MQSeries® Adapter Kernel for Multiplatforms

# Guida rapida

*Versione 1 Rilascio 1*

GC13-2959-05

MQSeries® Adapter Kernel for Multiplatforms

# Guida rapida

*Versione 1 Rilascio 1*

GC13-2959-05

**Nota:** Prima di utilizzare questo prodotto e le relative informazioni, consultare la sezione “Informazioni particolari” a pagina 127.

**Sesta edizione (aprile 2001)**

Questa edizione si riferisce alla versione 1, rilascio 1, livello di modifica 1 di MQSeries Adapter Kernel for Multiplatforms (numero prodotto 5648-D75) e a tutti i rilasci e modifiche successivi, salvo diverse indicazioni in nuove edizioni.

Eventuali commenti possono essere inviati a:

Selfin S.p.A.
Translation Assurance
via F. Giordani, 7
80122 Napoli

Tali commenti e suggerimenti potranno essere utilizzati liberamente dall'IBM e dalla Selfin e diventeranno proprietà esclusiva delle stesse.

# Indice

# Figure

# Tabelle

# Benvenuti in MQSeries Adapter Kernel - Guida rapida

Questo documento fornisce una descrizione di MQSeries® Adapter Kernel, illustrandone le modalità di pianificazione, di installazione e di utilizzo.

Per poter utilizzare il kernel, completare la seguente procedura:

1. Leggere "Capitolo 1. Introduzione a MQSeries Adapter Offering" a pagina 1.
2. Eseguire le operazioni necessarie per l'installazione. Per dettagli, consultare "Preparazione all'installazione" a pagina 39.
3. Installare il kernel. Per dettagli, consultare "Installazione del kernel" a pagina 41.
4. Controllare che l'istallazione sia stata eseguita correttamente. Per dettagli, consultare "Verifica dell'installazione" a pagina 46.
5. Configurare il kernel. Per dettagli, consultare "Configurazione del kernel" a pagina 58.
6. Se si desidera, è possibile configurare software facoltativo da utilizzare con il kernel. Per dettagli, consultare "Configurazione di MQSeries e di MQSeries Integrator" a pagina 91.
7. Creare gli adattatori utilizzando Adapter Builder di MQSeries, quindi testarli e distribuirli. Per ulteriori informazioni, consultare la documentazione relativa a MQSeries Adapter Builder.
8. Avviare il kernel. Per dettagli, consultare "Avvio del kernel" a pagina 92.

Per utilizzare queste informazioni, è necessario conoscere i prodotti prerequisiti e facoltativi. Consultare "Capitolo 2. Pianificazione dell'installazione del kernel" a pagina 31 e "Materiale di riferimento" a pagina 101.

## A chi si rivolge questo documento

Questo documento si rivolge a coloro che devono pianificare, installare o utilizzare MQSeries Adapter Kernel.

## Informazioni correlate

Per ulteriori informazioni, consultare:

- Il file `readme.txt`. Questo file potrebbe contenere informazioni disponibili solo dopo il completamento di questo manuale. Prima di eseguire l'installazione, il file `readme.txt` si trova nella directory principale del

CD-ROM del prodotto. Dopo aver eseguito l'installazione, il file `readme.txt` si trova nella directory principale di installazione di MQSeries Adapter Kernel.

- Il manuale *Problem Determination Guide*, codice documento GC34-5897, in cui viene fornita una descrizione degli strumenti, compresa la traccia, necessari per la risoluzione di problemi specifici con MQSeries Adapter Kernel. *Problem Determination Guide* è disponibile in MQSeries Adapter Kernel Information Center, installato insieme al prodotto.
- La documentazione in linea relativa all'API (Application Programming Interface), fornita insieme a MQSeries Adapter Kernel Information Center. Queste informazioni vengono fornite solo come un ausilio per comprendere le funzioni del kernel e della diagnostica. Consultare "Capitolo 5. Utilizzo delle API di MQSeries Adapter Kernel" a pagina 99.
- La documentazione relativa a MQSeries Adapter Builder, che include testi e un sistema di aiuto.
- Visitare il sito Web della famiglia dei prodotti MQSeries all'indirizzo www.ibm.com/software/ts/mqseries/.

  Seguendo i collegamenti da questo sito Web è possibile:

  – Visualizzare le informazioni più recenti relative alla famiglia di prodotti MQSeries, compreso MQSeries Adapter Offering.
  – I manuali MQSeries in formato HTML e PDF potrebbero includere informazioni più recenti.
  – Scaricare i SupportPac di MQSeries.

# Convenzioni

La documentazione di MQSeries Adapter Kernel utilizza le seguenti convenzioni tipografiche.

*Tabella 1. Convenzioni utilizzate in questo testo*

| Convenzione | Significato |
|---|---|
| **Grassetto** | Indica i nomi dei comandi. In riferimento alle GUI (Graphical User Interface), indica i menu, le voci di menu, le etichette e i pulsanti. |
| `Monospazio` | Indica il testo da immettere da un prompt dei comandi e i valori da riportare senza alcuna modifica come i nomi dei file, i percorsi e gli elementi dei linguaggi di programmazione quali funzioni, classi e metodi. Inoltre, il monospazio viene utilizzato per indicare il testo visualizzato sullo schermo e per gli esempi di codice. |
| *Corsivo* | Indica i valori delle variabili che devono essere specificati dall'utente; ad esempio, il nome di un file per *nomefile*. Inoltre, il corsivo viene utilizzato anche per i titoli dei libri e per evidenziare un determinato concetto o parola. |
| `%` | Rappresenta un prompt della shell di comandi UNIX per un comando che non richiede privilegi root. |
| `#` | Rappresenta un prompt della shell di comandi UNIX per un comando che richiede privilegi root. |
| `C:\>` | Rappresenta i prompt dei comandi del sistema Windows®. |
| **>** | Se utilizzato per descrivere un menu, indica una serie di selezioni di menu. Ad esempio, "Fare clic su **File > Nuovo**" significa "Dal menu **File**, fare clic sul comando **Nuovo**." |
| Immissione dei comandi | Laddove indicato di "immettere" o "inviare" un comando, digitare il comando, quindi premere Invio. Ad esempio, l'istruzione "Immettere il comando **ls**", significa digitare **ls** da un prompt dei comandi, quindi premere Invio. |
| [ ] | Racchiudere gli elementi facoltativi nelle descrizioni della sintassi. |
| { } | Racchiudere gli elenchi dai quali è necessario scegliere un elemento nelle descrizioni della sintassi. |
| \| | Separa le voci di un elenco di scelte racchiuso tra parentesi nelle descrizioni della sintassi ({ }). |
| ... | All'interno delle descrizioni della sintassi, indicano che è possibile ripetere l'elemento precedente una o più volte. Negli esempi, indicano che alcune informazioni sono state omesse per economia. |

**Nota:** Il termine Epic compare in alcuni valori e nomi del software del kernel, oltre che all'interno di questo testo. Per quanto concerne MQSeries Adapter Offering, questo termine non ha alcun significato particolare.

# Riepilogo delle modifiche

La sesta edizione (l'edizione corrente) include le seguenti modifiche e aggiunte rispetto alla quinta edizione:

- Aggiornamenti alle problematiche relative al flusso run-time per riflettere le molteplici modifiche. Consultare "Flusso di runtime" a pagina 14,
- Informazioni sull'utilizzo di MQSeries Adapter Kernel con WebSphere® Business Integrator. Per dettagli, consultare "Utilizzo di MQSeries Adapter Kernel con WebSphere Business Integrator e WebSphere Application Server" a pagina 29.
- Informazioni sul livello di supporto delle lingue nazionali fornito con i diversi tipi di adattatori. Per dettagli, consultare "Supporto lingue nazionali" a pagina 30.
- Ulteriori informazioni sulle istruzioni di installazione. Consultare "Installazione del kernel" a pagina 41,
- Informazioni sull'installazione non presidiata. Per dettagli, consultare "Utilizzo dell'installazione non presidiata" a pagina 51.
- Panoramica sulla configurazione del kernel. Per dettagli, consultare "Panoramica sulla configurazione" a pagina 58.
- Informazioni sui nuovi valori di intestazione. Per dettagli, consultare "Intestazione del descrittore messaggi di MQSeries Adapter Kernel" a pagina 111.

La quinta edizione includeva le seguenti modifiche e aggiunte rispetto alla quarta edizione:

- Informazioni sulle modalità di utilizzo del kernel sulle piattaforme Windows® 2000, OS/400®, HP-UX e Solaris. Il supporto per queste piattaforme era nuovo per MQSeries Adapter Kernel versione 1.1. In passato, il kernel era disponibile solo sulle piattaforme Windows NT® e AIX®.
- Aggiornamenti di tutte le istruzioni di installazione per riflettere MQSeries Adapter Kernel versione 1.1.
- Informazioni sulle modalità di utilizzo del file `aqmconfig.xml` per configurare MQSeries Adapter Kernel. In passato, la configurazione del kernel veniva eseguita utilizzando il file `aqmconfig.properties`. Per dettagli, consultare "File di configurazione" a pagina 65.
- Informazioni sulle nuove modalità di comunicazione MQ e JMS (Java Message Service). Per dettagli, consultare "Appendice A. Modalità di comunicazione" a pagina 103.

- Nuova ubicazione delle informazioni relative alle tracce: sono ora disponibili nel nuovo documento *Problem Determination Guide*. Consultare *Problem Determination Guide* per informazioni dettagliate.

# Capitolo 1. Introduzione a MQSeries Adapter Offering

IBM MQSeries Adapter Kernel fa parte di una serie di prodotti di integrazione di applicazioni chiamati IBM MQSeries Adapter Offering. MQSeries Adapter Offering gestisce la messaggistica MQSeries e altri servizi di messaggistica consentendo di ridurre i rischi, la complessità e i costi della gestione di integrazione point-to-point dei processi commerciali.

Nell'integrazione *point-to-point* ciascuna applicazione comunica individualmente con le altre applicazioni. Ogni interfaccia si differenzia dalle altre e vi sono molte interfacce differenti. Di solito, la modifica di un'applicazione richiede anche la modifica di molte interfacce. Man mano che il numero delle applicazioni incrementa, aumentano rapidamente anche i costi dell'integrazione punto-a-punto. L'integrazione di una nuova applicazione richiede solitamente più lavoro rispetto all'integrazione della precedente.

Con MQSeries Adapter Offering, è possibile passare dall'utilizzo dell'integrazione point-to-point all'utilizzo dell'integrazione *one-to-any*. I vantaggi dell'integrazione one-to-any sono molteplici, fra cui:

- Tutte le applicazioni possono utilizzare un'interfaccia comune.
- Dall'*applicazione di origine*, i dati vengono *instradati* sotto forma di *messaggio* a una o più *applicazioni di destinazione*.
- In genere, una modifica a un'applicazione interessa unicamente la sua interfaccia.
- L'utilizzo di un'interfaccia comune, che sia indipendente dall'applicazione utilizzata—ad esempio, uno standard industriale come XML (Extensible Markup Language)— può essere più conveniente da un punto di vista dei costi. E' possibile supportare più applicazioni con costi minori.
- Man mano che il numero delle applicazioni aumenta, un'integrazione one-to-any si rivela sempre più conveniente. In genere, l'aggiunta di una nuova applicazione non richiede significative modifiche alle interfacce delle altre applicazioni.
- E' possibile automatizzare il processo di integrazione e basarlo sull'utilizzo di modelli.

E' possibile utilizzare MQSeries Adapter senza apportare modifiche alle applicazioni oppure ai processi aziendali. Di solito, il processo di integrazione viene interamente eseguito in MQSeries Adapter Offering e quindi non occorre scrivere codice personalizzato.

In MQSeries Adapter Offering, l'interfaccia per o da un'applicazione viene fornita da un *adattatore*. Tutte le applicazioni richiedono almeno un adattatore per fornire l'interfaccia tra l'ambiente dell'applicazione e l'ambiente di messaggistica. Ciascun adattatore è specifico per un'applicazione e per un tipo di messaggio.

Se si desidera, è anche possibile utilizzare MQSeries Adapter Kernel con MQSeries Integrator per eseguire il brokering e conversioni dei messaggi. Inoltre, è possibile includere in MQSeries Adapter Offering ulteriori servizi IBM o anche forniti da terzi.

Gli adattatori possono essere utilizzati per effettuare la seguenti operazioni:

- Aggiungere un ordine di vendita.
- Sincronizzare un record di un cliente.
- Sincronizzare un record di inventario.
- Sincronizzare un articolo.
- Sincronizzare un ordine di vendita.

## Tempo di creazione e tempo di esecuzione

MQSeries Adapter Offering è composto da due componenti principali: Adapter Builder (chiamato anche builder) e Adapter Kernel (chiamato anche kernel). In questa sezione, vengono descritti sia i componenti che gli adattatori creati ed eseguiti mediante Adapter Offering.

**adattatore**

Software che fornisce un'interfaccia per e da un'applicazione. Gli adattatori vengono creati utilizzando MQSeries Adapter Builder. Solitamente, ciascun adattatore è creato per uno specifico tipo di messaggio inviato o ricevuto da un'applicazione. Gli adattatori non fanno parte di MQSeries Adapter Offering.

Un adattatore è costituito da un codice di origine C o Java™ che compila la libreria condivisa. Quando si eseguono insieme gli adattatori e MQSeries Adapter Kernel, eseguono funzionalità di runtime di MQSeries Adapter Offering.

In base alla struttura in MQSeries Adapter Builder, l'adattatore può contenere un'ampia gamma di funzioni quali flusso di controllo, flusso dati, navigazione sequenziale, branching condizionale, incluse decisione e interazione, digitazione dati, storage di contesto dei dati, trasformazione degli elementi di dati, controllo transazionale, operazioni logiche e personalizzazione del codice.

Gli adattatori possono essere riutilizzati.

Esistono due tipi principali di adattatori:

- Adattatori origine, per applicazioni per l'invio di dati.
- Adattatori destinazione, per applicazioni che ricevono dati.

Normalmente, l'invio di un tipo di messaggi tra due applicazioni richiede l'uso di un adattatore di origine e di un adattatore di destinazione. Se la seconda applicazione deve inviare un tipo di messaggio alla prima applicazione, è necessario utilizzare un altro adattatore di origine e un altro adattatore di destinazione. Per inviare un tipo di messaggio dalla prima applicazione alla seconda applicazione e poi inviare un altro tipo di messaggio dalla seconda applicazione alla prima, solitamente vengono impiegati quattro adattatori.

È necessario utilizzare un adattatore differente per ciascun tipo di messaggio.

Un terzo tipo di adattatore, l'adattatore bean di sessione di servizio Java, viene utilizzato quando bean enterprise e IBM WebSphere Application Server sono utilizzati sulla lato di destinazione del kernel. L'implementazione di WebSphere Application Server delle specifiche Sun Microsystems Enterprise JavaBeans (EJB) abilita l'utilizzo degli adattatori dei bean di sessione di servizio Java e di altri bean enterprise. Consultare la documentazione "Utilizzo di MQSeries Adapter Kernel con WebSphere Business Integrator e WebSphere Application Server" a pagina 29 e MQSeries Adapter Builder per ulteriori informazioni.

**MQSeries Adapter Builder**
Si tratta di un'interfaccia utente grafica che consente di creare un adattatore virtualmente idoneo per qualunque applicazione. L'interfaccia utente è simile all'interfaccia utente di MQSeries Integrator. Per ulteriori informazioni, vedere MQSeries Adapter Builder Information Center.

**MQSeries Adapter Kernel**
Una serie di API (Application Programming Interfaces), molti programmi eseguibili in C e Java e numerosi file di configurazione. Il kernel consente la distribuzione e l'esecuzione degli adattatori. Oltre a supportare direttamente gli adattatori, il kernel ne esegue le funzioni correlate, incluso l'instradamento semplice dei messaggi. Fornisce anche servizi di infrastruttura, quali creazione dei messaggi, controllo transazionale, traccia e interfaccia con MQSeries o altri software di messaggistica.

Il kernel viene installato su ogni singolo computer sul quale viene eseguito un adattatore di origine o di destinazione.

Con MQSeries Adapter Offering, i processi aziendali e le singole applicazioni possono restare isolate dalle specifiche del middleware, dai dettagli dei messaggi e da altre applicazioni. Un'interfaccia comune per la messaggistica consente di aggiungere nuove applicazioni senza modificare le applicazioni e i processi aziendali esistenti.

MQSeries Adapter Kernel può essere distribuito su due livelli. Un primo livello è il lato di origine di runtime; l'altro è il lato di destinazione di runtime. Questo tipo di distribuzione consente un livello delle prestazioni efficace e una riduzione del sovraccarico gestionale. Inoltre, non è necessario un terzo livello per l'instradamento e la consegna. Tuttavia, se si desidera, è possibile aggiungere MQSeries Integrator per eseguire eseguire operazioni di brokering, come attività di instradamento complesso, trasformazione e mediazione dei dati.

Se non specificato diversamente, la restante parte di questo documento si riferisce unicamente a MQSeries Adapter Kernel. Per ulteriore informazioni su MQSeries Adapter Builder, consultare Information Center del prodotto.

## Informazioni sul kernel

I componenti di runtime, ossia, il kernel e gli adattatori che vengono creati, consentono di:

1. Trasferire i dati da un'applicazione di origine a quella di destinazione.
2. Convertire i dati dell'applicazione di origine in un messaggio, normalmente per un formato indipendente dall'applicazione, che viene instradato attraverso il kernel, utilizzando MQSeries o altri software per la messaggistica.
3. Instradare il messaggio verso l'applicazione di destinazione.
4. Stabilire la modalità di trasmissione dei dati all'applicazione di destinazione.
5. Convertire i dati dal formato del messaggio instradato mediante il kernel, nel formato dell'applicazione di destinazione.

In questa sezione, viene fornita una descrizione approfondita delle funzionalità del kernel. Una descrizione dettagliata di queste funzionalità è contenuta anche in "Flusso di runtime" a pagina 14.

Esistono due lati del kernel:

- Il *lato origine*, che inizia quando il messaggio viene ricevuto dall'applicazione di origine e termina quando il messaggio viene collocato nella coda dei messaggi.
- Il *lato destinazione*, che inizia quando il messaggio viene richiamato dalla coda dei messaggi e termina all'invio del messaggio alla destinazione.

Di solito, ciascun lato si trova su un computer diverso; tuttavia, possono entrambi risiedere sul medesimo computer.

Consultare Figura 1 a pagina 6, in cui viene illustrata la sequenza descritta di seguito.

**Lato di origine del kernel**

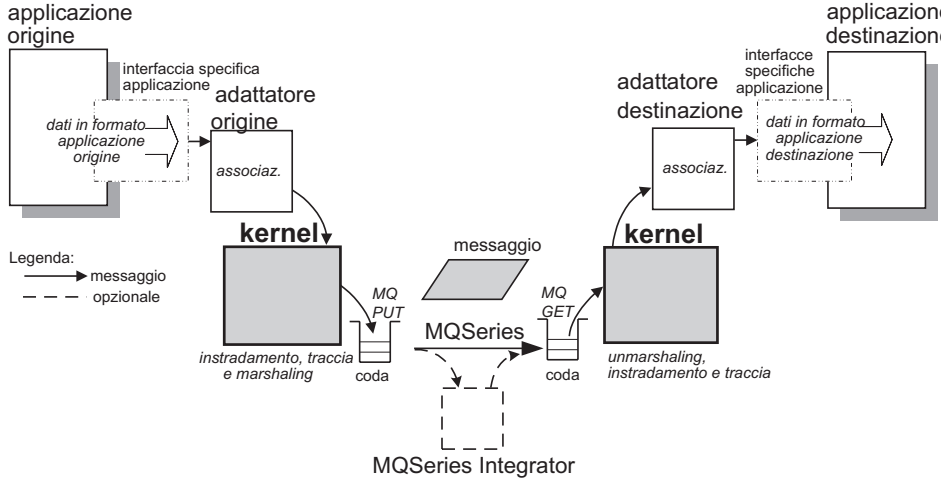
1. Sul lato di origine del kernel, l'applicazione di origine invia i dati nel *formato dell'applicazione di origine*, utilizzando un'*interfaccia specifica dell'applicazione*, a un adattatore di origine creato in MQSeries Adapter Builder. È necessario utilizzare un adattatore di origine diverso per ogni tipo di messaggio, ad esempio, per "aggiungere un ordine di vendita" o per "sincronizzare un record cliente".

   L'interfaccia specifica dell'applicazione deve essere sviluppata al di fuori di MQSeries Adapter Offering. La natura dell'interfaccia specifica dell'applicazione dipende dalle caratteristiche dell'applicazione di origine o di destinazione. Gli esempi includono chiamate API e uscite utente, letture e scritture di file, trigger del database e code messaggi.

   L'adattatore di origine viene eseguito nel processo dell'applicazione di origine. I daemon o server contenenti l'adattatore di origine devono essere eseguiti affinché l'adattatore di origine funzioni.
2. Il funzionamento dell'adattatore di origine dipende dal modo in cui è stato creato. Una funzione tipica è la trasformazione degli elementi di dati, cioè, l'associazione degli elementi dal formato dell'applicazione di origine in un *formato integration-messaging* per i dati di testo. I dati di testo e gli ulteriori metadati rappresentanti i valori di controllo sono collocati in un *oggetto message-holder* del kernel.
3. Quando l'adattatore di origine inoltra l'oggetto message-holder al kernel utilizzando l'*adattatore nativo*, i valori di controllo nell'oggetto message-holder (*valori di controllo messaggio*) sono utilizzati dal kernel per controllare il marshal dell'oggetto message-holder in un formato di messaggio di comunicazione e l'instradamento dei messaggi di comunicazione.

   Se il messaggio non contiene alcuni valori di controllo messaggio, il kernel può utilizzare i valori predefiniti o i valori di controllo messaggio richiamati dal file di configurazione. Per le definizioni dei valori di controllo messaggio, consultare "Valori di controllo messaggi" a pagina 16.
4. Le funzioni del kernel comprendono il *marshaling dei messaggi*, l'*instradamento* semplice e, se si desidera, la *creazione di traccia*. Consultare "Messaggio e formato del messaggio" a pagina 12, "Instradamento e consegna" a pagina 13 e "Funzione di traccia" a pagina 28.

*Figura 1. Panoramica su MQSeries Adapter Offering.*




**Consegna dal lato di origine al lato di destinazione del kernel**

5. Il kernel, utilizzando l'adattatore nativo, colloca i messaggi nella coda di messaggi appropriata.

   Sul lato di origine del kernel, vengono utilizzati i metodi riportati di seguito:

   - `sendMsg`, che invia il messaggio e viene restituito immediatamente. Inoltre, il metodo `sendMsg` può essere utilizzato anche con i metodi `begin`, `commit` e `rollback` per inviare messaggi *in maniera transazionale*; ossia, è possibile inviare i messaggi solo se le altre operazioni terminano con esito positivo. Per ulteriori informazioni, consultare "Funzioni transazionali" a pagina 28.
   - `sendRequestResponse`, che invia il messaggio e attende una risposta. Il metodo `sendRequestResponse` non può essere eseguito come transazione.

     Un terzo metodo, `sendResponse`, viene utilizzato sul lato di destinazione del kernel quando il mittente richiede una risposta.

   I messaggi vengono trasportati da MQSeries oppure da un altro software per la messaggistica. Consultare "Funzione di MQSeries o del software per la messaggistica utilizzato" a pagina 8. Si noti che il software di messaggistica deve essere già configurato per il supporto di MQSeries Adapter Offering.

   Se MQSeries Integrator è stato configurato nel kernel come destinazione, MQSeries Integrator può facoltativamente eseguire funzioni di brokering. Consultare "Funzione di MQSeries Integrator" a pagina 8. Se la

destinazione finale, una coda di messaggi, è stata configurata nelle regole o nei flussi di messaggi di MQSeries Integrator, MQSeries Integrator invia il messaggio alla coda di messaggi.

Il messaggio viene inserito nella coda appropriata.

**Lato di destinazione del kernel**

6. Sul lato di destinazione del kernel, sono disponibili due *modelli di consegna* per l'interfaccia tra l'applicazione di runtime e quella di destinazione.
   - Il modello più comune è quello *push*, in cui il kernel è responsabile dell'inizializzazione e della gestione della consegna del messaggio all'applicazione di destinazione. Di solito, per utilizzare questo modello non è necessario modificare l'applicazione di destinazione per consentire il supporto di MQSeries Adapter Offering.
   - Nel modello *pull*, l'applicazione di destinazione è responsabile della gestione della ricezione del messaggio. Questo modello richiede modifiche all'applicazione di destinazione per supportare MQSeries Adapter Offering. L'applicazione di destinazione deve gestire l'interfaccia del kernel nell'applicazione di destinazione.

   Si noti che nel modello push, sul lato di destinazione, i processi del kernel devono essere avviati dall'utente prima di ottenere o consegnare il messaggio. Consultare "Avvio del kernel" a pagina 92.

   Nel modello push, il kernel richiama il messaggio al di fuori della coda. Se la funzione traccia è abilitata, la creazione della traccia viene eseguita. Continua a instradare il messaggio selezionando l'adattatore di destinazione appropriato. In genere, è necessario utilizzare un adattatore differente per ciascun tipo di messaggio.
7. Il kernel consegna il messaggio all'adattatore di destinazione appropriato. L'adattatore di destinazione esegue le funzioni di cui è stato fornito al momento della creazione. Una funzione tipica è l'associazione degli elementi dal formato integration-messaging degli elementi al *formato dell'applicazione di destinazione*.

   E' possibile eseguire l'host degli adattatori di destinazione da un daemon di adattatore MQSeries Adapter Kernel o da WebSphere Application Server. Consultare "Utilizzo di MQSeries Adapter Kernel con WebSphere Business Integrator e WebSphere Application Server" a pagina 29 per ulteriori informazioni.
8. L'adattatore di destinazione invia i dati all'applicazione di destinazione nel formato dell'applicazione di destinazione utilizzando l'interfaccia specifica dell'applicazione sviluppata esternamente a MQSeries Adapter Offering.

9. Una volta che l'adattatore ha consegnato il messaggio, esso viene eliminato dalla coda dei messaggi. Quindi, il messaggio viene eliminato dalla coda.
10. Se l'adattatore di origine è stato impostato su un valore di controllo messaggio per richiedere una conferma, il kernel invia la conferma di consegna del messaggio o un'emissione dell'adattatore di destinazione all'adattatore origine utilizzando il metodo `sendResponse`.
11. In caso di errore, il kernel inserisce il messaggio originale nella coda di errore. Se il kernel non può inserire il messaggio originale nella coda di errore, non ha luogo alcuna conferma.

### Funzione di MQSeries o del software per la messaggistica utilizzato

I messaggi delle comunicazioni di MQSeries Adapter Offering vengono trasportati attraverso le code messaggi. Le code messaggi vengono fornite da software di messaggistica quali MQSeries o JMS (Java Message Service). I messaggi trasportati da MQSeries Adapter Offering utilizzano i seguenti tipi di coda:

- *Coda di ricezione*, nella terminologia di MQSeries Adapter Offering. Queste code vengono utilizzate come code di immissione principali per ricevere messaggi. Un'applicazione di destinazione può disporre di più code di ricezione.
- *Code di errore*, nella terminologia di MQSeries Adapter Offering. Queste code vengono utilizzate quando si riceve un messaggio da una coda di ricezione che non è possibile elaborare.
- Come opzione, sono disponibili anche *code di risposta*. Queste code vengono utilizzate con il metodo `sendRequestResponse`.

MQSeries Adapter Offering utilizza determinate funzioni di MQSeries, come i seguenti tipi di messaggi:

- Datagrammi, utilizzati dal metodo `sendMsg`.
- Richiesta, utilizzata dal metodo `sendRequestResponse`.
- Risposta, utilizzata dai metodi `sendRequestResponse` e `sendResponse`.

Se si desidera, MQSeries può fungere da interfaccia specifica dell'applicazione.

Consultare "Appendice B. Configurazioni convalidate" a pagina 109 per un elenco di configurazioni valide di MQSeries e MQSeries Adapter Offering. Per un elenco delle versioni supportate di MQSeries e di altro software, consultare "Software" a pagina 32.

### Funzione di MQSeries Integrator

MQSeries Integrator può essere facoltativamente distribuito con MQSeries Adapter Kernel. Può essere utilizzato per soddisfare numerosi requisiti per il brokering:

- Instradamento complesso, ossia, l'instradamento basato sul contenuto dell'intestazione o sul testo del messaggio. L'instradamento può variare in maniera dinamica contemporaneamente alle modifiche apportate al testo del messaggio. Per informazioni sull'instradamento complesso e semplice, consultare "Instradamento e consegna" a pagina 13.
- Trasformazione dei dati, ossia, il passaggio a un tipo di documento differente.
- Mediazione dei dati, ossia, modifica del contenuto del testo del messaggio. Ad esempio, se l'applicazione di origine fornisce il valore `each` in un campo per il quale l'applicazione di destinazione prevede il valore `ea`, la mediazione dei dati sostituisce il valore fornito con quello previsto.

E' possibile utilizzare MQSeries Integrator per eseguire la maggior delle operazioni di instradamento; è anche possibile utilizzare un numero ridotto di funzioni di instradamento di MQSeries Adapter Kernel.

Per un elenco di configurazioni valide di MQSeries Integrator e di MQSeries Adapter Offering, consultare "Appendice B. Configurazioni convalidate" a pagina 109. Per un elenco delle versioni supportate di MQSeries Integrator e di altro software, consultare "Software" a pagina 32.

## Descrizione del funzionamento del kernel

In questa sezione, vengono illustrati i seguenti argomenti:

- "Componenti del runtime del kernel"
- "Messaggio e formato del messaggio" a pagina 12
- "Instradamento e consegna" a pagina 13
- "Flusso di runtime" a pagina 14

## Componenti del runtime del kernel

Quando vengono eseguiti insieme gli adattatori creati, il codice personalizzato definito e MQSeries Adapter Kernel, questi elementi forniscono le funzionalità di MQSeries Adapter Offering.

I principali componenti del runtime del kernel sono i seguenti:

**adattatore di origine**
Software creato per un'applicazione specifica (utilizzando MQSeries Adapter Builder) per convertire i dati dell'applicazione in un formato per la messaggistica di integrazione (dati del testo). Di solito gli adattatori vengono eseguiti sullo stesso computer dell'applicazione di origine, nel processo dell'applicazione oppure come processo separato. I dati di origine includono file, strutture C e oggetti Java. Un esempio

di formato di messaggi di integrazione è XML, che normalmente segue uno standard industriale come OAG o RosettaNet.

**contenitore messaggi**
Un contenitore per metadati utilizzato dal kernel per racchiudere il messaggio e altri dati di controllo utilizzati dal kernel. Esempi di metadati includono gli identificativi di applicazione (gli identificativi logici) dell'applicazione di origine e di destinazione, la categoria del messaggio (ad esempio, OAG), il tipo di messaggio (ad esempio, "Ordine di acquisto") e i messaggi delle comunicazioni (dati di testo) da inviare o ricevere.

**adattatore nativo**
Software utilizzato per l'invio e la ricezione degli oggetti contenenti i messaggi. All'invio dei messaggi, l'adattatore nativo fornisce l'instradamento dei dati semplice e il supporto di uno o più meccanismi di trasporto delle comunicazioni. L'instradamento dei dati semplice è basato sui metadati nell'oggetto holder del messaggio, quali la categoria ed il tipo del messaggio. I messaggi possono essere inviati in maniera asincrona o sincrona. Se il meccanismo di trasporto delle comunicazioni utilizzato supporta la messaggistica transazionale, i messaggi possono essere inviati con il controllo transazionale a fase singola. Il supporto transazionale è limitato alle funzioni del meccanismo di trasporto impiegato. L'oggetto message-holder è organizzato in formato di messaggio di comunicazione utilizzato dal meccanismo di trasporto. Alla ricezione di un messaggio di comunicazione, l'adattatore nativo riorganizza il messaggio in oggetto message-holder.

**daemon dell'adattatore**
Un processo che crea un'istanza dei worker dell'adattatore. Una volta avviato, il daemon resta attivo. Per ogni applicazione di destinazione, esiste un solo daemon dell'adattatore per ciascuna coda di ricezione dell'applicazione.

**worker dell'adattatore**
Un processo che consegna ogni messaggio all'adattatore di destinazione appropriato. Ciascun worker gestisce un adattatore nativo. Il daemon dell'adattatore crea e avvia i worker.

L'utilizzo di più worker abilita la *consegna di messaggi a più thread* agli adattatori di destinazione. Ogni worker, insieme al rispettivo adattatore nativo, può gestire un thread. Nel caso di un solo worker, la consegna dei messaggi all'adattatore di destinazione e, quindi all'applicazione di destinazione, è a thread singolo.

Oltre alla gestione di un thread nativo, il worker effettua anche le seguenti attività:

- Crea un'istanza del client di traccia, se la traccia è abilitata.

- Crea un'istanza della classe di collegamento per ogni applicazione di destinazione.
- Seleziona l'adattatore di origine in base alla categoria e al tipo di testo del messaggio.
- Invia il messaggio all'adattatore di origine selezionato.
- Se non può eseguire una conferma, esegue un annullamento, imposta un flag per tutti gli altri worker nel daemon dell'adattatore, si arresta e interrompe anche l'esecuzione dell'adattatore nativo. Ciò indica che il messaggio presenta un problema. L'arresto di tutti i worker impedisce ad altri worker di elaborare di nuovo lo stesso messaggio relativo al problema con lo stesso risultato.
- Quando riconosce il flag impostato da un altro worker da arrestare, si arresta e interrompe anche l'esecuzione dell'adattatore nativo.

**adattatore di destinazione**
Software creato per un'applicazione specifica (solitamente utilizzando MQSeries Adapter Builder) per convertire i dati dal formato del messaggio di integrazione (dati di testo) ai tipi di dati richiesti dall'applicazione di destinazione. L'adattatore di destinazione richiama le API necessarie sull'applicazione di destinazione per consegnare il messaggio. Gli adattatori di destinazione vengono eseguiti stesso computer dell'applicazione o dell'applicazione client.

**adattatore bean di sessione del servizio Java**
Un tipo di adattatore EJB di linguaggio Java di cui viene eseguito l'host in un server EJB, quale WebSphere Application Server.

**componente di configurazione**
Dati utilizzati per risolvere gli identificativi logici in oggetti come i nomi delle code. I dati di configurazione possono essere specificati in un file o in una struttura LDAP del prodotto WebSphere Business Integrator. I dati controllano i seguenti aspetti della configurazione del kernel:

- Marshaling e instradamento dei messaggi
- Verifica dell'installazione
- Modalità di comunicazione
- Traccia

Per una descrizione completa del file di configurazione, consultare "File di configurazione" a pagina 65. Consultare la documentazione di WebSphere Business Integrator per le informazioni sulla configurazione per il kernel.

**componente di traccia**
Software che scrivono messaggi di traccia. Viene utilizzato dalla maggior parte dei componenti del kernel. Per una panoramica sulla

traccia, consultare "Funzione di traccia" a pagina 28; per informazioni più dettagliate, consultare *Problem Determination Guide*.

## Messaggio e formato del messaggio

In MQSeries e MQSeries Adapter Offering, un *messaggio* è un gruppo di dati inviato da un programma e destinato a un altro programma. Il formato del messaggio dipende sempre dall'ubicazione del messaggio nel flusso dei messaggi in un dato momento. MQSeries Adapter Kernel specifica i seguenti tre tipi di messaggi:

- *Messaggio di integrazione*—Un messaggio costituito dai dati inviati da un'applicazione di origine convertiti in un altro formato, quale XML per l'invio ad un'applicazione di destinazione. Il messaggio di integrazione è inserito in un oggetto message-holder come i dati di testo del messaggio. XML è uno standard per la rappresentazione dei dati. Quando si utilizza il formato XML, il formato viene definito mediante un *DTD* (Document Type Definition). Un DTD è rappresentato da uno o più file contenenti una definizione formale di un documento—in questo caso, il testo del messaggio. Sebbene consigliato, il testo del messaggio non deve presentare necessariamente un formato neutro per le applicazioni. Il formato del testo del messaggio può essere esclusivo o specifico, tuttavia questo tipo di formato non è consigliato.

  Il BOD (*Business Object Documents*) può essere utilizzato da MQSeries Adapter Offering per definire i testi nei messaggi di integrazione. Un BOD è una rappresentazione di un processo aziendale che riguarda una o più aziende. Ad esempio, "aggiunta di ordini di acquisto" "visualizzazione della disponibilità del prodotto" e "aggiunta di ordini di vendita" I BOD vengono definiti nel formato XML dall'OAG (Open Applications Group). L'utilizzo dei BOD è consigliato, ma non è obbligatorio.
- *oggetto message-holder*—Un oggetto contenente il messaggio di integrazione ed ulteriori metadati di intestazione rappresentanti i valori di controllo specifici di MQSeries Adapter Kernel. L'adattatore di origine crea l'oggetto message-holder, imposta le informazioni di controllo appropriate e, se esiste un messaggio di integrazione da inviare, imposta i dati di testo. Gli adattatori di destinazione ricevono gli oggetti message-holder, richiamano i dati di testo e li convertono in dati specifici dell'applicazione di destinazione. Gli adattatori di origine e di destinazione vengono creati mediante MQSeries Adapter Builder.
- *Messaggi di comunicazione*—Ciascuna informazione specifica del trasporto comunicazioni più l'oggetto contenente il messaggio, convertiti in uno specifico formato di messaggio per il trasporto comunicazioni utilizzato. Determinati trasporti delle comunicazioni supportano più di un formato per la messaggistica. Solitamente, i valori dei metadati di intestazione del kernel combinati con il messaggio di comunicazione sono considerati come dati dell'applicazione dal trasporto di comunicazione. Per ulteriori informazioni,

consultare “Appendice A. Modalità di comunicazione” a pagina 103. Esempi di trasporto MQSeries sono l’intestazione messaggio specifica di MQSeries più l’oggetto message-holder organizzato. Di seguito vengono elencati formati di MQSeries specifici:

– L’intestazione messaggi di MQSeries aggiunta da MQSeries
– Se si utilizza MQSeries Integrator, l’intestazione messaggi specifica della versione:
  - L’intestazione messaggi di MQSeries Integrator versione 1, se si utilizza MQSeries Integrator versione 1.1
  - L’intestazione messaggi di MQSeries Integrator versione 2, se si utilizza MQSeries Integrator versione 2
– I metadati dell’intestazione specifici del kernel che rappresentano i valori di controllo
– Il messaggio di integrazione (dati di testo)

Per un elenco e una descrizione dei principali campi utilizzati nelle intestazioni messaggi di MQSeries Adapter Offering, consultare “Appendice C. Intestazioni di messaggi” a pagina 111.

## Instradamento e consegna

Il kernel instrada ogni messaggio dall’adattatore di origine e lo consegna all’adattatore di destinazione appropriato. L’instradamento viene eseguito in due fasi:

1. Il lato origine del kernel inserisce il messaggio nella coda messaggi appropriata.
2. Il lato di destinazione del kernel riceve il messaggio dalla coda e richiama l’adattatore di destinazione appropriato.

L’instradamento è determinato da molti fattori:

• Code messaggi. Le code messaggi devono essere configurate per supportare la funzione di instradamento di MQSeries Adapter Offering.
• I valori di controllo messaggi nel messaggio. Questi includono l’identificativo logico di origine, l’identificativo logico di destinazione, l’identificativo logico di risposta, la categoria di testo, il tipo di testo, l’identificativo di transazione, l’identificativo messaggio, la richiesta di conferma e l’indicazione data/ora. Per dettagli, consultare “Valori di controllo messaggi” a pagina 16. L’identificativo logico di destinazione del messaggio può sovrascrivere il file di configurazione del kernel. L’instradamento può essere modificato dinamicamente reimpostando i valori di controllo messaggi in ciascuna intestazione messaggio. Invece, il contenuto dei dati di testo del messaggio (messaggio di integrazione) non può determinare l’instradamento.

- I valori di controllo messaggi nel file di configurazione del kernel. Il file può specificare gli identificativi logici di destinazione, i nomi code e gli adattatori di destinazione associati. Determinare e modificare la configurazione editando questo file. Per ulteriori informazioni, consultare "File di configurazione" a pagina 65.
- Facoltativamente, MQSeries Integrator, che può essere utilizzato in messaggi broker, con l'instradamento complesso. L'instradamento può variare in maniera dinamica contemporaneamente alle modifiche apportate al testo del messaggio. Consultare "Funzione di MQSeries Integrator" a pagina 8. MQSeries Adapter Offering può eseguire soltanto l'instradamento semplice. L'instradamento semplice è basato sulla combinazione dei valori di controllo messaggi nel messaggio e i valori di controllo messaggi nel file di configurazione. Non si basa sul contenuto del testo del messaggio.

E' possibile richiedere al kernel di confermare la consegna del messaggio. Si tratta di una conferma a livello di applicazione.

## Flusso di runtime

In questa sezione viene fornita una descrizione dettagliata del flusso di runtime. In particolare, viene illustrato il modo il cui il kernel invia, instrada, crea una traccia e consegna un messaggio in un ambiente di produzione tipico. Per un diagramma del flusso di runtime, fare riferimento alla Figura 2 a pagina 15.

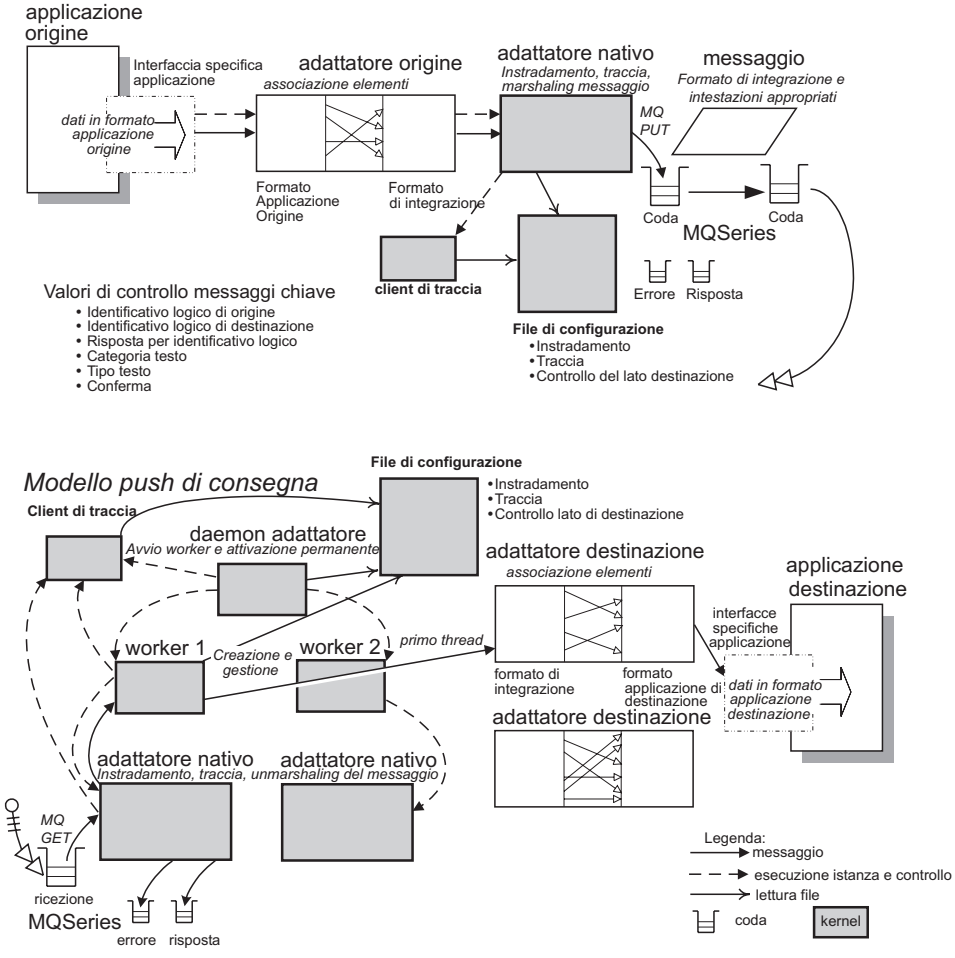



*Figura 2. Esecuzione del marshaling, invio e instradamento di un messaggio — panoramica.*

## Lato di origine del kernel

Questa sezione illustra il flusso di run-time sul lato di origine del kernel, descrive, cioè, come i dati si spostano dall'applicazione di origine attraverso un adattatore di origine su un trasporto di comunicazione. "Lato di destinazione del kernel" a pagina 20 descrive come i dati si spostano dal trasporto di comunicazione alla destinazione.

1. Utilizzando un'interfaccia specifica dell'applicazione, l'adattatore di origine riceve un messaggio dall'applicazione di origine. In genere, l'adattatore di origine viene richiamato dall'interfaccia specifica dell'applicazione.

2. L'adattatore di origine esegue le funzioni di cui è stato fornito in MQSeries Adapter Builder. Di solito, trasforma i dati nel formato dell'applicazione di origine in un formato di integrazione che non dipende dall'applicazione (per il testo del messaggio).

   L'adattatore di origine colloca molti valori di controllo messaggi nell'intestazione MQSeries Adapter Kernel; questi valori vengono utilizzati per preparare il messaggio. I primi cinque valori di controllo messaggi determinano l'organizzazione e l'instradamento e l'ultimo valore determina la conferma.

   **valori di controllo messaggi**

   **identificativo logico di origine**
   Identificativo logico dell'applicazione di origine. E' sempre richiesto nel messaggio.

   **identificativo logico di destinazione**
   Identificativo logico dell'applicazione di destinazione. Se non è presente nel messaggio, vengono utilizzati i valori predefiniti del file di configurazione. Nel file di configurazione, è possibile utilizzare più identificativi logici di destinazione al posto dei valori che non sono contenuti nel messaggio.

   **identificativo logico di risposta**
   L'identificativo logico dell'applicazione a cui inviare le risposte nel caso in cui venga richiesta una risposta. Viene impostato come valore predefinito sull'identificativo logico di origine nel messaggio.

   **categoria di testo**
   rappresenta il tipo di applicazione del messaggio—ad esempio, OAG o RosettaNet. È sempre richiesto nel messaggio.

   **tipo di testo**
   Rappresenta lo scopo specifico del messaggio—ad esempio, "aggiungere un ordine di vendita" o "sincronizzare l'inventario". È sempre richiesto nel messaggio.

   **conferma richiesta**
   Determina se l'applicazione di origine richiede una risposta. Di seguito vengono riportati i vari tipi di risposte:

   - Dati di risposta dall'applicazione di destinazione
   - Un messaggio di conferma BOD predefinito da OAG

     **Nota:** Il messaggio di conferma BOD viene predefinito da OAG. La categoria di testo è `OAG`, mentre il tipo di testo è `CONFIRM_BOD_003`. Può anche contenere dati.

Questa risposta è una conferma eseguita a livello dell'applicazione.

Quando il kernel utilizza il metodo `sendRequestResponse` per inviare il messaggio, il metodo `sendRequestResponse` utilizza solo la prima risposta ricevuta. Se il messaggio originale viene inviato a più destinazioni e si richiede una risposta (il che non è consigliabile), l'applicazione di origine riceve solo la prima risposta.

In base all'impostazione predefinita, non viene eseguita alcuna conferma; pertanto, non viene richiesta o inviata alcuna risposta.

3. L'adattatore di origine inizializza l'adattatore nativo, al quale passa:
   - L'identificativo logico dell'applicazione in cui è in esecuzione l'adattatore di origine.
   - L'oggetto message-holder, che contiene i valori di controllo messaggi ed i dati di testo del messaggio.
4. L'adattatore nativo analizza il file di configurazione per stabilire se la traccia è stata abilitata per l'identificativo logico di origine. Se la traccia è stata abilitata, l'adattatore nativo crea un'istanza di un client di traccia.
5. Il client di traccia analizza il file di configurazione per determinare il livello di traccia da utilizzare e per richiamare altri valori. Il client di traccia utilizza il livello di traccia per filtrare i messaggi di traccia. Per una panoramica sulla creazione della traccia, consultare "Funzione di traccia" a pagina 28; per informazioni più dettagliate, consultare *Problem Determination Guide*.
6. L'adattatore nativo cerca l'identificativo logico di destinazione nell'oggetto message-holder. Se presente, viene utilizzato.
   - In caso contrario, l'adattatore nativo ricerca l'identificativo logico di destinazione nel file di configurazione, in base all'identificativo logico di origine, alla categoria di testo e al tipo di testo.
   - In base all'identificativo logico di origine, l'adattatore nativo esegue una ricerca a più fasi dei valori relativi alla categoria e tipo di testo nel file di configurazione, secondo il seguente ordine:
     a. Valori del tipo e della categoria di testo specifici.
     b. Valore della categoria specifica di testo e valore del tipo di testo predefinito.
     c. Valore della categoria predefinita di testo e valore del tipo di testo specifico.
     d. Valori del tipo e della categoria di testo predefiniti.

     **Nota:** Il kernel utilizza ricerche a più fasi tutte le volte che cerca i valori nel file di configurazione.

7. Per ciascun identificativo logico di destinazione determinato nel passo precedente, l'adattatore nativo ricerca la *modalità di comunicazione*, in base all'identificativo logico di destinazione, alla categoria di testo e al tipo di testo. Di seguito vengono descritte le modalità di comunicazione supportate:

   **MQPP** Il kernel trasporta messaggi utilizzando i servizi di base di MQSeries.

   **MQRFH1** Il kernel trasporta messaggi utilizzando messaggi del broker e MQSeries utilizzando MQSeries Integrator versione 1.1.

   **MQRFH2** Il kernel trasporta i messaggi utilizzando MQSeries e messaggi dei broker mediante MQSeries Integrator versione 2.

   **MQBD** Il kernel trasporta messaggi utilizzando i servizi di base di MQSeries ma invia e riceve solo data di testo.

   **MQ** Il kernel trasporta messaggi utilizzando MQSeries.

   **JMS** Il kernel trasporta i messaggi utilizzando JMS (Java Message Service).

   **FILE** Il kernel inserisce e richiama i messaggi da un file. Questa modalità viene fornita solo per scopi diagnostici.

   In ciascuna modalità di comunicazione, la struttura di messaggio è differente. Consultare "Messaggio e formato del messaggio" a pagina 12. Per ulteriori informazioni sulle modalità di comunicazione, consultare "Appendice A. Modalità di comunicazione" a pagina 103.

   **Nota:** Se viene utilizzato MQSeries Integrator, la destinazione finale a cui MQSeries Integrator invia il messaggio deve utilizzare la stessa modalità di comunicazione utilizzata da MQSeries Integrator per ricevere i messaggi.

8. In base alla modalità di comunicazione, l'adattatore nativo esegue l'istanza di una classe secondaria interna per gestire il messaggio. Questa sottoclasse viene chiamata *servizio messaggio logico*. Ogni modalità di comunicazione presenta una sottoclasse servizio messaggio logico differente.

   L'adattatore nativo passa a questo servizio gli identificativi logici di destinazione, la categoria di testo e il tipo di testo.

9. La sottoclasse servizio messaggio logico individua i parametri necessari per inviare il messaggio. Ad esempio, se la modalità di comunicazione è MQPP, i parametri includono il formato ed i nomi delle code di errori, risposte e ricezione. In base agli identificativi logici di destinazione, alla

categoria di testo e al tipo di testo inoltrati, il servizio di messaggio logico esegue una ricerca a più fasi nel file di configurazione.

a. Valori del tipo e della categoria di testo specifici.
b. Valore della categoria specifica di testo e valore del tipo di testo predefinito.
c. Valore della categoria predefinita di testo e valore del tipo di testo specifico.
d. Valori del tipo e della categoria di testo predefiniti.

A questo punto, il servizio messaggio logico dispone di tutte le informazioni necessarie per instradare ed eseguire il marshaling del messaggio.

10. Il servizio messaggio logico effettua le seguenti attività:
    - Organizza il messaggio in modo appropriato alla modalità di comunicazione ed al formato. Ciascuna modalità di comunicazione utilizza un formato predefinito se non ne viene specificato uno. Ad esempio, se la modalità di comunicazione è MQRFH2, il servizio di messaggio logico crea intestazioni appropriate e struttura il messaggio per il trasporto utilizzando MQSeries ed il broker utilizzando MQSeries Integrator versione 2.
    - Invia il messaggio. Ad esempio, se la modalità di comunicazione è MQRFH2, il messaggio viene collocato nella coda di messaggi MQSeries appropriata.
11. Per inviare i messaggi, è possibile utilizzare due metodi.
    - Se l'adattatore nativo utilizza il metodo `sendMsg` per l'invio dei messaggi, l'adattatore nativo non aspetta la risposta.
    - Se l'adattatore nativo utilizza il metodo `sendRequestResponse` per inviare il messaggio, il servizio messaggio logico attende la risposta. L'adattatore nativo, mediante il servizio messaggio logico, controlla nella coda di risposta il *periodo di timeout di ricezione* impostato nel file di configurazione.

      Questo valore si basa sull'identificativo dell'applicazione di origine, sulla categoria e sul tipo di testo.

      – Se si riceve una conferma, l'adattatore nativo restituisce il messaggio.
      – Se la conferma non perviene entro il periodo di timeout specificato, l'adattatore nativo non restituisce il messaggio.
12. MQSeries o un altro software di messaggi trasporta il messaggio in base alla configurazione utilizzata. Se si desidera, MQSeries Integrator esegue i servizi di brokering. Consultare "Funzione di MQSeries Integrator" a pagina 8.
13. Se l'adattatore di origine non ha più bisogno dell'adattatore nativo, lo chiude per rendere le risorse disponibili.

## Lato di destinazione del kernel

Questa sezione descrive l'utilizzo di MQSeries Adapter Kernel stand-alone per ricevere ed elaborare i messaggi sul lato destinazione e l'utilizzo del kernel con WebSphere Application Server. Consultare "Utilizzo di MQSeries Adapter Kernel con WebSphere Business Integrator e WebSphere Application Server" a pagina 29 per informazioni dettagliate sull'utilizzo del kernel con JMS, il componente JMS Listener di WebSphere Business Integrator e WebSphere Application Server s ul lato destinazione del kernel. Questa sezione illustra il modello push di consegna in cui il kernel è responsabile dell'inizializzazione e della gestione della consegna del messaggio all'applicazione di destinazione. Per una breve descrizione dei modelli, consultare "modelli di consegna" a pagina 134.

### Panoramica sul worker dell'adattatore

In questa sezione vengono descritte la struttura e la funzionalità dei worker dell'adattatore MQSeries Adapter Kernel. Uno dei presupposti dell'architettura di MQSeries Adapter Kernel è che le applicazioni di destinazione non partecipano attivamente ai flussi dei dati di integrazione con le altre applicazione, per cui le applicazione solitamente non fanno parte attivamente del poll dei messaggi da elaborare. In questo caso, i dati dei messaggi devono essere attivamente inoltrati all'applicazione di destinazione. I worker dell'adattatore inoltrano i dati dei messaggi ad un'applicazione o a un altro servizio selezionando e richiamando una serie di tipi di interfacce di servizi.

MQSeries Adapter Kernel può eseguire l'host dei worker dell'adattatore eseguiti in un daemon stand-alone (daemon dell'adattatore) o in un server di applicazione Enterprise JavaBeans (correntemente, IBM WebSphere Application Server Advanced Edition). I messaggi arrivano al worker dell'adattatore in modi differenti, in base al tipo di ambiente di destinazione utilizzato. Se viene utilizzato un daemon di adattatore stand-alone, viene eseguito l'host di uno o più daemon dell'adattatore stand-alone che utilizzano l'adattatore nativo per ricevere i messaggi. Se viene utilizzato un server EJB, il componente Listener JMS riceve i messaggi e li inoltra ad un bean di messaggi worker (a volte denominato worker dell'adattatore message-driven-bean).

A prescindere dall'ambiente di destinazione utilizzato, una volta che il worker dell'adattatore ha ricevuto il messaggio, lo inoltra all'adattatore di destinazione appropriato. L'adattatore di destinazione, quindi, esegue le operazioni necessarie per la consegna del messaggio all'applicazione di destinazione. Gli adattatori di destinazione sono creati per applicazioni di destinazione specifiche. Il daemon dell'adattatore, il server di applicazione, il worker dell'adattatore stand-alone ed il bean di messaggio worker non sono specifici di una data applicazione di origine o destinazione.

Il worker dell'adattatore gestisce due tipi di interfacce dell'adattatore di destinazione: adattatori di comando EAB (Enterprice Access Builder) e bean di sessione servizio EJB. Ciascun tipo di adattatore include un gestore che imposta l'ambiente appropriato, accede a ulteriori informazioni di configurazione richieste per l'adattatore ed esegue altre attività di basso livello richieste per le operazioni dell'adattatore. Il gestore utilizzato dipende dal tipo di adattatore indicato nel file di configurazione. I due tipi di gestori eseguono le attività riportate di seguito:

- Il gestore EAB richiama la classe di collegamento utilizzata per fornire le informazioni di collegamento all'adattatore di destinazione e inizializza l'esecuzione di IBM Common Connector Framework (CCF). La classe di collegamento viene inoltrata all'identificativo logico dell'applicazione di destinazione per richiamare le informazioni di collegamento specifiche dell'applicazione.
- Il gestore EJB richiama un collegamento JNDI (Java Naming and Directory Interface™) e poi richiama l'interfaccia remota del bean di sessione servizio e le altre informazioni richieste per accedere al bean di sessione servizio.

Il flusso di base del processo di un worker dell'adattatore in un daemon dell'adattatore stand-alone è il seguente:

1. All'avvio, il daemon dell'adattatore esegue l'istanza di uno o più worker di adattatori stand-alone in base alle informazioni fornite nel file di configurazione del kernel. L'identificativo logico dell'applicazione e i valori della categoria di testo facoltativa e del tipo di testo sono inoltrati al daemon dell'adattatore. I valori della categoria di testo e del tipo di testo sono utilizzati per richiamare valori aggiuntivi di configurazione.
2. Ciascun worker dell'adattatore stand-alone esegue le attività riportate di seguito:
   a. Il worker dell'adattatore esegue l'istanza di un adattatore nativo e avvia la ricezione dei messaggi. Ciascun messaggio viene ricevuto sotto il controllo transazionale e restituito al worker dell'adattatore come oggetto message-holder.
   b. Per ciascun messaggio ricevuto, il worker dell'adattatore richiama dal file di configurazione il tipo di comando dell'adattatore di destinazione per l'elaborazione del messaggio e richiama il gestore appropriato al tipo di comando.
   c. Il gestore richiama dal file di configurazione le altre informazioni necessarie all'esecuzione dell'istanza dell'adattatore di destinazione. Viene eseguita l'istanza dell'adattatore di destinazione e viene inoltrato il messaggio.
   d. Se il messaggio è elaborato correttamente (senza eccezioni, errori o dati non corretti), il messaggio viene eliminato dalla coda dei messaggi in entrata. Se il messaggio non viene elaborato correttamente, viene collocato nella coda degli errori. Se il messaggio non viene elaborato

correttamente e non è possibile collocarlo nella coda degli errori, ne viene eseguito il roll back e tutti i worker vengono spenti.

Il flusso di base del processo di un worker dell'adattatore in WebSphere Application Server è il seguente:

1. Un processo JMS Listener operante con un server EJB di WebSphere Application Server Advanced riceve un messaggio JMS. Richiama, quindi, un bean di messaggio worker per elaborare il messaggio. L'identificativo logico dell'applicazione e i valori della categoria di testo facoltativa e del tipo di testo sono parte dell'ambiente del bean di messaggi worker. I valori della categoria di testo e del tipo di testo sono utilizzati per richiamare valori aggiuntivi di configurazione.
2. Ciascun bean di messaggio worker esegue le attività riportate di seguito:
   a. Il bean di messaggio worker esegue l'istanza di un adattatore nativo e utilizza il metodo `receiveMsg` sull'adattatore nativo, a cui inoltra il messaggio JMS. L'adattatore nativo converte il messaggio JMS in un oggetto messaggio e restituisce un oggetto message-holder.
   b. Per ciascun oggetto message-holder ricevuto, il worker dell'adattatore richiama dal file di configurazione il tipo di comando dell'adattatore di destinazione per l'elaborazione dell'oggetto message-holder e richiama il gestore appropriato al tipo di comando.
   c. Il gestore richiama dal file di configurazione le altre informazioni necessarie all'esecuzione dell'istanza dell'adattatore di destinazione. Viene eseguita l'istanza dell'adattatore di destinazione e viene inoltrato l'oggetto message-holder.
   d. Se l'oggetto message-holder è elaborato correttamente (senza eccezioni, errori o dati non corretti), il messaggio viene eliminato dalla coda dei messaggi in arrivo. Se l'oggetto message-holder non viene elaborato correttamente, viene collocato nella coda degli errori. Se il messaggio non viene elaborato correttamente e non è possibile collocarlo nella coda degli errori, ne viene eseguito il roll back e tutti i worker vengono spenti.

Il flusso di run-time sul lato destinazione con un daemon dell'adattatore e un worker dell'adattatore stand-alone è il seguente:

1. Esiste un daemon dell'adattatore per ogni coda di ricezione dell'applicazione di destinazione. Questo daemon è in esecuzione.

   All'avvio, viene fornito un nome utilizzato come identificativo dell'applicazione. Solitamente, ciascun nome di daemon dell'adattatore è basato sull'identificativo logico di destinazione —l'identificativo logico dell'applicazione di destinazione. Ad esempio, se il daemon dell'adattatore è in funzione per un'applicazione di destinazione il cui identificativo logico di destinazione è `ABC`, il nome del daemon dell'adattatore è `ABCdaemon`.

Tra i vari parametri che è possibile passare al daemon dell'adattatore all'avvio, vi è la categoria e il tipo di testo. L'adattatore nativo li utilizza successivamente per determinare la modalità di comunicazione e la coda di ricezione per i messaggi in arrivo.

Per le istruzioni di avvio del daemon dell'adattatore, consultare "Avvio del kernel" a pagina 92.

2. All'avvio il daemon dell'adattatore legge il file di configurazione per determinare se la traccia è abilitata o meno per tale nome di daemon di adattatore. In caso affermativo, il daemon dell'adattatore crea un'istanza di un client di traccia.

   Per dettagli sulla traccia, consultare *Problem Determination Guide*.

3. All'avvio, il daemon dell'adattatore esegue l'istanza del primo worker a cui inoltra il nome del daemon dell'adattatore e la categoria ed il tipo del testo del messaggio.
4. Il primo worker legge il file di configurazione per determinare se la traccia è abilitata o meno per tale nome di daemon dell'adattatore. Se la traccia è stata abilitata, il primo worker crea un'istanza di un client di traccia; quest'ultimo effettua una ricerca nel file di configurazione per determinare il livello di traccia. Per un elenco dei livelli di traccia validi, consultare *Problem Determination Guide*.
5. Il primo worker esegue una ricerca nel file di configurazione in base all'identificativo dell'applicazione del daemon, dei valori che indicano il numero minimo di worker di cui creare un'istanza e da avviare.

   Il primo worker ricerca anche l'*identificativo dell'applicazione di dipendenza*. Questo identificativo corrisponde al nome dell'applicazione servita dal worker e viene passato all'adattatore nativo.
6. Il daemon dell'adattatore esegue una query nel primo worker per stabilire il numero minimo di worker.
7. Il daemon dell'adattatore avvia il primo worker, quindi crea un'istanza e avvia il numero minimo di worker.

   Più worker consentono di abilitare la consegna di messaggi multithread agli adattatori di destinazione. Ogni worker, insieme al rispettivo adattatore nativo, può gestire un thread. Nel caso di un solo worker, la consegna dei messaggi all'adattatore di destinazione e, quindi all'applicazione di destinazione, è a thread singolo.

   Sui sistemi AIX, sono disponibili due criteri di pianificazione: una pianificazione basata sul processo e una pianificazione basata sul sistema. Nelle pianificazioni basate sui processi (impostazione predefinita), tutti i thread utenti sono associati ad un pool di thread del kernel del sistema operativo (OS) e sono eseguiti in un pool di processori virtuali. Durante la pianificazione basata sul sistema, ogni thread utente viene mappato con un singolo thread del sistema operativo e viene eseguito su un singolo processore virtuale. Se si utilizzano adattatori C di origine richiamati da

file eseguibili C su AIX, è necessario utilizzare una pianificazione basata sul sistema. Per informazioni sulle modalità di impostazione del criterio di pianificazione dei thread su AIX, fare riferimento al passo 6 a pagina 45.

Sui sistemi Windows, HP-UX, Solaris e OS/400 è supportata la sola pianificazione basata sul processo.

Gli altri worker completano le seguenti attività:

8. Ciascun worker crea un'istanza dell'adattatore nativo associato. A ogni worker viene associato un adattatore nativo. L'identificativo di applicazione di dipendenza, la categoria e il tipo di testo vengono passati all'adattatore nativo. Questi tre valori vengono utilizzati dall'adattatore nativo per determinare la modalità di comunicazione e, utilizzando il servizio messaggio logico, il formato e la coda di ricezione per i messaggi in entrata. Questo processo è simile a quello utilizzato per l'invio dei messaggi.
9. L'adattatore nativo riceve il messaggio di comunicazione dalla coda di ricezione sotto il controllo commit e lo converte in un oggetto message-holder. Vengono rimosse tutte le intestazioni specifiche del trasporto di comunicazione fatta eccezione per l'intestazione del kernel nativo.
10. L'adattatore nativo inoltra l'oggetto message-holder al worker che legge la categoria di testo, il tipo di testo e il valore di conferma richiesto dall'intestazione kernel nativo del messaggio.

    In base all'identificativo dell'applicazione di dipendenza, alla categoria di testo e al tipo di testo, il worker esegue una ricerca a più fasi nel file di configurazione del tipo di comandi di destinazione da richiamare, nell'ordine seguente:

    a. Valori del tipo e della categoria di testo specifici.
    b. Valore della categoria specifica di testo e valore del tipo di testo predefinito.
    c. Valore della categoria predefinita di testo e valore del tipo di testo specifico.
    d. Valori del tipo e della categoria di testo predefiniti.

    In base al tipo di comandi di destinazione, il worker determina il gestore del tipo dell'adattatore di destinazione e la classe Java che elabora il particolare tipo di adattatore. Crea un'istanza dell'adattatore di destinazione.
11. Esistono due tipi di gestori di adattatori: gestori di adattatori di destinazione dei comandi EAB e i gestori di adattatori di destinazione del bean di sessione servizio EJB. I differenti tipi di gestori di adattatori funzionano nel modo seguente:

**Nota:** Il gestore dell'adattatore di destinazione bean di sessione servizio EJB è supportato solo con WebSphere Business Integrator eseguito con WebSphere Application Server sulla piattaforma Windows NT.

- Se viene richiamato un gestore dell'adattatore di destinazione comandi EAB, esso inizializza l'ambiente CCF (Common Connector Framework), imposta una classe di collegamento con un nome ottenuto dal file di configurazione e richiama l'adattatore di destinazione EAB con il nome ottenuto dal file di configurazione.
- Il gestore dell'adattatore di destinazione bean di sessione servizio EJB deve interagire con WebSphere Business Integrator e WebSphere Application Server per ottenere le informazioni di configurazione appropriate e richiamare il bean di sessione servizio EJB. Consultare "Utilizzo di MQSeries Adapter Kernel con WebSphere Business Integrator e WebSphere Application Server" a pagina 29 per informazioni dettagliate sull'utilizzo del kernel con JMS, il componente JMS Listener di WebSphere Business Integrator e WebSphere Application Server s ul lato destinazione del kernel.

12. Ciascun tipo di adattatore ha un'interfaccia differente e classi di supporto richieste:
    - Un comando dell'adattatore di destinazione EAB presenta tre metodi di richiamo; questi metodi sono eseguiti nell'ordine di seguito riportato:
      a. Il metodo *set message input*, che imposta il messaggio da elaborare nell'adattatore di destinazione.
      b. Il metodo *execute*, che elabora il messaggio inserito nell'adattatore di destinazione mediante il metodo set message input e che resta in attesa di una risposta.
         1) L'adattatore di destinazione esegue le funzioni di cui è stato corredato. Solitamente, trasforma i dati dal messaggio di integrazione nel formato dell'applicazione di destinazione. Esegue la mappatura tra elementi.
         2) L'adattatore di destinazione, utilizzando un'interfaccia specifica dell'applicazione, invia il messaggio all'applicazione di destinazione.
         3) In base alla natura dell'applicazione di destinazione, l'applicazione di destinazione invia o meno una risposta di conferma all'adattatore di destinazione.
      c. Il metodo *get message output*, che riceve la risposta dall'adattatore di destinazione. La risposta può indicare semplicemente che l'applicazione di destinazione ha ricevuto il messaggio; può anche contenere dati.

- Un bean di sessione servizio EJB richiama un metodo che richiede un oggetto `TerminalDataContainer`. I dati restituiti dal metodo sono considerati dati di risposta e devono essere un oggetto di tipo `TerminalDataContainer`.

13. Se il comando dell'adattatore di destinazione non lancia un'eccezione oppure se non riceve un risposta di conferma BOD (che può indicare un errore), il worker conferma il messaggio ricevuto dalla coda di ricezione utilizzando l'adattatore nativo.
14. Nel caso in cui sia stato richiesta una conferma, il worker richiama il metodo `sendResponse` sull'adattatore nativo.
    - Se l'adattatore di destinazione crea una risposta, colloca l'identificato logico di risposta del messaggio originale nel campo dell'identificativo logico di destinazione del messaggio di risposta.
    - Se l'adattatore di destinazione non ha creato una risposta, il worker crea un messaggio di risposta di conferma BOD contenente lo stato di completamento.
      – Se non ci sono errori, l'esito dello stato di completamento è positivo.
      – Se ci sono errori, lo stato di completamento viene impostato su una condizione di errore.
15. La risposta viene inviata.
    a. Il worker invia il messaggio di risposta, se presente, all'adattatore nativo.
    b. L'adattatore nativo inserisce il messaggio di risposta nella coda di risposta.
    c. L'adattatore nativo invia il messaggio di risposta, in base al messaggio originale che ha ricevuto:
       - Nel caso di un messaggio di richiesta di MQSeries, l'adattatore nativo richiama le informazioni sulla coda per la risposta dal messaggio di richiesta di MQSeries. Queste informazioni sovrascrivono l'identificativo logico di destinazione nel messaggio.
       - Se non si tratta di un messaggio di richiesta di MQSeries, l'adattatore nativo utilizza il metodo `sendMsg` per inviare la risposta.
16. In caso si verifica un'eccezione o un messaggio di risposta BOD constato di errore, il worker registra il messaggio di eccezione in un file delle eccezioni chiamato `EpicSystemExceptionFile`*`nnnnnnnn`*`.log` che risiede sulla stessa directory del daemon dell'adattatore dove *nnnnnnnn* indica il numero del file di log. Inoltre, se le classi WebSphere Business Integrator sono installate, inviano un'eccezione al componente WebSphere Business Integrator Solution Management. Consultare "Messaggi di eccezione" a pagina 96.
17. Nel caso di un'eccezione o di un messaggio di risposta di conferma BOD con un stato di errore, il worker indica all'adattatore nativo di inserire il

messaggio originale nella coda di errore. Il nome di questa coda viene determinato dal file di configurazione in base all'identificativo logico di dipendenza, alla categoria e al tipo di testo del messaggio originale.

In base all'identificativo dell'applicazione di dipendenza, alla categoria di testo e al tipo di testo, il worker esegue una ricerca a più fasi nel file di configurazione nell'ordine seguente:

a. Valori del tipo e della categoria di testo specifici.
b. Valore della categoria specifica di testo e valore del tipo di testo predefinito.
c. Valore della categoria predefinita di testo e valore del tipo di testo specifico.
d. Valori del tipo e della categoria di testo predefiniti.

- Se l'adattatore nativo è in grado di inserire il messaggio di errore nella coda di errore, l'adattatore nativo conferma il messaggio dalla coda di ricezione.
- Se l'adattatore nativo non è in grado di inserire il messaggio di errore nella coda di errore, si verifica quanto riportato di seguito:
  a. Il worker indica all'adattatore nativo di eseguire un annullamento, ossia, di non eseguire la conferma.
  b. Il worker imposta un flag che indica a tutti worker eseguiti sul daemon dell'adattatore di arrestarsi. Ciò significa che vi sono problemi con il messaggio. L'arresto di tutti i worker impedisce ad altri worker di elaborare di nuovo lo stesso messaggio relativo al problema con lo stesso risultato.
  c. Se si verifica un errore di insufficienza di memoria, l'eccezione viene gestita come tutte le altre eccezioni, ma il worker imposta un flag per arrestarsi una volta completata l'elaborazione del messaggio corrente. In tal modo, si rende disponibile una maggiore quantità di memoria per gli altri worker.

18. Quando l'adattatore nativo notifica al worker che il lavoro è stato completato, il worker controlla i seguenti due flag:
    - Controlla se arrestare questo worker. L'arresto è necessario in condizioni di insufficienza di memoria Java.
    - Controlla se arrestare tutti i worker, per le stesse cause sopra riportate.
19. Se uno dei flag è impostato, il worker si arresta. Se nessuno dei flag è impostato, il worker elabora il messaggio successivo. Il worker richiede che l'adattatore nativo riceva un messaggio.
20. Se un messaggio di risposta non viene inserito nella coda di risposta oppure se un messaggio di errore viene inserito nella coda di errore, si verifica quanto riportato di seguito:
    a. MQSeries o il software per la messaggistica utilizzato consegna questo messaggio di nuovo al lato di origine del kernel.

b. Se l'adattatore di origine ha richiamato il metodo `sendRequestResponse` del relativo adattatore nativo, il kernel recupera il messaggio dalla coda di risposta e lo restituisce all'adattatore di origine. Se quest'ultimo ha richiamato il metodo `sendMsg`, il kernel inserisce il messaggio nella coda di ricezione dell'applicazione di origine.

## Funzioni transazionali

Una *transazione* è un insieme di operazioni che devono essere eseguite come singola unità di lavoro. Se tutte le operazioni che formano una transazione terminano con esito positivo, la transazione viene *confermata*, ossia, tutte le operazioni vengono eseguite. Se, invece, una o più operazioni che formano una transazione terminano con esito negativo, la transazione viene *annullata*, vale a dire, nessuna delle operazioni viene eseguita. Utilizzando le funzionalità transazionali di MQSeries Adapter Kernel, un adattatore di origine è in grado di eseguire una serie di operazioni come una singola unità, garantendo l'esito positivo di tutte le operazioni se la transazione viene confermata oppure l'esito negativo se la transazione viene annullata.

Le funzionalità transazionali possono essere create negli adattatori utilizzando MQSeries Adapter Builder oppure i metodi `begin`, `rollback` e `commit` sulla classe `EpicNativeAdapter` dell'API Java del kernel. Nel caso in cui uno di questi metodi venga utilizzato in maniera errata (ad esempio, il metodo `commit` viene richiamato senza aver prima richiamato `begin` oppure `begin` viene richiamato nell'ambito di un'altra transazione), il kernel ignora la chiamata e invia un avviso alla traccia. Per informazioni sulle modalità di utilizzo dell'API, consultare "Capitolo 5. Utilizzo delle API di MQSeries Adapter Kernel" a pagina 99.

### Limitazioni

Le limitazioni riportate di seguito sono relative alle funzionalità transazionali del kernel:

- Le transazioni non sono supportate con il metodo `sendRequestResponse`.
- Le transazioni nidificate (transazioni richiamate all'interno di altre transazioni) non sono supportate.
- Le transazioni non sono supportate dai modi di comunicazione; consultare "Appendice A. Modalità di comunicazione" a pagina 103 per ulteriori informazioni.

## Funzione di traccia

Un messaggio di traccia contiene lo stato dell'elaborazione di un messaggio a un determinato stadio all'interno del kernel. E' possibile utilizzare i messaggi di traccia per facilitare la diagnosi dei problemi con il kernel o con gli adattatori. MQSeries Adapter Kernel *Problem Determination Guide* descrive le modalità di utilizzo delle funzioni di traccia con il kernel.

## Utilizzo di MQSeries Adapter Kernel con WebSphere Business Integrator e WebSphere Application Server

In questa sezione viene descritto l'utilizzo di MQSeries Adapter Kernel con i prodotti WebSphere Business Integrator e WebSphere Application Server. Per ulteriori informazioni, consultare la documentazione di WebSphere Business Integrator.

### JMS Listener

WebSphere Business Integrator fornisce un componente chiamato JMS Listener che funzione con MQSeries Adapter Kernel e WebSphere Application Server Advanced Edition per fornire una modalità alternativa per la consegna dei messaggi alle applicazioni di destinazione. JMS Listener viene eseguito nel server EJB (Enterprise JavaBeans) di WebSphere Application Server. Questa sezione fornisce una panoramica sulla funzionalità di JMS Listener. Per ulteriori informazioni, inclusi dettagli sulla configurazione di WebSphere Business Integrator e JMS Listener, consultare la documentazione WebSphere Business Integrator. Consultare "Configurazione del kernel" a pagina 58 per informazioni sulla configurazione di MQSeries Adapter Kernel per il riconoscimento di JMS Listener come destinazione. L'utilizzo di JMS Listener come destinazione è equivalente all'invio di un massaggio ad un daemon dell'adattatore.

Prima di poter utilizzare JMS Listener, è necessario distribuire un worker di adattatore bean di messaggi MQSeries Adapter Kernel e gli adattatori bean di sessione servizio Java o gli adattatori EAB per il lato destinazione del kernel. Eseguire questa attività utilizzando MQSeries Adapter Builder. In un ambiente WebSphere Business Integrator, le operazioni del kernel in WebSphere Application Server sono simili alle operazioni con un daemon di adattatore stand-alone, ma JMS Listener riceve il messaggio per conto del worker di adattatore e richiama il worker di adattatore appropriato. In un ambiente MQSeries Adapter Kernel stand-alone, il daemon di adattatore avvia i worker di adattatore che a turno ricevono direttamente i messaggi.

La sequenza degli eventi MQSeries Adapter Kernel con JMS Listener è la seguente:

1. JMS Listener, controllando una coda JMS, riceve un oggetto messaggio JMS da un client EJB o da un'applicazione non EJB.
2. JMS Listener esegue l'istanza di un *bean di messaggio worker* a cui inoltra poi l'oggetto messaggio. Il bean di messaggio worker è un'istanza di un *bean di sessione*, un tipo di bean enterprise che racchiude i dati temporanei associati ad un client specifico.
3. Il bean di messaggio worker converte l'oggetto messaggio JMS in un oggetto message-holder MQSeries Adapter Kernel.

4. In base ai valori di intestazione del messaggio, il kernel richiama un adattatore EAB o un adattatore EJB. Se il tipo di adattatore da richiamare è EAB, il flusso dei dati è uguale all'ambiente stand-alone. Se il tipo di adattatore da richiamare è EJB, viene richiamato un gestore EJB che esegue le attività di seguito riportate:
   - Determina il bean di sessione servizio corretto (interfaccia home) da richiamare, il metodo appropriato da utilizzare e il tipo di parametri di immissione del metodo per l'oggetto `TerminalDataContainer`.
   - Converte i dati dell'applicazione contenuti nell'oggetto message-holder per la struttura di dati `TerminalDataContainer` appropriata per il bean di sessione servizio utilizzando un `Mapper` di classe. L'oggetto `TerminalDataContainer` contiene i metadati dell'oggetto message-holder e gli oggetti di applicazione. In molti casi, l'oggetto di applicazione è la stringa di documento XML dei dati di testo dell'oggetto message-holder.
   - Richiama il bean di sessione servizio, inoltrando l'oggetto `TerminalDataContainer` al metodo appropriato sul bean di sessione servizio. Il bean di sessione servizio, che è parte dell'adattatore servizio Java, è la destinazione del messaggio.
5. Se è stata richiesta una risposta, il bean di messaggio worker converte l'oggetto `TerminalDataContainer` di risposta in un oggetto message-holder ed invia la risposta utilizzando l'adattatore nativo.
6. Se si verifica un errore, il baen di messaggio worker colloca l'oggetto message-holder in una coda degli errori utilizzando l'adattatore nativo.

## Supporto lingue nazionali

MQSeries Adapter Kernel fornisce il supporto delle lingue nazionali se vengono utilizzati adattatori Java. Il supporto delle lingue nazionali non viene fornito con gli adattatori C.

# Capitolo 2. Pianificazione dell'installazione del kernel

Questo capitolo elenca i prerequisiti e i componenti di MQSeries Adapter Kernel.

Per ulteriori dettagli, visitare il sito Web della famiglia di prodotti MQSeries, al seguente indirizzo:

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

L'IBM si riserva il diritto di aggiornare le informazioni fornite in questo testo. Per le più recenti informazioni relative ai livelli di software supportati, consultare:

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

## Hardware

MQSeries Adapter Kernel viene eseguito sugli hardware:

- Macchina PC IBM (o compatibile) con Windows NT 4.0, Service Pack 5 o successivo, o Windows 2000, Service Pack 1.
- Un computer IBM RS/6000 sul quale sia in esecuzione AIX versione 4.3.2 o 4.3.3.
- Un computer HP Series 9000 sul quale sia in esecuzione HP-UX versione 11.0.
- Macchina Sun SPARC o UltraSPARC con Solaris versione 8.
- Un computer IBM AS/400 o iSeries sul quale sia in esecuzione OS/400 versione 4.4 o 4.5.

  **Nota:** Per installare MQSeries Adapter Kernel su OS/400, è necessario che un sistema Windows si interfaccia con il computer AS/400. Per dettagli, consultare "Prerequisiti per l'installazione di OS/400" a pagina 34.

MQSeries Adapter Kernel richiede almeno circa 25 MB di spazio sul disco per i dati e il codice del prodotto.

Assicurarsi che lo spazio sul disco sia sufficiente per poter contenere gli adattatori. Le dimensioni degli adattatori dipendono dalle dimensioni delle strutture dati, dalla complessità della creazione di corrispondenze e dal codice personalizzato utilizzato. Di seguito vengono riportati alcuni esempi sulle dimensioni degli adattatori sui sistemi Windows. In base alle specifiche esigenze, gli adattatori utilizzati possono richiedere una quantità maggiore o

minore di spazio su disco. Ciascun esempio rappresenta l'origine adattatore, il codice dell'adattatore compilato, l'origine API e la codifica API compilata in MB o KB.

- Adattatore di origine per l'aggiunta di ordini di vendita: 1,89 MB
- Adattatore di destinazione per la sincronizzazione di un record di un cliente: 389 KB
- Adattatore di destinazione per la sincronizzazione di un record di inventario: 161 KB
- Adattatore di destinazione per la sincronizzazione di un articolo: 249 KB
- Adattatore di destinazione per la sincronizzazione di un ordine di vendita: 579 KB

Inoltre, è necessario destinare 20 MB per lo spazio di lavoro per il kernel e gli adattatori. I requisiti dello spazio di lavoro possono variare in base a vari fattori, quali la quantità e le dimensioni delle code e le dimensioni dei file di traccia.

## Software

Questa sezione elenca il software supportato da utilizzare con MQSeries Adapter Kernel. Di seguito vengono indicati i livelli supportati. Consultare "Appendice B. Configurazioni convalidate" a pagina 109. I compilatori C sono richiesti per i sistemi di sviluppo e non per quelli di produzione. I programmi di compilazione C qui elencati sono stati testati con esito positivo con MQSeries Adapter Kernel; è possibile che altri programmi di compilazione C funzionino correttamente con il kernel, ma non sono ufficialmente supportati.

Per i sistemi Windows:

- Microsoft Windows NT versione 4.0, Service Pack 5 o successivo; o Microsoft Windows 2000, Service Pack 1. Per determinare versione e service pack di Microsoft Windows, aprire Windows Explorer e poi dare clic su **? > Informazioni su Windows**.
- Microsoft Visual C++ 6.0 Compiler.
- MQSeries versione 5.2 con SupportPac MA88.
- IBM Java Development Kit (JDK) versione 1.2.2 o 1.3.

  **Nota:** Windows NT e Windows 2000 sono correntemente le uniche piattaforme su cui MQSeries Adapter Kernel supporta JDK versione 1.3.

Per AIX:

- Sistema operativo AIX versione 4.3.2 o 4.3.3.
- IBM C Set++ per AIX versione 3.1.3.

- MQSeries versione 5.2 con SupportPac MA88.
- Java Development Kit versione 1.2.2. JDK 1.3 non è supportato.
- X Window System (X11R5 o successiva). Necessario per l'installazione ma non durante il runtime.

Per HP-UX:

- Sistema operativo HP-UX versione 11.0.
- HP-UX C/ANSI C Compiler. Per dettagli, consultare il file `readme.txt`.
- MQSeries versione 5.2 con SupportPac MA88.
- Java Development Kit versione 1.2.2. JDK 1.3 non è supportato.
- X Window System (X11R5 o successiva). Necessario per l'installazione ma non durante il runtime.

Per Solaris:

- Ambiente operativo Solaris versione 8.
- Sun Workshop Compilers C/C++. Per dettagli, consultare il file `readme.txt`.
- MQSeries versione 5.2 con SupportPac MA88.
- Java Development Kit versione 1.2.2. JDK 1.3 non è supportato.
- X Window System (X11R5 o successiva). Necessario per l'installazione ma non durante il runtime.

Per OS/400:

- Sistema operativo OS/400 versione 4.4 o 4.5, inclusi i seguenti programmi:
  - Java Toolkit e Java Developer Kit versione 1.2.2. JDK 1.3 non è supportato. Java Toolkit e Java Developer Kit vengono forniti con il numero di licenza 5769–JV1. Consultare "Prerequisiti per l'installazione di OS/400" a pagina 34 per ulteriori dettagli sulle versioni di Java Developer Kit richieste per installare MQSeries Adapter Kernel su un sistema AS/400.
  - L'opzione Host Servers, fornita con il numero di programma concesso su licenza 5769–SS1, opzione 12.
  - Qshell Interpreter, fornito con il numero di programma concesso su licenza 5769–SS1, opzione 30.
  - TCP/IP, fornito con il numero di programma concesso su licenza 5769–TC1.
  - Integrated Language Environment C per AS/400, fornito come con programma con licenza numero 5769–CX2.
- MQSeries versione 5.2 con SupportPac MA88.

Consultare "Prerequisiti per l'installazione di OS/400" a pagina 34 per ulteriori dettagli relativi all'installazione di MQSeries Adapter Kernel su OS/400.

I prodotti elencati di seguito sono supportati con MQSeries Adapter Kernel:

- MQSeries versione 5.2 con SupportPac MA88

  **Nota:** Se non viene utilizzato MQSeries, deve essere utilizzato un altro prodotto software di messaggistica, quale JMS (Java Message Service).

- MQSeries Integrator versione 1.1
- MQSeries Integrator versione 2

Per un elenco delle configurazioni convalidate di MQSeries Adapter Kernel, MQSeries e MQSeries Integrator, consultare "Appendice B. Configurazioni convalidate" a pagina 109.

## Prerequisiti per l'installazione di OS/400

Questa sezione descrive i prerequisiti per l'installazione di MQSeries Adapter Kernel su un sistema AS/400 o iSeries. Per istruzioni più dettagliate sull'installazione di MQSeries Adapter Kernel su un sistema AS/400, fare riferimento al passo 3 a pagina 42. Dal momento che i terminali di AS/400 non supportano la grafica Java, è necessario utilizzare una workstation abilitata alla grafica, come un sistema Windows, per eseguire il programma di installazione della GUI su base Java del kernel. La workstation può interfacciarsi con il sistema AS/400 in uno dei seguenti modi:

- Mediante l'AWT remoto, in cui tutta la grafica viene elaborata sul sistema AS/400 e visualizzata sulla workstation. Maggiori dettagli a riguardo vengono forniti in "Utilizzo dell'AWT remoto".
- Un client collegato, sul quale la workstation elabora e visualizza la grafica. Maggiori dettagli a riguardo vengono forniti in "Utilizzo di un client collegato" a pagina 35.

In questa sezione si presume che l'utente si serva di un sistema Windows come workstation abilitata alla grafica.

### Utilizzo dell'AWT remoto

Quando si utilizza l'AWT remoto, l'elaborazione della grafica Java viene eseguita sul sistema AS/400 e viene visualizzata su una workstation client collegata al sistema AS/400. Questa sezione descrive i requisiti da soddisfare per installare MQSeries Adapter Kernel su un sistema AS/400 utilizzando l'AWT remoto.

I programmi elencati di seguito devono essere installati con OS/400:

- Java Toolkit e Java Developer Kit versione 1.2.2. Java Toolkit e Java Developer Kit vengono forniti con il numero di licenza 5769–JV1. Le funzionalità remote di AWT su OS/400 vengono fornite da Java Developer Kit.

- TCP/IP, fornito con il numero di programma concesso su licenza 5769–TC1. Per maggiori informazioni su TCP/IP, consultare i documenti *AS/400 TCP/IP Fastpath Setup Information* e *AS/400 TCP/IP Configuration* disponibili dalla libreria AS/400 all'indirizzo www.ibm.com/servers/eserver/iseries/library/.

Di seguito vengono elencati i requisiti della workstation:

- Un personal computer IBM (o compatibile) sul quale sia in esecuzione Windows 95, Windows 98, Windows NT o Windows 2000.
- Una connessione TCP/IP al sistema AS/400.
- JDK 1.2.2 o successiva.

Per installare e avviare l'AWT remoto, completare i seguenti passi:

1. Assicurarsi che JDK 1.2.2 o successiva sia stato installato sulla workstation.
2. Assicurarsi che sia stata stabilita una connessione TCP/IP tra il sistema AS/400 e la workstation.
3. Copiare il file `RAWTGui.jar` dalla directory `/QIBM/ProdData/Java400/jdk12` sul sistema AS/400 in una directory sulla workstation.
4. Sulla workstation, visualizzare la directory in cui è stato copiato il file `RAWTGui.jar` e avviare AWT remoto immettendo il seguente comando:

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

**Nota:** Data l'intensità delle risorse impiegate per l'elaborazione grafica Java su un sistema AS/400, l'installazione di MQSeries Adapter Kernel utilizzando AWT remoto può potenzialmente impiegare più tempo rispetto all'utilizzo di un client collegato.

Per ulteriori informazioni su AWT remoto, visitare la libreria AS/400 all'indirizzo www.ibm.com/servers/eserver/iseries/library/.

### Utilizzo di un client collegato

Quando si utilizza un client collegato per installare MQSeries Adapter Kernel su un sistema AS/400, la grafica Java viene elaborata sulla workstation client e non sul sistema AS/400. Questa sezione descrive i requisiti da soddisfare per installare MQSeries Adapter Kernel su un sistema AS/400 utilizzando un client collegato.

I programmi elencati di seguito devono essere installati con OS/400:

- Java Toolkit e Java Developer Kit versione 1.2.2. Java Toolkit e Java Developer Kit vengono forniti con il numero di licenza 5769–JV1.
- L'opzione Host Servers, fornita con il numero di programma concesso su licenza 5769–SS1, opzione 12.
- TCP/IP, fornito con il numero di programma concesso su licenza 5769–TC1.

Di seguito vengono elencati i requisiti della workstation:

- Un personal computer IBM (o compatibile) sul quale sia in esecuzione Windows NT 4.0, Service Pack 5 o Windows 2000, Service Pack 1.
- Una connessione TCP/IP al sistema AS/400.
- JDK 1.2.2 o successiva.

## Componenti del kernel

Dopo l'installazione, MQSeries Adapter Kernel si trova nella directory principale. Questa directory contiene sottodirectory, che, a loro volta, possono contenere altre directory. La directory principale e le sottodirectory vengono elencate insieme a un riepilogo dei file necessari per l'installazione e la configurazione.

**principale**

Il nome predefinito è `C:\Program Files\MQAK` sui sistemi Windows, `/usr/lpp/mqak` su AIX, `/MQAK` su HP-UX, `/opt/MQAK` su Solaris e `/QIBM/ProdData/mqak` su OS/400. Contiene i seguenti elementi:

- Tutte le altre directory di MQSeries Adapter Kernel.
- Il file `aqmsetenv.bat` (sistemi Windows) o `aqmsetenv.sh` (UNIX), che, se si desidera, modifica le variabili di ambiente dopo l'installazione.
- Il file `readme.txt`.
- I file `aqmuninstall.bat` (sistemi Windows) e `aqmuninstall.sh` (UNIX).

**`bin`** Contiene i seguenti elementi:

- Le librerie di classi e le librerie condivise.
- Gli adattatori forniti come parte del kernel, da utilizzare unicamente per eseguire la verifica.
- Il file `aqmversion.bat` (sistemi Windows) o `aqmversion.sh` (UNIX e OS/400) file, uno script che viene eseguito per visualizzare il numero di versione del kernel.
- Il file `aqmcrtmsg.bat` (sistemi Windows) o `aqmcrtmsg.sh` (UNIX e OS/400), uno script da eseguire per creare un file XML utilizzato per convalidare il file di configurazione prima di utilizzarlo.
- Il file `aqmsndmsg.bat` (sistemi Windows) o `aqmsndmsg.sh` (UNIX e OS/400), uno script da eseguire per convalidare il file di configurazione prima di utilizzarlo.
- Il file `aqmstrad.bat` (sistemi Windows) o `aqmstrad.sh` (UNIX e OS/400), uno script da eseguire per avviare il daemon dell'adattatore.

- Il file `aqmstrtd.bat` (sistemi Windows) o `aqmstrtd.sh` (UNIX e OS/400), uno script da eseguire per avviare il server di traccia.

**documentation**
Contiene la documentazione relativa al prodotto, incluso Information Center.

**runtimefiles**
Contiene i file di runtime del kernel.

**samples**

Contiene esempi di adattatori e di file di configurazioni e di utilità associati. Questi esempi contengono informazioni molto utili.

**Nota:** Il kernel è stato progettato per essere utilizzato con gli adattatori che l'utente crea mediante MQSeries Adapter Builder. Il kernel non è stato creato per essere utilizzato mediante chiamate alle API del kernel dal solo codice personalizzato. Gli esempi degli adattatori vengono forniti unicamente per facilitare la comprensione delle funzioni del kernel e agevolare le attività di diagnostica.

- Esempi di adattatori.
- Il file di installazione del kernel, `aqmsetup`, con i valori che supportano gli esempi degli adattatori. Per ulteriori informazioni su questo file, consultare "File di installazione" a pagina 64.
- Il file di configurazione del kernel, `aqmconfig.xml`, con i valori che supportano gli esempi di adattatori, compresi i valori di traccia di esempio. Per ulteriori informazioni su questo file, consultare "File di configurazione" a pagina 65.

**toolkit**
Contiene SDK che è formato da:

- File di intestazione.
- File della libreria utilizzati durante la compilazione sui sistemi Windows.

**uninstall**
Contiene i file utilizzati per disinstallare il kernel.

**verification**
Contiene i file riportati di seguito, che supportano la verifica dell'installazione del kernel:

- Il file `aqmverifyinstall.bat` (sistemi Windows) o `aqmverifyinstall.sh` (UNIX e OS/400), uno script da eseguire per verificare l'installazione del kernel su un computer.
- Il file `aqmcreateq.bat` (sistemi Windows) o `aqmcreateq.sh` (UNIX e OS/400), uno script che crea le code di MQSeries per la verifica. Consultare "Creazione di code di MQSeries" a pagina 97.
- Il file `aqmconfig.xml`. Per ulteriori informazioni su questo file, consultare "File di configurazione" a pagina 65.
- Il file `aqmsetup`. Per ulteriori informazioni su questo file, consultare "File di installazione" a pagina 64.
- Il file `aqminstalltest.xml`.

# Capitolo 3. Installazione del kernel

Questo capitolo descrive i passi necessari per l'installazione e la verifica di MQSeries Adapter Kernel. L'installazione consiste dei seguenti passi generali:

Passo 1. Eseguire le operazioni necessarie per l'installazione. Per dettagli, consultare "Preparazione all'installazione".

Passo 2. Installare il kernel. Per dettagli, consultare "Installazione del kernel" a pagina 41.

Passo 3. Completare i vari passi successivi all'installazione. Per dettagli, consultare "Completamento della post-installazione" a pagina 44.

Passo 4. Controllare che l'istallazione sia stata eseguita correttamente. Per dettagli, consultare "Verifica dell'installazione" a pagina 46.

Questo capitolo tratta anche i seguenti argomenti:

- Utilizzo dell'installazione non presidiata per l'installazione MQSeries Adapter Kernel. Per dettagli, consultare "Utilizzo dell'installazione non presidiata" a pagina 51.
- Aggiornamento di MQSeries Adapter Kernel da una versione precedente. Per dettagli, consultare "Aggiornamento del kernel" a pagina 53.
- Eliminazione dell'installazione di MQSeries Adapter Kernel. Per dettagli, consultare "Rimozione del kernel" a pagina 54.

Dopo l'installazione del kernel, eseguire le attività riportate di seguito per prepararlo all'utilizzo:

1. Configurare il kernel. Per dettagli, consultare "Configurazione del kernel" a pagina 58.
2. Configurare il software per la messaggistica e quello facoltativo. Per dettagli, consultare "Configurazione di MQSeries e di MQSeries Integrator" a pagina 91.
3. Creare gli adattatori utilizzando MQSeries Adapter Builder, quindi testarli e distribuirli.
4. Avviare il kernel. Per dettagli, consultare "Avvio del kernel" a pagina 92.

## Preparazione all'installazione

MQSeries Adapter Kernel può essere installato unicamente dal responsabile oppure da un utente con autorizzazioni root. Occorre disporre delle autorizzazioni necessarie per creare e accedere ai file nell'ubicazione in cui si installa MQSeries Adapter Kernel e alle ubicazioni dei due file di configurazione del kernel. La directory corrente deve essere compresa nel

percorso dell'eseguibile. Verificare che tutti gli ID utente che eseguono il kernel dispongano dei permessi di esecuzione, scrittura e lettura.

È necessario disporre di autorizzazioni appropriate per eseguire operazioni MQSeries come la creazione dei gestori code e la creazione e l'accesso alle code. L'esecuzione di queste operazioni varia a seconda delle piattaforme. Per ulteriori informazioni sulla piattaforma, consultare *MQSeries Administration Guide*.

L'ID utente che avvia i processi del kernel deve trovarsi nel gruppo mqm. Vi sono due tipi di processi del kernel:

- Daemon dell'adattatore, uno per ciascuna applicazione di destinazione servita dal computer
- Server di traccia (facoltativo)

L'adattatore di origine viene eseguito nel processo dell'applicazione di origine. I daemon o server contenenti l'adattatore di origine devono essere avviati affinché l'adattatore di origine venga eseguito.

È necessario installare e configurare il kernel per eseguire gli adattatori che sono stati creati. Tuttavia, non occorre installare il kernel per installare MQSeries Adapter Builder o per utilizzarlo per la creazione degli adattatori.

Eseguire i passi successivi prima di iniziare l'installazione:

- Leggere il file `readme.txt` disponibile sul CD-ROM o sulla LAN. Questo file potrebbe contenere importanti informazioni disponibili solo dopo il completamento di questo manuale. Questo file si trova nella directory di installazione principale.
- Visitare il sito Web di MQSeries all'indirizzo www.ibm.com/software/ts/mqseries/. Questo sito potrebbe contenere importanti informazioni disponibili alla pubblicazione di questo manuale o con una nuova edizione.
- Per eseguire un aggiornamento da una precedente versione di MQSeries Adapter Kernel, consultare la sezione "Aggiornamento del kernel" a pagina 53 per istruzioni.
- Assicurarsi che siano stati soddisfatti i prerequisiti hardware e software. Per ulteriori dettagli, consultare "Hardware" a pagina 31 e "Software" a pagina 32. Per verificare l'installazione di MQSeries Adapter Kernel, è necessario aver installato e avviato MQSeries. Assicurarsi che il supporto Java di MQSeries sia stato installato e configurato.

## Installazione del kernel

Per installare MQSeries Adapter Kernel su un sistema Windows (Windows NT o Windows 2000), su una piattaforma UNIX (AIX, HP-UX o Solaris) o su OS/400, eseguire i passi specifici per il sistema di seguito riportati:

**Sui sistemi Windows:**

Passo 1. Per avviare il programma di installazione, effettuare le seguenti operazioni:

- Per eseguire l'installazione da una LAN, modificare la directory che contiene i file di installazione di MQSeries Adapter Kernel ed eseguire il file `install.bat`.
- Per eseguire l'installazione dal CD-ROM, inserire il CD-ROM di MQSeries Adapter Kernel nell'apposita unità. Nel caso in cui la funzione di esecuzione automatica sia abilitata, il programma di installazione viene avviato automaticamente; in caso contrario, eseguire il file `install.bat` che si trova nella directory principale del CD-ROM per avviare il programma di installazione.

**Nota:** Sui sistemi Windows, non occorre copiare il file `install.bat` in un'ubicazione diversa prima di eseguirlo. Durante il processo di installazione, viene richiesto di scegliere dove installare MQSeries Adapter Kernel.

Passo 2. Seguire le richieste visualizzate dal programma di installazione. Tuttavia, se si sceglie di installare MQSeries Adapter Kernel in un'ubicazione diversa da quella predefinita (`C:\Programmi\MQAK`), è necessario specificare il nome completo della directory di installazione, non il nome di percorso relativo.

**Su UNIX:**

Passo 1. Per avviare il programma di installazione, effettuare le seguenti operazioni:

- Per eseguire l'installazione da una LAN, modificare la directory che contiene i file di installazione di MQSeries Adapter Kernel ed eseguire lo script `install.sh`.
- Per eseguire l'installazione dal CD-ROM, inserire il CD-ROM di MQSeries Adapter Kernel nell'apposita unità e, se necessario, montare l'unità CD-ROM seguendo le istruzioni fornite nella documentazione relativa al sistema operativo. Eseguire lo script `install.sh` nella directory principale del CD-ROM.

Passo 2. Seguire le richieste visualizzate dal programma di installazione. Si noti che se si sceglie di installare MQSeries Adapter Kernel in un'ubicazione diversa da quella predefinita, è necessario specificare la directory di installazione come nome di percorso completo e non

come nome di percorso relativo. Le directory di installazione predefinite in UNIX sono le seguenti:

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

**Su OS/400:**

Passo 1. Controllare che tutti i prerequisiti elencati in “Hardware” a pagina 31, “prerequisiti software OS/400” a pagina 33 e “Prerequisiti per l’installazione di OS/400” a pagina 34 siano stati soddisfatti. L’installazione di MQSeries Adapter Kernel su un sistema OS/400 prevede l’utilizzo di un programma basato su InstallShield, che richiede l’impiego di una workstation che si interfaccia con il sistema AS/400; per ulteriori dettagli, consultare “Prerequisiti per l’installazione di OS/400” a pagina 34.

Passo 2. Creare un profilo utente denominato MQAKSRV sul sistema AS/400 utilizzando il comando **CRTUSRPRF** in un prompt CL (Control Language).

Passo 3. Di seguito vengono riportate due diverse procedure di installazione, a seconda che si utilizzi un AWT remoto oppure una workstation client collegata:

- Se l’installazione viene eseguita utilizzando l’AWT remoto, effettuare le seguenti operazioni:
  a. Verificare che l’AWT remoto sia stato installato e che sia in esecuzione. Per dettagli, consultare “Utilizzo dell’AWT remoto” a pagina 34.
  b. Assicurarsi che il file `installAS400.jar` sia accessibile sul sistema AS/400. Questo file può trovarsi sul file system integrato (IFS, Integrated File System) oppure su un’unità collegata al sistema AS/400. Se il file si trova sull’unità collegata, utilizzare il comando Create Link (**CRTLINK**) per creare un collegamento simbolico al file.
  c. Per migliorare il livello delle prestazioni del processo di installazione, eseguire il comando Create Java Program (**CRTJVAPGM**) nel file `installAS400.jar`.
  d. Eseguire il comando Run Java (**RUNJVA**) come riportato di seguito, dove *n.n.n.n* rappresenta l’indirizzo TCP/IP della workstation sulla quale è in esecuzione l’AWT remoto:

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

- Se l'installazione viene eseguita utilizzando una workstation client collegata, effettuare le seguenti operazioni:

  Passo a. Assicurarsi che siano stati soddisfatti i requisiti specificati in "Utilizzo di un client collegato" a pagina 35.

  Passo b. Accertarsi che l'opzione Host Servers sia installata e che sia in esecuzione sul computer AS/400. E' possibile avviare Host Servers utilizzando il comando di avvio di Host Servers (**STRHOSTSVR**) in un prompt CL.

  Passo c. Accertarsi che il protocollo TCP/IP sia installato e in esecuzione sul computer AS/400. E' possibile avviare TCP/IP utilizzando il comando Start TCP/IP (**STRTCP**) dal prompt CL.

  Passo d. Sulla workstation, aprire un prompt dei comandi e modificare la directory `AS400` del supporto di installazione MQSeries Adapter Kernel (una LAN o un CD-ROM).

  Passo e. Immettere il seguente comando:

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

Passo 4. Il programma di installazione viene avviato e visualizza il pannello **Signon to AS/400**. Immettere l'indirizzo TCP/IP del computer AS/400 nel campo **System:** e specificare l'ID e la password utente nei campi corrispondenti. Non selezionare la casella **Default User**. Fare clic su **Next**.

Passo 5. Seguire le richieste visualizzate dal programma di installazione. A seconda della velocità della rete e dei computer, il processo di installazione può richiedere anche un'ora. Sulla workstation viene visualizzata una barra di stato che indica lo stato dell'installazione.

Su OS/400, MQSeries Adapter Kernel viene installato sempre nella directory `/QIBM/ProdData/mqak` della root del file system integrato.

Passo 6. Impostare le variabili di ambiente CLASSPATH, PATH, e QIBM_MULTI_THREADED nel modo seguente:

- Aggiungere la directory `/QIBM/ProdData/mqak/bin` alla variabile di ambiente CLASSPATH.
- Aggiungere la directory `/QIBM/ProdData/mqak/bin` alla variabile di ambiente PATH.
- Impostare la variabile di ambiente QIBM_MULTI_THREADED su `Y`.

Passo 7. Aggiungere la libreria MQAK all'elenco delle librerie QSYS.LIB.

L'installazione del kernel è completa. Non appena installato, il kernel è configurato per supportare la verifica ma non la produzione. Verificare l'installazione eseguendo la procedura riportata in "Verifica dell'installazione" a pagina 46

a pagina 46. Una volta verificata l'installazione, seguire i passi riportati in "Completamento della post-installazione" per impostare le variabili di ambiente e spostare i file di configurazione per supportare la produzione nel proprio sito.

Installare il kernel su altri computer come richiesto.

## Completamento della post-installazione

Dopo l'installazione del kernel, procedere nel modo seguente:

Passo 1. Decidere dove collocare i file `aqmsetup` e `aqmconfig.xml`, utilizzati per configurare il kernel. Per ulteriori informazioni su questi file, consultare "Configurazione del kernel" a pagina 58.

**ATTENZIONE:**
**Se non si creano file di configurazione propri ma si utilizzano i file di configurazione forniti nella directory `samples`, l'installazione di una nuova versione del kernel li sovrascrive ed elimina la configurazione di produzione.**

Passo 2. Creare una directory per i due file di configurazione. Benché non sia obbligatorio, è consigliabile che i due file si trovino nella stessa directory. Posizionandoli esternamente alla directory in cui è installato MQSeries Adapter Kernel, la disinstallazione del kernel implica un numero minore di directory. Il processo di disinstallazione ignora le directory che non contengono i file di origine di MQSeries Adapter Kernel.

Passo 3. Copiare i file `aqmsetup` e `aqmconfig.xml` dalla directory `samples` nell'ubicazione desiderata. E' possibile inserirli su un'unità di rete diversa dall'ubicazione centrale, alla quale sia possibile accedere da più computer per agevolarne l'aggiornamento e il backup.

Se il file `aqmconfig.xml` viene ridenominato, il kernel non funziona correttamente. E' possibile ridenominare il file `aqmsetup`: la variabile di ambiente deve fare riferimento in modo corretto ad esso nel Passo 5 a pagina 45.

Passo 4. Utilizzando un editor di testo, modificare il file `aqmsetup` in modo che richiami la directory desiderata del file `aqmconfig.xml`. Utilizzare un nome percorso completo (non relativo) per la directory. Non specificare in questo percorso il nome del file. Ad esempio:

```
# Percorso del file di configurazione aqmconfig.xml.
AQMCONFIG=C:\Programmi\MQAK\Data\
```

Anche se il percorso che si desidera specificare per il file aqmconfig.xml è la stessa directory in cui si trova il file aqmsetup, è necessario utilizzare un nome percorso completo. Salvare e chiudere il file aqmsetup.

Passo 5. Impostare la variabile di ambiente AQMSETUPFILE in modo che faccia riferimento all'ubicazione del file aqmsetup (ad esempio, C:\Program Files\MQAK\Data\aqmsetup sui sistemi Windows, /MQAK/data/aqmsetup su UNIX o /home/*user_name*/aqmsetup su OS/400). Si noti che su OS/400, il file aqmsetup deve essere sempre ubicato nella directory IFS principale dell'utente corrente (cioè /home/*user_name*).

Se il kernel viene installato su un'unità di rete, eseguire questa procedura per ogni computer che vi accede.

Passo 6. Se si utilizza AIX e si intende utilizzare gli adattatori di origine nativi del linguaggio C richiamati da un programma C, impostare la variabile di ambiente AIXTHREAD_SCOPE sul valore S. Per impostare questa variabile di ambiente nella shell Bourne o Korn, immettere il seguente comando:

```
export AIXTHREAD_SCOPE=S
```

Per impostare questa variabile nella shell di C, utilizzare il seguente comando:

```
setenv AIXTHREAD_SCOPE S
```

Per impostare la variabile AIXTHREAD_SCOPE automaticamente al collegamento in AIX, aggiungere questo comando al file .profile (se si utilizza la shell Bourne o Korn) o al file .cshrc (se si utilizza la shell C).

Per ulteriori informazioni sui criteri di pianificazione, fare riferimento a 7 a pagina 23.

Passo 7. Se necessario, impostare la variabile di ambiente THREADS_FLAG. Questa variabile deve essere impostata solo se *tutte* le condizioni riportate di seguito sono vere:

- Solaris è il sistema operativo in uso.
- Si utilizza la JDK (Java Development Kit) versione 1.2.2.
- MQSeries viene utilizzato per trasportare i messaggi.
- Gli adattatori di origine e di destinazione sono scritti in C.

Solo nel caso in cui tutte queste condizioni siano vere, impostare la variabile di ambiente THREADS_FLAG su native. Per impostare questa variabile di ambiente nella shell Bourne o Korn, immettere il seguente comando:

```
export THREADS_FLAG=native
```

Per impostare questa variabile nella shell di C, utilizzare il seguente comando:

```
setenv THREADS_FLAG native
```

Per impostare la variabile THREADS_FLAG al collegamento in Solaris, aggiungere questo comando al file `.profile` (se si utilizza la shell Bourne o Korn) o al file `.cshrc` (se si utilizza la shell C).

Una volta completati i passi di post-installazione, eseguire le attività riportate di seguito per preparare il kernel all'utilizzo:

1. Eseguire le operazioni necessarie per la produzione. Consultare "Preparazione alla produzione" a pagina 57.
2. Editare il file di configurazione. Per dettagli, consultare "Configurazione del kernel" a pagina 58.
3. Configurare MQSeries e il software facoltativo. Consultare "Configurazione di MQSeries e di MQSeries Integrator" a pagina 91.
4. Per quanto concerne i sistemi di produzione, consultare la sezione "Consigli sulle prestazioni" a pagina 92.
5. Avviare il kernel. Consultare "Avvio del kernel" a pagina 92.
6. Impostare un piano per la gestione del kernel. Consultare "Gestione del kernel" a pagina 94,

## Verifica dell'installazione

Dopo aver installato il kernel, controllare che l'installazione sia stata eseguita correttamente: a tale scopo, eseguire lo script di verifica. Lo script invia un messaggio di verifica da una applicazione di origine utilizzando un adattatore di origine e poi a MQSeries utilizzando il kernel. Successivamente, si serve del kernel per ricevere il messaggio da MQSeries, quindi richiama un'applicazione di destinazione. Tutti questi processi vengono eseguiti su un singolo computer.

In questa verifica, l'applicazione di origine è una coda di MQSeries denominata TEST1, mentre l'applicazione di destinazione è la coda di MQSeries, TEST2.

La verifica prevede l'esecuzione delle seguenti attività:

- Si controlla che il kernel, insieme agli adattatori di origine e di destinazione, abbiano eseguito correttamente il marshaling e l'instradamento del messaggio di prova, utilizzando MQSeries come software per la messaggistica, all'interno del computer.

- Si controlla che i file `aqmconfig.xml` e `aqmsetup` siano stati forniti al momento dell'installazione. Questi file determinano la configurazione del kernel. Per ulteriori informazioni su questi file, consultare "Configurazione del kernel" a pagina 58.

E' possibile convalidare il file di configurazione prima di utilizzarlo. Consultare "Convalida del file di configurazione" a pagina 88.

Per utilizzare gli script di verifica dell'installazione forniti con MQSeries Adapter Kernel, è necessario che MQSeries sia stato installato e configurato sullo stesso computer sul quale eseguire gli script. Nel caso in cui si utilizzi un software per la messaggistica diverso da MQSeries, è possibile modificare questi script in modo che possano supportare il software utilizzato. Per eseguire questa modifica, completare la seguente procedura:

1. Modificare la directory `verification` dell'installazione del kernel.
2. Aprire il file `aqmconfig.xml` con un editor di testo e modificare la riga `<epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>` in `<epicmqppqueuemgr>`*`nome_gestore_code`*`</epicmqppqueuemgr>`, dove *nome_gestore_code* è il nome del gestore code utilizzato.
3. Modificare il file `aqmverifyinstall` come indicato di seguito:
   - Se si esegue una verifica dell'installazione su un sistema Windows, aprire il file `aqmverifyinstall.bat` con un editor di testo e modificare la riga `aqmcreateq TEST2` in `aqmcreateq TEST2` *`nome_gestore_code`*, dove *nome_gestore_code* è il nome del gestore code utilizzato.
   - Se la verifica dell'installazione viene eseguita su un sistema UNIX o OS/400, aprire il file `aqmverifyinstall.sh` con un editor di testo e modificare la riga `aqmcreateq.sh TEST2` in `aqmcreateq.sh TEST2` *`nome_gestore_code`*, dove *nome_gestore_code* è il nome del gestore code utilizzato.

Questa verifica utilizza alcuni componenti, quali il nome dell'adattatore di destinazione `com.ibm.epic.adapters.eak.test.InstallVerificationTest`, che non fanno parte del kernel. Questi componenti vengono forniti insieme al kernel solo per consentire l'esecuzione della verifica dell'installazione.

Una volta terminata la verifica, il daemon dell'adattatore di verifica viene arrestato.

La creazione di tracce non è abilitata durante la verifica.

## Procedura della verifica

Passo 1. La verifica crea e utilizza tre code di MQSeries. Se, prima di avviare la verifica, queste code contengono messaggi, non è possibile eseguire la verifica. Pertanto, è necessario cancellare tutti i messaggi contenuti nelle tre code elencate di seguito:

- TEST2AIQ
- TEST2AEQ
- TEST2RPL

Passo 2. Verificare di disporre dell'autorizzazione per installare e verificare il kernel. Consultare "Preparazione all'installazione" a pagina 39,

Passo 3. Avviare la verifica come indicato di seguito:

- Sui sistemi Windows, fare doppio clic sul file `aqmverifyinstall.bat` contenuto nella directory `verification`. In alternativa, aprire il prompt dei comandi, passare alla directory `verification` ed eseguire `aqmverifyinstall.bat`.
- Su UNIX, aprire un terminale, visualizzare la directory `verification` ed eseguire il file `aqmverifyinstall.sh`.
- Su OS/400, completare la procedura riportata di seguito:
  a. Avviare una sessione **qsh** immettendo il comando **STRQSH**.
  b. Copiare il file `/QIBM/ProdData/mqak/verification/aqmsetup` nella directory principale (`/home/`*`user_name`*).
  c. Visualizzare la directory `/QIBM/ProdData/mqak/verification`.
  d. Eseguire il file `aqmverifyinstall.sh`.

Il file `aqmverifyinstall` contiene commenti relativi al suo funzionamento.

Passo 4. Il messaggio `Prova per la verifica dell'installazione completata correttamente` indica che l'esito della verifica è positivo. Chiudere la finestra di verifica, se necessario.

Passo 5. Nel caso in cui la verifica abbia avuto un esito negativo, esaminare la finestra di verifica e il file di registrazione `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, per individuare l'errore.

Passo 6. Per informazioni sui problemi che possono verificarsi durante una verifica dell'installazione ed eventuali soluzioni, consultare "Problemi comuni di verifica".

Passo 7. Se si desidera, eseguire una verifica facoltativa. Per dettagli, consultare "Verifica facoltativa" a pagina 51.

Passo 8. Ritornare alla procedura di installazione e configurare il kernel per supportare l'operazione sulla propria postazione di lavoro. Saltare al passo 1 a pagina 44.

## Problemi comuni di verifica

Questa sezione descrive i problemi più frequenti relativi alla procedura di verifica, offrendone una possibile soluzione. Le informazioni più importanti contenute nei messaggi di eccezione, sono riportate in **grassetto**.

**Problema**: Il file `aqmsetup` non è stato trovato.

**Soluzione**: Assicurarsi che la variabile di ambiente AQMSETUPFILE sia impostata sul percorso del file aqmsetup nella directory verification.

**Messaggio di eccezione**:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: rilevato
throwable con messaggio <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterDirectory::getProperties():
Ricevuta eccezione <com.ibm.epic.adapters.eak.common.AdapterException>
Informazioni messaggio:

<AQM0002: com.ibm.epic.adapters.eak.common.
AdapterCfg::readConfig(String):
Ricevuta eccezione <java.io.FileNotFoundException>
Informazioni messaggio:

<C:\aqmsetup> Informazioni di programma aggiuntive <>.>
Informazioni di programma aggiuntive


<Errore durante la lettura del file di configurazione
[è possibile che il file o le chiavi in esso contenute non esistano]>.>
```

**Problema**: Il file aqmconfig.xml non è stato trovato.

**Soluzione**: Aprire il file aqmsetup nella directory verification e controllare che la voce AQMCONFIG= punti alla directory verification. Utilizzare un nome percorso completo. Verificare anche che il file aqmconfig.xml sia ubicato nella directory verification.

**Messaggio di eccezione**:

```
com.ibm.epic.adapters.eak.common.AdapterException: ID messaggio <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.common.AdapterDirectory::
getProperties(): Ricevuta eccezione
<java.io.FileNotFoundException> Informazioni messaggio:
<AQMCONFIG.xml> Informazioni di programma aggiuntive <>.>
```

**Problema**: La coda in cui inserire il messaggio, non esiste.

**Soluzione**: Utilizzare MQSeries per controllare che la coda riportata nel messaggio di eccezione (TEST2AIQ al momento della verifica) esista e che possa accettare messaggi. Consultare “Creazione di code di MQSeries” a pagina 97.

**Messaggio di eccezione**:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: rilevato
throwable con messaggio
<AQM0107: com.ibm.epic.adapters.eak.nativeadapter.LMSMQbase::
createMQOutputQueue(String):
Ricevuta MQException durante la creazione di una coda,
nome QManager <DEFAULT>
Nome coda <TEST2AIQ>:
codice di completamento <2> codice di errore <2085>.>
```

**Problema**: L'adattatore di destinazione non è stato trovato.

**Soluzione**: Assicurarsi che l'adattatore di destinazione specificato esiste: `com.ibm.epic.adapters.eak.test.InstallVerificationTest`. Verificare che la variabile di ambiente CLASSPATH includa la directory `bin` del kernel.

**Messaggio di eccezione**:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2:
Messaggio <<TEST2> <2000.05.18.09.41.43.781> <<Elaborazione messaggi.>
<com.ibm.epic.adapters.eak.common.AdapterException: ID messaggio <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker:
:instantiateClass(String, Class[], Object[]): Ricevuta eccezione
<java.lang.ClassNotFoundException> Informazioni messaggio:
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>
Informazioni di programma aggiuntive
<[Impossibile richiamare la classe per il nome della classe
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>]>.>>>>
```

**Problema**: Non è stato trovato alcun adattatore da poter caricare per consegnare il messaggio. L'identificativo logico di destinazione non presenta alcuna voce nel file `aqmconfig.xml` per il tipo e la categoria di testo specificati nel messaggio nella coda.

**Soluzione**: è possibile che, prima della verifica, la coda TEST2AIQ contenesse messaggi. Cancellare tutti i messaggi dalla coda TEST2AIQ ed eseguire di nuovo la verifica. L'unica voce per il nome della classe comando per l'applicazione TEST2 nel file `aqmconfig.xml` nella directory `verification` è relativo al tipo di testo di TESTBOD e alla categoria di testo di OAG.

**Messaggio di eccezione**:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2: Messaggio <<TEST2> <2000.05.18.10.28.43.105>
<<Elaborazione messaggi.> <com.ibm.epic.adapters.eak.common.
AdapterException:
ID messaggio <AQM0401> <AQM0401: com.ibm.epic.adapters.eak.
adapterdaemon.EpicAdapterWorker::processMessage(EpicMessage):

Impossibile determinare il nome della classe comando
da caricare per il messaggio ricevuto.>>>>
```

**Problema**: Il gestore code della verifica non è stato avviato.

**Soluzione**: Controllare che il gestore code di MQSeries predefinito sia stato avviato correttamente.

**Messaggio di eccezione**:

```
com.ibm.epic.adapters.eak.common.AdapterException: ID messaggio <AQM0104>
<AQM0104: com.ibm.epic.adapters.eak.nativeAdapter.queueCollection::
constructor(String,String,boolean,String,String,int):
Ricevuta MQException durante la creazione di una connessione QManager per
nome QManager <QMGRNAME>
Informazioni messaggio MQ:

codice di completamento <2> codice di errore <2059>.>
```

**Problema**: Si è verificato un errore di MQSeries.

**Soluzione**: Controllare che MQSeries sia stato installato e configurato correttamente e che sia in esecuzione sul computer. Esaminare il codice di errore di MQException e utilizzare il documento *Messaggi MQSeries* per stabilire le cause di questo errore.

**Messaggio di eccezione**:

```
Ricevuta MQException "TENTATIVO DI ESECUZIONE AZIONE."
Informazioni messaggio:

codice di completamento <codice_di_completamento>


codice di errore <codice_di_errore>
```

### Verifica facoltativa

Dopo aver controllato che il kernel sia stato installato correttamente sul primo computer, se si desidera, è possibile eseguire una verifica facoltativa, completando la seguente procedura:

1. Controllare che il kernel sia stato installato correttamente su un secondo computer, eseguendo la stessa verifica.
2. Assicurarsi che sia possibile inviare un messaggio di prova da un adattatore di origine su un computer a un adattatore di destinazione su un altro computer. Configurare manualmente ed eseguire questa verifica. Nel caso in cui si scelga di sviluppare questa verifica modificando i file di verifica originali forniti con il kernel, conservare una copia di questi file per il backup.

## Utilizzo dell'installazione non presidiata

MQSeries Adapter Kernel può essere installato su tutte le piattaforme utilizzando l'*installazione non presidiata*. L'installazione non presidiata consente di evitare il programma di installazione di MQSeries Adapter Kernel, in cui è necessario selezionare manualmente le opzioni desiderate. L'installazione non presidiata è utile se si desidera installare la configurazione predefinita su più macchine.

Per installare il kernel in modo non presidiato, eseguire i passi specifici del sistema operativo riportati di seguito:

**Sui sistemi Windows:**

Passo 1. Aprire un prompt dei comandi e passare alla directory che contiene i file di installazione di MQSeries Adapter Kernel.

Passo 2. Immettere il seguente comando:

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

dove *ubicazione_installazione* è l'ubicazione di installazione desiderata (ad esempio, `D:\mqak`).

**Su UNIX:**

Passo 1. In un terminale, passare alla directory che contiene i file di installazione di MQSeries Adapter Kernel. Per eseguire l'installazione dal CD-ROM, inserire il CD-ROM di MQSeries Adapter Kernel nell'apposita unità e, se necessario, montare l'unità CD-ROM seguendo le istruzioni fornite nella documentazione relativa al sistema operativo.

Passo 2. Immettere il seguente comando:

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

dove *ubicazione_installazione* è l'ubicazione di installazione desiderata) ad esempio, `/opt/mqak`).

**Su OS/400:**

Se non si utilizza un client collegato per accedere alla macchina AS/400, procedere nel modo seguente:

Passo 1. Assicurarsi che il file `installAS400.jar` sia accessibile sul sistema AS/400. Questo file può trovarsi sul file system integrato (IFS, Integrated File System) oppure su un'unità collegata al sistema AS/400. Se il file si trova sull'unità collegata, utilizzare il comando Create Link (**CRTLINK**) per creare un collegamento simbolico al file.

Passo 2. Per migliorare il livello delle prestazioni del processo di installazione, eseguire il comando Create Java Program (**CRTJVAPGM**) nel file `installAS400.jar`.

Passo 3. Se si utilizza un prompt CL o una sessione **qsh**, immettere uno dei seguenti comandi:

- Se si utilizza un prompt CL, immettere il seguente comando:

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

- Se si utilizza una sessione **qsh**, immettere il seguente comando:

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

Se si utilizza un client collegato per interfacciare con un sistema AS/400, procedere nel modo seguente:

Passo 1. Assicurarsi che siano stati soddisfatti i requisiti specificati in "Utilizzo di un client collegato" a pagina 35.

Passo 2. Accertarsi che l'opzione Host Servers sia installata e che sia in esecuzione sul computer AS/400. E' possibile avviare Host Servers utilizzando il comando Start Host Servers (**STRHOSTSVR**) dal prompt CL.

Passo 3. Accertarsi che il protocollo TCP/IP sia installato e in esecuzione sul computer AS/400. E' possibile avviare TCP/IP utilizzando il comando Start TCP/IP (**STRTCP**) dal prompt CL.

Passo 4. Sulla workstation, aprire un prompt dei comandi e modificare la directory `AS400` del supporto di installazione MQSeries Adapter Kernel (una LAN o un CD-ROM).

Passo 5. Immettere il seguente comando:

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

dove *machine_name* è l'indirizzo TCP/IP del sistema AS/400, *user_ID* è l'ID utente e *password* è la password.

## Aggiornamento del kernel

Se è stato installato MQSeries Adapter Kernel versione 1.0, con o senza CSD (Corrective Service Diskette) o MQSeries Adapter Kernel versione 1.1 con un livello di modifica precedente, eseguire i successivi passi prima di installare MQSeries Adapter Kernel versione 1.1 con il corrente livello di modifica:

Passo 1. Eseguire il back up dei file `aqmsetup` e `aqmconfig` (`aqmconfig.properties` o `aqmconfig.xml`) in un'ubicazione esterna alla directory di installazione di MQSeries Adapter Kernel.

Passo 2. Se MQSeries Adapter Kernel CSD è installato, disinstallarlo nel modo seguente:

- Su Windows NT, utilizzare uno dei seguenti metodi:
  - Dal menu Start, fare clic su **Programmi > MQSeries Adapter Kernel > Rimuovi CSD**.
  - Dal Pannello di controllo, eseguire l'utilità Installazione applicazioni.
  - Eseguire il file `aqmuninstallCSD.bat` nella directory principale del kernel.
  - Aprire un prompt dei comandi, visualizzare la directory principale del kernel e digitare il seguente comando:

```
java uninstallCSD
```

- Su AIX, visualizzare la directory principale del kernel e immettere uno dei seguenti comandi:

  ```
  aqmuninstallCSD.sh
  java uninstallCSD
  ```

Passo 3. Disinstallare MQSeries Adapter Kernel nel modo seguente:

- Su Windows NT, utilizzare uno dei seguenti metodi:
  - Dal menu Start, fare clic su **Programmi > MQSeries Adapter Kernel > Uninstall MQSeries Adapter Kernel**.
  - Dal Pannello di controllo, eseguire l'utilità Installazione applicazioni.
  - Eseguire il file `aqmuninstall.bat` nella directory root del kernel.
  - Aprire un prompt dei comandi, visualizzare la directory principale del kernel e digitare il seguente comando:

    ```
    java uninstall
    ```
- Su AIX, visualizzare la directory principale del kernel e immettere uno dei seguenti comandi:

  ```
  aqmuninstall.sh
  java uninstall
  ```

Passo 4. Installare MQSeries Adapter Kernel versione 1.1. Per dettagli, consultare "Installazione del kernel" a pagina 41.

Passo 5. Ripristinare i file `aqmsetup` e `aqmconfig` nelle precedenti ubicazioni nella directory di installazione di MQSeries Adapter Kernel. Se necessario, convertire il file `aqmconfig.properties` in un file `aqmconfig.xml`. Per ulteriori informazioni sul file `aqmconfig.xml`, consultare "File di configurazione" a pagina 65.

## Rimozione del kernel

E' possibile rimuovere un kernel in vari modi. Si noti che il processo di disinstallazione non elimina le directory e i file creati dopo l'installazione del kernel. in particolare, i file di registrazione e di dati copiati dall'utente.

- Sui sistemi Windows, utilizzare uno dei seguenti metodi:
  - Dal menu Start, fare clic su **Programmi > IBM MQSeries Adapter Kernel > Disinstalla MQSeries Adapter Kernel**.
  - Dal Pannello di controllo, eseguire l'utilità Installazione applicazioni.
  - Eseguire il file `aqmuninstall.bat` nella directory root del kernel.

– Per disinstallare il kernel in modalità non presidiata (cioè, senza che il programma di disinstallazione richieda dettagli o conferme), aprire un prompt dei comandi, passare alla directory di installazione del kernel ed immettere il seguente comando:

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

- Su UNIX, visualizzare la directory principale del kernel e immettere il seguente comando:

```
aqmuninstall.sh
```

Per disinstallare il kernel in modalità non presidiata (cioè, senza che il programma di disinstallazione richieda dettagli o conferme), passare alla directory root del kernel ed immettere il seguente comando:

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

- Su OS/400, utilizzare i metodi seguenti per disinstallare il kernel.
  – Se si utilizza AWT remoto per disinstallare il kernel, eseguire i passi riportati di seguito:

    Passo 1. Verificare che l'AWT remoto sia stato installato e che sia in esecuzione. Per dettagli, consultare "Utilizzo dell'AWT remoto" a pagina 34.

    Passo 2. Per migliorare il livello delle prestazioni del processo di disinstallazione, eseguire il comando Create Java Program (**CRTJVAPGM**) per il file `/QIBM/ProdData/mqak/uninstall/uninstall.jar`.

    Passo 3. Eseguire il comando Run Java (**RUNJVA**) come riportato di seguito, dove *n.n.n.n* rappresenta l'indirizzo TCP/IP della workstation sulla quale è in esecuzione l'AWT remoto:

```
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))
```

  – Se si utilizza una workstation client collegata per disinstallare il kernel, procedere nel modo seguente:

    Passo 1. Assicurarsi che siano stati soddisfatti i requisiti specificati in "Utilizzo di un client collegato" a pagina 35.

    Passo 2. Accertarsi che l'opzione Host Servers sia installata e che sia in esecuzione sul computer AS/400. E' possibile avviare Host Servers utilizzando il comando Start Host Servers (**STRHOSTSVR**) dal prompt CL.

    Passo 3. Accertarsi che il protocollo TCP/IP sia installato e in esecuzione sul computer AS/400. E' possibile avviare TCP/IP utilizzando il comando Start TCP/IP (**STRTCP**) dal prompt CL.

Passo 4. Copiare i file `uninstall.jar` e `uninstall.dat` dalla directory `/QIBM/ProdData/mqak/uninstall` sul sistema AS/400 in una directory sulla workstation client.

Passo 5. Immettere il seguente comando:

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

Per disinstallare il kernel in modalità non presidiata (cioè, senza che il programma di disinstallazione richieda dettagli o conferme), immettere il seguente comando:

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

dove *machine_name* è l'indirizzo TCP/IP del sistema AS/400, *user_ID* è l'ID utente e *password* è la password.

– Se si utilizza direttamente un sistema AS/400 in un prompt CL o una sessione **qsh** e si desidera disinstallare il kernel in modalità non presidiata (cioè, senza che il programma di disinstallazione richieda dettagli o conferme) immettere uno dei seguenti comandi:

  - In un prompt CL:

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

  - In una sessione **qsh**:

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

# Capitolo 4. Utilizzo del kernel

Questo capitolo contiene le sezioni elencate di seguito, che forniscono informazioni sulle modalità di utilizzo del kernel:



## Preparazione alla produzione

Prima di attivare il kernel, completare le attività riportate di seguito:

1. In base alle esigenze e alle condizioni specifiche, progettare l'architettura del sistema, compreso MQSeries Adapter Offering, MQSeries o un altro software per la messaggistica e, facoltativamente, MQSeries Integrator. In genere, l'architettura è unica per ciascuna stazione di lavoro.
2. Creare gli adattatori di origine e destinazione richiesti utilizzando MQSeries Adapter Builder, testarli e poi svilupparli.
3. Sviluppare interfacce specifiche dell'applicazione al di fuori di MQSeries Adapter Offering per:
   - Consentire all'adattatore di origine di acquisire i dati di applicazione dall'applicazione di origine
   - Consentire all'applicazione di destinazione di acquisire i dati dei messaggi dall'adattatore di destinazione

   L'esatta natura di un'interfaccia specifica dell'applicazione dipende dalle caratteristiche dell'applicazione di origine e da quella di destinazione. Di seguito vengono riportati alcuni esempi di interfacce specifiche dell'applicazione:
   - Chiamate API e uscite utente
   - Letture e scritture di file
   - Trigger di database
   - Code messaggi

4. Configurare il kernel per il supporto del flusso di runtime: invio, instradamento, creazione di traccia e consegna dei messaggi. Per informazioni sulla configurazione del kernel, consultare "Configurazione del kernel".
5. Configurare MQSeries o il software per la messaggistica utilizzato e, facoltativamente, MQSeries Integrator per il supporto dell'intera architettura del sistema. Consultare "Configurazione di MQSeries e di MQSeries Integrator" a pagina 91.
6. Se richiesto, sviluppare classi di collegamento Java per supportare la consegna dei messaggi. Queste classi sono specifiche per ciascuna applicazione di destinazione e devono essere utilizzate solo se l'adattatore di destinazione richiede informazioni per l'accesso e il collegamento all'applicazione.
7. Prima di attivarlo, eseguire una verifica dell'intero sistema, ossia, MQSeries Adapter Kernel e gli adattatori di origine e di destinazione, le interfacce specifiche dell'applicazione e il codice personalizzato.
8. Distribuire il sistema nell'ambiente di produzione.
9. Attivare il kernel avviando uno o più daemon di adattatori e facoltativamente i server di traccia. Accertarsi che l'applicazione di origine sia stata avviata. Se l'adattatore di origine viene eseguito nel processo dell'applicazione di origine, l'adattatore di origine viene avviato automaticamente con l'applicazione di origine; non sono richieste operazioni aggiuntive per avviare l'adattatore di origine. È necessario avviare tutti i daemon o i server che contengono l'adattatore di origine. Consultare "Avvio del kernel" a pagina 92.

## Configurazione del kernel

Questa sezione descrive la configurazione del kernel per l'utilizzo in uno specifico ambiente. "Panoramica sulla configurazione" fornisce una panoramica concettuale della configurazione del kernel. "File per l'avvio e la configurazione" a pagina 63 descrive i vari file che insieme definiscono una configurazione MQSeries Adapter Kernel. "File di installazione" a pagina 64 presenta il file `aqmsetup` che definisce la maggior parte delle impostazioni iniziali del kernel. "File di configurazione" a pagina 65 descrive il file `aqmconfig.xml` che fornisce al kernel informazioni di configurazione specifiche, quali il nome dell'applicazione di origine e di destinazione, gli adattatori di origine e di destinazione, code e programmi di gestione code, modalità di comunicazione e specifiche di traccia e di registrazione.

### Panoramica sulla configurazione

Questa sezione fornisce una panoramica concettuale sulla configurazione del kernel. E' importante conoscere il flusso di esecuzione del kernel prima di eseguirne la configurazione. Questa sezione descrive il flusso di esecuzione ad

un livello semplificato. Consultare “Flusso di runtime” a pagina 14 per informazioni dettagliate sul flusso di esecuzione.

Al livello di base, la configurazione di MQSeries Adapter Kernel è determinata dai dati scambiati tra le applicazioni. La configurazione deve considerare anche i seguenti fattori:

- Le applicazioni che ricevono i dati.
- Gli adattatori richiesti sul lato dell'origine e gli adattatori di destinazione, i daemon dell'adattatore e i componenti necessari sul lato di destinazione.
- Le modalità di comunicazione, i formati e i meccanismi di trasporto utilizzati.

La struttura e il formato dei dati sono differenti per ciascuna applicazione nella configurazione. Ad esempio, se una configurazione include due applicazioni, A e B, ognuna delle quali invia ordini di acquisto all'applicazione C, i dati inviati dall'applicazione A presentano un formato differente e tag con diverso significato dai dati dell'applicazione B. Per evitare che l'applicazione C riconosca ed analizzi i due differenti flussi di dati provenienti da due differenti applicazioni, i dati di ciascuna applicazione sono convertiti in un *messaggio di integrazione* in *formato integration-neutral*. Solitamente il formato integration-neutral è uno standard industriale basato su XML. L'unico formato di dati necessario all'applicazione C per poter eseguire il riconoscimento e l'analisi è il formato integration-neutral.

Figura 3 a pagina 60 illustra il flusso dei dati dalle applicazioni A e B alle applicazioni C e D. La figura viene seguita da una spiegazione dei vari flussi di dati.

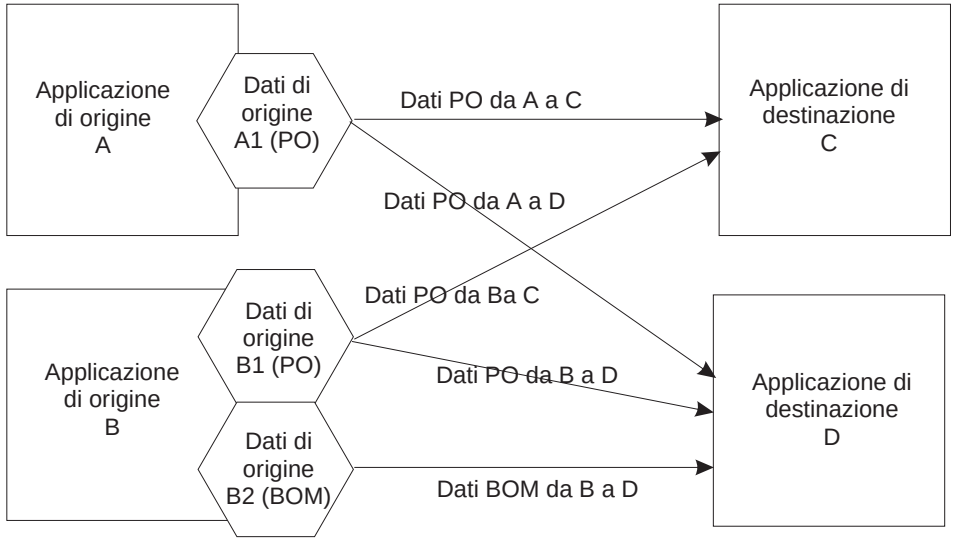



*Figura 3. Applicazioni connesse dai flussi di dati in una configurazione semplice*

In Figura 3, i dati di ordini di acquisto dall'applicazione A sono inviati alle applicazioni C e D, anche i dati di ordini di acquisto dall'applicazione B sono inviati alle applicazioni C e D; i dati relativi alle distinte dei materiali dall'applicazione B vengono inviati solo all'applicazione D. In ciascun caso, i dati sono convertiti in un formato industriale standard prima di essere inviati alle applicazione di destinazione. I dati di ordini di acquisto provenienti dalle applicazioni A e B vengono convertiti in un formato XML standard rappresentante i dati di ordini di acquisto. I dati delle distinte dei materiali provenienti dall'applicazione B vengono convertiti in un formato e XML standard rappresentante i dati delle distinte dei materiali.

Un trasporto di comunicazione, quale MQSeries o un'implementazione JMS (Java Message Service), viene utilizzato per inviare i dati all'applicazione o alle applicazioni di destinazione. Il messaggio di integrazione viene convertito in un *formato marshaling* richiesto dal trasporto di comunicazione specifico e poi consegnato al trasporto di comunicazione (ad esempio, una coda MQSeries). Ciascuna applicazione di destinazione può utilizzare trasporti di comunicazione e formati differenti per ricevere i messaggi. Ad esempio, l'applicazione C può utilizzare MQSeries per ricevere i messaggi e l'applicazione D può utilizzare JMS, come mostrato in Figura 4 a pagina 61. In questo caso, tutti i messaggi di integrazione inviati all'applicazione C (cioè, i dati relativi agli ordini di acquisto provenienti dall'applicazione A e B) sono convertiti in un formato marshaling MQSeries e tutti i messaggi di integrazione inviati all'applicazione D (i dati relativi agli ordini di acquisto provenienti dalle applicazioni A e B e i dati della distinta dei materiali

proveniente dall'applicazione B) sono convertiti in formato marshaling JMS. MQSeries Adapter Kernel utilizza i seguenti meccanismi per eseguire le conversioni:

- Viene utilizzato un *adattatore di origine* per convertire i dati dell'applicazione in un messaggio di integrazione. Gli adattatori di origine sono creati in MQSeries Adapter Builder.
- L'*adattatore nativo* viene utilizzato per convertire il messaggio di integrazione in un *messaggio di comunicazione*. L'adattatore nativo utilizza l'*LMS (logical message service)* per convertire il messaggio per il trasporto di comunicazione; l'LMS è specifico del trasporto di comunicazioni utilizzato. L'LMS poi utilizza un *programma di formattazione* per eseguire il marshal del messaggio in un trasporto.

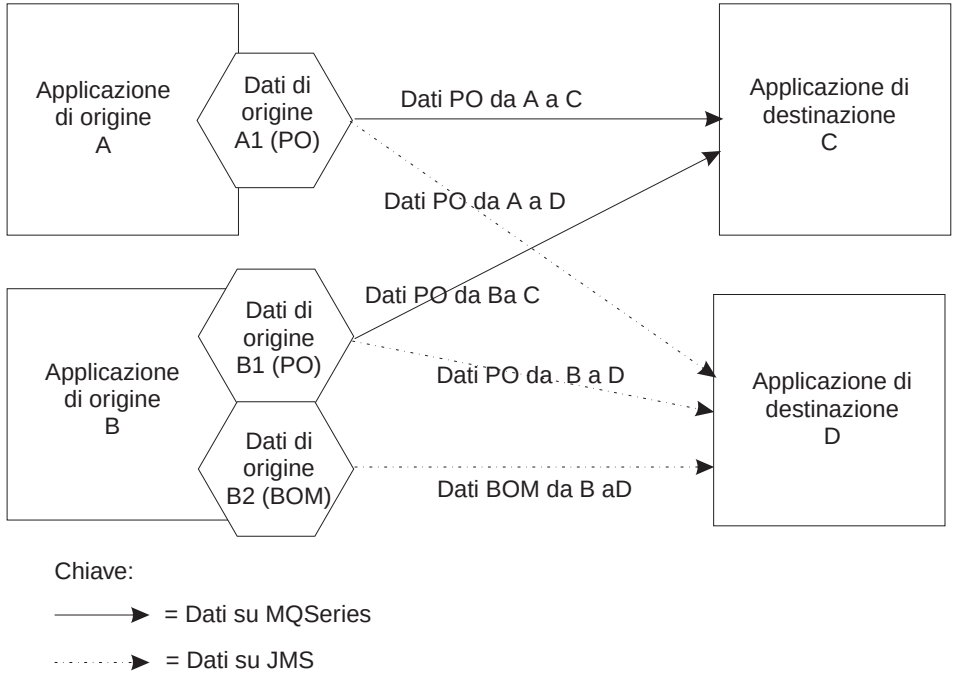



*Figura 4. Applicazioni connesse da differenti trasporti di comunicazione in una configurazione semplice.*

Il flusso di dati da un'applicazione al messaggio di integrazione per il messaggio di comunicazione è illustrato in Figura 5 a pagina 62 e Figura 6 a pagina 62. Quando la destinazione riceve un messaggio di comunicazione, le conversioni vengono invertite: l'adattatore nativo riconverte il messaggio di comunicazione in un messaggio di integrazione; poi, se necessario, l'adattatore

di destinazione converte il messaggio di integrazione nel formato di dati richiesto dall'applicazione di destinazione.

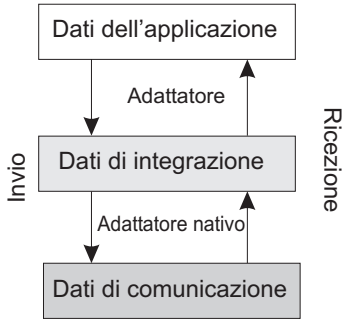



*Figura 5. Conversione dei dati*

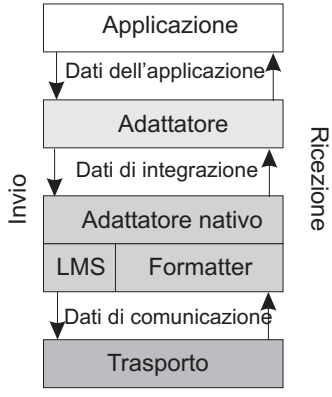



*Figura 6. Flusso di dati*

Ciascun punto per cui passano i dati deve essere rappresentato in un file di configurazione del kernel. Esistono tre requisiti di configurazione logica divisi dall'identificativo dell'applicazione (origine o destinazione). Un identificativo dell'applicazione può indicare un'applicazione origine o destinazione, in base ai messaggi inviati all'applicazione (destinazione) o provenienti da un'applicazione (origine). Il file di configurazione deve includere i seguenti tipi di informazione:

- Comunicazione
  - Destinazione dei dati
  - Il trasporto di comunicazione da utilizzare
  - Il metodo marshaling di comunicazione (programma di formattazione) da utilizzare
  - I requisiti di comunicazione, quali gestore code MQSeries e nomi code MQSeries o un fattore di connessione code JMS e nomi code JMS

- Adattatori (solo per il lato destinazione)
  - Gli adattatori richiesti per elaborare i dati
  - Il tipo di adattatori utilizzati (EAB o EJB)
  - Ulteriori informazioni per il tipo specifico di adattatore utilizzato
  - Per MQSeries Adapter Kernel stand-alone, le informazioni worker e daemon dell'adattatore
- Altro
  - Specifiche di traccia
  - Specifiche di log

Il flusso di dati relativi a parti differenti della configurazione viene illustrato in Figura 7.

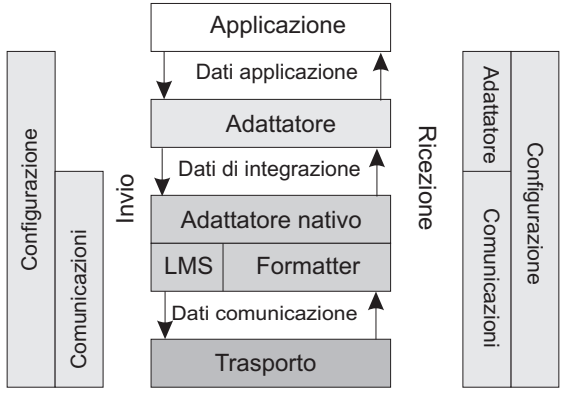



*Figura 7. Flusso di dati relativo alla configurazione*

Consultare "Sintassi e organizzazione del file di configurazione" a pagina 66 per ulteriori informazioni sull'associazione di questi requisiti di configurazione per gli elementi XML nel file `aqmconfig.xml` che controlla tali aspetti della configurazione del kernel. "Configurazioni comuni" a pagina 78 elenca molte configurazioni comuni.

## File per l'avvio e la configurazione

La configurazione del kernel viene determinata da numerosi file personalizzabili. Utilizzando un editor di testo standard, modificare i file per configurare il kernel in base alle singole esigenze. Per la configurazione del kernel, vengono utilizzati i seguenti file:

- Il file `aqmsetenv.bat` (sistemi Windows) o il file `aqmsetenv.sh` (UNIX) che imposta le variabili di ambiente. Editare questo file per modificare le variabili di ambiente del sistema dopo l'installazione, se si desidera. Le variabili di ambiente impostate da questo file includono PATH, CLASSPATH e LIBPATH. Queste variabili vengono impostate

automaticamente dal programma di installazione sui sistemi Windows. Per impostare queste variabili automaticamente al collegamento in UNIX, aggiungere i valori specificati nel file `aqmsetenv.sh` al file `.profile` (se si utilizza la shell Bourne o Kornl) o al file `.cshrc` (se si utilizza la shell C).

Per informazioni sulle impostazioni delle variabili di ambiente appropriate su OS/400, consultare il Passo 6 a pagina 43.

- Il file `aqmsetup`, che fornisce numerosi valori di di configurazione iniziali per il kernel. Per ulteriori informazioni, consultare "File di installazione".
- Il file `aqmconfig.xml`, che configura il kernel. Per ulteriori informazioni, consultare "File di configurazione" a pagina 65. Questo file contiene la maggior parte dei valori di configurazione del kernel.
- Il file `aqmcreateq.bat` (sistemi Windows) o il file `aqmcreateq.sh` (UNIX e OS/400), uno script che crea code di MQSeries. Consultare "Creazione di code di MQSeries" a pagina 97.

Tutti questi file contengono commenti che ne agevolano la modifica.

È consigliabile eseguire una copia di backup di questi file. Per ulteriori informazioni, consultare "Gestione del kernel" a pagina 94.

## File di installazione

Il file di installazione, `aqmsetup`, controlla molte delle impostazioni iniziali del kernel, tra cui:

- L'ubicazione del file di configurazione. Consultare "File di configurazione" a pagina 65.
- L'ubicazione dei DTD XML, nel caso in cui non si trovino nella directory corrente.
- Le variabili di ambiente JNI (Java Native Interface) per l'interfaccia C, per modificare l'ammontare di memoria utilizzato. Questa impostazione viene specificata quando un modulo eseguibile C avvia un processo e questo processo crea un'istanza di una macchina virtuale Java. La memoria utilizzata può essere controllata in questo caso eliminando i commenti e modificando le seguenti righe nel file `aqmsetup`:

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

Le dimensioni sono calcolate in byte.

Un esempio del file `aqmsetup` è disponibile in "Appendice E. Esempio del file di installazione" a pagina 125 ed è anche contenuto nella directory `samples` di installazione di MQSeries Adapter Kernel.

Se necessario, modificare il file di installazione la prima volta in cui MQSeries Adapter Kernel viene installato. Dopo l'installazione, questo file può essere modificato solo se il kernel riscontra un problema di insufficienza di memoria Java.

## File di configurazione

In questa sezione, viene descritto il file `aqmconfig.xml`, che determina la configurazione del kernel. "Sintassi e organizzazione del file di configurazione" a pagina 66 fornisce le informazioni sulla struttura del file di configurazione. Per informazioni più dettagliate sulla modifica del file di configurazione, consultare "Modifica del file di configurazione" a pagina 86.

La configurazione di MQSeries Adapter Kernel è determinata dal file XML `aqmconfig.xml`. Un esempio di questo file è contenuto in "Appendice D. Esempio del file di configurazione" a pagina 119; inoltre, è anche disponibile nella directory `samples` di installazione di MQSeries Adapter Kernel.

I valori specificati nel file di configurazione controllano i seguenti elementi del kernel:

- Identificativi logici di origine
- Identificativi logici di destinazione
- Informazioni worker e daemon dell'adattatore sul lato di destinazione
- Client di traccia
- Server di traccia
- Marshaling e instradamento dei messaggi, operazioni determinate dalle seguenti specifiche:
  - I nomi delle code di ricezione, di errore e di risposta
  - Una o più destinazioni predefinite a cui i messaggi vengono inviati
  - Il nome del gestore code MQSeries o fattore di connessione code JMS che invia o riceve il messaggio
  - Il timeout per la ricezione dei messaggi o risposte
  - La classe adattatore di destinazione sul lato di destinazione del kernel che elabora i singoli messaggi
  - Ulteriori informazioni specifiche per l'adattatore di destinazione
  - Il numero minimo di worker sul lato di destinazione (se viene eseguito MQSeries Adapter Kernel stand-alone)
  - L'abilitazione e la disabilitazione della traccia e il controllo del livello di traccia
  - L'abilitazione e la disabilitazione della registrazione di controllo
- Modalità di comunicazione

### Sintassi e organizzazione del file di configurazione

Dal momento che il file di configurazione di MQSeries Adapter Kernel si basa su LDAP (Lightweight Directory Access Protocol), la struttura del file di configurazione rispecchia il protocollo LDAP. L'elemento XML `Epic` rappresenta il livello superiore della directory LDAP, mentre gli oggetti LDAP di livello inferiore sono rappresentati da elementi XML nidificati all'interno dell'elemento di livello di superiore. Alcuni di questi elementi XML presentano attributi obbligatori che rappresentano le informazioni LDAP. I valori vengono aggiunti alla configurazione come contenuto degli elementi o come attributi degli elementi. Un esempio di un valore di configurazione assegnato come contenuto di un elemento è `<epictracelevel>-1</epictracelevel>`, che assegna il valore `-1` (tutti i messaggi possibili) all'elemento `epictracelevel`. Un esempio di valore di configurazione assegnato come attributo di un elemento è `<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">`, che assegna la classe `com.ibm.logging.ConsoleHandler` da utilizzare come gestore traccia.

Di seguito viene riportato un elenco degli elementi di livello superiore utilizzati nel file di configurazione. In "Elementi XML utilizzati nel file di configurazione" a pagina 69, viene elencato e descritto il set completo di elementi utilizzati nel file di configurazione. Per ulteriori dettagli sulle modalità di utilizzo di questi elementi, esaminare il file di configurazione di esempio.

- `Epic`—L'elemento di livello superiore richiesto per il file `aqmconfig.xml`.
- `ePICApplications`—L'elemento child obbligatorio dell'elemento `Epic`.
- `ePICApplication`—L'elemento child obbligatorio dell'elemento `ePICApplications`. Elenca e definisce le applicazioni servite dal kernel; per ciascuna applicazione, si richiede solo l'elemento `ePICApplication` definito per esteso (compresi gli elementi child).
- `AdapterRouting`—Un elemento child facoltativo di `ePICApplication`. Definisce il gestore code e le informazioni correlate.
- `ePICBodyCategory`—L'elemento child obbligatorio di `AdapterRouting`. Imposta la categoria di testo per i messaggi da instradare mediante il kernel.
- `ePICBodyType`—L'elemento child obbligatorio di `ePICBodyCategory`. Imposta il tipo di testo per i messaggi da instradare mediante il kernel. Contiene le definizioni per le voci, quali le definizioni di messaggio, le modalità di comunicazione per la ricezione dei messaggi e i programmi di formattazione dei messaggi.
- `ePICAdapterDaemonExtensions`—Un child facoltativo dell'elemento `ePICApplication` che rappresenta un daemon dell'adattatore. Contiene le informazioni correlate ai daemon dell'adattatore, compresi gli identificativi e il numero di worker dell'adattatore.

- `ePICTraceExtensions`—Un elemento child di `ePICApplication`, che rappresenta un'applicazione del client di traccia oppure un elemento del server di traccia. Definisce le informazioni correlate alla creazione di traccia.

Figura 8 a pagina 68 illustra la struttura di livello superiore del file di configurazione. Questo esempio non descrive il funzionamento, ma evidenzia semplicemente le relazioni e le dipendenze tra gli elementi di livello superiore. Per esaminare un esempio completo, consultare "Appendice D. Esempio del file di configurazione" a pagina 119.

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

*Figura 8. Struttura di livello superiore del file di configurazione*

Di seguito vengono forniti un elenco e la descrizione del set completo degli elementi utilizzati nel file di configurazione. Se per un elemento si specifica un valore predefinito, il kernel utilizza questo valore se un elemento della configurazione non richiede un valore specifico.

**Elementi XML utilizzati nel file di configurazione**

**Epic** Elemento di livello superiore utilizzato per il file di configurazione.

Elementi child:

- `context`
- `ePICApplications` (obbligatorio)

Attributi: `o="ePIC"` (obbligatorio)

**context**
Specifica la root del FSContext (file system context) JNDI (Java Naming and Directory Interface) file system context (FSContext) se vengono utilizzati oggetti JSM (Java Message Service). Il valore predefinito è la directory corrente. È obbligatorio se si utilizza JMS. Per informazioni sulle modalità di utilizzo degli oggetti JMS con MQSeries Adapter Kernel, consultare "Memorizzazione degli oggetti JMS" a pagina 106.

Elementi child: Nessuno

Attributi: Nessuno

**ePICApplications**
Contiene informazioni relative alle applicazioni servite dal kernel.

Elementi child: `ePICApplication` (obbligatorio)

Attributi: `o="ePICApplications"` (obbligatorio)

**ePICApplication**
Specifica le informazioni relative a un'applicazione servita dal kernel.

Elementi child:

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

Attributi: epicappid="*ID_applicazione*", dove *ID_applicazione* indica un identificativo di applicazione valido (obbligatorio)

**epiclogging**
Determina se eseguire il log di controllo. Il log di controllo richiede il prodotto WebSphere Business Integrator. Il valore predefinito è `false`.

Elementi child: Nessuno

Attributi: Nessuno

**epictrace**
Determina se utilizzare la funzione di traccia. Il valore predefinito è `false`.

Elementi child: Nessuno

Attributi: Nessuno

**epictracelevel**
Imposta il livello di traccia, utilizzando le costanti specificate dalla classe `com.ibm.logging.IRecordType`. Il valore predefinito è `0` (nessun messaggio). Consultare *Problem Determination Guide* per ulteriori informazioni sulla traccia e per un elenco completo dei livelli di traccia validi.

Elementi child: Nessuno

Attributi: Nessuno

**epictraceclientid**
Specifica il nome dell'applicazione client di traccia. Il valore predefinito è `TraceClient`.

Elementi child: Nessuno

Attributi: Nessuno

**epiclogoninfoclassname**
Specifica i nomi della classe di collegamento utilizzati per la connessione all'applicazione se si utilizza un adattatore EAB. Il valore predefinito è `com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault`.

Elementi child: Nessuno

Attributi: Nessuno

**AdapterRouting**
Contiene informazioni sui tipi di messaggi e sull'instradamento dei messaggi.

Elementi child:

- `epicmqppqueuemgr`

- epicuseremotequeuemanagertosend
- epicmqppqueuemgrhostname
- epicmqppqueuemgrportnumber
- epicmqppqueuemgrchannelname
- epicjmsconnectionfactoryname
- ePICBodyCategory (obbligatorio)

Attributi: cn="epicadapterrouting" (obbligatorio)

**epicmqppqueuemgr**
Se si utilizza MQSeries come meccanismo di trasporto, specifica il nome del gestore code da utilizzare. Nel caso in cui non sia stato specificato o sia stato specificato come DEFAULT, viene utilizzato il gestore code predefinito.

Elementi child: Nessuno

Attributi: Nessuno

**epicuseremotequeuemanagertosend**
Se MQSeries è utilizzato come meccanismo di trasporto, specifica se utilizzare un gestore di code remoto per l'invio dei messaggi. Il valore predefinito è false.

Elementi child: Nessuno

Attributi: Nessuno

**epicmqppqueuemgrhostname**
Se MQSeries è utilizzato come meccanismo di trasporto, specifica il nome host TCP/IP del meccanismo su cui risiede il gestore di code. Richiesto solo se viene utilizzato MQSeries Client.

Elementi child: Nessuno

Attributi: Nessuno

**epicmqppqueuemgrportnumber**
Se MQSeries è utilizzato come meccanismo di trasporto, specifica il numero di porta del processo server del gestore di code. Il valore predefinito è 1414 (valore predefinito di MQSeries). Richiesto solo se viene utilizzato MQSeries Client.

Elementi child: Nessuno

Attributi: Nessuno

**epicmqppqueuemgrchannelname**
Se si utilizza MQSeries come meccanismo di trasporto, specifica il nome del canale del server del gestore code. Richiesto solo se viene utilizzato MQSeries Client.

Elementi child: Nessuno

Attributi: Nessuno

**epicjmsconnectionfactoryname**
Se si utilizza JMS il meccanismo di trasporto, specifica il nome predefinito della connessione JMS. Il valore deve essere specificato nel formato *attributo=oggetto*, dove *attributo* è l'attributo LDAP e *oggetto* e l'oggetto connessione JMS. L'oggetto viene memorizzato nell'elemento `AdapterRouting`. Ad esempio, per un oggetto connessione JMS chiamato `QCFTEST1` con l'attributo LDAP `cn`, il valore specificato da questo elemento è `cn=QCFTEST1`.

Elementi child: Nessuno

Attributi: Nessuno

**ePICBodyCategory**
Specifica la categoria di testo del messaggio inviato.

Elementi child: `ePICBodyType` (obbligatorio)

Attributi: `epicbodycategory=`*`categoria_testo`*, dove *categoria_testo* specifica la categoria di testo dei messaggi inviati (obbligatorio)

**ePICBodyType**
Specifica il tipo di testo dei messaggi da inviare.

Elementi child:

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

- epicreceivefiledir
- epiccommitfiledir
- epicerrorfiledir

Attributi: epicbodytype=*tipo_testo*, dove *tipo_testo* specifica il tipo di testo dei messaggi inviati (obbligatorio)

**epiccommandclassname**
Specifica il nome di un adattatore di destinazione EAB o comando EJB richiamato per elaborare i messaggi. Richiesto se un daemon di adattatore o WebSphere Application Server è utilizzato per ricevere i messaggi.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommandtype**
Specifica il tipo di adattatore di destinazione. Possibili valori sono MQAKEAB e MQAKEJB. MQAKEAB specifica un adattatore di destinazione MQSeries Adapter Kernel EAB standard; MQAKEJB specifica l'utilizzo degli enterprise bean sul lato di destinazione del kernel in WebSphere Application Server. L'impostazione predefinita è MQAKEAB. E' richiesto il valore MQAKEJB quando l'adattatore di destinazione è un enterprise bean.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommandejbmapper**
Specifica il nome della classe TDCMapper utilizzata per associare i dati di emissione. L'impostazione predefinita è TDCGenericMapper. E' richiesto quando l'adattatore di destinazione è un enterprise bean.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommandejbmethod**
Specifica il nome del metodo di richiamo per un enterprise bean. Il metodo deve accettare un oggetto TerminalDataContainer come immissione e restituire un oggetto TerminalDataContainer. L'impostazione predefinita è execute. E' richiesto quando l'adattatore di destinazione è un enterprise bean.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommandejbmethodparmtype**
Specifica il nome classe dell'oggetto utilizzato come parametro per il

metodo richiamato sull'enterprise bean. L'impostazione predefinita è il nome classe dell'oggetto restituito da `TDCMapper`. E' richiesto quando l'adattatore di destinazione è un enterprise bean.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommandejburl**

Specifica l'URL (Uniform Resource Locator) di un enterprise bean, in forma `IIOP://`*`nome_host`*`:`*`porta`*, dove *nome_host* è il nome host del server EJB e *porta* è la porta di listening del nome server (per impostazione predefinita `900`). L'impostazione predefinita è `IIOP:///`. E' richiesto quando l'adattatore di destinazione è un enterprise bean.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommandejbinitialcontext**

Specifica il nome del fattore di contesto iniziale utilizzato per ricercare il nome home dell'enterprise bean. L'impostazione predefinita è `com.ibm.ejs.ns.jndi.CNInitialContextFactory`. E' richiesto quando l'adattatore di destinazione è un enterprise bean.

Elementi child: Nessuno

Attributi: Nessuno

**epicdestids**

Specifica gli identificativi di una o più applicazioni da utilizzare come destinazioni dei messaggi. È obbligatorio se l'applicazione invia messaggi e l'ID logico di destinazione è impostato su `NONE`.

Elementi child: Nessuno

Attributi: Nessuno

**epicreceivemode**

Specifica la modalità di comunicazione da utilizzare. Consultare "Appendice A. Modalità di comunicazione" a pagina 103 per un elenco di modalità di comunicazione valide e le relative spiegazioni. E' obbligatorio se l'applicazione riceve messaggi.

Elementi child: Nessuno

Attributi: Nessuno

**epicmessageformatter**

Specifica il programma di formattazione dei messaggi da utilizzare, a seconda del valore di `epicreceivemode` e del metodo di trasporto utilizzati. Per ulteriori informazioni sul programma di formattazione

dei messaggi e sui metodi di trasporto, consultare Tabella 10 a pagina 105 e Tabella 11 a pagina 105.

Elementi child: Nessuno

Attributi: Nessuno

**epicreceivetimeout**
Specifica, in millisecondi, il tempo di attesa del ricevente prima che i messaggi entrino in timeout. Il valore predefinito è `0`. Il valore `-1` indica che il timeout non è stato specificato (un tempo di attesa indefinito).

Elementi child: Nessuno

Attributi: Nessuno

**epicreceivemqppqueue**
Specifica il nome della coda dalla quale ricevere messaggi. Richiesto quando l'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. Consultare "Appendice A. Modalità di comunicazione" a pagina 103 per un elenco delle modalità di comunicazione MQSeries.

Elementi child: Nessuno

Attributi: Nessuno

**epicerrormqppqueue**
Specifica il nome della coda in cui inserire i messaggi di errore. Richiesto se viene utilizzata una coda di messaggi di errore e l'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. Consultare "Appendice A. Modalità di comunicazione" a pagina 103 per un elenco delle modalità di comunicazione MQSeries.

Elementi child: Nessuno

Attributi: Nessuno

**epicreplymqppqueue**
Specifica il nome della coda dalla quale ricevere messaggi di risposta. Richiesto se vengono utilizzate richieste di risposta e l'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. Consultare "Appendice A. Modalità di comunicazione" a pagina 103 per un elenco delle modalità di comunicazione MQSeries.

Elementi child: Nessuno

Attributi: Nessuno

**epicjmsreceivequeuename**
Specifica il nome della coda dalla quale ricevere messaggi. Richiesto per la modalità di comunicazione JMS. L'oggetto viene memorizzato

nell'elemento ePICBodyType. Il valore deve essere specificato nel formato *attributo=oggetto*, dove *attributo* è l'attributo LDAP e *oggetto* è il nome dell'oggetto coda JMS. Ad esempio, per un oggetto JMS chiamato TEST1AIQ con l'attributo LDAP cn, il valore specificato da questo elemento è cn=TEST1AIQ.

Elementi child: Nessuno

Attributi: Nessuno

**epicjmserrorqueuename**
Specifica il nome della coda in cui inserire i messaggi di errore. Richiesto se viene utilizzata una coda di messaggi di errore con la modalità di comunicazione JMS. L'oggetto viene memorizzato nell'elemento ePICBodyType. Il valore deve essere specificato nel formato *attributo=oggetto*, dove *attributo* è l'attributo LDAP e *oggetto* è il nome dell'oggetto coda JMS. Ad esempio, per un oggetto JMS chiamato TEST1AEQ, con l'attributo LDAP cn, il valore specificato da questo elemento è cn=TEST1AEQ.

Elementi child: Nessuno

Attributi: Nessuno

**epicjmsreplyqueuename**
Specifica il nome della coda dalla quale ricevere messaggi di risposta. Richiesto se vengono utilizzate richieste di risposta con la modalità di comunicazione JMS. L'oggetto viene memorizzato nell'elemento ePICBodyType. Il valore deve essere specificato nel formato *attributo=oggetto*, dove *attributo* è l'attributo LDAP e *oggetto* è il nome dell'oggetto coda JMS. Ad esempio, per un oggetto JMS chiamato TEST1RPL con l'attributo LDAP cn, il valore specificato da questo elemento è cn=TEST1RPL.

Elementi child: Nessuno

Attributi: Nessuno

**epicreceivefiledir**
Specifica il nome della directory dalla quale ricevere messaggi. Richiesto per la modalità di comunicazione FILE.

Elementi child: Nessuno

Attributi: Nessuno

**epiccommitfiledir**
Specifica il nome della directory in cui conservare i messaggi ricevuti in attesa che vengano confermati. Richiesto per la modalità di comunicazione FILE alla ricezione dei messaggi.

Elementi child: Nessuno

Attributi: Nessuno

**epicerrorfiledir**
Specifica il nome della directory in cui inserire i messaggi di errore. Richiesto se viene utilizzata una coda di messaggi di errore con la modalità di comunicazione FILE.

Elementi child: Nessuno

Attributi: Nessuno

**ePICAdapterDaemonExtensions**
Contiene informazioni relative alle estensioni dei daemon dell'adattatore.

Elementi child:

- `epicdepappid`
- `epicminworkers`

Attributi: `cn="epicappextensions"` (obbligatorio)

**ePICTraceExtensions**
Contiene informazioni sulle estensioni di traccia. Per una descrizione completa di questo elemento, consultare *Problem Determination Guide*.

Elementi child:

- `epicdepappid`
- `epictracesyncoperation`
- `epictracemessagefile`
- `epictracehandler`
- `ePICTraceHandler`

Attributi: `cn="epicappextensions"` (obbligatorio)

**epicdepappid**
Specifica l'identificativo dell'applicazione per cui è attivo il daemon dell'adattatore. Per impostazione predefinita, l'ID dell'applicazione da cui è stato avviato il daemon dell'adattatore.

Elementi child: Nessuno

Attributi: Nessuno

**epicminworkers**
Specifica il numero dei worker dell'adattatore avviati dal daemon dell'adattatore. Il valore predefinito è 1.

Elementi child: Nessuno

Attributi: Nessuno

### Configurazioni comuni

Questa sezione elenca i valori di configurazione per i vari scenari di configurazioni comuni, inclusi i valori per l'invio e la ricezione dei messaggi utilizzando i diversi trasporti di comunicazione. Quando un messaggio viene inviato, i valori di configurazione sono richiamati dal lato origine e dal lato destinazione; quando un messaggio viene ricevuto, i valori di configurazione sono richiamati solo dal lato destinazione. L'origine e la destinazione sono rappresentati dai rispettivi identificativi logici. In questo esempio l'origine e la destinazione sono su due differenti macchine. Se l'identificativo dell'applicazione di destinazione non è già impostato, viene determinato dal valore dell'elemento `epicdestids` nella configurazione dell'origine.

**Nota:** I scenari di configurazione elencano i valori di configurazione degli elementi applicabili e che possono essere impostati. Fare riferimento all'elenco degli elementi in "Elementi XML utilizzati nel file di configurazione" a pagina 69 per i relativi valori predefiniti.

**Configurazioni comuni di MQSeries:** Questa sezione fornisce le configurazioni comuni per l'utilizzo di MQSeries come trasporto di comunicazione. L'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries (ad esempio MQPP o MQRFH2). Vengono illustrati i seguenti scenari:

- Tabella 2 presenta gli elementi di configurazione da impostare per l'invio di un messaggio da un server MQSeries ad un server MQSeries.
- Tabella 3 a pagina 79 illustra gli elementi di configurazione da impostare per l'invio di un messaggio da un server MQSeries che utilizza un manager di code remoto ad un server MQSeries.
- Tabella 4 a pagina 80 presenta gli elementi di configurazione da impostare per l'invio di un messaggio da un client MQSeries che utilizza un server host ad un server MQSeries.
- Tabella 5 a pagina 81 presenta gli elementi di configurazione da impostare per la ricezione di un messaggio su un server MQSeries.
- Tabella 6 a pagina 82 presenta gli elementi di configurazione da impostare per la ricezione di un messaggio su un client MQSeries che utilizza un server host.

*Tabella 2. Configurazione comune: Invio di un messaggio da un server MQSeries ad un altro server MQSeries*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. |

*Tabella 2. Configurazione comune: Invio di un messaggio da un server MQSeries ad un altro server MQSeries (Continua)*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| L'elemento `epicmqppqueuemgr` specifica il nome del gestore delle code. Il gestore code deve esistere sulla macchina dell'applicazione di origine. | |
| | L'elemento `epicreceivemqppqueue` specifica il nome della coda di ricezione. Questa coda deve essere una coda remota MQSeries sulla macchina dell'applicazione di origine o parte di un cluster MQSeries. |
| L'elemento `epicreplymqppqueue` specifica il nome della coda di risposta. Questa coda deve essere una coda locale MQSeries sulla macchina di invio messaggio o parte di un cluster MQSeries. Utilizzato solo per richieste e risposte sincrone. | |
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione di una risposta prima del time out. | |

*Tabella 3. Configurazione comune: Invio di un messaggio da un server MQSeries ad un server MQSeries tramite un gestore di code remoto*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. |
| L'elemento `epicmqppqueuemgr` specifica il nome del gestore delle code. Il gestore code deve esistere sulla macchina dell'applicazione di origine. | L'elemento `epicmqppqueuemgr` specifica il nome del gestore delle code. Tale gestore di code deve esistere sulla macchina dell'applicazione di origine. Il nome deve essere specificato: non è possibile utilizzare un valore predefinito. |
| L'elemento `epicremotequeuemanagertosend` specifica che un gestore di code remoto viene utilizzato per l'invio dei messaggi. | |

*Tabella 3. Configurazione comune: Invio di un messaggio da un server MQSeries ad un server MQSeries tramite un gestore di code remoto (Continua)*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicreceivemqpqueue` specifica il nome della coda di ricezione. Questa coda deve essere una coda locale MQSeries sulla macchina dell'applicazione di destinazione o parte di un cluster MQSeries. |
| L'elemento `epicreplymqpqueue` specifica il nome della coda di risposta. Questa coda deve essere una coda locale MQSeries sulla macchina di invio messaggio o parte di un cluster MQSeries. Utilizzato solo per richieste e risposte sincrone. | |
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione di una risposta prima del time out. | |

*Tabella 4. Configurazione comune: Invio di messaggi da un client MQSeries che utilizza un server host ad un server MQSeries*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. |
| L'elemento `epicmqpqueuemgr` specifica il nome del gestore delle code. Tale gestore di code deve esistere sulla macchina host del client che esegue l'invio. | |
| L'elemento `epicmqpqueuemgrhostname` specifica il nome host della macchina server MQSeries. | |
| L'elemento `epicmqpqueuemgrportnumber` specifica il numero di porta del processo server del gestore di code sulla macchina server. | |
| L'elemento `epicmqpqueuemgrchannelnumber` specifica il numero di canale del server del gestore di code. | |

*Tabella 4. Configurazione comune: Invio di messaggi da un client MQSeries che utilizza un server host ad un server MQSeries (Continua)*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicreceivemqppqueue` specifica il nome della coda di ricezione. Questa coda deve essere una coda remota MQSeries sulla macchina dell'applicazione di origine o parte di un cluster MQSeries. |
| L'elemento `epicreplymqppqueue` specifica il nome della coda di risposta. Questa coda deve essere una coda locale MQSeries sulla macchina host del client di invio messaggio o parte di un cluster MQSeries. Utilizzato solo per richieste e risposte sincrone. | |
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione di una risposta prima del time out. | |

*Tabella 5. Configurazione comune: server MQSeries di ricezione messaggi*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| Non applicabile. | L'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. |
| | L'elemento `epicmqppqueuemgr` specifica il nome del gestore delle code. Tale gestore di code deve esistere sulla macchina dell'applicazione di origine. |
| | L'elemento `epicreceivemqppqueue` specifica il nome della coda di ricezione. Questa coda deve essere una coda locale MQSeries sulla macchina di destinazione. |
| | L'elemento `epicerrormqppqueue` specifica il nome della coda di errori. Questa coda deve essere una coda locale MQSeries sulla macchina di destinazione o parte di un cluster. Richiesto solo se viene utilizzato un adattatore worker. |

*Tabella 5. Configurazione comune: server MQSeries di ricezione messaggi (Continua)*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| | L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione per un messaggio prima del time out. |

*Tabella 6. Configurazione comune: client MQSeries che utilizza un server host per la ricezione dei messaggi.*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| Non applicabile. | L'elemento `epicreceivemode` specifica una modalità di comunicazione MQSeries. |
| | L'elemento `epicmqppqueuemgr` specifica il nome del gestore delle code. Tale gestore di code deve esistere sulla macchina host del client del programma per la ricezione dei messaggi. |
| | L'elemento `epicmqppqueuemgrhostname` specifica il nome host della macchina server MQSeries. |
| | L'elemento `epicmqppqueuemgrportnumber` specifica il numero di porta del processo server per le code sulla macchina server. |
| | L'elemento `epicmqppqueuemgrchannelnumber` specifica il numero di canale del server del gestore di code. |
| | L'elemento `epicreceivemqppqueue` specifica il nome della coda di ricezione. Questa coda deve essere una coda locale MQSeries sulla macchina host del client del programma di ricezione. |
| | L'elemento `epicerrormqppqueue` specifica il nome della coda di errori. Questa coda deve essere una coda locale MQSeries sulla macchina host client del programma di ricezione o parte di un cluster. Richiesto solo se viene utilizzato un adattatore worker. |

*Tabella 6. Configurazione comune: client MQSeries che utilizza un server host per la ricezione dei messaggi. (Continua)*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| | L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione per un messaggio prima del time out. |

**Configurazioni comuni JMS:** Questa sezione fornisce le configurazioni comuni per l'utilizzo di JMS come trasporto di comunicazione. L'elemento `epicreceivemode` specifica JMS.

Se l'implementazione MQSeries JMS viene utilizzata, devono esistere gli oggetti MQSeries appropriati. Ad esempio, un fattore di connessione code JMS deve essere correlato ad un gestore di code su un server MQSeries e una coda JMS deve essere correlata a una coda MQSeries. Non è necessario che gli oggetti MQSeries vengano elencati nella configurazione, ma devono esistere gli oggetti di supporto MQSeries.

Vengono illustrati i seguenti scenari:

- Tabella 7 illustra gli elementi di configurazione da impostare per l'invio di un messaggio via JMS.
- Tabella 8 a pagina 84 illustra gli elementi di configurazione da impostare per la ricezione dei messaggi via JMS.

*Tabella 7. Configurazione comune: Invio di un messaggio via JMS*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| | L'elemento `epicreceivemode` specifica la modalità di comunicazione JMS. |
| L'elemento `epicjmsconnectionfactoryname` specifica il nome del fattore di connessione code JMS. L'oggetto correlato deve esistere nella configurazione. | |
| | L'elemento `epicjmsreceivequeuename` specifica il nome della coda di ricezione JMS. L'oggetto correlato deve esistere nella configurazione. |

*Tabella 7. Configurazione comune: Invio di un messaggio via JMS (Continua)*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| L'elemento `epicjmsreplyqueuename` specifica il nome della coda di risposta JMS. L'oggetto correlato deve esistere nella configurazione. Utilizzato solo per richieste e risposte sincrone. | |
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione di una risposta prima del time out. | |

*Tabella 8. Configurazione comune: Ricezione di un messaggio via JMS*

| Configurazione di origine | Configurazione di destinazione |
|---|---|
| Non applicabile. | L'elemento `epicreceivemode` specifica la modalità di comunicazione JMS. |
| | L'elemento `epicjmsconnectionfactoryname` specifica il nome del fattore di connessione code JMS. L'oggetto correlato deve esistere nella configurazione. |
| | L'elemento `epicjmsreceivequeuename` specifica il nome della coda di ricezione JMS. L'oggetto correlato deve esistere nella configurazione. |
| | L'elemento `epicjmserrorqueuename` specifica il nome della coda di errori JMS. L'oggetto correlato deve esistere nella configurazione. |
| | L'elemento `epicmessageformatter` specifica il nome del programma di formattazione da utilizzare. |
| | L'elemento `epicreceivetimeout` specifica il tempo di attesa del programma di ricezione per un messaggio prima del time out. |

**Configurazioni comuni dell'adattatore:** Questa sezione fornisce le configurazioni comuni per il richiamo di un adattatore sul lato di destinazione. Vengono utilizzati diversi valori di configurazione in base

all'utilizzo di un adattatore di destinazione EAB -Enterprise Access Builder- (specificato dal valore epiccommandtype di MQAKEAB) o di un adattatore di destinazione bean di sessione di servizio EJB (specificato da un valore epiccommandtype di MQAKEJB).

**Nota:** Gli adattatori di destinazione bean di sessione di servizio EJB sono supportati solo da WebSphere Application Server.

Per il valore epiccommandtype di MQAKEAB, specificare i valori per gli elementi ulteriori riportati di seguito:

- epiclogoninfoclassname
- epiccommandclassname

Per il valore epiccommandtype di MQAKEJB, specificare i valori per gli elementi ulteriori riportati di seguito:

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

### Aggiunta delle informazioni sull'adattatore alla configurazione

Quando si aggiunge un nuovo adattatore alla configurazione del kernel, è necessario aggiungere al file di configurazione numerose specifiche. Per un esempio di file di configurazione minimo, fare riferimento al file aqmconfig.minimum.xml. Questo file è illustrato in "Appendice D. Esempio del file di configurazione" a pagina 119 ed è incluso anche nella directory samples dell'installazione di MQSeries Adapter Kernel.

Le specifiche riportate di seguito rappresentano le informazioni minime da inserire alla configurazione quando si aggiunge un nuovo adattatore:

- **Adattatore di origine** (invio messaggi):
  - L'identificativo dell'applicazione in cui viene eseguito l'adattatore di origine.
  - Il gestore code predefinito. Se si utilizza MQSeries come meccanismo di trasporto e viene installato ed eseguito sullo stesso computer dell'adattatore di origine, non è necessario configurare specificamente il gestore code.
  - Identificativi logici di destinazione per i messaggi. Se tutti i messaggi giungono alla stessa destinazione, utilizzare la categoria di testo e il tipo di testo DEFAULT.

- Una coda di ricezione per ciascun identificativo logico di destinazione al quale l'adattatore di origine invia i messaggi.

- **Adattatore di destinazione** (ricezione messaggi):
  - L'identificativo dell'applicazione in cui viene eseguito l'adattatore di destinazione.
  - Il gestore code predefinito. Se si utilizza MQSeries come meccanismo di trasporto e viene installato ed eseguito sullo stesso computer dell'adattatore di origine, non è necessario configurare specificamente il gestore code.
  - La modalità di ricezione per MQSeries. In genere, è lo stesso per tutti i messaggi; in questo caso, utilizzare la categoria di testo e il tipo di testo DEFAULT.
  - La coda di ricezione. Nel caso in cui sia la stessa per tutti i messaggi, utilizzare una categoria di testo e un tipo di testo DEFAULT.
  - La coda di errore, nel caso in cui si verifichi un errore quando l'adattatore di destinazione elabora il messaggio. In genere, è lo stesso per tutti i messaggi; in questo caso, utilizzare la categoria di testo e il tipo di testo DEFAULT.
  - Il nome della classe adattatore di destinazione da richiamare quando si riceve un messaggio. È specifico della categoria di testo e del tipo di testo.
  - Valore timeout di ricezione. Si consiglia di impostare un valore appropriato per prevenire un alto utilizzo della CPU. In genere, è lo stesso per tutti i messaggi; in questo caso, utilizzare la categoria di testo e il tipo di testo DEFAULT.

  Per gli adattatori di destinazione aggiuntivi, sono sufficienti le stesse informazioni se si utilizza la stessa coda di ricezione. In tal caso, l'unica informazione da specificare in maniera diversa è il nome della classe adattatore di destinazione da richiamare per la categoria e il tipo di testo specifici.

- **Specifiche di traccia**:
  - Se la traccia è attivata.
  - Il livello di traccia.
  - Ulteriori informazioni di traccia, quali la destinazione della traccia, per gli adattatori di origine e di destinazione. In base all'impostazione predefinita, la traccia viene visualizzata nella finestra di prompt dei comandi o nel terminale in cui è stato avviato il kernel.

### Modifica del file di configurazione

Utilizzare un editor di testo oppure un editor XML dedicato per modificare il file di configurazione. Per coloro che si servono di un editor XML, viene fornito il file DTD `aqmconfig.dtd`, disponibile nella directory `samples` di installazione del kernel. Un editor XML chiamato Xeena può essere scaricato

dal sito Web IBM alphaWorks all'indirizzo www.alphaworks.ibm.com. Le informazioni riportate di seguito si applicano al file di configurazione:

- Prima di modificare il file di configurazione, consultare tutte le informazioni sulla configurazione desiderata: i nomi delle applicazioni e delle code richiamate nella configurazione, i tipi di messaggi scambiati, il modo o i modi di comunicazione utilizzati e le informazioni sui programmi di traccia e sulle altre estensioni.
- Copiare il file `aqmconfig.xml` di esempio dalla directory `samples` nell'ubicazione desiderata. Non ridenominare la copia del file di configurazione. Modificare la copia.
- Utilizzare i commenti per identificare sezioni differenti del file di configurazione e per documentare i valori specifici utilizzati nel file di configurazione (ad esempio, gli identificativi delle applicazioni, il nome della coda di messaggi e i valori di timeout). In XML, i commenti sono introdotti dai caratteri `<!--` e terminano con i caratteri `-->`. I commenti possono occupare più righe, come nell'esempio riportato di seguito:

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

  In XML non è possibile inserire commenti all'interno di altri commenti.
- Organizzare il file di configurazione in base agli identificativi di applicazione. Conservare insieme le voci relative a ciascun identificativo di applicazione.
- Se non viene utilizzato un editor specifico per XML, utilizzare un editor di testo che non spezzi le righe al salvataggio del file. Esempi di questo tipo di editor di testo sono Notepad sui sistemi Windows e vi o Emacs su UNIX.
- È bene ricordare che XML opera una distinzione tra maiuscolo e minuscolo; utilizzare il formato corretto per tutti i nomi e gli attributi delle tag (elemento). In caso contrario, le tag possono invalidare il file di configurazione. Pertanto, è consigliabile utilizzare un editor XML dedicato.
- Se si desidera utilizzare i valori predefiniti per la categoria testo e tipo testo ed i valori non sono già impostati come predefiniti, è necessario configurare il valore DEFAULT per ciascuna impostazione nel file di configurazione. In caso contrario, il kernel non utilizza alcun valore predefinito.
- Convalidare il file di configurazione prima di attivarlo. Consultare "Convalida del file di configurazione" a pagina 88.
- Le modifiche apportate al file di configurazione diventano effettive al successivo riavvio del processo kernel. Se, durante la modifica del file di configurazione, è in esecuzione un processo, arrestarlo e avviarlo di nuovo per rendere effettive le modifiche. Se si modifica il file di configurazione utilizzato al momento, è necessario eseguire questa operazione con estrema cautela.

- Eseguire il backup del file di configurazione ogni volta che viene modificato.

### Convalida del file di configurazione

Dopo aver modificato il file di configurazione e prima di attivarlo, è necessario convalidarlo. A tale scopo, completare la procedura riportata di seguito:

1. Creare una directory di convalida per il file di configurazione in cui convalidarlo e configurare la prova.
2. Creare un messaggio XML di convalida.
3. Impostare le code messaggi per supportare la prova di convalida.
4. Configurare, quindi eseguire una prova del file di configurazione che preveda l'invio e la ricezione di un messaggio.
5. Esaminare i risultati della prova per determinare se il file di configurazione è corretto.

L'utilità che agevola la creazione di un messaggio XML di convalida e la prova di convalida da eseguire sul file di configurazione, vengono fornite entrambi come parti del kernel.

Durante la prova di convalida viene richiamato il metodo `sendMsg` e viene inviato un messaggio XML di convalida da un adattatore nativo sul lato di origine del kernel a un daemon dell'adattatore sul lato di destinazione del kernel. Non sono richiesti l'adattatore di origine e di destinazione. Tuttavia, è stato installato un adattatore di destinazione, è anche possibile verificare l'invio di un messaggio all'applicazione di destinazione.

Di seguito viene riportata la procedura.

**Nota:** Per agevolare questa procedura, vengono forniti numerosi script. Se si desidera, copiare gli script e modificare le copie per creare delle versioni personalizzate. Se si utilizza OS/400, le versioni UNIX degli script possono essere eseguite in una sessione **qsh**. E' possibile avviare una sessione **qsh** immettendo il comando Start QSH (**STRQSH**) da un prompt CL. prompt.

Passo 1. Aprire una finestra di prompt dei comandi.

Passo 2. Creare una directory di convalida per il file di configurazione. Copiare in questa directory il file di configurazione e il file di installazione.

Passo 3. Visualizzare la directory di convalida.

Passo 4. Immettere il seguente comando per creare il messaggio XML di convalida:

- `aqmcrtmsg.bat` (sistemi Windows)

- `aqmcrtmsg.sh` (UNIX e OS/400)

Passo 5. Viene visualizzato un elenco di opzioni. Selezionare un'opzione e premere il tasto di invio. Immettere un valore per ciascuna opzione. L'ordine in base al quale si specificano i valori è irrilevante. `set sourcelogicalid`, `set msgtype` e `set bodycategory` sono esempi di opzioni. È necessario specificare il valore per le opzioni 20, 21, 22 e 23. È possibile utilizzare l'opzione 24 o 241 per fornire i dati di testo del messaggio. Gli altri valori non sono obbligatori.

Passo 6. Immettere l'opzione 1 per creare il file XML di convalida. Questo file viene creato nella directory corrente e viene denominato `EpicMessage`*`nn`*`.xml`, dove *nn* è il numero del file XML.

Passo 7. Immettere l'opzione 0 per uscire dall'utilità di convalida.

Passo 8. Impostare le code messaggi appropriate per supportare la convalida.

Passo 9. Impostare la variabile di ambiente AQMSETUPFILE in modo che faccia riferimento temporaneamente al file di installazione nella directory di convalida.

- Da un prompt dei comandi sui sistemi Windows, immettere il seguente comando:

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

  dove `E:\` rappresenta l'unità corretta e *file_run_time* è la directory di convalida.
- Sui sistemi UNIX e OS/400, immettere il seguente comando. In questo esempio, si presume l'uso della Korn shell; nel caso in cui venga utilizzata una shell differente, modificare il comando.

  ```
  export AQMSETUPFILE=directory_principale/file_run_time/aqmsetup
  ```

  dove *directory_principale* è la directory di installazione del kernel e *file_run_time* è la directory di convalida. Su OS/400, il file `aqmsetup` deve essere sempre ubicato nel la directory home IFS (`/home/`*`nome_utente`*).

Se occorre, modificare il file di installazione nella directory di convalida in modo che richiami il file di configurazione da convalidare.

Passo 10. Eseguire una delle seguenti prove:

- Verificare solo il lato di origine del kernel.
- Verificare se il messaggio può essere instradato verso l'applicazione di destinazione. Per eseguire questa prova è necessario aver già installato un adattatore di destinazione.
- Traccia.

Eseguire, innanzitutto, una verifica del lato di origine, quindi del lato di destinazione. Arrestare il daemon dell'adattatore per verificare solo il lato di origine. Attivare il daemon dell'adattatore per verificare anche il lato di destinazione. Nel caso in cui l'adattatore di destinazione non sia stato già installato, è possibile verificare se il daemon dell'adattatore elabora il messaggio fino al punto in cui cerca di richiamare il comando per l'adattatore di destinazione appropriato. È consigliabile abilitare la creazione di traccia, soprattutto se l'adattatore di destinazione non è stato ancora installato.

Passo 11. Eseguire la prova di convalida. Da una directory qualsiasi, immettere il seguente comando:

- Sui sistemi Windows:

  ```
  aqmsndmsg.bat -a identificativo_logico_origine -f file_messaggi_XML
  ```

- Su UNIX e OS/400:

  ```
  aqmsndmsg.sh -a identificativo_logico_origine -f file_messaggi_XML
  ```

  dove:

  *identificativo_logico_origine*
  indica l'identificativo logico di origine. Questo valore deve corrispondere al valore dell'identificativo logico di origine immesso per l'opzione 20 nel Passo 5 a pagina 89.

  *file_messaggi_XML*
  indica il file di messaggi XML.

**Nota:** Immettendo il comando riportato di seguito, è possibile visualizzare un elenco di tutte le opzioni da utilizzare per questa prova:

Sui sistemi Windows:

```
aqmsndmsg.bat -?
```

Su UNIX e OS/400:

```
aqmsndmsg.sh -?
```

`-?` funziona solo sulla shell Korn; se si utilizza un'altra shell UNIX (la shell Bourne o C), utilizzare il carattere barra retroversa prima del punto interrogativo (`-\?`).

Passo 12. Esaminare i risultati. Il messaggio di convalida contiene la categoria, il tipo e i dati di testo validi.

- Se si esegue una verifica del solo lato di origine del kernel (vale a dire, se il daemon dell'adattatore non è stato avviato), esaminare la coda verso la quale il messaggio è stato instradato.

- Se il messaggio di convalida è contenuto in questa coda, le voci nel file di configurazione sono state convalidate.
- In caso contrario, controllare il file delle eccezioni. Se la creazione di traccia è stata abilitata, controllare i messaggi di traccia.

- Se si esegue una prova del lato di destinazione del kernel e l'adattatore di destinazione è stato installato, controllare l'applicazione di destinazione.
  - Se l'applicazione di destinazione riceve il messaggio di convalida, le voci del file di configurazione sono state convalidate.
  - In caso contrario, controllare il file delle eccezioni. Se la creazione di traccia è stata abilitata, controllare i messaggi di traccia.
- Se si esegue una prova del lato di destinazione del kernel e l'adattatore di destinazione non è stato installato, controllare la coda di errore per il messaggio di convalida e il file delle eccezioni per un messaggio di eccezione. Se la creazione di traccia è stata abilitata, controllare i messaggi di traccia.
  - Se il messaggio di convalida è contenuto nella coda di errore e in un messaggio di eccezione, le voci nel file di configurazione sono state convalidate.
  - In caso contrario, controllare il file delle eccezioni. Se la creazione di traccia è stata abilitata, controllare i messaggi di traccia.

Passo 13. Se necessario, modificare il file di configurazione e convalidarlo di nuovo.

## Configurazione di MQSeries e di MQSeries Integrator

Configurare MQSeries ed il software facoltativo MQSeries Integrator per supportare il kernel nel modo seguente:

In MQSeries:

- Molte code vengono utilizzate per la verifica dell'installazione. Se vengono utilizzate tali code per la prova o gli ambienti di produzione, ripulirle per verificare l'installazione. Per informazioni sulle code utilizzate per la verifica dell'installazione, consultare "Procedura della verifica" a pagina 47.
- Impostare le code per il supporto del trasporto dei messaggi in base allo schema di instradamento stabilito.
- Quando si creano le code, impostare la variabile di ambiente MAX_QUEUE_DEPTH sulle dimensioni massime della coda consentite.

In MQSeries Integrator, impostare le code di immissione ed emissione nelle regole (versione 1.1) o nei flussi di messaggi (versione 2) che corrispondono alle code configurate nel file di configurazione.

## Consigli sulle prestazioni

Di seguito vengono riportati alcuni consigli relativi al livello delle prestazioni, che si applicano specificamente a MQSeries Adapter Kernel:

- Quando si analizzano i DTD XML, assicurarsi che i file DTD si trovino nella stessa directory del processo che li analizza. In tal modo è possibile agevolare il processo nella ricerca dei DTD.
- Quando si inviano e ricevono messaggi di grandi dimensioni, è preferibile utilizzare il tipo di messaggio RFH2 piuttosto che XML.

Per consigli di carattere generale sulle prestazioni, consultare la documentazione relativa a MQSeries.

## Avvio del kernel

Per avviare il kernel, avviare i seguenti processi:

- Il daemon dell'adattatore per ciascuna applicazione di destinazione
- Server di traccia (facoltativo)

Se l'adattatore di origine viene eseguito nel processo dell'applicazione di origine, l'adattatore di origine viene avviato automaticamente con l'applicazione di origine; non sono richieste operazioni aggiuntive per avviare l'adattatore di origine. È necessario avviare tutti i daemon o i server che contengono l'adattatore di origine. Gli adattatori di origine non vengono avviati direttamente.

Avviare i singoli daemon dell'adattatore e il server di traccia completando la procedura riportata di seguito:

**Nota:** Per agevolare questa procedura, vengono forniti numerosi script. Se si desidera, copiare gli script e modificare le copie per creare delle versioni personalizzate. Se si utilizza OS/400, le versioni UNIX degli script possono essere eseguite in una sessione **qsh**. È possibile avviare una sessione **qsh** immettendo il comando Start QSH (**STRQSH**) da un prompt CL. prompt.

Passo 1. Avviare MQSeries o un diverso software di messaggistica e il software facoltativo, quale MQSeries Integrator.

Passo 2. Avviare tutti i software associati richiesti per il proprio sito — ad esempio, applicazioni (esterne al kernel) per leggere i messaggi di traccia dalle code.

Passo 3. Aprire un prompt dei comandi. Per ciascun daemon dell'adattatore, immettere il seguente comando:

- Sui sistemi Windows:

  ```
  aqmstrad.bat -a identificativo_applicazione [-bc categoria_testo
  -bt tipo_testo] [-noretry]
  ```

- Su UNIX e OS/400:

  ```
  aqmstrad.sh -a identificativo_applicazione [-bc categoria_testo
  -bt tipo_testo] [-noretry]
  ```

  dove

  **-a** *identificativo_applicazione*
  Indica l'identificativo logico di destinazione servito dal daemon dell'adattatore.

  **-bc** *categoria_testo*
  Specifica la categoria testo utilizzata dal worker del daemon dell'adattatore per determinare la modalità di comunicazione e le informazioni correlate per la ricezione dei messaggi. Se non viene specificato alcun valore, il daemon dell'adattatore utilizza il valore DEFAULT.

  **-bt** *tipo_testo*
  Specifica il tipo testo utilizzato dal worker del daemon dell'adattatore per determinare la modalità di comunicazione e le informazioni correlate per la ricezione dei messaggi. Se non viene specificato alcun valore, il daemon dell'adattatore utilizza il valore DEFAULT.

  **-noretry**
  Specifica che il worker si arresta automaticamente quando non ci sono più messaggi. Se `-noretry` non viene specificato, il worker richiede continuamente i messaggi alla coda e il daemon dell'adattatore deve essere arrestato manualmente.

  **Nota:** Se occorre modificare i parametri di avvio Java, editare il file `aqmstrad.bat` (sistemi Windows) o il file `aqmstrad.sh` (UNIX e OS/400). Per informazioni più dettagliate, leggere i commenti contenuti all'interno dei file.

Passo 4. Per ciascun server di traccia, immettere il seguente comando:

- Sui sistemi Windows:

  ```
  aqmstrtd.bat -modalità di ricezione -a identificativo_applicazione_origine
  ```

- Su UNIX e OS/400:

  ```
  aqmstrtd.sh -modalità di ricezione -a identificativo_applicazione_origine
  ```

  dove:

*-modalità di ricezione*
Indica la modalità di ricezione dei messaggi di traccia. È possibile specificare i seguenti valori:

– `socket`

– `ena`, ossia, l'adattatore nativo

**-a** *identificativo_applicazione_origine*
L'identificativo dell'applicazione di origine. Nel caso in cui non venga fornito alcun valore, viene utilizzato il valore predefinito `TraceServer` contenuto nel file di configurazione.

Per ulteriori informazioni sui server di traccia, consultare *Problem Determination Guide*.

Passo 5. Dopo aver avviato il daemon dell'adattatore o il server di traccia, una finestra del processo resta aperta fino all'arresto del daemon dell'adattatore. La finestra del processo può visualizzare le eccezioni. Consultare "Messaggi di eccezione" a pagina 96.

### Arresto del kernel

Per arrestare il kernel, arrestare i daemon dell'adattatore e i server di traccia. Vi sono vari metodi per arrestarli:

- Quando si avvia il daemon dell'adattatore, impostare il parametro `-noretry`. Consultare "Avvio del kernel" a pagina 92.
- Passare al prompt dei comandi (sistemi Windows) o al terminale (UNIX) da cui è stato avviato il daemon dell'adattatore o il server di traccia e premere **Ctrl-C**. Eseguire questa operazione per ogni daemon o server di traccia.
- Sui sistemi Windows, è possibile utilizzare Task Manager per arrestare il processo.
- Su UNIX, è possibile utilizzare il comando **ps** per determinare il numero del processo, quindi eseguire il comando **kill** per interromperlo.

## Gestione del kernel

Definire un piano per la gestione del kernel. Si consiglia di eseguire periodicamente il back up dei seguenti elementi:

- La configurazione come specificata nei seguenti file:
  – `aqmconfig.xml`
  – `aqmsetup`
- Gli adattatori che sono stati creati e i file associati

L'esecuzione del backup oppure la rimozione periodica del contenuto del file di traccia o di altri file utilizzati dal kernel per supportare l'elaborazione, non è richiesta. Se si desidera, è possibile eseguire il backup di questi file. Se i messaggi di traccia vengono instradati verso un singolo file anziché più file, le

dimensioni di questo file aumentano notevolmente. Se il livello di traccia è stato impostato per ottenere un livello elevato di dettagli (ad esempio, tutti i messaggi di traccia e di informazioni), è consigliabile eliminare periodicamente i file di traccia.

## Diagnostica dei problemi

È possibile utilizzare i messaggi di eccezione, di traccia e le code di errore di MQSeries per agevolare la diagnosi dei problemi. MQSeries Adapter Kernel produce messaggi di eccezione e, se la traccia è abilitata, genera anche messaggi di traccia. Per informazioni sulle modalità di diagnostica dei problemi in un ambiente MQSeries Adapter Kernel, consultare *Problem Determination Guide*.

Per comprendere i messaggi di eccezione e di traccia, è necessario conoscere il funzionamento del kernel. Il kernel utilizza una coda di errore per la gestione di alcuni errori. Consultare "Descrizione del funzionamento del kernel" a pagina 9.

È possibile identificare il messaggio che ha determinato i messaggi di eccezione e di traccia, esaminando la combinazione dell'identificativo univoco del messaggio e dell'identificativo univoco della transazione.

Non esiste alcun identificativo che consente di identificare in maniera definitiva i messaggi sia nella coda di errore che nel kernel. Tuttavia, è possibile correlare manualmente un messaggio nella coda di errore al corrispondente messaggio di eccezione o di traccia, oppure a entrambi. È possibile confrontare uno o più dei seguenti elementi:

- Timestamp approssimativo
- Coda per l'identificativo logico di origine
- Coda per l'identificativo logico di destinazione
- Categoria di testo
- Tipo di testo
- Identificativo univoco del messaggio
- Identificativo univoco della transazione

Se corrispondono, è possibile che il messaggio nella coda di errore sia stato correlato al corrispondente messaggio di eccezione o di traccia.

### Numero di versione

Eseguire `aqmversion.bat` (sistemi Windows) o `aqmversion.sh` (UNIX e OS/400) nella directory `bin` per visualizzare il numero di versione del kernel.

## Messaggi di eccezione

Il kernel crea i seguenti tipi di messaggi di eccezione:

- L'adattatore nativo sul lato di origine del kernel lancia eccezioni all'adattatore di origine. Per informazioni sulle modalità di gestione di queste eccezioni da parte dell'adattatore di origine, consultare la documentazione relativa a MQSeries Adapter Builder.
- L'adattatore nativo sul lato di destinazione del kernel lancia eccezioni al worker che gestisce l'adattatore nativo.
- Il worker scrive eccezioni nel file `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, che si trova nella stessa directory del worker.
- Il daemon dell'adattatore scrive messaggi di eccezione in un file delle eccezioni chiamato `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, che si trova nella stessa directory del daemon dell'adattatore. Dal momento che il daemon dell'adattatore e i worker si trovano nella medesima directory, utilizzano lo stesso file delle eccezioni. Il daemon dell'adattatore scrive i messaggi di eccezione nella console (cioè, il prompt dei comandi o il terminale utilizzato per avviarlo, se è stato avviato da una finestra).

I messaggi di eccezione di traccia del kernel si differenziano dai messaggi di eccezione di MQSeries. Di seguito viene riportato un esempio di messaggio di eccezione lanciato dal kernel:

```
2000.10.26 19:38:20.929 com.ibm.epic.adapters.eak.nativeadapter.LMSMQ
Nome thread=main receiveRequest(ENAService)  ePIC TEST2
TYPE_ERROR_EXC AQM5004: Ricevuta eccezione <com.ibm.epic.adapters.eak.common.
AdapterException> Informazioni messaggio: <AQM0114: com.ibm.epic.adapters.eak.
nativeadapter.MQNMRFH2Formatter::convertMessage(MQMessage): Previsto messaggio
con formato MQHRF2 e ricevuto messaggio con formato <MQSTR   >.>
per <unmarshall Message()> che dispone di dati non validi <(null)>
```

I valori di un messaggio di eccezione dipendono dalla natura del messaggio e solitamente includono:

- Timestamp
- Identificativo logico di origine
- Identificativo logico di destinazione
- Categoria di testo
- Tipo di testo
- Identificativo univoco del messaggio
- Identificativo univoco della transazione
- Informazioni di eccezione

Per informazioni sui problemi che possono verificarsi durante la prova dell'installazione e sulle possibili soluzioni da applicare, consultare "Problemi comuni di verifica" a pagina 48.

## Messaggi di traccia

È possibile configurare il kernel per la creazione di messaggi di traccia. Per ulteriori informazioni, consultare *Problem Determination Guide*.

## Utilità

### Creazione di code di MQSeries

Per rendere automatica la creazione delle code di MQSeries, è possibile utilizzare file batch o script della shell. Eseguire `aqmcreateq.bat` (sistemi Windows) o `aqmcreateq.sh` (UNIX e OS/400), utilizzando come parametro il nome dell'applicazione. Questi file creano per ciascuna applicazione le seguenti code:

- Coda di ricezione, chiamata *nome_applicazione*AIQ.
- Coda di errore, chiamata *nome_applicazione*AEQ.
- Coda di risposta, chiamata *nome_applicazione*RPL.

# Capitolo 5. Utilizzo delle API di MQSeries Adapter Kernel

Il kernel include API che vengono utilizzate per eseguire operazioni quali l'invio e la ricezione di messaggi, la creazione e l'analisi XML e la gestione della configurazione del kernel. Queste API vengono utilizzate dagli adattatori creati mediante MQSeries Adapter Builder. MQSeries Adapter Kernel Information Center fornisce una documentazione in linea relativa alle API nel formato Javadoc HTML.

Il kernel è stato progettato per essere utilizzato con adattatori creati dall'utente mediante MQSeries Adapter Builder. Il kernel non è stato creato per essere utilizzato mediante chiamate alle API del kernel dal solo codice personalizzato. La documentazione in linea relativa alle API viene fornita unicamente per facilitare la comprensione delle funzioni del kernel e per agevolare le attività di diagnostica.

Questa documentazione è disponibile nella directory `documentation`.

# Capitolo 6. Informazioni aggiuntive sul prodotto

Vi sono numerose fonti di informazioni di cui è possibile servirsi quando si utilizza MQSeries Adapter Offering. Per informazioni aggiuntive su MQSeries Adapter Kernel, consultare il documento *Problem Determination Guide*, disponibile in MQSeries Adapter Kernel Information Center, installato insieme al prodotto. *Problem Determination Guide* fornisce informazioni sulla risoluzione di problemi specifici che generalmente si verificano quando si utilizza il kernel. Per ulteriori dettagli su MQSeries Adapter Builder, fare riferimento all'Information Center del prodotto e all'aiuto in linea.

## Documentazione disponibile su Internet

Il sito Web della famiglia dei prodotti MQSeries è all'indirizzo www.ibm.com/software/ts/mqseries/. Seguendo i collegamenti da questo sito, è possibile:

- Visualizzare le informazioni più recenti sulla famiglia di prodotti MQSeries, compreso MQSeries Adapter Offering.
- I manuali MQSeries in formato HTML e PDF potrebbero includere informazioni più recenti. Il collegamento diretto alla pagine della libreria MQSeries è all'indirizzo www.ibm.com/software/ts/mqseries/library/manualsa/.
- Scaricare i SupportPac di MQSeries.

Per informazioni sull'utilizzo di MQSeries su OS/400, consultare la libreria OS/400 all'indirizzo www.ibm.com/servers/eserver/iseries/library/. Consultare anche i manuali disponibili specifici per OS/400 al sito Web della libreria MQSeries all'indirizzo www.ibm.com/software/ts/mqseries/library/manualsa/.

## Materiale di riferimento

Il materiale di riferimento elencato di seguito tratta gli argomenti illustrati nel presente testo:

- Per visitare il sito Web dell'Open Applications: www.openapplications.org/
- Per la guida Extensible Markup Language (XML) 1.0 W3C Recommendation andare all'indirizzo www.w3.org/TR/1998/Rec-xml-19980210

Questi non sono siti Web IBM.

# Appendice A. Modalità di comunicazione

Questa appendice fornisce informazioni sulle modalità di comunicazione supportate da MQSeries Adapter Kernel e sulle classi Java utilizzate per supportarle. Alcune modalità di comunicazione sono fornite come modalità facilitate per i programmi di formattazione predefiniti. Per un elenco dei programmi di formattazione utilizzati con le modalità facilitate, consultare Tabella 10 a pagina 105.

Sono supportate le seguenti modalità di comunicazione:

**MQPP** Il kernel trasporta i messaggi utilizzando i servizi di base di MQSeries. Questa è una modalità facilitata.

**MQRFH1** Il kernel trasporta i messaggi utilizzando messaggi del broker ed MQSeries mediante MQSeries Integrator versione 1.1. Questa è una modalità facilitata.

**MQRFH2** Il kernel trasporta i messaggi utilizzando messaggi del broker ed MQSeries mediante MQSeries Integrator versione 2. Questa è una modalità facilitata.

**MQBD** Il kernel trasporta i messaggi utilizzando i servizi di base di MQSeries ma invia e riceve solo dati di testo. Questa è una modalità facilitata. Di seguito vengono descritte le funzioni specifiche di questa modalità:

- Può inviare solo dati di testo, non i valori dell'intestazione dei messaggi.
- Può ricevere messaggi che contengono solo dati di testo. Per i messaggi ricevuti, utilizza i seguenti valori di intestazione messaggi predefiniti:
  - `SourceLogicalApplicationID`—Il valore dell'oggetto `ENAService` utilizzato per richiamare il metodo di ricezione.
  - `BodyCategory`—Il valore dell'oggetto `ENAService` utilizzato per richiamare il metodo di ricezione.
  - `BodyType`—Il valore dell'oggetto `ENAService` utilizzato per richiamare il metodo di ricezione.
  - `Acknowledgment`—Se il MQMessage ricevuto è un MQSeries REQUEST, `Acknowledgment` è impostato su 1.
  - `BodyData`—I dati del messaggio ricevuto da MQSeries.

Per tutte le altre impostazioni vengono utilizzati i normali valori predefiniti.

**MQ** Il kernel trasporta messaggi utilizzando i servizi di base di MQSeries.

**JMS** Il kernel trasporta i messaggi utilizzando JMS (Java Message Service). Per informazioni sulle modalità di utilizzo degli oggetti JMS con MQSeries Adapter Kernel, consultare "Memorizzazione degli oggetti JMS" a pagina 106.

**FILE** Il kernel inserisce e richiama i messaggi da un file. Questa modalità viene fornita solo per scopi diagnostici.

Tabella 9 elenca le modalità di comunicazione e le classi Java che le supportano. Tutte le classi Java sono contenute nel pacchetto Java `com.ibm.epic.adapters.eak.nativeadapter`. Si noti che le classi Java che supportano LMS (Logical Message Service) possono essere specificate come modalità di comunicazione, in questo caso, la classe stessa è utilizzata per supportare la comunicazione.

*Tabella 9. Modalità di comunicazione e classi Java di supporto*

| Modalità di comunicazione | Classe Java | Note |
|---|---|---|
| MQPP | `LMSMQBindingMQPP` | Richiede l'installazione di MQSeries |
| MQRFH1 | `LMSMQBindingMQRFH1` | Richiede l'installazione di MQSeries |
| MQRFH2 | `LMSMQBindingMQRFH2` | Richiede l'installazione di MQSeries |
| MQBD | `LMSMQMQBD` | Richiede l'installazione di MQSeries |
| MQ | `LMSMQBinding` | Richiede l'installazione di MQSeries |
| JMS | `LMSJMS` | Richiede l'installazione di JMS |
| FILE | `LMSFile` | Nessuna |

Tabella 10 a pagina 105 elenca le modalità di comunicazione e le relative interfacce dei programmi di formattazione associati. Tabella 11 a pagina 105 interfacce dei programmi di formattazione di riferimenti incrociati, i nomi della classe del programma di formattazione ed i relativi utilizzi. Tutti i programmi di formattazione sono contenuti nel pacchetto Java `com.ibm.epic.adapters.eak.nativeadapter`. Per la modalità di comunicazione, è possibile specificare una qualunque classe programma di formattazione; in

questo caso, la classe programma di formattazione viene utilizzata come programma di formattazione.

*Tabella 10. Modalità di comunicazione ed interfacce dei programmi di formattazione*

| Modalità di comunicazione | Interfaccia del programma di formattazione | Programma di formattazione predefinito |
|---|---|---|
| MQPP | MQFormatterInterface | MQNMXMLFormatter |
| MQRFH1 | MQFormatterInterface | MQNMRFH1Formatter |
| MQRFH2 | MQFormatterInterface | MQNMRFH2Formatter |
| MQBD | MQFormatterInterface | MQNMBDFormatter |
| MQ | MQFormatterInterface | MQNMXMLFormatter |
| JMS | JMSFormatterInterface | JMSNMRFH2Formatter |
| FILE | StringFormatterInterface | NMXMLFormatter |

*Tabella 11. Interfacce del programma di formattazione, nomi classi del programma di formattazione e finalità.*

| Interfaccia del programma di formattazione | Nome classe del programma di formattazione | Finalità |
|---|---|---|
| MQFormatterInterface | MQNMXMLFormatter | EpicMessage come XML |
| | MQNMRFH1Formatter | EpicMessage come RFH1 |
| | MQNMRFH2Formatter | EpicMessage come RFH2 |
| | MQNMDBFormatter | Solo dati di testo |
| JMSFormatterInterface | JMSNMXMLFormatter | EpicMessage come XML |
| | JMSNMRFH2Formatter | EpicMessage come RFH2 |
| | JMSNMBDFormatter | Solo dati di testo |
| StringFormatterInterface | NMXMLFormatter | EpicMessage come XML |

Tabella 12 elenca le classi LMS supportate e il relativo livello di supporto transazionale. Per informazioni sulle modalità di utilizzo delle transazioni con MQSeries Adapter Kernel, consultare “Funzioni transazionali” a pagina 28.

*Tabella 12. Classi LMS e supporto transazionale*

| Classe LMS | Supporto transazionale |
|---|---|
| LMSMQBindingMQPP | Fase singola |
| LMSMQBindingMQRFH1 | Fase singola |
| LMSMQBindingMQRFH2 | Fase singola |
| LMSMQMQBD | Fase singola |

*Tabella 12. Classi LMS e supporto transazionale (Continua)*

| Classe LMS | Supporto transazionale |
|---|---|
| LMSMQBinding | Fase singola |
| LMSJMS | Fase singola |
| LMSFILE | Nessun supporto |

## Memorizzazione degli oggetti JMS

I nomi degli oggetti JMS vengono memorizzati mediante l'implementazione del file FSContext di JNDI, disponibile con il SupportPac JMS di MQSeries. Il contesto (struttura delle directory) utilizzato dal kernel per FSContext segue la gerarchia LDAP utilizzando l'attributo di distinzione con il valore associato per il nome della directory. Ad esempio, per la gerarchia LDAP `o=ePIC, o=ePICApplications, epicappid=TEST1`, la struttura delle directory è `o-ePIC/o-ePICApplications/epicappid-TEST1`.

Per creare il contesto e gli oggetti, utilizzare lo strumento JMS Admin fornito con l'installazione JMS. La definizione e la modifica di un contesto rappresentano le procedure più importanti. La modifica del contesto introduce l'utente all'interno del contesto. Creare gli oggetti JMS nella posizione appropriata. Di seguito vengono riportati esempi di comandi per la creazione della struttura di contesto e di oggetti JMS. In questo esempio, l'ID di applicazione è `TEST1`.

```
#
# aqmjmscreatesample.scp 1.00 09Mar01
# Utilizzato per MQSeries Adapter Kernel
# Esempio di configurazione AQM JMS.
#
# Copyright (c) 2001 International Business Machines. Tutti i diritti riservati.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# This is a script to use with the JMS administration (JMSAdmin) tool
# which comes with MQSeries Support pac MA88.
# This tool requires the JMSAdmin.config to be set to use either
# FSCONTEXT (file) or LDAP. This script is setup to work with
# FSCONTEXT, but will work with LDAP with the following changes:
# - Change the "-" signs to "=".  Example: define ctx(o-ePIC)
#   becomes define ctx(o=ePIC)
```

```
# - In LDAP the contexts have to already be defined using the
#   LDAP administration tool.  For example you do not need
#   to "define ctx(o=ePIC) but only change into it with the
#   "change ctx(o-ePIC)" command.
# - There are some notes in the following script which highlight
#   differences when using LDAP.
#
#
# Example usage: MQSeries root\java\bin\jmsadmin.bat < aqmjmscreatesample.scp
#
# Some helpful commands:
# "display ctx" will display the context's of the context you are
# currently in.
# "=UP" means return to the parent context.  Example: change ctx(=UP)
# "=INIT" means return to root context. In this example one directory level
# above o-ePIC.  Example: change ctx(=INIT)
# "define xxx" is for creating either a context or object.
# "change xxx" is for changing/moving into the context.
#
# Always required.
define ctx(o-ePIC)
change ctx(o-ePIC)
# Always required.
define ctx(o-ePICApplications)
change ctx(o-ePICApplications)
# Application id is TEST1, requires a context.
define ctx(epicappid-TEST1)
change ctx(epicappid-TEST1)
# Always required.
define ctx(cn-epicadapterrouting)
change ctx(cn-epicadapterrouting)
# This will hold the JMS QueueConnectionFactory object.
# Note: These two steps are not required for LDAP.
define ctx(cn-QCFTEST1)
change ctx(cn-QCFTEST1)
# Create the JMS QueueConnectionFactory object whose name is QCFTEST1
# Using MQSeries in server (bindings) mode.
define qcf(QCFTEST1) qmgr(yourQManagerName) tran(BIND)
change ctx(=UP)
# BodyCategory is DEFAULT
define ctx(epicbodycategory-DEFAULT)
change ctx(epicbodycategory-DEFAULT)
# BodyType is DEFAULT
define ctx(epicbodytype-DEFAULT)
change ctx(epicbodytype-DEFAULT)
# This will hold the JMS Queue object whose name is TEST1AIQ.
# Note: These two steps are not required for LDAP.
define ctx(cn-TEST1AIQ)
change ctx(cn-TEST1AIQ)
# Create the JMS Queue object whose name is TEST1AIQ
# q(JMS Q Object Name) queue(MQSeries Queue name)
define q(TEST1AIQ) queue(TEST1AIQ)
# Can move up and define other contexts and JMS objects.
# Quit the administration tool.
end
```

# Appendice B. Configurazioni convalidate

Esistono varie possibilità di configurare e combinare i prodotti MQSeries, MQSeries Adapter Offering e MQSeries Integrator. Ciascuno di questi componenti della famiglia di prodotti di MQSeries offre una vasta gamma di funzioni e configurazioni. Inoltre, è anche possibile combinare le funzionalità di MQSeries, di MQSeries Adapter Offering e di MQSeries Integrator. Alcune funzionalità di un prodotto possono parzialmente sovrapporsi alle funzionalità fornite da altri componenti della stessa famiglia di prodotti. È necessario stabilire il modo in cui utilizzare e combinare le varie funzioni per l'instradamento e la consegna dei messaggi in MQSeries, MQSeries Adapter Offering e MQSeries Integrator.

Di seguito vengono riportate le configurazioni di MQSeries, di MQSeries Adapter Offering e di MQSeries Integrator convalidate al momento della pubblicazione di questo testo. Per le configurazioni convalidate più recenti, visitare il sito Web di MQSeries.

**MQSeries Adapter Kernel:**

- Invio di un messaggio con e senza richiesta di conferma.
- Utilizzo delle modalità di comunicazione JSM o MQSeries. Per le modalità valide di comunicazione, vedere "Appendice A. Modalità di comunicazione" a pagina 103.
- Instradamento e consegna dei messaggi:
  - Invio di un messaggio da un adattatore di origine a un adattatore di destinazione
  - Invio di un messaggio da un adattatore di origine a più adattatori di destinazione
  - Consegna di messaggi multithread, vale a dire, più worker
  - Impostazione dell'identificativo logico di destinazione su NONE nel messaggio, in modo che il file di configurazione del kernel venga utilizzato per determinare l'identificativo logico di destinazione in base alla categoria di testo, al tipo di testo e all'identificativo logico di origine.
  - Modello push per la consegna
  - Con la traccia abilitata

**Nota:** Consultare "Appendice C. Intestazioni di messaggi" a pagina 111. Questa appendice riporta i campi di intestazione dei messaggi di MQSeries Adapter Kernel che il kernel popola ed elabora.

- Applicazione dei prerequisiti indicati in “Hardware” a pagina 31 e “Software” a pagina 32.
- Utilizzo del file di configurazione, non LDAP, per definire la configurazione.

**MQSeries:**

- Senza utilizzare i cluster di MQSeries.

  **Nota:** Consultare “Appendice C. Intestazioni di messaggi” a pagina 111. Questa appendice riporta i campi di intestazione dei messaggi di MQSeries Adapter Kernel che il kernel popola ed elabora.

**MQSeries Integrator:**

- MQSeries Adapter Kernel e MQSeries possono instradare e consegnare il messaggio a MQSeries Integrator. Consultare le informazioni relative a MQSeries Integrator per determinarne la capacità di esecuzione del brokering di questi messaggi.
- Invio di messaggi dal lato di origine del kernel mediante MQSeries e MQSeries Integrator versione 2 e instradamento direttamente sul lato di destinazione del kernel. All’interno di MQSeries Integrator, il flusso dei messaggi viene configurato per l’instradamento in maniera statica. Tutti i messaggi che giungono al nodo MQInput del flusso vengono instradati direttamente in una coda specifica di MQOutput.

  **Nota:** Consultare “Appendice C. Intestazioni di messaggi” a pagina 111. Questa appendice riporta i campi di intestazione dei messaggi di MQSeries Adapter Kernel che il kernel popola ed elabora.

# Appendice C. Intestazioni di messaggi

MQSeries Adapter Offering utilizza numerose intestazioni di messaggi. Per scegliere le intestazioni e stabilire quando utilizzarle, consultare “Messaggio e formato del messaggio” a pagina 12.

Questa appendice elenca e descrive i campi di intestazione.

## Intestazione del descrittore messaggi di MQSeries Adapter Kernel

I valori di intestazione utilizzati da MQSeries Adapter Kernel. Questi valori sono posti in oggetti contenenti i messaggi. La colonna **Propagati in risposte?** elenca se il valore viene restituito o meno all’applicazione di origine in un messaggio di risposta quando l’applicazione di origine richiede una risposta. Alcuni valori sono utilizzati solo con WebSphere Business Integrator.

*Tabella 13. Intestazione di MQSeries Adapter Kernel*

| Nome intestazione | Propagati in risposte? | Significato o utilizzo |
|---|---|---|
| UniqueID | No | Identificativo univoco per ciascun messaggio. |
| TransactionID | Sì | Identificativo di transazione condiviso da ciascun messaggio e la relativa risposta, se presente. Equivalente a un Extricity PublicProcessID o un DataInterchange (DI) ApplicationID. |
| MessageType | No | Utilizzato per gateway e messaggi di log/traccia/eccezione. |
| SourceLogicalID | No | Identificativo logico dell’applicazione di origine. Equivalente ai nomi riservati in DI, PAM (Partner Agreement Manager) e BFM (Business Flow Manager). |

*Tabella 13. Intestazione di MQSeries Adapter Kernel (Continua)*

| Nome intestazione | Propagati in risposte? | Significato o utilizzo |
|---|---|---|
| DestinationLogicalID | No | Identificativo logico dell'applicazione di destinazione. Per DI e PAM, il valore predefinito è `nessuno`, ma può essere sovrascritto. |
| RespondToLogicalID | Sì | Identificativo logico a cui viene inviato un messaggio di risposta. Copiato in DestinationLogicalID per DI e in SourceLogicalID per PAM. |
| CorrelationID | No | Uso riservato. |
| GroupStatus | No | Uso riservato. |
| ProcessingCategory | No | Equivalente a PAM Public Process Identifier o a DI Command Process Identifier. |
| QosPolicy | No | Uso riservato. |
| DeliveryCategory | No | Equivalente a DI RequestorProfileID. |
| AckRequested | No | Determina se l'applicazione di origine richiede o meno un messaggio di risposta. |
| PublicationTopic | No | Uso riservato. |
| SessionID | No | Equivalente a DI BatchID. |
| EncryptionStatus | No | Determina il tipo di crittografia e firma del corpo. |
| TimeStampCreated | No | L'ora e la data di creazione del messaggio. |
| TimeStampExpired | No | L'ora e la data dopo le quali il messaggio non è più valido. Un valore di `-1` indica l'assenza di una scadenza. |
| Size | No | Uso riservato. |
| BodyType | No | Rappresenta la finalità specifica del messaggio. |

*Tabella 13. Intestazione di MQSeries Adapter Kernel (Continua)*

| Nome intestazione | Propagati in risposte? | Significato o utilizzo |
|---|---|---|
| BodyCategory | No | Rappresenta il tipo dell'applicazione del messaggio. |
| BodySecondaryType | No | Uso riservato. |
| UserArea | No | Area generale per i dati utente. |
| RelatedSubjectID | No | Utilizzato per la correlazione tra i processi. |
| ExternalID | No | Identificativo del possessore corrente (ad esempio, un utente o un partner commerciale) esterno all'ambiente dell'applicazione. |
| InternalID | No | Identificativo del possessore corrente (ad esempio, un utente o un partner commerciale) interno all'ambiente dell'applicazione. |
| BodySignature | No | Uso riservato. |
| TransportCorrelationID | Sì | Uso riservato. |

## Intestazione del descrittore di messaggi di MQSeries

Il contenuto del campo viene determinato da MQSeries. MQSeries Adapter Offering colloca i messaggi in code in base ai valori di controllo messaggio. Per dettagli, consultare "Valori di controllo messaggi" a pagina 16.

*Tabella 14. Intestazione di MQSeries*

| Sezione o campo | Significato o uso |
|---|---|
| Revision | Aggiornamento. |
| UniqueID | Ogni messaggio dispone di un identificativo univoco. |
| TransactionID | Il messaggio e la relativa risposta condividono lo stesso identificativo. |
| MessageType | Uso riservato. |
| SourceLogicalID | Identificativo logico dell'applicazione di origine. |

*Tabella 14. Intestazione di MQSeries (Continua)*

| | |
|---|---|
| DestinationLogicalID | Identificativo logico dell'applicazione di destinazione. |
| RespondToLogicalID | Un identificativo logico al quale inviare il messaggio di risposta. |
| CorrelationID | Uso riservato. |
| GroupStatus | Uso riservato. |
| ProcessingCategory | Uso riservato. |
| QosPolicy | Uso riservato. |
| DeliveryCategory | Uso riservato. |
| AckRequested | Determina se l'applicazione di origine richiede una risposta o meno. |
| PublicationTopic | Uso riservato. |
| SessionID | Uso riservato. |
| EncryptionStatus | Uso riservato. |
| TimeStampCreated | L'ora e la data di creazione del messaggio. |
| TimeStampExpired | L'ora e la data dopo le quali il messaggio non è più valido. |
| Size | Uso riservato. |
| BodyCategory | Rappresenta il tipo di applicazione del messaggio: ad esempio, OAG o RosettaNet. |
| BodyType | Rappresenta lo scopo specifico del messaggio: ad esempio, aggiungere un ordine di vendita o sincronizzare l'inventario. |
| BodySecondaryType | Riservato. |
| UserArea | Area generale per i dati utente. |
| BodyData | Dati di testo del messaggio. |

## MQSeries senza MQSeries Integrator

I valori di intestazione del kernel e i dati di testo vengono inseriti in un documento XML. Di seguito viene riportato un esempio di DTD che descrive il documento XML:

```
<!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) >
```

## Intestazione di MQSeries Integrator versione 1

L'intestazione di MQSeries Integrator versione 1, RFH1, è composta dai seguenti elementi:

1. Parte fissa
2. Intestazione Neon
3. Sezione di dati, che contiene l'intestazione del kernel e i dati di testo del messaggio

*Tabella 15. Intestazione di MQSeries Integrator versione 1 — RFH1*

| **Sezione o campo** | **Significato o uso** |
|---|---|
| Parte fissa | Utilizzata come specificato in MQSeries Integrator versione 1.1. |
| Intestazione Neon | Segue il formato dell'intestazione Neon. |
| OPT_APP_GRP | Valore SourceLogicalId. Ricavato dall'intestazione del kernel. |
| OPT_MSG_TYPE | BodyCategory+BodyType. Derivato dall'intestazione del kernel.<br><br>Esempio: Se BodyCategory è OAG e BodyType è SyncItem, il valore è OAG+SyncItem. |
| Sezione dei dati | È composta dai valori dell'intestazione del kernel seguiti dai dati del testo del messaggio. |

*Tabella 15. Intestazione di MQSeries Integrator versione 1 — RFH1 (Continua)*

| | |
|---|---|
| Intestazione del kernel | L'intestazione del kernel è racchiusa tra tag <EPICHEADER>*intestazione*</EPICHEADER>.<br><br>I valori dell'intestazione del kernel sono forniti nella sintassi XML. Sono presenti solo attributi con valori. I dati effettivi non si trovano in righe distinte. Esempio del formato di un valore: <MessageType>*valore*</MessageType>. |
| MessageType | Uso riservato. |
| SourceLogicalID | Identificativo logico dell'applicazione di origine. |
| DestinationLogicalID | Identificativo logico dell'applicazione di destinazione. |
| RespondToLogicalID | Identificativo logico al quale inviare il messaggio di risposta. |
| TimeStampCreated | L'ora e la data di creazione del messaggio. |
| TimeStampExpired | L'ora e la data dopo le quali il messaggio non è più valido. |
| TransactionID | Il messaggio e la relativa risposta condividono lo stesso identificativo. |
| UniqueID | Ogni messaggio dispone di un identificativo univoco. |
| AckRequested | Determina se l'applicazione di origine richiede una risposta. |
| ProcessingCategory | Riservato. |
| BodyCategory | Rappresenta il tipo di applicazione del messaggio: ad esempio, OAG o RosettaNet. |
| BodyType | Rappresenta lo scopo specifico del messaggio: ad esempio, aggiungere un ordine di vendita o sincronizzare l'inventario. |
| BodySecondaryType | Riservato. |
| UserArea | Dati dell'applicazione specifici dell'integrazione utente. |
| MsgHeaderVersion | Versione dell'intestazione del kernel (riservato). |
| CorrelationID | Specifico dell'integrazione utente. |
| GroupStatus | Specifico dell'integrazione utente. |
| QosPolicy | Riservato. |
| DeliveryCategory | Riservato. |
| PublicationTopic | Riservato. |
| SessionID | Riservato. |

*Tabella 15. Intestazione di MQSeries Integrator versione 1 — RFH1 (Continua)*

| | |
|---|---|
| EncryptionStatus | Riservato. |
| Dati di testo del messaggio | Dati di testo del messaggio. |

## Intestazione di MQSeries Integrator versione 2

L'intestazione di MQSeries Integrator versione 2, RFH2, è composta dai seguenti elementi:

1. Parte fissa
2. <cartella mcd> — descrittore contenuto messaggi
3. <cartella usr> — proprietà definite dall'applicazione (utente)
4. Sezione di dati, che contiene l'intestazione del kernel e i dati di testo del messaggio

*Tabella 16. Intestazione di MQSeries Integrator versione 2 — RFH2*

| **Sezione o campo** | **Significato o uso** |
|---|---|
| Parte fissa | Utilizzata come specificato in MQSeries Integrator versione 2. |
| <mcd> | XML se il messaggio è XML. Seguire le regole di MQSeries Integrator versione 2. |
| impostazione | Non utilizzato dal kernel. |
| tipo | Non utilizzato dal kernel. |
| formato | XML se il messaggio è XML. Seguire le regole di MQSeries Integrator versione 2. |
| <cartella usr> — proprietà definite dall'applicazione (utente) | È composta dai valori di intestazione del kernel. |
| Intestazione del kernel | Sono presenti solo attributi con valori. I dati effettivi non si trovano in righe distinte. |
| SourceLogicalID | Identificativo logico dell'applicazione di origine. |
| DestinationLogicalID | Identificativo logico dell'applicazione di destinazione. |
| MessageType | Uso riservato. |
| RespondToLogicalID | Un identificativo logico al quale inviare il messaggio di risposta. |
| TimeStampCreated | L'ora e la data di creazione del messaggio. |
| TimeStampExpired | L'ora e la data dopo le quali il messaggio non è più valido. |
| TransactionID | Il messaggio e la relativa risposta condividono lo stesso identificativo. |

*Tabella 16. Intestazione di MQSeries Integrator versione 2 — RFH2 (Continua)*

| | |
|---|---|
| UniqueID | Ogni messaggio dispone di un identificativo univoco. |
| ProcessingCategory | Riservato. |
| BodyCategory | Rappresenta il tipo di applicazione del messaggio: ad esempio, OAG o RosettaNet. |
| BodyType | Rappresenta lo scopo specifico del messaggio: ad esempio, aggiungere un ordine di vendita o sincronizzare l'inventario. |
| BodySecondaryType | Riservato. |
| AckRequested | Determina se l'applicazione di origine richiede una risposta. |
| UserArea | Dati dell'applicazione specifici dell'integrazione utente. |
| MsgHeaderVersion | Versione dell'intestazione del kernel (riservato). |
| CorrelationID | Specifico dell'integrazione utente. |
| GroupStatus | Specifico dell'integrazione utente. |
| QosPolicy | Riservato. |
| DeliveryCategory | Riservato. |
| PublicationTopic | Riservato. |
| SessionID | Riservato. |
| EncryptionStatus | Riservato. |
| Sezione dei dati | Dati di testo del messaggio. |

# Appendice D. Esempio del file di configurazione

In questa sezione viene riportata la versione del file `aqmconfig.xml` utilizzata al momento della pubblicazione del presente testo. “Esempio di un file di configurazione minimo” a pagina 123 riporta la versione del file `aqmconfig.minimum.xml` utilizzata al momento della pubblicazione di questo testo. Vedere i file `aqmconfig.xml` e `aqmconfig.minimum.xml` nella directory `samples` dell'installazione del kernel per la versione più recente dei file; gli esempi qui elencati potrebbero non essere aggiornati.

Per informazioni sulle modalità di interpretazione e di modifica del file di configurazione, consultare “File di configurazione” a pagina 65.

Questo file di configurazione di esempio include numerosi identificativi di applicazione. Sotto ciascun identificativo, viene visualizzato un gruppo di voci. Il file di configurazione di esempio contiene i seguenti identificativi di applicazione:

- 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. Tutti i diritti riservati. -->
<!--  -->
<!-- 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>
```

## Esempio di un file di configurazione minimo

Questa sezione fornisce un esempio di file di configurazione minimo da utilizzare con MQSeries Adapter Kernel. Per informazioni sul file di configurazione minimo, consultare “Aggiunta delle informazioni sull’adattatore alla configurazione” a pagina 85.

```
<?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">
               <epicmqpqueuemgr>DEFAULT</epicmqpqueuemgr>
               <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">
               <epicmqpqueuemgr>DEFAULT</epicmqpqueuemgr>
               <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>
```

# Appendice E. Esempio del file di installazione

Di seguito viene riportato un esempio del file di installazione aqmsetup, che definisce molti dei valori di configurazione iniziali del kernel, comprese numerose variabili di ambiente. Per ulteriori informazioni su questo file, consultare "File di installazione" a pagina 64. Il file aqmsetup si trova nella directory di installazione principale del kernel, samples.

```
#
# aqmsetup 1.01 01/03/27
# Sample AQM Adapter runtime parameter configuration file entries.
#
# Copyright (c) 2001 International Business Machines. Tutti i diritti riservati.
#
# 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
```

# Informazioni particolari

Queste informazioni sono state sviluppate per i prodotti e servizi offerti negli Stati Uniti. E' possibile che negli altri paesi l'IBM non offra i prodotti, le funzioni o i servizi illustrati in questo documento. Consultare il rappresentante IBM locale per informazioni sui prodotti e sui servizi disponibili nel proprio paese. Ogni riferimento relativo a prodotti, programmi o servizi IBM non implica che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti dall'IBM, possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino la violazione dei diritti di proprietà intellettuale o di altri diritti dell'IBM. E' responsabilità dell'utente valutare e verificare la possibilità di utilizzare altri programmi e/o prodotti, fatta eccezione per quelli espressamente indicati dall'IBM.

L'IBM può avere brevetti o domande di brevetto in corso relativi a quanto trattato nella presente pubblicazione. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. Chi desiderasse ricevere informazioni relative alle licenze può rivolgersi per iscritto a:

Director of Commercial Relations IBM Europe
Schoenaicher Str. 220
D-7030 Boeblingen
Deutschland

**Il seguente paragrafo non è valido per il Regno Unito o per tutti i paesi le cui leggi nazionali siano in contrasto con le disposizioni in esso contenute:** L'INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE QUESTA PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA", SENZA ALCUNA GARANZIA, ESPLICITA O IMPLICITA, IVI INCLUSE EVENTUALI GARANZIE DI COMMERCIABILITA' ED IDONEITA' AD UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe non essere a voi applicabile.

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche verranno incorporate nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare miglioramenti e/o modifiche al prodotto o al programma descritto in questa pubblicazione in qualsiasi momento e senza preavviso.

Tutti i riferimenti a siti Web non dell'IBM contenuti in questo documento sono forniti unicamente a scopo di consultazione. I materiali contenuti in tali siti Web non fanno parte di questo prodotto e l'utente si assume ogni rischio relativo al loro utilizzo.

L'IBM può utilizzare o divulgare le informazioni ricevute dagli utenti secondo le modalità ritenute appropriate, senza alcun obbligo nei loro confronti.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

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

Queste informazioni possono essere rese disponibili secondo condizioni contrattuali appropriate, compreso, in alcuni casi, il pagamento di un un corrispettivo.

Il programma su licenza descritto in queste informazioni e tutto il materiale su licenza ad esso relativo sono forniti dall'IBM nel rispetto delle condizioni previste dalla licenza d'uso.

Ogni dato qui contenuto è stato determinato in un ambiente controllato. Per questo motivo, i risultati ottenuti in altri ambienti possono essere molto diversi. Alcune misurazioni possono essere state effettuate su sistemi in via di sviluppo e non c'è alcuna garanzia che tali misurazioni corrispondano a quelle effettuate su sistemi disponibili. Inoltre, alcune misurazioni possono essere estrapolate. I risultati effettivi possono variare. Gli utenti di questo documento devono verificare i dati per il relativo ambiente.

Le informazioni relative a prodotti non IBM sono state ottenute dai fornitori di tali prodotti. L'IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti.

## Marchi

I seguenti termini sono marchi della International Business Machines Corporation negli Stati Uniti e/o altri paesi.

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

Lotus e LotusScript sono marchi della Lotus Development Corporation negli Stati Uniti e in altri paesi.

Java e tutti i marchi Java e logo sono marchi della Sun Microsystems, Inc. negli Stati Uniti e/o altri paesi.

Windows, Windows NT e il logo Windows sono marchi registrati della Microsoft Corporation negli Stati Uniti e/o in altri paesi.

Nomi di altri servizi, prodotti o società possono essere marchi di altre società.

# Glossario

Il glossario descrive i termini *chiave* utilizzati nella documentazione relativa a MQSeries Adapter Kernel.

Se un particolare termine o concetto è presente solo in una sezione, potrebbe non essere incluso nel glossario. Potrebbe comunque essere contenuto in "Indice analitico" a pagina 137.

Questo glossario non include termini relativi ad altri prodotti IBM come MQSeries.

**adattatore.** L'output di MQSeries Adapter Builder. Di solito, l'utente crea adattatori specifici per *un tipo di messaggio* che un'applicazione ha inviato o ricevuto. Pertanto, gli adattatori non fanno parte di MQSeries Adapter Offering. Un adattatore è costituito da un codice di origine C o Java che compila la libreria condivisa. Eseguiti insieme, gli adattatori e MQSeries Adapter Kernel, eseguono funzionalità di runtime di MQSeries Adapter Offering. In base alla struttura data dall'utente in MQSeries Adapter Builder, l'adattatore può contenere un'ampia gamma di funzioni quali flusso di controllo, flusso dati, navigazione sequenziale, branching condizionale, incluse decisione e interazione, digitazione dati, storing di contesto dei dati, trasformazione degli elementi di dati, controllo transazionale, operazioni logiche e personalizzazione del codice. E' possibile riutilizzare gli adattatori che vengono creati.

Consultare "tipo di messaggio" a pagina 135, "applicazione di origine" a pagina 132 e "applicazione di destinazione" a pagina 132.

**adattatore di destinazione.** Un adattatore che completa le seguenti attività:

- Riceve un messaggio (dal kernel e da MQSeries, oppure da un software per la messaggistica) inviato da un adattatore di origine.
- Elabora il messaggio di integrazione in base alla struttura dell'adattatore.
- Trasforma il messaggio di integrazione in un messaggio formattato specifico per l'applicazione che può essere ricevuto dall'applicazione di destinazione.
- Invio del messaggio all'applicazione di destinazione mediante un'interfaccia specifica dell'applicazione.
- Consente al worker di apprendere il momento in cui l'invio del messaggio all'applicazione di destinazione viene completato, in modo che il worker invii una conferma.

Se l'applicazione di destinazione può ricevere il messaggio di integrazione, non è richiesto un adattatore di destinazione.

Esiste un adattatore di destinazione per ogni tipo di messaggio. In genere, un'applicazione di destinazione può accettare più tipi di messaggi; pertanto, nella maggior parte dei casi, un'applicazione di destinazione è supportata da più adattatori di origine. Consultare "adattatore".

**adattatore di origine.** Un adattatore che completa le seguenti attività:

- Accetta o acquisisce dati strutturati da un'applicazione di origine utilizzando, di solito, un'interfaccia specifica dell'applicazione, sviluppata esternamente all'adattatore.
- Elabora i dati strutturati in base al modo in cui l'adattatore è stato creato.
- Trasforma i dati strutturati in un formato di messaggio di integrazione.
- Servendosi del kernel, inserisce il messaggio in una coda messaggi, per la consegna a uno o più adattatori di destinazione, quindi, all'applicazione di destinazione.

Esiste un adattatore di origine per ogni tipo di messaggio. In genere, un'applicazione di origine può inviare più tipi di messaggi; pertanto, nella maggior parte dei casi, un'applicazione di destinazione è supportata da più adattatori di origine.

Consultare "adattatore" a pagina 131.

**adattatore nativo.** Software utilizzato per l'invio e la ricezione degli oggetti contenenti i messaggi.

**adattatore nativo di MQSeries Adapter Kernel.** Sinonimo di adattatore nativo.

**adattatore servizi Java.** Un tipo di adattatore di linguaggio Java che, in un ambiente JMS Listener, fornisce le funzioni di un daemon di adattatore, di un programma di lavoro e di un adattatore di destinazione.

**applicazione di destinazione.** Un programma utilizzato per ricevere dati attraverso una rete di computer da un programma (l'applicazione di origine), che, normalmente, si trova su un altro computer.

**applicazione di origine.** Un programma utilizzato per inviare i dati attraverso una rete di computer a un programma (l'applicazione di destinazione), che, normalmente, si trova su un altro computer.

**bean di messaggio worker.** Un enterprise bean che esegue la funzione di un worker quando WebSphere Application Server è utilizzato sul lato di destinazione del kernel.

**BOD.** Business Object Document. Una rappresentazione di un processo aziendale standard che riguarda una o più aziende. Esempi sono aggiungere un ordine di acquisto, mostrare la disponibilità del prodotto e aggiungere un ordine di vendita. I BOD vengono definiti da OAG mediante XML. Consultare "OAG" a pagina 135 e "XML" a pagina 135.

BODs può essere utilizzato da MQSeries Adapter Offering per definire i corpi dei messaggi nei relativi messaggi di integrazione.

**categoria di testo.** I dati contenuti in un messaggio che rappresentano il tipo di applicazione del messaggio; ad esempio, OAG o RosettaNet. Appartiene alla serie dei valori di controllo messaggio. Consultare "valori di controllo messaggio" a pagina 135.

Inoltre, la categoria di testo agevola l'utente quando deve specificare il tipo di messaggio. Consultare "tipo di messaggio" a pagina 135.

**classe di collegamento.** Una classe Java specifica per ciascuna applicazione di destinazione e che può essere utilizzata per consegnare il messaggio all'applicazione di destinazione. La classe di collegamento viene utilizzata solo nel caso in cui l'adattatore di destinazione deve collegarsi all'applicazione di destinazione prima di consegnare il messaggio. Ogni classe di collegamento viene creata dall'utente. Il worker crea un'istanza di questa classe. Inoltre, la classe di collegamento esamina il file di configurazione per individuare i valori necessari all'adattatore di destinazione per supportare l'interfaccia specifica dell'applicazione nell'applicazione di destinazione. In genere, questi valori sono parametri di collegamento. Pertanto, i valori sono disponibili sull'adattatore di destinazione.

Insieme al kernel viene fornita anche una classe di collegamento fittizia.

**client di traccia.** Un componente del kernel che scrive messaggi di traccia.

**coda di errore.** Nella terminologia di MQSeries Adapter Offering, una coda messaggi utilizzata quando si riceve un messaggio da una coda di ricezione che non è possibile elaborare.

**coda di ricezione.** Nella terminologia di MQSeries Adapter Offering, una coda messaggi utilizzata come coda di immissione principale per ricevere messaggi. E' possibile avere diverse code di ricezione per un'applicazione di destinazione, ma solo una coda di ricezione per ciascuna combinazione di identificativo di applicazione, categoria corpo e tipo corpo.

**coda di risposta.** Una coda messaggi utilizzata per ricevere risposte. Viene utilizzata con il metodo del kernel `sendRequestResponse`.

**daemon dell'adattatore.** Software eseguibile che fa parte del kernel. Il daemon dell'adattatore viene utilizzato solo nel modello di consegna push. La sua funzione è quella di creare le istanze dei worker. Una volta avviato, il daemon resta attivo. Per ogni applicazione di destinazione, possono esserci uno o più daemon dell'adattatore.

In alcuni casi, il daemon funge anche da applicazione di destinazione. Esegue tutte le funzionalità richieste, quali l'utilizzo di un'adattatore di destinazione per inviare un messaggio di posta elettronica o per scrivere un record in un file.

**DTD.** (Document Type Definition). In XML, un file (o più file utilizzati insieme) che contiene una definizione formale di tipo particolare di documento. Specifica i nomi che possono essere utilizzati per gli elementi all'interno del DTD, il punto in cui questi elementi sono consentiti e le modalità di organizzazione degli elementi. In MQSeries Adapter Offering, è possibile utilizzare i DTD per definire i testi dei messaggi. Consultare "XML" a pagina 135 e "messaggio di integrazione" a pagina 134.

**file aqmconfig.xml.** Consultare "file di configurazione".

**file aqmsetup.** Consultare "file di installazione",

**file di configurazione.** Il file `aqmconfig.xml`, che contiene gran parte dei valori di configurazione del kernel. Per dettagli, consultare "File di configurazione" a pagina 65.

**file di installazione.** Un file che contiene molte impostazioni iniziali del kernel. Il nome predefinito del file è `aqmsetup`.

**formato indipendente dall'applicazione.** Consultare "messaggio di integrazione" a pagina 134.

**identificativo di applicazione di dipendenza.** Il nome dell'applicazione servita dal worker. Il worker ricava questo valore dal file di configurazione sulla base del nome del daemon dell'adattatore.

**identificativo logico dell'applicazione.** Un identificativo rappresenta l'applicazione alla quale è associato l'adattatore (un adattatore di origine oppure di destinazione). Consultare "identificativo logico di origine" e "identificativo logico di destinazione".

**identificativo logico di destinazione.** Un valore che rappresenta l'applicazione di destinazione. E' utilizzato, insieme ai valori di controllo messaggio, dal kernel per instradare ed eseguire il marshal dei messaggi. Consultare "valori di controllo messaggio" a pagina 135.

**identificativo logico di destinazione.** Un valore che rappresenta l'applicazione di destinazione associata a un adattatore di destinazione. Consultare "identificativo logico di destinazione" e "identificativo logico dell'applicazione".

**identificativo logico di origine.** Un valore che rappresenta l'applicazione di origine. E' utilizzato, insieme ai valori di controllo messaggio, dal kernel per instradare ed eseguire il marshal dei messaggi. Consultare "valori di controllo messaggio" a pagina 135, "identificativo logico dell'applicazione" e "identificativo logico di destinazione".

**identificativo logico di risposta.** L'identificativo logico dell'applicazione a cui vengono inviate le risposte, qualora venisse richiesta una risposta. Viene impostato come valore predefinito sull'identificativo logico di origine nel messaggio.

**interfaccia specifica dell'applicazione.** Un'interfaccia sviluppata all'esterno di MQSeries Adapter Offering per:

- Consentire all'adattatore di origine di accettare un messaggio dall'applicazione di origine.
- Consentire all'applicazione di destinazione di accettare un messaggio dall'adattatore di destinazione

**JMS Listener.** Un componente fornito dal prodotto WebSphere Business Integrator che consente la stretta integrazione tra MQSeries Adapter Kernel e WebSphere Application Server Advanced Edition.

**kernel.** Sinonimo di MQSeries Adapter Kernel.

**lato di destinazione del kernel.** La parte del kernel che inizia quando si riceve il messaggio da una coda messaggi e che termina quando il messaggio viene inviato all'adattatore di destinazione.

**lato di origine del kernel.** La parte del kernel che inizia quando si riceve il messaggio dall'adattatore di origine e che termina quando il messaggio viene inserito nella coda messaggi.

**messaggi delle comunicazioni.** Ciascuna informazione specifica del trasporto comunicazioni più l'oggetto contenente il messaggio, convertiti in uno specifico formato di messaggio per il trasporto comunicazioni utilizzato.

**messaggi di traccia.** Messaggi che indicano lo stato dell'elaborazione di un messaggio a un determinato stadio all'interno del kernel. E' possibile utilizzare i messaggi di traccia per facilitare la diagnosi dei problemi con il kernel o con gli adattatori.

Consultare "traccia" a pagina 135,

**messaggio.** In MQSeries, compreso MQSeries Adapter Offering, un gruppo di dati inviato da un programma e destinati a un altro programma.

**messaggio di integrazione.** Un messaggio di integrazione costituito da dati di applicazione in un formato indipendente dall'applicazione. Un esempio è rappresentato da un documento XML che l'adattatore di origine trasforma dal formato dell'applicazione di origine in XML.

**modalità di comunicazione.** La modalità utilizzata dal kernel per trasportare il messaggio e per eseguire i servizi del broker.

**modelli di consegna.** Esistono due modelli che consentono al kernel di interfacciarsi con l'applicazione di destinazione. Questi due modelli sono:

**modello push**
Il kernel è responsabile dell'inizializzazione e della gestione della consegna del messaggio all'applicazione di destinazione. Di solito, questo modello non richiede modifiche all'applicazione di destinazione per il supporto di MQSeries Adapter Offering.

**modello pull**
L'applicazione di destinazione è responsabile della gestione della consegna del messaggio. Per utilizzare questo modello, è necessario apportare modifiche all'applicazione di destinazione, in modo che possa supportare MQSeries Adapter Offering. L'applicazione di destinazione deve gestire l'interfaccia del kernel nell'applicazione di destinazione.

**modello di consegna pull.** Consultare "modelli di consegna",

**modello di consegna push.** Consultare "modelli di consegna",

**MQSeries Adapter Builder.** Software che consente a un utente di creare un adattatore per una qualsiasi applicazione, utilizzando una GUI (Graphical User Interface).

**MQSeries Adapter Kernel.** Un insieme di API e di programmi eseguibili, in C e Java, e di vari file di configurazione. Il kernel utilizza e supporta gli adattatori. Consultare "adattatore" a pagina 131,Oltre a supportare direttamente gli adattatori, il kernel esegue funzioni correlate, come l'instradamento dei messaggi, e per supportare i servizi dell'infrastruttura come la creazione dei messaggi, la creazione della traccia e i servizi che consentono di interfacciarsi con MQSeries o un altro software per la messaggistica.

**MQSeries Adapter Offering.** Un insieme di prodotti per l'integrazione delle applicazioni, formato da MQSeries Adapter Builder e MQSeries Adapter Kernel.

**OAG.** Open Applications Group. Un consorzio di industrie senza scopo di lucro che include alcuni dei più importanti produttori di componenti software. L'OAG definisce i BOD (Business Object Document).

**oggetto message-holder.** Un contenitore per metadati utilizzato dal kernel per racchiudere un messaggio di integrazione e atri dati di controllo.

**servizio di messaggi logici.** Un componente utilizzato dall'adattatore nativo per convertire i messaggi per il trasporto da parte del trasporto comunicazione.

**tipo di messaggio.** Un messaggio specificato mediante una combinazione univoca di categoria e tipo di testo. Consultare "categoria di testo" a pagina 132 e "tipo di testo".

**tipo di testo.** I dati contenuti in un messaggio che rappresentano lo scopo specifico del messaggio. Ad esempio: aggiungere un ordine di vendita oppure sincronizzare l'inventario. Appartiene alla serie dei valori di controllo messaggio. Consultare "valori di controllo messaggio".

Inoltre, il tipo di testo agevola l'utente quando deve specificare il tipo di messaggio. Consultare "tipo di messaggio",

**traccia.** Un insieme di processi utilizzati dal kernel per scrivere messaggi di traccia. Consultare "messaggi di traccia" a pagina 134.

**transazione.** Un insieme di operazioni che devono essere eseguite come singola unità lavoro. Se tutte le operazioni che formano una transazione terminano con esito positivo, la transazione viene eseguita, ossia, tutte le operazioni vengono completate. Se, invece, una o più operazioni che formano una transazione terminano con esito negativo, la transazione viene annullata, vale a dire, nessuna delle operazioni viene eseguita.

**valori di controllo messaggio.** Un termine che indica un insieme di valori dei messaggi (testo e intestazioni) e del file di configurazione che il kernel utilizza per controllare il marshaling e l'instradamento dei messaggi. Inoltre, questi valori vengono utilizzati dai singoli adattatori per controllare, in parte, la propria funzionalità.

**WebSphere Application Server Advanced Edition.** Un prodotto software dell'IBM che abilita l'utilizzo delle specifiche Sun Microsystems Enterprise JavaBeans (EJB). WebSphere Application Server Advanced Edition include un server EJB, in cui è possibile eseguire gli enterprise bean. Gli enterprise bean includono la logica business e i dati utilizzati e condivisi dai client EJB. Esistono due tipi di enterprise bean: bean di sessione, che includono oggetti e attività di breve durata, specifiche del client e bean entity, che includono i dati permanenti. Un tipo di bean di sessione chiamato bean di messaggio worker può essere utilizzato dal lato di destinazione di MQSeries Adapter Kernel.

**worker.** Software che fa parte del kernel. Il worker viene utilizzato solo nel modello di consegna push. Il daemon dell'adattatore crea e avvia i worker. Ciascun worker gestisce un adattatore nativo. Il worker consegna il messaggio all'adattatore di destinazione appropriato.

**XML.** Extensible Markup Language. Uno standard W3C per la rappresentazione dei dati.

# Indice analitico

# X

# Riservato ai commenti del lettore

**MQSeries® Adapter Kernel for Multiplatforms**
**Guida rapida**
**Versione 1 Rilascio 1**

**Pubblicazione N. GC13-2959-05**

Commenti relativi alla pubblicazione in oggetto potranno contribuire a migliorarla. Sono graditi commenti pertinenti alle informazioni contenute in questo manuale ed al modo in cui esse sono presentate. Si invita il lettore ad usare lo spazio sottostante citando, ove possibile, i riferimenti alla pagina ed al paragrafo.

Si prega di non utilizzare questo foglio per richiedere informazioni tecniche su sistemi, programmi o pubblicazioni e/o per richiedere informazioni di carattere generale.

Per tali esigenze si consiglia di rivolgersi al punto di vendita autorizzato o alla filiale IBM della propria zona oppure di chiamare il "Supporto Clienti" IBM al numero verde 167-017001.

I suggerimenti ed i commenti inviati potranno essere usati liberamente dall'IBM e dalla Selfin e diventeranno proprietà esclusiva delle stesse.

Commenti:

Si ringrazia per la collaborazione.

Per inviare i commenti è possibile utilizzare uno dei seguenti modi.
- Spedire questo modulo all'indirizzo indicato sul retro.
- Inviare un fax al numero: +39-081-660236
- Spedire una nota via email a: translationassurance@selfin.it

Se è gradita una risposta dalla Selfin, si prega di fornire le informazioni che seguono:

| | |
|---|---|
| Nome | Indirizzo |
| Società | |
| Numero di telefono | Indirizzo e-mail |

Indicandoci i Suoi dati, Lei avrà l'opportunità di ottenere dal responsabile del Servizio di Translation Assurance della Selfin S.p.A. le risposte ai quesiti o alle richieste di informazioni che vorrà sottoporci. I Suoi dati saranno trattati nel rispetto di quanto stabilito dalla legge 31 dicembre 1996, n.675 sulla "Tutela delle persone e di altri soggetti rispetto al trattamento di dati personali". I Suoi dati non saranno oggetto di comunicazione o di diffusione a terzi; essi saranno utilizzati "una tantum" e saranno conservati per il tempo strettamente necessario al loro utilizzo.

**Riservato ai commenti del lettore**
GC13-2959-05

Selfin S.p.A.
Translation Assurance

Via F. Giordani, 7

80122 NAPOLI

Printed in Denmark by IBM Danmark A/S

GC13-2959-05