启动 **qsh** 会话。

步骤 1. 启动 MQSeries 或其它消息传递软件以及可选软件，如 MQSeries Integrator。

步骤 2. 启动您站点所需的任何相关软件（例如，内核之外的应用程序）以便从队列中读取跟踪消息。

步骤 3. 打开命令提示。对于每个适配器守护程序，输入以下命令:

- 在 Windows 系统上:

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

- 在 UNIX 和 OS/400 上:

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

  其中

  -a *application_identifier*
  标识适配器守护程序服务的目的地逻辑标识。

  -bc *body_category*
  指定主体类别，适配器守护程序工作程序使用它来确定用于接收消息的通信方式和相关信息。如果没有提供值，适配器守护程序使用值 DEFAULT。

  -bt *body_type*
  指定主体类型，适配器守护程序工作程序使用它来确定用于接收消息的通信方式和相关信息。如果没有提供值，适配器守护程序使用值 DEFAULT。

  -noretry
  指定当没有其它消息时，工作程序自动停止。如果没有指定 -noretry，那么工作程序会不停地轮询队列中是否有消息，并且必须手工停止适配器守护程序。

  注: 如果需要修改 Java 启动参数，可编辑 aqmstrad.bat（Windows 系统）或 aqmstrad.sh（UNIX 和 OS/400）文件。详细信息请参阅文件内的注释。

步骤 4. 对于每个跟踪服务器，输入以下命令:

- 在 Windows 系统上:

  ```
  aqmstrtd.bat -how -a source_application_identifier
  ```

- 在 UNIX 和 OS/400 上:

  ```
  aqmstrtd.sh -how -a source_application_identifier
  ```

  其中:

  -how
  表示如何接收跟踪消息。可能的值包括:

  - socket
  - ena，即本机适配器

  -a *source_application_identifier*
  源应用程序标识。如果没有提供值，将使用配置文件中的缺省值 TraceServer。

请参阅 *Problem Determination Guide*，获得有关跟踪服务器的详细信息。

步骤 5. 在启动了适配器守护程序或跟踪服务器后，进程窗口将保持打开直到停止了适配器守护程序为止。进程窗口可显示异常。请参阅第81页的『异常消息』。

停止内核

要停止内核，需停止每个适配器守护程序和跟踪服务器。停止它们有以下几种方法:

- 当启动适配器守护程序时，设置参数 `-noretry`。请参阅第78页的『启动内核』。
- 转至启动适配器守护程序或跟踪服务器的命令提示（Windows 系统）或终端 (UNIX)，按 **Ctrl-C**。对于每个适配器守护程序或跟踪服务器执行这个步骤。
- 在 Windows 系统上，可以使用“任务管理器”来终止进程。
- 在 UNIX 上，可以使用 **ps** 命令来确定进程数，然后使用 **kill** 命令终止进程。

维护内核

建立内核维护计划。建议您定期备份下列项:

- 在下列文件中指定的配置:
  - `aqmconfig.xml`
  - `aqmsetup`
- 构建的适配器及其相关文件。

为支持内核自己的处理而备份或定期删除由内核使用的跟踪文件或其它文件中的内容，并不是一定要执行的。如果需要，可以备份这些文件。如果将跟踪消息路由到单个文件而不是多个文件，那么单个跟踪文件可能会变得非常大。如果将跟踪级别设置为高级别的详细信息（例如，所有跟踪消息或信息性消息），那么可以考虑定期删除跟踪文件。

诊断问题

可以使用异常消息、跟踪消息和 MQSeries 错误队列来帮助诊断问题。MQSeries Adapter Kernel 产生异常消息，并且如果启用了跟踪，还产生跟踪消息。请参阅 *Problem Determination Guide*，获得有关如何在 MQSeries Adapter Kernel 环境中诊断问题的信息。

要理解异常消息和跟踪消息，必须了解内核是如何工作的。内核使用一个错误队列来处理一些错误。请参阅第7页的『内核如何工作』。

可以通过唯一消息标识和唯一事务标识的组合来标识引起异常消息和跟踪消息的那些消息。

没有标识可让您明确地标识出在错误队列和内核两者中的相同消息。但是，可以手工将错误队列中的消息与相应的异常消息或跟踪消息（或两者）相关联。可以比较以下内容中的一个或多个:

- 大概的时间戳记
- 用于源逻辑标识的队列
- 用于目的地逻辑标识的队列
- 主体类别
- 主体类型
- 唯一消息标识
- 唯一事务标识

如果它们匹配，那么就很可能已经将错误队列中的消息与相应的异常消息或跟踪消息相关联。

版本号

运行 bin 目录中的 aqmversion.bat（Windows 系统）或 aqmversion.sh（UNIX 和 OS/400）以显示内核的版本号。

异常消息

内核产生以下几种类型的异常消息:

- 内核源方的本机适配器将异常发往源适配器。请参阅 MQSeries Adapter Builder 文档，获得有关源适配器如何处理这些异常的信息。
- 内核目标方的本机适配器将异常发往管理本机适配器的工作程序。
- 工作程序将异常写入 EpicSystemExceptionFile*nnnnnnnn*.log 文件，它位于与工作程序相同的目录中。
- 适配器守护程序将异常消息写入名为 EpicSystemExceptionFile*nnnnnnnn*.log 的异常文件中，它位于与适配器守护程序相同的目录中。由于适配器守护程序和它的工作程序位于相同目录，所以它们都写入同一个异常文件中。适配器守护程序还将异常消息写入控制台（如果它是从窗口启动的，那么这就是用于启动它的命令提示或终端）。

内核的跟踪异常消息与 MQSeries 异常消息不同。以下是来自内核的异常消息示例:

```
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)>
```

异常消息中的值取决于消息实质，可能会包含下列项:

- 时间戳记
- 源逻辑标识
- 目的地逻辑标识
- 主体类别
- 主体类型
- 唯一消息标识
- 唯一事务标识
- 异常信息

有关验证安装期间会遇到的常见问题和可能响应，请参阅第41页的『常见验证问题』。

跟踪消息

可以配置内核来产生跟踪消息。有关跟踪的信息，请参阅 *Problem Determination Guide*。

实用程序

创建 MQSeries 队列

可以使用批处理文件或外壳脚本来自动创建 MQSeries 队列。运行 `aqmcreateq.bat`（Windows 系统）或 `aqmcreateq.sh`（UNIX 和 OS/400），使用应用程序名作为参数。这些文件为每个应用程序创建下列队列:

- 接收队列，名为 *application_name*AIQ。
- 错误队列，名为 *application_name*AEQ。
- 应答队列，名为 *application_ name*RPL。

第5章 使用 MQSeries Adapter Kernel API

内核包含了用于诸如发送和接收消息、创建和语法分析 XML，以及管理内核配置等功能的 API。这些 API 是由 MQSeries Adapter Builder 创建的适配器使用的。“MQSeries Adapter Kernel 信息中心”包含了 Javadoc HTML 格式的相关联机 API 文档。

设计的内核是与适配器一起使用的，这些适配器是用户使用 MQSeries Adapter Builder 构建的。内核并不设计为在仅从定制代码调用内核 API 时使用。提供的联机 API 文档仅帮助理解内核是如何工作的并帮助诊断。

内核联机 API 文档位于 `documentation` 目录中。

第6章 获得附加信息

在使用 MQSeries Adapter Offering 时，有好几种有用的信息源。有关 MQSeries Adapter Kernel 的附加信息，请参阅 *Problem Determination Guide* 文档，该文档可从与产品一起安装的 “MQSeries Adapter Kernel 信息中心” 中获得。*Problem Determination Guide* 提供了解决在使用内核时可能产生的特定问题的有关信息。有关 MQSeries Adapter Builder 的信息，请参阅产品的 “信息中心” 和联机帮助系统。

因特网上的可用信息

MQSeries 产品系列 Web 站点是 www.ibm.com/software/ts/mqseries/。通过使用以下 Web 站点中的链接，您可以:

- 获得有关 MQSeries 产品系列的最新信息，包括 MQSeries Adapter Offering。
- 访问 HTML 和 PDF 格式的 MQSeries 书籍，可能包括本书的更新版本。MQSeries 书籍库页面的直接链接是 www.ibm.com/software/ts/mqseries/library/manualsa/。
- 下载 MQSeries SupportPac。

有关在 OS/400 上使用 MQSeries 的信息，请参阅位于 www.ibm.com/servers/eserver/iseries/library/ 上的 OS/400 书籍库。还可以从 Web 站点 www.ibm.com/software/ts/mqseries/library/manualsa/ 上的 MQSeries 书籍库中参阅 OS/400 特定书籍。

参考

下列参考资料讨论了本书中涉及的主题:

- “开放式应用程序组” Web 站点是 www.openapplications.org/
- “可扩展标记语言 (XML) 1.0 W3C 建议书” 位于 www.w3.org/TR/1998/Rec-xml-19980210 上

这些不是 IBM Web 站点。

附录A. 通信方式

本附录提供了有关 MQSeries Adapter Kernel 支持的通信方式以及用于支持它们的 Java 类的有关信息。提供了一些通信方式作为便捷方式并提供了缺省格式化器。请参阅第88页的表10, 了解与便捷方式一起使用的缺省格式化器。

支持下列通信方式:

MQPP 内核使用 MQSeries 基本服务传送消息。这是便捷方式。

MQRFH1 内核使用 MQSeries 传送消息, 使用 MQSeries Integrator 版本 1.1 代理消息。这是便捷方式。

MQRFH2 内核使用 MQSeries 传送消息, 使用 MQSeries Integrator 版本 2 代理消息。这是便捷方式。

MQBD 内核使用 MQSeries 基本服务传送消息, 但仅发送和接收主体数据。这是便捷方式。下列特征是这种方式所独有的:

- 它只能发送主体数据, 而不发送消息头值。
- 它能接收仅包含主体数据的消息。它对于接收到的消息使用下列缺省的消息头值:
  - `SourceLogicalApplicationID`--在接收方法调用中使用的 `ENAService` 对象中的值。
  - `BodyCategory`--在接收方法调用中使用的 `ENAService` 对象中的值。
  - `BodyType`--在接收方法调用中使用的 `ENAService` 对象的值。
  - `Acknowledgment`--如果接收到的 MQMessage 是 MQSeries REQUEST, 那么将 `Acknowledgment` 设置为 1。
  - `BodyData`--从 MQSeries 接收到的消息数据。

  所有其它头值都使用正常的缺省值。

MQ 内核使用 MQSeries 基本服务传送消息。

JMS 内核使用 “Java 消息服务 (JMS)” 传送消息。请参阅第89页的『使用 JMS 对象存储器』, 获得有关使用 JMS 对象与 MQSeries Adapter Kernel 的信息。

FILE 内核将消息放入文件并从文件中取出消息。这种方式仅为诊断目的提供。

表9列出了通信方式和支持它们的 Java 类。所有 Java 类都来自 Java 包 com.ibm.epic.adapters.eak.nativeadapter。注意，任何支持逻辑消息服务 (LMS) 的 Java 类都可指定为通信方式；在这种情况下，使用这个类本身来支持通信。

表 9. 通信方式和支持的 Java 类

通信方式	Java 类	备注
MQPP	LMSMQBindingMQPP	需要安装 MQSeries
MQRFH1	LMSMQBindingMQRFH1	需要安装 MQSeries
MQRFH2	LMSMQBindingMQRFH2	需要安装 MQSeries
MQBD	LMSMQMQBD	需要安装 MQSeries
MQ	LMSMQBinding	需要安装 MQSeries
JMS	LMSJMS	需要安装 JMS
FILE	LMSFile	无

表10列出通信方式及其相关的格式化器接口。表11交叉引用了格式化器接口、格式化器类名及其使用。所有格式化器类都来自 Java 包 com.ibm.epic.adapters.eak.nativeadapter。注意，可以为通信方式指定任何格式化器类；在这种情况下，指定的格式化器类被用作格式化器。

表 10. 通信方式和格式化器接口

通信方式	格式化器接口	缺省格式化器
MQPP	MQFormatterInterface	MQNMXMLFormatter
MQRFH1	MQFormatterInterface	MQNMRFH1Formatter
MQRFH2	MQFormatterInterface	MQNMRFH2Formatter
MQBD	MQFormatterInterface	MQNMBDFormatter
MQ	MQFormatterInterface	MQNMXMLFormatter
JMS	JMSFormatterInterface	JMSNMRFH2Formatter
FILE	StringFormatterInterface	NMXMLFormatter

表 11. 格式化器接口、格式化器类名和用途

格式化器接口	格式化器类名	用途
MQFormatterInterface	MQNMXMLFormatter	EpicMessage 作为 XML
	MQNMRFH1Formatter	EpicMessage 作为 RFH1
	MQNMRFH2Formatter	EpicMessage 作为 RFH2
	MQNMDBFormatter	仅主体数据

表 11. 格式化器接口、格式化器类名和用途 (续)

格式化器接口	格式化器类名	用途
JMSFormatterInterface	JMSNMXMLFormatter	EpicMessage 作为 XML
	JMSNMRFH2Formatter	EpicMessage 作为 RFH2
	JMSNMBDFormatter	仅主体数据
StringFormatterInterface	NMXMLFormatter	EpicMessage 作为 XML

表12 列出了支持的 LMS 类，以及它们对事务性支持的程度。请参阅第22页的『事务性能力』，获得有关在 MQSeries Adapter Kernel 中使用事务的信息。

表 12. LMS 类和事务性支持

LMS 类	事务性支持
LMSMQBindingMQPP	单阶段
LMSMQBindingMQRFH1	单阶段
LMSMQBindingMQRFH2	单阶段
LMSMQMQBD	单阶段
LMSMQBinding	单阶段
LMSJMS	单阶段
LMSFILE	不支持

使用 JMS 对象存储器

JMS 对象名称是使用 JNDI 的 FSContext 文件实现（它是 MQSeries JMS SupportPac 的一部分）存储的。通过对目录名使用带有相关值的专有属性，内核用于 FSContext 的上下文（目录结构）遵循 LDAP 层次结构。例如，对于 LDAP 层次结构 o=ePIC，o=ePICApplications，epicappid=TEST1，目录结构是 o-ePIC/o-ePICApplications/epicappid-TEST1。

要创建上下文和对象，可使用与 JMS 安装一起提供的 JMS 管理工具。基本步骤是定义一个上下文，然后更改上下文。更改上下文将使您进入上下文。在适当的地方创建 JMS 对象。以下是创建上下文结构和 JMS 对象的命令示例。在本例中，应用程序标识是 TEST1。

```
#
# 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
```

附录B. 已确认的配置

对于 MQSeries、MQSeries Adapter Offering 和 MQSeries Integrator，存在很多可能的配置和组合。这些 MQSeries 产品系列成员中的每一个都有丰富的功能与配置。而且还可以组合 MQSeries、MQSeries Adapter Offering 和 MQSeries Integrator 中的功能。一个 MQSeries 产品系列成员中的某些功能可能会与系列中其它成员提供的功能存在部分重叠。您必须确定如何使用和组合 MQSeries、MQSeries Adapter Offering 和 MQSeries Integrator 中不同的消息路由和传递功能。

MQSeries、MQSeries Adapter Offering 和 MQSeries Integrator 的下列配置在本书出版时已经过了确认。请参考 MQSeries Web 站点获得最新的确认过的配置。

MQSeries Adapter Kernel:

- 发送请求确认的消息以及不请求确认的消息。
- 使用 MQSeries 或 JMS 通信方式。如需有效的通信方式，请参阅第87页的『附录A. 通信方式』。
- 消息路由和传递:
  - 从一个源适配器发送消息到一个目标适配器
  - 从一个源适配器发送消息到多个目标适配器
  - 多线程消息传递，即，多个工作程序
  - 将消息中的目的地逻辑标识设置为 NONE，就可以使用内核的配置文件来基于主体类别、主体类型和源逻辑标识确定目的地逻辑标识
  - 传递的“推送”模型
  - 启用跟踪

  注: 请参阅第95页的『附录C. 消息头』。它包含内核填充和处理的 MQSeries Adapter Kernel 消息头字段。
- 具有在第25页的『硬件』和第26页的『软件』上显示的先决条件。
- 使用配置文件而不是 LDAP 来定义配置。

MQSeries:

- 不使用 MQSeries 群集。

  注: 请参阅第95页的『附录C. 消息头』。它包含内核填充和处理的 MQSeries Adapter Kernel 消息头字段。

MQSeries Integrator:

- MQSeries Adapter Kernel 和 MQSeries 可将消息路由并发送到 MQSeries Integrator。请参阅 MQSeries Integrator 信息，以确定它代理这些消息的能力。
- 通过 MQSeries 和 MQSeries Integrator 版本 2 从内核源方发送消息，并直接路由到内核的目标方。在 MQSeries Integrator 内部，消息流被配置为静态路由。所有到达流的 MQInput 节点的消息都被直接路由到特定的 MQOutput 队列。

  注： 请参阅第95页的『附录C. 消息头』。它包含内核填充和处理的 MQSeries Adapter Kernel 消息头字段。

附录C. 消息头

MQSeries Adapter Offering 使用几种消息头。请参阅第10页的『消息和消息格式』，了解在哪种情况下使用哪种消息头。

本附录列出并描述了消息头字段。

MQSeries Adapter Kernel 消息描述符头

MQSeries Adapter Kernel 使用头值。这些值放在消息存放器对象中。**在应答中传播?** 一栏列出了当源应用程序请求应答时，该值是否在应答消息中传回给源应用程序。某些值仅与 WebSphere Business Integrator 一起使用。

表 13. MQSeries Adapter Kernel 头

头名称	在应答中传播?	含义或用法
UniqueID	否	每个消息的唯一标识。
TransactionID	是	每个消息及其应答（如果有的话）共享的事务标识。等价于 Extricity PublicProcessID 或 DataInterchange (DI) ApplicationID。
MessageType	否	用于网关和“日志/跟踪/异常”消息。
SourceLogicalID	否	源应用程序的逻辑标识。等价于 DI、“伙伴协议管理器 Manager (PAM)”和“商业流程管理器 (BFM)”中的保留名。
DestinationLogicalID	否	目标应用程序的逻辑标识。对于 DI 和 PAM，缺省值是无，但是可以覆盖。
RespondToLogicalID	是	应答消息将被发送到的逻辑标识。复制到 DI 的 DestinationLogicalID 和 PAM 的 SourceLogicalID。
CorrelationID	否	保留使用。
GroupStatus	否	保留使用。

表 13. MQSeries Adapter Kernel 头 (续)

头名称	在应答中传播?	含义或用法
ProcessingCategory	否	等价于 “PAM 公共进程标识” 或 “DI 命令进程标识”。
QosPolicy	否	保留使用。
DeliveryCategory	否	等价于 DI RequestorProfileID。
AckRequested	否	确定源应用程序是否请求应答消息。
PublicationTopic	否	保留使用。
SessionID	否	等价于 DI BatchID。
EncryptionStatus	否	确定主体加密类型和签名。
TimeStampCreated	否	创建消息的时间与日期。
TimeStampExpired	否	在此时间与日期之后，消息不再有意义。值 -1 表示永不到期。
Size	否	保留使用。
BodyType	否	表示消息的特定目的。
BodyCategory	否	表示消息的应用程序类型。
BodySecondaryType	否	保留使用。
UserArea	否	用户数据的常规区域。
RelatedSubjectID	否	用于进程间相关。
ExternalID	否	应用程序环境之外的当前所有者（例如，一个用户或贸易伙伴）标识。
InternalID	否	应用程序环境内部的当前所有者（例如，一个用户或贸易伙伴）标识。
BodySignature	否	保留使用。
TransportCorrelationID	是	保留使用。

MQSeries 消息描述符头

字段内容是由 MQSeries 确定的。MQSeries Adapter Offering 将消息放入由消息控制值确定的队列中。有关详细信息，请参阅第13页的『消息控制值』。

表 14. MQSeries 头

节或字段	含义或用法
Revision	固定的。
UniqueID	每条消息有一个唯一标识。
TransactionID	消息及其应答共享相同的事务标识。
MessageType	保留使用。
SourceLogicalID	源应用程序的逻辑标识。
DestinationLogicalID	目标应用程序的逻辑标识。
RespondToLogicalID	应答消息将被发送到的逻辑标识。
CorrelationID	保留使用。
GroupStatus	保留使用。
ProcessingCategory	保留使用。
QosPolicy	保留使用。
DeliveryCategory	保留使用。
AckRequested	确定源应用程序是否请求应答。
PublicationTopic	保留使用。
SessionID	保留使用。
EncryptionStatus	保留使用。
TimeStampCreated	创建消息的时间与日期。
TimeStampExpired	在此时间与日期之后，消息不再有意义。
Size	保留使用。
BodyCategory	表示消息的应用程序类型，例如 OAG 或 RosettaNet。
BodyType	表示消息的特定目的，例如添加销售订单或同步化库存。
BodySecondaryType	保留的。
UserArea	用户数据的常规区域。
BodyData	消息主体数据。

没有 MQSeries Integrator 的 MQSeries

内核头值和主体数据都被放入一个 XML 文档中。以下是描述 XML 文档的 DTD 示例:

```
<!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)> <!-- The data will be escaped -->
<!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 版本 1 头

MQSeries Integrator 版本 1 头 RFH1 包含下列项:

1. 固定部分
2. Neon 头
3. 数据节，包含内核头和消息主体数据

表 15. MQSeries Integrator 版本 1 头 -- RFH1

节或字段	含义或用法

表 15. MQSeries Integrator 版本 1 头 -- RFH1 (续)

固定部分	按照 MQSeries Integrator 版本 1.1 指定的使用。
Neon 头	遵循 Neon 头格式。
OPT_APP_GRP	SourceLogicalId 值。取自内核头。
OPT_MSG_TYPE	BodyCategory+BodyType。从内核头派生。 例如: 如果 BodyCategory 是 OAG 并且 BodyType 是 SyncItem，那么该值是 OAG+SyncItem .
数据部分	由内核头值以及后跟的消息主体数据组成。
内核头	内核头包括在标记 <EPICHEADER>*header*</EPICHEADER> 之中。 内核头值使用 XML 语法。仅存在具有值的属性。实际数据不能在分开的行上。值的格式示例是: <MessageType>*value*</MessageType>。
MessageType	保留使用。
SourceLogicalID	源应用程序的逻辑标识。
DestinationLogicalID	目标应用程序的逻辑标识。
RespondToLogicalID	应答消息将被发送到的逻辑标识。
TimeStampCreated	创建消息的时间与日期。
TimeStampExpired	在此时间与日期之后，消息不再有意义。
TransactionID	消息及其应答共享相同的事务标识。
UniqueID	每条消息有一个唯一标识。
AckRequested	确定源应用程序是否请求应答。
ProcessingCategory	保留的。
BodyCategory	表示消息的应用程序类型，例如 OAG 或 RosettaNet。
BodyType	表示消息的特定目的，例如添加销售订单或同步化库存。
BodySecondaryType	保留的。
UserArea	用户集成特定的应用程序数据。
MsgHeaderVersion	内核头版本（保留的）。
CorrelationID	用户集成特定的。
GroupStatus	用户集成特定的。
QosPolicy	保留的。
DeliveryCategory	保留的。
PublicationTopic	保留的。

表 15. MQSeries Integrator 版本 1 头 -- RFH1 (续)

SessionID	保留的。
EncryptionStatus	保留的。
消息主体数据	消息主体数据。

MQSeries Integrator 版本 2 头

MQSeries Integrator 版本 2 头 RFH2 包含下列项:

1. 固定部分
2. <mcd> 文件夹 -- 消息内容描述符
3. <usr> 文件夹 -- 应用程序（用户）定义的特性
4. 数据节，包含内核头和消息主体数据

表 16. MQSeries Integrator 版本 2 头 -- RFH2

节或字段	含义或用法
固定部分	按照 MQSeries Integrator 版本 2 指定的使用。
<mcd>	如果消息是 XML，则是 XML。遵循 MQSeries Integrator 版本 2 规则。
set	内核不使用。
type	内核不使用。
format	如果消息是 XML，则是 XML。遵循 MQSeries Integrator 版本 2 规则。
<usr> 文件夹 -- 应用程序（用户）定义的特性	由内核头值组成。
内核头	仅存在具有值的属性。实际数据不能在分开的行上。
SourceLogicalID	源应用程序的逻辑标识。
DestinationLogicalID	目标应用程序的逻辑标识。
MessageType	保留使用。
RespondToLogicalID	应答消息将被发送到的逻辑标识。
TimeStampCreated	创建消息的时间与日期。
TimeStampExpired	在此时间与日期之后，消息不再有意义。
TransactionID	消息及其应答共享相同的事务标识。
UniqueID	每条消息有一个唯一标识。
ProcessingCategory	保留的。

表 16. MQSeries Integrator 版本 2 头 -- RFH2 (续)

BodyCategory	表示消息的应用程序类型，例如 OAG 或 RosettaNet。
BodyType	表示消息的特定目的，例如添加销售订单或同步化库存。
BodySecondaryType	保留的。
AckRequested	确定源应用程序是否请求应答。
UserArea	用户集成特定的应用程序数据。
MsgHeaderVersion	内核头版本（保留的）。
CorrelationID	用户集成特定的。
GroupStatus	用户集成特定的。
QosPolicy	保留的。
DeliveryCategory	保留的。
PublicationTopic	保留的。
SessionID	保留的。
EncryptionStatus	保留的。
数据部分	消息主体数据。

附录D. 配置文件样本

本节列出了编写本出版物时 aqmconfig.xml 文件的版本。第107页的『最小配置文件样本』 列出了编写本出版物时 aqmconfig.minimum.xml 文件的版本。请参阅位于内核安装的 samples 目录中的 aqmconfig.xml 和 aqmconfig.minimum.xml 文件以获得最新版本；这里列出的示例很可能已经过时。

请参阅第55页的『配置文件』，获得有关解释和编辑配置文件的信息。

在这个配置文件示例中包含了几个应用程序标识。在每个应用程序标识下列出了一组项。配置文件样本中包含下列应用程序标识:

- 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. -->
<!--  -->
<!-- CopyrightVersion 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. -->
          <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
          <!-- 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 -->
          <!--
          <epicmqpppqueuemgrhostname>localhost</epicmqpppqueuemgrhostname>
   -->
          <!-- MQSeries Client port to use for where the MQSeries server -->
          <!-- resides for TEST1.  No entry defaults to MQSeries default -->
          <!--
          <epicmqpppqueuemgrportnumber>1414</epicmqpppqueuemgrportnumber>
   -->
          <!-- MQSeries Client channel name to use for the MQSeries server, required -->
          <!--
          <epicmqpppqueuemgrchannelname>xyz</epicmqpppqueuemgrchannelname>
   -->
          <!-- 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 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">
               <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
                     <epiccommandclassname>com.ibm.epic.adapters.eak.test.InstallVerificationTest
                     </epiccommandclassname>
                     <epicreceivemode>MQPP</epicreceivemode>
                     <epicreceivemqpppqueue>TEST2AIQ</epicreceivemqpppqueue>
                     <epicerrormqpppqueue>TEST2AEQ</epicerrormqpppqueue>
                     <epicreplymqpppqueue>TEST2RPL</epicreplymqpppqueue>
                  </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">
               <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
                     <epicdestids>TEST1</epicdestids>
                     <epicreceivemode>MQPP</epicreceivemode>
                     <epicreceivemqpppqueue>TEST3AIQ</epicreceivemqpppqueue>
                  </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">
              <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
              <ePICBodyCategory epicbodycategory="DEFAULT">
                 <ePICBodyType epicbodytype="DEFAULT">
                    <epicreceivemode>MQPP</epicreceivemode>
                    <epicreceivemqppqueue>TraceServerAIQ</epicreceivemqppqueue>
                 </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>
```

最小配置文件样本

本节提供了用于 MQSeries Adapter Kernel 的最小配置文件的示例。请参阅第73页的『向配置添加适配器信息』，获得有关最小配置文件的信息。

```
<?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">
              <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
              <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">
               <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
               <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>
                     <epicreceivemqpppqueue>TEST2AIQ</epicreceivemqpppqueue>
                     <epicerrormqpppqueue>TEST2AEQ</epicerrormqpppqueue>
                     <epicreplymqpppqueue>TEST2RPL</epicreplymqpppqueue>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
      </ePICApplications>
   </Epic>
```

附录E. 设置文件样本

以下是 aqmsetup 文件的示例，它定义了几个内核初始配置值，包括几个环境变量。请参阅第55页的『设置文件』获得有关此文件的附加信息。aqmsetup 文件位于内核安装根目录的 samples 目录中。

```
#
# 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
```

声明

本信息是为在美国提供的产品和服务而开发的。IBM 可能未在其它国家提供本信息中所讨论的产品、服务或功能。请咨询本地的 IBM 代理以获得您所在区域当前可用的产品和服务的信息。所有对 IBM 产品、程序或服务的引用并不明示或暗示只可以使用 IBM 的产品、程序或服务。任何不侵犯 IBM 知识产权的具有相同功能的产品、程序或服务都可以代替使用。但是，对任何非 IBM 产品、程序或服务的操作评估和验证都由用户自行负责。

IBM 可能已经申请或正在申请与本信息有关的各项专利权。提供本信息并不表示允许您使用这些专利。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

涉及双字节 (DBCS) 信息的许可证查询，请联系您所在国的 IBM 知识产权部门或以书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

下列篇幅不适用于英联邦国家或任何当地法律与本条款不一致的其它国家: 国际商业机器公司以“照原样”方式提供本出版物，本出版物不带任何形式的担保，无论是明示的或暗示的，其中包括（但不限于），非侵害性、适销性或特定用途的适用性隐式担保。有些国家禁止对某些事物的明示的或暗示的担保推卸责任，因此上述条款对您或许不适用。

本资料可能会包含技术上的不精确性或印刷错误。此处提到的信息会定期更改；这些更改会合并至本资料的新版本中。IBM 可能会在不作声明的情况下，随时对本文中所说明的产品和（或）程序作改进和（或）更改。

本信息中引用的任何非 IBM Web 站点仅为方便起见，在任何情况下都不能作为对那些 Web 站点的认同。那些站点中的资料不是 IBM 产品的一部分，使用那些 Web 站点请风险自负。

IBM 也许会以它认为适当的方法使用或散发您提供的信息，而不必对您承担任何责任。

为了以下目的：(i) 允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换 (ii) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的合法用户请与下列地址联系：

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

只要遵守适当的条件和条款，包括某些情形下的一定数量的付款，都可获取这方面的信息。

本信息中描述的特许程序及所有可用于它的特许资料都是由 IBM 按 IBM 客户协议、IBM 国际编程许可证协议及我们之间的其它同等协议的各项条款提供的。

这里包含的任何性能数据都是在受控环境中确定的。因此，在其它操作环境中获得的结果可能会有显著不同。一些测量可能是在开发级系统上进行的，因此不保证这些测量会与常规可用系统上的测量相同。另外，某些测量可能是通过推算法估计的。实际结果可能会有所不同。本文档的用户应该针对他们的特定环境验证可用数据。

涉及非 IBM 产品的信息是从这些产品的供应商、他们出版的通告或其它公开可用的途径获得的。IBM 没有测试过这些产品，不能确认其性能的准确度、兼容性或任何与非 IBM 产品相关的其它声明。有关非 IBM 产品功能的问题应该与这些产品的供应商联系。

商标

以下术语是国际商业机器公司在美国和／或其它国家的商标：

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

Lotus 和 LotusScript 是莲花软件有限公司在美国和／或其它国家的商标。

Java 和所有基于 Java 的商标和徽标是 Sun 公司在美国和／或其它国家的商标或注册商标。

Windows、Windows NT 和 Windows 徽标是微软公司在美国和／或其它国家的商标。

其它公司、产品和服务名称，可能是其它公司的商标或服务标记。

词汇表

词汇表中包含了一些在 MQSeries Adapter Kernel 文档中使用的*关键*术语及其含义。

如果一个特殊概念或术语仅在一节中出现，那么它很可能未包括在词汇表中。但是，它很可能在第119页的『索引』中找到。

词汇表不包含其它 IBM 产品（如 MQSeries）的术语。

适配器. MQSeries Adapter Builder 的输出。通常，用户特定于要发送到应用程序或从应用程序发送的一*种消息类型* 来构建每个适配器。因此，适配器本身并非 MQSeries Adapter Offering 的一部分。适配器由编译成共享程序库的 C 或 Java 源代码组成。当适配器和 MQSeries Adapter Kernel 一起运行时，它们执行 MQSeries Adapter Offering 的运行时功能。根据用户在 MQSeries Adapter Builder 中建模的方式，适配器可以包含非常广泛的功能，如控制流、数据流、顺序导航、包含判定和迭代的条件转移、数据输入、数据上下文存储、变换数据元素、事务性控制、逻辑运算和定制代码等。可以重用已经创建的适配器。

请参阅第116页的『消息类型』、第117页的『源应用程序』和第118页的『目标应用程序』。

适配器守护程序. 作为内核一部分的可执行软件。适配器守护程序仅在推送传递模型中使用。它的目的是实例化工作程序。启动它之后，适配器守护程序保持活动。对于每个目标应用程序，可以有一个或多个适配器守护程序。

在某些情况下，适配器守护程序执行目标应用程序的角色。它执行必需功能，例如，使用目标适配器发送电子邮件消息或将记录写入文件。

aqmconfig.xml 文件. 请参阅第116页的『配置文件』。

aqmsetup 文件. 请参阅第117页的『设置文件』。

应用程序逻辑标识. 表示与适配器（源适配器或目标适配器）相关联的应用程序标识。请参阅第117页的『源逻辑标识』和第118页的『目标逻辑标识』。

与应用程序无关的格式. 请参阅第116页的『集成消息』。

应用程序特定接口. 出于以下目的之一开发的 MQSeries Adapter Offering 之外的接口:

- 允许源适配器从源应用程序获取消息。
- 允许目标应用程序从目标适配器获取消息。

BOD. 商业对象文档。在组织内或组织之间流动的标准商业处理的表示法。示例有: 添加采购订单、显示产品可用性以及添加销售订单。BOD 是由 OAG 使用 XML 定义的。请参阅第117页的『OAG』 和第118页的『XML』。

MQSeries Adapter Offering 可使用 BOD 来定义它的集成消息中的消息主体。

主体类别. 消息中包含的表示消息的应用程序类型的数据，例如 OAG 或 RosettaNet。它属于消息控制值集合。请参阅第116页的『消息控制值』。

主体类别还帮助指定消息类型。请参阅第116页的『消息类型』。

主体类型. 消息中包含的表示消息特定目的（例如，添加销售订单或同步化库存）的数据。它属于消息控制值集合。请参阅第116页的『消息控制值』。

主体类型还帮助指定消息类型。请参阅『消息类型』。

配置文件. aqmconfig.xml 文件，包含大部分内核配置值。有关详细信息，请参阅第55页的『配置文件』。

通信消息. 任何通信传送工具特定的信息加上消息存放器对象，转换成特定于所使用的通信传送工具的消息传递格式。

通信方式. 内核用来传送消息以及执行代理服务的方式。

目的地逻辑标识. 表示目标应用程序的值。内核将它与其它消息控制值一起使用，以路由消息和编组消息。请参阅『消息控制值』。

传递模型. 到目标应用程序的内核接口有两种模型。这两种模型是:

推送 内核负责启动和管理将消息传递到目标应用程序。通常这个模型不需要更改目标应用程序以支持 MQSeries Adapter Offering。

拉入 目标应用程序负责管理消息传递。这个模型需要更改目标应用程序以支持 MQSeries Adapter Offering。目标应用程序必须管理至目标应用程序的内核接口。

相关应用程序标识. 工作程序服务的应用程序名称。工作程序基于适配器守护程序的名称从配置文件获得相关应用程序标识。

DTD. 文档类型定义。在 XML 中，通常是包含一种特殊类型文档的形式定义的一个文件（或同时使用几个文件）。它指定可用于 DTD 内部元素的名称、在 DTD 内可允许使用元素的位置，以及这些元素是如何相互适应的。在 MQSeries Adapter Offering 中，可以使用 DTD 来定义消息主体。请参阅第118页的『XML』 和『集成消息』。

错误队列. 以 MQSeries Adapter Offering 术语来讲，它是在无法处理从接收队列获得的消息时使用的消息队列。

集成消息. 由应用程序数据组成的消息，这些数据使用集成的与应用程序无关的格式。例如，源适配器从源应用程序格式转换为 XML 的 XML 文档。

Java 服务适配器. 一种 Java 语言适配器的类型，在 JMS Listener 环境中，它提供适配器守护程序、工作程序和目标适配器的功能。

JMS Listener. 由 WebSphere Business Integrator 产品提供的一个组件，它可以将 MQSeries Adapter Kernel 和 WebSphere Application Server 高级版紧密地结合在一起。

内核. 同 MQSeries Adapter Kernel。

逻辑消息服务. 本机适配器用来将消息转换成供通信传送工具用格式的组件。

登录类. 特定于每个目标应用程序的 Java 类，可用于帮助将消息发送到目标应用程序。仅当在发送消息之前目标适配器必须登录到目标应用程序时才需要登录类。每个登录类都是由用户编写的。工作程序实例化登录类。登录类查寻配置文件以查找目标适配器所需的支持到目标应用程序的应用程序特定接口的值。通常，这些值是登录参数。这样，目标适配器就可以使用这些值了。

内核还提供了不执行任何操作的哑登录类。

消息. 在 MQSeries（包括 MQSeries Adapter Offering）中，由一个程序发送的要传递给另一个程序的数据集合。

消息控制值. 消息（主体和头）和配置文件中一组值的总称，内核使用它们来控制编组和路由消息，每个适配器使用它们来部分地控制功能的执行。

消息存放器对象. 内核使用的元数据容器，用来封装集成消息和其它控制数据。

消息类型. 由主体类别和主体类型的唯一组合指定的消息。请参阅第115页的『主体类别』和第115页的『主体类型』。

MQSeries Adapter Builder. 允许用户使用图形用户界面 (GUI) 为任何实际应用程序构建适配器的软件。

MQSeries Adapter Kernel. 以 C 和 Java 语言编写的一组 API 和几个可执行程序，以及几个配置文件。内核使用并支持适配器。请参阅第115页的『适配器』。除了直接支持适配器外，内核还执行相关功能，最重要的有：路由消息和基础结构服务（如消息构造）、跟踪以及与 MQSeries 或其它消息传递软件交互。

MQSeries Adapter Offering. 一组应用程序集成产品，由 MQSeries Adapter Builder 和 MQSeries Adapter Kernel 组成。

MQSeries Adapter Kernel 本机适配器. 同本机适配器。

本机适配器. 用于发送和接收消息存放器对象的软件。

OAG. 开放式应用程序组。一个非赢利性的行业社团，由许多在商业软件组件互操作性领域中非常杰出的股东组成。OAG 定义“商业对象文档 (BOD)”。

传递的拉入模型. 请参阅第116页的『传递模型』。

传递的推送模型. 请参阅第116页的『传递模型』。

接收队列. 以 MQSeries Adapter Offering 术语来讲，它是用作主输入队列以接收消息的消息队列。每个目标应用程序可以有多个接收队列，但是对于应用程序标识、主体类别和主体类型的每个组合仅能有一个接收队列。

应答队列. 用于接收应答的消息队列。它与内核的 `sendRequestResponse` 方法一起使用。

响应逻辑标识. 在请求应答时，向其发送应答的应用程序逻辑标识。它的缺省值是消息中的源逻辑标识。

设置文件. 包含一些内核初始设置的文件。文件的缺省名称是 `aqmsetup`。

源适配器. 执行下列任务的适配器：

- 从源应用程序接受或获取结构化数据（通常使用在适配器外部开发的应用程序特定接口）。
- 根据适配器建模的方式处理结构化数据。
- 将结构化数据转换为集成消息格式。
- 使用内核，将消息放入消息队列，以便传递到一个或多个目标适配器，之后再传递到目标应用程序。

对于每个消息类型，都有一个源适配器。通常，一个源应用程序可以发送多种消息类型；因此，在大多数情况下，有多个源适配器支持一个源应用程序。

请参阅第115页的『适配器』。

源应用程序. 在计算机网络上，将数据发送到通常位于另一台计算机上的应用程序（也叫做目标应用程序）的程序。

源逻辑标识. 表示源应用程序的值。内核将它与其它消息控制值一起使用，以路由消息和编组消息。请参阅第116页的『消息控制值』、第115页的『应用程序逻辑标识』和第118页的『目标逻辑标识』。

内核的源方. 内核功能的一部分，它在从源适配器接收到消息时开始，在消息放入消息队列中时结束。

目标适配器. 执行下列任务的适配器：

- 接收由源适配器发送的消息（从内核和 MQSeries 或其它消息传递软件）。
- 根据适配器建模的方式处理集成消息。
- 将集成消息转换为目标应用程序可接收的应用程序特定格式消息。
- 使用应用程序特定接口将消息发送到目标应用程序。
- 在它结束将消息发送到目标应用程序时通知工作程序，以使工作程序发送确认。

如果目标应用程序可以接收集成消息，那么可能不需要目标适配器。

对于每个消息类型，都有一个目标适配器。通常，一个目标应用程序可以接受多种消息类型；因此，在大多数情况下，有多个目标适配器支持一个目标应用程序。请参阅第115页的『适配器』。

目标应用程序. 在计算机网络上，从通常位于另一台计算机上的应用程序（也叫做源应用程序）接收数据的程序。

目标逻辑标识. 表示与目标适配器相关的目标应用程序的值。请参阅『目标逻辑标识』和第115页的『应用程序逻辑标识』。

内核的目标方. 内核功能的一部分，它在从消息队列获得消息时开始，在发送消息到目标适配器之后结束。

跟踪客户机. 写跟踪消息的内核组件。

跟踪消息. 包含内核中某一点处理消息状态的消息。可以使用跟踪消息帮助诊断内核问题或适配器问题。

请参阅『跟踪』。

跟踪. 内核用来写跟踪消息的进程集合。请参阅『跟踪消息』。

事务. 必须作为一个不可分的工作单元执行的一组操作。如果构成某一事务的所有操作都成功，那么提交事务；也就是执行所有操作。如果构成某一事务的一个或多个操作失败，则回滚事务；即没有执行任何操作。

WebSphere Application Server 高级版. IBM 的一个软件产品，它允许使用 Sun Microsystems Enterprise JavaBean (EJB) 规范。“WebSphere Application Server 高级版”包含一个 enterprise bean 可在其中运行的 EJB 服务器。Enterprise bean 封装了商业逻辑以及由 EJB 客户机使用和共享的数据。有两种类型的 enterprise bean: 会话 bean，它封装暂时的客户机特定的任务和对象；实体 bean，它封装永久数据。一种称为工作程序消息 bean 的会话 bean 可用在 MQSeries Adapter Kernel 的目标方。

工作程序. 作为内核一部分的软件。工作程序仅在推送传递模型中使用。适配器守护程序启动并创建工作程序。每个工作程序管理一个本机适配器。工作程序将每个消息发送到适当的目标适配器。

工作程序消息 bean. 一种 enterprise bean，当 WebSphere Application Server 用在内核的目标方时它执行工作程序的功能。

XML. 可扩展标记语言。数据表示法的 W3C 标准。

索引

B

D

E

H

J

M

O

S

W

X

中国印刷

GB84-0445-01