MQSeries Adapter Kernel for Multiplatforms

Einstieg

Version 1 Release 1

GC12-2980-01

MQSeries Adapter Kernel for Multiplatforms

Einstieg

Version 1 Release 1

GC12-2980-01

Anmerkung

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Bemerkungen” auf Seite 141 gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business Symbol ist eine Marke der International Business Machines Corporation
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle Java-basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Änderungen in der IBM Terminologie

Die ständige Weiterentwicklung der deutschen Sprache nimmt auch Einfluß auf die IBM Terminologie. Durch die daraus resultierende Umstellung der IBM Terminologie, kann es u. U. vorkommen, dass in diesem Handbuch sowohl alte als auch neue Termini gleichbedeutend verwendet werden. Dies ist der Fall, wenn auf ältere existierende Handbuchausschnitte und/oder Programmteile zurückgegriffen wird.

Zweite Ausgabe (Mai 2001)

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
MQSeries Adapter Kernel for Multiplatforms Quick Beginnings Version 1 Release 1,
IBM Form GC34-5855-05,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderungen des Textes bleiben vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
Mai 2001

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Willkommen bei MQSeries Adapter Kernel Einstieg

Dieses Dokument beschreibt MQSeries Adapter Kernel und erläutert die Vorgehensweise zur Planung, Installation und Verwendung des Produkts.

Gehen Sie folgendermaßen vor, um den Kernel für den Einsatz vorzubereiten:

1. Lesen Sie „Kapitel 1. Informationen zu MQSeries Adapter Offering" auf Seite 1.
2. Bereiten Sie die Installation vor. Weitere Informationen finden Sie unter „Installation vorbereiten" auf Seite 46.
3. Installieren Sie den Kernel. Weitere Informationen finden Sie unter „Kernel installieren" auf Seite 47.
4. Überprüfen Sie die Installation. Weitere Informationen finden Sie unter „Installation überprüfen" auf Seite 54.
5. Konfigurieren Sie den Kernel. Weitere Informationen finden Sie unter „Kernel konfigurieren" auf Seite 68.
6. Konfigurieren Sie bei Bedarf wahlweise installierbare Software, die Sie zusammen mit dem Kernel einsetzen wollen. Weitere Informationen finden Sie unter „MQSeries und MQSeries Integrator konfigurieren" auf Seite 104.
7. Erstellen Sie Ihre Adapter mit Hilfe von MQSeries Adapter Builder. Testen Sie dann die Adapter, und geben Sie sie für den Einsatz frei. Weitere Informationen zu MQSeries Adapter Builder finden Sie in der Dokumentation zu dieser Komponente.
8. Starten Sie den Kernel. Weitere Informationen finden Sie unter „Kernel starten" auf Seite 105.

Zum besseren Verständnis dieser Informationen sind Kenntnisse über Voraussetzungen und Zusatzprogramme erforderlich (siehe „Kapitel 2. Installation des Kernels planen" auf Seite 37) sowie „Referenzinformationen" auf Seite 113).

Zielgruppe

Dieses Buch richtet sich an alle, die den Einsatz von MQSeries Adapter Kernel planen, das Produkt installieren oder es verwenden wollen.

Referenzinformationen

Für zusätzliche Informationen stehen Ihnen folgende Quellen zur Verfügung:

- Die Datei `readme.txt`. Diese Datei enthält gegebenenfalls Informationen, die erst nach Fertigstellung dieses Buchs verfügbar waren. Wenn das Produkt noch nicht installiert ist, finden Sie die Datei `readme.txt` im Stammverzeichnis auf der Produkt-CD-ROM. Nach der Installation befindet sich die Datei `readme.txt` im Stammverzeichnis, das Sie bei der Installation von MQSeries Adapter Kernel angegeben haben.
- Das Handbuch *Problem Determination Guide*, Formnummer GC34-5897, enthält eine Beschreibung der Tools, einschließlich der Trace-Funktion, für die Lösung spezifischer Probleme, die in Verbindung mit MQSeries Adapter Kernel auftreten können. Sie finden das Handbuch *Problem Determination Guide* im MQSeries Adapter Kernel Informationszentrum, das mit dem Produkt installiert wird.
- Die Onlinedokumentation zur Anwendungsprogrammierschnittstelle (API), die ebenfalls im MQSeries Adapter Kernel Informationszentrum zur Verfügung steht. Diese Information dient lediglich dazu, die Funktionsweise des Kernels zu erläutern, und soll Hinweise zur Diagnose geben. (Siehe „Kapitel 5. APIs von MQSeries Adapter Kernel verwenden" auf Seite 111.)
- Die Informationen zu MQSeries Adapter Builder, einschließlich Handbüchern und Hilfefunktion.
- Die Website der MQSeries-Produktfamilie finden Sie unter www.ibm.com/software/ts/mqseries/.

  Über die Links auf dieser Website haben Sie folgende Möglichkeiten:

  – Abrufen von aktuellen Informationen zur MQSeries-Produktfamilie, einschließlich MQSeries Adapter Offering.
  – Zugreifen auf MQSeries-Bücher im HTML- und PDF-Format, gegebenenfalls einschließlich einer aktuelleren Ausgabe dieses Buches.
  – Herunterladen von MQSeries-SupportPacs

Konventionen

In der Dokumentation von MQSeries Adapter Kernel werden die folgenden typografischen Konventionen und Zeichen verwendet.

Tabelle 1. In diesem Handbuch verwendete Konventionen

Konvention	Bedeutung
Fett	Zeigt Befehlsnamen an. Bei grafischen Benutzerschnittstellen sind dies Menüs, Menüpunkte, Bezeichnungen und Schaltflächen.
`Monospace-Schrift`	Bezeichnet Text, den Sie an der Eingabeaufforderung eingeben müssen, und Werte, die Sie wie angegeben verwenden müssen, z. B. Dateinamen und Pfade sowie Elemente von Programmiersprachen wie Funktionen, Klassen und Methoden. Texte in Anzeigen und Codebeispiele sind ebenfalls in Monospace-Schrift.
Kursiv	Werte von Variablen, die Sie angeben müssen (z. B. müssen Sie für *Dateiname* den Namen einer Datei eingeben). Hervorhebungen und Buchtitel sind ebenfalls kursiv dargestellt.
`%`	Steht für die Eingabeaufforderung der UNIX-Befehls-Shell für Befehle, die nicht die Root-Berechtigung erfordern.
`#`	Steht für die Eingabeaufforderung der UNIX-Befehls-Shell für Befehle, für die die Root-Berechtigung erforderlich ist.
`C:\>`	Steht für die Eingabeaufforderung auf Windows-Systemen.
`>`	Zeigt in der Beschreibung eines Menüs eine Folge von Menüpunkten an, die ausgewählt werden müssen. Beispiel: "Klicken Sie auf **Datei > Neu**" bedeutet "Öffnen Sie das Menü **Datei**, und klicken Sie auf den Befehl **Neu**."
Befehle eingeben	Wenn Sie aufgefordert werden, einen Befehl "einzugeben", geben Sie den Befehl ein und drücken Sie die Eingabetaste. Beispiel: Die Anweisung "Geben Sie den Befehl **ls** ein" bedeutet, dass Sie **ls** an der Eingabeaufforderung eingeben und dann die Eingabetaste drücken.
`[ ]`	Schließen optionale Elemente in einer Syntaxbeschreibung ein.
`{ }`	Schließen Listen in der Syntaxbeschreibung ein, aus denen Sie ein Element auswählen müssen.
`|`	Trennt Elemente in einer Auswahlliste in einer Syntaxbeschreibung, die in geschweifte Klammern ({ }) eingeschlossen ist.
...	Auslassungen in einer Syntaxbeschreibung bedeuten, dass Sie das vorausgehende Element mehrfach wiederholen können. Auslassungen in Beispielen bedeuten, dass an der betreffenden Stelle Informationen aus Platzgründen weggelassen wurden.

Anmerkung: Der Begriff Epic wird Ihnen in einigen Werten und Namen der Kernel-Software und in diesem Buch begegnen. Dieser Begriff hat in Bezug auf MQSeries Adapter Offering keine spezielle Bedeutung.

Zusammenfassung der Änderungen

Das vorliegende Handbuch (zweite Ausgabe) enthält die folgenden Erweiterungen und Änderungen zur ersten Ausgabe:

- Aktualisierung des Abschnitts über die Laufzeitverarbeitung zur Darstellung verschiedener Änderungen (siehe „Laufzeitverarbeitung" auf Seite 17).
- Informationen zur Verwendung von MQSeries Adapter Kernel in Verbindung mit WebSphere Business Integrator. Weitere Informationen finden Sie unter „MQSeries Adapter Kernel mit WebSphere Business Integrator und WebSphere Application Server verwenden" auf Seite 34.
- Informationen zum Umfang der Unterstützung in der Landessprache, die mit verschiedenen Adaptern zur Verfügung gestellt wird. Weitere Informationen finden Sie unter „Unterstützung in der Landessprache" auf Seite 35.
- Überarbeitung der Installationsanweisungen (siehe „Kernel installieren" auf Seite 47).
- Informationen zur automatischen Installation. Weitere Informationen finden Sie unter „Automatische Installation verwenden" auf Seite 60.
- Eine konzeptionelle Konfigurationsübersicht zur Unterstützung der Konfiguration des Kernels. Weitere Informationen finden Sie unter „Übersicht über die Konfiguration" auf Seite 69.
- Informationen zu neuen Header-Werten. Weitere Informationen finden Sie unter „MQSeries Adapter Kernel - Nachrichtendeskriptor-Header" auf Seite 123.

Die erste Ausgabe enthielt die folgenden Erweiterungen und Änderungen zu vorhergehenden Ausgaben, die nur in Englisch verfügbar sind:

- Informationen zur Verwendung des Kernels auf Windows 2000-, OS/400-, HP-UX- und Solaris-Plattformen. Die Unterstützung dieser Plattformen war neu in MQSeries Adapter Kernel Version 1.1. Der Kernel war bisher nur für Windows NT und AIX verfügbar.
- Aktualisierungen aller Installationsanweisungen zur Anpassung an den Installationsprozess von MQSeries Adapter Kernel Version 1.1.
- Informationen zur Verwendung der Datei `aqmconfig.xml` für die Konfiguration von MQSeries Adapter Kernel. Der Kernel wurde bisher über die Datei `aqmconfig.properties` konfiguriert. Weitere Informationen finden Sie unter „Die Konfigurationsdatei" auf Seite 75.

- Informationen zu den neuen Übertragungsmodi von MQ und JMS (Java Message Service). Weitere Informationen finden Sie unter „Anhang A. Übertragungsmodus" auf Seite 115.
- Informationen zur Trace-Funktion wurden aus diesem Dokument entfernt und in das neue Dokument *Problem Determination Guide* übernommen. Ausführliche Informationen finden Sie im Handbuch *Problem Determination Guide*.

Kapitel 1. Informationen zu MQSeries Adapter Offering

IBM MQSeries Adapter Kernel gehört zu einer Gruppe von Produkten für die Anwendungsintegration, die in der Produktfamilie IBM MQSeries Adapter Offering zusammengefasst sind. MQSeries Adapter Offering ermöglicht es Ihnen in Verbindung mit der MQSeries-Nachrichtenübermittlung und anderen Nachrichtenservices, die Risiken, Komplexität und Kosten der Verwaltung der Punkt-zu-Punkt-Integration Ihrer Geschäftsprozesse zu reduzieren.

Bei der *Punkt-zu-Punkt*-Integration kommuniziert jede Anwendung über separate Schnittstellen mit den einzelnen anderen Anwendungen (1:1-Integration). Jede dieser Schnittstellen ist anders, und es gibt viele verschiedene Schnittstellen. Wird eine der Anwendungen geändert, hat dies in der Regel zur Folge, dass auch viele Schnittstellen geändert werden müssen. Mit der zunehmenden Zahl der eingesetzten Anwendungen steigen entsprechend die Kosten für die Punkt-zu-Punkt-Integration. Bei jeder neuen Anwendung, die integriert wird, entsteht ein höherer Aufwand für die Integration als bei der vorhergehenden.

Mit MQSeries Adapter Offering können Sie von der Punkt-zu-Punkt-Integration zu einer *1:n*-Integration migrieren. Eine 1:n-Integration hat neben vielen anderen vor allem folgende Vorteile:

- Alle Anwendungen können eine gemeinsame Schnittstelle verwenden.
- Daten aus einer *Quellenanwendung* werden in Form einer *Nachricht* an eine oder mehrere *Zielanwendungen weitergeleitet*.
- Wird eine Anwendung geändert, betrifft das in der Regel nur die Schnittstelle dieser einen Anwendung.
- Durch die Verwendung einer gemeinsamen Schnittstelle, die anwendungsneutral ist, z. B. ein Industriestandard wie die Formatierungssprache XML (Extensible Markup Language), können die Kosten sogar noch weiter gesenkt werden. Es können mehr Anwendungen unterstützt werden, und das mit geringerem Aufwand.
- Mit einer zunehmenden Zahl der eingesetzten Anwendungen wird die 1:n-Integration sogar noch kostengünstiger. Denn wenn eine neue Anwendung hinzugefügt wird, hat dies in der Regel keine nennenswerten Änderungen an den Schnittstellen aller anderen Anwendungen zur Folge.
- Die Durchführung der Integration kann automatisiert werden und auf der Basis von Schablonen erfolgen.

MQSeries Adapter Offering kann eingesetzt werden, ohne dass irgendwelche Änderungen von Anwendungen oder Geschäftsprozessen erforderlich sind. Alle Integrationsaufgaben werden in MQSeries Adapter Offering ausgeführt, sodass die Notwendigkeit für die Erstellung benutzerspezifischer Anwendungen verringert wird.

In MQSeries Adapter Offering übernimmt ein *Adapter* die Funktion der Schnittstelle zu einer Anwendung. Alle Anwendungen benötigen mindestens einen Adapter als Schnittstelle zwischen der Anwendungsumgebung und der Nachrichtenübermittlungsumgebung. Jeder Adapter ist speziell für eine Anwendung und eine Nachrichtenart zuständig.

MQSeries Adapter Kernel kann optional zusammen mit MQSeries Integrator eingesetzt werden, um Broker-Funktionen und Nachrichtenkonvertierungen auszuführen. MQSeries Adapter Offering kann durch Serviceangebote von IBM und Drittanbietern ergänzt werden.

Adapter können beispielsweise folgende Aufgaben ausführen:

- Hinzufügen eines Verkaufsauftrags
- Synchronisieren eines Kundendatensatzes
- Synchronisieren eines Inventardatensatzes
- Synchronisieren eines Artikels
- Synchronisieren eines Verkaufsauftrags

Erstellungszeit und Laufzeit

MQSeries Adapter Offering besteht aus zwei Hauptkomponenten: Adapter Builder (das Erstellungsprogramm) und Adapter Kernel (oder einfach nur Kernel). In diesem Abschnitt werden diese beiden Komponenten und die Adapter, die von Adapter Offering erstellt und ausgeführt werden, beschrieben.

Adapter

Ein Adapter ist ein Softwareprogramm, das eine Schnittstelle zu einer Anwendung bereitstellt. Adapter werden mit Hilfe von MQSeries Adapter Builder erstellt. In der Regel wird für jede einzelne Nachrichtenart, die von einer oder an eine Anwendung gesendet wird, ein spezieller Adapter erstellt. Adapter selbst sind nicht Teil von MQSeries Adapter Offering.

Ein Adapter besteht aus einem C- oder Java-Quellencode, der kompiliert und in einer gemeinsam benutzten Bibliothek abgelegt wird. Werden die Adapter und MQSeries Adapter Kernel gemeinsam ausgeführt, bilden sie zusammen die Laufzeitumgebung von MQSeries Adapter Offering.

Abhängig davon, welche Funktionen für den Adapter bei seiner Erstellung in MQSeries Adapter Builder definiert wurden, kann er ganz unterschiedliche Aufgaben übernehmen, z. B. Steuerungsfluss, Datenfluss, sequenzielle Navigation, bedingte Verzweigung einschließlich Entscheidung und Iteration, Dateneingabe, Speicherung von Datenkontexten, Konvertierung von Datenelementen, Transaktionssteuerung, logische Operationen und benutzerdefinierter Code.

Adapter können wiederverwendet werden.

Es gibt zwei Hauptadapterarten:

- Quellenadapter für Anwendungen, die Daten senden.
- Zieladapter für Anwendungen, die Daten empfangen.

Um eine Nachrichtenart von einer Anwendung an eine zweite Anwendung zu senden, werden in der Regel ein Quellenadapter und ein Zieladapter benötigt. Wenn die zweite Anwendung eine Nachrichtenart an die erste Anwendung senden muss, sind ein anderer Quellenadapter und ein anderer Zieladapter erforderlich. Das heißt, um eine Nachrichtenart von der ersten Anwendung an die zweite Anwendung zu senden und anschließend eine andere Nachrichtenart von der zweiten an die erste Anwendung zurückzuschicken, sind vier Adapter erforderlich.

Für jede Nachrichtenart ist ein eigener Adapter erforderlich.

Ein dritter Adaptertyp, der Session-Bean-Adapter des Java-Service, wird eingesetzt, wenn IBM WebSphere Application Server und Enterprise-Beans in der Zielverarbeitungsschicht des Kernels verwendet werden. Die WebSphere Application Server-Implementierung der Sun Microsystems Enterprise JavaBeans (EJB)-Spezifikation ermöglicht die Verwendung von Session-Bean-Adaptern des Java-Service und anderer Enterprise-Beans. Weitere Informationen finden Sie unter „MQSeries Adapter Kernel mit WebSphere Business Integrator und WebSphere Application Server verwenden" auf Seite 34 und in der MQSeries Adapter Builder-Dokumentation.

MQSeries Adapter Builder

MQSeries Adapter Builder ist eine grafische Benutzerschnittstelle (GUI), über die ein Benutzer einen Adapter für praktisch jede Anwendung erstellen kann. Die Benutzerschnittstelle ist der von MQSeries Integrator ähnlich. Weitere Informationen finden Sie im Informationszentrum von MQSeries Adapter Builder.

MQSeries Adapter Kernel

MQSeries Adapter Kernel besteht aus einer Gruppe von Anwendungsprogrammierschnittstellen (APIs), verschiedenen ausführbaren Programmen (in C und Java) sowie verschiedenen Konfigurationsdateien. Der Kernel ermöglicht den Einsatz und die Ausführung von Adaptern. Neben der direkten Unterstützung von Adaptern führt der Kernel zugehörige Funktionen aus, z. B. das einfache Routing von Nachrichten. Darüber hinaus stellt er Infrastruktur-Services für die Nachrichtenerstellung, Transaktionssteuerung, Durchführung von Traces und die Kommunikation mit MQSeries oder einer anderen Nachrichtenübermittlungssoftware zur Verfügung.

Der Kernel wird auf jedem Computer installiert, auf dem ein Quellen- oder Zieladapter ausgeführt wird.

In MQSeries Adapter Offering sind Geschäftsprozesse und die einzelnen Anwendungen isoliert und damit unabhängig von den Spezifikationen der Middleware, den jeweiligen Nachrichtendetails und anderen Anwendungen. Eine gemeinsame Schnittstelle für die Nachrichtenübermittlung ermöglicht das Hinzufügen neuer Anwendungen, ohne dass vorhandene Anwendungen oder Geschäftsprozesse geändert werden müssen.

MQSeries Adapter Kernel kann in zwei Schichten eingesetzt werden. Eine Schicht ist die Quellenverarbeitung zur Laufzeit, die zweite Schicht ist die Zielverarbeitung zur Laufzeit. Dieses 2-Schichten-Modell führt zu einer effizienten Verarbeitung und zu einer Verringerung des Verwaltungsaufwands. Eine dritte Schicht für das Routing und die Übermittlung ist zwischen diesen beiden Verarbeitungsschichten nicht erforderlich. Sie können jedoch zusätzlich MQSeries Integrator einsetzen, um Broker-Funktionen wie komplexes Routing, Datenkonvertierung und Datenänderung auszuführen.

Sofern nicht anders angegeben, behandelt dieses Dokument nur MQSeries Adapter Kernel. Weitere Informationen zu MQSeries Adapter Builder finden Sie im Informationszentrum dieses Produkts.

Informationen zum Kernel

In ihrer Grundausstattung werden in der Laufzeitumgebung, d. h. vom Kernel und den von Ihnen erstellten Adaptern, die folgenden Aufgaben ausgeführt:

1. Übertragen von Daten von einer Quellenanwendung an eine Zielanwendung.
2. Konvertieren der Daten der Quellenanwendung in eine Nachricht, in der Regel in einem anwendungsneutralen Format, die mit Hilfe von MQSeries oder einer anderen Nachrichtenübermittlungssoftware durch den Kernel weitergeleitet wird.

3. Weiterleiten der Nachricht an die Zielanwendung.
4. Bestimmen der Route, auf der die Daten an die Zielanwendung gelangen.
5. Konvertieren der Daten aus dem Format, in dem die Nachricht durch den Kernel und durch einen Adapter weitergeleitet wurde, in das Format der Zielanwendung.

Dieser Abschnitt soll vorerst nur eine Übersicht über die Funktionalität des Kernels geben. Sie wird im Abschnitt „Laufzeitverarbeitung" auf Seite 17 ausführlich erläutert.

Der Kernel besteht aus zwei Verarbeitungsschichten:

- Der *Quellenverarbeitung*, die mit dem Empfangen der Nachricht von der Quellenanwendung beginnt und mit dem Einreihen der Nachricht in eine Nachrichtenwarteschlange endet.
- Der *Zielverarbeitung*, die mit dem Empfangen der Nachricht von der Nachrichtenwarteschlange beginnt und mit dem Senden der Nachricht an das jeweilige Ziel endet.

Jede dieser Schichten befindet sich normalerweise auf einem anderen Computer, sie können sich jedoch auch auf demselben Computer befinden.

Siehe Abb. 1 auf Seite 6. Sie zeigt den im Folgenden beschriebenen Ablauf.

Quellenverarbeitung des Kernels

1. Bei der Quellenverarbeitung des Kernels sendet die Quellenanwendung die Daten im *Format der Quellenanwendung* über eine *anwendungsspezifische Schnittstelle* an einen Quellenadapter, der mit MQSeries Adapter Builder erstellt wurde. Für jede Nachrichtenart, z. B. für das „Hinzufügen eines Verkaufsauftrags" oder das „Synchronisieren eines Kundendatensatzes", ist ein eigener Adapter erforderlich.

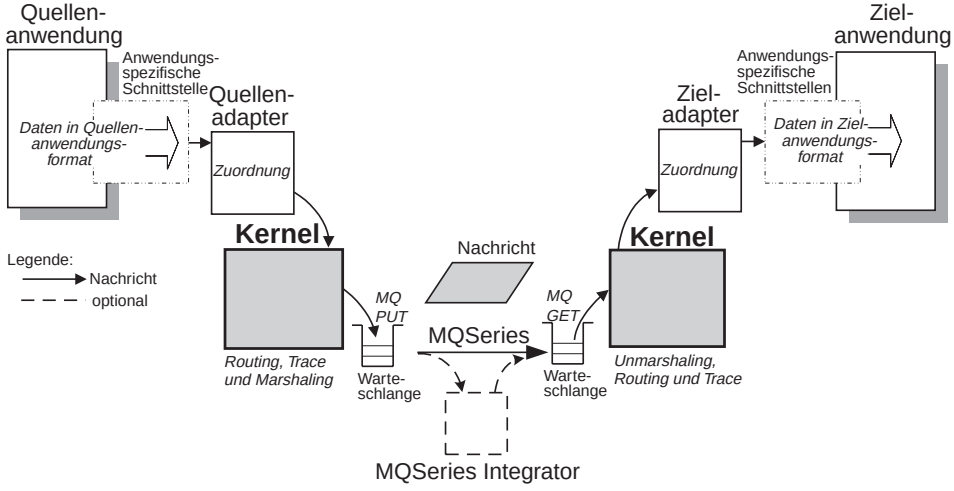
   Die anwendungsspezifische Schnittstelle muss außerhalb von MQSeries Adapter Offering entwickelt werden. Die genaue Funktionsweise der anwendungsspezifischen Schnittstelle ist von den Merkmalen der Quellen- bzw. der Zielanwendung abhängig. Sie können zum Beispiel API-Aufrufe und Benutzer-Exits, Lese- und Schreibzugriffe auf Dateien, Auslöser für Datenbankoperationen sowie Nachrichtenwarteschlangen enthalten.

   Beachten Sie, dass der Quellenadapter im Prozess der Quellenanwendung ausgeführt wird. Jeder Dämon oder Server, der den Quellenadapter enthält, muss aktiv sein, damit der Quellenadapter ordnungsgemäß arbeiten kann.

2. Der Quellenadapter führt seine Funktion so aus, wie sie bei seiner Erstellung definiert wurde. Eine typische Funktion ist die Konvertierung von Datenelementen, d. h. das Zuordnen von Nachrichtendatenelementen aus dem Quellenanwendungsformat zu Elementen in einem *integrierten Datenformat für die Nachrichtenübermittlung*. Die Nachrichtendaten und zusätzlichen Metadaten mit den Steuerwerten werden in einem *Nachrichtenträgerobjekt* im Kernel abgelegt.
3. Wenn der Quellenadapter das Nachrichtenträgerobjekt über den *internen Adapter* an den Kernel übergibt, verwendet dieser im Nachrichtenträgerobjekt enthaltene Steuerwerte (*Nachrichtensteuerungswerte*), um das Marshaling des Nachrichtenträgerobjekts in ein Übertragungsnachrichtenformat und das Routing der Übertragungsnachrichten zu steuern.

   Wenn die Nachricht keine Nachrichtensteuerungswerte enthält, kann der Kernel Standardwerte oder Nachrichtensteuerungswerte, die in der Konfigurationsdatei angegeben sind, verwenden. Definitionen von Nachrichtensteuerungswerten finden Sie unter „Nachrichtensteuerungswerte" auf Seite 18.
4. Der Kernel führt seine Funktionen aus, d. h. *Nachrichten-Marshaling*, einfaches *Routing* und *Traces* (optional). Weitere Informationen finden Sie unter „Nachricht und Nachrichtenformat" auf Seite 14, „Routing und Übermittlung" auf Seite 15 und „Trace-Funktion" auf Seite 33.

Abbildung 1. Übersicht über MQSeries Adapter Offering.



Übermittlung von der Quellenverarbeitungsschicht an die Zielverarbeitungsschicht des Kernels

5. Der Kernel reiht die Nachricht über seinen internen Adapter in die zugehörige Nachrichtenwarteschlange ein.

   Bei der Quellenverarbeitung werden zwei Sendemethoden verwendet:

   - `sendMsg` - Die Nachricht wird gesendet, eine Antwort wird nicht abgewartet. Die Methode `sendMsg` kann auch zusammen mit den Methoden `begin`, `commit` und `rollback` verwendet werden, um Nachrichten *transaktionsabhängig* zu senden, d. h. Nachrichten können nur dann gesendet werden, wenn andere Operationen erfolgreich abgeschlossen wurden. Weitere Informationen finden Sie unter „Transaktionsfunktionen des Kernels" auf Seite 33.
   - `sendRequestResponse` - Die Nachricht wird gesendet und eine Antwort wird abgewartet. Die Methode `sendRequestResponse` kann nicht transaktionsabhängig verwendet werden.

     Bei der Zielverarbeitung des Kernels wird darüber hinaus eine dritte Methode, `sendResponse`, verwendet, wenn der Sender eine Antwort anfordert.

   Die Nachricht wird von MQSeries oder einer anderen Nachrichtenübermittlungssoftware transportiert. Weitere Informationen finden Sie unter „Die Rolle von MQSeries oder anderen Nachrichtenübermittlungsprodukten" auf Seite 9. Beachten Sie, dass die Nachrichtenübermittlungssoftware bereits für die Unterstützung von MQSeries Adapter Offering konfiguriert sein muss.

   Wenn MQSeries Integrator im Kernel als Ziel konfiguriert ist, kann MQSeries Integrator optional Broker-Funktionen ausführen. Weitere Informationen finden Sie unter „Die Rolle von MQSeries Integrator" auf Seite 10. Wenn der Zielort, eine Nachrichtenwarteschlange, in den Regeln oder Nachrichtenflüssen von MQSeries Integrator konfiguriert wurde, sendet MQSeries Integrator die Nachricht an diese Nachrichtenwarteschlange.

   Die Nachricht wird an die zugehörige Nachrichtenwarteschlange übergeben.

Zielverarbeitung des Kernels

6. Bei der Zielverarbeitung des Kernels werden für die Schnittstelle zwischen der Laufzeitumgebung und der Zielanwendung zwei mögliche *Übermittlungsmodelle* verwendet.
   - Das gebräuchlichste Modell ist das *Push*-Modell, wobei der Kernel für das Einleiten und Verwalten der Nachrichtenübermittlung zuständig ist. Bei Verwendung des Push-Modells ist in der Regel keine Änderung der Zielanwendung zur Unterstützung von MQSeries Adapter Offering erforderlich.
   - Im *Pull*-Modell ist die Zielanwendung für den Empfang der Nachricht zuständig. Bei Verwendung des Pull-Modells ist eine Änderung der Zielanwendung für die Unterstützung von MQSeries Adapter Offering erforderlich. Die Zielanwendung muss die Schnittstelle zwischen Kernel und Zielanwendung verwalten.

   Wird das Push-Modell eingesetzt, muss der Benutzer vorher in der Zielverarbeitungsschicht die Prozesse des Kernels starten, damit die Nachricht empfangen und übermittelt werden kann. Weitere Informationen finden Sie unter „Kernel starten" auf Seite 105.

   Im Push-Modell ruft der Kernel die Nachricht aus der Nachrichtenwarteschlange ab. Er führt Traces durch, sofern dies aktiviert wurde. Anschließend wählt er den zugehörigen Zieladapter aus und leitet die Nachricht weiter. Im Allgemeinen ist für jede Nachrichtenart ein eigener Zieladapter erforderlich.
7. Der Kernel übermittelt die Nachricht an den zugehörigen Zieladapter. Der Zieladapter führt die Funktionen aus, die bei seiner Erstellung definiert wurden. Eine typische Funktion besteht darin, Elemente des integrierten Datenformats für die Nachrichtenübermittlung Elementen des *Zielanwendungsformats* zuzuordnen.

   Zieladapter können unter einem MQSeries Adapter Kernel-Adapterdämon oder unter WebSphere Application Server ausgeführt werden. Letzteres wird unter „MQSeries Adapter Kernel mit WebSphere Business Integrator und WebSphere Application Server verwenden" auf Seite 34 näher behandelt.
8. Der Zieladapter sendet die Daten im Zielanwendungsformat über eine anwendungsspezifische Schnittstelle, die außerhalb von MQSeries Adapter Offering entwickelt wurde, an die Zielanwendung.
9. Nachdem der Zieladapter die Nachricht übermittelt hat, wird die Nachricht von der Nachrichtenwarteschlange festgeschrieben (COMMIT-Operation). Dabei wird die Nachricht aus der Warteschlange entfernt.

10. Wenn der Quellenadapter über einen Nachrichtensteuerungswert festgelegt hat, dass eine Empfangsbestätigung erforderlich ist, schickt der Kernel mit Hilfe der Methode `sendResponse` entweder eine Bestätigung der Nachrichtenübermittlung oder die Zieladapterausgabe an den Quellenadapter.
11. Im Fehlerfall reiht der Kernel die ursprüngliche Nachricht in die Fehlerwarteschlange ein. Wenn der Kernel die ursprüngliche Nachricht nicht in die Fehlerwarteschlange einreihen kann, wird keine COMMIT-Operation ausgeführt.

Die Rolle von MQSeries oder anderen Nachrichtenübermittlungsprodukten

Die Übertragungsnachrichten von MQSeries Adapter Offering werden über Nachrichtenwarteschlangen transportiert. Nachrichtenwarteschlangen werden von Nachrichtenübermittlungsprodukten wie MQSeries oder Java Message Service (JMS) bereitgestellt. Für Nachrichten, die von MQSeries Adapter Offering transportiert werden, werden folgende Warteschlangen verwendet:

- *Empfangswarteschlangen*, in der Terminologie von MQSeries Adapter Offering. Dies sind die Haupteingabewarteschlangen für den Empfang von Nachrichten. Es kann mehrere Empfangswarteschlangen pro Zielanwendung geben.
- *Fehlerwarteschlangen*, in der Terminologie von MQSeries Adapter Offering. Diese Warteschlangen werden verwendet, wenn eine Nachricht, die aus einer Empfangswarteschlange abgerufen wurde, nicht verarbeitet werden konnte.
- Optional: *Antwortwarteschlangen*. Sie werden in Verbindung mit der Methode `sendRequestResponse` verwendet.

MQSeries Adapter Offering verwendet bestimmte MQSeries-Funktionsmerkmale, zum Beispiel die folgenden Nachrichtenarten:

- Datagramme - Werden von der Methode `sendMsg` verwendet.
- Anforderungsnachrichten - Werden von der Methode `sendRequestResponse` verwendet.
- Antwortnachrichten - Werden von den Methoden `sendRequestResponse` und `sendResponse` verwendet.

MQSeries kann optional als anwendungsspezifische Schnittstelle auftreten.

Eine Liste der überprüften Konfigurationen von MQSeries und MQSeries Adapter Offering finden Sie in „Anhang B. Überprüfte Konfigurationen" auf Seite 121. Eine Liste der unterstützten Versionen von MQSeries und anderer Software finden Sie unter „Software" auf Seite 38.

Die Rolle von MQSeries Integrator

MQSeries Adapter Kernel kann optional zusammen mit MQSeries Integrator eingesetzt werden. MQSeries Integrator stellt über seine Broker-Funktionen bei Bedarf zusätzliche Verarbeitungsmöglichkeiten zur Verfügung:

- Komplexes Routing, d. h. ein Routing abhängig vom Inhalt des Nachrichten-Headers oder der Nachricht selbst. Das Routing kann sich dynamisch mit dem Nachrichteninhalt ändern. Informationen zum komplexen Routing und einfachen Routing finden Sie unter „Routing und Übermittlung" auf Seite 15.
- Datenkonvertierung, d. h. Ändern der Dokumentart.
- Datenänderung, d. h. Ändern des Nachrichteninhalts. Wenn beispielsweise die Quellenanwendung in einem Feld den Wert `jeder` bereitstellt, die Zielanwendung in diesem Feld jedoch den Wert `der` erwartet, wird der bereitgestellte Wert bei der Datenänderung durch den erwarteten Wert ersetzt.

Sie können MQSeries Integrator für die meisten Routing-Aufgaben in Ihrer Anwendungsumgebung verwenden; Sie können dementsprechend auch auf die Verwendung einiger der Routing-Funktionen von MQSeries Adapter Kernel verzichten.

Eine Liste der überprüften Konfigurationen von MQSeries Integrator und MQSeries Adapter Offering finden Sie in „Anhang B. Überprüfte Konfigurationen" auf Seite 121. Eine Liste der unterstützten Versionen von MQSeries Integrator und anderer Software finden Sie unter „Software" auf Seite 38.

Funktionsweise des Kernels

In diesem Abschnitt werden die folgenden Themen behandelt:

Komponenten der Kernel-Laufzeitumgebung

Die von Ihnen erstellten Adapter, der von Ihnen entwickelte anwendungsspezifische Code sowie MQSeries Adapter Kernel bilden, wenn Sie zusammen ausgeführt werden, die Funktionalität von MQSeries Adapter Offering.

Die Hauptkomponenten der Kernel-Laufzeitumgebung sind:

Quellenadapter
Software, die für eine bestimmte Anwendung erstellt wurde (in der Regel mit MQSeries Adapter Builder) und Daten dieser Anwendung in ein integriertes Format (mit den Nachrichtendaten) für die Nachrichtenübermittlung konvertiert. Quellenadapter werden in der Regel auf derselben Maschine wie die Quellenanwendung ausgeführt, entweder als Teil des Anwendungsprozesses oder als separater Prozess. Beispiele für Quellendaten sind Dateien, C-Strukturen und Java-Objekte. Ein Beispiel für ein integriertes Format für die Nachrichtenübermittlung ist XML; solche Formate basieren in der Regel auf einem Industriestandard wie OAG oder RosettaNet.

Nachrichtenträger
Ein Container für Metadaten, den der Kernel für die Kapselung der integrierten Nachricht und anderer von ihm benötigten Steuerdaten verwendet. Beispiele für Metadaten sind Anwendungs-IDs (logische IDs) der Quellen- und Zielanwendungen, die Nachrichtenkategorie (z. B. OAG), der Nachrichtenzweck (z. B. ″Einkaufsauftrag″) sowie die Übertragungsnachricht (mit den Nachrichtendaten), die gesendet oder empfangen wird.

Interner Adapter
Software, die für das Senden und Empfangen von Nachrichtenträgerobjekten verwendet wird. Der interne Adapter stellt für das Senden von Nachrichten ein einfaches Daten-Routing sowie Unterstützung für einen oder mehrere Transportmechanismen für Übertragungen zur Verfügung. Das einfache Daten-Routing basiert auf Metadaten im Nachrichtenträgerobjekt, z. B. Nachrichtenkategorie und Nachrichtenzweck. Nachrichten können asynchron oder synchron gesendet werden. Wenn der zu Grunde liegende Transportmechanismus für die Übertragung eine transaktionsgesteuerte Nachrichtenübermittlung unterstützt, können Nachrichten unter einer einphasigen Transaktionssteuerung gesendet werden. Der Umfang der Transaktionsunterstützung ist durch den Leistungsumfang des verwendeten Transportmechanismus begrenzt. Das Nachrichtenträgerobjekt wird in das Übertragungsnachrichtenformat integriert (Marshaling), das vom Transportmechanismus verwendet wird. Wenn der interne Adapter eine Übertragungsnachricht empfängt, erstellt er aus der Nachricht das Nachrichtenträgerobjekt im Zielformat (Unmarshaling).

Adapterdämon

Ein Prozess, der Exemplare von Adaptermanagern erstellt. Nachdem der Adapterdämon gestartet wurde, bleibt er aktiv. Für jede Zielanwendung kann es einen Adapterdämon für jede Empfangswarteschlange der Anwendung geben.

Adaptermanager

Ein Prozess, der jede einzelne Nachricht an den zugehörigen Zieladapter übermittelt. Jeder Adaptermanager verwaltet einen einzigen internen Adapter. Die Adaptermanager werden vom Adapterdämon erstellt und gestartet.

Durch den Einsatz mehrerer Adaptermanager ist eine *Multithread-Nachrichtenübermittlung* an Zieladapter möglich. Jeder Adaptermanager kann zusammen mit seinem internen Adapter einen Thread ausführen. Wenn nur ein Adaptermanager vorhanden ist, erfolgt die Nachrichtenübermittlung an den Zieladapter, und damit an die Zielanwendung, im Singlethread-Verfahren.

Neben der Verwaltung eines internen Adapters führt der Adaptermanager folgende Tasks aus:

- Er erstellt ein Exemplar des Trace-Clients, wenn die Trace-Funktion aktiviert ist.
- Er erstellt ein Exemplar der Anmeldeklasse, die für jede Zielanwendung spezifisch ist.
- Er wählt den Zieladapter abhängig vom Nachrichtenzweck und der Nachrichtenkategorie aus.
- Er sendet die Nachricht an den ausgewählten Zieladapter.
- Wenn er keine COMMIT-Operation ausführen kann, setzt er den Vorgang zurück (ROLLBACK-Operation), setzt eine Markierung für alle anderen Adaptermanager, die unter demselben Adapterdämon aktiv sind, und fährt sich selbst und seinen internen Adapter herunter. Dies weist darauf hin, dass es bei der Verarbeitung der Nachricht einen Fehler gegeben hat. Durch Herunterfahren aller Adaptermanager kann verhindert werden, dass andere Adaptermanager die fehlerhafte Nachricht mit demselben Ergebnis erneut verarbeiten.
- Erkennt er eine Markierung, die von einem anderen Adaptermanager gesetzt wurde, fährt er sich selbst und seinen internen Adapter herunter.

Zieladapter
Software, die für eine bestimmte Anwendung erstellt wurde (in der Regel mit MQSeries Adapter Builder) und Daten aus einem integrierten Format (mit den Nachrichtendaten) für die Nachrichtenübermittlung in den für die jeweilige Zielanwendung erforderlichen Datentyp konvertiert. Der Zieladapter ruft die benötigten APIs der Zielanwendung auf, um die Nachricht zu übermitteln. Zieladapter werden auf derselben Maschine wie die Anwendung oder der Anwendungs-Client ausgeführt.

Session-Bean-Adapter des Java-Service
Ein in der Programmiersprache Java erstellter EJB-Adapter, der unter einem EJB-Server, z. B. WebSphere Application Server, ausgeführt wird.

Konfigurationskomponente
Daten, die für die Auflösung logischer IDs in Objekte, z. B. Namen von Warteschlangen, verwendet werden. Die Konfigurationsdaten können in einer Datei oder in der LDAP-Struktur des Produkts WebSphere Business Integrator angegeben werden. Die Daten steuern die folgenden Aspekte der Kernel-Konfiguration:

- Marshaling und Routing von Nachrichten
- Überprüfen der Installation
- Übertragungsmodus
- Trace-Funktion

Eine vollständige Beschreibung der Konfigurationsdatei finden Sie unter „Die Konfigurationsdatei" auf Seite 75. Die Dokumentation zu WebSphere Business Integrator enthält Informationen zur Konfiguration dieses Produkts für die Zusammenarbeit mit dem Kernel.

Trace-Komponente
Eine Software, die Trace-Nachrichten schreibt. Die meisten anderen Komponenten des Kernels verwenden die Trace-Komponente. Eine Übersicht über die Trace-Funktion finden Sie unter „Trace-Funktion" auf Seite 33 und weitere Informationen zu Traces im Handbuch *Problem Determination Guide*.

Nachricht und Nachrichtenformat

In MQSeries und MQSeries Adapter Offering ist eine *Nachricht* eine Datensammlung, die von einem Programm an ein anderes Programm gesendet wird. Das Format der Nachricht zu einem bestimmten Zeitpunkt ist davon abhängig, an welcher Stelle des Nachrichtenflusses sie sich zu diesem Zeitpunkt befindet. MQSeries Adapter Kernel unterscheidet drei Arten von Nachrichten:

- *Integrierte Nachricht* - Eine Nachricht, die aus Daten einer Quellenanwendung besteht, die vor dem Senden an eine Zielanwendung in ein anderes Format wie z. B. XML konvertiert wurden. Die integrierte Nachricht wird als der Nachrichteninhalt in das Nachrichtenträgerobjekt eingefügt. XML ist ein Standard für die Darstellung von Daten. Handelt es sich um ein XML-Format, wird das Format durch eine *Document Type Definition* (DTD) definiert. Eine DTD besteht aus einer oder mehreren Dateien, die eine formale Definition eines Dokuments, in diesem Fall des Nachrichteninhalts, enthalten. Auch wenn es dringend empfohlen wird, muss der Nachrichteninhalt kein anwendungsneutrales Format aufweisen. Das Format des Nachrichteninhalts kann systemspezifisch oder anderweitig spezifiziert sein; ein solches Format wird jedoch nicht empfohlen.

  Business Object Documents (BODs) können von MQSeries Adapter Offering verwendet werden, um Nachrichteninhalte in integrierten Nachrichten zu definieren. Ein BOD ist eine Abbildung eines Standardgeschäftsprozesses, der innerhalb einer Organisation oder zwischen Organisationen abläuft. Beispiele für solche Prozesse: Hinzufügen eines Einkaufsauftrags, Anzeigen der Produktverfügbarkeit oder Hinzufügen eines Verkaufsauftrags. BODs werden als XML-Dokumente von der Open Applications Group (OAG) definiert. Die Verwendung von BODs wird empfohlen, ist jedoch nicht erforderlich.
- *Nachrichtenträgerobjekt* - Ein Objekt, das die integrierte Nachricht und zusätzliche Header-Metadaten (bestehend aus MQSeries Adapter Kernel-spezifischen Steuerwerten) enthält. Der Quellenadapter erstellt das Nachrichtenträgerobjekt und fügt die zugehörigen Steuerinformationen sowie, wenn eine integrierte Nachricht gesendet werden soll, die Nachrichtendaten ein. Zieladapter empfangen das Nachrichtenträgerobjekt, rufen die Nachrichtendaten ab und konvertieren sie in das spezifische Datenformat der Zielanwendung. Quellenadapter und Zieladapter werden mit MQSeries Adapter Builder erstellt.
- *Übertragungsnachricht* - Alle den Transportmechanismus für die Übertragung betreffenden Informationen sowie das Nachrichtenträgerobjekt, die in das spezifische Nachrichtenübermittlungsformat des für die Übertragung verwendeten Transportmechanismus konvertiert wurden. Von einigen Transportmechanismen für Übertragungen werden mehrere Nachrichtenübermittlungsformate unterstützt. In der Regel betrachtet der Transportmechanismus die Header-Metadaten des Kernels zusammen mit der Übertra-

gungsnachricht als Anwendungsdaten. Weitere Informationen finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115. Ein MQSeries-Transport besteht beispielsweise aus dem MQSeries-spezifischen Nachrichten-Header plus dem durch das Marshaling erstellten Nachrichtenträgerobjekt. Spezifische MQSeries-Formate sind:

– Der MQSeries-Nachrichten-Header, der von MQSeries hinzugefügt wird.
– Wenn MQSeries Integrator verwendet wird, der versionsspezifische Nachrichten-Header:
  - Der Nachrichten-Header von MQSeries Integrator Version 1, wenn MQSeries Integrator Version 1.1 verwendet wird.
  - Der Nachrichten-Header von MQSeries Integrator Version 2, wenn MQSeries Integrator Version 2 verwendet wird.
– Die Kernel-spezifischen Header-Metadaten mit den Steuerwerten.
– Die integrierte Nachricht (die Nachrichtendaten)

Eine Liste der relevanten Felder in den Nachrichten-Headern von MQSeries Adapter Offering mit einer Beschreibung finden Sie in „Anhang C. Nachrichten-Header" auf Seite 123.

Routing und Übermittlung

Der Kernel ruft jede einzelne Nachricht vom Quellenadapter ab und übermittelt sie an den zugehörigen Zieladapter. Dieses Routing erfolgt in zwei Stufen:

1. Die Quellenverarbeitung des Kernels reiht die Nachricht in die zugehörige Nachrichtenwarteschlange ein.
2. Die Zielverarbeitung des Kernels ruft die Nachricht aus der Nachrichtenwarteschlange ab und ruft den zugehörigen Zieladapter auf.

Der Routing-Verlauf wird von mehreren Faktoren bestimmt:

- Nachrichtenwarteschlangen. Als wesentliche Voraussetzung müssen Nachrichtenwarteschlangen zur Unterstützung des Routings von MQSeries Adapter Offering konfiguriert werden.
- Die Nachrichtensteuerungswerte in der Nachricht. Dazu gehören: logische Quellen-ID, logische Ziel-ID, logische Antwort-ID, Nachrichtenkategorie, Nachrichtenzweck, Transaktions-ID, Nachrichten-ID, Bestätigungsanforderung und Zeitmarken. Weitere Informationen finden Sie unter „Nachrichtensteuerungswerte" auf Seite 18. Die logische Ziel-ID in der Nachricht kann den entsprechenden Wert in der Konfigurationsdatei des Kernels überschreiben. Das Routing kann sich dynamisch mit diesen Nachrichtensteuerungswerten in jedem Nachrichten-Header ändern. Der Inhalt der Nachrichtendaten (der integrierten Nachricht) hat dagegen keine Auswirkung auf das Routing.

- Die Nachrichtensteuerungswerte in der Konfigurationsdatei des Kernels. Die Datei kann logische Ziel-IDs, Warteschlangennamen und zugeordnete Zieladapter angeben. Sie können die Konfiguration festlegen und ändern, indem Sie diese Datei editieren. Weitere Informationen finden Sie unter „Die Konfigurationsdatei" auf Seite 75.
- Optional kann MQSeries Integrator mit seinen Broker-Funktionen für Nachrichten eingesetzt werden, z. B. für komplexes Routing. Das Routing kann sich dynamisch mit dem Nachrichteninhalt ändern. Weitere Informationen finden Sie unter „Die Rolle von MQSeries Integrator" auf Seite 10. MQSeries Adapter Offering selbst kann nur ein einfaches Routing durchführen. Einfaches Routing beruht auf einer Kombination aus den Nachrichtensteuerungswerten in der Nachricht mit den zugeordneten Nachrichtensteuerungswerten in der Konfigurationsdatei. Der Nachrichteninhalt wird dabei nicht berücksichtigt.

Der Kernel kann aufgefordert werden, die Übermittlung einer Nachricht zu bestätigen. Dabei handelt es sich um eine Empfangsbestätigung auf Anwendungsebene.

Laufzeitverarbeitung

In diesem Abschnitt wird die Laufzeitverarbeitung im Einzelnen beschrieben, d. h. wie der Kernel in einer typischen Produktionsumgebung eine Nachricht sendet, weiterleitet, Traces durchführt und eine Nachricht übermittelt. Abb. 2 zeigt ein Diagramm der Laufzeitverarbeitung.

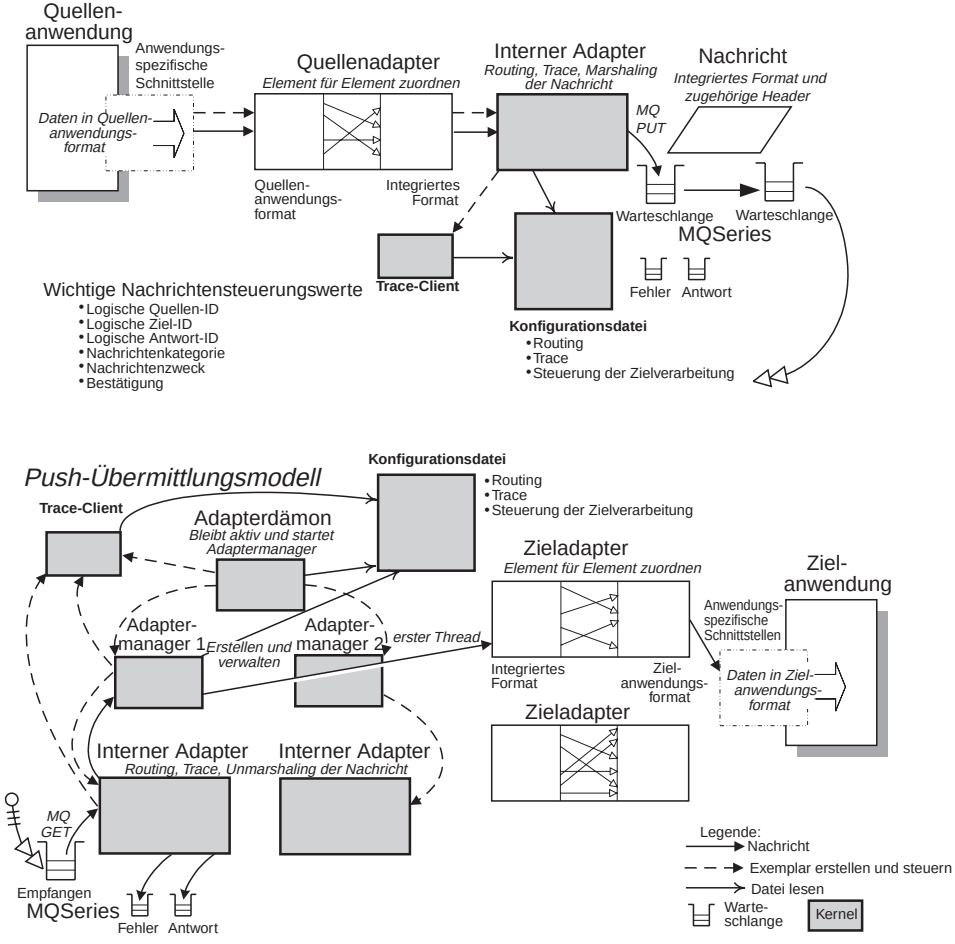


Abbildung 2. Marshaling, Senden, Weiterleiten einer Nachricht und Durchführen von Traces - Übersicht.

Quellenverarbeitung des Kernels

Dieser Abschnitt beschreibt die Laufzeitverarbeitung in der Quellenverarbeitungsschicht des Kernels, d. h. wie Daten von einer Quellenanwendung über einen Quellenadapter zu einem Transportmechanismus für die Übertragung fließen. „Zielverarbeitung des Kernels" auf Seite 23 beschreibt, wie Daten vom Transportmechanismus für die Übertragung an ihr Ziel gelangen.

1. Der Quellenadapter erhält über eine anwendungsspezifische Schnittstelle eine Nachricht von der Quellenanwendung. In der Regel wird der Quellenadapter von der anwendungsspezifischen Schnittstelle aufgerufen.
2. Der Quellenadapter führt die Funktionen aus, die bei seiner Erstellung mit MQSeries Adapter Builder definiert wurden. Seine Standardaufgabe besteht darin, die Daten aus dem Format der Quellenanwendung in ein anwendungsneutrales, integriertes Format (für den Nachrichteninhalt) zu konvertieren.

   Darüber hinaus fügt der Quellenadapter verschiedene Nachrichtensteuerungswerte in den MQSeries Adapter Kernel-Header ein, die eine Art Briefumschlag für die Nachricht bilden. Die ersten fünf Nachrichtensteuerungswerte betreffen das Marshaling und Routing und der letzte Wert die Empfangsbestätigung.

   Nachrichtensteuerungswerte

   Logische Quellen-ID
   Logische ID der Quellenanwendung. Dieser Wert muss in jeder Nachricht stehen.

   Logische Ziel-ID
   Logische ID der Zielanwendung. Wenn dieser Wert nicht in der Nachricht steht, werden Standardwerte aus der Konfigurationsdatei verwendet. Die Konfigurationsdatei kann mehrere logische Ziel-IDs enthalten, die verwendet werden können, wenn in der Nachricht keine Werte angegeben sind.

   Logische Antwort-ID
   Die logische ID der Anwendung, an die Antworten gesendet werden, wenn eine Antwort angefordert wird. Standardmäßig ist dies die logische Quellen-ID in der Nachricht.

   Nachrichtenkategorie
   Dieser Wert beschreibt die Anwendungsart der Nachricht, z. B. OAG oder RosettaNet. Dieser Wert muss in jeder Nachricht stehen.

Nachrichtenzweck
Dieser Wert beschreibt den Zweck der Nachricht, z. B. „Hinzufügen eines Verkaufsauftrags" oder „Synchronisieren des Inventars". Dieser Wert muss in jeder Nachricht stehen.

Bestätigung angefordert
Dieser Wert gibt an, ob die Quellenanwendung eine Antwort anfordert. Für die Antwort kann eines der folgenden Formate verwendet werden:

- Antwortdaten der Zielanwendung
- Eine BOD-Bestätigung nach OAG-Standard (Confirm_BOD-Nachricht)

  Anmerkung: Die Confirm_BOD-Nachricht hat ein von der OAG (Open Applications Group) vordefiniertes Format. Sie enthält als Nachrichtenkategorie den Wert `OAG` und als Nachrichtenzweck den Wert `CONFIRM_BOD_003`. Sie kann darüber hinaus auch Daten enthalten.

Bei dieser Antwort handelt es sich um eine Empfangsbestätigung auf Anwendungsebene.

Wenn der Kernel die Nachricht mit Hilfe der Methode `sendRequestResponse` sendet, wird nur die erste Antwort, die von der Methode `sendRequestResponse` empfangen wird, verwendet. Wurde die ursprüngliche Nachricht an mehrere Zieladressen gesendet und eine Antwort angefordert (was nicht empfohlen wird), wird nur die erste Antwort an die Quellenanwendung zurückgeschickt.

Die Empfangsbestätigung ist standardmäßig inaktiviert, d. h. es wird keine Antwort angefordert oder gesendet.

3. Der Quellenadapter initialisiert den internen Adapter und übergibt ihm Folgendes:
   - Die logische ID der Anwendung, unter der der Quellenadapter aktiv ist.
   - Das Nachrichtenträgerobjekt mit den Nachrichtensteuerungswerten und dem Nachrichteninhalt.
4. Der interne Adapter liest die Konfigurationsdatei, um festzustellen, ob für die logische Quellen-ID die Trace-Funktion aktiviert ist. Ist dies der Fall, startet der interne Adapter ein Exemplar des Trace-Clients.

5. Der Trace-Client liest die Konfigurationsdatei, um festzustellen, welche Trace-Stufe verwendet werden soll, und um andere Werte abzurufen. Der Trace-Client filtert Trace-Nachrichten entsprechend der angegebenen Trace-Stufe heraus. Eine Übersicht über die Trace-Funktion finden Sie unter „Trace-Funktion" auf Seite 33; weitere Informationen zu Traces können Sie dem Handbuch *Problem Determination Guide* entnehmen.
6. Der interne Adapter sucht im Nachrichtenobjekt nach einer logischen Ziel-ID. Ist eine vorhanden, wird sie verwendet.
   - Ist die logische Ziel-ID nicht angegeben, sucht der interne Adapter abhängig von der logischen Quellen-ID, der Nachrichtenkategorie und dem Nachrichtenzweck in der Konfigurationsdatei nach dem Standardwert für die logische Ziel-ID.
   - Anhand der logischen Quellen-ID führt der interne Adapter eine aus mehreren Schritten bestehende Suche nach der Nachrichtenkategorie und dem Nachrichtenzweck in der Konfigurationsdatei in der folgenden Reihenfolge durch:
     a. Er sucht spezifische Werte für die Nachrichtenkategorie und den Nachrichtenzweck.
     b. Er sucht einen spezifischen Wert für die Nachrichtenkategorie und einen Standardwert für den Nachrichtenzweck.
     c. Er sucht einen Standardwert für die Nachrichtenkategorie und einen spezifischen Wert für den Nachrichtenzweck.
     d. Er sucht Standardwerte für die Nachrichtenkategorie und den Nachrichtenzweck.

   Anmerkung: Der Kernel verwendet immer diese mehrstufige Suchfunktion, wenn er in der Konfigurationsdatei nach Werten sucht.
7. Für jede logische Ziel-ID, die im vorhergehenden Schritt gefunden wurde, ermittelt der interne Adapter abhängig von der logischen Ziel-ID, der Nachrichtenkategorie und dem Nachrichtenzweck den *Übertragungsmodus*. Die folgenden Übertragungsmodi werden unterstützt:

   MQPP Der Kernel transportiert Nachrichten mit Hilfe von MQSeries-Grundfunktionen.

   MQRFH1 Der Kernel transportiert Nachrichten mit Hilfe von MQSeries und führt das Nachrichten-Brokering mit Hilfe von MQSeries Integrator Version 1.1 aus.

   MQRFH2 Der Kernel transportiert Nachrichten mit Hilfe von MQSeries und führt das Nachrichten-Brokering mit Hilfe von MQSeries Integrator Version 2 aus.

MQBD Der Kernel transportiert Nachrichten mit Hilfe von MQSeries-Grundfunktionen, sendet und empfängt aber nur Nachrichtendaten.

MQ Der Kernel transportiert Nachrichten mit Hilfe von MQSeries.

JMS Der Kernel transportiert Nachrichten mit Hilfe von Java Message Service (JMS).

FILE Der Kernel stellt Nachrichten in eine Datei und ruft sie aus einer Datei ab. Dieser Modus wird nur zu Diagnosezwecken zur Verfügung gestellt.

Jeder Übertragungsmodus verwendet eine andere Nachrichtenstruktur. Weitere Informationen finden Sie unter „Nachricht und Nachrichtenformat" auf Seite 14. Weitere Informationen zum Übertragungsmodus finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115.

Anmerkung: Bei Verwendung von MQSeries Integrator muss der Zielort, an den MQSeries Integrator die Nachricht sendet, denselben Übertragungsmodus zum Empfang von Nachrichten wie MQSeries Integrator verwenden, damit er Nachrichten empfangen kann.

8. Abhängig vom Übertragungsmodus erstellt der interne Adapter für sich selbst ein Exemplar einer Unterklasse für die Bearbeitung der Nachricht. Diese Unterklasse wird als *logischer Nachrichtenservice* bezeichnet. Für jeden Übertragungsmodus gibt es eine eigene Unterklasse, d. h. einen eigenen logischen Nachrichtenservice.

   Der interne Adapter übergibt die logischen Ziel-IDs, die Nachrichtenkategorie und den Nachrichtenzweck an den logischen Nachrichtenservice.

9. Der logische Nachrichtenservice erhält so die Parameter, die er für das Senden der Nachricht benötigt. Wenn der Übertragungsmodus zum Beispiel MQPP ist, enthalten die Parameter das Format und die Namen der Empfangs-, Antwort- und Fehlerwarteschlangen. Abhängig von der logischen Ziel-ID, der Nachrichtenkategorie und dem Nachrichtenzweck, die an ihn übergeben wurden, führt der logische Nachrichtenservice eine aus mehreren Schritten bestehende Suche in der Konfigurationsdatei aus:
   a. Er sucht spezifische Werte für die Nachrichtenkategorie und den Nachrichtenzweck.
   b. Er sucht einen spezifischen Wert für die Nachrichtenkategorie und einen Standardwert für den Nachrichtenzweck.
   c. Er sucht einen Standardwert für die Nachrichtenkategorie und einen spezifischen Wert für den Nachrichtenzweck.

d. Er sucht Standardwerte für die Nachrichtenkategorie und den Nachrichtenzweck.

Zu diesem Zeitpunkt verfügt der logische Nachrichtenservice über alle Informationen, die er für das Weiterleiten und Marshaling der Nachricht benötigt.

10. Der logische Nachrichtenservice führt die folgenden Tasks aus:
    - Er bereitet die Nachricht dem Übertragungsmodus und -format entsprechend auf (Marshaling). Jeder Übertragungsmodus verwendet ein Standardformat, sofern kein anderes Format angegeben wurde. Für den Übertragungsmodus MQRFH2 zum Beispiel erstellt der logische Nachrichtenservice entsprechende Header und strukturiert die Nachricht für den Transport durch MQSeries und für die Broker-Verarbeitung durch MQSeries Integrator Version 2.
    - Er sendet die Nachricht. Im Übertragungsmodus MQRFH2 zum Beispiel reiht er die Nachricht in die zugehörige MQSeries-Nachrichtenwarteschlange ein.
11. Zum Senden der Nachricht stehen zwei Methoden zur Verfügung:
    - Wenn der interne Adapter die Nachricht mit der Methode `sendMsg` sendet, wartet der interne Adapter nicht auf eine Antwort.
    - Wenn der interne Adapter die Nachricht mit der Methode `sendRequestResponse` sendet, wartet der logische Nachrichtenservice auf die Antwort. Der interne Adapter überwacht über den logischen Nachrichtenservice die Antwortwarteschlange, bis der Wert für die *Zeitlimitüberschreitung bei Empfang*, der in der Konfigurationsdatei festgelegt ist, erreicht ist.

      Die Zeitlimitüberschreitung bei Empfang ist von der ID der Quellenanwendung, der Nachrichtenkategorie und dem Nachrichtenzweck abhängig.
      - Wenn eine Bestätigung empfangen wird, gibt der interne Adapter die Nachricht zurück.
      - Wenn bis zur Zeitlimitüberschreitung bei Empfang keine Bestätigung empfangen wird, gibt der interne Adapter keine Nachricht zurück.
12. MQSeries oder ein anderes Nachrichtenübermittlungsprodukt transportiert die Nachricht entsprechend ihrer Konfiguration. Optional werden von MQSeries Integrator Broker-Services ausgeführt. Weitere Informationen finden Sie unter „Die Rolle von MQSeries Integrator" auf Seite 10.
13. Wenn der Quellenadapter den internen Adapter nicht mehr benötigt, schließt er ihn, um Ressourcen freizugeben.

Zielverarbeitung des Kernels

Dieser Abschnitt beschreibt die Verwendung von MQSeries Adapter Kernel in einer Standalone-Umgebung zum Empfangen und Verarbeiten von Nachrichten in der Zielverarbeitungsschicht und gibt eine allgemeine Übersicht über die Verwendung des Kernels in Verbindung mit WebSphere Application Server. Unter „MQSeries Adapter Kernel mit WebSphere Business Integrator und WebSphere Application Server verwenden" auf Seite 34 wird die Verwendung des Kernels mit JMS, mit der Komponente JMS Listener von WebSphere Business Integrator und mit WebSphere Application Server in der Zielverarbeitungsschicht des Kernels behandelt. Dieser Abschnitt beschreibt das Push-Übermittlungsmodell, bei dem der Kernel für das Einleiten und Verwalten der Nachrichtenübermittlung an die Zielanwendung zuständig ist. Eine Kurzbeschreibung der Modelle finden Sie unter „Übermittlungsmodelle" auf Seite 148.

Adaptermanager - Übersicht

Dieser Abschnitt beschreibt die Struktur und Funktionsweise der Adaptermanager von MQSeries Adapter Kernel. Eine der Grundannahmen der MQSeries Adapter Kernel-Architektur ist, dass Zielanwendungen nicht aktiv an der Integration von Datenflüssen mit anderen Anwendungen beteiligt sind, d. h., dass Anwendungen in der Regel nicht aktiv Nachrichten zur Verarbeitung abrufen. Das bedeutet andererseits, dass Nachrichtendaten aktiv an die Zielanwendung übergeben werden müssen. Adaptermanager übergeben Nachrichtendaten an eine Anwendung oder einen anderen Service, indem sie eine Reihe von Serviceschnittstellenarten auswählen und aufrufen. MQSeries Adapter Kernel unterstützt Adaptermanager, die entweder unter einem Standalone-Dämon (Adapterdämon) oder einem Enterprise JavaBeans-Anwendungsserver (momentan IBM WebSphere Application Server Advanced Edition) ausgeführt werden. Nachrichten erreichen den Adaptermanager auf verschiedene Art und Weise, je nachdem, welche Art von Zielumgebung vorhanden ist. Wenn ein Standalone-Adapterdämon verwendet wird, können unter ihm mehrere Adaptermanager aktiv sein, die über den internen Adapter Nachrichten empfangen. Wird ein EJB-Server verwendet, empfängt die Komponente 'JMS Listener' Nachrichten und übergibt sie an eine Adaptermanager-Nachrichten-Bean (die manchmal auch als nachrichtengesteuerte Adaptermanager-Bean bezeichnet wird).

Unabhängig von der verwendeten Zielumgebung leitet der Adaptermanager die empfangene Nachricht an den richtigen Zieladapter weiter. Der Zieladapter führt dann die erforderliche Verarbeitung durch und übermittelt die Nachricht an die Zielanwendung. Zieladapter werden für die Zusammenarbeit mit bestimmten Zielanwendungen erstellt. Der Adapterdämon, Anwendungsserver, Standalone-Adaptermanager und die Adaptermanager-Nachrichten-Bean sind dagegen nicht an eine bestimmte Quellen- oder Zielanwendung geknüpft.

Der Adaptermanager unterstützt zwei Arten von Zieladapterschnittstellen: Enterprice Access Builder (EAB)-Befehlsadapter und Session-Beans des EJB-Service. Jeder Adaptertyp verfügt über eine Steuerroutine, die die entsprechende Umgebung einrichtet, auf alle zusätzlichen, vom Adapter benötigten Konfigurationsdaten zugreift und weitere untergeordnete Tasks ausführt, die für den Betrieb des Adapters erforderlich sind. Die jeweils verwendete Steuerroutine ist vom Adaptertyp, der in der Konfigurationsdatei aufgelistet ist, abhängig. Die Steuerroutinen der beiden Adaptertypen führen die folgenden zusätzlichen Tasks aus:

- Die EAB-Steuerroutine enthält eine Anmeldeklasse, die Verbindungsinformationen für den Zieladapter bereitstellt; außerdem initialisiert sie die IBM Common Connector Framework (CCF)-Laufzeitumgebung. Der Anmeldeklasse wird die logische ID der Zielanwendung übergeben, mit deren Hilfe sie die anwendungsspezifischen Anmeldedaten abruft.
- Die EJB-Steuerroutine stellt eine Verbindung über die Schnittstelle Java Naming and Directory Interface™ (JNDI) her und ruft dann die ferne Schnittstelle der Service-Session-Bean sowie weitere Informationen ab, die für den Zugriff auf die Service-Session-Bean erforderlich sind.

Ein Adaptermanager unter einem Standalone-Adapterdämon führt folgende Basisverarbeitung durch:

1. Bei seinem Start erstellt der Adapterdämon ein oder mehrere Exemplare des Standalone-Adaptermanagers, entsprechend den Informationen in der Konfigurationsdatei des Kernels. Die logische ID der Anwendung und gegebenenfalls auch Werte für die Nachrichtenkategorie und den Nachrichtenzweck werden an den Adapterdämon übergeben. Mit Hilfe der Werte für die Nachrichtenkategorie und den Nachrichtenzweck werden zusätzliche Konfigurationswerte abgerufen.
2. Jeder Standalone-Adaptermanager führt die folgenden Tasks aus:
   a. Der Adaptermanager erstellt ein Exemplar des internen Adapters und startet das Empfangen von Nachrichten. Jede Nachricht wird transaktionsgesteuert empfangen und als Nachrichtenträgerobjekt an den Adaptermanager zurückgegeben.
   b. Für jede empfangene Nachricht ruft der Adaptermanager aus der Konfigurationsdatei die Befehlsart des Zieladapters für die Verarbeitung der Nachricht ab und ruft dann die zugehörige Steuerroutine für die Befehlsart auf.
   c. Die Steuerroutine ruft aus der Konfigurationsdatei alle zusätzlichen Informationen ab, die sie benötigt, um ein Exemplar des Zieladapters zu erstellen. Sie erstellt ein Exemplar des Zieladapters und übergibt ihm die Nachricht.

d. Wenn die Nachricht erfolgreich verarbeitet wird (d. h. ohne Ausnahmebedingungen, Fehler oder falsche Rückgabedaten), wird sie von der Eingangsnachrichtenwarteschlange festgeschrieben. Ist die Verarbeitung nicht erfolgreich, wird die Nachricht in eine Fehlerwarteschlange eingereiht. Wenn die Nachricht nicht erfolgreich verarbeitet wird und auch nicht in eine Fehlerwarteschlange eingereiht werden kann, wird sie zurückgesetzt (ROLLBACK-Operation), und alle Adaptermanager werden beendet.

Ein Adaptermanager unter WebSphere Application Server führt folgende Basisverarbeitung durch:

1. Ein JMS Listener-Prozess, der mit dem EJB-Server von WebSphere Application Server Advanced Edition zusammenarbeitet, empfängt eine JMS-Nachricht. Er ruft dann eine Adaptermanager-Nachrichten-Bean auf, um die Nachricht zu verarbeiten. Die logische ID der Anwendung und gegebenenfalls auch Werte für die Nachrichtenkategorie und den Nachrichtenzweck sind Teil der Umgebung der Adaptermanager-Nachrichten-Bean. Mit Hilfe der Werte für die Nachrichtenkategorie und den Nachrichtenzweck werden zusätzliche Konfigurationswerte abgerufen.
2. Jede Adaptermanager-Nachrichten-Bean führt die folgenden Tasks aus:
   a. Die Adaptermanager-Nachrichten-Bean erstellt ein Exemplar des internen Adapters und verwendet die Methode `receiveMsg` des internen Adapters, wobei sie ihr die JMS-Nachricht übergibt. Der interne Adapter konvertiert die JMS-Nachricht in ein Nachrichtenobjekt und gibt das Nachrichtenträgerobjekt zurück.
   b. Für jedes empfangene Nachrichtenträgerobjekt ruft der Adaptermanager aus der Konfigurationsdatei die Befehlsart des Zieladapters für die Verarbeitung des Nachrichtenträgerobjekts ab und ruft dann die zugehörige Steuerroutine für die Befehlsart auf.
   c. Die Steuerroutine ruft aus der Konfigurationsdatei alle zusätzlichen Informationen ab, die sie benötigt, um ein Exemplar des Zieladapters zu erstellen. Sie erstellt ein Exemplar des Zieladapters und übergibt ihm das Nachrichtenträgerobjekt.
   d. Wenn das Nachrichtenträgerobjekt erfolgreich verarbeitet wird (d. h. ohne Ausnahmebedingungen, Fehler oder falsche Rückgabedaten), wird es von der Eingangsnachrichtenwarteschlange festgeschrieben. Ist die Verarbeitung nicht erfolgreich, wird das Nachrichtenträgerobjekt in eine Fehlerwarteschlange eingereiht. Wenn die Nachricht nicht erfolgreich verarbeitet wird und auch nicht in eine Fehlerwarteschlange eingereiht werden kann, wird sie zurückgesetzt (ROLLBACK-Operation), und alle Adaptermanager werden beendet.

Die Laufzeitverarbeitung in der Zielverarbeitungsschicht sieht bei Verwendung eines Adapterdämons und eines Standalone-Adaptermanagers wie folgt aus:

1. Für jede Empfangswarteschlange der Zielanwendung gibt es einen Adapterdämon. Der Adapterdämon wird gestartet.

   Bei seinem Start erhält er einen Namen, der als Anwendungs-ID verwendet wird. Die Namen der einzelnen Adapterdämonen basieren auf der logischen Ziel-ID, d. h. der logischen ID der Zielanwendung. Wenn der Adapterdämon zum Beispiel einer Zielanwendung mit der logischen Ziel-ID `ABC` zugeordnet ist, erhält der Adapterdämon den Namen `ABCdaemon`.

   Andere Parameter, die beim Starten des Adapterdämons an ihn übergeben werden können, sind die Nachrichtenkategorie und der Nachrichtenzweck. Mit Hilfe dieser Parameter ermittelt der interne Adapter später den Übertragungsmodus und die Empfangswarteschlange für ankommende Nachrichten.

   Anweisungen zum Starten des Adapterdämons finden Sie unter „Kernel starten" auf Seite 105.

2. Bei seinem Start liest der Adapterdämon die Konfigurationsdatei, um festzustellen, ob die Trace-Funktion für seinen Namen aktiviert ist. Ist die Trace-Funktion aktiviert, erstellt der Adapterdämon ein Exemplar des Trace-Clients.

   Weitere Informationen zu Traces finden Sie im Handbuch *Problem Determination Guide*.

3. Bei seinem Start erstellt der Adapterdämon den ersten Adaptermanager und übergibt ihm seinen eigenen Namen sowie die Nachrichtenkategorie und den Nachrichtenzweck.

4. Der erste Adaptermanager liest die Konfigurationsdatei, um festzustellen, ob für diesen Adapterdämonnamen die Trace-Funktion aktiviert ist. Ist dies der Fall, erstellt der erste Adaptermanager ein Exemplar des Trace-Clients, der dann die Konfigurationsdatei liest, um die zu verwendende Trace-Stufe festzustellen. Eine Liste mit gültigen Trace-Stufen finden Sie im Handbuch *Problem Determination Guide*.

5. Der erste Adaptermanager sucht abhängig von der Anwendungs-ID des Adapterdämons in der Konfigurationsdatei nach den Werten, die die minimale Anzahl der zu erstellenden und zu startenden Adaptermanager angeben.

   Der erste Adaptermanager sucht außerdem nach der *ID der abhängigen Anwendung*. Die ID der abhängigen Anwendung ist der Name der Anwendung, für die der Adaptermanager Services bereitstellt. Sie wird später an den internen Adapter übergeben.

6. Der Adapterdämon ruft vom ersten Adaptermanager die Mindestanzahl der Adaptermanager ab.

7. Der Adapterdämon startet den ersten Adaptermanager und erstellt und startet anschließend die Mindestanzahl von Adaptermanagern.

   Durch den Einsatz mehrerer Adaptermanager ist eine Multithread-Nachrichtenübermittlung an Zieladapter möglich. Jeder Adaptermanager kann zusammen mit seinem internen Adapter einen Thread ausführen. Wenn nur ein Adaptermanager vorhanden ist, erfolgt die Nachrichtenübermittlung an den Zieladapter, und damit an die Zielanwendung, im Singlethread-Verfahren.

   Auf AIX-Systemen stehen zwei Zeitplanungsrichtlinien für Threads zur Verfügung: prozessbasierte Zeitplanung und systembasierte Zeitplanung. Bei der prozessbasierten Zeitplanung (die Standardeinstellung) werden alle Benutzer-Threads einem Pool von Betriebssystemkernel-Threads zugeordnet und in einem Pool virtueller Prozessoren ausgeführt. Bei der systembasierten Zeitplanung wird jeder Benutzer-Thread einem einzelnen Betriebssystemkernel-Thread zugeordnet und auf einem einzelnen virtuellen Prozessor ausgeführt. Wenn Sie C-Quellenadapter verwenden, die von ausführbaren C-Dateien unter AIX aufgerufen werden, müssen Sie die systembasierte Zeitplanung verwenden. Informationen zum Einstellen der Thread-Zeitplanungsrichtlinie unter AIX finden Sie in Schritt 6 auf Seite 52.

   Beachten Sie, dass unter Windows, HP-UX, Solaris und OS/400 nur prozessbasierte Zeitplanung unterstützt wird.

Die anderen Adaptermanager führen, ebenso wie der erste Adaptermanager, die folgenden Schritte aus:

8. Jeder Adaptermanager erstellt ein Exemplar des ihm zugeordneten internen Adapters. Jedem Adaptermanager ist ein interner Adapter zugeordnet. Die ID der abhängigen Anwendung, die Nachrichtenkategorie und der Nachrichtenzweck werden an den internen Adapter übergeben. Der interne Adapter ermittelt mit Hilfe dieser drei Werte den Übertragungsmodus und mit Hilfe des logischen Nachrichtenservices das Format und die Empfangswarteschlange für ankommende Nachrichten. Dieser Prozess ähnelt dem Prozess für das Senden von Nachrichten.
9. Der interne Adapter ruft die Übertragungsnachricht aus der Empfangswarteschlange ab, wobei eine COMMIT-Operation ausgeführt wird, und konvertiert sie in ein Nachrichtenträgerobjekt. Er entfernt alle Header, die für den Transportmechanismus für die Übertragung spezifisch sind, mit Ausnahme des Kernel-eigenen Headers.
10. Der interne Adapter übergibt das Nachrichtenträgerobjekt an den Adaptermanager, der die Werte für die Nachrichtenkategorie, den Nachrichtenzweck und die Bestätigungsanforderung aus dem Kernel-eigenen Header der Nachricht liest.

Abhängig von der ID der abhängigen Anwendung, der Nachrichtenkategorie und dem Nachrichtenzweck führt der Adaptermanager in der Konfigurationsdatei eine aus mehreren Schritten bestehende Suche nach der aufzurufenden Zielbefehlsart in der folgenden Reihenfolge aus:

a. Er sucht spezifische Werte für die Nachrichtenkategorie und den Nachrichtenzweck.
b. Er sucht einen spezifischen Wert für die Nachrichtenkategorie und einen Standardwert für den Nachrichtenzweck.
c. Er sucht einen Standardwert für die Nachrichtenkategorie und einen spezifischen Wert für den Nachrichtenzweck.
d. Er sucht Standardwerte für die Nachrichtenkategorie und den Nachrichtenzweck.

Abhängig von der Zielbefehlsart ermittelt der Adaptermanager die zugehörige Steuerroutine des Zieladaptertyps, eine Java-Klasse, von der dieser bestimmte Adaptertyp verarbeitet wird. Er erstellt ein Exemplar dieses bestimmten Zieladapters.

11. Es gibt zwei Arten von Adaptertypsteuerroutinen: Steuerroutinen von EAB-Befehlszieladaptern und Steuerroutinen von Session-Bean-Zieladaptern des EJB-Service. Die verschiedenen Arten von Adaptersteuerroutinen arbeiten folgendermaßen:

    Anmerkung: Die Steuerroutine für den Session-Bean-Zieladapter des EJB-Service wird nur in Verbindung mit WebSphere Business Integrator unter WebSphere Application Server auf einer Windows NT-Plattform unterstützt.

    - Wenn eine Steuerroutine des EAB-Befehlszieladapters aufgerufen wird, startet sie die Common Connector Framework (CCF)-Umgebung, definiert eine Anmeldeklasse mit einem Namen, der aus der Konfigurationsdatei abgerufen wird, und ruft den EAB-Zieladapter mit dem aus der Konfigurationsdatei abgerufenen Namen auf.
    - Ein Session-Bean-Zieladapter des EJB-Service muss mit WebSphere Business Integrator und WebSphere Application Server kommunizieren, um die zugehörigen Konfigurationsdaten abzurufen und die EJB-Service-Session-Bean aufzurufen. Unter „MQSeries Adapter Kernel mit WebSphere Business Integrator und WebSphere Application Server verwenden” auf Seite 34 wird die Verwendung des Kernels zusammen mit JMS, mit der Komponente JMS Listener von WebSphere Business Integrator und mit WebSphere Application Server in der Zielverarbeitungsschicht des Kernels behandelt.

12. Jeder Adaptertyp hat eine andere Schnittstelle und benötigt Unterstützungsklassen:
    - Ein EAB-Zieladapterbefehl kann drei Methoden aufrufen; diese Methoden werden in der folgenden Reihenfolge ausgeführt:
        a. Die Methode *set message input*, mit der die zu verarbeitende Nachricht in den Zieladapter gestellt wird.
        b. Die Methode *execute*, von der die Nachricht, die zuvor von der Methode 'set message input' in den Zieladapter gestellt wurde, verarbeitet wird und die dann wartet.
            1) Der Zieladapter führt die Funktionen aus, die bei seiner Erstellung mit MQSeries Adapter Builder definiert wurden. Seine Standardaufgabe besteht darin, die Daten aus der integrierten Nachricht in das Format der Zielanwendung zu konvertieren. Er ordnet Element für Element zu.
            2) Der Zieladapter sendet die Nachricht über eine anwendungsspezifische Schnittstelle an die Zielanwendung.
            3) Abhängig von der Funktionsweise der jeweiligen Zielanwendung sendet sie eine Antwort an den Zieladapter oder nicht.
        c. Die Methode *get message output*, von der die Antwort vom Zieladapter abgerufen wird. Die Antwort besteht möglicherweise nur aus der Information, dass die Zielanwendung die Nachricht empfangen hat; sie kann jedoch auch Daten enthalten.
    - Eine Session-Bean des EJB-Service ruft eine einzige Methode auf, die ein `TerminalDataContainer`-Objekt benötigt. Die von der Methode zurückgegebenen Daten werden als Antwortdaten betrachtet und müssen ein Objekt des Typs `TerminalDataContainer` sein.
13. Wenn der Zieladapterbefehl keine Ausnahmebedingung ausgibt oder nicht über eine Confirm_BOD-Antwort verfügt (was auf einen Fehler hinweisen kann), schreibt der Adaptermanager die aus der Empfangswarteschlange empfangene Nachricht mit Hilfe des internen Adapters fest (COMMIT).

14. Wenn eine Bestätigung angefordert wurde, ruft der Adaptermanager die Methode `sendResponse` auf dem internen Adapter auf.
    - Wenn der Zieladapter eine Antwort erstellt hat, stellt er die logische Antwort-ID der ursprünglichen Nachricht in das Feld für die logische Ziel-ID in der Anwortnachricht.
    - Wenn der Zieladapter keine Antwort erstellt hat, erstellt der Adaptermanager eine Confirm_BOD-Antwortnachricht mit dem Fertigstellungsstatus.
      – Sind keine Fehler aufgetreten, zeigt der Fertigstellungsstatus die erfolgreiche Ausführung an.
      – Im Fehlerfall zeigt der Fertigstellungsstatus eine Fehlerbedingung an.
15. Die Antwort wird gesendet.
    a. Der Adaptermanager sendet die Antwortnachricht, falls eine erstellt wurde, an den internen Adapter.
    b. Der interne Adapter reiht die Antwortnachricht in die Antwortwarteschlange ein.
    c. Der interne Adapter sendet die Antwortnachricht abhängig von der ursprünglichen Nachricht, die er empfangen hat:
       - Handelte es sich um eine MQSeries-Anforderungsnachricht, erhält der interne Adapter die Warteschlangeninformationen für die Antwort aus der MQSeries-Anforderungsnachricht. Diese Warteschlangeninformationen überschreiben die logische Ziel-ID in der Nachricht.
       - Handelte es sich nicht um eine MQSeries-Anforderungsnachricht, verwendet der interne Adapter die Methode `sendMsg` zum Senden der Antwort.
16. Im Falle einer Ausnahmebedingung oder einer Confirm_BOD-Antwortnachricht mit einem Fehlerstatus schreibt der Adaptermanager eine Ausnahmebedingungsnachricht in eine Ausnahmedatei mit dem Namen `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, die sich in demselben Verzeichnis wie der Adapterdämon befindet. Dabei steht *nnnnnnnn* für die Nummer der Protokolldatei. Wenn die WebSphere Business Integrator-Klassen installiert sind, senden diese Klassen darüber hinaus eine Ausnahme an die Komponente WebSphere Business Integrator Solution Management. Weitere Informationen finden Sie unter „Ausnahmebedingungsnachrichten” auf Seite 109.

17. Im Falle einer Ausnahmebedingung oder einer Confirm_BOD-Antwortnachricht mit einem Fehlerstatus weist der Adaptermanager den internen Adapter an, die ursprüngliche Nachricht in die Fehlerwarteschlange einzureihen. Der Name der Fehlerwarteschlange wird aus der Konfigurationsdatei gelesen, abhängig von der ID der abhängigen Anwendung, der Nachrichtenkategorie und dem Nachrichtenzweck der ursprünglichen Nachricht.

    Abhängig von der ID der abhängigen Anwendung, der Nachrichtenkategorie und dem Nachrichtenzweck führt der Adaptermanager eine aus mehreren Schritten bestehende Suche in der Konfigurationsdatei in der folgenden Reihenfolge aus:

    a. Er sucht spezifische Werte für die Nachrichtenkategorie und den Nachrichtenzweck.
    b. Er sucht einen spezifischen Wert für die Nachrichtenkategorie und einen Standardwert für den Nachrichtenzweck.
    c. Er sucht einen Standardwert für die Nachrichtenkategorie und einen spezifischen Wert für den Nachrichtenzweck.
    d. Er sucht Standardwerte für die Nachrichtenkategorie und den Nachrichtenzweck.

    - Wenn der interne Adapter die Fehlernachricht in die Fehlerwarteschlange einreihen kann, wird der interne Adapter angewiesen, für die Nachricht aus der Empfangswarteschlange eine COMMIT-Operation durchzuführen.
    - Wenn der interne Adapter die Fehlernachricht nicht in die Fehlerwarteschlange einreihen kann, geschieht Folgendes:
        a. Der Adaptermanager weist den internen Adapter an, den Vorgang zurückzusetzen (ROLLBACK-Operation), d. h. er wird nicht festgeschrieben.
        b. Der Adaptermanager setzt eine Markierung, mit der alle Adaptermanager, die unter demselben Adapterdämon aktiv sind, heruntergefahren werden. Dies bedeutet, dass es bei der Verarbeitung der Nachricht einen Fehler gegeben hat. Durch Herunterfahren aller Adaptermanager kann verhindert werden, dass andere Adaptermanager die fehlerhafte Nachricht mit demselben Ergebnis erneut verarbeiten.
        c. Wenn ein Speicherengpass die Ursache für den Fehler war, wird die Ausnahmebedingung wie alle anderen Ausnahmen behandelt, außer dass der Adaptermanager eine Markierung für sich selbst setzt, die dazu führt, dass er sich selbst stoppt, nachdem er die Verarbeitung der aktuellen Nachricht beendet hat. Dadurch wird für andere Adaptermanager mehr Speicher verfügbar.

18. Wenn der interne Adapter dem Adaptermanager mitteilt, dass die Verarbeitung beendet ist, überprüft dieser, ob die beiden folgenden Markierungen gesetzt sind:
    - Die Markierung, dass dieser Adaptermanager gestoppt werden muss. Die Ursache kann eine abnormale Java-Speicherbedingung sein.
    - Die Markierung, dass alle Adaptermanager gestoppt werden müssen. Die Ursache ist im vorhergehenden Schritt beschrieben.
19. Ist eine dieser Markierungen gesetzt, wird der Adaptermanager gestoppt. Ist keine Markierung gesetzt, verarbeitet der Adaptermanager die nächste Nachricht. Der Adaptermanager fordert den internen Adapter auf, eine neue Nachricht zu empfangen.
20. Wenn eine Antwortnachricht in die Antwortwarteschlange oder eine Fehlernachricht in die Fehlerwarteschlange eingereiht wurde, geschieht Folgendes:
    a. MQSeries oder eine andere Nachrichtenübermittlungssoftware schickt die Nachricht an die Quellenverarbeitung des Kernels zurück.
    b. Wenn der Quellenadapter die Methode `sendRequestResponse` seines internen Adapters aufgerufen hat, ruft der Kernel die Nachricht aus der Antwortwarteschlange ab und übergibt sie an den Quellenadapter. Hat der Quellenadapter die Methode `sendMsg` aufgerufen, reiht der Kernel die Nachricht in die Empfangswarteschlange der Quellenanwendung ein.

Transaktionsfunktionen des Kernels

Eine *Transaktion* besteht aus einer Gruppe von Operationen, die als untrennbare Arbeitseinheit ausgeführt werden müssen. Wenn alle Operationen einer Transaktion erfolgreich waren, wird die Transaktion *festgeschrieben* (COMMIT), d. h. alle Operationen werden ausgeführt. Wenn eine oder mehrere Operationen einer Transaktion fehlgeschlagen sind, wird die Transaktion *zurückgesetzt* (ROLLBACK), d. h. keine der Operationen wird ausgeführt. Mit Hilfe der Transaktionsverarbeitung von MQSeries Adapter Kernel kann ein Quellenadapter eine Reihe von Operationen als eine einzige Einheit ausführen, wobei sichergestellt ist, dass alle Operationen erfolgreich waren, wenn die Transaktion festgeschrieben wird, und keine Operationen ausgeführt wurden, wenn die Transaktion zurückgesetzt wird.

Die Transaktionsverarbeitung kann auf zwei Arten in Adapter eingefügt werden: mit Hilfe von MQSeries Adapter Builder oder durch Verwenden der Methoden `begin`, `rollback` und `commit` mit der Klasse `EpicNativeAdapter` der Java-API des Kernels. Wenn eine Transaktionsmethode in einem unzulässigen Kontext aufgerufen wird (z. B. Aufrufen der Methode `commit` ohne vorheriges Aufrufen der Methode `begin` oder Aufrufen der Methode `begin` innerhalb einer anderen Transaktion), löscht der Kernel den Aufruf und gibt eine Warnung an die Trace-Funktion aus. Informationen zur Verwendung der API finden Sie in „Kapitel 5. APIs von MQSeries Adapter Kernel verwenden" auf Seite 111.

Einschränkungen

Für die Transaktionsverarbeitung des Kernels gelten die folgenden Einschränkungen:

- Mit der Methode `sendRequestResponse` werden keine Transaktionen unterstützt.
- Verschachtelte Transaktionen (d. h. Transaktionen, die innerhalb von anderen Transaktionen aufgerufen werden) werden nicht unterstützt.
- Transaktionen werden nicht von jedem Übertragungsmodus unterstützt. Weitere Informationen finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115.

Trace-Funktion

Eine Trace-Nachricht enthält den Verarbeitungsstatus einer Nachricht an einem bestimmten Punkt im Kernel. Mit Hilfe von Trace-Nachrichten können Sie mögliche Ursachen für Probleme mit dem Kernel oder den Adaptern erkennen. Weitere Informationen zur Verwendung der Trace-Funktion des Kernels finden Sie im Handbuch *Problem Determination Guide* von MQSeries Adapter Kernel.

MQSeries Adapter Kernel mit WebSphere Business Integrator und WebSphere Application Server verwenden

Dieser Abschnitt beschreibt die Verwendung von MQSeries Adapter Kernel zusammen mit den Produkten WebSphere Business Integrator und WebSphere Application Server. Weitergehende Informationen finden Sie in der WebSphere Business Integrator-Dokumentation.

JMS Listener

WebSphere Business Integrator enthält eine Komponente mit dem Namen JMS Listener, die in Zusammenarbeit mit MQSeries Adapter Kernel und WebSphere Application Server Advanced Edition eine alternative Möglichkeit zur Übermittlung von Nachrichten an Zielanwendungen bietet. JMS Listener wird im Enterprise JavaBeans (EJB)-Server von WebSphere Application Server ausgeführt. Dieser Abschnitt gibt eine Übersicht über die Funktionalität von JMS Listener. Zusätzliche Informationen, einschließlich Details zur Konfiguration von WebSphere Business Integrator und JMS Listener finden Sie in der WebSphere Business Integrator-Dokumentation. Unter „Kernel konfigurieren" auf Seite 68 finden Sie Informationen zur Konfiguration von MQSeries Adapter Kernel, sodass er JMS Listener als Ziel erkennt. Die Verwendung von JMS Listener als ein Ziel entspricht dem Senden einer Nachricht an einen Adapterdämon.

Bevor JMS Listener verwendet werden kann, müssen Sie eine Adaptermanager-Nachrichten-Bean von MQSeries Adapter Kernel sowie entweder Session-Bean-Adapter des Java-Service oder EAB-Adapter für die Zielverarbeitungsschicht des Kernels einsetzen. Führen Sie diese Tasks mit Hilfe von MQSeries Adapter Builder aus. In einer WebSphere Business Integrator-Umgebung ist der Betrieb des Kernels unter WebSphere Application Server mit dem unter einem Standalone-Adapterdämon vergleichbar, außer dass nicht der Adaptermanager, sondern JMS Listener die Nachricht empfängt und den zugehörigen Adaptermanager aufruft. In einer MQSeries Adapter Kernel-Standalone-Umgebung startet der Adapterdämon die Adaptermanager, die dann Nachrichten direkt empfangen.

Bei einer Zusammenarbeit von MQSeries Adapter Kernel mit JMS Listener ist der Ablauf wie folgt:

1. JMS Listener überwacht eine JMS-Warteschlange und empfängt ein JMS-Nachrichtenobjekt, entweder von einem EJB-Client oder einer EJB-fremden Anwendung.
2. JMS Listener erstellt ein Exemplar der *Adaptermanager-Nachrichten-Bean* und übergibt ihr das Nachrichtenobjekt. Die Adaptermanager-Nachrichten-Bean ist ein Exemplar einer *Session-Bean*, d. h. eine Enterprise-Bean, die temporäre Daten, die einem bestimmten Client zugeordnet sind, einkapselt.

3. Die Adaptermanager-Nachrichten-Bean konvertiert das JMS-Nachrichtenobjekt in ein MQSeries Adapter Kernel-Nachrichtenträgerobjekt.
4. Ausgehend von den Header-Werten der Nachricht ruft der Kernel entweder einen EAB-Adapter oder einen EJB-Adapter auf. Wenn es sich bei dem aufzurufenden Adapter um einen EAB-Adapter handelt, entspricht der Datenfluss dem in einer Standalone-Umgebung. Wenn es sich bei dem aufzurufenden Adapter um einen EJB-Adapter handelt, wird eine EJB-Steuerroutine aufgerufen, die folgende Tasks ausführt:
   - Sie bestimmt, welche Service-Session-Bean (Home-Schnittstelle) und welche Methode aufgerufen werden, und sie bestimmt den Eingabeparametertyp der Methode für das `TerminalDataContainer`-Objekt.
   - Sie konvertiert die Anwendungsdaten im Nachrichtenträgerobjekt mit Hilfe einer `Mapper`-Klasse in die entsprechende `TerminalDataContainer`-Datenstruktur für die Service-Session-Bean.

     Das Objekt `TerminalDataContainer` enthält die Metadaten und die Anwendungsobjekte des Nachrichtenträgerobjekts. In vielen Fällen entspricht das Anwendungsobjekt der XML-Dokumentzeichenfolge der Nachrichtendaten des Nachrichtenträgerobjekts.
   - Sie ruft die Service-Session-Bean auf und übergibt das Objekt `TerminalDataContainer` an die zugehörige Methode der Service-Session-Bean. Die Service-Session-Bean, die Teil des Java-Serviceadapters ist, ist das Ziel der Nachricht.
5. Wenn eine Antwort angefordert wurde, konvertiert die Adaptermanager-Nachrichten-Bean das `TerminalDataContainer`-Antwortobjekt in ein Nachrichtenträgerobjekt und sendet die Antwort mit Hilfe des internen Adapters.
6. Bei einem Fehler reiht die Adaptermanager-Nachrichten-Bean das Nachrichtenträgerobjekt mit Hilfe des internen Adapters in eine Fehlerwarteschlange ein.

Unterstützung in der Landessprache

MQSeries Adapter Kernel stellt Unterstützung in der Landessprache bereit, wenn Java-Adapter verwendet werden. Für C-Adapter wird keine Unterstützung in der Landessprache zur Verfügung gestellt.

Kapitel 2. Installation des Kernels planen

In diesem Kapitel werden die Voraussetzungen für die Installation von MQSeries Adapter Kernel aufgelistet und die Komponenten des Kernels beschrieben.

Aktuelle Informationen erhalten Sie auf der Website der MQSeries-Produktfamilie unter:

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

IBM behält sich das Recht vor, die hier angegebenen Informationen nach Bedarf zu aktualisieren. Aktuelle Informationen über die unterstützten Softwareversionen finden Sie unter:

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

Hardware

MQSeries Adapter Kernel läuft auf der folgenden Hardware:

- Einem IBM PC (oder kompatiblen PC) mit Windows NT 4.0 und Service Pack 5 (oder höher) oder mit Windows 2000 und Service Pack 1.
- Einem System IBM RS/6000 mit AIX Version 4.3.2 oder 4.3.3.
- Einem System HP Series 9000 mit HP-UX Version 11.0.
- Einem System Sun SPARC oder UltraSPARC mit Solaris Version 8.
- Ein System IBM AS/400 oder iSeries mit OS/400 Version 4.4 oder 4.5.

Anmerkung: Die Installation von MQSeries Adapter Kernel unter OS/400 erfordert ein Windows-System, mit dem das System AS/400 verbunden ist. Weitere Informationen finden Sie unter „Voraussetzungen für eine Installation unter OS/400" auf Seite 40.

Für den Produktcode und die Produktdaten von MQSeries Adapter Kernel ist ein Plattenspeicherplatz von mindestens 25 MB erforderlich.

Stellen Sie sicher, dass genügend Plattenspeicherplatz für die Adapter verfügbar ist. Deren Größe ist von der Größe der Datenstrukturen, der Komplexität der Zuordnungen und dem verwendeten benutzerspezifischen Code abhängig. Im Folgenden sind einige Beispiele für verschiedene Adaptergrößen auf Windows-Systemen aufgeführt. Die Adapter in Ihrer Anwendungsumgebung erfordern gegebenenfalls etwas mehr oder weniger Plattenspeicherplatz. Die

in den Beispielen angegebene Größe umfasst den Adapter-Quellencode, kompilierten Adaptercode, API-Quellencode und kompilierten API-Code in MB bzw. KB.

- Quellenadapter für das Hinzufügen eines Verkaufsauftrags: 1,89 MB
- Zieladapter für das Synchronisieren eines Kundendatensatzes: 389 KB
- Zieladapter für das Synchronisieren eines Inventardatensatzes: 161 KB
- Zieladapter für das Synchronisieren eines Artikels: 249 KB
- Zieladapter für das Synchronisieren eines Verkaufsauftrags: 579 KB

Zusätzlich werden mindestens 20 MB Arbeitsspeicher für den Kernel und die Adapter benötigt. Der tatsächlich benötigte Arbeitsspeicher ist von einer Reihe von Faktoren abhängig, z. B. von der Anzahl und Größe der Warteschlangen und der Größe der Trace-Dateien.

Software

In diesem Abschnitt wird die unterstützte Software aufgelistet, die in Verbindung mit MQSeries Adapter Kernel eingesetzt werden kann. Es werden die unterstützten Versionen angegeben. Siehe „Anhang B. Überprüfte Konfigurationen” auf Seite 121. Beachten Sie, dass C-Compiler auf Entwicklungssystemen erforderlich sind, jedoch nicht auf Produktionssystemen. Die aufgelisteten C-Compiler wurden erfolgreich mit MQSeries Adapter Kernel getestet; andere C-Compiler arbeiten möglicherweise problemlos mit dem Kernel zusammen, werden aber offiziell nicht unterstützt.

Windows-Systeme:

- Microsoft Windows NT Version 4.0 mit Service Pack 5 (oder höher) oder Microsoft Windows 2000 mit Service Pack 1. Sie können feststellen, welche Version und welches Service Pack von Microsoft Windows Sie einsetzen, indem Sie Windows Explorer öffnen und auf **? > Info** klicken.
- Microsoft Visual C++ 6.0 Compiler
- MQSeries Version 5.2 mit Support Pack MA88.
- IBM Java Development Kit (JDK) Version 1.2.2 oder 1.3.

  Anmerkung: Windows NT und Windows 2000 sind zurzeit die einzigen Plattformen, auf denen JDK Version 1.3 von MQSeries Adapter Kernel unterstützt wird.

AIX:

- Betriebssystem AIX Version 4.3.2 oder 4.3.3
- IBM C Set++ for AIX Version 3.1.3
- MQSeries Version 5.2 mit Support Pack MA88.
- Java Development Kit Version 1.2.2. JDK 1.3 wird nicht unterstützt.

- X Window System (X11R5 oder höher). Dies wird für die Installation benötigt, jedoch nicht zur Laufzeit.

HP-UX:

- Betriebssystem HP-UX Version 11.0
- HP-UX C/ANSI C-Compiler. Weitere Informationen finden Sie in der Datei `readme.txt`.
- MQSeries Version 5.2 mit Support Pack MA88.
- Java Development Kit Version 1.2.2. JDK 1.3 wird nicht unterstützt.
- X Window System (X11R5 oder höher). Dies wird für die Installation benötigt, jedoch nicht zur Laufzeit.

Solaris:

- Betriebsumgebung Solaris Version 8
- Sun Workshop Compilers C/C++. Weitere Informationen finden Sie in der Datei `readme.txt`.
- MQSeries Version 5.2 mit Support Pack MA88.
- Java Development Kit Version 1.2.2. JDK 1.3 wird nicht unterstützt.
- X Window System (X11R5 oder höher). Dies wird für die Installation benötigt, jedoch nicht zur Laufzeit.

OS/400:

- Betriebssystem OS/400 Version 4.4 oder 4.5, einschließlich der folgenden Programme:
  - Java Toolkit und Java Developer Kit Version 1.2.2. JDK 1.3 wird nicht unterstützt. Java Toolkit und Java Developer Kit werden als Lizenzprogramm mit der Nummer 5769–JV1 geliefert. Weitere Informationen zu Versionen von Java Developer Kit, die für die Installation von MQSeries Adapter Kernel auf einem AS/400-System erforderlich sind, finden Sie unter „Voraussetzungen für eine Installation unter OS/400" auf Seite 40.
  - Die Option Host Servers, die als Lizenzprogramm mit der Nummer 5769–SS1, Option 12, geliefert wird.
  - Qshell Interpreter, der als Lizenzprogramm mit der Nummer 5769–SS1, Option 30, geliefert wird.
  - TCP/IP, das als Lizenzprogramm mit der Nummer 5769–TC1 geliefert wird.
  - Integrated Language Environment C for AS/400, das unter der Lizenzprogrammnummer 5769–CX2 lieferbar ist.
- MQSeries Version 5.2 mit Support Pack MA88.

Informationen zu weiteren Voraussetzungen für die Installation von MQSeries Adapter Kernel unter OS/400 finden Sie unter „Voraussetzungen für eine Installation unter OS/400" auf Seite 40.

Die folgenden Produkte werden mit MQSeries Adapter Kernel unterstützt:

- MQSeries Version 5.2 mit Support Pack MA88

  Anmerkung: Wenn MQSeries nicht verwendet wird, muss ein anderes Nachrichtenübermittlungsprodukt, z. B. eine Implementierung von Java Message Service (JMS), eingesetzt werden.
- MQSeries Integrator Version 1.1
- MQSeries Integrator Version 2

Eine Liste der geprüften Konfigurationen mit MQSeries Adapter Kernel, MQSeries und MQSeries Integrator finden Sie in „Anhang B. Überprüfte Konfigurationen" auf Seite 121.

Voraussetzungen für eine Installation unter OS/400

In diesem Abschnitt werden die Voraussetzungen für die Installation von MQSeries Adapter Kernel auf einem AS/400- oder iSeries-System beschrieben. Genaue Anweisungen für die Installation von MQSeries Adapter Kernel auf einem AS/400-System finden Sie unter Schritt 3 auf Seite 48. Da AS/400-Terminals keine Java-Grafik unterstützen, ist eine grafikfähige Workstation, z. B. ein Windows-System, erforderlich, um das Java-basierte Installationsprogramm mit seiner grafischen Benutzerschnittstelle auszuführen. Es gibt zwei Möglichkeiten, die Workstation mit dem System AS/400 zu verbinden:

- Über ein fernes AWT, wobei die gesamte Grafik auf dem AS/400-System verarbeitet und auf der Workstation angezeigt wird. Eine ausführliche Beschreibung dazu finden Sie unter „Fernes AWT verwenden".
- Als angeschlossener Client, wobei die Workstation die Grafik verarbeitet und anzeigt. Eine ausführliche Beschreibung dazu finden Sie unter „Einen angeschlossenen Client verwenden" auf Seite 41.

In diesem Abschnitt wird davon ausgegangen, dass Sie ein Windows-System als die grafikfähige Workstation verwenden.

Fernes AWT verwenden

Bei Verwendung eines fernen AWT erfolgt die Verarbeitung der Java-Grafik auf dem AS/400-System, und die Anzeige der Grafik auf einer Client-Workstation, die an das AS/400-System angeschlossen ist. Dieser Abschnitt beschreibt die Voraussetzungen für die Installation von MQSeries Adapter Kernel auf einem AS/400-System unter Verwendung eines fernen AWT.

Auf dem OS/400-System müssen die folgenden Programme installiert sein:

- Java Toolkit und Java Developer Kit Version 1.2.2. Java Toolkit und Java Developer Kit werden als Lizenzprogramm mit der Nummer 5769–JV1 geliefert. Die Funktion des fernen AWT unter OS/400 wird durch das Java Developer Kit bereitgestellt.

- TCP/IP, das als Lizenzprogramm mit der Nummer 5769–TC1 geliefert wird. Weitere Informationen zu TCP/IP finden Sie in den Dokumenten *AS/400 TCP/IP Fastpath Setup Information* und *AS/400 TCP/IP Configuration*, die in der AS/400-Bibliothek unter www.ibm.com/servers/eserver/iseries/library/ verfügbar sind.

Für die Workstation gelten folgende Voraussetzungen:

- Ein IBM PC (oder kompatibler PC) mit Windows 95, Windows 98, Windows NT oder Windows 2000
- Eine TCP/IP-Verbindung mit dem AS/400-System
- JDK 1.2.2 oder höher

Gehen Sie zum Installieren und Starten des fernen AWT folgendermaßen vor:

1. Stellen Sie sicher, dass JDK 1.2.2 oder höher auf der Workstation installiert ist.
2. Stellen Sie sicher, dass eine TCP/IP-Verbindung zwischen dem AS/400-System und der Workstation besteht.
3. Kopieren Sie die Datei `RAWTGui.jar` aus dem Verzeichnis `/QIBM/ProdData/Java400/jdk12` auf dem AS/400-System in ein Verzeichnis auf der Workstation.
4. Wechseln Sie auf der Workstation in das Verzeichnis, in das Sie die Datei `RAWTGui.jar` kopiert haben, und starten Sie das ferne AWT durch Eingabe des folgenden Befehls:

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

Anmerkung: Da die Verarbeitung von Java-Grafiken auf einem AS/400-System sehr ressourcenintensiv ist, kann die Installation von MQSeries Adapter Kernel mit Hilfe eines fernen AWT gegebenenfalls erheblich länger dauern als bei Verwendung eines angeschlossenen Clients.

Weitere Informationen zum fernen AWT finden Sie in der AS/400-Bibliothek unter www.ibm.com/servers/eserver/iseries/library/.

Einen angeschlossenen Client verwenden

Bei Verwendung eines angeschlossenen Clients für die Installation von MQSeries Adapter Kernel auf einem AS/400-System erfolgt die Verarbeitung der Java-Grafik auf der Client-Workstation, und nicht auf dem AS/400-System. Dieser Abschnitt beschreibt die Voraussetzungen für die Installation von MQSeries Adapter Kernel auf einem AS/400-System unter Verwendung eines angeschlossenen Clients.

Auf dem OS/400-System müssen die folgenden Programme installiert sein:

- Java Toolkit und Java Developer Kit Version 1.2.2. Java Toolkit und Java Developer Kit werden als Lizenzprogramm mit der Nummer 5769–JV1 geliefert.
- Die Option Host Servers, die als Lizenzprogramm mit der Nummer 5769–SS1, Option 12, geliefert wird.
- TCP/IP, das als Lizenzprogramm mit der Nummer 5769–TC1 geliefert wird.

Für die Workstation gelten folgende Voraussetzungen:

- Ein IBM PC (oder kompatibler PC) mit Windows NT 4.0 und Service Pack 5 oder mit Windows 2000 und Service Pack 1.
- Eine TCP/IP-Verbindung mit dem AS/400-System
- JDK 1.2.2 oder höher

Komponenten des Kernels

Nach der Installation befindet sich MQSeries Adapter Kernel in seinem Stammverzeichnis (root). Es enthält Unterverzeichnisse, die ihrerseits wieder Verzeichnisse enthalten können. Das Stammverzeichnis und seine Unterverzeichnisse werden im Folgenden aufgelistet, zusammen mit einer Übersicht der Dateien, die für die Installation und Konfiguration von besonderer Bedeutung sind.

root Der Standardname lautet `C:\Programme\MQAK` auf Windows-Systemen, `/usr/lpp/mqak` unter AIX, `/MQAK` unter HP-UX, `/opt/MQAK` unter Solaris und `/QIBM/ProdData/mqak` unter OS/400. Es enthält Folgendes:

- Alle anderen Verzeichnisse von MQSeries Adapter Kernel.
- Die Datei `aqmsetenv.bat` (Windows-Systeme) oder `aqmsetenv.sh` (UNIX), mit deren Hilfe die Umgebungsvariablen des jeweiligen Systems nach der Installation bei Bedarf geändert werden.
- Die Datei `readme.txt`.
- Die Datei `aqmuninstall.bat` (Windows-Systeme) oder `aqmuninstall.sh` (UNIX).

bin Dieses Verzeichnis enthält Folgendes:

- Klassenbibliotheken und gemeinsam benutzte Bibliotheken.
- Adapter, die integrierter Bestandteil des Kernels sind (nur für Prüfungszwecke).
- Die Datei `aqmversion.bat` (Windows-Systeme) oder `aqmversion.sh` (UNIX und OS/400), ein Skript, das ausgeführt wird, um die Versionsnummer des Kernels anzuzeigen.

- Die Datei `aqmcrtmsg.bat` (Windows-Systeme) oder `aqmcrtmsg.sh` (UNIX und OS/400), ein Skript, das ausgeführt wird, um eine XML-Datei zu erstellen, mit deren Hilfe die Konfigurationsdatei überprüft wird, bevor die Produktionsumgebung aktiviert wird.
- Die Datei `aqmsndmsg.bat` (Windows-Systeme) oder `aqmsndmsg.sh` (UNIX und OS/400), ein Skript, das ausgeführt wird, um die Konfigurationsdatei zu prüfen, bevor die Produktionsumgebung aktiviert wird.
- Die Datei `aqmstrad.bat` (Windows-Systeme) oder `aqmstrad.sh` (UNIX und OS/400), ein Skript, das zum Starten des Adapterdämons ausgeführt wird.
- Die Datei `aqmstrtd.bat` (Windows-Systeme) oder `aqmstrtd.sh` (UNIX und OS/400), ein Skript, das zum Starten des Trace-Servers ausgeführt wird.

documentation
Dieses Verzeichnis enthält die Produktdokumentation, einschließlich des Informationszentrums.

runtimefiles
Dieses Verzeichnis enthält die Laufzeitdateien des Kernels.

samples

Dieses Verzeichnis enthält Beispiele für Adapter und zugeordnete Konfigurations- und Zwischendateien. Sie können sie für eigene Tests und zum Kennenlernen des Produkts verwenden.

Anmerkung: Der Kernel zeigt sein Leistungsvermögen am besten, wenn er zusammen mit Adaptern eingesetzt wird, die mit MQSeries Adapter Builder erstellt wurden. Er sollte nicht nur für Aufrufe der Kernel-APIs durch Benutzerprogramme verwendet werden. Die Adapterbeispiele sollen lediglich dabei helfen, die Funktionsweise des Kernels zu verstehen und Diagnoseaufgaben auszuführen.

- Adapterbeispiele
- Die Setup-Datei des Kernels, `aqmsetup`, die Werte zur Unterstützung der Adapterbeispiele enthält. Weitere Informationen zu dieser Datei finden Sie unter „Die Setup-Datei” auf Seite 74.

- Die Konfigurationsdatei des Kernels, `aqmconfig.xml`, die Werte zur Unterstützung der Adapterbeispiele, einschließlich von Trace-Beispielwerten enthält. Weitere Informationen zu dieser Datei finden Sie unter „Die Konfigurationsdatei" auf Seite 75.

toolkit

Dieses Verzeichnis enthält ein Softwareentwicklungs-Toolkit (SDK, Software Development Toolkit) bestehend aus:

- Header-Dateien
- Bibliotheksdateien für die Kompilierung auf Windows-Systemen

uninstall

Dieses Verzeichnis enthält Dateien für die Deinstallation des Kernels.

verification

Dieses Verzeichnis enthält die folgenden Dateien für die Prüfung der Kernel-Installation:

- Die Datei `aqmverifyinstall.bat` (Windows-Systeme) oder `aqmverifyinstall.sh` (UNIX und OS/400), ein Skript, das zur Prüfung der Kernel-Installation auf einem Computer ausgeführt wird.
- Die Datei `aqmcreateq.bat` (Windows-Systeme) oder `aqmcreateq.sh` (UNIX und OS/400), ein Skript, das MQSeries-Warteschlangen für die Prüfung erstellt. Weitere Informationen finden Sie unter „MQSeries-Warteschlangen erstellen" auf Seite 110.
- Die Datei `aqmconfig.xml`. Weitere Informationen zu dieser Datei finden Sie unter „Die Konfigurationsdatei" auf Seite 75.
- Die Datei `aqmsetup`. Weitere Informationen zu dieser Datei finden Sie unter „Die Setup-Datei" auf Seite 74.
- Die Datei `aqminstalltest.xml`.

Kapitel 3. Kernel installieren

Dieser Abschnitt beschreibt die notwendigen Schritte zum Installieren und Überprüfen von MQSeries Adapter Kernel. Der Installationsprozess besteht aus folgenden generellen Schritten:

Schritt 1. Vorbereiten der Installation. Weitere Informationen finden Sie unter „Installation vorbereiten" auf Seite 46.

Schritt 2. Installieren des Kernels. Weitere Informationen finden Sie unter „Kernel installieren" auf Seite 47.

Schritt 3. Ausführen bestimmter Schritte nach Installationsabschluss. Weitere Informationen finden Sie unter „Schritte nach Installationsabschluss" auf Seite 51.

Schritt 4. Überprüfen der Installation. Weitere Informationen finden Sie unter „Installation überprüfen" auf Seite 54.

In diesem Kapitel werden darüber hinaus folgende Themen behandelt:

- Verwenden der automatischen Installation (Silent Installation) zum Installieren von MQSeries Adapter Kernel. Weitere Informationen finden Sie unter „Automatische Installation verwenden" auf Seite 60.
- Upgrade von MQSeries Adapter Kernel von einer früheren Version. Weitere Informationen finden Sie unter „Upgrade des Kernels durchführen" auf Seite 62.
- Entfernen einer Installation von MQSeries Adapter Kernel. Weitere Informationen finden Sie unter „Kernel entfernen" auf Seite 63.

Führen Sie nach der Installation des Kernels die folgenden Tasks aus, um ihn für den Einsatz vorzubereiten:

1. Konfigurieren Sie den Kernel. Weitere Informationen finden Sie unter „Kernel konfigurieren" auf Seite 68.
2. Konfigurieren Sie die Nachrichtenübermittlungssoftware und wahlweise installierbare Software. Weitere Informationen finden Sie unter „MQSeries und MQSeries Integrator konfigurieren" auf Seite 104.
3. Erstellen Sie Ihre Adapter mit Hilfe von MQSeries Adapter Builder. Testen Sie dann die Adapter, und geben Sie sie für den Einsatz frei.
4. Starten Sie den Kernel. Weitere Informationen finden Sie unter „Kernel starten" auf Seite 105.

Installation vorbereiten

Für die Installation von MQSeries Adapter Kernel benötigen Sie Administrator- oder Root-Berechtigung. Sie müssen dort, wo Sie MQSeries Adapter Kernel installieren möchten, und dort, wo Sie die beiden Kernel-Konfigurationsdateien angelegt haben, über die Berechtigung verfügen, Dateien zu erstellen und auf Dateien zuzugreifen. Das aktuelle Verzeichnis muss im Pfad für ausführbare Dateien angegeben sein. Stellen Sie sicher, dass alle Benutzer-IDs, unter denen der Kernel ausgeführt wird, über Lese-, Schreib- und Ausführungsberechtigungen verfügen.

Sie müssen berechtigt sein, MQSeries-Operationen wie das Erstellen von Warteschlangenmanagern oder das Erstellen von und Zugreifen auf Warteschlangen auszuführen. Diese Operationen werden auf unterschiedliche Art und Weise auf verschiedenen Plattformen ausgeführt. Weitere Informationen finden Sie im Handbuch *MQSeries Administration Guide* für Ihre Plattform.

Die Benutzer-ID zum Starten der Kernel-Prozesse muss der Gruppe 'mqm' angehören. Es wird zwischen zwei Arten von Kernel-Prozessen unterschieden:

- Der Adapterdämon, von dem es einen für jede vom Computer unterstützte Zielanwendung gibt.
- Der Trace-Server (optional)

Beachten Sie, dass der Quellenadapter im Prozess der Quellenanwendung ausgeführt wird. Jeder Dämon oder Server, der den Quellenadapter enthält, muss gestartet werden, damit der Quellenadapter ausgeführt werden kann.

Sie müssen den Kernel installieren und konfigurieren, damit er die von Ihnen erstellten Adapter ausführen kann. Der Kernel muss jedoch nicht installiert werden, um MQSeries Adapter Builder zu installieren oder damit Adapter zu erstellen.

Führen Sie vor Beginn der Installation folgende Schritte aus:

- Lesen Sie die Datei `readme.txt`, die sich auf der CD-ROM oder im lokalen Netz befindet. Diese Datei enthält möglicherweise wichtige Informationen, die erst nach Fertigstellung dieses Buchs verfügbar waren. Sie befindet sich im Stammverzeichnis der Installation.
- Besuchen Sie die MQSeries-Website unter www.ibm.com/software/ts/mqseries/. Dort finden Sie möglicherweise wichtige Informationen, die erst nach Veröffentlichung dieses Buchs verfügbar waren, und gegebenenfalls sogar eine neue Ausgabe dieses Buchs.
- Wenn Sie ein Upgrade von einer früheren Version von MQSeries Adapter Kernel durchführen, finden Sie Anweisungen dazu unter „Upgrade des Kernels durchführen" auf Seite 62.

- Stellen Sie sicher, dass die Hardware- und Softwarevoraussetzungen erfüllt sind. Informationen dazu finden Sie unter „Hardware" auf Seite 37 und unter „Software" auf Seite 38. MQSeries muss installiert und aktiv sein, bevor Sie die Installation von MQSeries Adapter Kernel überprüfen können. Stellen Sie sicher, dass die Java-Unterstützung von MQSeries installiert und konfiguriert ist.

Kernel installieren

Führen Sie die folgenden betriebssystemspezifischen Schritte aus, um MQSeries Adapter Kernel auf einem Windows-System (Windows NT oder Windows 2000), einer UNIX-Plattform (AIX, HP-UX oder Solaris) oder unter OS/400 zu installieren:

Auf Windows-Systemen:

Schritt 1. Starten Sie das Installationsprogramm wie folgt:

- Wenn Sie die Installation von einem lokalen Netz durchführen, wechseln Sie in das Verzeichnis mit den Installationsdateien von MQSeries Adapter Kernel und führen Sie die Datei `install.bat` aus.
- Wenn Sie von der CD-ROM installieren, legen Sie die CD-ROM von MQSeries Adapter Kernel in das CD-ROM-Laufwerk ein. Wenn automatisches Ausführen aktiviert ist, wird das Installationsprogramm automatisch gestartet, andernfalls müssen Sie die Datei `install.bat` im Stammverzeichnis der CD-ROM ausführen, um das Installationsprogramm zu starten.

Anmerkung: Auf Windows-Systemen müssen Sie die Datei `install.bat` nicht an eine andere Stelle kopieren, bevor Sie sie ausführen. Während des Installationsprozesses werden Sie aufgefordert, den Installationsort für MQSeries Adapter Kernel anzugeben.

Schritt 2. Folgen Sie der Bedienerführung des Installationsprogramms. Wenn Sie MQSeries Adapter Kernel in einem anderen als dem Standardverzeichnis installieren (auf Windows-Systemen ist dies `C:\Programme\MQAK`), müssen Sie den vollständig qualifizierten Pfad des Installationsverzeichnisses angeben, der relative Pfad ist nicht ausreichend.

Unter UNIX:

Schritt 1. Starten Sie das Installationsprogramm wie folgt:

- Wenn Sie die Installation von einem lokalen Netz durchführen, wechseln Sie in das Verzeichnis mit den Installationsdateien von MQSeries Adapter Kernel und führen Sie das Skript `install.sh` aus.
- Wenn Sie von der CD-ROM installieren, legen Sie die CD-ROM von MQSeries Adapter Kernel in das CD-ROM-Laufwerk ein und hängen Sie, falls erforderlich, das CD-ROM-Laufwerk an, so wie in Ihrer Betriebssystemdokumentation beschrieben. Führen Sie das Skript `install.sh` im Stammverzeichnis der CD-ROM aus.

Schritt 2. Folgen Sie der Bedienerführung des Installationsprogramms. Wenn Sie MQSeries Adapter Kernel in einem anderen als dem Standardverzeichnis installieren, müssen Sie den vollständig qualifizierten Pfad des Installationsverzeichnisses angeben, der relative Pfad ist nicht ausreichend. Die Standardinstallationsverzeichnisse unter UNIX lauten wie folgt:

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

Unter OS/400:

Schritt 1. Stellen Sie sicher, dass alle unter „Hardware" auf Seite 37, „OS/400-Softwarevoraussetzungen" auf Seite 39 und „Voraussetzungen für eine Installation unter OS/400" auf Seite 40 genannten Voraussetzungen erfüllt sind. Beachten Sie, dass für die Installation von MQSeries Adapter Kernel unter OS/400 ein InstallShield-Programm verwendet wird, für das eine an das AS/400-System angeschlossene Workstation erforderlich ist. Weitere Informationen finden Sie unter „Voraussetzungen für eine Installation unter OS/400" auf Seite 40.

Schritt 2. Erstellen Sie ein Benutzerprofil mit dem Namen MQAKSRV auf dem AS/400-System, indem Sie den Befehl **CRTUSRPRF** an einer CL-Eingabeaufforderung (Control Language = Steuersprache) eingeben.

Schritt 3. Abhängig davon, ob Sie zum Ausführen der Installation ein fernes AWT oder eine angeschlossene Client-Workstation verwenden, müssen Sie folgende Schritte ausführen:

- Gehen Sie folgendermaßen vor, wenn Sie die Installation mit Hilfe eines fernen AWT entfernen:
  a. Stellen Sie sicher, dass das ferne AWT installiert und aktiv ist. Weitere Informationen finden Sie unter „Fernes AWT verwenden" auf Seite 40.

b. Stellen Sie sicher, dass das AS/400-System auf die Datei `installAS400.jar` zugreifen kann. Die Datei muss sich entweder im Integrated File System (IFS) oder auf einer am AS/400-System angeschlossenen Einheit befinden. Wenn sich die Datei auf einer angeschlossenen Einheit befindet, erstellen Sie mit dem Befehl **CRTLINK** (Create Link) eine symbolische Verbindung zu der Datei.

c. Führen Sie den Befehl **CRTJVAPGM** (Create Java Program) an der Datei `installAS400.jar` aus, um den Installationsprozess zu beschleunigen.

d. Führen Sie den Befehl **RUNJVA** (Run Java) wie unten angegeben aus, wobei *n.n.n.n* für die TCP/IP-Adresse der Workstation steht, auf der das ferne AWT aktiv ist:

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

- Gehen Sie folgendermaßen vor, wenn Sie die Installation mit Hilfe einer angeschlossenen Client-Workstation entfernen:

  Schritt a. Stellen Sie sicher, dass die unter „Einen angeschlossenen Client verwenden" auf Seite 41 genannten Voraussetzungen erfüllt sind.

  Schritt b. Stellen Sie sicher, dass die Option Host Servers auf der AS/400-Maschine installiert und aktiv ist. Sie können Host Servers durch Eingeben des Befehls **STRHOSTSVR** (Start Host Servers) an einer CL-Eingabeaufforderung starten.

  Schritt c. Stellen Sie sicher, dass TCP/IP auf der AS/400-Maschine installiert und aktiv ist. Sie können TCP/IP durch Eingeben des Befehls **STRTCP** (Start TCP/IP) an einer CL-Eingabeaufforderung starten.

  Schritt d. Öffnen Sie auf der Workstation eine Eingabeaufforderung, und wechseln Sie in das Verzeichnis `AS400` der Installationsquelle für MQSeries Adapter Kernel (entweder ein lokales Netz oder die CD-ROM).

  Schritt e. Geben Sie folgenden Befehl ein:

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

Schritt 4. Das Installationsprogramm wird gestartet und ruft die Anzeige **Signon to AS/400** (Anmelden in AS/400) auf. Geben Sie die TCP/IP-Adresse der AS/400-Maschine im Feld **System:** und Ihre Benut zer-ID und Ihr Kennwort in den entsprechenden Feldern ein. Aktivieren Sie nicht das Kontrollkästchen **Default User** (Standardbenutzer). Klicken Sie auf **Next** (Weiter).

Schritt 5. Folgen Sie der Bedienerführung des Installationsprogramms. Je nach der Übertragungsgeschwindigkeit des Netzes und der Systeme, kann der Installationsprozess bis zu einer Stunde dauern. Ein Statusbalken auf der Workstation zeigt den aktuellen Status der Installation an.

Beachten Sie, dass MQSeries Adapter Kernel unter OS/400 immer im Verzeichnis `/QIBM/ProdData/mqak` im Stammverzeichnis des Integrated File System (IFS) installiert wird.

Schritt 6. Setzen Sie die Umgebungsvariablen CLASSPATH, PATH und QIBM_MULTI_THREADED wie folgt:

- Fügen Sie das Verzeichnis `/QIBM/ProdData/mqak/bin` der Umgebungsvariablen CLASSPATH hinzu.
- Fügen Sie das Verzeichnis `/QIBM/ProdData/mqak/bin` der Umgebungsvariablen PATH hinzu.
- Setzen Sie die Umgebungsvariable QIBM_MULTI_THREADED auf `Y`.

Schritt 7. Fügen Sie die Bibliothek MQAK der Bibliotheksliste QSYS.LIB hinzu.

Die Installation des Kernels ist beendet. Der Kernel ist jetzt so konfiguriert, dass er die Installationsüberprüfung unterstützt, ein Produktionsbetrieb ist in der installierte Umgebung jedoch noch nicht möglich. Überprüfen Sie die Installation, indem Sie die unter „Installation überprüfen” auf Seite 54 beschriebenen Schritte ausführen. Gehen Sie nach der Überprüfung der Installation wie unter „Schritte nach Installationsabschluss” auf Seite 51 beschrieben vor, um die Umgebungsvariablen zu setzen und mehrere Konfigurationsdateien zu verschieben, damit in der installierten Umgebung auch ein Produktionsbetrieb möglich ist.

Installieren Sie den Kernel gegebenenfalls auf weiteren Computern.

Schritte nach Installationsabschluss

Führen Sie nach der Installation des Kernels die folgenden Schritte aus:

Schritt 1. Entscheiden Sie, wo die Dateien `aqmsetup` und `aqmconfig.xml`, die für die Konfiguration des Kernels verwendet werden, gespeichert werden. Weitere Informationen zu diesen Dateien finden Sie unter „Kernel konfigurieren" auf Seite 68.

Achtung:
Wenn Sie keine eigenen Konfigurationsdateien erstellen, sondern die Standardkonfigurationsdateien im Verzeichnis `samples` für den Produktionsbetrieb verwenden, werden diese Dateien bei der Installation einer neuen Version des Kernels überschrieben und damit Ihre Produktionskonfiguration zerstört.

Schritt 2. Erstellen Sie ein Verzeichnis für die beiden Konfigurationsdateien. Sie müssen sich nicht in demselben Verzeichnis befinden, dies wird jedoch der Einfachheit halber empfohlen. Wenn Sie zu diesem Zweck ein Verzeichnis außerhalb des Installationsverzeichnisses von MQSeries Adapter Kernel erstellen, bleiben bei einer späteren Deinstallation des Kernels weniger Verzeichnisse zurück. Bei der Deinstallation bleiben Verzeichnisse, die etwas anderes als die Originaldateien von MQSeries Adapter Kernel enthalten, erhalten.

Schritt 3. Kopieren Sie die Dateien `aqmsetup` und `aqmconfig.xml` aus dem Verzeichnis `samples` in das von Ihnen erstellte Verzeichnis. Dabei kann es sich um ein Netzlaufwerk oder einen anderen zentralen Standort handeln, auf den viele Computer zugreifen können, um so das Aktualisieren und Sichern dieser Dateien zu erleichtern.

Wenn Sie die Datei `aqmconfig.xml` umbenennen, ist eine korrekte Ausführung des Kernels nicht mehr möglich. Sie können die Datei `aqmsetup` umbenennen, sofern Sie in Schritt 5 auf Seite 52 eine Umgebungsvariable mit einem entsprechenden Verweis auf die Datei erstellen.

Schritt 4. Editieren Sie die Datei `aqmsetup` in einem Texteditor, sodass sie auf das gewünschte Verzeichnis mit der Datei `aqmconfig.xml` zeigt. Verwenden Sie einen vollständig qualifizierten Pfad (keinen relativen Pfad) als Adresse des Verzeichnisses. Der Dateiname selbst darf nicht Teil des Pfades sein. Beispiel:

```
# Pfad der Konfigurationsdatei aqmconfig.xml.
AQMCONFIG=C:\Programme\MQAK\Data\
```

Auch wenn das gewünschte Verzeichnis für die Datei `aqmconfig.xml` dasselbe ist, in dem sich auch die Datei `aqmsetup` befindet, müssen Sie hier den vollständig qualifizierten Pfad eingeben. Speichern und schließen Sie die Datei `aqmsetup`.

Schritt 5. Setzen Sie die Umgebungsvariable AQMSETUPFILE, sodass sie auf das Verzeichnis der Datei `aqmsetup` zeigt (z. B. `C:\Programme\MQAK\Data\aqmsetup` auf Windows-Systemen, `/MQAK/data/aqmsetup` unter UNIX bzw. `/home/`*`Benutzername`*`/aqmsetup` unter OS/400). Beachten Sie, dass sich die Datei `aqmsetup` unter OS/400 immer im IFS-Stammverzeichnis des aktuellen Benutzers befinden muss (d. h. in `/home/`*`Benutzername`*`).`

Wenn der Kernel auf einem Netzlaufwerk installiert ist, führen Sie diesen Schritt für jeden Computer aus, der auf das Laufwerk zugreift.

Schritt 6. Wenn Sie unter AIX Quellenadapter verwenden wollen, die in der Programmiersprache C erstellt wurden und die von einem C-Programm aufgerufen werden, setzen Sie die Umgebungsvariable AIXTHREAD_SCOPE auf den Wert `S`. Geben Sie folgenden Befehl ein, um die Umgebungsvariable in der Bourne- oder Korn-Shell zu setzen:

```
export AIXTHREAD_SCOPE=S
```

Geben Sie folgenden Befehl ein, um die Umgebungsvariable in der C-Shell zu setzen:

```
setenv AIXTHREAD_SCOPE S
```

Damit die Variable AIXTHREAD_SCOPE automatisch gesetzt wird, wenn Sie sich bei AIX anmelden, müssen Sie diesen Befehl in Ihrer `.profile`-Datei (bei Verwendung der Bourne- oder Korn-Shell) bzw. in Ihrer `.cshrc`-Datei (bei Verwendung der C-Shell) hinzufügen.

Weitere Informationen zu Zeitplanungsrichtlinien finden Sie unter Schritt 7 auf Seite 27.

Schritt 7. Setzen Sie bei Bedarf die Umgebungsvariable THREADS_FLAG. Sie müssen diese Variable nur setzen, wenn *alle* im Folgenden genannten Bedingungen zutreffen:

- Sie verwenden das Betriebssystem Solaris.
- Sie verwenden Java Development Kit (JDK) Version 1.2.2.
- Sie verwenden MQSeries für den Nachrichtentransport.
- Ihre Quellen- und Zieladapter wurden in C erstellt.

Wenn alle diese Bedingungen zutreffen, setzen Sie die Umgebungsvariable THREADS_FLAG auf `native`. Geben Sie folgenden Befehl ein, um die Umgebungsvariable in der Bourne- oder Korn-Shell zu setzen:

```
export THREADS_FLAG=native
```

Geben Sie folgenden Befehl ein, um die Umgebungsvariable in der C-Shell zu setzen:

```
setenv THREADS_FLAG native
```

Damit die Variable THREADS_FLAG automatisch gesetzt wird, wenn Sie sich bei Solaris anmelden, müssen Sie diesen Befehl in Ihrer `.profile`-Datei (bei Verwendung der Bourne- oder Korn-Shell) bzw. in Ihrer `.cshrc`-Datei (bei Verwendung der C-Shell) hinzufügen.

Führen Sie nach Beendigung der Schritte nach Installationsabschluss die folgenden Tasks aus, um den Kernel für den Einsatz vorzubereiten:

1. Bereiten Sie den Produktionsbetrieb vor. Weitere Informationen finden Sie unter „Produktionsumgebung vorbereiten" auf Seite 67.
2. Editieren Sie die Konfigurationsdatei. Weitere Informationen finden Sie unter „Kernel konfigurieren" auf Seite 68.
3. Konfigurieren Sie MQSeries und wahlweise installierbare Software. Weitere Informationen finden Sie unter „MQSeries und MQSeries Integrator konfigurieren" auf Seite 104.
4. Beachten Sie in Bezug auf Produktionssysteme die Informationen unter „Empfehlungen zur Leistungsverbesserung" auf Seite 105.
5. Starten Sie den Kernel. Weitere Informationen finden Sie unter „Kernel starten" auf Seite 105.
6. Erstellen Sie einen Wartungsplan für den Kernel. Weitere Informationen finden Sie unter „Kernel warten" auf Seite 108.

Installation überprüfen

Prüfen Sie nach Abschluss der Installation durch Ausführen eines Prüfungs-Skripts, ob der Kernel korrekt installiert wurde. Das Skript sendet mit Hilfe eines Quellenadapters eine Testnachricht von einer Quellenanwendung, die dann vom Kernel an MQSeries weitergeleitet wird. Danach empfängt es über den Kernel die Nachricht von MQSeries und ruft einen Zieladapter auf. Alle diese Prozesse werden auf einem einzigen Computer ausgeführt.

Die Quellenanwendung bei dieser Prüfung ist eine MQSeries-Warteschlange mit dem Namen TEST1. Die Zielanwendung ist eine andere MQSeries-Warteschlange mit dem Namen TEST2.

Bei der Prüfung werden die folgenden Tasks ausgeführt:

- Es wird geprüft, ob der Kernel zusammen mit den zu diesem Zweck bereitgestellten Testadaptern (einem Quellen- und einem Zieladapter) die Testnachricht korrekt verarbeitet (Marshaling) und über MQSeries als Nachrichtenübermittlungssoftware weitergeleitet hat, wobei das Routing innerhalb des Testcomputers stattfindet.
- Die während der Installation bereitgestellten Dateien `aqmconfig.xml` und `aqmsetup` werden geprüft. In ihnen ist die Kernel-Konfiguration festgelegt. Weitere Informationen zu diesen Dateien finden Sie unter „Kernel konfigurieren” auf Seite 68.

Sie können die Konfigurationsdatei überprüfen, bevor Sie den Produktionsbetrieb aufnehmen. Weitere Informationen finden Sie unter „Konfigurationsdatei überprüfen” auf Seite 100.

Die von MQSeries Adapter Kernel bereitgestellten Skripts für die Installationsüberprüfung gehen davon aus, dass MQSeries auf der Maschine, auf der die Skripts ausgeführt werden, installiert und konfiguriert ist. Wenn Sie eine andere Nachrichtenübermittlungssoftware als MQSeries verwenden, können Sie die Skripts für die Installationsüberprüfung ändern, sodass sie Ihre Nachrichtenübermittlungssoftware unterstützen. Gehen Sie dazu folgendermaßen vor:

1. Wechseln Sie in das Installationsverzeichnis `verification` des Kernels.
2. Öffnen Sie die Datei `aqmconfig.xml` in einem Texteditor, und ändern Sie die Zeile `<epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>` in `<epicmqppqueuemgr>`*`WS_Manager_Name`*`</epicmqppqueuemgr>`, wobei *WS_Manager_Name* für den Namen Ihres Warteschlangenmanagers steht.

3. Editieren Sie die Datei `aqmverifyinstall` wie folgt:
   - Wenn Sie die Installationsüberprüfung auf einem Windows-System durchführen, öffnen Sie die Datei `aqmverifyinstall.bat` in einem Texteditor und ändern Sie die Zeile `aqmcreateq TEST2` in `aqmcreateq TEST2 WS_Manager_Name`, wobei *WS_Manager_Name* für den Namen Ihres Warteschlangenmanagers steht.
   - Wenn Sie die Installationsüberprüfung unter UNIX oder OS/400 durchführen, öffnen Sie die Datei `aqmverifyinstall.sh` in einem Texteditor und ändern Sie die Zeile `aqmcreateq.sh TEST2` in `aqmcreateq.sh TEST2 WS_Manager_Name`, wobei *WS_Manager_Name* für den Namen Ihres Warteschlangenmanagers steht.

Bei dieser Überprüfung werden Komponenten verwendet, z. B. ein Zieladapter (`com.ibm.epic.adapters.eak.test.InstallVerificationTest`), die nicht Teil des Kernels sind. Sie werden zwar zusammen mit dem Kernel zur Verfügung gestellt, dienen aber ausschließlich der Installationsüberprüfung.

Nach Abschluss der Überprüfung wird der Adapterdämon für die Überprüfung gestoppt.

Die Trace-Funktion ist während der Überprüfung nicht aktiviert.

Prüfverfahren

Schritt 1. Für die Überprüfung werden drei MQSeries-Warteschlangen erstellt. Wenn diese Warteschlangen bereits vor Beginn der Überprüfung Nachrichten enthalten, schlägt die Überprüfung fehl. Löschen Sie deshalb die Nachrichten in den folgenden Warteschlangen:
- TEST2AIQ
- TEST2AEQ
- TEST2RPL

Schritt 2. Stellen Sie sicher, dass Sie über die Berechtigung zum Installieren des Kernels und Überprüfen der Kernel-Installation verfügen. Weitere Informationen finden Sie unter „Installation vorbereiten” auf Seite 46.

Schritt 3. Starten Sie die Überprüfung wie folgt:
- Klicken Sie auf Windows-Systemen doppelt auf die Datei `aqmverifyinstall.bat` im Verzeichnis `verification`. Oder öffnen Sie eine Eingabeaufforderung, wechseln Sie in das Verzeichnis `verification`, und führen Sie die Datei `aqmverifyinstall.bat` aus.

- Öffnen Sie unter UNIX ein Terminal, wechseln Sie in das Verzeichnis `verification`, und führen Sie die Datei `aqmverifyinstall.sh` aus.
- Gehen Sie unter OS/400 folgendermaßen vor:
  a. Starten Sie eine **qsh**-Sitzung, indem Sie den Befehl **STRQSH** eingeben.
  b. Kopieren Sie die Datei `/QIBM/ProdData/mqak/verification/aqmsetup` in Ihr Stammverzeichnis (`/home/`*`Benutzername`*).
  c. Wechseln Sie in das Verzeichnis `/QIBM/ProdData/mqak/verification`.
  d. Führen Sie die Datei `aqmverifyinstall.sh` aus.

Die Datei `aqmverifyinstall` enthält Kommentare zum Ablauf des Prüfverfahrens.

Schritt 4. Die Nachricht `Installationsfunktionstest erfolgreich beendet` zeigt an, dass die Überprüfung erfolgreich war. Schließen Sie, falls nötig, das Prüfungsfenster.

Schritt 5. Werten Sie im Fehlerfall das Prüfungsfenster und die Protokolldatei `EpicSystemExceptionFile`*`nnnnnnnn`*`.log` aus, um den Fehler zu bestimmen.

Schritt 6. Eine Beschreibung typischer Probleme, die bei der Überprüfung auftreten können, und wie Sie darauf reagieren können, finden Sie unter „Typische Probleme bei der Installationsüberprüfung".

Schritt 7. Führen Sie bei Bedarf zusätzliche Überprüfungen durch. Weitere Informationen finden Sie unter „Zusätzliche Überprüfungen" auf Seite 59.

Schritt 8. Kehren Sie zum Installationsverfahren zurück, und konfigurieren Sie den Kernel für die Unterstützung des Betriebs in Ihrer Installationsumgebung. Fahren Sie mit Schritt 1 auf Seite 51 fort.

Typische Probleme bei der Installationsüberprüfung

Dieser Abschnitt enthält einige typische Probleme, die während der Installationsüberprüfung auftreten können, sowie mögliche Lösungen. Wichtige Informationen in den Ausnahmebedingungsnachrichten sind **fett** hervorgehoben.

Problem: Die Datei `aqmsetup` wurde nicht gefunden.

Antwort: Stellen Sie sicher, dass der Wert für die Umgebungsvariable AQMSETUPFILE auf die Position der Datei `aqmsetup` im Verzeichnis `verification` zeigt.

Ausnahmebedingungsnachricht:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterDirectory::getProperties():
Empfangene Ausnahme: <com.ibm.epic.adapters.eak.common.AdapterException>
Nachrichteninformationen: <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterCfg::readConfig(String):
Empfangene Ausnahme: <java.io.FileNotFoundException> Nachrichteninforma-
tionen: <C:\aqmsetup> Zusätzliche Programminformationen: <>.>
Zusätzliche Programminformationen: <Fehler beim Lesen der Konfigurations-
datei [Eventuell ist die Datei nicht vorhanden oder enthält keine
Schlüssel]>.>
```

Problem: Die Datei `aqmconfig.xml` wurde nicht gefunden.

Antwort: Editieren Sie die Datei `aqmsetup` im Verzeichnis `verification`, und stellen Sie sicher, dass der Eintrag `AQMCONFIG=` auf das Verzeichnis `verification` zeigt. Geben Sie einen vollständig qualifizierten Pfad an. Stellen Sie außerdem sicher, dass sich die Datei `aqmconfig.xml` im Verzeichnis `verification` befindet.

Ausnahmebedingungsnachricht:

```
com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.common.AdapterDirectory::
getProperties(): Empfangene Ausnahme:
<java.io.FileNotFoundException> Nachrichteninformationen:
<AQMCONFIG.xml> Zusätzliche Programminformationen <>.>
```

Problem: Die Warteschlange, in die die Nachricht eingereiht werden soll, existiert nicht.

Antwort: Verwenden Sie MQSeries, um sicherzustellen, dass die in der Ausnahmebedingungsnachricht angegebene Warteschlange (TEST2AIQ bei der Installationsüberprüfung) existiert und Nachrichten entgegennehmen kann. Weitere Informationen finden Sie unter „MQSeries-Warteschlangen erstellen” auf Seite 110.

Ausnahmebedingungsnachricht:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message
<AQM0107: com.ibm.epic.adapters.eak.nativeadapter.LMSMQbase::
createMQOutputQueue(String):
MQ-Ausnahme (MQException) empfangen beim Erstellen einer Warteschlange,
 WS-Manager <DEFAULT>
Warteschlange <TEST2AIQ>:
Beendigungscode <2> Ursachencode <2085>.>
```

Problem: Der Zieladapter wurde nicht gefunden.

Antwort: Stellen Sie sicher, dass der in der Nachricht angegebene Zieladapter (com.ibm.epic.adapters.eak.test.InstallVerificationTest) existiert. Stellen Sie sicher, dass die Umgebungsvariable CLASSPATH das Verzeichnis bin des Kernels enthält.

Ausnahmebedingungsnachricht:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2:
Message <<TEST2> <2000.05.18.09.41.43.781> <<Nachrichten werden
verarbeitet.> <com.ibm.epic.adapters.eak.common.AdapterException:
MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker:
:instantiateClass(String, Class[], Object[]): Empfangene Ausnahme:
<java.lang.ClassNotFoundException> Nachrichteninformationen:
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>
Zusätzliche Programminformationen <[Für den Klassennamen
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>
kann keine Klasse abgerufen werden]>.>>>>
```

Problem: Ein für die Übermittlung der Nachricht erforderlicher Adapter konnte nicht geladen werden. Für die logische Ziel-ID gibt es in der Datei aqmconfig.xml keinen Eintrag für den Nachrichtenzweck und die Nachrichtenkategorie, die in der Nachricht in der Warteschlange angegeben sind.

Antwort: Die wahrscheinlichste Ursache für diese Ausnahmebedingungsnachricht während der Installationsüberprüfung ist, dass die Warte- schlange TEST2AIQ bereits vor Beginn der Prüfung Nachrichten ent- hielt. Löschen Sie alle Nachrichten aus der Warteschlange TEST2AIQ, und wiederholen Sie die Überprüfung. Der einzige Eintrag für einen Befehlsklassennamen für die Anwendung TEST2 in der Datei aqmconfig.xml im Verzeichnis verification ist TESTBOD für den Nachrichtenzweck und OAG für die Nachrichtenkategorie.

Ausnahmebedingungsnachricht:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2: Message <<TEST2> <2000.05.18.10.28.43.105>
<<Nachrichten werden verarbeitet.> <com.ibm.epic.adapters.eak.common.
AdapterException:
MessageID <AQM0401> <AQM0401: com.ibm.epic.adapters.eak.
adapterdaemon.EpicAdapterWorker::processMessage(EpicMessage):
Zu ladender Befehlsklassenname für eine empfangene Nachricht
kann nicht abgerufen werden.>>>>
```

Problem: Der Warteschlangenmanager für die Überprüfung wurde nicht gestartet.

Antwort: Stellen Sie sicher, dass der standardmäßige MQSeries-Warteschlangenmanager erfolgreich gestartet wurde.

Ausnahmebedingungsnachricht:

```
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):
MQ-Ausnahme (MQException) empfangen beim Erstellen der
WS-Managerverbindung für WS-Manager <QMGRNAME>
MQ-Nachrichteninformationen: Beendigungscode <2> Ursachencode <2059>.>
```

Problem: Ein allgemeiner MQSeries-Fehler ist aufgetreten.

Antwort: Stellen Sie sicher, dass MQSeries korrekt installiert und konfiguriert ist und auf der Maschine ausgeführt wird. Werten Sie den Ursachencode MQException aus, und bestimmen Sie mit Hilfe des Dokuments *MQSeries Messages* den Auslöser für den Ursachencode.

Ausnahmebedingungsnachricht:

```
MQ-Ausnahme (MQException) empfangen: "ACTION ATTEMPTED." Nachrichten-
informationen:
Beendigungscode <completion_code> Ursachencode <reason_code>
```

Zusätzliche Überprüfungen

Nachdem Sie überprüft haben, ob der Kernel auf dem ersten Computer korrekt installiert ist, können Sie optional folgende Schritte ausführen:

1. Überprüfen Sie mit demselben Prüfverfahren, ob der Kernel auch auf einem zweiten Computer korrekt installiert wurde.
2. Überprüfen Sie, ob Sie eine Testnachricht von einem Quellenadapter auf einem Computer an einen Zieladapter auf einem anderen Computer senden können. Diese Überprüfung sollten Sie manuell konfigurieren und ausführen. Wenn Sie für diese Überprüfung die ursprünglichen Prüfdateien, die mit dem Kernel bereitgestellt wurden, ändern, erstellen Sie zu Sicherungszwecken eine Kopie der ursprünglichen Prüfdateien.

Automatische Installation verwenden

MQSeries Adapter Kernel kann mit Hilfe der *automatischen Installation* (Silent Installation) auf allen Plattformen installiert werden. Durch Verwendung der automatischen Installation können Sie das MQSeries Adapter Kernel-Installationsprogramm umgehen, in dem Sie die gewünschten Installationsoptionen manuell auswählen müssen. Die automatische Installation ist dann hilfreich, wenn Sie die Standardkonfiguration auf mehreren Maschinen installieren möchten.

Führen Sie für eine automatische Installation des Kernels die folgenden betriebssystemspezifischen Schritte aus:

Auf Windows-Systemen:

Schritt 1. Öffnen Sie eine Eingabeaufforderung, und wechseln Sie in das Verzeichnis mit den Installationsdateien von MQSeries Adapter Kernel.

Schritt 2. Geben Sie folgenden Befehl ein:

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

Dabei steht *Installationsverzeichnis* für das gewünschte Verzeichnis für die Installation (z. B. `D:\mqak`).

Unter UNIX:

Schritt 1. Wechseln Sie an einem Terminal in das Verzeichnis mit den Installationsdateien von MQSeries Adapter Kernel. Wenn Sie von der CD-ROM installieren, legen Sie die CD-ROM von MQSeries Adapter Kernel in das CD-ROM-Laufwerk ein und hängen Sie, falls erforderlich, das CD-ROM-Laufwerk an, so wie in Ihrer Betriebssystemdokumentation beschrieben.

Schritt 2. Geben Sie folgenden Befehl ein:

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

Dabei steht *Installationsverzeichnis* für das gewünschte Verzeichnis für die Installation (z. B. `/opt/mqak`).

Unter OS/400:

Gehen Sie folgendermaßen vor, wenn Sie keinen angeschlossenen Client verwenden, um auf die AS/400-Maschine zuzugreifen:

Schritt 1. Stellen Sie sicher, dass das AS/400-System auf die Datei `installAS400.jar` zugreifen kann. Die Datei muss sich entweder im Integrated File System (IFS) oder auf einer am AS/400-System angeschlossenen Einheit befinden. Wenn sich die Datei auf einer ange-

schlossenen Einheit befindet, erstellen Sie mit dem Befehl **CRTLINK** (Create Link) eine symbolische Verbindung zu der Datei.

Schritt 2. Führen Sie den Befehl **CRTJVAPGM** (Create Java Program) an der Datei `installAS400.jar` aus, um den Installationsprozess zu beschleunigen.

Schritt 3. Geben Sie abhängig davon, ob Sie eine CL-Eingabeaufforderung oder eine **qsh**-Sitzung verwenden, einen der folgenden Befehle ein:

- Wenn Sie eine CL-Eingabeaufforderung verwenden, geben Sie folgenden Befehl ein:

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

- Wenn Sie eine **qsh**-Sitzung verwenden, geben Sie folgenden Befehl ein:

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

Gehen Sie folgendermaßen vor, wenn Sie einen angeschlossenen Client für die Kommunikation mit der AS/400-Maschine verwenden:

Schritt 1. Stellen Sie sicher, dass die unter „Einen angeschlossenen Client verwenden” auf Seite 41 genannten Voraussetzungen erfüllt sind.

Schritt 2. Stellen Sie sicher, dass die Option Host Servers auf der AS/400-Maschine installiert und aktiv ist. Sie können Host Servers durch Eingeben des Befehls **STRHOSTSVR** (Start Host Servers) an einer CL-Eingabeaufforderung (Control Language = Steuersprache) starten.

Schritt 3. Stellen Sie sicher, dass TCP/IP auf der AS/400-Maschine installiert und aktiv ist. Sie können TCP/IP durch Eingeben des Befehls **STRTCP** (Start TCP/IP) an einer CL-Eingabeaufforderung starten.

Schritt 4. Öffnen Sie auf der Workstation eine Eingabeaufforderung, und wechseln Sie in das Verzeichnis `AS400` der Installationsquelle für MQSeries Adapter Kernel (entweder ein lokales Netz oder die CD-ROM).

Schritt 5. Geben Sie folgenden Befehl ein:

```
java -cp installAS400.jar run -silent -os400 Maschinenname Benutzer-ID Kennwort
```

Dabei steht *Maschinenname* für die TCP/IP-Adresse des AS/400-Systems, *Benutzer-ID* für Ihre Benutzer-ID und *Kennwort* für Ihr Kennwort.

Upgrade des Kernels durchführen

Wenn Sie MQSeries Adapter Kernel Version 1.0 mit oder ohne CSD (Corrective Service Diskette) oder MQSeries Adapter Kernel Version 1.1 mit einer früheren Änderungsstufe installiert haben, führen Sie die folgenden Schritte aus, bevor Sie MQSeries Adapter Kernel Version 1.1 mit der aktuellen Änderungsstufe installieren:

Schritt 1. Sichern Sie die Dateien `aqmsetup` und `aqmconfig` (`aqmconfig.properties` oder `aqmconfig.xml`) in einem Verzeichnis außerhalb des Installationsverzeichnisses von MQSeries Adapter Kernel.

Schritt 2. Wenn eine CSD von MQSeries Adapter Kernel installiert ist, deinstallieren Sie sie wie folgt:

- Verwenden Sie unter Windows NT eine der folgenden Methoden:
  - Klicken Sie im Startmenü von Windows NT auf **Programme > MQSeries Adapter Kernel > Remove CSD**.
  - Verwenden Sie in der Systemsteuerung den Ordner 'Software', um das Programm zu entfernen.
  - Führen Sie die Datei `aqmuninstallCSD.bat` im Stammverzeichnis des Kernels aus.
  - Öffnen Sie eine Eingabeaufforderung, wechseln Sie in das Stammverzeichnis des Kernels, und geben Sie folgenden Befehl ein:

    ```
    java uninstallCSD
    ```
- Wechseln Sie unter AIX in das Stammverzeichnis des Kernels, und geben Sie einen der folgenden Befehle ein:

  ```
  aqmuninstallCSD.sh
  ```

  ```
  java uninstallCSD
  ```

Schritt 3. Deinstallieren Sie MQSeries Adapter Kernel wie folgt:

- Verwenden Sie unter Windows NT eine der folgenden Methoden:
  - Klicken Sie im Startmenü von Windows NT auf **Programme > MQSeries Adapter Kernel > Uninstall MQSeries Adapter Kernel**.
  - Verwenden Sie in der Systemsteuerung den Ordner 'Software', um das Programm zu entfernen.
  - Führen Sie die Datei `aqmuninstall.bat` im Stammverzeichnis des Kernels aus.
  - Öffnen Sie eine Eingabeaufforderung, wechseln Sie in das Stammverzeichnis des Kernels, und geben Sie folgenden Befehl ein:

    ```
    java uninstall
    ```

- Wechseln Sie unter AIX in das Stammverzeichnis des Kernels, und geben Sie einen der folgenden Befehle ein:

  ```
  aqmuninstall.sh
  ```

  ```
  java uninstall
  ```

Schritt 4. Installieren Sie MQSeries Adapter Kernel Version 1.1. Weitere Informationen finden Sie unter „Kernel installieren" auf Seite 47.

Schritt 5. Schreiben Sie die Dateien `aqmsetup` und `aqmconfig` in ihre ursprünglichen Verzeichnisse im Installationsverzeichnis von MQSeries Adapter Kernel zurück. Konvertieren Sie gegebenenfalls die Datei `aqmconfig.properties` in eine Datei mit dem Namen `aqmconfig.xml`. Weitere Informationen zur Datei `aqmconfig.xml` finden Sie unter „Die Konfigurationsdatei" auf Seite 75.

Kernel entfernen

Sie haben verschiedene Möglichkeiten, den Kernel zu entfernen. Beachten Sie, dass der Deinstallationsprozess keine Dateien und Verzeichnisse, die nach der Installation des Kernels erstellt wurden, entfernt. Dazu gehören zum Beispiel alle Protokolldateien und alle vom Benutzer kopierten Datendateien.

- Verwenden Sie auf Windows-Systemen eine der folgenden Methoden:
  - Klicken Sie im Startmenü auf **Programme > IBM MQSeries Adapter Kernel > Uninstall MQSeries Adapter Kernel**.
  - Verwenden Sie in der Systemsteuerung den Ordner 'Software', um das Programm zu entfernen.
  - Führen Sie die Datei `aqmuninstall.bat` im Stammverzeichnis des Kernels aus.
  - Um eine automatische Deinstallation des Kernels durchzuführen (d. h., ohne dass das Deinstallationsprogramm zu weiteren Eingaben oder zu Bestätigungen auffordert), öffnen Sie eine Eingabeaufforderung, wechseln Sie in das Installationsverzeichnis des Kernels, und geben Sie folgenden Befehl ein:

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

- Wechseln Sie unter UNIX in das Stammverzeichnis des Kernels, und geben Sie folgenden Befehl ein:

  ```
  aqmuninstall.sh
  ```

  Um eine automatische Deinstallation des Kernels durchzuführen (d. h., ohne dass das Deinstallationsprogramm zu weiteren Eingaben oder zu Bestätigungen auffordert), wechseln Sie in das Stammverzeichnis des Kernels, und geben Sie folgenden Befehl ein:

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

- Verwenden Sie unter OS/400 eine der folgenden Methoden zum Deinstallieren des Kernels:
  - Gehen Sie folgendermaßen vor, wenn Sie den Kernel mit Hilfe eines fernen AWT deinstallieren:

    Schritt 1. Stellen Sie sicher, dass das ferne AWT installiert und aktiv ist. Weitere Informationen finden Sie unter „Fernes AWT verwenden" auf Seite 40.

    Schritt 2. Führen Sie den Befehl **CRTJVAPGM** (Create Java Program) an der Datei `/QIBM/ProdData/mqak/uninstall/uninstall.jar` aus, um den Deinstallationsprozess zu beschleunigen.

    Schritt 3. Führen Sie den Befehl **RUNJVA** (Run Java) wie unten angegeben aus, wobei *n.n.n.n* für die TCP/IP-Adresse der Workstation steht, auf der das ferne AWT aktiv ist:

    ```
    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))
    ```

  - Gehen Sie folgendermaßen vor, wenn Sie den Kernel mit Hilfe einer angeschlossenen Client-Workstation deinstallieren:

    Schritt 1. Stellen Sie sicher, dass die unter „Einen angeschlossenen Client verwenden" auf Seite 41 genannten Voraussetzungen erfüllt sind.

    Schritt 2. Stellen Sie sicher, dass die Option Host Servers auf der AS/400-Maschine installiert und aktiv ist. Sie können Host Servers durch Eingeben des Befehls **STRHOSTSVR** (Start Host Servers) an einer CL-Eingabeaufforderung (Control Language = Steuersprache) starten.

    Schritt 3. Stellen Sie sicher, dass TCP/IP auf der AS/400-Maschine installiert und aktiv ist. Sie können TCP/IP durch Eingeben des Befehls **STRTCP** (Start TCP/IP) an einer CL-Eingabeaufforderung starten.

Schritt 4. Kopieren Sie die Dateien `uninstall.jar` und `uninstall.dat` aus dem Verzeichnis `/QIBM/ProdData/mqak/uninstall` auf dem AS/400-System in ein Verzeichnis auf der Client-Workstation.

Schritt 5. Geben Sie folgenden Befehl ein:

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

Um eine automatische Deinstallation des Kernels durchzuführen (d. h., ohne dass das Deinstallationsprogramm zu weiteren Eingaben oder zu Bestätigungen auffordert), geben Sie folgenden Befehl ein:

```
java -cp uninstall.jar run -silent -os400 Maschinenname
Benutzer-ID
Kennwort
```

Dabei steht *Maschinenname* für die TCP/IP-Adresse des AS/400-Systems, *Benutzer-ID* für Ihre Benutzer-ID und *Kennwort* für Ihr Kennwort.

– Wenn Sie direkt auf dem AS/400-System an einer CL-Eingabeaufforderung oder in einer **qsh**-Sitzung arbeiten und eine automatische Deinstallation des Kernels durchführen möchten (d. h., ohne dass das Deinstallationsprogramm zu weiteren Eingaben oder zu Bestätigungen auffordert), geben Sie folgenden Befehl ein:

  - An der CL-Eingabeaufforderung:

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

  - In einer **qsh**-Sitzung:

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

Kapitel 4. Kernel verwenden

Dieses Kapitel enthält die folgenden Informationen zur Verwendung des Kernels:

Produktionsumgebung vorbereiten

Führen Sie die folgenden Tasks aus, bevor Sie den Kernel in Ihrer Produktionsumgebung einsetzen:

1. Erstellen Sie ein Modell der gesamten Systemarchitektur, einschließlich MQSeries Adapter Offering, MQSeries bzw. einer anderen Nachrichtenübermittlungssoftware und optional MQSeries Integrator, auf der Basis der Anforderungen und Bedingungen in Ihrer Anwendungsumgebung. In der Regel erfordert jede Anwendungsumgebung ihre eigene Architektur.
2. Erstellen Sie die erforderlichen Quellen- und Zieladapter mit Hilfe von MQSeries Adapter Builder. Testen Sie dann die Adapter, und geben Sie sie für den Einsatz frei.
3. Entwickeln Sie außerhalb von MQSeries Adapter Offering anwendungsspezifische Schnittstellen für folgende Aufgaben:
   - Dem Quellenadapter ermöglichen, die Anwendungsdaten von der Quellenanwendung abzurufen.
   - Der Zielanwendung ermöglichen, die Nachrichtendaten vom Zieladapter abzurufen.

   Die genaue Funktionsweise der anwendungsspezifischen Schnittstelle ist von den Merkmalen der Quellen- bzw. der Zielanwendung abhängig. Anwendungsspezifische Schnittstellen können beispielsweise Folgendes enthalten:
   - API-Aufrufe und Benutzer-Exits
   - Befehle zum Lesen und Schreiben von Dateien
   - Auslöser für Datenbankoperationen
   - Nachrichtenwarteschlangen

4. Konfigurieren Sie den Kernel für die Unterstützung der Laufzeitverarbeitung: Senden, Routing, Durchführen von Traces sowie Übermitteln von Nachrichten. Weitere Informationen zur Konfiguration des Kernels finden Sie unter „Kernel konfigurieren".
5. Konfigurieren Sie MQSeries bzw. ein anderes Nachrichtenübermittlungsprodukt und gegebenenfalls MQSeries Integrator für die Unterstützung Ihrer Systemarchitektur. Weitere Informationen finden Sie unter „MQSeries und MQSeries Integrator konfigurieren" auf Seite 104.
6. Entwickeln Sie bei Bedarf Java-Anmeldeklassen für die Unterstützung der Nachrichtenübermittlung. Sie sind für jede Zielanwendung spezifisch. Sie sind nur erforderlich, wenn der Zieladapter Informationen zum Anmelden bei der Zielanwendung und zum Herstellen einer Verbindung mit der Zielanwendung benötigt.
7. Testen Sie das gesamte System, d. h. MQSeries Adapter Kernel mit Ihren Quellen- und Zieladaptern, Ihren anwendungsspezifischen Schnittstellen und Ihrem benutzerspezifischen Code, bevor Sie das System in Ihrer Produktionsumgebung einsetzen.
8. Setzen Sie das System in der Produktionsumgebung ein.
9. Aktivieren Sie den Kernel, indem Sie einen oder mehrere Adapterdämonen und optional Trace-Server starten. Stellen Sie sicher, dass die Quellenanwendung gestartet wurde. Wenn der Quellenadapter im Prozess der Quellenanwendung ausgeführt wird, wird er automatisch mit der Quellenanwendung gestartet, sodass keine weiteren Schritte zum Starten des Quellenadapters unternommen werden müssen. Alle Dämonen oder Server, die den Quellenadapter enthalten, müssen gestartet werden. Weitere Informationen finden Sie unter „Kernel starten" auf Seite 105.

Kernel konfigurieren

Dieser Abschnitt beschreibt die Konfiguration des Kernels für die Verwendung in Ihrer Umgebung. Der Abschnitt „Übersicht über die Konfiguration" auf Seite 69 gibt eine konzeptionelle Übersicht über die Kernel-Konfiguration. Unter „Start- und Konfigurationsdateien" auf Seite 74 werden die verschiedenen Dateien beschrieben, die zusammen eine MQSeries Adapter Kernel-Konfiguration definieren. „Die Setup-Datei" auf Seite 74 beschäftigt sich mit der Datei `aqmsetup`, in der eine Reihe von Anfangswerten für den Kernel definiert werden. Unter „Die Konfigurationsdatei" auf Seite 75 finden Sie eine Beschreibung der Datei `aqmconfig.xml`, die spezifische Konfigurationsdaten für den Kernel bereitstellt, z. B. die Namen von Quellen- und Zielanwendungen, Quellen- und Zieladaptern, Warteschlangen und Warteschlangenmanagern, Übertragungsmodi sowie Protokoll- und Trace-Spezifikationen.

Übersicht über die Konfiguration

Dieser Abschnitt gibt eine konzeptionelle Übersicht über die Kernel-Konfiguration. Es ist wichtig, den Verarbeitungsablauf im Kernel während der Laufzeit zu verstehen, bevor Sie den Kernel konfigurieren. Dieser Abschnitt enthält eine vereinfachte Beschreibung der Laufzeitverarbeitung. Ausführliche Informationen hierzu finden Sie unter „Laufzeitverarbeitung” auf Seite 17.

Grundsätzlich gilt, dass die Konfiguration von MQSeries Adapter Kernel durch die Daten bestimmt wird, die zwischen Anwendungen ausgetauscht werden. Bei der Konfiguration müssen außerdem folgende Faktoren berücksichtigt werden:

- Die Anwendungen, die die Daten empfangen.
- Die Adapter, die in der Quellenverarbeitungsschicht erforderlich sind, sowie die Zieladapter, Adapterdämonen und Adaptermanager, die in der Zielverarbeitungsschicht erforderlich sind.
- Die Übertragungsmodi, Marshaling-Formate und Transportmechanismen, die verwendet werden.

Jede Anwendung in der Konfiguration verwendet andere Datenstrukturen und -formate. Wenn eine Konfiguration zum Beispiel zwei Anwendungen, A und B, umfasst, die beide Auftragsdaten an Anwendung C senden, haben die Daten von Anwendung A mit großer Wahrscheinlichkeit ein anderes Format als die von Anwendung B, und die Anweisungen (Tags) in den Formaten haben unterschiedliche Bedeutungen. Um zu verhindern, dass Anwendung C die unterschiedlichen Datenflüsse von den beiden Anwendungen erkennen und syntaktisch analysieren muss, werden die Daten der beiden Anwendungen in eine *integrierte Nachricht*, d. h. in ein *neutrales Integrationsformat* konvertiert. Bei dem neutralen Integrationsformat handelt es sich in der Regel um einen auf XML basierenden Industriestandard. Das einzige Datenformat, das Anwendung C erkennen und syntaktisch analysieren muss, ist das neutrale Integrationsformat.

Abb. 3 auf Seite 70 zeigt den Datenfluss von den Anwendungen A und B zu den Anwendungen C und D. Im Anschluss werden die verschiedenen in der Abbildung dargestellten Datenflüsse erläutert.

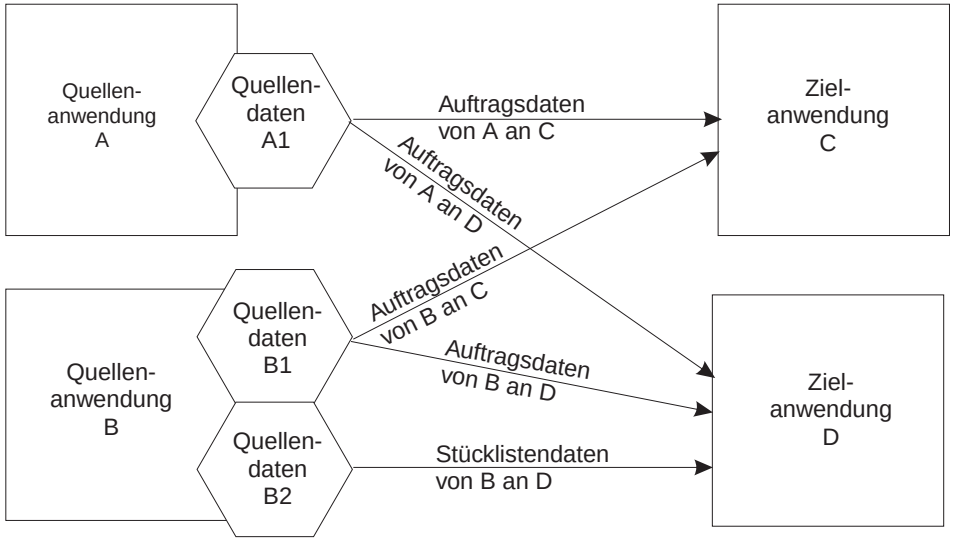


Abbildung 3. Datenfluss zwischen Anwendungen in einer einfachen Konfiguration

In Abb. 3 sind folgende Datenflüsse erkennbar: Anwendung A sendet Auftragsdaten an die Anwendungen C und D, Anwendung B sendet ebenfalls Auftragsdaten an die Anwendungen C und D; außerdem sendet Anwendung B Stücklistendaten an Anwendung D. In allen Fällen werden Daten in ein Industriestandardformat konvertiert, bevor sie an die Zielanwendungen gesendet werden. Dabei werden die Auftragsdaten von den Anwendungen A und B in ein XML-Standardformat für Auftragsdaten konvertiert. Die Stücklistendaten von Anwendung B werden in ein XML-Standardformat für Stücklistendaten konvertiert.

Die Daten werden über einen Transportmechanismus für die Übertragung, z. B. MQSeries, oder eine Implementierung von Java Message Service (JMS), an die Zielanwendung(en) gesendet. Die integrierte Nachricht wird in das *Marshaling-Format*, das für den verwendeten Transportmechanismus für die Übertragung erforderlich ist, konvertiert und dann an den Transportmechanismus (z. B. eine MQSeries-Warteschlange) übergeben. Jede Zielanwendung kann zum Empfangen von Nachrichten einen anderen Transportmechanismus für die Übertragung und ein anderes Marshaling-Format verwenden. Anwendung C kann Nachrichten beispielsweise über MQSeries empfangen, während Anwendung D Nachrichten über JMS empfängt (siehe Abb. 4 auf Seite 71). In diesem Fall werden alle integrierten Nachrichten für Anwendung C (die Auftragsdaten von den Anwendungen A und B) in ein Marshaling-Format für MQSeries und alle integrierten Nachrichten für Anwendung D (die Auftragsdaten von den Anwendungen A und B sowie

die Stücklistendaten von Anwendung B) in ein Marshaling-Format für JMS konvertiert. MQSeries Adapter Kernel verwendet für diese Konvertierungen folgenden Mechanismus:

- Ein *Quellenadapter* konvertiert Anwendungsdaten in eine integrierte Nachricht. Quellenadapter werden mit MQSeries Adapter Builder erstellt.
- Der *interne Adapter* konvertiert die integrierte Nachricht in eine *Übertragungsnachricht*. Der interne Adapter verwendet den *logischen Nachrichtenservice* (Logical Message Service, LMS), um die Nachricht für die Übertragung über den Transportmechanismus zu konvertieren; der verwendete LMS ist vom jeweiligen Transportmechanismus für die Übertragung abhängig. Mit Hilfe eines *Formatierungsprogramms* führt der LMS ein Marshaling der Nachricht für den Transport durch.

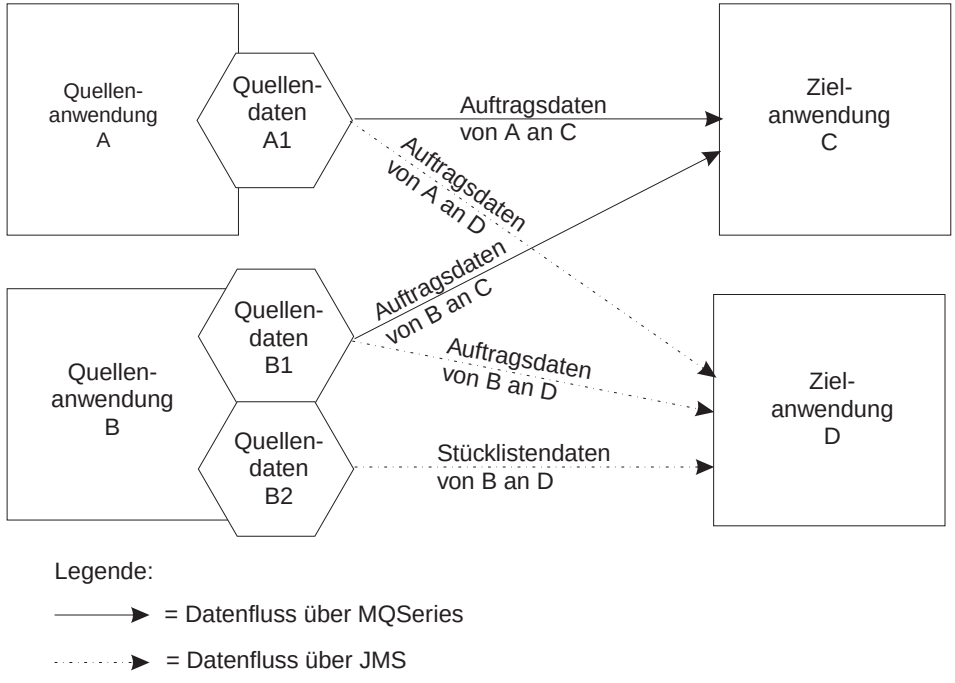


Abbildung 4. Eine einfache Konfiguration mit Anwendungen, die verschiedene Transportmechanismen für die Übertragung miteinander verbunden sind

Der Datenfluss von der Anwendung über die integrierte Nachricht zur Übertragungsnachricht wird in Abb. 5 auf Seite 72 und Abb. 6 auf Seite 72 gezeigt. Nachdem eine Übertragungsnachricht am Ziel empfangen wurde, werden die Konvertierungsvorgänge umgekehrt: Der interne Adapter konvertiert die Übertragungsnachricht zurück in die integrierte Nachricht, anschlie-

ßend konvertiert der Zieladapter die integrierte Nachricht bei Bedarf in das Datenformat, das von der Zielanwendung erwartet wird.

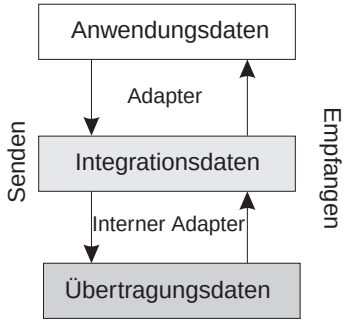


Abbildung 5. Datenkonvertierung

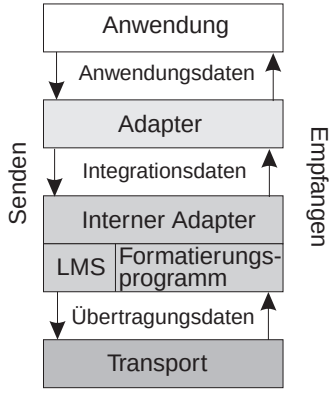


Abbildung 6. Datenfluss

Jede Station der Daten im Datenfluss muss in der Konfigurationsdatei des Kernels definiert werden. Dazu sind drei logische Gruppen mit Konfigurationsdaten erforderlich, die sich durch ihre jeweilige Anwendungs-ID (Quelle oder Ziel) unterscheiden. Eine Anwendungs-ID kann für eine Quellen- oder eine Zielanwendung stehen, je nachdem, ob die Anwendung Nachrichten sendet (Quelle) oder Nachrichten empfängt (Ziel). Die Konfigurationsdatei muss die folgenden Informationen enthalten:

- Übertragungsinformationen
  - Ziel der Daten
  - Verwendeter Transportmechanismus für die Übertragung
  - Verwendete Marshaling-Methode (Formatierungsprogramm) für die Übertragung

  - Zu Grunde liegende Übertragungsvoraussetzungen, z. B. Namen von MQSeries-Warteschlangenmanagern und MQSeries-Warteschlangen bzw. von JMS-QueueConnectionFactory-Objekten und JMS-Warteschlangen.
- Adapterinformationen (nur für die Zielverarbeitungsschicht)
  - Erforderliche Adapter für die Datenverarbeitung
  - Verwendeter Adaptertyp (EAB oder EJB)
  - Zusätzliche Informationen zum verwendeten Adaptertyp
  - Bei Standalone-MQSeries Adapter Kernel: Informationen zum Adapterdämon und zum Adaptermanager
- Sonstige Informationen
  - Trace-Spezifikationen
  - Protokollspezifikationen

Abb. 7 zeigt die Beziehung der Stationen des Datenflusses zu den verschiedenen Abschnitten in der Konfiguration.

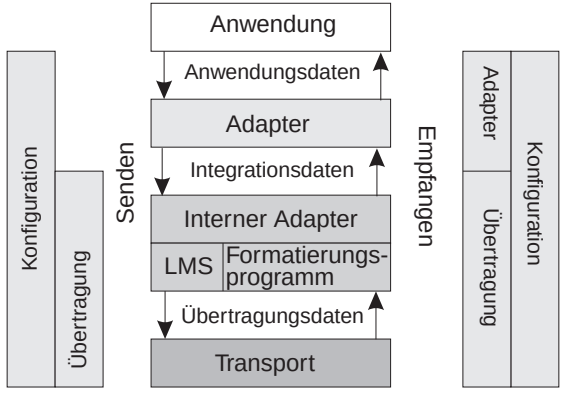


Abbildung 7. Beziehung zwischen Datenfluss und Konfiguration

Ausführliche Informationen zur Zuordnung dieser Konfigurationsdaten zu XML-Elementen in der Datei `aqmconfig.xml` , über die diese Aspekte der Kernel-Konfiguration gesteuert werden, finden Sie unter „Syntax und Organisation der Konfigurationsdatei" auf Seite 76.

Unter „Typische Konfigurationen" auf Seite 88 werden mehrere allgemeine Konfigurationen aufgelistet.

Start- und Konfigurationsdateien

Die Konfiguration des Kernels erfolgt über eine Reihe editierbarer Dateien. Sie können die Dateien in einem Standardtexteditor editieren und für Ihre Anwendungsumgebung konfigurieren. Die folgenden Dateien sind für die Konfiguration wichtig:

- Die Datei `aqmsetenv.bat` (Windows-Systeme) bzw. `aqmsetenv.sh` (UNIX), in der die Umgebungsvariablen gesetzt werden. Editieren Sie diese Datei, um Systemumgebungsvariablen bei Bedarf nach der Installation zu ändern. In dieser Datei werden zum Beispiel die Umgebungsvariablen PATH, CLASSPATH und LIBPATH gesetzt. Auf Windows-Systemen werden diese Variablen automatisch vom Installationsprogramm gesetzt. Damit die Variablen unter UNIX automatisch gesetzt werden, wenn Sie sich anmelden, müssen Sie die in der Datei `aqmsetenv.sh` angegebenen Werte Ihrer `.profile`-Datei (bei Verwendung der Bourne- oder Korn-Shell) bzw. `.cshrc`-Datei (bei Verwendung der C-Shell) hinzufügen.

  Informationen zum Setzen der Umgebungsvariablen unter OS/400 finden Sie in Schritt 6 auf Seite 50.
- Die Datei `aqmsetup`, die verschiedene Werte für die Erstkonfiguration des Kernels enthält. Weitere Informationen finden Sie unter „Die Setup-Datei".
- Die Datei `aqmconfig.xml`, die die Konfigurationsdaten für den Kernel enthält. Weitere Informationen finden Sie unter „Die Konfigurationsdatei" auf Seite 75. Diese Datei enthält die meisten für die Konfiguration des Kernels benötigten Daten.
- Die Datei `aqmcreateq.bat` (Windows-Systeme) bzw. `aqmcreateq.sh` (UNIX und OS/400), die Skripts zum Erstellen von MQSeries-Warteschlangen enthält. Weitere Informationen finden Sie unter „MQSeries-Warteschlangen erstellen" auf Seite 110.

Alle diese Dateien enthalten Kommentare, die das Editieren erleichtern sollen.

Es wird empfohlen, von diesen Dateien Sicherungskopien zu erstellen. Weitere Informationen finden Sie unter „Kernel warten" auf Seite 108.

Die Setup-Datei

Die Setup-Datei, `aqmsetup` enthält eine Reihe der Anfangseinstellungen für den Kernel, einschließlich der folgenden Angaben:

- Das Verzeichnis der Konfigurationsdatei. Weitere Informationen finden Sie unter „Die Konfigurationsdatei" auf Seite 75.
- Das Verzeichnis der XML-DTDs, sofern es nicht das aktuelle Verzeichnis ist.
- Java Native Interface (JNI)-Umgebungsvariablen für die C-Schnittstelle, um die verwendete Speicherkapazität zu ändern.

  Dies gilt dann, wenn ein ausführbares C-Modul einen Prozess startet und von diesem Prozess ein Exemplar der JVM (Java Virtual Machine) erstellt wird.

Die Speicherbelegung kann in diesem Fall dadurch gesteuert werden, dass der Kommentar für die folgenden Zeilen in der Datei `aqmsetup` aufgehoben wird und diese geändert werden:

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

Die Speichergrößen sind in Bytes angegeben.

In „Anhang E. Beispiel der Setup-Datei" auf Seite 139 finden Sie ein Beispiel für die Datei `aqmsetup`. Diese Beispieldatei befindet sich im Installationsverzeichnis `samples` von MQSeries Adapter Kernel.

Editieren Sie bei Bedarf die Setup-Datei, wenn MQSeries Adapter Kernel zum ersten Mal installiert wird. Editieren Sie die Datei nach der Installation nur, wenn der Kernel auf einen Java-Speicherengpass trifft. Gehen Sie dann wie oben beschrieben vor.

Die Konfigurationsdatei

In diesem Abschnitt wird die Datei `aqmconfig.xml` beschrieben, in der die Konfiguration des Kernels festgelegt wird. Unter „Syntax und Organisation der Konfigurationsdatei" auf Seite 76 finden Sie Informationen zur Struktur der Konfigurationsdatei. Hinweise zu einem erfolgreichen Editieren der Konfigurationsdatei finden Sie unter „Konfigurationsdatei editieren" auf Seite 99.

Die Konfiguration von MQSeries Adapter Kernel wird in einer XML-Datei mit dem Namen `aqmconfig.xml` festgelegt. In „Anhang D. Beispiel der Konfigurationsdatei" auf Seite 133 und im Installationsverzeichnis `samples` von MQSeries Adapter Kernel finden Sie ein Beispiel für diese Konfigurationsdatei.

Über die in der Konfigurationsdatei angegebenen Werte werden die folgenden Elemente des Kernels gesteuert:

- Logische Quellen-ID
- Logische Ziel-ID
- Adapterdämonen und Adaptermanager in der Zielverarbeitungsschicht
- Trace-Clients
- Trace-Server

- Marshaling und Routing von Nachrichten, für die die folgenden Spezifikationen erforderlich sind:
  - Die Namen von Empfangs-, Fehler- und Antwortwarteschlangen.
  - Eine oder mehrere Standardzieladressen, an die Nachrichten gesendet werden sollen.
  - Der Name des MQSeries-Warteschlangenmanagers oder des JMS-QueueConnectionFactory-Objekts, das die Nachricht abruft oder sendet.
  - Das Zeitlimit für das Empfangen von Nachrichten oder Antworten.
  - Die Zieladapterklasse in der Zielverarbeitungsschicht des Kernels, von der die einzelnen Nachrichten verarbeitet werden.
  - Zusätzliche Informationen zum Zieladapter.
  - Die Mindestanzahl von Adaptermanagern in der Zielverarbeitungsschicht (bei einem MQSeries Adapter Kernel-Standalone-Betrieb).
  - Aktivieren und Inaktivieren der Trace-Funktion sowie Steuern der Trace-Stufe.
  - Aktivieren und Inaktivieren des Prüfprotokolls.
- Übertragungsmodus

Syntax und Organisation der Konfigurationsdatei

Die Konfiguration von MQSeries Adapter Kernel basiert auf LDAP (Lightweight Directory Access Protocol), d. h. die Struktur der Konfigurationsdatei entspricht den LDAP-Spezifikationen. Das XML-Element der höchsten Ebene, `Epic`, stellt dabei die höchste Ebene des LDAP-Verzeichnisses dar, und XML-Elemente unterhalb des Elements der höchsten Ebene entsprechen untergeordneten LDAP-Objekten. Das XML-Element der höchsten Ebene, `Epic`, stellt dabei die höchste Ebene des LDAP-Verzeichnisses dar, und XML-Elemente unterhalb des Elements der höchsten Ebene entsprechen untergeordneten LDAP-Objekten. Einige XML-Elemente verfügen über erforderliche Attribute, die LDAP-Informationen entsprechen. Bei den Werten, die der Konfigurationsdatei hinzugefügt werden, handelt es sich entweder um den Inhalt von Elementen oder um Attribute von Elementen. Ein Konfigurationswert, der den Inhalt eines Elements angibt, ist zum Beispiel `<epictracelevel>-1</epictracelevel>`, d. h., dem Element `epictracelevel` wird der Wert `-1` (alle möglichen Nachrichten) zugewiesen. Ein Konfigurationswert, der ein Attribut eines Elements angibt, ist zum Beispiel `<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">` , d. h., die Klasse `com.ibm.logging.ConsoleHandler` wird als Trace-Steuerroutine festgelegt.

Im Folgenden sind die Elemente aufgelistet und beschrieben, die in der Konfigurationsdatei als Elemente der höchsten Ebenen verwendet werden können. Eine vollständige Aufzählung und Beschreibung der in der Konfigurationsdatei verwendbaren Elemente finden Sie unter „In der Konfigurationsdatei verwendete XML-Elemente” auf Seite 79.

Die Beispielkonfigurationsdatei zeigt, wie die verschiedenen Elemente im Kontext verwendet werden.

- `Epic` - Das erforderliche Element der höchsten Ebene in der Datei `aqmconfig.xml`.
- `ePICApplications` - Das erforderliche untergeordnete Element des Elements `Epic`.
- `ePICApplication` - Das erforderliche untergeordnete Element des Elements `ePICApplications`. Mit diesem Element werden die Anwendungen, die Services des Kernels nutzen sollen, aufgelistet und definiert; für jede Anwendung ist ein vollständig definiertes Element `ePICApplication` (einschließlich untergeordneter Elemente) erforderlich.
- `AdapterRouting` - Ein optionales untergeordnetes Element des Elements `ePICApplication`. Es definiert den Warteschlangenmanager und zugehörige Informationen.
- `ePICBodyCategory` - Das erforderliche untergeordnete Element des Elements `AdapterRouting`. Es legt die Nachrichtenkategorie für Nachrichten fest, die vom Kernel weitergeleitet werden sollen.
- `ePICBodyType` - Das erforderliche untergeordnete Element des Elements `ePICBodyCategory`. Es legt den Nachrichtenzweck für Nachrichten fest, die vom Kernel weitergeleitet werden sollen. Es enthält Definitionen für Elemente wie Nachrichtenziele, Übertragungsmodi für den Nachrichtenempfang und Formatierungsprogramme für Nachrichten.
- `ePICAdapterDaemonExtensions` - Ein optionales untergeordnetes Element des Elements `ePICApplication`, das einen Adapterdämon angibt. Es enthält Informationen zum Adapterdämon, einschließlich Anwendungs-IDs und Anzahl der Adaptermanager.
- `ePICTraceExtensions` - Ein optionales untergeordnetes Element des Elements `ePICApplication`, das eine Trace-Client-Anwendung oder ein Trace-Server-Element angibt. Es definiert Trace-spezifische Informationen.

Abb. 8 auf Seite 78 zeigt die Struktur der höchsten Ebenen in der Konfigurationsdatei. Es handelt sich dabei nicht um ein funktionsfähiges Beispiel einer Konfigurationsdatei. Es soll lediglich die Beziehungen und Abhängigkeiten zwischen den Elementen der höchsten Ebenen verdeutlichen. Ein vollständiges Beispiel finden Sie in „Anhang D. Beispiel der Konfigurationsdatei” auf Seite 133.

```
<?xml version="1.0" encoding="UTF-8"?>
<Epic o="ePIC">
  <ePICApplications o="ePICApplications">
    <!-- Das folgende Element <ePICApplication> konfiguriert den Kernel für
    die Arbeit mit der Anwendung APP1. -->
    <ePICApplication epicappid="APP1">
      <!-- Die Elemente in diesem Abschnitt geben Protokoll- und Trace-
      Informationen für die Anwendung APP1 an. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Die folgenden Elemente geben den WS-Manager und seine Attribute an. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Die folgenden Elemente geben Details zum Transportieren und
             Verarbeiten von Nachrichten von APP1 an. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    </ePICApplication>
    <!-- Das folgende Element <ePICApplication> startet einen Adapterdämon
    für die Anwendung APP1. -->
    <ePICApplication epicappid="APP1Daemon">
      <!-- Spezifikationen für den Adapterdämon APP1Daemon, der mit der
      Anwendung APP1 arbeitet. -->
      <ePICAdapterDaemonExtensions cn="epicappextensions">
        <epicdepappid>APP1</epicdepappid>
        <epicminworkers>1</epicminworkers>
      </ePICAdapterDaemonExtensions>
    </ePICApplication>
    <!-- Das folgende Element <ePICApplication> konfiguriert den Kernel für
    die Arbeit mit der Anwendung APP2. -->
    <ePICApplication epicappid="APP2">
      <!-- Die folgenden Elemente geben Protokoll- und Trace-Informationen
      Anwendung APP1 an. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Die folgenden Elemente geben den WS-Manager und seine Attribute an. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Die folgenden Elemente geben Details zum Transportieren und
             Verarbeiten von Nachrichten von APP2 an. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- Das folgende Element <ePICApplication> konfiguriert einen
    Trace-Client mit dem Namen TraceClient. -->
    <ePICApplication epicappid="TraceClient">
      <ePICTraceExtensions cn="epicappextensions">
        <!-- Die folgenden Elemente geben Attribute des Trace-Clients an. -->
      </ePICTraceExtensions>
    </ePICApplication>
  </ePICApplications>
</Epic>
```

Abbildung 8. Struktur der Elemente der höchsten Ebene in der Konfigurationsdatei

Im Folgenden sind alle Elemente aufgelistet und beschrieben, die in der Konfigurationsdatei verwendet werden können. Für alle Elemente, für die ein Standardwert angegeben ist, gilt, dass der Kernel den Standardwert verwendet, wenn für eines dieser Elemente in der Konfigurationsdatei ein Wert erforderlich, aber nicht explizit angegeben ist.

In der Konfigurationsdatei verwendete XML-Elemente

Epic Element der höchsten Ebene in der Konfigurationsdatei.

Untergeordnete Elemente:

- `context`
- `ePICApplications` (erforderlich)

Attribute: `o="ePIC"` (erforderlich)

context
Gibt bei der Verwendung von JMS-Objekten das Stammverzeichnis für den JNDI-Dateisystemkontext (FSContext) an. Standardmäßig ist dies das aktuelle Verzeichnis. Diese Angabe ist bei Verwendung von JMS erforderlich. Weitere Informationen zur Verwendung von JMS-Objekten mit MQSeries Adapter Kernel finden Sie unter „JMS-Objektspeicher verwenden" auf Seite 119.

Untergeordnete Elemente: Keine

Attribute: Keine

ePICApplications
Enthält Informationen zu den Anwendungen, die auf Services des Kernels zugreifen.

Untergeordnete Elemente: `ePICApplication` (erforderlich)

Attribute: `o="ePICApplications"` (erforderlich)

ePICApplication
Gibt Informationen zu einer einzelnen Anwendung an, die auf Services des Kernels zugreift.

Untergeordnete Elemente:

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

Attribute: `epicappid="Anwendungs-ID"`, wobei *Anwendungs-ID* für eine gültige Anwendungs-ID steht (erforderlich)

epiclogging
Gibt an, ob ein Prüfprotokoll erstellt werden soll. Für die Erstellung von Prüfprotokollen ist das Produkt WebSphere Business Integrator erforderlich. Der Standardwert ist `false`.

Untergeordnete Elemente: Keine

Attribute: Keine

epictrace
Gibt an, ob Traces durchgeführt werden sollen. Der Standardwert ist `false`.

Untergeordnete Elemente: Keine

Attribute: Keine

epictracelevel
Stellt anhand der für die Klasse `com.ibm.logging.IRecordType` angegebenen Werte die Trace-Stufe ein. Der Standardwert ist `0` (keine Nachrichten). Weitere Informationen zu Traces und eine vollständige Liste der gültigen Trace-Stufen finden Sie im Handbuch *Problem Determination Guide*.

Untergeordnete Elemente: Keine

Attribute: Keine

epictraceclientid
Gibt den Namen der Trace-Client-Anwendung an. Der Standardwert ist `TraceClient`.

Untergeordnete Elemente: Keine

Attribute: Keine

epiclogoninfoclassname
Gibt den Namen einer Anmeldeklasse an, die bei einem EAB-Adapter für die Herstellung der Verbindung mit der Anwendung verwendet wird. Der Standardwert ist `com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault`.

Untergeordnete Elemente: Keine

Attribute: Keine

AdapterRouting
Enthält Informationen zu Nachrichtenarten und zum Routing von Nachrichten.

Untergeordnete Elemente:

- epicmqppqueuemgr
- epicuseremotequeuemanagertosend
- epicmqppqueuemgrhostname
- epicmqppqueuemgrportnumber
- epicmqppqueuemgrchannelname
- epicjmsconnectionfactoryname
- ePICBodyCategory (erforderlich)

Attribute: cn="epicadapterrouting" (erforderlich)

epicmqppqueuemgr
Gibt den Namen des zu verwendenden Warteschlangenmanagers an, wenn MQSeries als Transportmechanismus eingesetzt wird. Ist dieses Element gar nicht oder mit dem Standardwert DEFAULT angegeben, wird der standardmäßige Warteschlangenmanager verwendet.

Untergeordnete Elemente: Keine

Attribute: Keine

epicuseremotequeuemanagertosend
Gibt an, ob ein ferner Warteschlangenmanager zum Senden von Nachrichten verwendet werden soll, wenn MQSeries als Transportmechanismus eingesetzt wird. Der Standardwert ist false.

Untergeordnete Elemente: Keine

Attribute: Keine

epicmqppqueuemgrhostname
Gibt den TCP/IP-Host-Namen der Maschine mit dem Warteschlangenmanager an, wenn MQSeries als Transportmechanismus eingesetzt wird. Diese Angabe ist nur erforderlich, wenn MQSeries Client verwendet wird.

Untergeordnete Elemente: Keine

Attribute: Keine

epicmqppqueuemgrportnumber
Gibt die Anschlussnummer für den Serverprozess des Warteschlangenmanagers an, wenn MQSeries als Transportmechanismus eingesetzt wird. Der Standardwert ist 1414 (der MQSeries-Standardwert). Diese Angabe ist nur erforderlich, wenn MQSeries Client verwendet wird.

Untergeordnete Elemente: Keine

Attribute: Keine

epicmqppqueuemgrchannelname
Gibt den Kanalnamen des Warteschlangenmanager-Servers an, wenn MQSeries als Transportmechanismus eingesetzt wird. Diese Angabe ist nur erforderlich, wenn MQSeries Client verwendet wird.

Untergeordnete Elemente: Keine

Attribute: Keine

epicjmsconnectionfactoryname
Gibt den Factory-Namen für die JMS-Verbindung an, wenn JMS als Transportmechanismus verwendet wird. Der Wert muss im Format *Attribut=Objekt* angegeben werden, wobei *Attribut* für das LDAP-Attribut und *Objekt* für das JMS-Verbindungsobjekt steht. Das Objekt muss unter dem Element `AdapterRouting` gespeichert sein. Für ein JMS-Verbindungsobjekt mit dem Namen `QCFTEST1` mit einem LDAP-Attribut `cn` muss zum Beispiel `cn=QCFTEST1` als Wert für dieses Element angegeben werden.

Untergeordnete Elemente: Keine

Attribute: Keine

ePICBodyCategory
Gibt die Nachrichtenkategorie der gesendeten Nachrichten an.

Untergeordnete Elemente: `ePICBodyType` (erforderlich)

Attribute: `epicbodycategory=`*`body_category`*, wobei *body_category* die Nachrichtenkategorie der gesendeten Nachrichten angibt (erforderlich)

ePICBodyType
Gibt den Nachrichtenzweck der gesendeten Nachrichten an.

Untergeordnete Elemente:

- `epiccommandclassname`
- `epiccommandtype`
- `epiccommandejbmapper`
- `epiccommandejbmethod`
- `epiccommandejbmethodparmtype`
- `epiccommandejburl`
- `epiccommandejbinitialcontext`
- `epicdestids`
- `epicreceivemode`
- `epicmessageformatter`
- `epicreceivetimeout`
- `epicreceivemqppqueue`
- `epicerrormqppqueue`

- epicreplymqppqueue
- epicjmsreceivequeuename
- epicjmserrorqueuename
- epicjmsreplyqueuename
- epicreceivefiledir
- epiccommitfiledir
- epicerrorfiledir

Attribute: epicbodytype=*body_type*, wobei *body_type* den Nachrichtenzweck der gesendeten Nachrichten angibt (erforderlich)

epiccommandclassname
Gibt den Namen eines EAB-Zieladapters oder EJB-Befehls an, der für die Verarbeitung von Nachrichten aufgerufen wird. Diese Angabe ist erforderlich, wenn ein Adapterdämon oder WebSphere Application Server zum Empfangen von Nachrichten verwendet wird.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommandtype
Gibt den Typ des Zieladapters an. Gültige Werte sind MQAKEAB und MQAKEJB. MQAKEAB gibt einen Standard-EAB-Zieladapter für MQSeries Adapter Kernel an; MQAKEJB gibt an, dass Enterprise-Beans in der Zielverarbeitungsschicht des Kernels unter WebSphere Application Server verwendet werden. Der Standardwert ist MQAKEAB. Der Wert MQAKEJB ist erforderlich, wenn es sich bei dem Zieladapter
um eine Enterprise-Bean handelt.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommandejbmapper
Gibt den Namen der TDCMapper-Klasse an, die zum Zuordnen von Eingabedaten verwendet wird. Der Standardwert ist TDCGenericMapper. Die Angabe ist erforderlich, wenn es sich bei dem Zieladapter um eine Enterprise-Bean handelt.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommandejbmethod
Gibt den Namen der Methode an, die von einer Enterprise-Bean aufgerufen wird. Die Methode muss ein TerminalDataContainer-Objekt als Eingabe akzeptieren und ein TerminalDataContainer-Objekt

zurückgeben. Der Standardwert ist `execute`. Die Angabe ist erforderlich, wenn es sich bei dem Zieladapter um eine Enterprise-Bean handelt.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommandejbmethodparmtype
Gibt den Klassennamen des Objekts an, das als Parameter für die Methode, die von der Enterprise-Bean aufgerufen wird, verwendet wird. Der Standardwert ist der Klassenname des Objekts, das von `TDCMapper` zurückgegeben wird. Die Angabe ist erforderlich, wenn es sich bei dem Zieladapter um eine Enterprise-Bean handelt.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommandejburl
Gibt die URL-Adresse einer eingesetzten Enterprise-Bean im Format `IIOP://`*`Host-Name`*`:`*`Anschluss`* an, wobei *Host-Name* für den Host-Namen des EJB-Servers und *Anschluss* für den Anschluss steht, an dem der Namensserver empfangsbereit ist (standardmäßig `900`). Der Standardwert ist `IIOP:///`. Die Angabe ist erforderlich, wenn es sich bei dem Zieladapter um eine Enterprise-Bean handelt.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommandejbinitialcontext
Gibt den Namen des InitialContextFactory-Objekts an, das für die Suche nach dem Home-Namen der Enterprise-Bean verwendet wird. Der Standardwert ist `com.ibm.ejs.ns.jndi.CNInitialContextFactory`. Die Angabe ist erforderlich, wenn es sich bei dem Zieladapter um eine Enterprise-Bean handelt.

Untergeordnete Elemente: Keine

Attribute: Keine

epicdestids
Gibt die IDs der Anwendungen an, die als Zieladressen für Nachrichten verwendet werden. Diese Angabe ist erforderlich, wenn die Anwendung Nachrichten sendet und die logische Ziel-ID auf `NONE` gesetzt ist.

Untergeordnete Elemente: Keine

Attribute: Keine

epicreceivemode
Gibt den zu verwendenden Übertragungsmodus an. Eine Liste der gültigen Übertragungsmodi mit Erläuterungen finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115. Diese Angabe ist erforderlich, wenn die Anwendung Nachrichten empfängt.

Untergeordnete Elemente: Keine

Attribute: Keine

epicmessageformatter
Gibt das zu verwendende Nachrichtenformatierungsprogramm an; diese Angabe ist von dem Wert für `epicreceivemode` und vom eingesetzten Transportmechanismus abhängig. Weitere Informationen zu Nachrichtenformatierungsprogrammen und Transportmechanismen finden Sie in Tabelle 10 auf Seite 117 und Tabelle 11 auf Seite 117.

Untergeordnete Elemente: Keine

Attribute: Keine

epicreceivetimeout
Gibt die Zeit in Millisekunden an, die der Empfänger auf Nachrichten wartet. Der Standardwert ist `0`. Der Wert `-1` bedeutet, dass kein Zeitlimit besteht (es wird unendlich gewartet).

Untergeordnete Elemente: Keine

Attribute: Keine

epicreceivemqppqueue
Gibt den Namen der Warteschlange an, aus der Nachrichten empfangen werden. Diese Angabe ist erforderlich, wenn das Element `epicreceivemode` einen MQSeries-Übertragungsmodus angibt. Eine Liste der MQSeries-Übertragungsmodi finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115.

Untergeordnete Elemente: Keine

Attribute: Keine

epicerrormqppqueue
Gibt den Namen der Warteschlange an, in die Fehlernachrichten eingereiht werden. Diese Angabe ist erforderlich, wenn eine Fehlernachrichten-Warteschlangensteuerung verwendet wird und das Element `epicreceivemode` einen MQSeries-Übertragungsmodus angibt. Eine Liste der MQSeries-Übertragungsmodi finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115.

Untergeordnete Elemente: Keine

Attribute: Keine

epicreplymqppqueue
Gibt den Namen der Warteschlange an, aus der Antwortnachrichten empfangen werden. Diese Angabe ist erforderlich, wenn Antwortanforderungen verwendet werden und das Element `epicreceivemode` einen MQSeries-Übertragungsmodus angibt. Eine Liste der MQSeries-Übertragungsmodi finden Sie in „Anhang A. Übertragungsmodus" auf Seite 115.

Untergeordnete Elemente: Keine

Attribute: Keine

epicjmsreceivequeuename
Gibt den Namen der Warteschlange an, aus der Nachrichten empfangen werden. Diese Angabe ist für den Übertragungsmodus JMS erforderlich. Das Objekt muss unter dem Element `ePICBodyType` gespeichert sein. Der Wert muss im Format *`Attribut=Objekt`* angegeben werden, wobei *Attribut* für das LDAP-Attribut und *Objekt* für den Namen des JMS-Warteschlangenobjekts steht. Für ein JMS-Objekt mit dem Namen `TEST1AIQ` mit einem LDAP-Attribut `cn` wird zum Beispiel `cn=TEST1AIQ` als Wert für dieses Element angegeben.

Untergeordnete Elemente: Keine

Attribute: Keine

epicjmserrorqueuename
Gibt den Namen der Warteschlange an, in die Fehlernachrichten eingereiht werden. Diese Angabe ist erforderlich, wenn eine Fehlernachrichten-Warteschlangensteuerung in Verbindung mit dem Übertragungsmodus JMS verwendet wird. Das Objekt muss unter dem Element `ePICBodyType` gespeichert sein. Der Wert muss im Format *`Attribut=Objekt`* angegeben werden, wobei *Attribut* für das LDAP-Attribut und *Objekt* für den Namen des JMS-Warteschlangenobjekts steht. Für ein JMS-Objekt mit dem Namen `TEST1AEQ` mit einem LDAP-Attribut `cn` wird zum Beispiel `cn=TEST1AEQ` als Wert für dieses Element angegeben.

Untergeordnete Elemente: Keine

Attribute: Keine

epicjmsreplyqueuename
Gibt den Namen der Warteschlange an, aus der Antwortnachrichten empfangen werden. Diese Angabe ist erforderlich, wenn Antwortanforderungen in Verbindung mit dem Übertragungsmodus JMS verwendet werden. Das Objekt muss unter dem Element `ePICBodyType` gespeichert sein. Der Wert muss im Format *`Attribut=Objekt`* angegeben werden, wobei *Attribut* für das LDAP-Attribut und *Objekt* für den Namen des JMS-Warteschlangenobjekts steht. Für ein JMS-Objekt mit

dem Namen TEST1RPL mit einem LDAP-Attribut cn wird zum Beispiel cn=TEST1RPL als Wert für dieses Element angegeben.

Untergeordnete Elemente: Keine

Attribute: Keine

epicreceivefiledir
Gibt den Namen des Verzeichnisses an, aus dem Nachrichten empfangen werden. Diese Angabe ist für den Übertragungsmodus FILE erforderlich.

Untergeordnete Elemente: Keine

Attribute: Keine

epiccommitfiledir
Gibt den Namen des Verzeichnisses an, in dem empfangene Nachrichten aufbewahrt werden, bis sie festgeschrieben (COMMIT-Operation) werden. Diese Angabe ist für den Übertragungsmodus FILE erforderlich, wenn Nachrichten empfangen werden.

Untergeordnete Elemente: Keine

Attribute: Keine

epicerrorfiledir
Gibt den Namen des Verzeichnisses an, in das Fehlernachrichten eingereiht werden. Diese Angabe ist erforderlich, wenn eine Fehlernachrichten-Warteschlangensteuerung in Verbindung mit dem Übertragungsmodus FILE verwendet wird.

Untergeordnete Elemente: Keine

Attribute: Keine

ePICAdapterDaemonExtensions
Enthält Informationen zu Adapterdämon-Erweiterungen.

Untergeordnete Elemente:

- epicdepappid
- epicminworkers

Attribute: cn="epicappextensions" (erforderlich)

ePICTraceExtensions
Enthält Informationen zu Trace-Erweiterungen. Eine ausführliche Beschreibung dieses Elements und der untergeordneten Elemente finden Sie im Handbuch *Problem Determination Guide*.

Untergeordnete Elemente:

- epicdepappid
- epictracesyncoperation
- epictracemessagefile
- epictracehandler
- ePICTraceHandler

Attribute: cn="epicappextensions" (erforderlich)

epicdepappid
Gibt die ID der Anwendung an, die auf Services des Adapterdämons zugreift. Als Standardwert wird die Anwendungs-ID verwendet, mit der der Adapterdämon gestartet wurde.

Untergeordnete Elemente: Keine

Attribute: Keine

epicminworkers
Gibt die Anzahl der Adaptermanager an, die vom Adapterdämon gestartet werden. Der Standardwert ist 1.

Untergeordnete Elemente: Keine

Attribute: Keine

Typische Konfigurationen

In diesem Abschnitt werden die Konfigurationswerte für einige typische Konfigurationsszenarios aufgelistet, einschließlich der Werte zum Senden und Empfangen von Nachrichten über verschiedene Transportmechanismen für die Übertragung. Wenn eine Nachricht gesendet wird, werden Konfigurationswerte sowohl von der Quellenverarbeitungsschicht als auch der Zielverarbeitungsschicht abgerufen; wird eine Nachricht empfangen, werden Konfigurationswerte nur von der Zielverarbeitungsschicht abgerufen. Quelle und Ziel werden durch ihre jeweiligen logischen IDs repräsentiert. Bei den Beispielen in diesem Abschnitt wird davon ausgegangen, dass sich die Quelle und das Ziel auf zwei verschiedenen Maschinen befinden. Wenn die ID der Zielanwendung noch nicht festgelegt ist, wird sie aus dem Wert des Elements epicdestids in der Konfiguration der Quelle ermittelt.

Anmerkung: Die Konfigurationsszenarios listen die Konfigurationswerte auf, die für Elemente gültig sind und angegeben werden können. Informationen zu den gültigen Standardwerten für das jeweilige Element finden Sie in der Liste des Abschnitts „In der Konfigurationsdatei verwendete XML-Elemente” auf Seite 79.

Typische MQSeries-Konfigurationen: Dieser Abschnitt beschreibt typische Konfigurationen bei der Verwendung von MQSeries als Transportmechanismus für die Übertragung. Das Element `epicreceivemode` gibt einen MQSeries-Übertragungsmodus an (z. B. MQPP oder MQRFH2). Folgende Szenarios werden aufgelistet:

- Tabelle 2 zeigt Konfigurationselemente, die für das Senden einer Nachricht von einem MQSeries-Server an einen anderen MQSeries-Server angegeben werden müssen.
- Tabelle 3 auf Seite 90 zeigt Konfigurationselemente, die für das Senden einer Nachricht von einem MQSeries-Server, der einen fernen Warteschlangenmanager verwendet, an einen anderen MQSeries-Server angegeben werden müssen.
- Tabelle 4 auf Seite 91 zeigt Konfigurationselemente, die für das Senden einer Nachricht von einem MQSeries-Client, der einen Host-Server verwendet, an einen MQSeries-Server angegeben werden müssen.
- Tabelle 5 auf Seite 92 zeigt Konfigurationselemente, die für das Empfangen einer Nachricht auf einem MQSeries-Server angegeben werden müssen.
- Tabelle 6 auf Seite 93 zeigt Konfigurationselemente, die für das Empfangen einer Nachricht auf einem MQSeries-Client, der einen Host-Server verwendet, angegeben werden müssen.

Tabelle 2. Typische Konfiguration: Senden einer Nachricht von einem MQSeries-Server an einen anderen MQSeries-Server

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicreceivemode` gibt einen MQSeries-Übertragungsmodus an.
Das Element `epicmqppqueuemgr` gibt den Namen des Warteschlangenmanagers an. Dieser Warteschlangenmanager muss auf der Maschine mit der Quellenanwendung vorhanden sein.	
	Das Element `epicreceivemqppqueue` gibt den Namen der Empfangswarteschlange an. Diese Warteschlange muss eine ferne MQSeries-Warteschlange auf der Maschine mit der Zielanwendung sein oder einem MQSeries-Cluster angehören.

Tabelle 2. Typische Konfiguration: Senden einer Nachricht von einem MQSeries-Server an einen anderen MQSeries-Server (Forts.)

Quellenkonfiguration	Zielkonfiguration
Das Element `epicreplymqppqueue` gibt den Namen der Antwortwarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Maschine des Senders sein oder einem MQSeries-Cluster angehören. Das Element wird nur für synchrone Anforderungen und Antworten verwendet.	
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine Antwort wartet, bevor er die Verbindung beendet.	

Tabelle 3. Typische Konfiguration: Senden einer Nachricht von einem MQSeries-Server an einen anderen MQSeries-Server über einen fernen Warteschlangenmanager

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicreceivemode` gibt einen MQSeries-Übertragungsmodus an.
Das Element `epicmqppqueuemgr` gibt den Namen des Warteschlangenmanagers an. Dieser Warteschlangenmanager muss auf der Maschine mit der Quellenanwendung vorhanden sein.	Das Element `epicmqppqueuemgr` gibt den Namen des Warteschlangenmanagers an. Dieser Warteschlangenmanager muss auf der Maschine mit der Zielanwendung vorhanden sein. Der Name muss angegeben werden; es kann kein Standardwert verwendet werden.
Das Element `epicremotequeuemanagertosend` gibt an, dass Nachrichten über einen fernen Warteschlangenmanager gesendet werden.	
	Das Element `epicreceivemqppqueue` gibt den Namen der Empfangswarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Maschine mit der Zielanwendung sein oder einem MQSeries-Cluster angehören.

Tabelle 3. Typische Konfiguration: Senden einer Nachricht von einem MQSeries-Server an einen anderen MQSeries-Server über einen fernen Warteschlangenmanager (Forts.)

Quellenkonfiguration	Zielkonfiguration
Das Element `epicreplymqppqueue` gibt den Namen der Antwortwarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Maschine des Senders sein oder einem MQSeries-Cluster angehören. Das Element wird nur für synchrone Anforderungen und Antworten verwendet.	
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine Antwort wartet, bevor er die Verbindung beendet.	

Tabelle 4. Typische Konfiguration: Senden einer Nachricht von einem MQSeries-Client, der einen Host-Server verwendet, an einen MQSeries-Server

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicreceivemode` gibt einen MQSeries-Übertragungsmodus an.
Das Element `epicmqppqueuemgr` gibt den Namen des Warteschlangenmanagers an. Dieser Warteschlangenmanager muss auf der Host-Maschine des Sender-Clients vorhanden sein.	
Das Element `epicmqppqueuemgrhostname` gibt den Host-Namen der MQSeries-Servermaschine an.	
Das Element `epicmqppqueuemgrportnumber` gibt die Anschlussnummer für den Serverprozess des Warteschlangen-managers auf der Servermaschine an.	
Das Element `epicmqppqueuemgrchannelnumber` gibt die Kanalnummer des Servers des Warte-schlangenmanagers an.	

Tabelle 4. Typische Konfiguration: Senden einer Nachricht von einem MQSeries-Client, der einen Host-Server verwendet, an einen MQSeries-Server (Forts.)

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicreceivemqpqueue` gibt den Namen der Empfangswarteschlange an. Diese Warteschlange muss eine ferne MQSeries-Warteschlange auf der Maschine mit der Zielanwendung sein oder einem MQSeries-Cluster angehören.
Das Element `epicreplymqpqueue` gibt den Namen der Antwortwarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Host-Maschine des Sender-Clients sein oder einem MQSeries-Cluster angehören. Das Element wird nur für synchrone Anforderungen und Antworten verwendet.	
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine Antwort wartet, bevor er die Verbindung beendet.	

Tabelle 5. Typische Konfiguration: MQSeries-Server empfängt eine Nachricht

Quellenkonfiguration	Zielkonfiguration
Nicht zutreffend.	Das Element `epicreceivemode` gibt einen MQSeries-Übertragungsmodus an.
	Das Element `epicmqppqueuemgr` gibt den Namen des Warteschlangenmanagers an. Dieser Warteschlangenmanager muss auf der Maschine mit der Zielanwendung vorhanden sein.
	Das Element `epicreceivemqpqueue` gibt den Namen der Empfangswarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Zielmaschine sein.

Tabelle 5. Typische Konfiguration: MQSeries-Server empfängt eine Nachricht (Forts.)

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicerrormqppqueue` gibt den Namen der Fehlerwarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Zielmaschine sein oder einem Cluster angehören. Diese Angabe ist nur erforderlich, wenn ein Adaptermanager verwendet wird.
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
	Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine zu empfangende Nachricht wartet, bevor er die Verbindung beendet.

Tabelle 6. Typische Konfiguration: MQSeries-Client, der einen Host-Server verwendet, empfängt eine Nachricht

Quellenkonfiguration	Zielkonfiguration
Nicht zutreffend.	Das Element `epicreceivemode` gibt einen MQSeries-Übertragungsmodus an.
	Das Element `epicmqppqueuemgr` gibt den Namen des Warteschlangenmanagers an. Dieser Warteschlangenmanager muss auf der Host-Maschine des Empfänger-Clients vorhanden sein.
	Das Element `epicmqppqueuemgrhostname` gibt den Host-Namen der MQSeries-Servermaschine an.
	Das Element `epicmqppqueuemgrportnumber` gibt die Anschlussnummer für den Serverprozess des Warteschlangenprozesses auf der Servermaschine an.
	Das Element `epicmqppqueuemgrchannelnumber` gibt die Kanalnummer des Servers des Warteschlangenmanagers an.
	Das Element `epicreceivemqppqueue` gibt den Namen der Empfangswarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Host-Maschine des Empfänger-Clients sein.

Tabelle 6. Typische Konfiguration: MQSeries-Client, der einen Host-Server verwendet, empfängt eine Nachricht (Forts.)

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicerrormqppqueue` gibt den Namen der Fehlerwarteschlange an. Diese Warteschlange muss eine lokale MQSeries-Warteschlange auf der Host-Maschine des Empfänger-Clients sein oder einem Cluster angehören. Diese Angabe ist nur erforderlich, wenn ein Adaptermanager verwendet wird.
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
	Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine zu empfangende Nachricht wartet, bevor er die Verbindung beendet.

Typische JMS-Konfigurationen: Dieser Abschnitt beschreibt typische Konfigurationen bei der Verwendung von JMS als Transportmechanismus für die Übertragung. Das Element `epicreceivemode` gibt JMS als Übertragungsmodus an.

Wenn die JMS-Implementierung von MQSeries verwendet wird, müssen die entsprechenden MQSeries-Objekte vorhanden sein. Zum Beispiel muss ein JMS-QueueConnectionFactory-Objekt einem Warteschlangenmanager auf einem MQSeries-Server und eine JMS-Warteschlange einer MQSeries-Warteschlange zugeordnet sein. MQSeries-Objekte müssen nicht in der Konfiguration aufgelistet sein, die MQSeries-Unterstützungsobjekte müssen jedoch vorhanden sein.

Folgende Szenarios werden aufgelistet:

- Tabelle 7 auf Seite 95 zeigt Konfigurationselemente, die für das Senden einer Nachricht über JMS angegeben werden müssen.
- Tabelle 8 auf Seite 96 zeigt Konfigurationselemente, die für das Empfangen einer Nachricht über JMS angegeben werden müssen.

Tabelle 7. Typische Konfiguration: Senden einer Nachricht über JMS

Quellenkonfiguration	Zielkonfiguration
	Das Element `epicreceivemode` gibt JMS als Übertragungsmodus an.
Das Element `epicjmsconnectionfactoryname` gibt den Namen des JMS-QueueConnectionFactory-Objekts an. Das Referenzobjekt muss in der Konfiguration vorhanden sein.	
	Das Element `epicjmsreceivequeuename` gibt den Namen der JMS-Empfangswarteschlange an. Das Referenzobjekt muss in der Konfiguration vorhanden sein.
Das Element `epicjmsreplyqueuename` gibt den Namen der JMS-Antwortwarteschlange an. Das Referenzobjekt muss in der Konfiguration vorhanden sein. Das Element wird nur für synchrone Anforderungen und Antworten verwendet.	
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine Antwort wartet, bevor er die Verbindung beendet.	

Tabelle 8. Typische Konfiguration: Empfangen einer Nachricht über JMS

Quellenkonfiguration	Zielkonfiguration
Nicht zutreffend.	Das Element `epicreceivemode` gibt JMS als Übertragungsmodus an.
	Das Element `epicjmsconnectionfactoryname` gibt den Namen des JMS-QueueConnectionFactory-Objekts an. Das Referenzobjekt muss in der Konfiguration vorhanden sein.
	Das Element `epicjmsreceivequeuename` gibt den Namen der JMS-Empfangswarteschlange an. Das Referenzobjekt muss in der Konfiguration vorhanden sein.
	Das Element `epicjmserrorqueuename` gibt den Namen der JMS-Fehlerwarteschlange an. Das Referenzobjekt muss in der Konfiguration vorhanden sein.
	Das Element `epicmessageformatter` gibt den Namen des verwendeten Formatierungsprogramms an.
	Das Element `epicreceivetimeout` gibt die Zeit an, die der Empfänger auf eine zu empfangende Nachricht wartet, bevor er die Verbindung beendet.

Typische Adapterkonfigurationen: Dieser Abschnitt beschreibt typische Konfigurationen, wenn in der Zielverarbeitungsschicht ein Adapter aufgerufen wird. Welche Konfigurationswerte verwendet werden, ist davon abhängig, ob es sich bei dem Zieladapter um einen Enterprise Access Builder (EAB)-Zieladapter (für Element `epiccommandtype` ist der Wert `MQAKEAB` angegeben) oder einen Session-Bean-Zieladapter des EJB-Service (für Element `epiccommandtype` ist der Wert `MQAKEJB` angegeben) handelt.

Anmerkung: Session-Bean-Zieladapter des EJB-Service werden nur in Verbindung mit WebSphere Application Server unterstützt.

Wenn das Element `epiccommandtype` den Wert `MQAKEAB` hat, geben Sie Werte für die folgenden zusätzlichen Elemente an:

- `epiclogoninfoclassname`
- `epiccommandclassname`

Wenn das Element `epiccommandtype` den Wert `MQAKEJB` hat, geben Sie Werte für die folgenden zusätzlichen Elemente an:

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

Adapterinformationen zur Konfiguration hinzufügen

Wenn der Kernel-Konfiguration ein neuer Adapter hinzugefügt wird, müssen der Konfigurationsdatei einige Mindestspezifikationen hinzugefügt werden. Die Beispielkonfigurationsdatei `aqmconfig.minimum.xml` enthält eine Mindestkonfiguration.

Sie finden diese Datei in „Anhang D. Beispiel der Konfigurationsdatei" auf Seite 133 und im Installationsverzeichnis `samples` von MQSeries Adapter Kernel.

Die folgenden Spezifikationen müssen der Konfigurationsdatei mindestens hinzugefügt werden, wenn die Konfiguration um einen neuen Adapter erweitert wird:

- **Quellenadapter** (sendet Nachrichten):
  - Die ID der Anwendung, unter der der Quellenadapter aktiv ist.

- Der Standard-Warteschlangenmanager. Wenn MQSeries als Transportmechanismus verwendet wird und auf derselben Maschine wie der Quellenadapter installiert und aktiv ist, müssen Sie den Warteschlangenmanager nicht extra konfigurieren.
- Logische Ziel-IDs für Nachrichten. Gehen alle Nachrichten an dieselbe Zieladresse, geben Sie DEFAULT als Nachrichtenkategorie und DEFAULT als Nachrichtenzweck an.
- Eine Empfangswarteschlange für jede logische Ziel-ID, an die der Quellenadapter Nachrichten sendet.

• **Zieladapter** (empfängt Nachrichten):
- Die ID der Anwendung, unter der der Zieladapter aktiv ist.
- Der Standard-Warteschlangenmanager. Wenn MQSeries als Transportmechanismus verwendet wird und auf derselben Maschine wie der Quellenadapter installiert und aktiv ist, müssen Sie den Warteschlangenmanager nicht extra konfigurieren.
- Der Antwortmodus für MQSeries. In der Regel gilt für alle Nachrichten derselbe Modus; geben Sie in diesem Fall DEFAULT als Nachrichtenkategorie und DEFAULT als Nachrichtenzweck an.
- Die Empfangswarteschlange. Wenn für alle Nachrichten dieselbe Warteschlange gilt, geben Sie DEFAULT als Nachrichtenkategorie und DEFAULT als Nachrichtenzweck an.
- Die Fehlerwarteschlange für den Fall, dass während der Nachrichtenverarbeitung durch den Zieladapter ein Fehler auftritt. In der Regel gilt für alle Nachrichten derselbe Modus; geben Sie in diesem Fall DEFAULT als Nachrichtenkategorie und DEFAULT als Nachrichtenzweck an.
- Der Klassenname des Zieladapters, der beim Empfang einer Nachricht aufgerufen werden soll. Diese Angabe ist spezifisch für die Nachrichtenkategorie und den Nachrichtenzweck.
- Zeitlimitüberschreitung bei Empfang. Es wird empfohlen, einen geeigneten Wert anzugeben, um eine zu hohe CPU-Belastung zu vermeiden. In der Regel gilt für alle Nachrichten derselbe Modus; geben Sie in diesem Fall DEFAULT als Nachrichtenkategorie und DEFAULT als Nachrichtenzweck an.

Diese Informationen sind auch für weitere Zieladapter ausreichend, wenn dieselbe Empfangswarteschlange verwendet wird. In diesem Fall ist der Klassenname des Zieladapters, der für die spezifische Nachrichtenkategorie und den spezifischen Nachrichtenzweck aufgerufen wird, die einzige Information, die gesondert angegeben werden muss.

- **Trace-Spezifikationen**:
  - Trace-Funktion ein- oder ausgeschaltet
  - Die Trace-Stufe
  - Zusätzliche Trace-Spezifikationen, einschließlich Trace-Zieladresse, für Quellen- und Zieladapter. Standardmäßig erfolgt die Trace-Ausgabe im Fenster der Eingabeaufforderung oder an dem Terminal, an dem der Kernel gestartet wurde.

Konfigurationsdatei editieren

Verwenden Sie zum Editieren der Konfigurationsdatei einen Texteditor oder einen dedizierten XML-Editor. Für Benutzer von XML-Editoren gibt es im Installationsverzeichnis `samples` des Kernels eine DTD-Datei mit dem Namen `aqmconfig.dtd`. Ein XML-Editor mit dem Namen Xeena kann von der IBM alphaWorks-Website unter www.alphaworks.ibm.com heruntergeladen werden. Beim Editieren der Konfigurationsdatei sollten Sie folgende Empfehlungen beachten:

- Stellen Sie alle erforderlichen Informationen zur gewünschten Konfiguration zusammen, bevor Sie mit dem Editieren der Konfigurationsdatei beginnen. Dazu gehören die Namen der zur Konfiguration gehörenden Anwendungen und Warteschlangen, die auszutauschenden Nachrichtenarten, der oder die verwendeten Übertragungsmodi sowie Informationen zu Trace-Programmen und sonstigen Erweiterungen.
- Kopieren Sie die Beispieldatei `aqmconfig.xml` aus dem Verzeichnis `samples` in das gewünschte Verzeichnis. Benennen Sie die Kopie der Konfigurationsdatei nicht um. Editieren Sie die Kopie.
- Fügen Sie Kommentare ein, um die verschiedenen Abschnitte der Konfigurationsdatei zu beschreiben und die in Ihrer Konfiguration verwendeten spezifischen Werte zu dokumentieren (z. B. Anwendungs-IDs, Namen von Nachrichtenwarteschlangen und Zeitlimitüberschreitungswerte). In XML beginnen Kommentare mit der Zeichenfolge `<!--` und enden mit der Zeichenfolge `-->`. Kommentare können über mehrere Zeilen gehen, wie im folgenden Beispiel gezeigt:

  ```
  <!--
       Kommentar
  -->
  ```

  Beachten Sie, dass XML innerhalb von Kommentaren keine weiteren Kommentare zulässt.
- Organisieren Sie die Konfigurationsdatei nach den Anwendungs-IDs. Fassen Sie die Einträge zu jeder Anwendungs-ID in einem Abschnitt zusammen.
- Wenn Sie keinen dedizierten XML-Editor einsetzen, verwenden Sie einen Texteditor, der beim Speichern der Datei das Zeilenende beibehält und die Zeilen nicht umbricht. Beispiele für solche Editoren sind 'Notepad' auf Windows-Systemen und 'vi' oder 'Emacs' auf UNIX-Systemen.

- Denken Sie daran, dass XML zwischen Groß- und Kleinschreibung unterscheidet; achten Sie also besonders bei Namen von Elementen und Attributen auf die korrekte Schreibweise. Durch eine falsche Schreibweise für ein Element oder Attribut kann die Konfigurationsdatei ungültig werden. Durch die Verwendung eines dedizierten XML-Editors können solche Fehler vermieden werden.
- Wenn Sie für die Nachrichtenkategorie und den Nachrichtenzweck Standardwerte verwenden wollen, diese Werte aber noch nicht automatisch eingetragen sind, müssen Sie für beide Werte in der Konfigurationsdatei den Standardwert DEFAULT angeben. Ohne diese Angaben verwendet der Kernel gar keine Standardwerte.
- Überprüfen Sie die Konfigurationsdatei, bevor Sie den Produktionsbetrieb aufnehmen. Weitere Informationen finden Sie unter „Konfigurationsdatei überprüfen".
- Die Änderungen in der Konfigurationsdatei werden wirksam, sobald anschließend ein Kernel-Prozess gestartet wird. Ist ein Prozess aktiv, während Sie die Konfigurationsdatei ändern, müssen Sie ihn stoppen und erneut starten, damit die Änderungen wirksam werden. Seien Sie besonders vorsichtig, wenn Sie die Konfigurationsdatei für den aktuellen Produktionsbetrieb ändern.
- Erstellen Sie nach jedem Editieren der Konfigurationsdatei eine Sicherung.

Konfigurationsdatei überprüfen

Es wird empfohlen, die Konfigurationsdatei nach dem Editieren zu überprüfen, bevor sie für den Produktionsbetrieb verwendet wird. Gehen Sie zum Überprüfen der Konfigurationsdatei folgendermaßen vor:

1. Erstellen Sie ein Verzeichnis für die Gültigkeitsprüfung der Konfigurationsdatei, in dem Sie den Test für die Gültigkeitsprüfung erstellen und ausführen.
2. Erstellen Sie eine XML-Nachricht für die Gültigkeitsprüfung.
3. Richten Sie Nachrichtenwarteschlangen für den Test ein.
4. Erstellen Sie den Test für die Gültigkeitsprüfung der Konfigurationsdatei, bei dem eine Nachricht gesendet und eine Nachricht empfangen wird, und führen Sie ihn aus.
5. Werten Sie die Ergebnisse des Tests aus, um festzustellen, ob die Konfigurationsdatei korrekt ist.

Sowohl das Dienstprogramm zum Erstellen einer XML-Nachricht für die Gültigkeitsprüfung als auch der Test für die Gültigkeitsprüfung der Konfigurationsdatei werden standardmäßig vom Kernel zur Verfügung gestellt.

Der Test für die Gültigkeitsprüfung der Konfigurationsdatei ruft die Methode `sendMsg` auf und sendet eine XML-Nachricht für die Gültigkeitsprüfung von einem internen Adapter in der Quellenverarbeitungsschicht des Kernels an einen Adapterdämon in der Zielverarbeitungsschicht des Kernels. Ein Quellen- oder Zieladapter ist für diesen Test nicht erforderlich. Wenn jedoch ein Zieladapter vorhanden ist, können Sie die Testnachricht auch an die betreffende Zielanwendung senden.

Das Prüfverfahren wird im Folgenden beschrieben.

Anmerkung: Um die Durchführung des Verfahrens zu erleichtern, werden einige Skripts zur Verfügung gestellt. Sie können diese Skripts bei Bedarf kopieren und die Kopien anschließend editieren, um sie an Ihre Erfordernisse anzupassen. Beachten Sie, dass unter OS/400 die UNIX-Versionen der Skripts in einer **qsh**-Sitzung ausgeführt werden können. Sie starten eine **qsh**-Sitzung, indem Sie den Befehl **STRQSH** (Start QSH) an der CL-Eingabeaufforderung eingeben.

Schritt 1. Öffnen Sie eine Eingabeaufforderung.

Schritt 2. Erstellen Sie ein Verzeichnis für die Gültigkeitsprüfung der Konfigurationsdatei. Kopieren Sie die Konfigurationsdatei und die Setup-Datei in dieses Verzeichnis.

Schritt 3. Wechseln Sie in das Verzeichnis für die Gültigkeitsprüfung.

Schritt 4. Geben Sie den folgenden Befehl zum Erstellen einer XML-Nachricht für die Gültigkeitsprüfung ein:

- `aqmcrtmsg.bat` (Windows-Systeme)
- `aqmcrtmsg.sh` (UNIX und OS/400)

Schritt 5. Es wird eine Liste mit Optionen angezeigt. Wählen Sie eine Option aus, und drücken Sie die Eingabetaste. Geben Sie für jede ausgewählte Option einen Wert ein. Die Werte können in einer beliebigen Reihenfolge eingegeben werden. Beispiele für Optionen sind `set sourcelogicalid`, `set msgtype` und `set bodycategory`. Sie müssen Werte für die Optionen 20, 21, 22 und 23 eingeben. Über die Option 24 bzw. 241 können Sie einen Nachrichteninhalt eingeben. Weitere Werte sind nicht erforderlich.

Schritt 6. Geben Sie Option 1 zum Erstellen der XML-Datei für die Gültigkeitsprüfung ein. Die XML-Datei für die Gültigkeitsprüfung wird im aktuellen Verzeichnis erstellt und erhält den Namen `EpicMessage`*nn*`.xml`, wobei *nn* für die Nummer der XML-Datei steht.

Schritt 7. Geben Sie Option 0 zum Verlassen des Dienstprogramms für die Gültigkeitsprüfung ein.

Schritt 8. Richten Sie die benötigten Nachrichtenwarteschlangen für die Gültigkeitsprüfung ein.

Schritt 9. Stellen Sie die Umgebungsvariable AQMSETUPFILE vorübergehend so ein, dass sie auf die Setup-Datei im Verzeichnis für die Gültigkeitsprüfung zeigt.

- Geben Sie auf Windows-Systemen an einer Eingabeaufforderung Folgendes ein:

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

  Dabei steht `E:\` für das betreffende Laufwerk und *Laufzeitdateien* für das Verzeichnis für die Gültigkeitsprüfung.
- Geben Sie unter UNIX und OS/400 den weiter unten folgenden Befehl ein. Bei dem Befehlsbeispiel wird davon ausgegangen, dass Sie die Korn-Shell verwenden. Wenn Sie eine andere Shell verwenden, ändern Sie den Befehl entsprechend.

  ```
  export AQMSETUPFILE=Stammverzeichnis/Laufzeitdateien/aqmsetup
  ```

  Dabei steht *Stammverzeichnis* für das Installationsverzeichnis des Kernels und *Laufzeitdateien* für das Verzeichnis für die Gültigkeitsprüfung. Unter OS/400 muss sich die Datei `aqmsetup` immer in Ihrem IFS-Stammverzeichnis befinden (`/home/`*`Benutzername`*).

Editieren Sie bei Bedarf die Setup-Datei im Verzeichnis für die Gültigkeitsprüfung, sodass sie auf die zu prüfende Konfigurationsdatei zeigt.

Schritt 10. Wählen Sie aus, was getestet werden soll:

- Nur die Quellenverarbeitung des Kernels.
- Das vollständige Routing der Nachricht zur Zielanwendung. Für diesen Test muss bereits ein Zieladapter vorhanden sein.
- Trace-Funktion

Testen Sie zuerst die Quellenverarbeitung und dann die Zielverarbeitung. Schalten Sie den Adapterdämon aus, um nur die Quellenverarbeitung zu testen. Schalten Sie den Adapterdämon ein, um auch die Zielverarbeitung zu testen. Wenn noch kein Zieladapter vorhanden ist, können Sie die Nachrichtenverarbeitung durch den Adapterdämon bis zu dem Punkt testen, an dem er den Befehl für den zugehörigen Zieladapter aufruft. Es wird empfohlen, die Trace-Funktion zu aktivieren, besonders dann, wenn noch kein Zieladapter vorhanden ist.

Schritt 11. Führen Sie den Test für die Gültigkeitsprüfung aus. Geben Sie in einem beliebigen Verzeichnis den folgenden Befehl ein:

- Auf Windows-Systemen:

  ```
  aqmsndmsg.bat -a logische_Quellen_ID -f XML_Nachrichtendatei
  ```
- Unter UNIX und OS/400:

```
aqmsndmsg.sh -a logische_Quellen_ID -f XML_Nachrichtendatei
```

Dabei gilt Folgendes:

logische_Quellen_ID
Gibt die logische Quellen-ID an. Dieser Wert muss der logischen Quellen-ID entsprechen, die in Schritt 5 auf Seite 101 für Option 20 eingegeben wurde.

XML_Nachrichtendatei
Gibt die XML-Nachrichtendatei an.

Anmerkung: Sie können eine Liste mit allen Optionen für diesen Test anzeigen, indem Sie folgenden Befehl eingeben:

Auf Windows-Systemen:

```
aqmsndmsg.bat -?
```

Unter UNIX und OS/400:

```
aqmsndmsg.sh -?
```

Beachten Sie, dass der Parameter `-?` nur in der Korn-Shell funktioniert. Wenn Sie eine andere UNIX-Shell verwenden (die Bourne-Shell oder die C-Shell), fügen Sie vor dem Fragezeichen einen umgekehrten Schrägstrich ein (`-\?`).

Schritt 12. Werten Sie die Ergebnisse aus. Die Nachricht für die Gültigkeitsprüfung enthält die richtige Nachrichtenkategorie, den richtigen Nachrichtenzweck und die richtigen Daten.

- Wenn Sie nur die Quellenverarbeitung des Kernels testen (d. h. der Adapterdämon wurde nicht gestartet), werten Sie die Warteschlange aus, an die die Nachricht weitergeleitet werden sollte.
  - Wenn sich Ihre Nachricht für die Gültigkeitsprüfung in dieser Warteschlange befindet, sind die entsprechenden Einträge in der Konfigurationsdatei gültig.
  - Wenn sich Ihre Nachricht für die Gültigkeitsprüfung nicht in dieser Warteschlange befindet, überprüfen Sie die Ausnahmedatei. Ist die Trace-Funktion aktiviert, überprüfen Sie die Trace-Nachrichten.
- Wenn Sie die Zielverarbeitung des Kernels testen und ein Zieladapter vorhanden ist, überprüfen Sie die Zielanwendung.
  - Wenn Ihre Nachricht für die Gültigkeitsprüfung von der Zielanwendung empfangen wurde, sind die entsprechenden Einträge in der Konfigurationsdatei gültig.

- Wenn Ihre Nachricht für die Gültigkeitsprüfung von der Zielanwendung nicht empfangen wurde, überprüfen Sie die Ausnahmedatei. Ist die Trace-Funktion aktiviert, überprüfen Sie die Trace-Nachrichten.

- Wenn Sie die Zielverarbeitung des Kernels testen, aber kein Zieladapter vorhanden ist, überprüfen Sie, ob sich die Nachricht für die Gültigkeitsprüfung in der Fehlerwarteschlange befindet und ob die Ausnahmedatei eine Ausnahmebedingungsnachricht enthält. Ist die Trace-Funktion aktiviert, überprüfen Sie die Trace-Nachrichten.
  - Wenn sich Ihre Nachricht für die Gültigkeitsprüfung in der Fehlerwarteschlange befindet und eine Ausnahmebedingungsnachricht vorhanden ist, sind die entsprechenden Einträge in der Konfigurationsdatei gültig.
  - Wenn sich Ihre Nachricht für die Gültigkeitsprüfung nicht in der Fehlerwarteschlange befindet, überprüfen Sie die Ausnahmedatei. Ist die Trace-Funktion aktiviert, überprüfen Sie die Trace-Nachrichten.

Schritt 13. Ändern Sie gegebenenfalls die Konfigurationsdatei, und wiederholen Sie die Gültigkeitsprüfung.

MQSeries und MQSeries Integrator konfigurieren

Gehen Sie folgendermaßen vor, um MQSeries und wahlweise installierbare Software wie MQSeries Integrator für die Unterstützung des Kernels zu konfigurieren:

In MQSeries:

- Für die Überprüfung der Installation werden mehrere Warteschlangen verwendet. Wenn Sie diese Warteschlangen für Ihren Test oder Ihre Produktionsumgebung verwenden wollen, löschen Sie deren Inhalt vor der Installationsüberprüfung. Weitere Informationen zu den Warteschlangen, die für die Installationsüberprüfung verwendet werden, finden Sie unter „Prüfverfahren" auf Seite 55.
- Richten Sie Warteschlangen für die Unterstützung des Nachrichtentransports entsprechend dem von Ihnen entworfenen Transportschema ein.
- Setzen Sie beim Erstellen von Warteschlangen den Wert für die Umgebungsvariable MAX_QUEUE_DEPTH auf die maximal zulässige Warteschlangenlänge.

Richten Sie in MQSeries Integrator Eingabe- und Ausgabewarteschlangen in Regeln (Version 1.1) bzw. Nachrichtenflüssen (Version 2) ein, die den Warteschlangen entsprechen, die Sie in der Konfigurationsdatei angegeben haben.

Empfehlungen zur Leistungsverbesserung

Die folgenden Empfehlungen zur Leistungsverbesserung gelten speziell für MQSeries Adapter Kernel:

- Stellen Sie sicher, dass sich bei einer syntaktischen Analyse von XML-DTDs die betreffenden DTD-Dateien in demselben Verzeichnis befinden wie der Prozess, der die Syntaxanalyse ausführt. Dadurch wird der Aufwand des Prozesses zum Auffinden der DTDs verringert.
- Beim Senden und Empfangen langer Nachrichten führt die Nachrichtenart RFH2 zu einer besseren Leistung als die Nachrichtenart XML.

Allgemeine Empfehlungen zur Verbesserung des Leistungsverhaltens finden Sie in der MQSeries-Dokumentation.

Kernel starten

Starten Sie den Kernel, indem Sie die folgenden Komponenten starten:

- Adapterdämonen für die einzelnen Zielanwendungen
- Trace-Server (optional)

Wenn der Quellenadapter im Prozess der Quellenanwendung ausgeführt wird, wird er automatisch mit der Quellenanwendung gestartet, sodass keine weiteren Schritte zum Starten des Quellenadapters unternommen werden müssen. Alle Dämonen oder Server, die Quellenadapter enthalten, müssen gestartet werden. Quellenadapter werden nicht direkt gestartet.

Gehen Sie zum Starten der einzelnen Adapterdämonen und Trace-Server folgendermaßen vor:

Anmerkung: Um die Durchführung des Verfahrens zu erleichtern, werden einige Skripts zur Verfügung gestellt. Sie können diese Skripts bei Bedarf kopieren und die Kopien anschließend editieren, um sie an Ihre Erfordernisse anzupassen. Beachten Sie, dass unter OS/400 die UNIX-Versionen der Skripts in einer **qsh**-Sitzung ausgeführt werden können. Sie starten eine **qsh**-Sitzung, indem Sie den Befehl **STRQSH** (Start QSH) an der CL-Eingabeaufforderung eingeben.

Schritt 1. Starten Sie MQSeries oder eine andere Nachrichtenübermittlungssoftware sowie wahlweise installierbare Software (z. B. MQSeries Integrator).

Schritt 2. Starten Sie alle anderen in Ihrer Anwendungsumgebung benötigten Programme, z. B. Anwendungen (außerhalb des Kernels) zum Lesen von Trace-Nachrichten aus Warteschlangen.

Schritt 3. Öffnen Sie eine Eingabeaufforderung. Geben Sie für jeden Adapterdämon den folgenden Befehl ein:

- Auf Windows-Systemen:

  ```
  aqmstrad.bat -a Anwendungs_ID [-bc Nachrichtenkategorie
  -bt Nachrichtenzweck] [-noretry]
  ```

- Unter UNIX und OS/400:

  ```
  aqmstrad.sh -a Anwendungs_ID [-bc Nachrichtenkategorie
  -bt Nachrichtenzweck] [-noretry]
  ```

  Dabei gilt Folgendes:

  -a *Anwendungs_ID*
  Gibt die logische Ziel-ID der Anwendung an, die auf Services des Adapterdämons zugreift.

  -bc *Nachrichtenkategorie*
  Gibt die Nachrichtenkategorie an, die der Adaptermanager des Adapterdämons verwendet, um den Übertragungsmodus und die Referenzinformationen zum Empfangen von Nachrichten zu ermitteln. Wenn kein Wert angegeben wird, verwendet der Adapterdämon den Standardwert DEFAULT.

  -bt *Nachrichtenzweck*
  Gibt den Nachrichtenzweck an, den der Adaptermanager des Adapterdämons verwendet, um den Übertragungsmodus und die Referenzinformationen zum Empfangen von Nachrichten zu ermitteln. Wenn kein Wert angegeben wird, verwendet der Adapterdämon den Standardwert DEFAULT.

  -noretry
  Gibt an, dass der Adaptermanager automatisch gestoppt wird, wenn keine weiteren Nachrichten vorliegen. Wenn `-noretry` nicht angegeben wird, fragt der Adaptermanager die Warteschlange weiter nach Nachrichten ab und der Adapterdämon muss manuell gestoppt werden.

  Anmerkung: Wenn Sie Java-Startparameter ändern müssen, editieren Sie die Datei `aqmstrad.bat` (Windows-Systeme) bzw. die Datei `aqmstrad.sh` (UNIX und OS/400). Weitere Informationen geben Ihnen die in der jeweiligen Datei enthaltenen Kommentare.

Schritt 4. Geben Sie für jeden Trace-Server den folgenden Befehl ein:

- Auf Windows-Systemen:

  `aqmstrtd.bat -Wie -a ID_der_Quellenanwendung`

- Unter UNIX und OS/400:

  `aqmstrtd.sh -Wie -a ID_der_Quellenanwendung`

  Dabei gilt Folgendes:

  -Wie
  Gibt an, wie die Trace-Nachrichten empfangen werden sollen. Mögliche Werte sind:
  - `socket`
  - `ena`, d. h. interner Adapter

  -a *ID_der_Quellenanwendung*
  Dies ist die ID der Quellenanwendung. Ist kein Wert angegeben, wird der Wert `TraceServer` aus der Konfigurationsdatei verwendet.

Weitere Informationen zu Trace-Servern finden Sie im Handbuch *Problem Determination Guide*.

Schritt 5. Nachdem ein Adapterdämon oder Trace-Server gestartet wurde, bleibt ein Prozessfenster geöffnet, bis Sie den Adapterdämon stoppen. In dem Prozessfenster können Ausnahmebedingungen angezeigt werden. Weitere Informationen finden Sie unter „Ausnahmebedingungsnachrichten" auf Seite 109.

Kernel stoppen

Stoppen Sie den Kernel, indem Sie alle Adapterdämonen und Trace-Server stoppen. Dazu haben Sie verschiedene Möglichkeiten:

- Geben Sie beim Starten des Adapterdämons den Parameter `-noretry` an. Weitere Informationen finden Sie unter „Kernel starten" auf Seite 105.
- Drücken Sie in der Eingabeaufforderung (Windows-Systeme) oder an dem Terminal (UNIX), von dem aus der Adapterdämon oder Trace-Server gestartet wurde, die Tastenkombination **Strg-C**. Führen Sie diesen Schritt für jeden Adapterdämon bzw. Trace-Server aus.
- Auf Windows-Systemen können Sie den Prozess über den Task-Manager beenden.
- Unter UNIX können Sie mit dem Befehl **ps** die Nummer des Prozesses ermitteln und den Prozess anschließend mit dem Befehl **kill** beenden.

Kernel warten

Erstellen Sie einen Wartungsplan für den Kernel. Es wird empfohlen, die folgenden Komponenten regelmäßig zu sichern:

- Die Konfiguration, so wie sie in den folgenden Dateien definiert ist:
  - `aqmconfig.xml`
  - `aqmsetup`
- Adapter, die Sie erstellt haben, einschließlich aller zugehörigen Dateien

Ein Sichern oder regelmäßiges Löschen des Inhalts von Trace- oder sonstigen Dateien, die vom Kernel für seine Verarbeitung verwendet werden, ist nicht erforderlich. Bei Bedarf können Sie jedoch auch diese Dateien sichern. Wenn Trace-Nachrichten nicht in mehrere Dateien, sondern nur in eine einzige Datei weitergeleitet werden, kann diese Datei sehr groß werden. Wenn eine Trace-Stufe eingestellt ist, bei der sehr viele Detailinformationen gesammelt werden (z. B. alle Trace-Nachrichten oder Informationsnachrichten), kann es sinnvoll sein, die Trace-Dateien regelmäßig zu löschen.

Problemdiagnose

Ausnahmebedingungsnachrichten, Trace-Nachrichten und die MQSeries-Fehlerwarteschlange können Ihnen bei der Problemdiagnose helfen. MQSeries Adapter Kernel erstellt Ausnahmebedingungsnachrichten und, falls die Trace-Funktion aktiviert ist, auch Trace-Nachrichten. Informationen zur Problemdiagnose in einer MQSeries Adapter Kernel-Umgebung finden Sie im Handbuch *Problem Determination Guide*.

Für das Verständnis der Ausnahmebedingungsnachrichten und der Trace-Nachrichten ist es wichtig, die Funktionsweise des Kernels zu verstehen. Der Kernel verwendet eine Fehlerwarteschlange für die Bearbeitung bestimmter Fehler. Weitere Informationen finden Sie unter „Funktionsweise des Kernels” auf Seite 10.

An der Kombination aus eindeutiger Nachrichten-ID und eindeutiger Transaktions-ID können Sie erkennen, von welcher Nachricht Ausnahmebedingungsnachrichten und Trace-Nachrichten verursacht wurden.

Es gibt keine ID, an der Sie dieselbe Nachricht in der Fehlerwarteschlange und im Kernel definitiv erkennen können. Sie können eine Nachricht in der Fehlerwarteschlange jedoch manuell mit der entsprechenden Ausnahmebedingungsnachricht oder Trace-Nachricht (oder beiden) korrelieren.

Vergleichen Sie dazu einen oder mehrere der folgenden Werte:

- Ungefähre Zeitmarke
- Warteschlange für die logische Quellen-ID
- Warteschlange für die logische Ziel-ID
- Nachrichtenkategorie
- Nachrichtenzweck
- Eindeutige Nachrichten-ID
- Eindeutige Transaktions-ID

Stimmen die Werte überein, ist die Wahrscheinlichkeit sehr hoch, dass Sie die Nachricht in der Fehlerwarteschlange mit der entsprechenden Ausnahmebedingungsnachricht oder Trace-Nachricht korreliert haben.

Versionsnummer

Führen Sie die Datei `aqmversion.bat` (Windows-Systeme) bzw. `aqmversion.sh` (UNIX und OS/400) im Verzeichnis `bin` aus, um die Versionsnummer des Kernels anzuzeigen.

Ausnahmebedingungsnachrichten

Der Kernel erstellt die folgenden Arten von Ausnahmebedingungsnachrichten:

- Der interne Adapter in der Quellenverarbeitungsschicht des Kernels gibt Ausnahmebedingungen an den Quellenadapter aus. In der MQSeries Adapter Builder-Dokumentation ist beschrieben, wie der Quellenadapter diese Ausnahmebedingungen bearbeitet.
- Der interne Adapter in der Zielverarbeitungsschicht des Kernels gibt Ausnahmebedingungen an den Adaptermanager aus, der den internen Adapter verwaltet.
- Der Adaptermanager schreibt Ausnahmebedingungen in die Datei `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, die sich in demselben Verzeichnis wie der Adaptermanager befindet.
- Der Adapterdämon schreibt Ausnahmebedingungsnachrichten in eine Ausnahmedatei mit dem Namen `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, die sich in demselben Verzeichnis wie der Adapterdämon befindet. Da sich der Adapterdämon und seine Adaptermanager in demselben Verzeichnis befinden, schreiben Sie alle in dieselbe Ausnahmedatei. Der Adapterdämon schickt Ausnahmebedingungsnachrichten außerdem an die Konsole (d. h. an die Eingabeaufforderung bzw. an das Terminal, von dem aus er gestartet wurde, sofern er in einem Fenster gestartet wurde).

Die Trace-Ausnahmebedingungsnachrichten des Kernels unterscheiden sich von den Ausnahmebedingungsnachrichten von MQSeries. Hier ein Beispiel für eine Ausnahmebedingungsnachricht des Kernels:

```
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: Empfangene Ausnahme: <com.ibm.epic.adapters.eak.common.
AdapterException> Nachrichteninformationen: <AQM0114: com.ibm.epic.adapters.eak.
nativeadapter.MQNMRFH2Formatter::convertMessage(MQMessage): Es wurde eine Nach-
richt mit einem MQHRF2-Format erwartet, die empfangene Nachricht hat jedoch das
Format <MQSTR   >.> für <unmarshall Message()> mit ungültigen Daten <(null)>
```

Die Werte in einer Ausnahmebedingungsnachricht sind von der Nachrichtenart abhängig und können unter anderem folgende Elemente enthalten:

- Zeitmarke
- Logische Quellen-ID
- Logische Ziel-ID
- Nachrichtenkategorie
- Nachrichtenzweck
- Eindeutige Nachrichten-ID
- Eindeutige Transaktions-ID
- Ausnahmebedingungsinformation

Eine Beschreibung typischer Probleme, die bei der Installationsüberprüfung auftreten können, und wie Sie darauf reagieren können, finden Sie unter „Typische Probleme bei der Installationsüberprüfung” auf Seite 56.

Trace-Nachrichten

Der Kernel kann so konfiguriert werden, dass er Trace-Nachrichten erstellt. Weitere Informationen zu Traces finden Sie im Handbuch *Problem Determination Guide*.

Dienstprogramme

MQSeries-Warteschlangen erstellen

Mit Hilfe von Stapeldateien oder Shell-Skripts können Sie das Erstellen von MQSeries-Warteschlangen automatisieren. Führen Sie die Datei `aqmcreateq.bat` (Windows-Systeme) bzw. `aqmcreateq.sh` (UNIX und OS/400) aus, wobei Sie den Anwendungsnamen als Parameter angeben. Diese Dateien erstellen für jede Anwendung die folgenden Warteschlangen:

- Eine Empfangswarteschlange mit dem Namen *Anwendungsname*AIQ.
- Eine Fehlerwarteschlange mit dem Namen *Anwendungsname*AEQ.
- Eine Antwortwarteschlange mit dem Namen *Anwendungsname*RPL.

Kapitel 5. APIs von MQSeries Adapter Kernel verwenden

Der Kernel enthält APIs, die für Funktionen wie das Senden und Empfangen von Nachrichten, das Erstellen und Durchführen einer Syntaxanalyse von XML-Formaten sowie das Verwalten der Kernel-Konfiguration verwendet werden. Diese APIs werden von den Adaptern verwendet, die mit MQSeries Adapter Builder erstellt wurden. Das Informationszentrum von MQSeries Adapter Kernel enthält eine zugehörige Online-API-Dokumentation im Javadoc-HTML-Format.

Der Kernel zeigt sein Leistungsvermögen am besten, wenn er zusammen mit Adaptern eingesetzt wird, die vom Benutzer mit MQSeries Adapter Builder erstellt wurden. Er sollte nicht nur für Aufrufe der Kernel-APIs durch Benutzerprogramme verwendet werden. Die Online-API-Dokumentation dient lediglich dazu, die Funktionsweise des Kernels zu erläutern und Hinweise zur Diagnose zu geben.

Die Online-API-Dokumentation befindet sich im Verzeichnis `documentation`.

Kapitel 6. Weitere Informationsquellen

Dieses Kapitel gibt einen Überblick über einige Informationsquellen, die bei der Verwendung von MQSeries Adapter Offering nützlich sein können. Weitere Informationen zu MQSeries Adapter Kernel finden Sie im Dokument *Problem Determination Guide*, das im Informationszentrum von MQSeries Adapter Kernel verfügbar ist. Das Informationszentrum wird mit dem Produkt installiert. Das Dokument *Problem Determination Guide* enthält Informationen zur Lösung spezifischer Probleme, die bei der Verwendung des Kernels auftreten können. Weitere Informationen zu MQSeries Adapter Builder finden Sie im Informationszentrum und in der Onlinehilfefunktion dieses Produkts.

Im Internet verfügbare Informationen

Die Website der MQSeries-Produktfamilie finden Sie unter www.ibm.com/software/ts/mqseries/.
Über die Links auf dieser Website haben Sie folgende Möglichkeiten:

- Abrufen von aktuellen Informationen zur MQSeries-Produktfamilie, einschließlich MQSeries Adapter Offering.
- Zugreifen auf MQSeries-Bücher im HTML- und PDF-Format, gegebenenfalls einschließlich einer aktuelleren Ausgabe dieses Buches. Über den Link www.ibm.com/software/ts/mqseries/library/manualsa/ können Sie direkt auf die MQSeries-Bibliothek zugreifen.
- Herunterladen von MQSeries-SupportPacs

Weitere Informationen zur Verwendung von MQSeries unter OS/400 finden Sie in der OS/400-Bibliothek unter www.ibm.com/servers/eserver/iseries/library/. Hierzu werden auch die OS/400-spezifischen Bücher in der MQSeries-Bibliothek auf der Website www.ibm.com/software/ts/mqseries/library/manualsa/ empfohlen.

Referenzinformationen

In den folgenden Referenzmaterialien finden Sie weitere Informationen zu einigen der in diesem Dokument behandelten Themen:

- Die Website der Open Applications Group (OAG) heißt www.openapplications.org/
- Die W3C-Empfehlung zu XML 1.0 (Extensible Markup Language) finden Sie unter www.w3.org/TR/1998/Rec-xml-19980210

Hierbei handelt es sich nicht um IBM Websites.

Anhang A. Übertragungsmodus

Dieser Anhang enthält Informationen zu den von MQSeries Adapter Kernel unterstützten Übertragungsmodi und zu den Java-Klassen, die zu deren Unterstützung verwendet werden. Einige der Übertragungsmodi werden als Komfortmodi mit Standardformatierungsprogrammen bereitgestellt. Weitere Informationen zu Standardformatierungsprogrammen, die diese Komfortmodi verwenden, finden Sie in Tabelle 10 auf Seite 117.

Die folgenden Übertragungsmodi werden unterstützt:

MQPP Der Kernel transportiert Nachrichten mit Hilfe von MQSeries-Grundfunktionen. Dies ist ein Komfortmodus.

MQRFH1 Der Kernel transportiert Nachrichten mit Hilfe von MQSeries und führt das Nachrichten-Brokering mit Hilfe von MQSeries Integrator Version 1.1 aus. Dies ist ein Komfortmodus.

MQRFH2 Der Kernel transportiert Nachrichten mit Hilfe von MQSeries und führt das Nachrichten-Brokering mit Hilfe von MQSeries Integrator Version 2 aus. Dies ist ein Komfortmodus.

MQBD Der Kernel transportiert Nachrichten mit Hilfe von MQSeries-Grundfunktionen, sendet und empfängt aber nur Nachrichtendaten. Dies ist ein Komfortmodus. Dieser Modus verfügt über die folgenden eindeutigen Merkmale:

- Er kann nur die Nachrichtendaten (Nachrichteninhalt) senden, nicht die Werte aus dem Nachrichten-Header.
- Er kann Nachrichten empfangen, die nur die Nachrichtendaten enthalten. Für empfangene Nachrichten werden die folgenden Header-Standardwerte verwendet:
  - `SourceLogicalApplicationID` - Der Wert im Objekt `ENAService`, das im Aufruf der Empfangsmethode verwendet wird.
  - `BodyCategory` - Der Wert im Objekt `ENAService`, das im Aufruf der Empfangsmethode verwendet wird.
  - `BodyType` - Der Wert im Objekt `ENAService`, das im Aufruf der Empfangsmethode verwendet wird.
  - `Acknowledgment` (Bestätigung) - Wenn es sich bei der empfangenen MQSeries-Nachricht um einen MQSeries-REQUEST handelt, wird der Wert für `Acknowledgment` auf 1 gesetzt.

– `BodyData` - Die von MQSeries empfangenen Nachrichtendaten (der Nachrichteninhalt).

Für alle anderen Header-Werte gelten die Standardwerte.

MQ Der Kernel transportiert Nachrichten mit Hilfe von MQSeries-Grundfunktionen.

JMS Der Kernel transportiert Nachrichten mit Hilfe von Java Message Service (JMS). Weitere Informationen zur Verwendung von JMS-Objekten mit MQSeries Adapter Kernel finden Sie unter „JMS-Objektspeicher verwenden” auf Seite 119.

FILE Der Kernel stellt Nachrichten in eine Datei und ruft sie aus einer Datei ab. Dieser Modus wird nur zu Diagnosezwecken zur Verfügung gestellt.

In Tabelle 9 werden die Übertragungsmodi und die sie unterstützenden Java-Klassen aufgelistet. Alle Java-Klassen stammen aus dem Java-Paket `com.ibm.epic.adapters.eak.nativeadapter`. Beachten Sie, dass jede Java-Klasse, die LMS (Logical Message Service) unterstützt, als Übertragungsmodus angegeben werden kann. In diesem Fall wird die Klasse selbst für die Übertragungsunterstützung verwendet.

Tabelle 9. Übertragungsmodi und unterstützende Java-Klassen

Übertragungsmodus	Java-Klasse	Anmerkungen
MQPP	`LMSMQBindingMQPP`	Erfordert die Installation von MQSeries
MQRFH1	`LMSMQBindingMQRFH1`	Erfordert die Installation von MQSeries
MQRFH2	`LMSMQBindingMQRFH2`	Erfordert die Installation von MQSeries
MQBD	`LMSMQMQBD`	Erfordert die Installation von MQSeries
MQ	`LMSMQBinding`	Erfordert die Installation von MQSeries
JMS	`LMSJMS`	Erfordert die Installation von JMS
FILE	`LMSFile`	Keine

In Tabelle 10 werden die Übertragungsmodi und die ihnen zugeordneten Formatierungsprogrammschnittstellen aufgelistet. Tabelle 11 enthält Querverweise zwischen Formatierungsprogrammschnittstellen und -klassennamen sowie deren Verwendung. Alle Formatierungsprogramme stammen aus dem Java-Paket `com.ibm.epic.adapters.eak.nativeadapter`. Beachten Sie, dass jede Formatierungsprogrammklasse als Übertragungsmodus angegeben werden kann. In diesem Fall wird die angegebene Formatierungsprogrammklasse als Formatierungsprogramm verwendet.

Tabelle 10. Übertragungsmodi und Formatierungsprogrammschnittstellen

Übertragungsmodus	**Formatierungsprogramm-schnittstelle**	**Standardformatierungs-programm**
MQPP	`MQFormatterInterface`	`MQNMXMLFormatter`
MQRFH1	`MQFormatterInterface`	`MQNMRFH1Formatter`
MQRFH2	`MQFormatterInterface`	`MQNMRFH2Formatter`
MQBD	`MQFormatterInterface`	`MQNMBDFormatter`
MQ	`MQFormatterInterface`	`MQNMXMLFormatter`
JMS	`JMSFormatterInterface`	`JMSNMRFH2Formatter`
FILE	`StringFormatterInterface`	`NMXMLFormatter`

Tabelle 11. Formatierungsprogrammschnittstellen, Formatierungsprogrammklassen und deren Funktionen

Formatierungsprogramm-schnittstelle	**Formatierungsprogramm-klasse**	**Funktion**
`MQFormatterInterface`	`MQNMXMLFormatter`	EpicMessage als XML
	`MQNMRFH1Formatter`	EpicMessage als RFH1
	`MQNMRFH2Formatter`	EpicMessage als RFH2
	`MQNMDBFormatter`	Nur Nachrichtendaten
`JMSFormatterInterface`	`JMSNMXMLFormatter`	EpicMessage als XML
	`JMSNMRFH2Formatter`	EpicMessage als RFH2
	`JMSNMBDFormatter`	Nur Nachrichtendaten
`StringFormatterInterface`	`NMXMLFormatter`	EpicMessage als XML

In Tabelle 12 werden die unterstützten LMS-Klassen und deren Grad an Transaktionsunterstützung aufgelistet. Weitere Informationen zur Verwendung von Transaktionen mit MQSeries Adapter Kernel finden Sie unter „Transaktionsfunktionen des Kernels" auf Seite 33.

Tabelle 12. LMS-Klassen und Transaktionsunterstützung

LMS-Klasse	**Transaktionsunterstützung**
LMSMQBindingMQPP	Einphasig
LMSMQBindingMQRFH1	Einphasig
LMSMQBindingMQRFH2	Einphasig
LMSMQMQBD	Einphasig
LMSMQBinding	Einphasig
LMSJMS	Einphasig
LMSFILE	Keine Unterstützung

JMS-Objektspeicher verwenden

Die Namen der JMS-Objekte werden mit Hilfe der Dateiimplementierung FSContext von JNDI gespeichert, die Bestandteil des MQSeries JMS-SupportPac ist. Der Kontext (Verzeichnisstruktur), den der Kernel für FSContext verwendet, entspricht der LDAP-Hierarchie, da das Unterscheidungsattribut mit dem zugeordneten Wert für den Verzeichnisnamen verwendet wird. Für die LDAP-Hierarchie `o=ePIC, o=ePICApplications, epicappid=TEST1` lautet die Verzeichnisstruktur zum Beispiel `o-ePIC/o-ePICApplications/epicappid-TEST1`.

Erstellen Sie den Kontext und die Objekte mit dem JMS-Verwaltungs-Tool (JMS Admin), das zusammen mit JMS installiert wird. Die grundlegenden Schritte bestehen aus dem Erstellen eines Kontextes und der anschließenden Änderung des Kontextes. Ändern des Kontextes heißt, Sie wechseln in diesen Kontext. Erstellen Sie die JMS-Objekte an den dafür vorgesehenen Orten. Das folgende Beispiel enthält Befehle zum Erstellen der Kontextstruktur und von JMS-Objekten. In diesem Beispiel wird `TEST1` als Anwendungs-ID verwendet.

```
#
# 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
#
#
# Dies ist ein Skript zur Verwendung mit dem JMS-Verwaltungs-Tool
# (JMSAdmin), das in MQSeries Support-Pack MA88 enthalten ist.
# Für dieses Tool muss in JMSAdmin.config entweder FSCONTEXT (Datei)
# oder LDAP angegeben sein. Dieses Skript wurde für die Verwendung von
# FSCONTEXT erstellt, für LDAP sind folgende Änderungen erforderlich:
# - Ersetzen der Minuszeichen (-) durch Gleichheitszeichen (=).
#   Beispiel: 'define ctx(o-ePIC)' wird zu 'define ctx(o=ePIC)'
# - In LDAP müssen die Kontexte bereits mit dem LDAP-Verwaltungs-Tool
#   definiert worden sein. Sie müssen zum Beispiel "define ctx(o=ePIC)"
#   nicht definieren, sondern brauchen es nur mit dem Befehl
#   "change ctx(o-ePIC)" zu ändern.
# - Das folgende Skript enthält Anmerkungen, die auf Unterschiede
#   zur Verwendung von LDAP hinweisen.
#
#
# Beispielsyntax: MQSeries root\java\bin\jmsadmin.bat < aqmjmscreatesample.scp
#
```

```
# Einige hilfreiche Befehle:
# "display ctx" zeigt den Kontext an, in dem Sie sich
# gerade befinden.
# "=UP" bedeutet zurück zum übergeordneten Kontext. Beispiel: change ctx(=UP)
# "=INIT" bedeutet zurück zum Stammkontext. In diesem Beispiel ist dies
# eine Verzeichnisebene über o-ePIC. Beispiel: change ctx(=INIT)
# "define xxx" erstellt einen Kontext oder ein Objekt.
# "change xxx" bedeutet Kontext ändern bzw. Kontext wechseln.
#
# Immer erforderlich.
define ctx(o-ePIC)
change ctx(o-ePIC)
# Immer erforderlich.
define ctx(o-ePICApplications)
change ctx(o-ePICApplications)
# Anwendungs-ID ist TEST1, erfordert einen Kontext.
define ctx(epicappid-TEST1)
change ctx(epicappid-TEST1)
# Immer erforderlich.
define ctx(cn-epicadapterrouting)
change ctx(cn-epicadapterrouting)
# Kontext für das JMS-Objekt QueueConnectionFactory.
# Anmerkung: Diese beiden Schritte sind für LDAP nicht erforderlich.
define ctx(cn-QCFTEST1)
change ctx(cn-QCFTEST1)
# Erstellen des JMS-Objekts QueueConnectionFactory mit dem Namen QCFTEST1
# MQSeries wird im Server-Modus (Bindings) verwendet.
define qcf(QCFTEST1) qmgr(yourQManagerName) tran(BIND)
change ctx(=UP)
# Nachrichtenkategorie (BodyCategory) ist DEFAULT
define ctx(epicbodycategory-DEFAULT)
change ctx(epicbodycategory-DEFAULT)
# Nachrichtenzweck (BodyType) ist DEFAULT
define ctx(epicbodytype-DEFAULT)
change ctx(epicbodytype-DEFAULT)
# Kontext für das JMS-Objekt Queue mit dem Namen TEST1AIQ.
# Anmerkung: Diese beiden Schritte sind für LDAP nicht erforderlich.
define ctx(cn-TEST1AIQ)
change ctx(cn-TEST1AIQ)
# Erstellen des JMS-Objekts Queue mit dem Namen TEST1AIQ
# q(Name des Objekts JMS Q) queue(Name der MQSeries-Warteschlange)
define q(TEST1AIQ) queue(TEST1AIQ)
# Kann in Kontext aufsteigen und andere Kontexte und JMS-Objekte definieren.
# Verwaltungs-Tool verlassen.
end
```

Anhang B. Überprüfte Konfigurationen

Es gibt viele mögliche Konfigurationen und Kombinationen von MQSeries, MQSeries Adapter Offering und MQSeries Integrator. Jedes dieser Mitglieder der MQSeries-Produktfamilie verfügt über zahlreiche Funktionen und Konfigurationsmöglichkeiten. Darüber hinaus können Funktionen von MQSeries, MQSeries Adapter Offering und MQSeries Integrator kombiniert werden. Es ist möglich, dass sich Funktionen eines Mitglieds der MQSeries-Produktfamilie teilweise mit den Funktionen eines anderen Mitglieds der Produktfamilie überschneiden. Sie selbst müssen entscheiden, wie Sie die verschiedenen Funktionen für das Routing und die Übermittlung von Nachrichten in MQSeries, MQSeries Adapter Offering und MQSeries Integrator verwenden und kombinieren.

Bis zum Zeitpunkt der Veröffentlichung wurden die folgenden Konfigurationen von MQSeries, MQSeries Adapter Offering und MQSeries Integrator überprüft. Auf der MQSeries-Website können Sie die aktuellen überprüften Konfigurationen finden.

MQSeries Adapter Kernel:

- Senden einer Nachricht mit und ohne Anforderung einer Empfangsbestätigung.
- Verwenden des MQSeries- oder JMS-Übertragungsmodus. Informationen zu gültigen Übertragungsmodi finden Sie unter „Anhang A. Übertragungsmodus" auf Seite 115.
- Nachrichten-Routing und -übermittlung:
  - Senden einer Nachricht von einem Quellenadapter an einen Zieladapter
  - Senden einer Nachricht von einem Quellenadapter an mehrere Zieladapter
  - Multithread-Nachrichtenübermittlung, d. h. mehrere Adaptermanager
  - Setzen der logischen Ziel-ID in der Nachricht auf NONE, sodass die Konfigurationsdatei des Kernels verwendet wird, um die logische Ziel-ID abhängig von der Nachrichtenkategorie, dem Nachrichtenzweck und der logischen Quellen-ID zu bestimmen.
  - Push-Übermittlungsmodell
  - Aktivierte Trace-Funktion

Anmerkung: Siehe „Anhang C. Nachrichten-Header" auf Seite 123. Er enthält eine Beschreibung der Felder für Nachrichten-Header von MQSeries Adapter Kernel, die vom Kernel ausgefüllt und verarbeitet werden.

- Konfiguration mit den unter „Hardware" auf Seite 37 und „Software" auf Seite 38 beschriebenen Voraussetzungen.
- Verwenden der Konfigurationsdatei, nicht LDAP, zum Definieren der Konfiguration.

MQSeries:

- Keine Verwendung von MQSeries-Clustern.

Anmerkung: Siehe „Anhang C. Nachrichten-Header" auf Seite 123. Er enthält eine Beschreibung der Felder für Nachrichten-Header von MQSeries Adapter Kernel, die vom Kernel ausgefüllt und verarbeitet werden.

MQSeries Integrator:

- MQSeries Adapter Kernel und MQSeries können Nachrichten an MQSeries Integrator weiterleiten und übermitteln. In den Produktinformationen von MQSeries Integrator erfahren Sie, wie Sie die Broker-Funktionen für diese Nachrichten anwenden können.
- Senden von Nachrichten aus der Quellenverarbeitungsschicht des Kernels über MQSeries und MQSeries Integrator Version 2 und direktes Routing zur Zielverarbeitungsschicht des Kernels. Innerhalb von MQSeries Integrator ist der Nachrichtenfluss so konfiguriert, dass ein statisches Routing stattfindet. Alle Nachrichten, die am MQInput-Knoten (MQEmpfangsknoten) des Nachrichtenflusses ankommen, werden direkt an eine bestimmte Warteschlange eines MQOutput-Knotens (MQSendeknotens) weitergeleitet.

Anmerkung: Siehe „Anhang C. Nachrichten-Header" auf Seite 123. Er enthält eine Beschreibung der Felder für Nachrichten-Header von MQSeries Adapter Kernel, die vom Kernel ausgefüllt und verarbeitet werden.

Anhang C. Nachrichten-Header

MQSeries Adapter Offering verwendet mehrere Nachrichten-Header. Informationen zu den verschiedenen Headern und unter welchen Bedingungen sie verwendet werden, finden Sie unter „Nachricht und Nachrichtenformat" auf Seite 14.

In diesem Anhang werden die Felder der Nachrichten-Header aufgelistet und beschrieben.

MQSeries Adapter Kernel - Nachrichtendeskriptor-Header

Header-Werte, die von MQSeries Adapter Kernel verwendet werden. Diese Werte werden in Nachrichtenträgerobjekte eingefügt. In der Spalte **Rückgabe in Antworten?** ist angegeben, ob der Wert in einer Antwortnachricht an die Quellenanwendung zurückgegeben wird, wenn die Quellenanwendung eine Antwort anfordert. Einige Werte werden nur mit WebSphere Business Integrator verwendet.

Tabelle 13. MQSeries Adapter Kernel-Header

Header-Name	Rückgabe in Antworten?	Bedeutung bzw. Verwendung
UniqueID	Nein	Eindeutige ID für jede einzelne Nachricht.
TransactionID	Ja	Transaktions-ID, die von jeder einzelnen Nachricht und deren Antwort (falls vorhanden) gemeinsam benutzt wird. Entspricht einer Extricity-PublicProcessID oder einer DataInterchange (DI)-ApplicationID.
MessageType	Nein	Wird für Gateway- und Protokoll-/Trace-/Ausnahmenachrichten verwendet.

Tabelle 13. MQSeries Adapter Kernel-Header (Forts.)

Header-Name	**Rückgabe in Antworten?**	**Bedeutung bzw. Verwendung**
SourceLogicalID	Nein	Logische ID der Quellenanwendung. Entspricht reservierten Namen in DataInterchange (DI) , Partner Agreement Manager (PAM) und Business Flow Manager (BFM).
DestinationLogicalID	Nein	Logische ID der Zielanwendung. Für DI und PAM ist `none` der Standardwert, der jedoch überschrieben werden kann.
RespondToLogicalID	Ja	Logische ID, an die eine Antwortnachricht gesendet werden soll. Wird für DI in DestinationLogicalID und für PAM in SourceLogicalID kopiert.
CorrelationID	Nein	Reservierte Verwendung.
GroupStatus	Nein	Reservierte Verwendung.
ProcessingCategory	Nein	Entspricht einer PAM-PublicProcessID oder einer DI-CommandProcessID.
QosPolicy	Nein	Reservierte Verwendung.
DeliveryCategory	Nein	Entspricht einer DI-RequestorProfileID.
AckRequested	Nein	Legt fest, ob die Quellenanwendung eine Antwortnachricht anfordert oder nicht.
PublicationTopic	Nein	Reservierte Verwendung.
SessionID	Nein	Entspricht einer DI-BatchID.
EncryptionStatus	Nein	Legt die Art der Nachrichtenverschlüsselung und der Signatur fest.
TimeStampCreated	Nein	Zeit und Datum der Nachrichtenerstellung.

Tabelle 13. MQSeries Adapter Kernel-Header (Forts.)

Header-Name	**Rückgabe in Antworten?**	**Bedeutung bzw. Verwendung**
TimeStampExpired	Nein	Zeitpunkt (Zeit und Datum), an dem die Nachricht bedeutungslos wird. Der Wert -1 bedeutet, dass die Nachricht nie abläuft.
Size	Nein	Reservierte Verwendung.
BodyType	Nein	Gibt den jeweiligen Zweck der Nachricht an.
BodyCategory	Nein	Gibt den Anwendungstyp der Nachricht an.
BodySecondaryType	Nein	Reservierte Verwendung.
UserArea	Nein	Allgemeiner Bereich für Benutzerdaten.
RelatedSubjectID	Nein	Wird für Interprozess-korrelation verwendet.
ExternalID	Nein	ID des aktuellen Eigners (z. B. ein Benutzer oder Handelspartner) außerhalb der Anwendungs-umgebung.
InternalID	Nein	ID des aktuellen Eigners (z. B. ein Benutzer oder Handelspartner) innerhalb der Anwendungs-umgebung.
BodySignature	Nein	Reservierte Verwendung.
TransportCorrelationID	Ja	Reservierte Verwendung.

Nachrichtendeskriptor-Header von MQSeries

Der Inhalt der Felder wird von MQSeries festgelegt. MQSeries Adapter Offering reiht Nachrichten abhängig von den Nachrichtensteuerungswerten in Warteschlangen ein. Weitere Informationen finden Sie unter „Nachrichtensteuerungswerte" auf Seite 18.

Tabelle 14. MQSeries-Header

Abschnitt bzw. Feld	Bedeutung bzw. Belegung
Revision	Fest.
UniqueID	Jede Nachricht besitzt eine eindeutige ID.
TransactionID	Eine Nachricht und deren Antwort benutzen dieselbe Transaktions-ID.
MessageType	Reservierte Verwendung.
SourceLogicalID	Logische ID der Quellenanwendung.
DestinationLogicalID	Logische ID der Zielanwendung.
RespondToLogicalID	Eine logische ID, an die die Antwortnachricht gesendet wird.
CorrelationID	Reservierte Verwendung.
GroupStatus	Reservierte Verwendung.
ProcessingCategory	Reservierte Verwendung.
QosPolicy	Reservierte Verwendung.
DeliveryCategory	Reservierte Verwendung.
AckRequested	Gibt an, ob die Quellenanwendung eine Antwort anfordert oder nicht.
PublicationTopic	Reservierte Verwendung.
SessionID	Reservierte Verwendung.
EncryptionStatus	Reservierte Verwendung.
TimeStampCreated	Zeit und Datum der Nachrichtenerstellung.
TimeStampExpired	Zeitpunkt (Zeit und Datum), an dem die Nachricht bedeutungslos wird.
Size	Reservierte Verwendung.
BodyCategory	Gibt den Anwendungstyp der Nachricht an, z. B. OAG oder RosettaNet.
BodyType	Gibt den genauen Zweck der Nachricht an, z. B. Hinzufügen eines Verkaufsauftrags oder Synchronisieren des Inventars.
BodySecondaryType	Reserviert.
UserArea	Allgemeiner Bereich für Benutzerdaten.

Tabelle 14. MQSeries-Header (Forts.)

BodyData	Nachrichteninhalt.

MQSeries ohne MQSeries Integrator

Die Werte des Kernel-Headers und der Nachrichteninhalt werden in einem XML-Dokument abgelegt. Im Folgenden ein Beispiel für die DTD, in der das XML-Dokument beschrieben ist:

```
<!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) >
```

Header von MQSeries Integrator Version 1

Header von MQSeries Integrator Version 1, RFH1, bestehen aus folgenden Elementen:

1. Fester Abschnitt
2. Neon-Header
3. Datenbereich mit dem Kernel-Header und dem Nachrichteninhalt

Tabelle 15. Header von MQSeries Integrator Version 1 - RFH1

Abschnitt bzw. Feld	**Bedeutung bzw. Belegung**
Fester Abschnitt	Wird wie in MQSeries Integrator Version 1.1 angegeben verwendet.
Neon-Header	Entspricht dem Format des Neon-Headers.
OPT_APP_GRP	Der Wert von SourceLogicalId. Wird aus dem Kernel-Header übernommen.
OPT_MSG_TYPE	BodyCategory+BodyType. Werden aus dem Kernel-Header abgeleitet. Beispiel: Ist BodyCategory = OAG und BodyType = SyncItem, lautet der Wert OAG+SyncItem.
Datenbereich	Besteht aus den Werten des Kernel-Headers, gefolgt von den Daten des Nachrichteninhalts.
Kernel-Header	Der Kernel-Header hat folgendes Format: <EPICHEADER>*Header*</EPICHEADER>. Die Werte des Kernel-Headers sind in XML-Syntax angegeben. Es sind nur Attribute mit Werten angegeben. Die tatsächlichen Daten befinden sich nicht in separaten Zeilen. Beispiel für das Format eines Wertes: <MessageType>*Wert*</MessageType>.
MessageType	Reservierte Verwendung.
SourceLogicalID	Logische ID der Quellenanwendung.
DestinationLogicalID	Logische ID der Zielanwendung.
RespondToLogicalID	Logische ID, an die die Antwortnachricht gesendet wird.
TimeStampCreated	Zeit und Datum der Nachrichtenerstellung.
TimeStampExpired	Zeitpunkt (Zeit und Datum), an dem die Nachricht bedeutungslos wird.
TransactionID	Eine Nachricht und deren Antwort benutzen dieselbe Transaktions-ID.
UniqueID	Jede Nachricht besitzt eine eindeutige ID.

Tabelle 15. Header von MQSeries Integrator Version 1 - RFH1 (Forts.)

AckRequested	Dieser Wert gibt an, ob die Quellenanwendung eine Antwort anfordert.
ProcessingCategory	Reserviert.
BodyCategory	Gibt den Anwendungstyp der Nachricht an, z. B. OAG oder RosettaNet.
BodyType	Gibt den genauen Zweck der Nachricht an, z. B. Hinzufügen eines Verkaufsauftrags oder Synchronisieren des Inventars.
BodySecondaryType	Reserviert.
UserArea	Benutzerspezifische Anwendungsdaten für die Integration.
MsgHeaderVersion	Version des Kernel-Headers (reserviert).
CorrelationID	Benutzerspezifische Daten für die Integration.
GroupStatus	Benutzerspezifische Daten für die Integration.
QosPolicy	Reserviert.
DeliveryCategory	Reserviert.
PublicationTopic	Reserviert.
SessionID	Reserviert.
EncryptionStatus	Reserviert.
Nachrichteninhalt	Nachrichteninhalt.

Header von MQSeries Integrator Version 2

Header von MQSeries Integrator Version 2, RFH2, bestehen aus folgenden Elementen:

1. Fester Abschnitt
2. Ordner <mcd> - Deskriptor für Nachrichteninhalt
3. Ordner <usr> - Anwendungsspezifische (benutzerdefinierte) Eigenschaften
4. Datenbereich mit dem Kernel-Header und dem Nachrichteninhalt

Tabelle 16. Header von MQSeries Integrator Version 2 - RFH2

Abschnitt bzw. Feld	**Bedeutung bzw. Belegung**
Fester Abschnitt	Wird wie in MQSeries Integrator Version 2 angegeben verwendet.
<mcd>	XML, wenn es sich um eine Nachricht im XML-Format handelt. Gemäß den Regeln von MQSeries Integrator Version 2.
Set	Wird nicht vom Kernel verwendet.
Typ	Wird nicht vom Kernel verwendet.
Format	XML, wenn es sich um eine Nachricht im XML-Format handelt. Gemäß den Regeln von MQSeries Integrator Version 2.
Ordner <usr> - Anwendungs-spezifische (benutzerdefinierte) Eigenschaften	Enthält die Werte des Kernel-Headers.
Kernel-Header	Es sind nur Attribute mit Werten angegeben. Die tatsächlichen Daten befinden sich nicht in separaten Zeilen.
SourceLogicalID	Logische ID der Quellenanwendung.
DestinationLogicalID	Logische ID der Zielanwendung.
MessageType	Reservierte Verwendung.
RespondToLogicalID	Eine logische ID, an die die Antwortnachricht gesendet wird.
TimeStampCreated	Zeit und Datum der Nachrichtenerstellung.
TimeStampExpired	Zeitpunkt (Zeit und Datum), an dem die Nachricht bedeutungslos wird.
TransactionID	Eine Nachricht und deren Antwort benutzen dieselbe Transaktions-ID.
UniqueID	Jede Nachricht besitzt eine eindeutige ID.
ProcessingCategory	Reserviert.

Tabelle 16. Header von MQSeries Integrator Version 2 - RFH2 (Forts.)

BodyCategory	Gibt den Anwendungstyp der Nachricht an, z. B. OAG oder RosettaNet.
BodyType	Gibt den genauen Zweck der Nachricht an, z. B. Hinzufügen eines Verkaufsauftrags oder Synchronisieren des Inventars.
BodySecondaryType	Reserviert.
AckRequested	Dieser Wert gibt an, ob die Quellenanwendung eine Antwort anfordert.
UserArea	Benutzerspezifische Anwendungsdaten für die Integration.
MsgHeaderVersion	Version des Kernel-Headers (reserviert).
CorrelationID	Benutzerspezifische Daten für die Integration.
GroupStatus	Benutzerspezifische Daten für die Integration.
QosPolicy	Reserviert.
DeliveryCategory	Reserviert.
PublicationTopic	Reserviert.
SessionID	Reserviert.
EncryptionStatus	Reserviert.
Datenbereich	Nachrichteninhalt.

Anhang D. Beispiel der Konfigurationsdatei

Dieser Abschnitt enthält die zum Zeitpunkt dieser Veröffentlichung aktuelle Version der Datei `aqmconfig.xml`. Unter „Beispiel einer Mindestkonfiguration” auf Seite 138 finden Sie die zum Zeitpunkt dieser Veröffentlichung aktuelle Version der Datei `aqmconfig.minimum.xml`. Das Installationsverzeichnis `samples` des Kernels enthält die neuesten Versionen der Dateien `aqmconfig.xml` und `aqmconfig.minimum.xml`; die hier gezeigten Beispiele sind möglicherweise nicht mehr auf dem neuesten Stand.

Informationen zur Interpretation und zum Editieren der Konfigurationsdatei finden Sie unter „Die Konfigurationsdatei” auf Seite 75.

In diesem Beispiel der Konfigurationsdatei werden mehrere Anwendungs-IDs verwendet. Unter jeder Anwendungs-ID werden eine Reihe von Einträgen aufgeführt. Das Beispiel der Konfigurationsdatei enthält die folgenden Anwendungs-IDs:

- 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. -->
          <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
          <!-- Use the remote Q Manager for sending messages.  Remote queue -->
          <!-- definitions are not required.  true - use remote Q Manager, -->
          <!-- false - do not use remote Q Manager.  No entry defaults to false -->
          <epicuseremotequeuemanagertosend>false</epicuseremotequeuemanagertosend>
          <!-- MQSeries Client hostname for where the MQSeries server -->
          <!-- resides for TEST1.  Required if using MQSeries Client -->
          <!--
          <epicmqppqueuemgrhostname>localhost</epicmqppqueuemgrhostname>
   -->
          <!-- MQSeries Client port to use for where the MQSeries server -->
          <!-- resides for TEST1.  No entry defaults to MQSeries default -->
          <!--
          <epicmqppqueuemgrportnumber>1414</epicmqppqueuemgrportnumber>
   -->
          <!-- MQSeries Client channel name to use for the MQSeries server, required -->
          <!--
          <epicmqppqueuemgrchannelname>xyz</epicmqppqueuemgrchannelname>
   -->
          <!-- JMS example for TEST1.  Refers to the JMS Connection factory name. -->
               <!-- Requires the attribute describing the object plus the attributes value. -->
               <!-- For JMS the attribute is 'cn'. -->
          <!--
          <epicjmsconnectionfactoryname>cn=QCFTEST1</epicjmsconnectionfactoryname>
   -->
          <ePICBodyCategory epicbodycategory="DEFAULT">
             <ePICBodyType epicbodytype="DEFAULT">
<!-- Contains the Command selection criteria when processing -->
<!-- a message into an application.  Will be used by the -->
 <!-- AdapterDaemon - Command to invoke. -->
               <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
               </epiccommandclassname>
<!-- Represents the type of command the "epiccommandclassname" -->
<!-- represents.  MQAKEAB is the EAB style interface.  MQAKEJB -->
<!-- is an EJB Service Session Bean.  No entry defaults to MQAKEAB -->
               <epiccommandtype>MQAKEAB</epiccommandtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- method name to invoke.  No entry defaults to "execute". -->
               <epiccommandejbmethod>execute</epiccommandejbmethod>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- parameter type for the method specified by "epiccommandejbmethod". -->
<!-- This will be the same datatype returned by the -->
<!-- TerminalDataContainerMapper.  No entry defaults -->
<!-- to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
               <epiccommandejbmethodparmtype>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
               </epiccommandejbmethodparmtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- URL where the EJB specified in "epiccommandclassname" -->
<!-- has been deployed in the form "IIOP://hostname:900/". -->
<!-- No entry defaults to "IIOP:///". -->
               <epiccommandejburl>IIOP:///</epiccommandejburl>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Initial Context Factory used to 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>
```

Beispiel einer Mindestkonfiguration

Dieser Abschnitt enthält ein Beispiel für eine Mindestkonfiguration von MQSeries Adapter Kernel. Informationen zur Datei mit der Mindestkonfiguration finden Sie unter „Adapterinformationen zur Konfiguration hinzufügen" auf Seite 97.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.minimum.xml 1.00 00/11/07                                    -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Sample AQM Configuration showing a minimum configuration for the       -->
<!-- following conditions:                                                  -->
<!-- 1) Going from applicationid TEST1 to TEST2.  TEST1 is not receiving  -->
<!--    messages. -->
<!-- 2) TEST2 has no special application requirements.  -->
<!-- 3) TEST2 is using 1 worker.                         -->
<!-- 4) Using MQSeries with the default QManager installed on each machine.    -->
<!--    and using default format.                                           -->
<!-- 5) No specific body category and body type being used.                 -->
<!-- 6) Using default tracing to the console.   -->
<!--  -->
<!--  -->
<!--  -->
<!-- Copyright (c) 2000 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--     -->
   <Epic o="ePIC">
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1  -->
         <ePICApplication epicappid="TEST1">
     <!--  Tracing on/off.  If no entry defaults to false. -->
            <epictrace>false</epictrace>
     <!-- Trace levels - 512=TYPE_ERROR_EXC (Exceptions),-1=TYPE_ALL (All possible messages). -->
            <epictracelevel>0</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
     <!-- Default destinations to send messages to. -->
                     <epicdestids>TEST2</epicdestids>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
         <!-- The following is for Test Application ID: TEST2 -->
         <ePICApplication epicappid="TEST2">
            <epictrace>false</epictrace>
            <epictracelevel>512</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
      <!-- AdapterDaemon - Command to invoke. -->
                     <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                     </epiccommandclassname>
                     <epicreceivemode>MQ</epicreceivemode>
      <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
      <!-- -1 means never ending.  No entry defaults to 0. -->
      <!-- milliseconds.  Used when receiving messages. -->
                     <epicreceivetimeout>30000</epicreceivetimeout>
                     <epicreceivemqppqueue>TEST2AIQ</epicreceivemqppqueue>
                     <epicerrormqppqueue>TEST2AEQ</epicerrormqppqueue>
                     <epicreplymqppqueue>TEST2RPL</epicreplymqppqueue>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
      </ePICApplications>
   </Epic>
```

Anhang E. Beispiel der Setup-Datei

Dieser Anhang enthält ein Beispiel der Datei aqmsetup, in der mehrere Werte für die Erstkonfiguration des Kernels, einschließlich einer Reihe von Umgebungsvariablen, definiert sind. Weitere Informationen zu dieser Datei finden Sie unter „Die Setup-Datei" auf Seite 74. Die Datei aqmsetup befindet sich im Verzeichnis samples des Kernel-Installationsverzeichnisses 'root'.

```
#
# 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
```

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur diese Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. Anstelle der Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Veröffentlichung ist keine Lizenzierung dieser Patente verbunden. Lizenzanfragen sind schriftlich an IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France zu richten. Anfragen an obige Adresse müssen auf Englisch formuliert werden.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Hinweise in dieser Veröffentlichung auf Websites anderer Unternehmen als IBM dienen nur der Information; es kann kein Anspruch auf diese Websites abgeleitet werden. Die in diesen Websites aufgeführten Komponenten gehören nicht zum Lieferumfang dieses IBM Produkts; die Verwendung dieser Websites erfolgt auf Risiko des Kunden.

Werden an IBM Informationen gesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung (i) den Austausch von Informationen zwischen unabhängig erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

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

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der IBM Kundenvereinbarung, der IBM International Programming License Agreement oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen. Diese Daten stellen deshalb keine Leistungsgarantie dar.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Marken

Folgende Namen sind in gewissen Ländern Marken der IBM Corporation:

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

Lotus und LotusScript sind in gewissen Ländern Marken der Lotus Development Corporation.

Java sowie alle Java-basierten Marken und Logos sind in gewissen Ländern Marken oder eingetragene Marken der Sun Microsystems Inc.

Windows, Windows NT und das Logo von Windows sind in gewissen Ländern Marken der Microsoft Corporation.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken oder Dienstleistungsmarken anderer Unternehmen sein.

Glossar

Das Glossar enthält *Schlüsselbegriffe* der MQSeries Adapter Kernel-Dokumentation und erläutert deren Bedeutung.

Wenn ein bestimmter Begriff nur in einem einzigen Abschnitt der Dokumentation verwendet wird, ist er möglicherweise nicht in diesem Glossar enthalten. Sie finden ihn dann jedoch mit großer Wahrscheinlichkeit im „Index" auf Seite 151.

Das Glossar enthält keine Begriffe zu anderen IBM Produkten wie z. B. MQSeries.

Adapter. Die Ausgabe von MQSeries Adapter Builder. In der Regel erstellt der Benutzer einen speziellen Adapter für *jede einzelne Nachrichtenart*, die von einer oder an eine Anwendung gesendet wird. Das heißt, die Adapter selbst sind nicht Teil von MQSeries Adapter Offering. Ein Adapter besteht aus einem C- oder Java-Quellencode, der kompiliert und in einer gemeinsam benutzten Bibliothek abgelegt wird. Werden die Adapter und MQSeries Adapter Kernel gemeinsam ausgeführt, bilden sie zusammen die Laufzeitumgebung von MQSeries Adapter Offering. Abhängig davon, welche Funktionen der Benutzer für den Adapter bei dessen Erstellung mit Hilfe von MQSeries Adapter Builder definiert hat, kann dieser ganz unterschiedliche Aufgaben übernehmen, z. B. Steuerungsfluss, Datenfluss, sequenzielle Navigation, bedingte Verzweigung einschließlich Entscheidung und Iteration, Dateneingabe, Speicherung von Datenkontexten, Konvertierung von Datenelementen, Transaktionssteuerung, logische Operationen und benutzerdefinierter Code. Sie können einmal erstellte Adapter wiederverwenden.

Siehe „Nachrichtenart" auf Seite 147, „Quellenanwendung" auf Seite 148 und „Zielanwendung" auf Seite 149.

Adapterdämon. Ausführbare Software, die Teil des Kernels ist. Der Adapterdämon wird nur im Push-Übermittlungsmodell verwendet. Seine Aufgabe ist es, die Adaptermanager-Exemplare zu erstellen. Nachdem der Adapterdämon gestartet wurde, bleibt er aktiv. Für jede Zielanwendung kann es einen oder mehrere Adapterdämonen geben.

In einigen Fällen übernimmt der Adapterdämon die Rolle einer Zielanwendung. Er führt die erforderlichen Funktionen aus, z. B. Senden einer E-Mail-Nachricht mit Hilfe eines Zieladapters oder Schreiben eines Datensatzes in eine Datei.

Adaptermanager. Software, die Bestandteil des Kernels ist. Der Adaptermanager wird nur im Push-Übermittlungsmodell verwendet. Die Adaptermanager werden vom Adapterdämon erstellt und gestartet. Jeder Adaptermanager verwaltet einen internen Adapter. Der Adaptermanager übermittelt jede einzelne Nachricht an den zugehörigen Zieladapter.

Adaptermanager-Nachrichten-Bean. Eine Enterprise-Bean, die die Funktion eines Adaptermanagers ausführt, wenn WebSphere Application Server in der Zielverarbeitungsschicht des Kernels verwendet wird.

Anmeldeklasse. Eine Java-Klasse, die für jede Zielanwendung spezifisch ist und für die Übermittlung der Nachricht an die Zielanwendung verwendet werden kann. Die Anmeldeklasse wird nur benötigt, wenn sich der Zieladapter bei der Zielanwendung anmelden muss, bevor er die Nachricht übermittelt. Alle Anmeldeklassen werden von den Benutzern geschrieben. Der Adaptermanager erstellt ein Exemplar der Anmeldeklasse. Die Anmeldeklasse sucht in der Konfigurationsdatei nach den Werten, die der

Zieladapter für die Unterstützung der anwendungsspezifischen Schnittstelle zur Zielanwendung benötigt. Bei diesen Werten handelt es sich in der Regel um Anmeldeparameter. Auf diese Weise werden diese Werte dem Zieladapter zur Verfügung gestellt.

Der Kernel stellt standardmäßig eine leere Anmeldeklasse ohne Funktionalität bereit.

Anwendungsneutrales Format. Siehe „Integrierte Nachricht".

Anwendungsspezifische Schnittstelle. Eine Schnittstelle, die außerhalb von MQSeries Adapter Offering für eine der folgenden Aufgaben entwickelt wurde:

- Dem Quellenadapter ermöglichen, eine Nachricht von der Quellenanwendung abzurufen.
- Der Zielanwendung ermöglichen, eine Nachricht vom Zieladapter abzurufen.

aqmconfig.xml. Siehe „Konfigurationsdatei" auf Seite 147.

aqmsetup. Siehe „Setup-Datei" auf Seite 148.

BOD. Business Object Document. Die Darstellung eines Standardgeschäftsprozesses, der innerhalb einer Organisation oder zwischen Organisationen abläuft. Beispiele: Hinzufügen eines Einkaufsauftrags, Anzeigen der Produktverfügbarkeit oder Hinzufügen eines Verkaufsauftrags. BODs werden von der OAG als XML-Dokumente definiert. Siehe „OAG" auf Seite 148 und „XML" auf Seite 149.

MQSeries Adapter Offering kann BODs verwenden, um Nachrichteninhalte als integrierte Nachrichten zu definieren.

DTD. Document Type Definition (Dokumentartdefinition). In XML in der Regel eine Datei (oder mehrere kombiniert benutzte Dateien), die eine formale Definition einer bestimmten Dokumentart enthält. In der Datei sind die Namen festgelegt, die für Elemente in der DTD verwendet werden können, wo in der DTD Elemente stehen dürfen und wie die Elemente zusammenpassen. In MQSeries Adapter Offering können Sie mit Hilfe von DTDs den Nachrichteninhalt definieren. Siehe „XML" auf Seite 149 und „Integrierte Nachricht".

Empfangswarteschlange. In der Terminologie von MQSeries Adapter Offering eine Nachrichtenwarteschlange, die als Haupteingabewarteschlange für den Empfang von Nachrichten dient. Es kann mehrere Empfangswarteschlangen pro Zielanwendung geben, aber nur eine Empfangswarteschlange für jede Kombination aus Anwendungs-ID, Nachrichtenkategorie und Nachrichtenzweck.

Fehlerwarteschlange. In der Terminologie von MQSeries Adapter Offering eine Warteschlange für Nachrichten, die aus einer Warteschlange abgerufen wurden, aber nicht verarbeitet werden konnten.

ID der abhängigen Anwendung. Der Name der Anwendung, für die der Adaptermanager Services bereitstellt. Der Adaptermanager ruft, abhängig vom Namen des Adapterdämons, die ID der abhängigen Anwendung aus der Konfigurationsdatei ab.

Integrierte Nachricht. Eine Nachricht, die Anwendungsdaten zu Integrationszwecken in einem anwendungsneutralen Format enthält. Ein Beispiel dafür ist ein XML-Dokument, das vom Quellenadapter aus dem Format der Quellenanwendung in ein XML-Format konvertiert wurde.

Interner Adapter. Software, die für das Senden und Empfangen von Nachrichtenträgerobjekten verwendet wird.

Interner Adapter von MQSeries Adapter Kernel. Siehe interner Adapter.

Java-Serviceadapter. Ein in der Programmiersprache Java erstellter Adapter, der in einer JMS Listener-Umgebung die Funktionen eines Adapterdämons, Adaptermanagers und Zieladapters bereitstellt.

JMS Listener. Eine Komponente des Produkts WebSphere Business Integrator, die eine weitgehende Integration zwischen MQSeries Adapter Kernel und WebSphere Application Server Advanced Edition ermöglicht.

Kernel. Synonym für MQSeries Adapter Kernel.

Konfigurationsdatei. Dabei handelt es sich um die Datei `aqmconfig.xml`, die den Großteil der Konfigurationsdaten des Kernels enthält. Weitere Informationen finden Sie unter „Die Konfigurationsdatei" auf Seite 75.

Logische Antwort-ID. Die logische ID der Anwendung, an die Antworten gesendet werden, wenn eine Antwort angefordert wird. Standardmäßig ist dies die logische Quellen-ID in der Nachricht.

Logische Anwendungs-ID. Die ID der Anwendung, der ein Adapter (ein Quellen- oder ein Zieladapter) zugeordnet ist. Siehe „Logische Quellen-ID" und „Logische Ziel-ID".

Logische Quellen-ID. Ein Wert, der eine Quellenanwendung bezeichnet. Er wird zusammen mit anderen Nachrichtensteuerungswerten vom Kernel zum Weiterleiten und Marshaling von Nachrichten verwendet. Siehe „Nachrichtensteuerungswerte", „Logische Anwendungs-ID" und „Logische Ziel-ID".

Logischer Nachrichtenservice. Eine Komponente, die der interne Adapter verwendet, um Nachrichten für die Übertragung über den Transportmechanismus zu konvertieren.

Logische Ziel-ID. Ein Wert, der die Zielanwendung bezeichnet, die einem Zieladapter zugeordnet ist. Siehe „Logische Ziel-ID" und „Logische Anwendungs-ID".

Logische Ziel-ID. Ein Wert, der die Zielanwendung bezeichnet. Er wird zusammen mit anderen Nachrichtensteuerungswerten vom Kernel zum Weiterleiten und Marshaling von Nachrichten verwendet. Siehe „Nachrichtensteuerungswerte".

MQSeries Adapter Builder. Software, mit der ein Benutzer über eine grafische Benutzerschnittstelle für praktisch alle Anwendungen einen Adapter erstellen kann.

MQSeries Adapter Kernel. Eine Gruppe von APIs und verschiedenen ausführbaren Programmen, in C und Java, sowie verschiedenen Konfigurationsdateien. Der Kernel verwendet und unterstützt Adapter. Siehe „Adapter" auf Seite 145. Neben der direkten Unterstützung von Adaptern führt der Kernel zugehörige Funktionen aus; zu deren wichtigsten gehören das Routing von Nachrichten sowie Infrastruktur-Services für die Nachrichtenerstellung, Durchführung von Traces und Kommunikation mit MQSeries oder einer anderen Nachrichtenübermittlungssoftware.

MQSeries Adapter Offering. Eine Gruppe von Produkten für die Anwendungsintegration, bestehend aus MQSeries Adapter Builder und MQSeries Adapter Kernel.

Nachricht. In MQSeries, einschließlich MQSeries Adapter Offering, eine Datensammlung, die von einem Programm an ein anderes Programm gesendet wird.

Nachrichtenart. Eine Nachricht, die durch eine eindeutige Kombination aus Nachrichtenkategorie und Nachrichtenzweck definiert ist. Siehe „Nachrichtenkategorie" und „Nachrichtenzweck" auf Seite 148.

Nachrichtenkategorie. Daten in einer Nachricht, die den Anwendungstyp der Nachricht angeben, z. B. OAG oder RosettaNet. Diese Daten sind Teil der Nachrichtensteuerungswerte. Siehe „Nachrichtensteuerungswerte".

Die Nachrichtenkategorie hilft außerdem bei der Definition der Nachrichtenart. Siehe „Nachrichtenart".

Nachrichtensteuerungswerte. Ein Sammelbegriff für eine Gruppe von Werten in den Nachrichten (Inhalt und Header) und in der Konfigurationsdatei, die vom Kernel zur Steuerung des Marshalings und Routings von Nachrichten und teilweise von den Adaptern zur Steuerung der Ausführung ihrer Funktionen verwendet werden.

Nachrichtenträgerobjekt. Ein Container für Metadaten, den der Kernel für die Kapselung einer integrierten Nachricht und anderer Steuerdaten verwendet.

Nachrichtenzweck. Daten in einer Nachricht, die den besonderen Zweck der Nachricht angeben, z. B. Hinzufügen von Verkaufsaufträgen oder Synchronisieren des Inventars. Diese Daten sind Teil der Nachrichtensteuerungswerte. Siehe „Nachrichtensteuerungswerte" auf Seite 147.

Der Nachrichtenzweck hilft bei der Definition der Nachrichtenart. Siehe „Nachrichtenart" auf Seite 147.

OAG. Open Applications Group. Ein nicht kommerzielles Konsortium, in dem viele führende Anbieter von Anwendungsprogrammen für Interoperabilitäts-Lösungen vertreten sind. Die OAG definiert Business Object Documents (BODs).

Pull-Übermittlungsmodell. Siehe „Übermittlungsmodelle".

Push-Übermittlungsmodell. Siehe „Übermittlungsmodelle".

Quellenadapter. Ein Adapter, der die folgenden Tasks ausführt:

- Entgegennehmen oder Abrufen von strukturierten Daten aus einer Quellenanwendung (in der Regel über eine anwendungsspezifische Schnittstelle, die außerhalb des Adapters entwickelt wurde).
- Verarbeiten der strukturierten Daten, so wie es im Adapter definiert ist.
- Konvertieren der strukturierten Daten in ein integriertes Nachrichtenformat.
- Einreihen der Nachricht in eine Nachrichtenwarteschlange (mit Hilfe des Kernels) zur Übermittlung an einen oder mehrere Zieladapter und damit an die Zielanwendung(en).

Für jede Nachrichtenart gibt es einen Quellenadapter. Eine Quellenanwendung kann in der Regel mehrere Nachrichtenarten senden, sodass eine Quellenanwendung in den meisten Fällen von mehreren Quellenadaptern unterstützt wird. Siehe „Adapter" auf Seite 145.

Quellenanwendung. Ein Programm, mit dem über ein Computernetz Daten an ein anderes Programm (die so genannte Zielanwendung) gesendet werden, das sich normalerweise auf einem anderen Computer befindet.

Quellenverarbeitung des Kernels. Der Teil der Kernel-Funktionalität, der mit dem Empfang der Nachricht vom Quellenadapter beginnt und mit dem Einreihen der Nachricht in eine Nachrichtenwarteschlange endet.

Setup-Datei. Eine Datei, die verschiedene Anfangswerte des Kernels enthält. Der Standardname der Datei ist `aqmsetup`.

Trace-Client. Eine Kernel-Komponente, die Trace-Nachrichten erstellt.

Trace-Funktion. Eine Gruppe von Prozessen, die der Kernel zum Erstellen von Trace-Nachrichten verwendet. Siehe „Trace-Nachrichten".

Trace-Nachrichten. Nachrichten, die den Verarbeitungsstatus einer Nachricht an einem bestimmten Punkt im Kernel enthalten. Mit Hilfe von Trace-Nachrichten können Sie mögliche Ursachen für Probleme mit dem Kernel oder den Adaptern erkennen.

Siehe „Trace-Funktion".

Transaktion. Eine Gruppe von Operationen, die als untrennbare Arbeitseinheit ausgeführt werden müssen. Wenn alle Operationen einer Transaktion erfolgreich waren, wird die Transaktion festgeschrieben, d. h. alle Operationen werden ausgeführt. Wenn eine oder mehrere Operationen einer Transaktion fehlgeschlagen sind, wird die Transaktion zurückgesetzt, d. h. keine der Operationen wird ausgeführt.

Übermittlungsmodelle. Der Kernel verwendet zwei Modelle bei der Nachrichtenübermittlung an die Zielanwendung. Dabei handelt es sich um folgende Modelle:

Push Der Kernel ist für das Einleiten und Verwalten der Nachrichtenübermittlung an die Zielanwendung zuständig. Bei Ver-

wendung dieses Modells ist normalerweise keine Änderung der Zielanwendung für die Unterstützung von MQSeries Adapter Offering erforderlich.

Pull Die Zielanwendung ist für die Verwaltung der Nachrichtenübermittlung zuständig. Bei Verwendung dieses Modells ist eine Änderung der Zielanwendung für die Unterstützung von MQSeries Adapter Offering erforderlich. Die Zielanwendung muss die Schnittstelle zwischen dem Kernel und der Zielanwendung verwalten.

Übertragungsmodus. Der vom Kernel verwendete Modus zum Transportieren der Nachricht und Ausführen von Broker-Services.

Übertragungsnachricht. Alle den Transportmechanismus für die Übertragung betreffenden Informationen sowie das Nachrichtenträgerobjekt, das in das spezifische Nachrichtenübermittlungsformat für den verwendeten Transportmechanismus konvertiert wurde.

Warteschlange für Antwortnachrichten. Eine Nachrichtenwarteschlange für den Empfang von Antwortnachrichten. Sie wird in Verbindung mit der Methode `sendRequestResponse` des Kernels verwendet.

WebSphere Application Server Advanced Edition. Ein Softwareprodukt von IBM, das die Verwendung der Sun Microsystems Enterprise JavaBeans (EJB)-Spezifikation ermöglicht. WebSphere Application Server Advanced Edition enthält einen EJB-Server, unter dem Enterprise-Beans ausgeführt werden können. Enterprise-Beans beinhalten die Geschäftslogik und -daten, die von EJB-Clients verwendet und gemeinsam benutzt werden. Es gibt zwei Arten von Enterprise-Beans: Session-Beans, die temporäre, Client-spezifische Tasks und Objekte beinhalten, und Entity-Beans, die permanente Daten beinhalten. Eine Session-Bean, die so genannte Adaptermanager-Nachrichten-Bean, kann in der Zielverarbeitungsschicht von MQSeries Adapter Kernel verwendet werden.

XML. Extensible Markup Language. Ein W3C-Standard für die Darstellung von Daten.

Zieladapter. Ein Adapter, der die folgenden Tasks ausführt:

- Empfangen einer Nachricht (vom Kernel und von MQSeries oder einer anderen Nachrichtenübermittlungssoftware), die von einem Quellenadapter gesendet wurde.
- Verarbeiten der integrierten Nachricht, so wie es im Adapter definiert ist.
- Konvertieren der integrierten Nachricht in eine anwendungsspezifisch formatierte Nachricht, die von der Zielanwendung empfangen werden kann.
- Senden der Nachricht an die Zielanwendung über eine anwendungsspezifische Schnittstelle.
- Benachrichtigen des Adaptermanagers, sobald die Nachricht an die Zielanwendung gesendet wurde, damit dieser eine Bestätigung senden kann.

Wenn die Zielanwendung die integrierte Nachricht empfangen kann, ist gegebenenfalls kein Zieladapter erforderlich.

Für jede Nachrichtenart gibt es einen Zieladapter. Eine Zielanwendung kann in der Regel mehrere Nachrichtenarten entgegennehmen, sodass eine Zielanwendung in den meisten Fällen von mehreren Zieladaptern unterstützt wird. Siehe „Adapter" auf Seite 145.

Zielanwendung. Ein Programm, das über ein Computernetz Daten von einem anderen Programm (der so genannten Quellenanwendung) empfängt, das sich normalerweise auf einem anderen Computer befindet.

Zielverarbeitung des Kernels. Der Teil der Kernel-Funktionalität, der mit dem Abrufen der Nachricht aus einer Nachrichtenwarteschlange beginnt und mit dem Senden der Nachricht an den Zieladapter endet.

Index

N

O

P

Q

R

S

T

U

V

W

X

Z

Antwort

MQSeries Adapter Kernel for Multiplatforms
Einstieg
Version 1 Release 1

IBM Form GC12-2980-01

Anregungen zur Verbesserung und Ergänzung dieser Veröffentlichung nehmen wir gerne entgegen. Bitte informieren Sie uns über Fehler, ungenaue Darstellungen oder andere Mängel.

Zur Klärung technischer Fragen sowie zu Liefermöglichkeiten und Preisen wenden Sie sich bitte entweder an Ihre IBM Geschäftsstelle, Ihren IBM Geschäftspartner oder Ihren Händler.

Unsere Telefonauskunft ″HALLO IBM″ (Telefonnr.: 01803/31 32 33) steht Ihnen ebenfalls zur Klärung allgemeiner Fragen zur Verfügung.

Kommentare:

Danke für Ihre Bemühungen.

Sie können ihre Kommentare betr. dieser Veröffentlichung wie folgt senden:
- Als Brief an die Postanschrift auf der Rückseite dieses Formulars
- Als E-Mail an die folgende Adresse: ibmterm@de.ibm.com

Name	Adresse
Firma oder Organisation	
Rufnummer	E-Mail-Adresse

Antwort
GC12-2980-01

IBM Deutschland GmbH
SW TSC Germany

70548 Stuttgart

GC12-2980-01

GC12-2980-01