MQSeries for Compaq OpenVMS Alpha

Systemverwaltung

Version 5 Release 1

SC12-2998-00

MQSeries for Compaq OpenVMS Alpha

Systemverwaltung

Version 5 Release 1

SC12-2998-00

Anmerkung

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in „Anhang L. Bemerkungen” auf Seite 395 gelesen werden.

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

Änderungen in der IBM Terminologie

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

Erste Ausgabe (Mai 2001)

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
MQSeries for Compaq Open VMS Alpha System Administration Guide Version 5 Release 1,
IBM Form SC34-5884-00,

herausgegeben von International Business Machines Corporation, USA

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

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

Änderungen des Textes bleiben vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
Mai 2001

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Zu diesem Handbuch

MQSeries for Compaq OpenVMS Alpha V5.1, das in diesem Buch auch als MQSeries for Compaq OpenVMS oder, sofern der Kontext es zulässt, einfach als MQSeries bezeichnet wird, ist Teil der MQSeries-Produktfamilie. Diese Produkte stellen Services für die Anwendungsprogrammierung zur Verfügung, mit deren Hilfe Anwendungsprogramme über *Nachrichtenwarteschlangen* miteinander kommunizieren können. Diese Art der Kommunikation wird als *kommerzielle Nachrichtenübertragung* bezeichnet. Die beteiligten Anwendungen können sich auf verschiedenen Knoten auf einer Vielzahl von Maschinen- und Betriebssystemtypen befinden. Sie verwenden eine gemeinsame Anwendungsprogrammierschnittstelle, die so genannte Schnittstelle für Nachrichtenwarteschlangen (MQI, Message Queue Interface), so dass Programme, die auf der einen Plattform entwickelt wurden, ohne weiteres auf eine andere Plattform übertragen werden können.

Dieses Buch beschreibt die Aspekte der Systemverwaltung von MQSeries for Compaq OpenVMS Alpha V5.1, und die Services, die es für die Unterstützung einer kommerziellen Nachrichtenübertragung in einer OpenVMS-Umgebung bereitstellt. Dazu gehört das Verwalten der Warteschlangen, die Anwendungen zum Empfangen ihrer Nachrichten verwenden, sowie das Sicherstellen des Zugriffs von Anwendungen auf die von ihnen benötigten Warteschlangen.

Zielgruppe

Dieses Buch richtet sich in erste Linie an Systemadministratoren und Systemprogrammierer, die die Konfigurations- und Verwaltungs-Tasks für MQSeries ausführen. Es ist aber auch für Anwendungsprogrammierer hilfreich, die Kenntnisse über Verwaltungs-Tasks von MQSeries benötigen.

Voraussetzungen zur Benutzung dieses Handbuchs

Für das Verständnis dieses Buches sind gute Kenntnisse über das Betriebssystem OpenVMS und seine Dienstprogramme erforderlich. Es wird nicht vorausgesetzt, dass Sie bereits mit Message-Queuing-Produkten gearbeitet haben, Kenntnisse über die grundlegenden Konzepte des Message-Queuing sollten jedoch vorhanden sein.

Benutzung des Handbuchs

Der Hauptteil dieses Buchs enthält:

- eine Einführung in MQSeries,
- eine Beschreibung der täglichen Verwaltungsaufgaben für ein System OpenVMS, wobei Themen wie die Verwaltung von lokalen und fernen MQSeries-Objekten, Sicherheit, Transaktionsunterstützung und Fehlerbestimmung behandelt werden.

Benutzung der Anhänge

Die Anhänge stellen Referenzmaterial bereit. Einige enthalten Informationen, die bei zukünftigen Versionen in andere MQSeries-Bücher übernommen werden.

Informationen zu MQSeries im Internet

MQSeries-URL

Die Internet-Adresse der Homepage der MQSeries-Produktfamilie lautet:

```
http://www.ibm.com/software/ts/mqseries/
```

Neuerungen in MQSeries for Compaq OpenVMS Alpha V5.1

In dieser Ausgabe des Handbuchs *MQSeries for Compaq OpenVMS Alpha V5.1 Systemverwaltung* werden die folgenden neuen Funktionen beschrieben:

MQSeries-WS-Manager-Cluster

MQSeries-Warteschlangenmanager können zu einem WS-Manager-Cluster verbunden werden. In einem Cluster können WS-Manager die von ihnen verwalteten Warteschlangen allen anderen WS-Managern zur Verfügung stellen. Jeder WS-Manager kann eine Nachricht an einen beliebigen WS-Manager innerhalb desselben Clusters senden, ohne dass explizite Kanaldefinitionen, Definitionen ferner Warteschlangen oder Übertragungswarteschlangen für die einzelnen Zieladressen erforderlich sind. Die wichtigsten Vorteile von MQSeries-Clustern sind:

- weniger Systemverwaltungs-Tasks
- erhöhte Verfügbarkeit
- gleichmäßige Auslastung

Weitere Informationen finden Sie unter „Cluster" auf Seite 14.

Anmerkung: MQSeries-Cluster unterscheiden sich von OpenVMS-Clustern. Der Begriff *Cluster* bezieht sich hier auf einen MQSeries-WS-Manager-Cluster. Ein OpenVMS-Cluster wird immer als *OpenVMS-Cluster* bezeichnet. Weitere Informationen zu OpenVMS-Clustern finden Sie in „Kapitel 16. MQSeries für OpenVMS und Cluster-Umgebungen" auf Seite 229.

MQSeries-Verwaltungsschnittstelle (MQSeries Administration Interface, MQAI)

MQSeries for Compaq OpenVMS unterstützt jetzt die MQSeries-Verwaltungsschnittstelle (MQSeries Administration Interface, MQAI), eine Programmierschnittstelle, die die Verwendung von PCF-Nachrichten zum Konfigurieren von MQSeries vereinfacht. Weitere Informationen zu MQAI, einschließlich einer ausführlichen Beschreibung der Befehle, finden Sie im Handbuch *MQSeries Administration Interface Programming Guide and Reference*.

Größe von Nachrichtenwarteschlangen

Eine Nachrichtenwarteschlange kann maximal 2 GB groß sein.

Gesteuerter, bestätigter Abschluss eines WS-Managers

Der Befehl **endmqm** wurde um eine neue Option erweitert, die einen gesteuerten, synchronen Abschluss eines WS-Managers ermöglicht.

Java-Unterstützung

MQSeries for Compaq OpenVMS arbeitet jetzt mit Java-Compilern.

Web-Verwaltung

Sie können die folgenden Verwaltungs-Tasks für MQSeries for Compaq OpenVMS jetzt in einem HTML-Browser (z. B. Netscape Navigator oder Microsoft Internet Explorer) auf einem Windows NT-System von einem fernen Standort ausführen:

- sich als MQSeries-Administrator anmelden
- einen WS-Manager auswählen und MQSC-Befehle dafür ausgeben
- MQSC-Scripts erstellen, editieren und löschen

Teil 1. Anleitung

Kapitel 1. Einführung in MQSeries

Dieses Kapitel gibt eine Einführung in MQSeries for Compaq OpenVMS aus Sicht eines Administrators und beschreibt die grundlegenden Konzepte von MQSeries und der Nachrichtenübertragung. Es enthält die folgenden Abschnitte:

MQSeries und Message-Queuing

MQSeries ermöglicht Anwendungsprogrammen mit Hilfe von Message-Queuing die Teilnahme an einer nachrichtengesteuerten Verarbeitung. Anwendungsprogramme können über verschiedene Plattformen miteinander kommunizieren, indem sie die entsprechenden Softwareprodukte für Message-Queuing verwenden. Beispielsweise können OpenVMS- und MVS/ESA-Anwendungen über MQSeries for Compaq OpenVMS bzw. MQSeries for OS/390 kommunizieren. Die Anwendungen sind von den technischen Spezifikationen der zu Grunde liegenden Kommunikationshardware unabhängig.

MQSeries-Produkte implementieren eine einheitliche Anwendungsprogrammierschnittstelle, die so genannte Schnittstelle für Nachrichtenwarteschlangen (MQI, Message Queue Interface) auf jeder Plattform, auf der die Anwendungen ausgeführt werden. Das ermöglicht eine einfache Übertragung von Anwendungen von einer Plattform auf eine andere.

MQI wird im Handbuch *MQSeries Application Programming Reference* ausführlich beschrieben.

Zeitunabhängige Anwendungen

Durch Message-Queuing kann der Austausch von Nachrichten zwischen den sendenden und empfangenden Programmen zeitunabhängig erfolgen. Das heißt, dass die sendenden und empfangenden Anwendungen voneinander entkoppelt werden, der Sender also die Verarbeitung fortsetzen kann, ohne darauf warten zu müssen, dass der Empfänger den Eingang der Nachricht bestätigt. Tatsächlich muss die Zielanwendung nicht einmal aktiv sein, wenn die Nachricht gesendet wird. Sie kann die Nachricht abrufen, nachdem sie gestartet wurde.

Nachrichtengesteuerte Verarbeitung

Bei der Ankunft in einer Warteschlange können Nachrichten eine Anwendung automatisch über einen Mechanismus, die so genannte *Auslösefunktion*, starten. Bei Bedarf können die Anwendungen gestoppt werden, nachdem die Nachricht(en) verarbeitet wurde(n).

Nachrichten und Warteschlangen

Nachrichten und Warteschlangen sind die Basiskomponenten eines Message-Queuing-Systems.

Nachrichten

Eine *Nachricht* ist eine Zeichenfolge von Bytes, die für die Anwendungen, die diese Zeichenfolge verwenden, eine Bedeutung hat. Nachrichten werden zur Übertragung von Informationen von einer Anwendung an eine andere (oder an verschiedene Komponenten derselben Anwendung) verwendet. Die Anwendungen können auf derselben Plattform oder auf verschiedenen Plattformen aktiv sein.

MQSeries-Nachrichten bestehen aus zwei Teilen:

- *Anwendungsdaten*

  Der Inhalt und die Struktur von Anwendungsdaten werden durch das Anwendungsprogramm, das diese Daten verwendet, definiert.
- *Nachrichtendeskriptor*

  Der Nachrichtendeskriptor identifiziert die Nachricht und enthält weitere Steuerinformationen, z. B. den Nachrichtentyp und die Priorität, die der Nachricht von der sendenden Anwendung zugewiesen wurden.

  Das Format des Nachrichtendeskriptors wird von MQSeries definiert. Eine vollständige Beschreibung des Nachrichtendeskriptors finden Sie im Handbuch *MQSeries Application Programming Reference*.

Nachrichtenlänge

In MQSeries beträgt die maximale Nachrichtenlänge 100 MB (wobei 1 MB = 1 048 576 Bytes). In der Praxis kann die Nachrichtenlänge durch folgende Einstellungen begrenzt sein:

- die maximale Nachrichtenlänge, die für die Empfangswarteschlange definiert wurde.
- die maximale Nachrichtenlänge, die für den Warteschlangenmanager definiert wurde.
- die maximale Nachrichtenlänge, die entweder von der sendenden oder von der empfangenden Anwendung definiert wurde.
- durch den für die Nachricht verfügbaren Speicherbereich.

Es können mehrere Nachrichten erforderlich sein, um alle Informationen, die von einer Anwendung angefordert werden, zu senden.

Warteschlangen

Eine *Warteschlange* ist eine Datenstruktur, die zum Speichern von Nachrichten verwendet wird. Die Nachrichten können von Anwendungsprogrammen oder einem WS-Manager (als Teil seines normalen Betriebsablaufs) in die Warteschlange eingereiht werden.

Jede Warteschlange ist einem *WS-Manager* zugeordnet. Der WS-Manager ist für die Verwaltung der ihm zugeordneten Warteschlangen und für die Speicherung aller von ihm empfangenen Nachrichten in den entsprechenden Warteschlangen zuständig.

Die maximale Größe einer Warteschlange beträgt 2 GB. Weitere Informationen zur Planung des für Warteschlangen erforderlichen Speicherbereichs finden Sie im Handbuch *MQSeries Planning Guide*, oder besuchen Sie die folgende Website für plattformspezifische Leistungsberichte:

http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html

Senden und Empfangen von Nachrichten durch Anwendungen

Anwendungen senden und empfangen Nachrichten über *MQI-Aufrufe*. Um zum Beispiel eine Nachricht in eine Warteschlange einzureihen, geht eine Anwendung folgendermaßen vor:

1. Sie öffnet die betreffende Warteschlange mit dem MQI-Aufruf MQOPEN.
2. Sie reiht die Nachricht mit dem MQI-Aufruf MQPUT in die Warteschlange ein.
3. Eine andere Anwendung kann die Nachricht mit dem MQI-Aufruf MQGET aus der betreffenden Warteschlange abrufen.

Weitere Informationen zu MQI-Aufrufen finden Sie im Handbuch *MQSeries Application Programming Reference*.

Vordefinierte und dynamische Warteschlangen

Warteschlangen können nach der Art und Weise ihrer Erstellung unterschieden werden:

- *Vordefinierte Warteschlangen* werden von einem Administrator mit Hilfe des entsprechenden Befehlssatzes erstellt. Beispiel: Mit dem MQSC-Befehl DEFINE QLOCAL wird eine vordefinierte lokale Warteschlange erstellt. Vordefinierte Warteschlangen sind permanent, d. h., sie existieren unabhängig von den Anwendungen, von denen sie verwendet werden, und bleiben auch bei einem Neustart von MQSeries erhalten.
- *Dynamische Warteschlangen* werden erstellt, wenn eine Anwendung eine OPEN-Anforderung absetzt und dabei den Namen einer *Modellwarteschlange* angibt. Die erstellte Warteschlange basiert auf einer Vorlage mit einer Warteschlangendefinition, der so genannten Modellwarteschlange. Eine Modellwarteschlange können Sie mit dem MQSC-Befehl DEFINE QMODEL erstellen. Jede dynamische Warteschlange, die aus einer Modellwarteschlange erstellt wird, übernimmt die Attribute der betreffenden Modellwarteschlange, z. B. die maximale Anzahl der Nachrichten, die gespeichert werden können.

  Modellwarteschlangen verfügen über ein Attribut, das angibt, ob die dynamische Warteschlange permanent oder temporär ist. Permanente Warteschlangen bleiben beim Neustart einer Anwendung oder eines WS-Managers erhalten, temporäre Warteschlangen dagegen gehen bei einem Neustart verloren.

Nachrichten aus Warteschlangen abrufen

In MQSeries können Anwendungen, die über eine entsprechende Berechtigung verfügen, Nachrichten nach den folgenden Abrufalgorithmen aus einer Warteschlange abrufen:

- First In/First Out (FIFO)
- Nachrichtenpriorität, so wie im Nachrichtendeskriptor definiert. Nachrichten mit derselben Priorität werden nach der FIFO-Methode abgerufen.
- ein Programmaufruf für eine bestimmte Nachricht

Die jeweils verwendete Methode wird durch den MQGET-Aufruf aus der Anwendung bestimmt.

Objekte

Viele der in diesem Buch beschriebenen Tasks beinhalten eine Bearbeitung von MQSeries-*Objekten*. In MQSeries Version 5.1 gibt es folgende Objektarten: WS-Manager, Warteschlangen, Prozessdefinitionen, Kanäle, Cluster und Namenslisten.

Die Bearbeitung oder *Verwaltung* von Objekten umfasst Folgendes:
- Starten und Stoppen von WS-Managern
- Erstellen von Objekten, insbesondere Warteschlangen, für Anwendungen
- Arbeiten mit Kanälen zum Erstellen von Kommunikationspfaden zu WS-Managern auf anderen (fernen) Systemen. Dies wird im Handbuch *MQSeries Intercommunication* ausführlich beschrieben.
- Erstellen von *Clustern* aus WS-Managern, um den Gesamtverwaltungsprozess zu vereinfachen oder um eine gleichmäßige Auslastung zu erreichen.

Die folgenden Kapitel dieses Buches enthalten detaillierte Informationen zur Verwaltung:
- „Kapitel 2. Eine Einführung in die MQSeries-Verwaltung" auf Seite 19
- „Kapitel 3. WS-Manager mit Hilfe von Steuerbefehlen verwalten" auf Seite 25
- „Kapitel 4. Lokale MQSeries-Objekte verwalten" auf Seite 35
- „Kapitel 6. Ferne MQSeries-Objekte verwalten" auf Seite 65

Objektnamen

Die für MQSeries-Objekte gültige Namenskonvention ist vom jeweiligen Objekt abhängig. Jedes Exemplar eines WS-Managers wird durch seinen Namen identifiziert. Dieser Name muss innerhalb eines Netzes verbundener WS-Manager eindeutig sein, damit jeder WS-Manager den Ziel-WS-Manager, an den eine bestimmte Nachricht gesendet werden soll, unzweifelhaft identifizieren kann. Für die anderen Objektarten gilt, dass jedem Objekt ein Name zugeordnet wird, über den es angesprochen werden kann. Diese Namen müssen innerhalb eines WS-Managers und einer Objektart eindeutig sein. Es kann zum Beispiel eine Warteschlange und einen Prozess mit demselben Namen geben, es kann jedoch nicht zwei Warteschlangen mit demselben Namen geben.

In MQSeries können Namen maximal 48 Zeichen lang sein; eine Ausnahme bilden *Kanäle*, deren Namen maximal 20 Zeichen lang sein dürfen. Weitere Informationen zu Namen finden Sie unter „Regeln für die Benennung von MQSeries-Objekten" auf Seite 253.

Objekte verwalten

Sie können Objekte folgendermaßen erstellen, ändern, anzeigen oder löschen:
- über Steuerbefehle, die über eine Tastatur eingegeben werden.
- über MQSC-Befehle (MQSeries Commands), die über eine Tastatur eingegeben oder aus einer Datei gelesen werden können.
- über PCF-Befehle (Programmable Command Format), die in einem automatisierten Programm verwendet werden können.
- über MQAI-Aufrufe (MQSeries Administration Interface, MQSeries-Verwaltungsschnittstelle) in einem Programm.

Weitere Informationen finden Sie in „Kapitel 2. Eine Einführung in die MQSeries-Verwaltung" auf Seite 19.

Objektattribute

Die Eigenschaften eines Objekts werden durch seine Attribute definiert. Einige Attribute können Sie angeben, andere nur anzeigen. Die maximale Nachrichtenlänge für eine Warteschlange wird zum Beispiel durch das Attribut *MaxMsgLength* definiert. Dieses Attribut können Sie beim Erstellen einer Warteschlange angeben. Das Attribut *DefinitionType* gibt an, wie die Warteschlange erstellt wurde. Dieses Attribut können Sie nur anzeigen.

In MQSeries kann ein Attribut auf zwei Arten angesprochen werden:

- über seinen PCF-Namen, z. B. *MaxMsgLength*.
- über seinen MQSC-Namen, z. B. MAXMSGL.

Der formale Name eines Attributs ist sein PCF-Name. Da die MQSC-Funktion in diesem Buch eine wichtige Rolle spielt, wird in Beispielen in der Regel der MQSC-Name eines Attributs verwendet, und nicht sein PCF-Name.

MQSeries-Warteschlangenmanager

Ein Warteschlangenmanager (WS-Manager) stellt Queuing-Services für Anwendungen bereit und verwaltet die ihm zugeordneten Warteschlangen. Er stellt Folgendes sicher:

- Objektattribute werden den empfangenen Befehlen entsprechend geändert.
- Besondere Ereignisse wie Auslöseereignisse oder Instrumentierungsereignisse werden generiert, sobald die entsprechenden Bedingungen eintreten.
- Nachrichten werden so in die richtige Warteschlange eingereiht, wie von der Anwendung durch den MQPUT-Aufruf gefordert. Sollte dies nicht möglich sein, wird die Anwendung informiert und ein entsprechender Ursachencode übergeben.

Jede Warteschlange gehört zu einem bestimmten WS-Manager und wird als *lokale Warteschlange* des betreffenden WS-Managers bezeichnet. Der WS-Manager, mit dem eine Anwendung verbunden ist, wird als der lokale WS-Manager der betreffenden Anwendung bezeichnet. Für die Anwendung sind die Warteschlangen ihres lokalen WS-Managers lokale Warteschlangen.

Eine *ferne Warteschlange* ist nichts anderes als eine Warteschlange, die einem anderen WS-Manager zugeordnet ist.

Ein *ferner WS-Manager* ist jeder WS-Manager, der nicht der lokale WS-Manager ist. Ein ferner WS-Manager kann sich auf einem fernen System im Netz oder auf demselben System wie der lokale WS-Manager befinden.

MQSeries unterstützt mehrere WS-Manager auf demselben System.

MQI-Aufrufe

Ein WS-Manager-Objekt kann in einigen MQI-Aufrufen verwendet werden. Sie können zum Beispiel mit dem MQI-Aufruf MQINFO die Attribute des WS-Manager-Objekts abfragen.

Anmerkung: In ein WS-Manager-Objekt können Sie keine Nachrichten einreihen. Nachrichten werden immer in Warteschlangenobjekte eingereiht, und nicht in WS-Manager-Objekte.

MQSeries-Warteschlangen

Warteschlangen werden in MQSeries folgendermaßen definiert:
- mit dem MQSC-Befehl DEFINE
- mit dem PCF-Befehl 'Create Queue'.

In den Befehlen werden der Typ der Warteschlange und ihre Attribute angegeben. Zum Beispiel verfügt ein lokales Warteschlangenobjekt über Attribute, die angeben, welche Aktion ausgeführt wird, wenn Anwendungen MQI-Aufrufe für diese Warteschlange ausgeben. Über Attribute wird zum Beispiel Folgendes angegeben:

- Anwendungen können Nachrichten aus der Warteschlange abrufen (GET ist aktiviert).
- Anwendungen können Nachrichten in die Warteschlange einreihen (PUT ist aktiviert).
- Eine Anwendung greift exklusiv auf die Warteschlange zu, oder mehrere Anwendungen können gemeinsam darauf zugreifen.
- die maximale Anzahl an Nachrichten, die gleichzeitig in der Warteschlange gespeichert werden können (maximale Warteschlangenlänge).
- die maximale Länge von Nachrichten, die in die Warteschlange eingereiht werden können.

Weitere Informationen zum Definieren von Warteschlangenobjekten finden Sie in den Handbüchern *MQSeries MQSC-Befehle* und *MQSeries Programmable System Management*.

Warteschlangenobjekte verwenden

In MQSeries gibt es verschiedene Arten von Warteschlangenobjekten. Jede Objektart kann mit Hilfe der Produktbefehle bearbeitet werden, und jeder werden auf unterschiedliche Art und Weise tatsächliche Warteschlangen zugeordnet:

- **Lokales Warteschlangenobjekt**

  Ein lokales Warteschlangenobjekt identifiziert eine lokale Warteschlange, die dem WS-Manager zugeordnet ist, mit dem die Anwendung verbunden ist. Alle Warteschlangen sind lokale Warteschlangen in dem Sinne, dass jede Warteschlange einem WS-Manager zugeordnet ist, und die Warteschlange für diesen WS-Manager eine lokale Warteschlange darstellt.

- **Fernes Warteschlangenobjekt**

  Ein fernes Warteschlangenobjekt identifiziert eine Warteschlange, die einem anderen WS-Manager zugeordnet ist. Diese Warteschlange muss als lokale Warteschlange des betreffenden WS-Managers definiert werden. Anhand der Informationen, die Sie beim Definieren eines fernen Warteschlangenobjekts angeben, kann der lokale WS-Manager den fernen WS-Manager ausfindig machen, so dass alle Nachrichten für die ferne Warteschlange an den richtigen WS-Manager gesendet werden.

  Bevor Anwendungen Nachrichten an eine Warteschlange eines fernen WS-Managers senden können, müssen Sie eine Übertragungswarteschlange und Kanäle zwischen den WS-Managern definieren, es sei denn, Sie haben mehrere WS-Manager in einem Cluster zusammengefasst. Weitere Informationen zu Clustern finden Sie unter „Fernverwaltung mit Hilfe von Clustern" auf Seite 66.

- **Aliaswarteschlangenobjekt**

  Mit Hilfe eines Aliaswarteschlangenobjekts können Anwendungen auf eine Warteschlange zugreifen, indem sie in MQI-Aufrufen indirekt darauf verweisen. Wenn ein Aliaswarteschlangenname in einem MQI-Aufruf verwendet wird, wird der Name entweder in den Namen einer lokalen oder den einer fernen Warteschlange aufgelöst. Dies gibt Ihnen die Möglichkeit, die von einer Anwendung verwendeten Warteschlangen zu ändern, ohne die Anwendung selbst ändern zu müssen. Sie ändern einfach die Definition der Aliaswarteschlange und geben dort den Namen der neuen Warteschlange ein, in den der Aliasname aufgelöst werden soll.

  Eine Aliaswarteschlange ist keine Warteschlange, sondern ein Objekt, über das Sie auf eine andere Warteschlange zugreifen können.

- **Modellwarteschlangenobjekt**

  Ein Modellwarteschlangenobjekt definiert eine Gruppe von Warteschlangenattributen, die als Vorlage zum Erstellen einer dynamischen Warteschlange verwendet wird. Dynamische Warteschlangen werden vom WS-Manager erstellt, wenn eine Anwendung einen MQOPEN-Aufruf absetzt und es sich bei dem darin angegebenen Warteschlangennamen um den Namen einer Modellwarteschlange handelt. Die auf diese Weise erstellte dynamische Warteschlange ist eine lokale Warteschlange, deren Attribute aus der Modellwarteschlangendefinition übernommen werden. Der Name der dynamischen Warteschlange kann von der Anwendung angegeben oder vom WS-Manager generiert und an die Anwendung zurückgegeben werden.

  Auf diese Weise definierte dynamische Warteschlangen können temporäre Warteschlangen sein, die beim Neustart des Produkts verloren gehen, oder permanente Warteschlangen, die bei einem Neustart erhalten bleiben.

Spezielle lokale Warteschlangen in MQSeries

MQSeries verwendet einige lokale Warteschlangen für ganz bestimmte Operationen. Diese Warteschlangen ***müssen*** Sie definieren, bevor MQSeries sie verwenden kann.

Anwendungswarteschlangen: Eine Warteschlange, die von einer Anwendung (über MQI-Aufrufe) verwendet wird, wird als *Anwendungswarteschlange* bezeichnet. Dabei kann es sich um eine lokale Warteschlange des WS-Managers, mit dem eine Anwendung verbunden ist, oder um eine ferne Warteschlange, die einem anderen WS-Manager zugeordnet ist, handeln.

Anwendungen können Nachrichten in lokale und ferne Warteschlangen einreihen. Sie können Nachrichten jedoch nur aus einer lokalen Warteschlange abrufen.

Initialisierungswarteschlangen: *Initialisierungswarteschlangen* sind Warteschlangen, die von der Auslösefunktion verwendet werden. Ein WS-Manager reiht eine Auslösenachricht in eine Initialisierungswarteschlange ein, sobald ein Auslöseereignis eintritt. Ein Auslöseereignis ist eine logische Kombination aus Bedingungen, die von einem WS-Manager erkannt wird. Ein Auslöseereignis kann zum Beispiel generiert werden, wenn die Anzahl der Nachrichten in einer Warteschlange eine vordefinierte Größe erreicht. Dieses Ereignis veranlasst den WS-Manager, eine Auslösenachricht in eine bestimmte Initialisierungswarteschlange einzureihen. Diese Auslösenachricht wird von einem *Auslösemonitor*, einer speziellen Anwendung zum Überwachen einer Initialisierungswarteschlange, abgerufen. Der Auslösemonitor startet dann das Anwendungsprogramm, das in der Auslösenachricht angegeben ist.

Wenn ein WS-Manager die Auslösefunktion verwenden soll, muss mindestens eine Initialisierungswarteschlange für den betreffenden WS-Manager definiert werden.

Weitere Informationen finden Sie unter „Objekte für Auslösefunktion verwalten" auf Seite 57 und „runmqtrm (Auslösemonitor starten)" auf Seite 308. Weitere Informationen zur Auslösefunktion finden Sie im Handbuch *MQSeries Application Programming Guide*.

Übertragungswarteschlangen: In einer *Übertragungswarteschlange* werden temporär Nachrichten gespeichert, die an einen fernen WS-Manager gerichtet sind. Sie müssen mindestens eine Übertragungswarteschlange für jeden fernen WS-Manager definieren, an den der lokale WS-Manager Nachrichten direkt senden soll. Diese Warteschlangen werden auch bei der Fernverwaltung verwendet (siehe unter „Fernverwaltung von einem lokalen WS-Manager aus mit Hilfe von MQSC-Befehlen" auf Seite 67). Weitere Informationen zur Verwendung von Übertragungswarteschlangen bei einem verteilten Message-Queuing finden Sie im Handbuch *MQSeries Intercommunication*.

Cluster-Übertragungswarteschlangen: Jeder WS-Manager in einem Cluster verfügt über eine Cluster-Übertragungswarteschlange mit dem Namen SYSTEM-.CLUSTER.TRANSMIT.QUEUE. Eine Definition dieser Warteschlange wird standardmäßig für jeden WS-Manager der Version 5.1 erstellt.

Ein WS-Manager in einem Cluster kann über die Cluster-Übertragungswarteschlange Nachrichten an alle anderen WS-Manager in demselben Cluster senden.

WS-Manager in einem Cluster können darüber hinaus mit WS-Managern kommunizieren, die dem Cluster nicht angehören. Dazu müssen Sie, wie in einer herkömmlichen Umgebung mit verteiltem Message-Queuing, Kanäle und eine Übertragungswarteschlange zwischen einem WS-Manager innerhalb des Clusters und einem anderen WS-Manager außerhalb des Clusters definieren.

Bei der Namensauflösung hat die Cluster-Übertragungswarteschlange Vorrang vor der Standardübertragungswarteschlange. Wenn ein WS-Manager, der dem Cluster nicht angehört, eine Nachricht in eine ferne Warteschlange einreiht, wird immer die Standardübertragungswarteschlange verwendet, falls keine Übertragungswarteschlange mit demselben Namen wie dem des Ziel-WS-Managers vorhanden ist.

Wenn ein WS-Manager einem Cluster angehört, wird standardmäßig die Warteschlange SYSTEM.CLUSTER.TRANSMIT.QUEUE verwendet, es sei denn, die Zielwarteschlange gehört nicht demselben Cluster an.

Warteschlangen für nicht zustellbare Nachrichten: In einer *Warteschlange für nicht zustellbare Nachrichten* werden Nachrichten gespeichert, die nicht an ihr vorgesehenes Ziel weitergeleitet werden können. Dies ist zum Beispiel dann der Fall, wenn die Zielwarteschlange voll ist. Der Namen der standardmäßigen Warteschlange für nicht zustellbare Nachrichten lautet SYSTEM.DEAD.LETTER.QUEUE. Diese Warteschlangen werden auf anderen Plattformen auch als Warteschlangen für nicht übermittelte Nachrichten bezeichnet.

Für verteiltes Message-Queuing sollten Sie eine Warteschlange für nicht zustellbare Nachrichten für jeden beteiligten WS-Manager definieren.

Befehlswarteschlangen: Die Befehlswarteschlange SYSTEM.ADMIN.COMMAND-.QUEUE ist eine lokale Warteschlange, an die Anwendungen, die über eine entsprechende Berechtigung verfügen, MQSeries-Befehle zur Verarbeitung senden können. Diese Befehle werden dann von einer MQSeries-Komponente, dem so genannten Befehlsserver, abgerufen. Der Befehlsserver prüft die Befehle, übergibt die gültigen Befehle zur Verarbeitung an den WS-Manager und gibt Antworten an die jeweilige Warteschlange für zu beantwortende Nachrichten zurück.

Eine Befehlswarteschlange wird für jeden WS-Manager erstellt, und zwar automatisch bei der Erstellung des WS-Managers.

Warteschlangen für zu beantwortende Nachrichten: Wenn eine Anwendung eine Anforderungsnachricht sendet, kann die Anwendung, die diese Nachricht empfängt, eine Antwortnachricht an die sendende Anwendung zurückschicken. Diese Nachricht wird in eine Warteschlange, die so genannte Warteschlange für zu beantwortende Nachrichten, eingereiht. Dabei handelt es sich normalerweise um eine lokale Warteschlange der sendenden Anwendung. Der Name der Warteschlange für zu beantwortende Nachrichten wird von der sendenden Anwendung als Teil des Nachrichtendeskriptors angegeben.

Ereigniswarteschlangen: MQSeries Version 5.1 unterstützt Instrumentierungsereignisse, die zum Überwachen von WS-Managern unabhängig von MQI-Anwendungen verwendet werden können. Instrumentierungsereignisse können auf verschiedene Weise generiert werden, z. B. wie folgt:

- Eine Anwendung versucht eine Nachricht in eine Warteschlange einzureihen, die nicht verfügbar ist oder nicht existiert.
- Eine Warteschlange ist voll.
- Ein Kanal wird gestartet.

Wenn ein Instrumentierungsereignis eintritt, reiht der WS-Manager eine Ereignisnachricht in eine Ereigniswarteschlange ein. Diese Nachricht kann dann von einer Überwachungsanwendung gelesen werden, die einen Administrator informieren oder selbst Aktionen zur Fehlerbehebung einleiten kann, wenn das Ereignis auf einen Fehler hinweist.

Anmerkung: Auslöseereignisse unterscheiden sich insofern deutlich von Instrumentierungsereignissen, als sie nicht durch dieselben Bedingungen verursacht werden und keine Ereignisnachrichten generieren.

Weitere Informationen zu Instrumentierungsereignissen finden Sie im Handbuch *MQSeries Programmable System Management*.

Prozessdefinitionen

Ein *Prozessdefinitionsobjekt* definiert eine Anwendung, die als Antwort auf ein Auslöseereignis auf einem WS-Manager von MQSeries gestartet wird. Weitere Informationen finden Sie unter „Initialisierungswarteschlangen" auf Seite 11.

Die Prozessdefinitionsattribute umfassen die Anwendungs-ID, den Anwendungstyp und anwendungsspezifische Daten.

Verwenden Sie zum Erstellen einer Prozessdefinition den MQSC-Befehl DEFINE PROCESS oder den PCF-Befehl 'Create Process'.

Kanäle

Kanäle sind Objekte, die einen Kommunikationspfad zwischen zwei WS-Managern bereitstellen. Kanäle werden beim verteilten Message-Queuing zum Verschieben von Nachrichten von einem WS-Manager zu einem anderen verwendet. Durch sie werden Anwendungen von den zu Grunde liegenden Übertragungsprotokollen unabhängig. Die WS-Manager können sich auf derselben oder auf verschiedenen Plattformen befinden. Damit WS-Manager miteinander kommunizieren können, müssen Sie ein Kanalobjekt für den WS-Manager, der Nachrichten senden soll, und ein anderes (komplementäres) Kanalobjekt für den WS-Manager, der die Nachrichten empfangen soll, definieren.

Weitere Informationen zu Kanälen und zu deren Verwendung finden Sie im Handbuch *MQSeries Intercommunication* sowie unter „Kanäle und Übertragungswarteschlangen für Fernverwaltung vorbereiten" auf Seite 68.

Cluster

In einem herkömmlichen MQSeries-Netz arbeitet bei einem verteilten Message-Queuing jeder WS-Manager unabhängig. Damit ein WS-Manager Nachrichten an einen anderen WS-Manager senden kann, muss für ihn eine Übertragungswarteschlange, ein Kanal zum fernen WS-Manager und eine Definition einer fernen Warteschlange erstellt werden, und das für jede Warteschlange, an die der WS-Manager Nachrichten senden soll.

Ein Cluster ist eine Gruppe von WS-Managern, die so konfiguriert ist, dass die WS-Manager über ein einziges Netz direkt miteinander kommunizieren können, ohne dass komplexe Übertragungswarteschlangen, Kanäle und Warteschlangendefinitionen benötigt werden.

Anmerkung: MQSeries-Cluster unterscheiden sich von OpenVMS-Clustern. Der Begriff *Cluster* bezieht sich hier auf einen MQSeries-WS-Manager-Cluster. Ein OpenVMS-Cluster wird immer als *OpenVMS-Cluster* bezeichnet. Weitere Informationen zu OpenVMS-Clustern finden Sie in „Kapitel 16. MQSeries für OpenVMS und Cluster-Umgebungen" auf Seite 229.

Weitere Informationen zu Clustern finden Sie in „Kapitel 6. Ferne MQSeries-Objekte verwalten" auf Seite 65 und im Handbuch *Cluster-Unterstützung in MQSeries*.

Namenslisten

Eine Namensliste ist ein MQSeries-Objekt, das eine Liste anderer MQSeries-Objekte enthält. Namenslisten werden üblicherweise von Anwendungen wie Auslösemonitoren verwendet, um eine Warteschlangengruppe zu identifizieren. Der Vorteil einer Namensliste besteht darin, dass sie unabhängig von Anwendungen verwaltet wird; d. h., sie kann aktualisiert werden, ohne dass Anwendungen, die darauf zugreifen, gestoppt werden müssen. Außerdem ist die Namensliste nicht betroffen, wenn in einer Anwendung ein Fehler auftritt, so dass andere Anwendungen sie weiter verwenden können. Namenslisten werden auch mit WS-Manager-Clustern verwendet, so dass Sie eine Liste von Clustern verwalten können, auf die von mehreren MQSeries-Objekten verwiesen wird.

Systemstandardobjekte

Die *Systemstandardobjekte* bilden eine Gruppe von Objektdefinitionen, die automatisch zusammen mit einem WS-Manager erstellt werden. Sie können jede dieser Objektdefinitionen kopieren und ändern, um sie in Anwendungen in Ihrer Installationsumgebung zu verwenden. Standardobjektnamen beginnen immer mit SYSTEM.DEF; die standardmäßige lokale Warteschlange heißt zum Beispiel SYSTEM.DEFAULT.LOCAL.QUEUE und die standardmäßige Empfängerwarteschlange SYSTEM.DEF.RECEIVER. Sie können diese Objekte nicht umbenennen, da Standardobjekte mit diesen Namen immer vorhanden sein müssen. Wenn Sie ein Objekt definieren, werden alle Attribute, die Sie nicht explizit angeben, aus dem zugehörigen Standardobjekt kopiert. Wenn Sie zum Beispiel eine lokale Warteschlange definieren, werden die Attribute, die Sie nicht angeben, von der Standardwarteschlange SYSTEM.DEFAULT.LOCAL.QUEUE übernommen.

Weitere Informationen zu Systemstandardwerten finden Sie in „Anhang B. Systemstandardwerte” auf Seite 327.

Lokale und ferne Verwaltung

Lokale Verwaltung bezeichnet die Ausführung von Verwaltungs-Tasks für WS-Manager, die Sie auf Ihrem lokalen System definiert haben. Sie können aber auch auf andere Systeme zugreifen, z. B. über das TCP/IP-Terminalemulationsprogramm **telnet**, und dort Verwaltungs-Tasks ausführen. In MQSeries kann dies ebenfalls als lokale Verwaltung angesehen werden, weil keine Kanäle benötigt werden, d. h., die Kommunikation wird vom Betriebssystem gesteuert.

MQSeries unterstützt die Verwaltung von einem Einzelpunkt aus. Dies wird als *Fernverwaltung* bezeichnet. Dadurch ist es Ihnen möglich, Befehle an Ihrem lokalen System auszugeben, die auf einem anderen System verarbeitet werden. Dazu müssen Sie sich nicht an dem anderen System anmelden, Sie müssen jedoch vorher die erforderlichen Kanäle definieren. Der WS-Manager und der Befehlsserver auf dem Zielsystem müssen aktiv sein. Sie können zum Beispiel einen Fernbefehl zum Ändern einer Warteschlangendefinition auf einem fernen WS-Manager ausgeben. Nicht alle Befehle können auf diese Weise ausgegeben werden. Dies gilt insbesondere für Befehle zum Erstellen und Starten von WS-Managern und zum Starten von Befehlsservern. Um diese Art von Tasks auszuführen, müssen Sie sich entweder an dem fernen System anmelden und die Befehle dort ausgeben, oder Sie müssen einen Prozess erstellen, der die Befehle an Ihrer Stelle ausgeben kann.

Clients und Server

MQSeries unterstützt Client/Server-Konfigurationen für MQSeries-Anwendungen.

Ein *MQSeries-Client* ist ein Teil des MQSeries-Produkts, das auf einem System installiert wird, damit es MQI-Aufrufe von Anwendungen annehmen und sie an ein *MQI-Server*-System übergeben kann. Dort werden sie dann von einem WS-Manager verarbeitet. In der Regel befinden sich Client und Server auf verschiedenen Systemen, sie können sich jedoch auch auf demselben System befinden.

Ein *MQI-Server* ist ein WS-Manager, der Queuing-Services für einen oder mehrere Clients bereitstellt. Alle MQSeries-Objekte, z. B. Warteschlangen, sind nur auf dem WS-Manager-System, d. h. auf dem MQI-Serversystem, vorhanden. Ein Server unterstützt auch normale lokale MQSeries-Anwendungen.

Der Unterschied zwischen einem MQI-Server und einem gewöhnlichen WS-Manager besteht darin, dass ein Server über eine dedizierte Kommunikationsverbindung mit jedem einzelnen Client verfügt. Weitere Informationen zum Erstellen von Kanälen für Clients und Server finden Sie im Handbuch *MQSeries Intercommunication*.

Allgemeine Informationen zur Client-Unterstützung finden Sie im Handbuch *MQSeries Clients*.

MQSeries-Anwendungen in einer Client/Server-Umgebung

Client-MQSeries-Anwendungen, die mit einem Server verbunden sind, können MQI-Aufrufe auf dieselbe Weise wie lokale Anwendungen ausgeben. Die Client-Anwendung gibt den Aufruf MQCONN aus, um eine Verbindung mit einem bestimmten WS-Manager herzustellen. Jeder weitere MQI-Aufruf, in dem die Verbindungskennung angegeben wird, die von der Verbindungsanforderung zurückgegeben wurde, wird dann von diesem WS-Manager verarbeitet.

Sie müssen Ihre Anwendungen mit den zugehörigen Client-Bibliotheken verbinden. Weitere Informationen finden Sie im Handbuch *MQSeries Clients*.

WS-Manager-Funktionen erweitern

Die von einem WS-Manager bereitgestellten Funktionen können um Folgendes erweitert werden:

- Benutzer-Exits
- installierbare Services

Benutzer-Exits

Benutzer-Exits stellen Benutzern einen Mechanismus zur Verfügung, der es ihnen ermöglicht, ihren eigenen Code in eine WS-Manager-Funktion einzufügen. Folgende Benutzer-Exits werden unterstützt:

- **Kanal-Exits**

  Diese Exits ändern die Betriebsart von Kanälen. Kanal-Exits werden im Handbuch *MQSeries Intercommunication* beschrieben.

- **Datenkonvertierungs-Exits**

  Diese Exits erstellen Quellcodefragmente, die in Anwendungsprogramme eingefügt werden können, um Daten von einem Format in ein anderes zu konvertieren. Datenkonvertierungs-Exits werden im Handbuch *MQSeries Application Programming Guide* beschrieben.

- **Exits für Cluster-Auslastung**

  Die von diesem Exit ausgeführte Funktion wird vom Hersteller des Exits definiert. Informationen zur Aufrufdefinition finden Sie im Handbuch *Cluster-Unterstützung in MQSeries*.

Alle Exit-Arten beziehen sich auf verteiltes Message-Queuing. Weitere Informationen zu diesen Exits und zu ihrer Verwendung finden Sie im Handbuch *MQSeries Intercommunication*.

Installierbare Services

Installierbare Services bieten eine größere Funktionalität als Exits, da sie über formalisierte Schnittstellen (eine API) mit mehreren Eingangspunkten verfügen.

Eine Implementierung eines installierbaren Services wird als *Servicekomponente* bezeichnet. Sie können die im Produkt integrierten Komponenten verwenden oder eigene Komponenten erstellen, von denen die von Ihnen benötigten Funktionen ausgeführt werden.

Zurzeit werden die folgenden installierbaren Services bereitgestellt:

- **Berechtigungsservice**

  Mit Hilfe des Berechtigungsservices können Sie eigene Sicherheitsfunktionen erstellen.

  Dieser Service wird vom Objektberechtigungsmanager (OAM, Object Authority Manager), einer standardmäßig mit dem Produkt verfügbaren Servicekomponente implementiert. Der OAM ist standardmäßig aktiv, d. h., Sie müssen ihn nicht erst konfigurieren. Sie können die Berechtigungsservice-Schnittstelle verwenden, um andere Komponenten zu erstellen, die den OAM ersetzen oder erweitern. Weitere Informationen zum OAM finden Sie in „Kapitel 7. MQSeries-Objekte schützen" auf Seite 81.

- **Namensservice**

  Der Namensservice ermöglicht die gemeinsame Benutzung von Warteschlangen, weil Anwendungen mit seiner Hilfe ferne Warteschlangen identifizieren können, so als ob es lokale Warteschlangen wären. Eine Standard-Servicekomponente, die den Namensservice implementiert, wird mit MQSeries Version 5.1 bereitgestellt. Sie verwendet OSF-DCE (Open Software Foundation - Distributed Computing Environment). Sie haben auch die Möglichkeit, eine eigene Namensservicekomponente zu erstellen, z. B., wenn Sie DCE nicht installiert haben. Der Namensservice ist standardmäßig inaktiv.

Weitere Informationen finden Sie in „Kapitel 12. Namensservice verwenden" auf Seite 177 sowie im Handbuch *MQSeries Programmable System Management*.

Sicherheit

Die Produkte von MQSeries Version 5 bieten zwei Methoden zur Bereitstellung von Sicherheit:

- den Objektberechtigungsmanager (OAM, Object Authority Manager)
- DCE-Sicherheit

Objektberechtigungsmanager (OAM)

Berechtigungen zur Verwendung von MQI-Aufrufen und Befehlen sowie für den Zugriff auf Objekte werden vom Objektberechtigungsmanager (OAM, Object Authority Manager) bereitgestellt, der standardmäßig aktiviert ist. Der Zugriff auf MQSeries-Definitionseinheiten wird über MQSeries-Benutzergruppen und den OAM gesteuert. Es steht eine Befehlszeilenschnittstelle zur Verfügung, über die Administratoren Berechtigungen nach Bedarf erteilen oder widerrufen können.

Weitere Informationen zum Erstellen von Berechtigungsservicekomponenten finden Sie im Handbuch *MQSeries Programmable System Management*.

DCE-Sicherheit

MQSeries stellt Kanal-Exits bereit, die DCE-GSS (Generic Security Service) verwenden. Weitere Informationen finden Sie im Handbuch *MQSeries Intercommunication*.

Transaktionsunterstützung

Ein Anwendungsprogramm kann ein Gruppe von Aktualisierungen zu einer *Arbeitseinheit* zusammenfassen. Diese Aktualisierungen sind in der Regel logisch miteinander verbunden und müssen alle erfolgreich ausgeführt werden, damit die Datenintegrität erhalten bleibt. Wenn eine Aktualisierung erfolgreich ist, eine andere aber nicht, geht die Datenintegrität verloren.

Wurde eine Arbeitseinheit erfolgreich abgeschlossen, wird eine COMMIT-Operation ausgeführt, d. h., die Aktualisierungen werden festgeschrieben. Ab diesem Zeitpunkt sind alle Aktualisierungen, die in dieser Arbeitseinheit ausgeführt wurden, permanent bzw. nicht mehr rückgängig zu machen. Kann die Arbeitseinheit nicht erfolgreich abgeschlossen werden, werden alle Aktualisierungen zurückgesetzt. Der Prozess, von dem Arbeitseinheiten entweder festgeschrieben (COMMIT-Operation) oder zurückgesetzt werden, heißt Synchronisationspunktkoordinierung.

Eine *lokale* Arbeitseinheit ist eine Arbeitseinheit, in der nur die Ressourcen des MQSeries-WS-Managers aktualisiert werden. In diesem Fall wird die Synchronisationspunktkoordinierung vom WS-Manager selbst mit Hilfe einer einphasigen Festschreibung bereitgestellt.

Eine *globale* Arbeitseinheit ist eine Arbeitseinheit, in der Ressourcen, die anderen Ressourcenmanagern zugeordnet sind, z. B. XA-kompatible Datenbanken, aktualisiert werden. In diesem Fall muss eine zweiphasige Festschreibung verwendet werden, und die Arbeitseinheit kann vom WS-Manager selbst koordiniert werden.

Weitere Informationen finden Sie in „Kapitel 10. Transaktionsunterstützung” auf Seite 125.

Kapitel 2. Eine Einführung in die MQSeries-Verwaltung

Dieses Kapitel gibt eine Einführung zum Thema MQSeries-Verwaltung.

Zu den Verwaltungs-Tasks gehören das Erstellen, Starten, Ändern, Anzeigen, Stoppen und Löschen von MQSeries-Objekten (WS-Manager, Warteschlangen, Prozesse, Namenslisten, Cluster und Kanäle).

Dieses Kapitel enthält die folgenden Abschnitte:

Lokale und ferne Verwaltung

MQSeries-Objekte werden lokal oder fern verwaltet.

Lokale Verwaltung bezeichnet die Ausführung von Verwaltungs-Tasks für WS-Manager, die Sie auf Ihrem lokalen System definiert haben. Sie können aber auch auf andere Systeme zugreifen, z. B. über das TCP/IP-Terminalemulationsprogramm **telnet**, und dort Verwaltungs-Tasks ausführen. In MQSeries kann dies ebenfalls als lokale Verwaltung angesehen werden, weil keine Kanäle benötigt werden, d. h., die Kommunikation wird vom Betriebssystem gesteuert.

MQSeries unterstützt die Verwaltung von einem Einzelpunkt aus. Dies wird als Fernverwaltung bezeichnet. Dadurch ist es Ihnen möglich, Befehle an Ihrem lokalen System auszugeben, die auf einem anderen System verarbeitet werden. Dazu müssen Sie sich nicht an dem anderen System anmelden, Sie müssen jedoch vorher die erforderlichen Kanäle definieren. Der WS-Manager und der Befehlsserver auf dem Zielsystem müssen aktiv sein. Sie können zum Beispiel einen fernen Befehl zum Ändern einer Warteschlangendefinition auf einem fernen WS-Manager ausgeben.

Nicht alle Befehle können auf diese Weise ausgegeben werden. Dies gilt insbesondere für Befehle zum Erstellen und Starten von WS-Managern und zum Starten von Befehlsservern. Um diese Art von Tasks auszuführen, müssen Sie sich entweder an dem fernen System anmelden und die Befehle dort ausgeben, oder Sie müssen einen Prozess erstellen, der die Befehle an Ihrer Stelle ausgeben kann.

In „Kapitel 6. Ferne MQSeries-Objekte verwalten" auf Seite 65 wird die Fernverwaltung ausführlich beschrieben.

Verwaltungs-Tasks mit Hilfe von Steuerbefehlen ausführen

Mit Hilfe von *Steuerbefehlen* können Sie Verwaltungs-Tasks für WS-Manager ausführen.

Weitere Informationen zu Steuerbefehlen finden Sie in „Kapitel 3. WS-Manager mit Hilfe von Steuerbefehlen verwalten" auf Seite 25.

Verwaltungs-Tasks mit Hilfe von MQSC-Befehlen ausführen

MQSC-Befehle (MQSeries Commands) werden zum Verwalten von WS-Manager-Objekten einschließlich des WS-Managers selbst, Kanälen, Warteschlangen und Prozessdefinitionen verwendet. So gibt es zum Beispiel Befehle zum Definieren, Ändern, Anzeigen und Löschen einer einzelnen Warteschlange.

Sie führen MQSC-Befehle aus, indem Sie den Steuerbefehl **runmqsc** in einer Befehlszeile eingeben. Sie können MQSC-Befehle folgendermaßen ausführen:

- interaktiv, indem Sie sie über die Tastatur eingeben (siehe „Die MQSC-Funktion interaktiv verwenden" auf Seite 37).
- als eine Befehlsfolge aus einer ASCII-Textdatei (siehe „MQSC-Befehle aus Textdateien ausführen" auf Seite 40).

Sie können den Befehl **runmqsc** in drei Modi ausführen, abhängig von den mit dem Befehl angegebenen Optionen:

- *Prüfmodus*, bei dem MQSC-Befehle auf einem lokalen WS-Manager geprüft, aber nicht wirklich ausgeführt werden.
- *direkter Modus*, bei dem MQSC-Befehle auf einem lokalen WS-Manager ausgeführt werden.
- *indirekter Modus*, bei dem MQSC-Befehle auf einem fernen WS-Manager ausgeführt werden.

Weitere Informationen zur Verwendung der MQSC-Funktionen und -Textdateien finden Sie unter „MQSC-Befehle aus Textdateien ausführen" auf Seite 40. Weitere Informationen zum Befehl **runmqsc** finden Sie unter „runmqsc (MQSeries-Befehle ausführen)" auf Seite 304.

Objektattribute, die in MQSC angegeben werden, werden in diesem Buch in Großbuchstaben geschrieben (z. B. RQMNAME), obwohl bei ihnen die Groß-/Kleinschreibung nicht beachtet werden muss. Namen von MQSC-Attributen dürfen maximal acht Zeichen lang sein.

MQSC-Befehle sind auf anderen Plattformen verfügbar, einschließlich AS/400 und OS/390.

Eine Zusammenfassung der MQSC-Befehle finden Sie in „Anhang D. Befehlssätze - Übersicht" auf Seite 335.

Eine Beschreibung der einzelnen MQSC-Befehle und der jeweiligen Syntax finden Sie im Handbuch *MQSeries MQSC-Befehle*.

Weitere Informationen zur Verwendung von MQSC-Befehlen bei der lokalen Verwaltung finden Sie unter „Lokale Verwaltungs-Tasks mit Hilfe von MQSC-Befehlen ausführen" auf Seite 36.

Verwaltungs-Tasks mit Hilfe von PCF-Befehlen ausführen

Die PCF-Befehle (Programmable Command Format) in MQSeries dienen dazu, Verwaltungs-Tasks innerhalb eines Verwaltungsprogramms auszuführen. Auf diese Weise können Sie aus einem Programm heraus Warteschlangen und Prozessdefinitionen erstellen sowie WS-Manager ändern.

PCF-Befehle decken denselben Funktionsbereich ab, der auch von der MQSC-Funktion bereitgestellt wird.

Weitere Informationen finden Sie unter „PCF-Befehle" auf Seite 61.

Eine vollständige Beschreibung der PCF-Datenstrukturen sowie Anweisungen zu deren Implementierung finden Sie im Handbuch *MQSeries Programmable System Management*.

Für einen einfacheren Programmierungszugriff auf PCF-Nachrichten können Sie die MQSeries-Verwaltungsschnittstelle (MQAI) verwenden. Dies wird unter „Verwendung von PCF-Befehlen mit Hilfe von MQAI vereinfachen" auf Seite 62 ausführlich beschrieben.

Attribute in MQSC- und PCF-Befehlen

Objektattribute, die in MQSC angegeben werden, werden in diesem Buch in Großbuchstaben geschrieben (z. B. RQMNAME), obwohl bei ihnen die Groß-/Kleinschreibung nicht beachtet werden muss. Die Attributnamen dürfen maximal acht Zeichen lang sein, so dass es bei einigen Namen, z. B. QDPHIEV, nicht gerade leicht ist, ihre Bedeutung zu erkennen. Objektattribute in PCF-Befehlen werden kursiv geschrieben. Die Länge ihrer Namen ist nicht auf acht Zeichen begrenzt, so dass sie einfacher zu verstehen sind. RQMNAME entspricht dem PCF-Befehl *RemoteQMgrName* und QDPHIEV dem PCF-Befehl *QDepthHighEvent*.

PCF-Escape-Befehle

PCF-Escape-Befehle sind PCF-Befehle, deren Nachrichtentext MQSC-Befehle enthalten. Mit PCF-Befehlen können Sie Befehle an einen fernen WS-Manager senden. Weitere Informationen zu PCF-Escape-Befehlen finden Sie im Handbuch *MQSeries Programmable System Management*.

Erläuterungen zu MQSeries-Dateinamen

Jede MQSeries-Warteschlange, jeder WS-Manager, jede Namensliste und jedes Prozessobjekt wird durch eine Datei repräsentiert. Da es sich bei Objektnamen nicht notwendigerweise um gültige Dateinamen handelt, konvertiert der WS-Manager den Objektnamen bei Bedarf in einen gültigen Dateinamen.

Der Pfad zu einem WS-Manager-Verzeichnis setzt sich aus folgenden Teilen zusammen:

- Einem Präfix - Der erste Teil des Namens:
  `MQS_ROOT:[MQM]`

  Dieses Präfix wird in der Konfigurationsdatei des WS-Managers definiert.
- Einem Literal:
  `QMGRS`

- Einem codierten WS-Manager-Namen; das ist der aus dem WS-Manager-Namen durch eine Umwandlung erstellte gültige Verzeichnisname. Aus dem WS-Manager-Namen

  ```
  QUEUE.MANAGER
  ```

  wird zum Beispiel der Verzeichnisname

  ```
  QUEUE$MANAGER
  ```

  Dieser Prozess wird als *Namensumwandlung* bezeichnet.

Umwandlung des WS-Manager-Namens

In MQSeries kann der Name eines WS-Managers maximal 48 Zeichen lang sein.

Der Name eines WS-Managers kann zum Beispiel folgendermaßen lauten:

```
QUEUE.MANAGER.ACCOUNTING.SERVICES
```

Allerdings wird jeder WS-Manager durch eine Datei repräsentiert, und sowohl für die maximale Länge eines Dateinamens als auch für die Zeichen, die der Name enthalten darf, gibt es Einschränkungen. Aus diesem Grund werden Namen von Dateien, die Objekte repräsentieren, automatisch umgewandelt, damit sie die Anforderungen des Dateisystems erfüllen.

Für die Umwandlung von WS-Manager-Namen (zum Beispiel für WS-Manager mit dem Namen `QUEUE.MANAGER`) gelten folgende Regeln:

1. Zeichen werden einzeln umgewandelt:
   · wird zu $
   / wird zu _
   % wird zu _
2. Der Name ist noch nicht gültig:
   a. Er wird auf acht Zeichen abgeschnitten.
   b. Es wird ein numerisches Suffix aus drei Zeichen angehängt.

Im Beispiel wird, das Standardpräfix vorausgesetzt, der WS-Manager-Name zu:

```
MQS_ROOT:[MQM.QMGRS.QUEUE$MANAGER]
```

Der Umwandlungsalgorithmus ermöglicht zudem auf Dateisystemen ohne Berücksichtigung der Groß-/Kleinschreibung eine Erkennung von Namen, die sich nur in der Groß-/Kleinschreibung unterscheiden.

Umwandlung von Objektnamen

Bei Objektnamen handelt es sich nicht notwendigerweise um gültige Dateinamen. Daher müssen die Objektnamen gegebenenfalls umgewandelt werden. Die verwendete Methode unterscheidet sich von der für die Umwandlung von WS-Manager-Namen, weil es für jeden WS-Manager eine große Zahl anderer Objekte geben kann, auch wenn es pro System nur ein paar WS-Manager gibt. Nur Prozessdefinitionen, Warteschlangen und Namenslisten werden im Dateisystem dargestellt; diese Überlegungen gelten nicht für Kanäle.

Wenn ein neuer Name durch den Umwandlungsprozess generiert wird, besteht keine direkte Beziehung zu dem ursprünglichen Objektnamen. Mit dem Befehl **dspmqfls** können Sie die ursprünglichen (realen) Objektnamen anzeigen.

Warteschlangendateinamen beginnen mit dem Buchstaben 'Q'.

Weitere Informationen zur Benennung von Objekten finden Sie unter „Regeln für die Benennung von MQSeries-Objekten" auf Seite 253.

Erläuterungen zur Groß-/Kleinschreibung

Groß-/Kleinschreibung in Steuerbefehlen

OpenVMS wird normalerweise als ein Betriebssystem beschrieben, das nicht zwischen Groß- und Kleinschreibung unterscheidet. Das heißt zum Beispiel, dass mit jedem der folgenden drei Befehle ein WS-Manager mit dem Namen QUEUEMANAGER erstellt werden kann.

```
$ crtmqm QueueManager
$ crtmqm queuemanager
$ crtmqm QUEUEMANAGER
```

In MQSeries for Compaq OpenVMS können Sie die Groß-/Kleinschreibung des Namens eines WS-Managers (oder eines ähnlichen Parameters) schützen, indem Sie ihn in doppelte Anführungszeichen setzen. Bei Verwendung von doppelten Anführungszeichen wird mit jedem der folgenden drei Befehle ein anderer WS-Manager erstellt.

```
$ crtmqm "QueueManager"   erstellt einen WS-Manager mit dem
Namen  QueueManager.
$ crtmqm "queuemanager"   erstellt einen WS-Manager mit dem Namen
queuemanager.
$ crtmqm "QUEUEMANAGER"   erstellt einen WS-Manager mit dem Namen
QUEUEMANAGER.
```

Erläuterungen zu MQSeries-Namen

Mit OpenVMS Version 7.2 wurde der folgende neue Befehl eingeführt:

```
$ set process /parse_style = ( traditional | extended )
```

Dieser Befehl ändert die Art und Weise, wie OpenVMS Zeichen in Groß- und Kleinschreibung bearbeitet.

Wenn der Befehl **set process /parse_style** nicht verwendet wird oder die Option **traditional** angegeben wird, verhält sich OpenVMS in Bezug auf die Groß-/Kleinschreibung wie bisher.

Wird der Befehl mit der Option **extended** angegeben, ändert dies das Verhalten der Laufzeit-Bibliotheksroutine LIB$GET_FOREIGN dahingehend, dass jeder von ihr abgerufene Text in seiner ursprünglichen Schreibweise erhalten bleibt. Da MQSeries diese Routine zum Abrufen von Befehlszeilenparametern verwendet, bleibt die Groß-/Kleinschreibung der Parameter erhalten, auch wenn die Parameter nicht in Anführungszeichen gesetzt sind.

Zum Beispiel werden mit der folgenden Befehlsfolge drei verschiedene WS-Manager erstellt. Beachten Sie, dass die Parameter nicht in Anführungszeichen gesetzt sind.

```
$ set process /parse_style = extended
$ crtmqm QueueManager   erstellt einen WS-Manager mit dem Namen
QueueManager.
$ crtmqm queuemanager   erstellt einen WS-Manager mit dem Namen
queuemanager.
$ crtmqm QUEUEMANAGER   erstellt einen WS-Manager mit dem Namen
QUEUEMANAGER
```

Der OpenVMS-Befehl 'set process /parse_style' ändert nicht nur die Behandlung der Groß-/Kleinschreibung, sondern bietet weitere Änderungsfunktionen. Es wird empfohlen, die Informationen zu diesem Befehl im OpenVMS DCL Dictionary zu lesen, bevor Sie ihn auf Ihrem System anwenden.

Groß-/Kleinschreibung in MQSC-Befehlen

Bei MQSeries-Steuerbefehlen (z. B. **runmqsc** zum Aufrufen der MQSC-Funktion) wird nicht zwischen Groß- und Kleinschreibung unterschieden.

MQSC-Befehle und deren Attribute können sowohl in Klein- als auch in Großbuchstaben eingegeben werden. Objektnamen in MQSC-Befehlen werden automatisch in Großbuchstaben umgesetzt, es sei denn, die Namen sind in *einfache* Anführungszeichen gesetzt. Ist der Name nicht in einfache Anführungszeichen gesetzt, wird das Objekt mit einem Namen in Großbuchstaben verarbeitet. Weitere Informationen finden Sie im Handbuch *MQSeries MQSC-Befehle*.

Kapitel 3. WS-Manager mit Hilfe von Steuerbefehlen verwalten

In diesem Kapitel wird beschrieben, wie Sie Operationen für WS-Manager und Befehlsserver ausführen können. Es enthält die folgenden Abschnitte:

Steuerbefehle verwenden

Steuerbefehle werden dazu verwendet, Operationen für WS-Manager, Befehlsserver und Kanäle auszuführen. Steuerbefehle können in drei Kategorien eingeteilt werden (siehe Tabelle 1 auf Seite 25).

Tabelle 1. Kategorien von Steuerbefehlen

Kategorie	Beschreibung
WS-Manager-Befehle	Zu den Steuerbefehlen für WS-Manager gehören Befehle zum Erstellen, Starten, Stoppen und Löschen von WS-Managern und Befehlsservern.
Kanalbefehle	Zu den Kanalbefehlen gehören Befehle zum Starten und Beenden von Kanälen und Kanalinitiatoren.
Dienstprogrammbefehle	Zu den Dienstprogrammbefehlen gehören Befehle für folgende Aufgaben und Funktionen: • Ausführen von MQSC-Befehlen • Konvertierungs-Exits • Berechtigungsverwaltung • Aufzeichnen und Wiederherstellen von Datenträger-Images von WS-Manager-Ressourcen • Anzeigen und Auflösen von Transaktionen • Auslösemonitore • Anzeigen der Dateinamen von MQSeries-Objekten

Informationen zu Verwaltungs-Tasks für Kanäle finden Sie im Handbuch *MQSeries Intercommunication*.

Steuerbefehle verwenden

In MQSeries for Compaq OpenVMS werden Steuerbefehle an einer DCL-Eingabeaufforderung eingegeben. Bei dem Befehlsnamen und den Optionen wird nicht zwischen Groß- und Kleinschreibung unterschieden, die Parameter werden

jedoch abhängig von der OpenVMS-Prozessoption und davon, ob sie zum Schutz der Groß-/Kleinschreibung in doppelte Anführungszeichen gesetzt sind, gegebenenfalls in Großbuchstaben umgesetzt. Weitere Informationen zur Auswirkung des OpenVMS-Befehls und von doppelten Anführungszeichen auf die Groß-/Kleinschreibung finden Sie unter „Erläuterungen zur Groß-/Kleinschreibung" auf Seite 23.

Betrachten wir hierzu folgenden Beispielbefehl:

```
crtmqm -u system.dead.letter.queue "jupiter.queue.manager"
```

- Der Name der Warteschlange für nicht zustellbare Nachrichten kann zu SYSTEM.DEAD.LETTER.QUEUE werden, obwohl er in Kleinbuchstaben eingegeben wurde. Ob der Name automatisch in Großbuchstaben umgesetzt wird, ist von der Einstellung des OpenVMS-Befehls **set process/parse_style** abhängig (siehe „Erläuterungen zur Groß-/Kleinschreibung" auf Seite 23).
- Der Name des WS-Managers ist mit ″jupiter.queue.manager″ angegeben und unterscheidet sich damit von ″JUPITER.queue.manager″, weil er in doppelte Anführungszeichen gesetzt wurde.

Geben Sie Befehle daher genau so ein, wie sie in den Beispielen angegeben sind.

Einen WS-Manager erstellen

Ein WS-Manager verwaltet die ihm zugeordneten Ressourcen, insbesondere seine Warteschlangen. Er stellt Anwendungen Queuing-Services für MQI-Aufrufe und Befehle zum Erstellen, Ändern, Anzeigen und Löschen von MQSeries-Objekten zur Verfügung.

Bevor Sie mit Nachrichten und Warteschlangen arbeiten können, müssen Sie mindestens einen WS-Manager und die ihm zugeordneten Objekte erstellen. Ein WS-Manager wird mit dem MQSeries-Steuerbefehl **crtmqm** erstellt. Der Befehl **crtmqm** erstellt automatisch die erforderlichen Standardobjekte sowie Systemobjekte. Standardobjekte bilden die Basis für alle von Ihnen erstellten Objektdefinitionen, Systemobjekte sind für den WS-Manager-Betrieb erforderlich. Nachdem ein WS-Manager und seine Objekte erstellt wurden, können Sie den WS-Manager mit dem Befehl **strmqm** starten.

Richtlinien für die Erstellung von WS-Managern

Bevor Sie einen WS-Manager erstellen, müssen Sie verschiedene Punkte überprüfen (insbesondere in einer Produktionsumgebung). Arbeiten Sie die folgende Prüfliste durch:

- einen eindeutigen WS-Manager-Namen angeben
- die Anzahl der WS-Manager begrenzen
- einen Standard-WS-Manager angeben
- eine Warteschlange für nicht zustellbare Nachrichten angeben
- eine Standardübertragungswarteschlange angeben
- die erforderlichen Protokollierungsparameter angeben
- die Konfigurationsdateien nach dem Erstellen eines WS-Managers sichern

Die Tasks in dieser Liste werden im folgenden Abschnitt erläutert.

Einen eindeutigen WS-Manager-Namen angeben

Achten Sie beim Erstellen eines WS-Managers darauf, dass es in Ihrem *gesamten* Netz keinen anderen WS-Manager mit demselben Namen gibt. WS-Manager-Namen werden zum Zeitpunkt der Erstellung nicht überprüft, und nur mit Namen, die eindeutig sind, können Sie Kanäle für verteiltes Queuing verwenden.

Eine Möglichkeit, Eindeutigkeit sicherzustellen, besteht darin, jeden WS-Manager-Namen mit einem Präfix zu versehen, das seinem jeweiligen (eindeutigen) Knotennamen entspricht. Wenn es zum Beispiel einen Knoten mit dem Namen `accounts` gibt, können Sie dem WS-Manager den Namen `accounts.saturn.queue.manager` geben, wobei `saturn` einen bestimmten WS-Manager identifiziert und `queue.manager` eine Erweiterung darstellt, die Sie für alle WS-Manager verwenden. Sie können diese Erweiterung auch weglassen, müssen dann aber beachten, dass es sich bei `accounts.saturn` und `accounts.saturn.queue.manager` um *verschiedene* WS-Manager-Namen handelt.

Wenn Sie MQSeries für die Kommunikation mit anderen Unternehmen verwenden, können Sie den Namen Ihres Unternehmens als Präfix angeben. Diese Möglichkeit wird in den Beispielen aus Gründen der Verständlichkeit jedoch nicht genutzt.

Anmerkung: WS-Manager-Namen in Steuerbefehlen werden abhängig von der OpenVMS-Prozessoption und davon, ob sie zum Schutz der Groß-/Kleinschreibung in doppelte Anführungszeichen gesetzt sind, gegebenenfalls in Großbuchstaben umgesetzt. Dies bedeutet, dass Sie zwei WS-Manager mit den Namen `jupiter.queue.manager` und `JUPITER.queue.manager` erstellen können. Weitere Informationen zur Auswirkung der OpenVMS-Prozessoption und von doppelten Anführungszeichen auf die Groß-/Kleinschreibung finden Sie unter „Erläuterungen zur Groß-/Kleinschreibung" auf Seite 23.

Die Anzahl der WS-Manager begrenzen

Sie können so viele WS-Manager erstellen, wie es die vorhandenen Ressourcen zulassen. Da jedoch jeder WS-Manager seine eigenen Ressourcen benötigt, ist es generell besser, auf einem Knoten einen WS-Manager mit 100 Warteschlangen als zehn WS-Manager mit jeweils zehn Warteschlangen zu verwalten.

Auf Produktionssystemen werden viele Knoten gemeinsam mit einem einzigen WS-Manager ausgeführt, auf größeren Servermaschinen können jedoch mehrere WS-Manager ausgeführt werden.

Den Standard-WS-Manager angeben

Jeder Knoten sollte über einen Standard-WS-Manager verfügen, auch wenn es möglich ist, MQSeries auf einem Knoten ohne Standard-WS-Manager zu konfigurieren.

Ein WS-Manager wird mit dem Befehl **crtmqm** erstellt. Eine detaillierte Beschreibung dieses Befehls und seiner Parameter finden Sie unter „crtmqm (WS-Manager erstellen)" auf Seite 260.

Was ist ein Standard-WS-Manager?

Der Standard-WS-Manager ist der WS-Manager, mit dem Anwendungen verbunden werden, wenn sie in einem Aufruf MQCONN keinen WS-Manager-Namen angeben. Der Standard-WS-Manager ist außerdem für die Verarbeitung von MQSC-Befehlen zuständig, wenn Sie den Befehl **runmqsc** aufrufen, ohne einen WS-Manager-Namen anzugeben.

Wie wird ein Standard-WS-Manager angegeben?

Geben Sie dazu im Befehl **crtmqm** die Option -q an, um festzulegen, dass es sich bei dem zu erstellenden WS-Manager um den Standard-WS-Manager handelt. Lassen Sie diese Option weg, wenn der WS-Manager, den Sie erstellen, kein Standard-WS-Manager sein soll.

Wenn Sie einen WS-Manager als Standard-WS-Manager angeben, wird die Spezifikation eines eventuell bereits vorhandenen Standard-WS-Managers auf dem Knoten durch die neuen Angaben *ersetzt*.

Was geschieht bei einer Änderung des Standard-WS-Managers?

Wenn Sie den Standard-WS-Manager ändern, müssen Sie berücksichtigen, dass sich dies auf andere Benutzer oder Anwendungen auswirken kann. Die Änderung hat keine Auswirkung auf Anwendungen mit bestehenden Verbindungen, weil die Anwendungen bei jedem weiteren MQI-Aufruf die Kennung aus ihrem ursprünglichen Verbindungsaufruf verwenden können. Diese Kennung stellt sicher, dass die Aufrufe an denselben WS-Manager gerichtet werden. Alle Anwendungen, die nach der Änderung eine Verbindung herstellen, werden mit dem neuen Standard-WS-Manager verbunden.

Möglicherweise ist genau dies Ihre Absicht, Sie sollten es jedoch immer berücksichtigen, bevor Sie den Standard-WS-Manager ändern.

Eine Warteschlange für nicht zustellbare Nachrichten angeben

Die Warteschlange für nicht zustellbare Nachrichten ist eine lokale Warteschlange, in die Nachrichten eingereiht werden, die nicht an ihre eigentliche Zieladresse weitergeleitet werden können.

Achtung:
Es ist unbedingt erforderlich, dass jeder WS-Manager in Ihrem Netz über eine Warteschlange für nicht zustellbare Nachrichten verfügt. Wenn nicht, können Fehler in Anwendungsprogrammen dazu führen, dass Kanäle geschlossen oder Antworten zu Verwaltungsbefehlen nicht empfangen werden.

Wenn beispielsweise eine Anwendung versucht, eine Nachricht in eine Warteschlange eines anderen WS-Managers einzureihen, aber einen falschen Warteschlangennamen angibt, wird der Kanal gestoppt und die Nachricht verbleibt in der Übertragungswarteschlange. Andere Anwendungen können diesen Kanal dann nicht mehr für ihre Nachrichten verwenden.

Diese Auswirkung auf die Kanäle wird vermieden, wenn die WS-Manager über Warteschlangen für nicht zustellbare Nachrichten verfügen. Die nicht zustellbare Nachricht wird dann an ihrer letzten Empfangsstation einfach in die Warteschlange für nicht zustellbare Nachrichten eingereiht, und der Kanal und seine Übertragungswarteschlange sind weiter verfügbar.

Deshalb sollten Sie beim Erstellen eines WS-Managers die Option -u verwenden, um den Namen der Warteschlange für nicht zustellbare Nachrichten anzugeben. Sie können auch einen MQSC-Befehl verwenden, um die Attribute eines WS-Managers zu ändern und die zu verwendende Warteschlange für nicht zustellbare Nachrichten anzugeben. Ein Beispiel für den MQSC-Befehl ALTER finden Sie unter „WS-Manager-Attribute ändern” auf Seite 40.

Wenn Sie Nachrichten in einer Warteschlange für nicht zustellbare Nachrichten finden, können Sie diese Nachrichten mit Hilfe der in MQSeries integrierten Steuerroutine der Warteschlange für nicht zustellbare Nachrichten verarbeiten. Weitere Informationen zur Steuerroutine der Warteschlange für nicht zustellbare Nachrichten und dazu, wie Sie die Anzahl der Nachrichten, die ansonsten möglicherweise in die Warteschlange für nicht zustellbare Nachrichten eingereiht werden, senken können, finden Sie unter „Kapitel 8. MQSeries-Steuerroutine der Warteschlange für nicht zustellbare Nachrichten" auf Seite 105.

Eine Standardübertragungswarteschlange angeben

Eine Übertragungswarteschlange ist eine lokale Warteschlange, in die Nachrichten an einen fernen WS-Manager für die anstehende Übertragung eingereiht werden. Die Standardübertragungswarteschlange wird dann verwendet, wenn keine explizite Übertragungswarteschlange definiert wurde. Jedem WS-Manager kann eine Standardübertragungswarteschlange zugeordnet werden.

Verwenden Sie beim Erstellen eines WS-Managers die Option -d, um den Namen der Standardübertragungswarteschlange anzugeben. Dadurch wird die Warteschlange jedoch nicht tatsächlich erstellt; dies müssen Sie später explizit tun. Weitere Informationen finden Sie unter „Mit lokalen Warteschlangen arbeiten" auf Seite 45.

Die erforderlichen Protokollierungsparameter angeben

Mit dem Befehl **crtmqm** können Sie Protokollierungsparameter wie den Protokolltyp, den Pfad und die Größe der Protokolldatei angeben. In einer Entwicklungsumgebung können Sie normalerweise die Standardwerte für die Protokollierungsparameter verwenden. Sie können die Standardwerte jedoch ändern, z. B. in folgenden Fällen:

- Sie verfügen über ein System im unteren Leistungsbereich, das keine großen Protokolle unterstützt.
- Sie gehen davon aus, dass sich in Ihren Warteschlangen gleichzeitig sehr viele lange Nachrichten befinden.

Weitere Informationen zur Angabe von Protokollierungsparametern

- mit Hilfe des Befehls **crtmqm** finden Sie unter „crtmqm (WS-Manager erstellen)" auf Seite 260,
- mit Hilfe von Konfigurationsdateien finden Sie unter „Die Zeilengruppe Log" auf Seite 188.

Die Konfigurationsdateien nach dem Erstellen eines WS-Managers sichern

Hier sind zwei Konfigurationsdateien zu beachten:

1. Bei der Produktinstallation wird die MQSeries-Konfigurationsdatei (mqs.ini) erstellt. Sie enthält eine Liste mit WS-Managern, die jedes Mal aktualisiert wird, wenn Sie einen WS-Manager erstellen oder löschen. Es gibt pro Knoten eine Datei mqs.ini.
2. Wenn Sie einen neuen WS-Manager erstellen, wird automatisch eine neue WS-Manager-Konfigurationsdatei (qm.ini) erstellt. Sie enthält Konfigurationsparameter für den WS-Manager.

Von diesen Dateien sollten Sie Sicherungen erstellen. Wenn Sie später einen anderen WS-Manager erstellen und dabei Probleme auftreten, können Sie die Sicherungen wiederherstellen, nachdem Sie die Ursache des Problems beseitigt haben. Als generelle Regel wird empfohlen, die Konfigurationsdateien jedes Mal zu sichern, wenn Sie einen neuen WS-Manager erstellen.

Weitere Informationen zu Konfigurationsdateien finden Sie in „Kapitel 13. MQSeries konfigurieren” auf Seite 179.

Einen Standard-WS-Manager erstellen

Ein Standard-WS-Manager wird mit dem Befehl **crtmqm** erstellt. Der Befehl **crtmqm** mit der Option -q führt Folgendes aus:

- Er erstellt einen Standard-WS-Manager mit dem Namen `saturn.queue.manager`.
- Er erstellt die Standard- und Systemobjekte.
- Er gibt die Namen der Standardübertragungswarteschlange und der Warteschlange für nicht zustellbare Nachrichten des WS-Managers an.

```
crtmqm -q -d MY.DEFAULT.XMIT.QUEUE -u SYSTEM.DEAD.LETTER.QUEUE "saturn.queue.manager"
```

Dabei gilt:

-q Gibt an, dass dieser WS-Manager der Standard-WS-Manager ist.

-d MY.DEFAULT.XMIT.QUEUE
Dies ist der Name der Standardübertragungswarteschlange.

-u SYSTEM.DEAD.LETTER.QUEUE
Dies ist der Name der Warteschlange für nicht zustellbare Nachrichten.

“saturn.queue.manager”
Dies ist der Name des WS-Managers. Bei dem Befehl **crtmqm** muss dies der letzte Parameter sein.

Ein Standard-WS-Manager hat den Vorteil, dass Sie für Ihn einige Befehle (z. B. **strmqm** und **runmqsc**) ausgeben können, ohne einen WS-Manager-Namen angeben zu müssen. Bei anderen Befehlen (z. B. **endmqm** und **dltmqm**) muss ein WS-Manager-Name angegeben werden.

Beachten Sie, dass der WS-Manager-Name in diesem Beispiel in Kleinbuchstaben geschrieben ist und die Kleinschreibung durch doppelte Anführungszeichen geschützt ist. Weitere Informationen zur Groß-/Kleinschreibung bei Parametern finden Sie unter „Erläuterungen zur Groß-/Kleinschreibung” auf Seite 23.

Einen WS-Manager starten

Damit ein von Ihnen erstellter WS-Manager Befehle verarbeiten kann, müssen Sie ihn starten. Starten Sie den WS-Manager, indem Sie folgenden Befehl eingeben:

```
strmqm "saturn.queue.manager"
```

Der Befehl **strmqm** gibt die Steuerung erst zurück, nachdem der WS-Manager gestartet wurde und bereit ist, Verbindungsanforderungen zu empfangen.

Einen vorhandenen WS-Manager zum Standard-WS-Manager erklären

Wenn Sie einen Standard-WS-Manager erstellen, wird der Name des Standard-WS-Managers in die Zeilengruppe *DefaultQueueManager* in der MQSeries-Konfigurationsdatei (mqs.ini) eingefügt. Die Zeilengruppe und deren Inhalt werden automatisch generiert, sofern Sie noch nicht vorhanden sind.

In folgenden Fällen müssen Sie die Zeilengruppe editieren:

- **Sie möchten einen vorhandenen WS-Manager zum Standard-WS-Manager erklären.** Zu diesem Zweck müssen Sie den WS-Manager-Namen in der Zeilengruppe durch den Namen des neuen Standard-WS-Managers ersetzen. Dies muss manuell in einem Texteditor erfolgen.
- **Der Knoten verfügt nicht über einen Standard-WS-Manager, und Sie wollen einen vorhandenen WS-Manager zum Standard-WS-Manager erklären.** Zu diesem Zweck müssen Sie die Zeilengruppe *DefaultQueueManager* mit dem erforderlichen Namen selbst erstellen.
- **Sie haben versehentlich einen anderen WS-Manager zum Standard-WS-Manager erklärt und wollen den ursprünglichen Standard-WS-Manager wiederherstellen.** Editieren Sie zu diesem Zweck die Zeilengruppe *DefaultQueueManager* in der MQSeries-Konfigurationsdatei, und ersetzen Sie den Namen des falschen Standard-WS-Managers durch den Namen des gewünschten Standard-WS-Managers.

Weitere Informationen zu Konfigurationsdateien finden Sie in „Kapitel 13. MQSeries konfigurieren” auf Seite 179. Nachdem Sie die erforderlichen Änderungen in der Zeilengruppe vorgenommen haben, stoppen Sie den WS-Manager, und starten Sie ihn erneut.

Einen WS-Manager stoppen

Ein WS-Manager wird mit dem Befehl **endmqm** gestoppt. Geben Sie zum Beispiel zum Stoppen des WS-Managers mit dem Namen `saturn.queue.manager` folgenden Befehl ein:

```
endmqm "saturn.queue.manager"
```

Gesteuerter Abschluss

Der Befehl **endmqm** führt standardmäßig einen *gesteuerten* Abschluss des angegebenen WS-Manager aus. Dies kann eine Weile dauern, denn bei einem gesteuerten Abschluss wird gewartet, bis *alle* verbundenen Anwendungen ihre Verbindungen getrennt haben.

Verwenden Sie diese Abschlussart, damit Anwendungen mitgeteilt wird, dass sie stoppen sollen. Wenn Sie den Befehl

```
endmqm -c "saturn.queue.manager"
```

eingeben, erhalten Sie keine Rückmeldung, nachdem alle Anwendungen gestoppt wurden. (Der Befehl endmqm -c “saturn.queue.manager” entspricht dem Befehl endmqm “saturn.queue.manager”.)

Sofortiger Abschluss

Bei einem sofortigen Abschluss werden alle aktuellen MQI-Aufrufe noch ausgeführt, aber neue Aufrufe nicht mehr akzeptiert. Bei dieser Abschlussart wird nicht gewartet, bis Anwendungen ihre Verbindungen mit dem WS-Manager trennen.

Dies ist der normale Weg zum Stoppen des WS-Managers, nachdem ein gesteuerter Abschluss nicht innerhalb der üblichen Zeit erfolgreich abgeschlossen wurde. Geben Sie für einen sofortigen Abschluss folgenden Befehl ein:

```
endmqm -i "saturn.queue.manager"
```

Erzwungener Abschluss

Achtung: Verwenden Sie diese Methode nur, wenn alle anderen Versuche, den WS-Manager mit dem Befehl **endmqm** zu stoppen, erfolglos waren. Diese Methode kann unvorhersehbare Folgen für verbundene Anwendungen haben.

Wenn ein sofortiger Abschluss erfolglos ist, müssen Sie auf einen *erzwungenen* Abschluss zurückgreifen, indem Sie die Option -p angeben. Beispiel:

```
endmqm -p "saturn.queue.manager"
```

Dieser Befehl stoppt sofort den gesamten WS-Manager-Code.

Anmerkung: Nach einem erzwungenen Abschluss oder einem Fehler des WS-Managers wurde der WS-Manager möglicherweise beendet, ohne dass der von ihm verwaltete gemeinsam benutzte Speicher bereinigt wurde. Dies kann zu Problemen beim Neustart führen. Informationen zur Verwendung des Dienstprogramms MONMQ zum Bereinigen des Speichers nach einer abrupten Beendigung dieser Art finden Sie unter „Gemeinsam benutzten Speicher mit MONMQ verwalten” auf Seite 375.

Probleme beim Abschluss eines WS-Managers

Probleme beim Abschluss eines WS-Manager werden häufig durch Anwendungen verursacht. Zum Beispiel, wenn Anwendungen

- MQI-Rückkehrcodes nicht ordnungsgemäß überprüfen,
- keine Benachrichtigung über einen gesteuerten Abschluss anfordern,
- beendet werden, ohne dass die Verbindung mit dem WS-Manager getrennt wird (durch Ausgeben des Aufrufs MQDISC).

Wenn der Abschluss eines WS-Manager sehr lange dauert oder Sie glauben, dass der WS-Manager nicht gestoppt werden kann, können Sie den Befehl **endmqm** durch Drücken der Tastenkombination Strg-Y abbrechen. Danach können Sie einen anderen **endmqm**-Befehl eingeben, dieses Mal jedoch mit einer Option für einen sofortigen oder erzwungenen Abschluss.

Eine detaillierte Beschreibung des Befehls **endmqm** und seiner Optionen finden Sie unter „endmqm (WS-Manager beenden)” auf Seite 282.

Einen WS-Manager erneut starten

Verwenden Sie zum erneuten Starten eines WS-Manager den folgenden Befehl:

```
strmqm "saturn.queue.manager"
```

Einen WS-Manager löschen

Wenn Sie einen WS-Manager löschen wollen, stoppen Sie ihn zunächst, und geben Sie dann folgenden Befehl ein:

```
dltmqm "saturn.queue.manager"
```

Achtung: Das Löschen eines WS-Managers ist ein drastischer Schritt, weil Sie damit gleichzeitig alle ihm zugeordneten Ressourcen löschen. Das gilt nicht nur für alle Warteschlangen und die darin enthaltenen Nachrichten, sondern auch für alle Objektdefinitionen.

Eine Beschreibung des Befehls **dltmqm** und seiner Optionen finden Sie unter „dltmqm (WS-Manager löschen)" auf Seite 264. Stellen Sie sicher, dass nur vertrauenswürdige Administratoren über die Berechtigung zum Ausführen dieses Befehls verfügen.

Kapitel 4. Lokale MQSeries-Objekte verwalten

In diesem Kapitel Kapitel wird beschrieben, wie lokale MQSeries-Objekte verwaltet werden, um Anwendungsprogramme zu unterstützen, die MQI (Message Queue Interface, Schnittstelle für Nachrichtenwarteschlangen) verwenden. Lokale Verwaltung bedeutet in diesem Kontext das Erstellen, Anzeigen, Ändern, Kopieren und Löschen von MQSeries-Objekten.

Dieses Kapitel enthält folgende Abschnitte:

Unterstützen von Anwendungsprogrammen, die MQI verwenden

MQSeries-Anwendungsprogramme benötigen bestimmte Objekte, damit sie erfolgreich ausgeführt werden können. Abb. 1 zeigt das Beispiel einer Anwendung, die Nachrichten aus einer Warteschlange abruft, sie verarbeitet und Ergebnisse an eine andere Warteschlange desselben WS-Managers sendet.

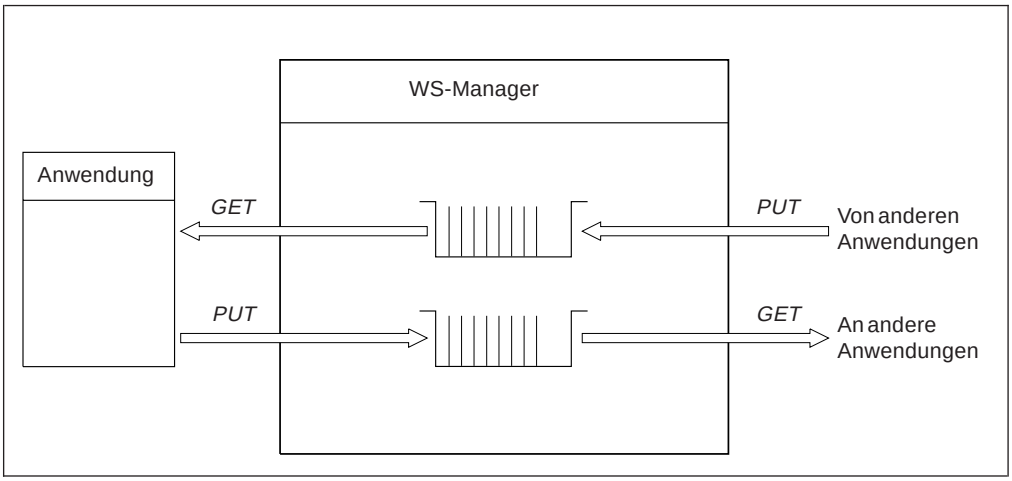


Abbildung 1. Warteschlangen, Nachrichten und Anwendungen

Anwendungen können Nachrichten zwar in lokale und in ferne Warteschlangen einreihen (mit MQPUT), Nachrichten direkt abrufen können Sie jedoch nur aus lokalen Warteschlangen (mit MQGET).

Bevor diese Anwendung ausgeführt werden kann, müssen folgende Bedingungen erfüllt sein:

- Der WS-Manager muss vorhanden und aktiv sein.
- Die erste Anwendungswarteschlange, aus der die Nachrichten abgerufen werden, muss definiert sein.

- Die zweite Warteschlange, in die die Nachrichten von den Anwendungen eingereiht werden, muss ebenfalls definiert sein.
- Die Anwendung muss eine Verbindung mit dem WS-Manager herstellen können. Dazu muss sie mit dem Produktcode verbunden sein. Weitere Informationen finden Sie im Handbuch *MQSeries Application Programming Guide*.
- Die Anwendungen, die die Nachrichten in die erste Warteschlange einreihen, müssen ebenfalls mit einem WS-Manager verbunden sein. Handelt es sich um ferne Anwendungen, müssen für sie außerdem Übertragungswarteschlangen und Kanäle konfiguriert sein. Dieser Teil des Systems wird in Abb. 1 auf Seite 35 nicht gezeigt.

Lokale Verwaltungs-Tasks mit Hilfe von MQSC-Befehlen ausführen

In diesem Abschnitt wird angenommen, dass Sie Befehle mit Hilfe von **runmqsc** ausgeben. Dies kann interaktiv erfolgen, durch Eingeben der Befehle über die Tastatur, oder indem Sie SYS$INPUT umleiten, um eine Befehlsfolge aus einer ASCII-Textdatei auszuführen. Das Format der Befehle ist in beiden Fällen dasselbe.

Eine Beschreibung der einzelnen MQSC-Befehle und der jeweiligen Syntax finden Sie im Handbuch *MQSeries MQSC-Befehle*.

MQSC-Befehle (MQSeries Script Commands) können zum Verwalten von WS-Manager-Objekten einschließlich des WS-Managers selbst, Clustern, Kanälen, Warteschlangen, Namenslisten und Prozessdefinitionen verwendet werden. Dieser Abschnitt behandelt WS-Manager, Warteschlangen und Prozessdefinitionen. Informationen zur Verwaltung von Kanalobjekten finden Sie unter 'DQM implementation' im Handbuch *MQSeries Intercommunication*.

MQSC-Befehle für einen WS-Manager werden mit Hilfe des Befehls **runmqsc** ausgegeben. Dies kann interaktiv erfolgen, durch Eingeben der Befehle über die Tastatur, oder indem Sie die Standardeingabe umleiten, um eine Befehlsfolge aus einer ASCII-Textdatei auszuführen. Das Format der Befehle ist in beiden Fällen dasselbe.

Sie können den Befehl **runmqsc** in drei Modi ausführen, abhängig von den mit dem Befehl angegebenen Optionen:

- *Prüfmodus*, bei dem MQSC-Befehle auf einem lokalen WS-Manager geprüft, aber nicht wirklich ausgeführt werden.
- *direkter Modus*, bei dem MQSC-Befehle auf einem lokalen WS-Manager ausgeführt werden.
- *indirekter Modus*, bei dem MQSC-Befehle auf einem fernen WS-Manager ausgeführt werden.

Objektattribute, die in MQSC angegeben werden, werden in diesem Buch in Großbuchstaben geschrieben (z. B. RQMNAME), obwohl bei ihnen die Groß-/Kleinschreibung nicht beachtet werden muss. (Weitere Informationen zur Groß-/Kleinschreibung finden Sie unter „Groß-/Kleinschreibung in MQSC-Befehlen" auf Seite 24.) Namen von MQSC-Attributen dürfen maximal acht Zeichen lang sein.

Vor dem Start

Bevor Sie MQSC-Befehle ausführen können, müssen Sie den WS-Manager, der die Befehle ausführen soll, erstellt und gestartet haben (siehe „Einen Standard-WS-Manager erstellen" auf Seite 30.

MQSeries-Objektnamen

In den Beispielen werden lange Namen für Objekte verwendet. Dies soll Ihnen helfen, den Objekttyp, mit dem sie es zu tun haben, zu identifizieren.

Wenn Sie MQSC-Befehle ausgeben, müssen Sie nur den lokalen Namen der Warteschlange angeben. In den Beispielen werden Namen wie dieser verwendet:

```
ORANGE.LOCAL.QUEUE
```

Der Teil LOCAL.QUEUE des Namens wurde gewählt, damit leicht erkennbar ist, dass es sich um eine lokale Warteschlange handelt. Es bedeutet *nicht*, dass Namen von lokalen Warteschlangen generell so anfangen müssen.

Als Name des WS-Managers wird `saturn.queue.manager` verwendet.

Der Teil `queue.manager` des Namens wurde gewählt, damit leicht erkennbar ist, dass es sich bei diesem Objekt um einen WS-Manager handelt. Es bedeutet *nicht*, dass Namen von WS-Managern generell so lauten müssen.

Sie können auch andere Namen verwenden, allerdings müssen Sie dann in den Beispielen alle Befehle mit diesen Namen entsprechend ändern.

Eingabe und Ausgabe umleiten

Um die Migration von anderen Betriebssystemen nach OpenVMS weiter zu erleichtern, unterstützt MQSeries die UNIX-Darstellung der Umleitungsanzeiger für sys$input, sys$output und sys$error wie folgt:

< gibt die Quelle für SYS$INPUT an.
> gibt die Quelle für SYS$OUTPUT an.
2> gibt die Quelle für SYS$ERROR an.

Diese Funktion ist auch in den ausführbaren Versionen der Beispielprogramme enthalten. Sie ist jedoch nicht im Quellcode der Beispiele enthalten, so dass sie nicht mehr verfügbar ist, wenn Sie die Beispiele aus dem Quellcode neu erstellen.

Die MQSC-Funktion interaktiv verwenden

Um Befehle interaktiv zu verwenden, geben Sie an einer DCL-Eingabeaufforderung Folgendes ein:

```
runmqsc
```

In diesem Befehl wurde kein WS-Manager-Name angegeben, d. h., die MQSC-Befehle werden vom Standard-WS-Manager verarbeitet. Jetzt können Sie jeden beliebigen MQSC-Befehl eingeben. Geben Sie zum Beispiel folgenden Befehl ein:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

Um anzuzeigen, dass ein Befehl in der nächsten Zeile fortgesetzt wird, müssen Fortsetzungszeichen verwendet werden:

- Ein Minuszeichen (-) gibt an, dass der Befehl am Anfang der nächsten Zeile fortgesetzt wird.
- Ein Pluszeichen (+) gibt an, dass der Befehl mit dem ersten belegten Zeichen in der nächsten Zeile fortgesetzt wird.

MQSC-Befehle ausgeben

Die Befehlseingabe endet mit dem Schlusszeichen einer belegten Zeile, bei dem es sich nicht um ein Fortsetzungszeichen handelt. Sie können eine Befehlseingabe auch explizit durch Eingabe eines Strichpunktes (;) beenden. (Dies ist besonders dann hilfreich, wenn Sie am Ende der letzten Zeile einer Befehlseingabe versehentlich ein Fortsetzungszeichen eingeben.)

Rückmeldung von MQSC-Befehlen

Wenn Sie Befehle der MQSC-Funktion ausgeben, werden vom WS-Manager Bedienernachrichten zurückgegeben, um Ihre Aktionen zu bestätigen oder Ihnen mitzuteilen, welche Fehler Sie gemacht haben. Beispiel:

```
AMQ8006: Die MQSeries-Warteschlange wurde erstellt.
 .
 .
 .
AMQ8405: Syntaxfehler in oder bei folgendem Befehlssegment:-
z

AMQ8426: Gültige MQSC-Befehle:

    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
```

Durch die erste Nachricht wird bestätigt, dass eine Warteschlange erstellt wurde; die zweite weist Sie auf einen Syntaxfehler in Ihrem Befehl hin. Diese Nachrichten werden an die Standardausgabeeinheit gesendet. Wenn Sie den Befehl nicht korrekt eingegeben haben, schlagen Sie im Handbuch *MQSeries MQSC-Befehle* die richtige Syntax nach.

Interaktive Eingabe von MQSC-Befehlen beenden

Geben Sie den MQSC-Befehl END ein, um die interaktive Eingabe von MQSC-Befehlen zu beenden:

```
END
```

Alternativ können Sie die Eingabe beenden, indem Sie das EOF-Zeichen <Strg Z> eingeben.

Wenn Sie eine Eingabe aus einer anderen Quelle umleiten, z. B. aus einer Textdatei, ist dies nicht erforderlich.

WS-Manager-Attribute anzeigen

Verwenden Sie zum Anzeigen der Attribute des WS-Managers, die im Befehl **runmqsc** angegeben wurden, den folgenden MQSC-Befehl:

```
DISPLAY QMGR ALL
```

Abb. 2 zeigt eine typische Ausgabe.

```
  1 : display qmgr all
AMQ8408: Details zu DISPLAY QMGR werden angezeigt.
   DESCR( )                                 DEADQ( )
   DEFXMITQ( )                              CHADEXIT( )
   CLWLEXIT( )                              CLWLDATA( )
   REPOS( )                                 REPOSNL( )
   COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)     QMNAME(saturn.queue.manager)
   CRDATE(2001-01-16)                       CRTIME(11.13.56)
   ALTDATE(2001-01-16)                      ALTTIME(11.13.56)
   QMID(saturn.queue.manager_2001-01-16_11.13.56)
   TRIGINT(999999999)                       MAXHANDS(256)
   MAXUMSGS(10000)                          AUTHOREV(DISABLED)
   INHIBTEV(DISABLED)                       LOCALEV(DISABLED)
   REMOTEEV(DISABLED)                       PERFMEV(DISABLED)
   STRSTPEV(ENABLED)                        CHAD(DISABLED)
   CHADEV(DISABLED)                         CLWLLEN(100)
   MAXMSGL(4194304)                         CCSID(819)
   MAXPRTY(9)                               CMDLEVEL(510)
   PLATFORM(OpenVMS)                        SYNCPT
   DISTL(YES)
```

Abbildung 2. Typische Ausgabe eines Befehls DISPLAY QMGR

Der Parameter ALL im Befehl DISPLAY QMGR bewirkt, dass alle Attribute des WS-Managers angezeigt werden. Da bei der Ausführung des Befehls kein WS-Manager-Name angegeben wurde, sind folgende Informationen in der Ausgabe besonders wichtig: der Name des Standard-WS-Managers (`saturn.queue.manager`), der Name der Warteschlange für nicht zustellbare Nachrichten (SYSTEM.DEAD-.LETTER.QUEUE) und der Name der Befehlswarteschlange (SYSTEM.ADMIN-.COMMAND.QUEUE).

Überprüfen Sie nun zunächst, ob diese Warteschlangen erstellt wurden, indem Sie folgenden Befehl eingeben:

```
DISPLAY QUEUE (SYSTEM.*)
```

Daraufhin wird eine Liste der Warteschlangen angezeigt, die mit dem Wortstamm 'SYSTEM.*' beginnen. Die runde Klammer muss angegeben werden.

Einen WS-Manager verwenden, der nicht der Standard-WS-Manager ist

Sie können mit dem Befehl **runmqsc** den Namen eines WS-Managers angeben, damit MQSC-Befehle nicht vom Standard-WS-Manager, sondern von einem anderen lokalen WS-Manager ausgeführt werden. Geben Sie zum Beispiel folgenden Befehl ein, wenn MQSC-Befehle von dem WS-Manager `jupiter.queue.manager` ausgeführt werden sollen:

```
runmqsc "jupiter.queue.manager"
```

Jetzt werden alle von Ihnen eingegebenen MQSC-Befehle von diesem WS-Manager verarbeitet, vorausgesetzt er befindet sich auf demselben Knoten und ist bereits aktiv.

Sie können MQSC-Befehle auch auf einem fernen WS-Manager ausführen (siehe „Ferne Ausführung von MQSC-Befehlen" auf Seite 71).

WS-Manager-Attribute ändern

Verwenden Sie zum Ändern der Attribute des WS-Managers, die im Befehl **runmqsc** angegeben wurden, den MQSC-Befehl ALTER QMGR, und geben Sie die Attribute und Werte an, die Sie ändern wollen. Geben Sie zum Beispiel folgende Befehle ein, um die Attribute von `jupiter.queue.manager` zu ändern:

```
runmqsc "jupiter.queue.manager"

ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

Mit dem Befehl ALTER QMGR wird die verwendete Warteschlange für nicht zustellbare Nachrichten geändert, und es können Sperrereignisse aktiviert werden.

MQSC-Befehle aus Textdateien ausführen

Die interaktive Ausführung von MQSC-Befehlen eignet sich gut für schnelle Tests; für sehr lange Befehle oder für Befehlsfolgen, die Sie wiederholt ausführen, sollten Sie die Eingaben jedoch über eine Textdatei bereitstellen. (Weitere Informationen zu Umleitungsanzeigern finden Sie unter „Eingabe und Ausgabe umleiten" auf Seite 37.) Erstellen Sie zu diesem Zweck zunächst in einem Texteditor eine Textdatei mit den MQSC-Befehlen. Mit dem folgenden Befehl wird zum Beispiel eine Befehlsfolge ausgeführt, die in der Textdatei `myprog.in` enthalten ist:

```
runmqsc < myprog.in
```

Auf entsprechende Weise können Sie auch die Ausgabe in eine Datei umleiten. Eine Datei, die Eingaben für MQSC-Befehle enthält, wird als *MQSC-Befehlsdatei* bezeichnet. Die Ausgabedatei mit Antworten vom WS-Manager wird als *Berichtsdatei* bezeichnet.

Um sowohl SYS$INPUT als auch SYS$OUTPUT des Befehls **runmqsc** umzuleiten, können Sie den Befehl wie folgt eingeben:

```
runmqsc < myprog.in > myprog.out
```

Dieser Befehl ruft die MQSC-Befehle in der MQSC-Befehlsdatei myprog.in. auf. Da kein WS-Manager-Name angegeben wurde, werden die MQSC-Befehle vom Standard-WS-Manager ausgeführt. Die Ausgabe wird an die Berichtsdatei myprog.out gesendet. Abb. 3 zeigt einen Auszug aus der MQSC-Befehlsdatei myprog.in und Abb. 4 auf Seite 42 den zugehörigen Auszug aus der Berichtsdatei myprog.out. Um SYS$INPUT und SYS$OUTPUT des Befehls **runmqsc** für einen WS-Manager (saturn.queue.manager) umzuleiten, bei dem es sich nicht um den Standard-WS-Manager handelt, können Sie den Befehl wie folgt eingeben:

```
runmqsc "saturn.queue.manager" < myprog.in > myprog.out
```

MQSC-Befehlsdateien

MQSC-Befehle werden in einer lesbaren Form geschrieben, d. h. in Form eines ASCII-Textes. Abb. 3 zeigt einen Auszug aus der MQSC-Befehlsdatei mit einem MQSC-Befehl (DEFINE QLOCAL) und den zugehörigen Attributen. Eine Beschreibung der einzelnen MQSC-Befehle und der jeweiligen Syntax finden Sie im Handbuch *MQSeries MQSC-Befehle*.

```
 .
 .
 .
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE  +
       DESCR(' ') +
       PUT(ENABLED) +
       DEFPRTY(0) +
       DEFPSIST(NO) +
       GET(ENABLED) +
       MAXDEPTH(5000) +
       MAXMSGL(1024) +
       DEFSOPT(SHARED) +
       NOHARDENBO +
       USAGE(NORMAL) +
       NOTRIGGER
 .
 .
 .
```

Abbildung 3. Auszug aus der MQSC-Befehlsdatei myprog.in

Aus Gründen der Portierbarkeit zwischen MQSeries-Umgebungen wird empfohlen, die Zeilenlänge in MQSC-Befehlsdateien auf 72 Zeichen zu begrenzen. Das Pluszeichen zeigt an, dass der Befehl in der nächsten Zeile fortgesetzt wird.

In MQSeries for Compaq OpenVMS dürfen Zeilen maximal 80 Zeichen lang sein, einschließlich Fortsetzungszeichen. Das Pluszeichen zeigt an, dass der Befehl in der nächsten Zeile fortgesetzt wird.

MQSC-Berichte

Der Befehl **runmqsc** gibt einen *Bericht* zurück, der an SYS$OUTPUT gesendet wird. Der Bericht enthält Folgendes:

- einen Header, der MQSC als Quelle des Berichts identifiziert:

  ```
  MQSeries-Befehle werden gestartet.
  ```

- optional eine nummerierte Liste der ausgegebenen MQSC-Befehle. Standardmäßig wird der Text der Eingabe in der Ausgabe zurückgegeben. In der Ausgabe steht vor jedem Befehl eine Folgenummer (siehe Abb. 4). Sie können die Ausgabe jedoch unterdrücken, indem Sie im Befehl **runmqsc** die Option -e angeben.
- eine Syntaxfehlernachricht für jeden fehlerhaften Befehl.
- eine *Bedienernachricht* zu jedem ausgeführten Befehl. Die Bedienernachricht zu einem erfolgreich ausgeführten Befehl DEFINE QLOCAL lautet zum Beispiel:

  ```
  AMQ8006: Die MQSeries-Warteschlange wurde erstellt.
  ```

- andere Nachrichten zu allgemeinen Fehlern während der Ausführung der Scriptdatei.
- eine kurze statistische Zusammenfassung des Berichts mit der Anzahl der gelesenen Befehle, der Anzahl der Befehle mit Syntaxfehlern und der Anzahl der Befehle, die nicht verarbeitet werden konnten.

Anmerkung: Der WS-Manager versucht nur die Befehle zu verarbeiten, die keine Syntaxfehler enthalten.

```
MQSeries-Befehle werden gestartet.
 .
 .
    12:     DEFINE QLOCAL('RED.LOCAL.QUEUE') REPLACE  +
      :            DESCR(' ') +
      :            PUT(ENABLED) +
      :            DEFPRTY(0) +
      :            DEFPSIST(NO) +
      :            GET(ENABLED) +
      :            MAXDEPTH(5000) +
      :            MAXMSGL(1024) +
      :            DEFSOPT(SHARED) +
      :            USAGE(NORMAL) +
      :            NOTRIGGER
AMQ8006: Die MQSeries-Warteschlange wurde erstellt.
      :
 .
 .
15 MQSC-Befehle gelesen.
Bei 0 Befehlen liegt ein Syntaxfehler vor.
0 Befehl(e) kann (können) nicht verarbeitet werden.
```

Abbildung 4. Auszug aus der MQSC-Berichtsdatei myprog.out

Standardmäßig verfügbare MQSC-Befehlsdateien ausführen

Bei der Installation von MQSeries for Compaq OpenVMS wird die folgende MQSC-Befehlsdatei bereitgestellt:

amqscos0.tst
Definitionen von Objekten, die von Beispielprogrammen verwendet werden.

Die Datei befindet sich im Verzeichnis `MQS_EXAMPLES:`.

Befehle mit Hilfe von runmqsc prüfen

Mit dem Befehl **runmqsc** können Sie MQSC-Befehle auf einem lokalen WS-Manager prüfen, ohne sie tatsächlich auszuführen. Geben Sie dazu im Befehl **runmqsc** die Option -v an, zum Beispiel:

```
runmqsc -v < myprog.in > myprog.out
```

Wenn Sie **runmqsc** für eine MQSC-Befehlsdatei ausführen, prüft der WS-Manager jeden Befehl und gibt einen Bericht zurück, ohne die MQSC-Befehle wirklich auszuführen. Auf diese Weise können Sie die Syntax aller Befehle in einer Befehlsdatei prüfen. Dies ist besonders wichtig, wenn Sie eine große Zahl von Befehlen aus einer Befehlsdatei ausführen.

Dieser Bericht ähnelt dem Bericht, der in Abb. 4 auf Seite 42 gezeigt wird.

Mit dieser Methode können Sie jedoch keine fernen MQSC-Befehle prüfen. Angenommen, Sie geben folgenden Befehl ein:

```
runmqsc -w 30 -v "jupiter.queue.manager" < myprog.in > myprog.out
```

In diesem Fall wird die Option -w , die angibt, dass es sich um einen fernen WS-Manager handelt, ignoriert, und der Befehl wird lokal im Prüfmodus ausgeführt.

MQSC-Probleme lösen

Wenn Ihre MQSC-Befehle nicht ausgeführt werden, überprüfen Sie anhand der folgenden Prüfliste, ob eines der dort beschriebenen allgemeinen Probleme in Ihrem Fall zutrifft. Der angezeigte Fehler gibt nicht immer einen klaren Hinweis auf das tatsächliche Problem.

Denken Sie bei der Verwendung des Befehls **runmqsc** an Folgendes:

- Verwenden Sie zum Umleiten der Eingabe aus einer Datei den Richtungsanzeiger <. Wenn Sie den Richtungsanzeiger nicht angeben, interpretiert der WS-Manager den Dateinamen als WS-Manager-Namen und gibt folgende Fehlernachricht aus:

```
AMQ8118: MQSeries-Warteschlangenmanager nicht vorhanden.
```

Probleme mit MQSC

- Wenn Sie die Ausgabe in eine Datei umleiten, verwenden Sie dazu den Richtungsanzeiger >. Standardmäßig wird die Ausgabe in das Verzeichnis gestellt, in dem Sie den Befehl **runmqsc** ausgeführt haben. Geben Sie einen vollständig qualifizierten Dateinamen an, um die Ausgabe an eine bestimmte Datei in einem bestimmten Verzeichnis zu senden.
- Überprüfen Sie, ob Sie den WS-Manager, der die Befehle ausführen soll, erstellt haben.

  Schauen Sie dazu in der Konfigurationsdatei mqs.ini nach, die sich standardmäßig im Verzeichnis MQS_ROOT:[MQM] befindet. Diese Datei enthält die Namen der WS-Manager und den Namen des Standard-WS-Managers (sofern vorhanden).
- Der WS-Manager muss bereits gestartet worden sein. Wenn nicht, starten Sie ihn (siehe „Einen WS-Manager starten" auf Seite 30). Sie erhalten eine Fehlernachricht, wenn er bereits gestartet wurde.
- Geben Sie einen WS-Manager-Namen mit dem Befehl **runmqsc** ein, wenn Sie keinen Standard-WS-Manager definiert haben, andernfalls erhalten Sie folgenden Fehler:

```
AMQ8146: MQSeries-Warteschlangenmanager nicht verfügbar.
```

  Informationen zum Korrigieren dieses Fehlers finden Sie unter „Einen vorhandenen WS-Manager zum Standard-WS-Manager erklären" auf Seite 31.
- Sie können einen MQSC-Befehl nicht als **runmqsc**-Parameter angeben. Der folgende Beispielbefehl ist daher ungültig:

```
runmqsc DEFINE QLOCAL(FRED)
```

- Sie können MQSC-Befehle erst in der DCL-Eingabeaufforderung eingeben, nachdem Sie den Befehl **runmqsc** ausgegeben haben. Beispiel:

```
DEFINE QLOCAL(QUEUE1)

%DCL-W-PARMDEL, ungültiger Parameterbegrenzer - Sonderzeichen überprüfen
```

- Sie können über **runmqsc** keine Steuerbefehle ausführen. Zum Beispiel können Sie keinen WS-Manager starten, wenn Sie MQSC-Befehle interaktiv ausführen:

```
$ runmqsc
0790997, 5724-A38 (C) Copyright IBM Corp. 1996, 2001  ALL RIGHTS RESERVED.
MQSeries-Befehle werden gestartet.


strmqm saturn.queue.manager
     1 : strmqm saturn.queue.manager
AMQ8405: Syntaxfehler in oder bei folgendem Befehlssegment:-
s

AMQ8426: Gültige MQSC-Befehle:

    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND

*CANCEL*


Einen MQSC-Befehl gelesen.
Bei einem Befehl liegt ein Syntaxfehler vor.
Alle gültigen MQSC-Befehle verarbeitet.
$
```

Siehe auch „Probleme bei der Verwendung ferner MQSC-Befehle" auf Seite 73.

Mit lokalen Warteschlangen arbeiten

Dieser Abschnitt enthält Beispiele für einige der MQSC-Befehle, die Sie verwenden können. Eine vollständige Beschreibung dieser Befehle finden Sie im Handbuch *MQSeries MQSC-Befehle*.

Eine lokale Warteschlange definieren

Für eine Anwendung ist der lokale WS-Manager derjenige WS-Manager, mit dem sie verbunden ist. Warteschlangen, die vom lokalen WS-Manager verwaltet werden, werden als lokale Warteschlangen dieses WS-Managers bezeichnet.

Verwenden Sie den MQSC-Befehl DEFINE QLOCAL zum Erstellen einer Definition einer lokalen Warteschlange und ebenso zum Erstellen der Datenstruktur, die als Warteschlange bezeichnet wird. Außerdem haben Sie die Möglichkeit, die Eigenschaften der Warteschlange, die von der standardmäßigen lokalen Warteschlange übernommen wurden, zu ändern.

Die in diesem Beispiel definierte Warteschlange, ORANGE.LOCAL.QUEUE, verfügt über die folgenden Eigenschaften:

- Sie darf GET-Aufrufe ausführen (GET aktiviert), jedoch keine PUT-Aufrufe (PUT inaktiviert); und sie arbeitet nach dem FIFO-Verfahren (First In/First Out).
- Es handelt sich um eine 'gewöhnliche' Warteschlange, d. h., es ist keine Initialisierungs- oder Übertragungswarteschlange, und sie generiert keine Auslösenachrichten.
- Die maximale Warteschlangenlänge umfasst 1000 Nachrichten; die maximale Nachrichtenlänge beträgt 2000 Bytes.

Dies wird mit folgendem MQSC-Befehl erreicht:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
       DESCR('Warteschlange für Nachrichten von anderen Systemen') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE(NORMAL)
```

Anmerkungen:

1. Bei den meisten dieser Attribute handelt es sich um die vom Produkt voreingestellten Standardattribute. Sie werden hier zur Veranschaulichung jedoch ebenfalls angezeigt. Sie müssen nur angegeben werden, wenn Sie nicht die Standardwerte verwenden wollen oder wenn die Standardwerte geändert werden müssen (siehe auch „Standardobjektattribute anzeigen" auf Seite 47).
2. USAGE (NORMAL) gibt an, dass es sich bei dieser Warteschlange um eine Übertragungswarteschlange handelt.
3. Wenn es auf demselben WS-Manager bereits eine Warteschlange mit dem Namen ORANGE.LOCAL.QUEUE gibt, führt der Befehl zu einem Fehler. Verwenden Sie das Attribut REPLACE, wenn Sie die vorhandene Definition einer Warteschlange überschreiben wollen, beachten Sie jedoch die Informationen unter „Attribute einer lokalen Warteschlange ändern" auf Seite 48.

Eine Warteschlange für nicht zustellbare Nachrichten definieren

Jeder WS-Manager sollte über eine lokale Warteschlange verfügen, die als Warteschlange für nicht zustellbare Nachrichten verwendet wird, damit Nachrichten, die nicht an ihre eigentliche Zieladresse zugestellt werden können, für spätere Abrufe gespeichert werden können. Sie müssen den WS-Manager explizit über die Warteschlange für nicht zustellbare Nachrichten informieren. Dies können Sie tun, indem Sie eine Warteschlange für nicht zustellbare Nachrichten im Befehl **crtmqm** angeben, Sie können sie aber auch später mit dem Befehl ALTER QMGR angeben. Die Warteschlange für nicht zustellbare Nachrichten muss jedoch definiert werden, bevor sie verwendet werden kann.

Das Produkt verfügt standardmäßig über eine Warteschlange für nicht zustellbare Nachrichten mit dem Namen SYSTEM.DEAD.LETTER.QUEUE. Diese Warteschlange wird automatisch erstellt, wenn Sie das Beispiel ausführen. Sie können diese Definition bei Bedarf ändern. Sie muss jedoch nicht umbenannt werden.

Für eine Warteschlange für nicht zustellbare Nachrichten bestehen keine besonderen Anforderungen, außer:

- Es muss eine lokale Warteschlange sein.
- Das Attribut MAXMSGL (maximale Nachrichtenlänge) muss so festgelegt werden, dass die Warteschlange die längsten vom WS-Manager zu bearbeitenden Nachrichten **sowie** die Größe des Headers der Warteschlange für nicht zustellbare Nachrichten (MQDLH) aufnehmen kann.

MQSeries stellt eine Steuerroutine der Warteschlange für nicht zustellbare Nachrichten bereit, über die Sie angeben können, wie Nachrichten, die in einer Warteschlange für nicht zustellbare Nachrichten gefunden werden, verarbeitet oder entfernt werden. Weitere Informationen finden Sie in „Kapitel 8. MQSeries-Steuerroutine der Warteschlange für nicht zustellbare Nachrichten" auf Seite 105.

Standardobjektattribute anzeigen

Wenn Sie ein MQSeries-Objekt definieren, werden alle Attribute, die Sie nicht angeben, vom Standardobjekt übernommen. Wenn Sie zum Beispiel eine lokale Warteschlange definieren, werden alle Attribute, die Sie in der Definition nicht angeben, von der standardmäßigen lokalen Warteschlange SYSTEM.DEFAULT.LOCAL-.QUEUE übernommen. Mit folgendem Befehl können Sie diese Attribute im Einzelnen anzeigen:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE) ALL
```

Anmerkung: Die Syntax dieses Befehls unterscheidet sich von der des entsprechenden DEFINE-Befehls.

Sie können einzelne Attribute zur Anzeige auswählen. Beispiel:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
        MAXDEPTH +
        MAXMSGL +
        CURDEPTH
```

Dieser Befehl zeigt die drei angegebenen Attribute folgendermaßen an:

```
AMQ8409: Warteschlangendetails werden angezeigt.
    QUEUE(ORANGE.LOCAL.QUEUE)
    MAXDEPTH(1000)
    MAXMSGL(2000)
    CURDEPTH(0)
```

Das Attribut CURDEPTH gibt die aktuelle Warteschlangenlänge an, d. h. die Anzahl der Nachrichten in der Warteschlange. Durch Anzeigen dieses Attributs können Sie die Warteschlangenlänge ständig überwachen, um sicherzustellen, dass immer Platz für neue Nachrichten verfügbar ist.

Eine lokale Warteschlangendefinition kopieren

Mit Hilfe des Attributs LIKE im Befehl DEFINE können Sie eine Warteschlangendefinition kopieren. Beispiel:

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

Dieser Befehl erstellt eine Warteschlange mit denselben Attributen wie die Warteschlange ORANGE.LOCAL.QUEUE, statt mit den Attributen der standardmäßigen lokalen Warteschlange.

Mit diesem Format des Befehls DEFINE können Sie auch eine Warteschlangendefinition kopieren und eine oder mehrere Änderungen an den übernommenen Attributen vornehmen. Beispiel:

```
DEFINE QLOCAL (THIRD.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE) +
       MAXMSGL(1024)
```

Dieser Befehl kopiert die Attribute der Warteschlange ORANGE.LOCAL.QUEUE in die Warteschlange THIRD.QUEUE, legt aber gleichzeitig die maximale Nachrichtenlänge auf 1024 statt auf 2000 Bytes fest.

Anmerkungen:

1. Mit dem Attribut LIKE in einem DEFINE-Befehl kopieren Sie nur die Warteschlangenattribute. Die Nachrichten in der Warteschlange werden nicht kopiert.
2. Wenn Sie eine lokale Warteschlange definieren, ohne das Attribut LIKE anzugeben, entspricht das dem Befehl DEFINE LIKE(SYSTEM.DEFAULT.LOCAL-.QUEUE).

Attribute einer lokalen Warteschlange ändern

Sie haben zwei Möglichkeiten zum Ändern von Warteschlangenattributen: den Befehl ALTER QLOCAL oder den Befehl DEFINE QLOCAL mit dem Attribut REPLACE. Die Warteschlange ORANGE.LOCAL.QUEUE wurde unter „Eine lokale Warteschlange definieren” auf Seite 45 definiert. Angenommen, Sie möchten die maximale Nachrichtenlänge für diese Warteschlange auf 10 000 Bytes erweitern.

- Der ALTER-Befehl dafür lautet:

```
ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
```

  Dieser Befehle ändert ein einzelnes Attribut, nämlich die maximale Nachrichtenlänge. Alle anderen Attribute bleiben unverändert.

- Der DEFINE-Befehl mit der Option REPLACE dafür könnte zum Beispiel lauten:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
```

Dieser Befehl ändert nicht nur die maximale Nachrichtenlänge, sondern auch alle anderen Attribute, für die die Standardwerte eingestellt werden. Die Warteschlange kann jetzt PUT-Aufrufe ausführen, was sie vorher nicht konnte. Die Zulassung von PUT-Aufrufen ist die Standardeinstellung, die durch die Warteschlange SYSTEM.DEFAULT.LOCAL.QUEUE festgelegt wurde (sofern Sie dieses Attribut dort nicht geändert haben).

Wenn Sie die maximale Nachrichtenlänge einer vorhandenen Warteschlange ***erweitern***, hat dies keine Auswirkung auf die dort enthaltenen Nachrichten. Alle neuen Nachrichten müssen jedoch dem neuen Kriterium entsprechen.

Inhalt einer lokalen Warteschlange löschen

Mit folgendem Befehl löschen Sie die Nachrichten in einer lokalen Warteschlange mit dem Namen MAGENTA.QUEUE:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

In folgenden Fällen können Sie den Inhalt einer Warteschlange nicht löschen:

- Die Warteschlange enthält nicht festgeschriebene Nachrichten, die an einem Synchronisationspunkt in die Warteschlange eingereiht wurden.
- Eine Anwendung hat die Warteschlange geöffnet.

Eine lokale Warteschlange löschen

Mit dem MQSC-Befehl DELETE QLOCAL können Sie eine lokale Warteschlange löschen. Eine Warteschlange kann nicht gelöscht werden, wenn sie Nachrichten enthält, die nicht festgeschrieben wurden. Wenn die Warteschlange jedoch eine oder mehrere festgeschriebene Nachrichten und keine nicht festgeschriebenen Nachrichten enthält, müssen Sie zum Löschen die Option PURGE angeben. Beispiel:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Indem Sie NOPURGE statt PURGE angeben, können Sie sicherstellen, dass die Warteschlange nicht gelöscht wird, wenn sie festgeschriebene Nachrichten enthält.

Warteschlangen durchsuchen

Um den Inhalt der Nachrichten in einer Warteschlange zu durchsuchen, stellt MQSeries for OpenVMS einen Beispiel-WS-Browser zur Verfügung. Der Browser liegt sowohl als Quelle als auch als ausführbares Modul vor. Die Standard-dateinamen und -pfade lauten:

Quelle

MQS_EXAMPLES:AMQSBCG0.C

Ausführbares Modul

[.BIN]AMQSBCG.EXE unter

MQS_EXAMPLES:

.

Der Befehl akzeptiert zwei Parameter:

- den Namen der Warteschlange, z. B. SYSTEM.ADMIN.RESPQ.TEST
- den Namen des WS-Managers, z. B. JJJH

In diesem Beispiel lautet der Befehl wie folgt:

```
amqsbcg "SYSTEM.ADMIN.RESPQ.TEST" "JJJH"
```

Es gibt keine Standardwerte; beide Parameter sind erforderlich. Abb. 5 auf Seite 51 zeigt ein typisches Ergebnis dieses Befehls.

```
$ amqsbcg "SYSTEM.ADMIN.RESPQ.TEST" "JJJH"

AMQSBCG0 - Start
**********************

 MQOPEN - 'SYSTEM.ADMIN.RESPQ.TEST'


 MQGET für Nachricht 1
****Nachrichtendeskriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 819
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204A4A4A48202020202020206EC8753A13200000'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'JJJH                                            '
  ** Identity Context
  UserIdentifier : 'SYSTEM      '
  AccountingToken :
   X'0536353534300000000000000000000000000000000000000000000000000006'
  ApplIdentityData : '                                    '
  ** Origin Context
  PutApplType    : '12'
  PutApplName    : 'AMQSPUT.EXE                '
  PutDate  : '20010129'    PutTime  : '19483901'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '14'

****   Nachricht   ****

 Länge - 14 Bytes

00000000:  7465 7374 206D 6573 7361 6765 2031      'Testnachricht 1  '
```

Abbildung 5. Typische Ergebnisanzeige eines WS-Browsers (Teile- 1 von 3)

```
 MQGET für Nachricht 2
****Nachrichtendeskriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 819
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204A4A4A4820202020202020206EC8753A23200000'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'JJJH                                          '
  ** Identity Context
  UserIdentifier : 'SYSTEM      '
  AccountingToken :
   X'0536353534300000000000000000000000000000000000000000000000000006'
  ApplIdentityData : '                                        '
  ** Origin Context
  PutApplType    : '12'
  PutApplName    : 'AMQSPUT.EXE                '
  PutDate  : '20010129'    PutTime  : '19484323'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '14'

****   Nachricht   ****

 Länge - 14 Bytes

00000000:  6D65 7373 6167 6520 3220 4441 5441      'Nachricht 2 Daten  '


 MQGET für Nachricht 3
****Nachrichtendeskriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 819
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204A4A4A4820202020202020206EC8753A33200000'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'JJJH                                          '
  ** Identity Context
  UserIdentifier : 'SYSTEM      '
  AccountingToken :
   X'0536353534300000000000000000000000000000000000000000000000000006'
  ApplIdentityData : '                                        '
  ** Origin Context
  PutApplType    : '12'
  PutApplName    : 'AMQSPUT.EXE                '
  PutDate  : '20010129'    PutTime  : '19491145'
  ApplOriginData : '    '
```

Abbildung 5. Typische Ergebnisanzeige eines WS-Browsers (Teile- 2 von 3)

```
GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '28'

****   Nachricht   ****

 Länge - 28 Bytes

00000000:  6D65 7373 6167 6520 3320 6461 7461 202D 'Nachricht 3 Daten -'
00000010:  2065 6E64 206F 6620 696E 666F           ' Ende der Info    '


 MQGET für Nachricht 4
****Nachrichtendeskriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 819
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204A4A4A48202020202020206EC8753A43200000'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'JJJH                                          '
  ** Identity Context
  UserIdentifier : 'SYSTEM      '
  AccountingToken :
   X'0536353534300000000000000000000000000000000000000000000000000006'
  ApplIdentityData : '                                '
  ** Origin Context
  PutApplType    : '12'
  PutApplName    : 'AMQSPUT.EXE                '
  PutDate  : '20010129'    PutTime  : '19510318'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '81'

****   Nachricht   ****

 Länge - 81 Bytes

00000000:  4A4F 484E 534F 4E2C 4441 5649 4420 4D52 'JOHNSON,DAVID MR'
00000010:  2020 2020 3239 2D4A 414E 2D32 3030 3120 '    29. JAN 2001 '
00000020:  3133 3A34 3220 3431 3233 3030 3831 2031 '13:42 41230081 1'
00000030:  3238 332E 3334 2020 3030 3235 2E32 3220 '283.34  0025.22 '
00000040:  2030 3030 302E 3030 2020 3739 3235 2E36 ' 0000.00  7925.6'
00000050:  35                                      '5              '



 Keine weiteren Nachrichten
 MQCLOSE
 MQDISC
$
```

Abbildung 5. Typische Ergebnisanzeige eines WS-Browsers (Teile- 3 von 3)

Mit Aliaswarteschlangen arbeiten

Eine Aliaswarteschlange ermöglicht das Umleiten von MQI-Aufrufen. Bei einer Aliaswarteschlange handelt es sich nicht um eine tatsächliche Warteschlange, sondern um eine Definition, deren Name in den Namen einer tatsächlichen Warteschlange aufgelöst wird. Die Definition einer Aliaswarteschlange enthält den Namen einer Zielwarteschlange, der durch das Attribut TARGQ (*BaseQName* in PCF) angegeben wird. Wenn eine Anwendung in einem MQI-Aufruf den Namen einer Aliaswarteschlange angibt, löst der WS-Manager diesen Namen zur Laufzeit in den Namen der tatsächlichen Warteschlange auf.

Es wurde zum Beispiel eine Anwendung entwickelt, die Nachrichten in eine Warteschlange mit dem Namen MY.ALIAS.QUEUE einreiht. Bei einem MQOPEN-Aufruf und einem PUT zum Einreihen einer Nachricht gibt die Anwendung den Namen dieser Warteschlange an. Der Anwendung ist nicht bekannt, dass es sich bei der Warteschlange um eine Aliaswarteschlange handelt. Bei jedem MQI-Aufruf mit diesem Aliasnamen löst der WS-Manager den Aliasnamen in den Namen der tatsächlichen Warteschlange auf, bei der es sich sowohl um eine lokale als auch um eine ferne Warteschlange dieses WS-Managers handeln kann.

Indem Sie den Wert des Attributs TARGQ ändern, können Sie MQI-Aufrufe an eine andere Warteschlange, gegebenenfalls auch die eines anderen WS-Managers, umleiten. Dies ist bei Verwaltungs- und Migrationsaufgaben sowie bei Maßnahmen zum Ausgleichen der Auslastung hilfreich.

Eine Aliaswarteschlange definieren

Der folgende Befehl erstellt eine Aliaswarteschlange:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGQ (YELLOW.QUEUE)
```

Dieser Befehl leitet MQI-Aufrufe, in denen die Warteschlange MY.ALIAS.QUEUE angegeben ist, an die Warteschlange YELLOW.QUEUE um. Der Befehl erstellt die Zielwarteschlange jedoch nicht, d. h., die MQI-Aufrufe schlagen fehl, wenn die Warteschlange YELLOW.QUEUE zur Laufzeit nicht existiert.

Sie können die MQI-Aufrufe an eine andere Warteschlange umleiten, indem Sie die Definition der Aliaswarteschlange ändern. Beispiel:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGQ (MAGENTA.QUEUE) REPLACE
```

Dieser Befehl leitet MQI-Aufrufe an eine andere Warteschlange mit dem Namen MAGENTA.QUEUE weiter.

Mit Hilfe von Aliaswarteschlangen können Sie einer einzelnen Warteschlange (der Zielwarteschlange) verschiedene Attribute für verschiedene Anwendungen zuweisen. Definieren Sie zu diesem Zweck zwei Aliasnamen, für jede Anwendung einen. Angenommen, es gibt zwei Anwendungen:

- Anwendung ALPHA kann Nachrichten in Warteschlange YELLOW.QUEUE einreihen, darf jedoch keine Nachrichten daraus abrufen.
- Anwendung BETA kann Nachrichten aus der Warteschlange YELLOW.QUEUE abrufen, darf jedoch keine Nachrichten einreihen.

Dies erreichen Sie mit Hilfe der folgenden Befehle:

```
* Dieser Aliasname ist PUT-aktiviert und GET-inaktiviert für Anwendung ALPHA

DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
       TARGQ (YELLOW.QUEUE) +
       PUT (ENABLED) +
       GET(DISABLED)

* Dieser Aliasname ist PUT-inaktiviert und GET-aktiviert für Anwendung BETA

DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
       TARGQ (YELLOW.QUEUE) +
       PUT (DISABLED) +
       GET(ENABLED)
```

ALPHA verwendet in ihren MQI-Aufrufen den Warteschlangennamen ALPHAS.ALIAS.QUEUE, und BETA verwendet den Warteschlangennamen BETAS-.ALIAS.QUEUE. Beide greifen auf dieselbe Warteschlange zu, jedoch auf unterschiedliche Weise.

Die Attribute LIKE und REPLACE können Sie beim Definieren von Aliaswarteschlangen auf dieselbe Weise verwenden wie beim Definieren von lokalen Warteschlangen.

Andere Befehle für Aliaswarteschlangen verwenden

Zum Anzeigen oder Ändern der Attribute von Aliaswarteschlangen bzw. zum Löschen des Aliaswarteschlangenobjekts können Sie die entsprechenden MQSC-Befehle verwenden. Beispiel:

```
* Attribute des Warteschlangenalias anzeigen
* ALL = Alle Attribute anzeigen

DISPLAY QUEUE (ALPHAS.ALIAS.QUEUE) ALL


* ALTER = Basiswarteschlangenname ändern, in den der Aliasname aufgelöst wird.
* FORCE = Änderung erzwingen, auch wenn Warteschlange geöffnet ist.

ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGQ(ORANGE.LOCAL.QUEUE) FORCE


* Diese Warteschlange löschen, falls möglich.

DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

Eine Aliaswarteschlange kann zum Beispiel in folgenden Fällen nicht gelöscht werden: Eine Anwendung hat die Aliaswarteschlange geöffnet, oder sie hat eine Warteschlange geöffnet, in deren Name der Name der Aliaswarteschlange aufgelöst wird. Weitere Informationen hierzu und zu anderen Befehlen für Aliaswarteschlangen finden Sie im Handbuch *MQSeries MQSC-Befehle*.

Mit Modellwarteschlangen arbeiten

Ein WS-Manager erstellt eine *dynamische Warteschlange*, wenn er von einer Anwendung einen MQI-Aufruf empfängt, in dem der Name einer Warteschlange angegeben ist, die als Modellwarteschlange definiert wurde. Der Name der neuen dynamischen Warteschlange wird beim Erstellen der Warteschlange vom WS-Manager generiert. Eine *Modellwarteschlange* ist eine Vorlage, in der die Attribute für alle dynamischen Warteschlangen, die auf ihrer Basis erstellt werden, angegeben sind.

Mit Hilfe von Modellwarteschlangen können Anwendungen bei Bedarf sehr einfach Warteschlangen erstellen.

Eine Modellwarteschlange definieren

Eine Modellwarteschlange mit bestimmten Attributen wird auf dieselbe Weise definiert wie eine lokale Warteschlange. Modellwarteschlangen und lokale Warteschlangen verfügen über dieselben Attribute, außer dass Sie für Modellwarteschlangen angeben können, ob die dynamischen Warteschlangen als temporäre oder permanente Warteschlangen erstellt werden sollen. (Permanente Warteschlangen bleiben beim Neustart von WS-Managern erhalten, temporäre Warteschlangen gehen verloren.) Beispiel:

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
       DESCR('Warteschlange für Nachrichten von Anwendung X') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE (NORMAL) +
       DEFTYPE (PERDYN)
```

Dieser Befehl erstellt eine Modellwarteschlangendefinition. Das Attribut DEFTYPE legt fest, dass die auf Basis dieser Vorlage erstellten Warteschlangen als permanente dynamische Warteschlangen erstellt werden.

Anmerkung: Die nicht angegebenen Attribute werden automatisch aus der Standardwarteschlange SYSYTEM.DEFAULT.MODEL.QUEUE kopiert.

Die Attribute LIKE und REPLACE können Sie beim Definieren von Modellwarteschlangen auf dieselbe Weise verwenden wie beim Definieren von lokalen Warteschlangen.

Andere Befehle für Modellwarteschlangen verwenden

Zum Anzeigen oder Ändern der Attribute einer Modellwarteschlange bzw. zum Löschen des Modellwarteschlangenobjekts können Sie die entsprechenden MQSC-Befehle verwenden. Beispiel:

```
* Attribute der Modellwarteschlange anzeigen
* ALL = Alle Attribute anzeigen

DISPLAY QUEUE (GREEN.MODEL.QUEUE) ALL


* ALTER = Modell ändern, um PUT-Aufrufe für alle dynamischen Warteschlangen
* zu aktivieren, die aus diesem Modell erstellt werden.

ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)


* Diese Modellwarteschlange löschen:

DELETE QMODEL (RED.MODEL.QUEUE)
```

Objekte für Auslösefunktion verwalten

MQSeries bietet eine Funktion zum automatischen Starten einer Anwendung, wenn in einer Warteschlange bestimmte Bedingungen zutreffen. Ein Beispiel für eine solche Bedingung ist, wenn die Anzahl der Nachrichten in einer Warteschlange eine bestimmte Größe erreicht. Diese Funktion, die so genannte *Auslösefunktion* (Triggering), wird im Handbuch *MQSeries Application Programming Guide* ausführlich beschrieben. In diesem Abschnitt wird beschrieben, wie die erforderlichen Objekte zur Unterstützung der Auslösefunktion in MQSeries for Compaq OpenVMS definiert werden.

Anwendungswarteschlange für die Auslösefunktion definieren

Eine Anwendungswarteschlange ist eine lokale Warteschlange, die von Anwendungen über MQI-Aufrufe zum Übertragen von Nachrichten verwendet wird. Für die Auslösefunktion müssen mehrere Warteschlangenattribute für die Anwendungswarteschlange definiert werden. Die Auslösefunktion selbst wird durch das Attribut *Trigger* (TRIGGER in MQSC) aktiviert.

Im folgenden Beispiel wird ein Auslöseereignis generiert, sobald sich in der lokalen Warteschlange MOTOR.INS.QUEUE 100 Nachrichten mit Priorität 5 oder höher befinden:

```
DEFINE QLOCAL (MOTOR.INS.QUEUE) +
    PROCESS (MOTOR.INS.PROC) +
       MAXMSGL (2000) +
       DEFPSIST (YES) +
       INITQ (MOTOR.INS.INT.Q) +
       TRIGGER +
       TRIGTYPE (DEPTH) +
       TRIGDPTH (100)+
       TRIGMPRI (5)
```

Dabei gilt:

QLOCAL (MOTOR.INS.QUEUE)
Gibt den Namen der zu definierenden Anwendungswarteschlange an.

PROCESS (MOTOR.INS.PROC)
Gibt den Namen der Anwendung an, die von einem Auslösemonitorprogramm gestartet wird.

MAXMSGL (2000)
Gibt die maximale Länge für Nachrichten in der Warteschlange an.

DEFPSIST (YES)
Gibt an, dass Nachrichten in dieser Warteschlange permanent sind.

INITQ (MOTOR.INS.INT.Q)
Dies ist der Name der Initialisierungswarteschlange, in die der WS-Manager die Auslösenachrichten einreiht.

TRIGGER
Dies ist der Wert für das Attribut 'Trigger'.

TRIGTYPE (DEPTH)
Gibt an, dass ein Auslöseereignis generiert wird, sobald die Anzahl der Nachrichten mit der in TRIMPRI angegebenen Priorität den in TRIGDPTH angegebenen Wert erreicht.

TRIGDPTH (100)
Gibt die Anzahl der Nachrichten an, die zum Generieren eines Auslöseereignisses erforderlich sind.

TRIGMPRI (5)
Dies ist die Priorität der Nachrichten, die der WS-Manager zählen muss, um zu entscheiden, ob ein Auslöseereignis generiert wird. Es werden nur Nachrichten mit Priorität 5 oder höher gezählt.

Eine Initialisierungswarteschlange definieren

Sobald ein Auslöseereignis eintritt, reiht der WS-Manager eine Auslösenachricht in die Initialisierungswarteschlange ein, die in der Definition der Anwendungswarteschlange angegeben wurde. Für Initialisierungswarteschlangen gibt es keine speziellen Attribute, als Anleitung kann jedoch die folgende Definition der lokalen Warteschlange MOTOR.INS.INT.Q dienen:

```
DEFINE QLOCAL(MOTOR.INS.INT.Q) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (10)
```

Eine Prozessdefinition erstellen

Eine Prozessdefinition wird mit dem Befehl DEFINE PROCESS erstellt. Eine Prozessdefinition ordnet einer Anwendungswarteschlange die Anwendung zu, von der Nachrichten aus der Warteschlange verarbeitet werden. Dazu wird das Attribut PROCESS für die Anwendungswarteschlange MOTOR.INS.QUEUE verwendet. Der folgende MQSC-Befehl definiert den erforderlichen Prozess, MOTOR.INS.PROC, der in diesem Beispiel verwendet wird:

```
DEFINE PROCESS (MOTOR.INS.PROC) +
    DESCR ('Nachricht mit Versicherungsantrag verarbeiten') +
               APPLTYPE (OPENVMS) +
    APPLICID ('DKA0:[MQM.ADMIN.TEST]IRMP01.EXE') +
               USERDATA ('öffnen, schließen, 235')
```

Dabei gilt:

MOTOR.INS.PROC
Dies ist der Name der Prozessdefinition (maximal 15 Zeichen).

DESCR ('Nachricht mit Versicherungsantrag verarbeiten')
Dies ist der beschreibende Text (hinter dem Schlüsselwort) für das Anwendungsprogramm, auf das sich die Defition bezieht. Dieser Text wird angezeigt, wenn Sie den Befehl DISPLAY PROCESS verwenden. Anhand des Textes können Sie die jeweilige Funktion des Prozesses erkennen. Wenn die Zeichenfolge Leerzeichen enthält, müssen Sie sie in einfache Anführungszeichen setzen.

APPLTYPE (OPENVMS)
Dies ist der Anwendungstyp, der unter OpenVMS ausgeführt wird.

APPLICID ('DKA0:[MQM.ADMIN.TEST]IRMP01.EXE')
Dies ist der Name des ausführbaren Programms der Anwendung.

USERDATA ('öffnen, schließen, 235')
Dies sind benutzerdefinierte Daten, die von der Anwendung verwendet werden können.

Prozessdefinition anzeigen

Mit dem Befehl DISPLAY PROCESS und dem Schlüsselwort ALL können Sie die Ergebnisse Ihrer Definition prüfen. Beispiel:

```
DISPLAY PROCESS (MOTOR.INS.PROC) ALL


    24 : DISPLAY PROCESS (MOTOR.INS.PROC) ALL
AMQ8407: Details zu DISPLAY PROCESS werden angezeigt.
    DESCR ('Nachricht mit Versicherungsantrag verarbeiten') +
    APPLICID ('DKA0:[MQM.ADMIN.TEST]IRMP01.EXE') +
    USERDATA (öffnen, schließen, 235) +
    PROCESS (MOTOR.INS.PROC) +
    APPLTYPE (OPENVMS)
```

Mit dem MQSC-Befehl ALTER PROCESS können Sie eine vorhandene Prozessdefinition ändern, und mit dem Befehl DELETE PROCESS können Sie eine Prozessdefinition löschen.

Kapitel 5. Verwaltungs-Tasks automatisieren

Dieses Kapitel setzt voraus, dass Sie bereits über Erfahrungen mit dem Verwalten von MQSeries-Objekten verfügen.

Irgendwann werden Sie zu der Überzeugung kommen, dass es gut für Ihre Installation ist, einige Verwaltungs- und Überwachungs-Tasks zu automatisieren. Sie können Verwaltungs-Tasks sowohl für lokale als auch ferne WS-Manager mit Hilfe von PCF-Befehlen (Programmable Command Format) automatisieren.

Dieses Kapitel beschreibt folgende Aufgaben:

- Verwenden von programmierbaren Befehlsformaten in PCF-Befehlen zum Automatisieren von Verwaltungs-Tasks
- Verwenden des Befehlsservers in „Befehlsserver für Fernverwaltung verwalten" auf Seite 63

PCF-Befehle

PCF-Befehle in MQSeries dienen dazu, Verwaltungs-Tasks innerhalb eines Verwaltungsprogramms auszuführen. Auf diese Weise können Sie aus einem Programm heraus Warteschlangen, Prozessdefinitionen und Namenslisten erstellen sowie WS-Manager ändern.

PCF-Befehle decken denselben Funktionsbereich ab, der auch von der MQSC-Funktion bereitgestellt wird.

Daher können Sie ein Programm schreiben, das PCF-Befehle von einem einzelnen Knoten aus an jeden beliebigen WS-Manager im Netz ausgibt. Auf diese Weise können Sie Verwaltungs-Tasks sowohl zentralisieren als auch automatisieren.

Jeder PCF-Befehl besteht aus einer Datenstruktur, die in den Bereich der Anwendungsdaten einer MQSeries-Nachricht eingebettet wird. Jeder einzelne Befehl wird, wie jede andere Nachricht auch, mit dem MQI-Aufruf MQPUT an den Ziel-WS-Manager gesendet. Der Befehlsserver des WS-Managers, der die Nachricht empfängt, interpretiert sie als Befehlsnachricht und führt den Befehl aus. Zum Abrufen der Antwort gibt die Anwendung einen MQGET-Aufruf aus, und die Daten werden werden in einer anderen Datenstruktur zurückgegeben. Die Anwendung kann die Antwort dann verarbeiten und entsprechend reagieren.

Anmerkung: Anders als MQSC-Befehle besitzen PCF-Befehle und deren Antworten kein lesbares Textformat.

Im Folgenden ist kurz zusammengefasst, was der Anwendungsprogrammierer beim Erstellen einer PCF-Befehlsnachricht angeben muss:

Nachrichtendeskriptor
Hierbei handelt es sich um einen standardmäßigen MQSeries-Nachrichtendeskriptor mit folgenden Angaben:
Die Nachrichtenart (*MsqType*) ist MQMT_REQUEST.
Das Nachrichtenformat (*Format*) ist MQFMT_ADMIN.

Anwendungsdaten

Sie enthalten die PCF-Nachricht einschließlich PCF-Header mit folgenden Angaben:

Als PCF-Nachrichtenart (*Type*) ist MQCFT_COMMAND angegeben.

Die Befehls-ID gibt den Befehl an, z. B. *Warteschlange ändern* (MQCMD_CHANGE_Q).

Eine vollständige Beschreibung der PCF-Datenstrukturen sowie Anweisungen zu deren Implementierung finden Sie im Handbuch *MQSeries Programmable System Management*.

Attribute in MQSC- und PCF-Befehlen

Objektattribute, die in MQSC angegeben werden, sind in diesem Buch in Großbuchstaben geschrieben (z. B. RQMNAME), obwohl bei ihnen die Groß-/Kleinschreibung nicht beachtet werden muss. Namen von MQSC-Attributen dürfen maximal acht Zeichen lang sein.

Objektattribute in PCF-Befehlen, die nicht auf acht Zeichen begrenzt sind, werden kursiv dargestellt. Beispiel: Dem MQSC-Attribut RQMNAME entspricht das PCF-Attribut *RemoteQMgrName*.

PCF-Escape-Befehle

PCF-Escape-Befehle sind PCF-Befehle, deren Nachrichtentext MQSC-Befehle enthalten. Mit PCF-Befehlen können Sie Befehle an einen fernen WS-Manager senden. Weitere Informationen zu PCF-Escape-Befehlen finden Sie im Handbuch *MQSeries Programmable System Management*.

Verwendung von PCF-Befehlen mit Hilfe von MQAI vereinfachen

MQAI (MQSeries Administration Interface) ist eine Verwaltungsschnittstelle in MQSeries, die auf der OpenVMS-Plattform verfügbar ist.

Sie führt Verwaltungs-Tasks auf einem WS-Manager mit Hilfe von *Datensäcken* aus. Mit Datensäcken können Eigenschaften (oder Parameter) von Objekten leichter bearbeitet werden als mit PCF-Befehlen.

Verwendungsmöglichkeiten der Verwaltungsschnittstelle MQAI:

- **Vereinfachen der Verwendung von PCF-Nachrichten**

  Die Verwaltungsschnittstelle MQAI erleichtert die Verwaltung von MQSeries. Sie müssen keine eigenen PCF-Nachrichten erstellen, wodurch die Probleme, die üblicherweise mit komplexen Datenstrukturen verbunden sind, vermieden werden.

  Um Parameter in Programmen, die mit MQI-Aufrufen geschrieben wurden, zu übergeben, muss die PCF-Nachricht den Befehl und Einzelangaben der Zeichenfolge bzw. ganzzahlige Daten enthalten. Zu diesem Zweck muss das Programm für jede Struktur mehrere Anweisungen enthalten, darüber hinaus muss Speicherbereich zugeordnet werden. Dies ist eine langwierige und arbeitsintensive Aufgabe.

  Dagegen übergeben Programme, die MQAI verwenden, Parameter an den entsprechenden Datensack und müssen nur eine Anweisung für jede Struktur enthalten. Durch die Verwendung von MQAI-Datensäcken entfällt die Notwendig-

keit, Feldgruppen zu bearbeiten und Speicher anzulegen; darüber hinaus wird dadurch eine gewisse Isolierung von den Einzelangaben des PCF-Befehls erreicht.

- **Implementieren sich selbst verwaltender Anwendungen und Verwaltungs-Tools**

  Der Active Directory Service in MQSeries for Windows NT and Windows 2000 Version 5.2 beispielsweise verwendet MQAI. (Zurzeit gibt es kein Beispiel für diese Verwendung auf der OpenVMS-Plattform.)

- **Einfachere Behandlung von Fehlerbedingungen**

  Es ist schwierig, Rückkehrcodes von MQSC-Befehlen zu erhalten; durch die Verwendung von MQAI ist es für das Programm jedoch einfacher, Fehlerbedingungen zu bearbeiten.

Nachdem Sie einen Datensack erstellt und mit Daten gefüllt haben, können Sie mit dem Aufruf 'mqExecute' eine Nachricht mit einem Verwaltungsbefehl an den Befehlsserver eines WS-Managers senden. Der Aufruf wartet anschließend auf Antwortnachrichten. Der Aufruf 'mqExecute' übernimmt den Datenaustausch mit dem Befehlsserver und gibt Antworten in einem Datensack zurück.

Weitere Informationen zur Verwendung der Verwaltungsschnittstelle MQAI finden Sie im Handbuch *MQSeries Administration Interface Programming Guide and Reference*.

Weitere allgemeine Informationen zu PCF-Befehlen finden Sie im Handbuch *MQSeries Programmable System Management*.

Befehlsserver für Fernverwaltung verwalten

Jedem WS-Manager kann ein Befehlsserver zugeordnet werden. Ein Befehlsserver verarbeitet alle von fernen WS-Managern eingehenden Befehle bzw. PCF-Befehle von Anwendungen. Er übergibt die Befehle zur Verarbeitung an den WS-Manager und gibt, abhängig vom Absender des Befehls, einen Beendigungscode oder eine Bedienernachricht zurück.

Ein Befehlsserver ist für alle Verwaltungsfunktionen, die PCF-Befehle oder MQAI verwenden, sowie für die Fernverwaltung zwingend erforderlich.

Anmerkung: Bei einer Fernverwaltung müssen Sie sicherstellen, dass der Ziel-WS-Manager aktiv ist. Andernfalls können die Nachrichten mit den Befehlen den WS-Manager, von dem sie ausgegeben werden, nicht verlassen. Stattdessen werden diese Nachrichten in die lokale Übertragungswarteschlange eingereiht, die den fernen WS-Manager versorgt. Diese Situation sollte unter allen Umständen vermieden werden.

Befehlsserver starten

Der Befehlsserver wird mit folgendem Befehl gestartet:

```
strmqcsv "saturn.queue.manager"
```

Dabei steht `saturn.queue.manager` für den WS-Manager, für den der Befehlsserver gestartet wird.

Status des Befehlsservers anzeigen

Bei einer Fernverwaltung müssen Sie sicherstellen, dass der Befehlsserver des Ziel-WS-Managers aktiv ist. Ist er nicht aktiv, können ferne Befehle nicht verarbeitet werden. Alle Nachrichten mit Befehlen werden in die Befehlswarteschlange des Ziel-WS-Managers eingereiht.

Der Befehl zum Anzeigen des Status des Befehlsservers eines WS-Managers, in diesem Beispiel `saturn.queue.manager`, lautet wie folgt:

```
dspmqcsv "saturn.queue.manager"
```

Diesen Befehl müssen Sie auf der Zielmaschine ausgeben. Wenn der Befehlsserver aktiv ist, wird folgende Nachricht zurückgegeben:

```
AMQ8027    Status des MQSeries-Befehlsservers:   Aktiv
```

Befehlsserver stoppen

Der Befehl zum Beenden eines Befehlsservers lautet in diesem Beispiel:

```
endmqcsv "saturn.queue.manager"
```

Sie haben zwei Möglichkeiten, den Befehlsserver zu stoppen:

- Verwenden Sie für einen gesteuerten Stopp den Befehl **endmqcsv** mit der Option -c (dies ist der Standard).
- Verwenden Sie für einen sofortigen Stopp den Befehl **endmqcsv** mit der Option -i.

Anmerkung: Beim Stoppen eines WS-Managers wird auch der ihm zugeordnete Befehlsserver gestoppt (sofern einer gestartet wurde).

Kapitel 6. Ferne MQSeries-Objekte verwalten

Dieses Kapitel beschreibt, wie MQSeries-Objekte auf einem anderen WS-Manager verwaltet werden. Außerdem wird beschrieben, wie Sie ferne Warteschlangenobjekte zum Steuern der Zieladressen von Nachrichten und Antwortnachrichten verwenden können.

Es enthält die folgenden Abschnitte:

Weitere Informationen zu Kanälen und Kanalattributen sowie zu deren Konfiguration finden Sie im Handbuch *MQSeries Intercommunication*.

Kanäle, Cluster und fernes Queuing

Ein WS-Manager kommuniziert mit einem anderen WS-Manager, indem er eine Nachricht sendet und, falls erforderlich, ein Antwort erhält. Für den empfangenden WS-Manager gibt es in Bezug auf seinen Standort folgende Möglichkeiten:
- Er befindet sich auf derselben Maschine.
- Er befindet sich auf einer anderen Maschine an demselben Standort oder am anderen Ende der Welt.
- Er wird auf derselben Plattform als lokaler WS-Manager ausgeführt.
- Er wird auf einer anderen Plattform, die von MQSeries unterstützt wird, ausgeführt.

Die Nachrichten können folgende Absender haben:
- Benutzerdefinierte Anwendungsprogramme, die Daten von einem Knoten an einen anderen übertragen.
- Benutzerdefinierte Verwaltungsanwendungen, die PCF-Befehle oder die Schnittstelle MQAI oder ADSI (Active Directory Service Interface) verwenden.
- WS-Manager, die Folgendes senden:
  - Instrumentierungsereignisnachrichten an andere WS-Manager
  - MQSC-Befehle, die von einem Befehl **runmqsc** im indirekten Modus ausgegeben werden (wobei die Befehle auf einem anderen WS-Manager ausgeführt werden).

Damit eine Nachricht an einen fernen WS-Manager gesendet werden kann, benötigt der lokale WS-Manager einen Mechanismus, mit dem er den Eingang von Nachrichten erkennen und sie transportieren kann. Dieser Mechanismus muss folgende Komponenten umfassen:
- mindestens einen Kanal
- eine Übertragungswarteschlange
- einen Nachrichtenkanalagenten (MCA)
- ein Kanalempfangsprogramm
- einen Kanalinitiator

Ein Kanal ist eine einseitige Übertragungsverbindung zwischen zwei WS-Managern, über die Nachrichten an beliebig viele Warteschlangen des fernen WS-Managers übertragen werden können.

Für jeden Ausgang des Kanals gibt es eine separate Definition. Wenn zum Beispiel an dem einen Ausgang ein Sender oder Server ist, muss am anderen Ausgang ein Empfänger oder Requester sein. Ein einfacher Kanal besteht aus einer *Definition eines Senderkanals* am Ausgang mit dem lokalen WS-Manager und einer *Definition eines Empfängerkanals* am Ausgang mit dem fernen WS-Manager. Die beiden Definitionen müssen denselben Namen haben und zusammen einen einfachen Kanal bilden.

Wenn der ferne WS-Manager auf Nachrichten, die vom lokalen WS-Manager gesendet werden, antworten soll, muss ein zweiter Kanal konfiguriert werden, über den Antworten zurück an den lokalen WS-Manager gesendet werden.

Kanäle werden mit dem MQSC-Befehl DEFINE CHANNEL definiert. Für die Beispiele in diesem Kapitel gelten die Standardkanalattribute, sofern nicht anders angegeben.

An jedem Kanalausgang gibt es einen Nachrichtenkanalagenten (MCA), der das Senden und Empfangen von Nachrichten steuert. Seine Aufgabe besteht darin, Nachrichten aus der Übertragungswarteschlange abzurufen und sie an die Kommunikationsverbindung zwischen den WS-Managern zu übergeben.

Eine Übertragungswarteschlange ist eine lokale Warteschlange mit einer besonderen Funktion, in der Nachrichten temporär aufbewahrt werden, bevor sie vom Nachrichtenkanalagenten abgerufen und an den fernen WS-Manager gesendet werden. Der Name der Übertragungswarteschlange wird in einer *Definition einer fernen Warteschlange* angegeben.

„Kanäle und Übertragungswarteschlangen für Fernverwaltung vorbereiten" auf Seite 68 zeigt, wie diese Definitionen zum Konfigurieren einer Fernverwaltung verwendet werden.

Weitere allgemeine Informationen zum Konfigurieren eines verteilten Queuing finden Sie im Handbuch *MQSeries Intercommunication*.

Fernverwaltung mit Hilfe von Clustern

In einem herkömmlichen MQSeries-Netz arbeitet bei einem verteilten Message-Queuing jeder WS-Manager unabhängig. Damit ein WS-Manager Nachrichten an einen anderen WS-Manager senden kann, muss für ihn eine Übertragungswarteschlange, ein Kanal zum fernen WS-Manager und eine Definition einer fernen Warteschlange definiert werden, und das für jede Warteschlange, an die der WS-Manager Nachrichten senden soll.

Ein *Cluster* ist eine Gruppe von WS-Managern, die so konfiguriert ist, dass die WS-Manager über ein einziges Netz direkt miteinander kommunizieren können, ohne dass komplexe Definitionen für Übertragungswarteschlangen, Kanäle und Warteschlangen benötigt werden. Cluster können einfach konfiguriert werden und bestehen in der Regel aus WS-Managern, zwischen denen logische Beziehungen bestehen und die Daten oder Anwendungen gemeinsam benutzen.

Nachdem ein Cluster erstellt wurde, können die WS-Manager miteinander kommunizieren, *ohne dass komplizierte Definitionen für Kanäle oder ferne Warteschlangen benötigt werden*. Selbst der kleinste Cluster trägt dazu bei, den Aufwand für die Systemverwaltung zu senken.

Für den Aufbau eines Netzes von WS-Managern in einem Cluster sind weniger Definitionen erforderlich als für den Aufbau einer herkömmlichen Umgebung für verteiltes Queuing. Je weniger Definitionen Sie erstellen, desto schneller und einfacher können Sie ein Netz installieren oder ändern. Darüber hinaus wird das Risiko von Fehlern in den Definitionen verringert.

Zum Konfigurieren eines Clusters benötigen Sie normalerweise pro WS-Manager eine Definition für einen Cluster-Sender (CLUSSDR) und eine Definition für einen Cluster-Empfänger (CLUSRCVR). Sie benötigen keine Definitionen für Übertragungswarteschlangen oder ferne Warteschlangen. Die Regeln der Fernverwaltung gelten auch in einem Cluster, die Definitionen selbst sind jedoch wesentlich einfacher.

Weitere Informationen zu Clustern und Cluster-Attributen sowie zu deren Konfiguration finden Sie im Handbuch *Cluster-Unterstützung in MQSeries*.

Fernverwaltung von einem lokalen WS-Manager aus mit Hilfe von MQSC-Befehlen

In diesem Abschnitt wird beschrieben, wie ein ferner WS-Manager von einem lokalen WS-Manager aus verwaltet wird. Sie können die Fernverwaltung von einem lokalen Knoten aus folgendermaßen implementieren:

- mit MQSC-Befehlen
- mit PCF-Befehlen

Die Vorbereitung der Warteschlangen und Kanäle ist in beiden Fällen nahezu dieselbe. Die Beispiele in diesem Buch enthalten MQSC-Befehle, da sie einfacher zu verstehen sind. Sie können die Beispiele jedoch bei Bedarf ohne weiteres auf PCF-Befehle übertragen. Weitere Informationen zum Erstellen von Verwaltungsprogrammen mit Hilfe von PCF-Befehlen finden Sie im Handbuch *MQSeries Programmable System Management*.

Bei der Fernverwaltung senden Sie MQSC-Befehle an einen fernen WS-Manager, entweder interaktiv oder aus einer Textdatei mit den Befehlen. Der ferne WS-Manager kann sich auf derselben Maschine oder, was typischer ist, auf einer anderen Maschine befinden. Die Fernverwaltung von WS-Managern ist in verschiedenen MQSeries-Umgebungen möglich, einschließlich AIX, AS/400, MVS/ESA und OS/2.

Zum Implementieren der Fernverwaltung müssen Sie bestimmte Objekte erstellen. Sofern Sie keine besonderen Anforderungen stellen, können die Standardwerte (z. B. für die Nachrichtenlänge) unverändert übernommen werden.

WS-Manager für Fernverwaltung vorbereiten

Abb. 6 auf Seite 68 zeigt die Konfiguration von WS-Managern und Kanälen, die für eine Fernverwaltung mit Hilfe des Befehls **runmqsc** erforderlich sind. `source.queue.manager` ist der *Quellen*-WS-Manager, von dem aus Sie MQSC-Befehle ausgeben und an den die Ergebnisse dieser Befehle (Bedienernachrichten), wenn möglich, zurückgegeben werden. `target.queue.manager` ist der Ziel-WS-Manager, der die Befehle verarbeitet und Bedienernachrichten generiert.

Anmerkung: Bei dem `source.queue.manager` ***muss*** es sich um den Standard-WS-Manager handeln. Weitere Informationen zum Erstellen eines WS-Managers finden Sie unter „crtmqm (WS-Manager erstellen)" auf Seite 260.

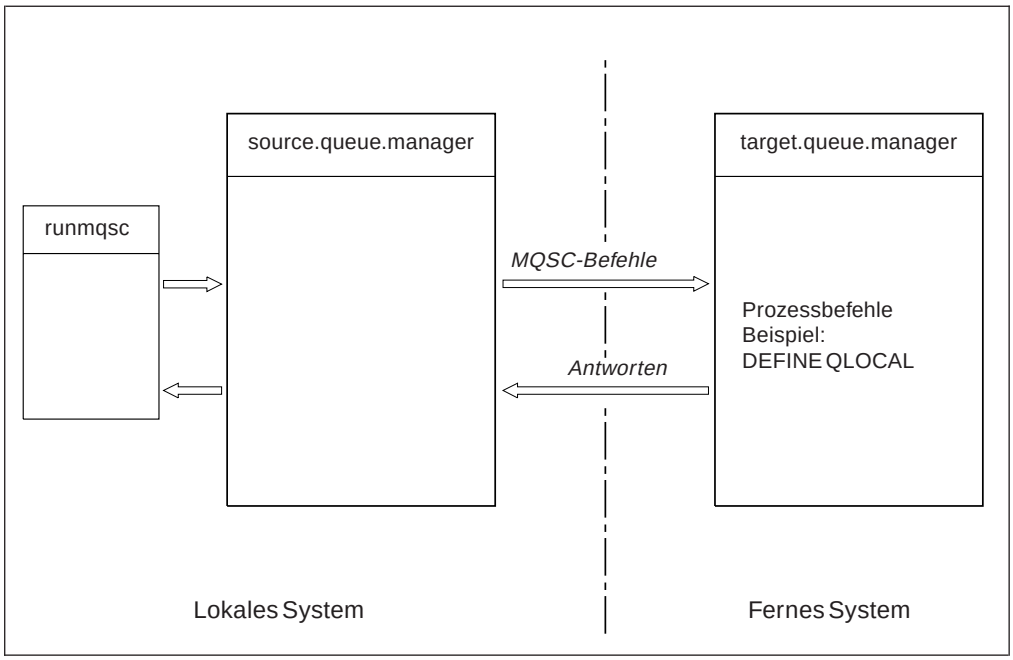


Abbildung 6. Fernverwaltung

Sofern noch nicht geschehen, müssen Sie auf beiden Systemen Folgendes tun:
- den WS-Manager mit dem Befehl **crtmqm** erstellen.
- den WS-Manager mit dem Befehl **strmqm** starten.

Diese Befehle müssen lokal oder über eine Netzfunktion, z. B. Telnet, ausgeführt werden.

Für den Ziel-WS-Manager gilt:
- Die Befehlswarteschlange SYSTEM.ADMIN.COMMAND.QUEUE muss vorhanden sein. Diese Warteschlange wird standardmäßig zusammen mit einem WS-Manager erstellt.
- Der Befehlsserver muss gestartet werden (mit dem Befehl **strmqcsv**).

Kanäle und Übertragungswarteschlangen für Fernverwaltung vorbereiten

Für die ferne Ausführung von MQSC-Befehlen müssen Sie zwei Kanäle, einen für jede Richtung, und die ihnen zugeordneten Übertragungswarteschlangen definieren. In diesem Beispiel wird angenommen, dass TCP/IP als Transportart verwendet wird und Sie die betreffende TCP/IP-Adresse kennen.

Über den Kanal `source.to.target` werden MQSC-Befehle vom Quellen-WS-Manager an die Zieladresse gesendet. Der Sender an diesem Kanal ist der WS-Manager `source.queue.manager`, und der Empfänger ist der WS-Manager `target.queue.manager`. Über den Kanal `target.to.source` werden die Ausgaben von Befehlen und alle für den Quellen-WS-Manager generierten Bedienernachrichten zurückgegeben. Außerdem müssen Sie für jeden Sender eine Übertragungswarteschlange definieren.

Dabei handelt es sich um eine lokale Warteschlange mit dem Namen des empfangenden WS-Managers. Der XMITQ-Name muss mit dem Namen des fernen WS-Managers übereinstimmen, damit die Fernverwaltung funktioniert; es sei denn, Sie verwenden einen WS-Manager-Aliasnamen. Abb. 7 zeigt diese Konfiguration im Überblick.

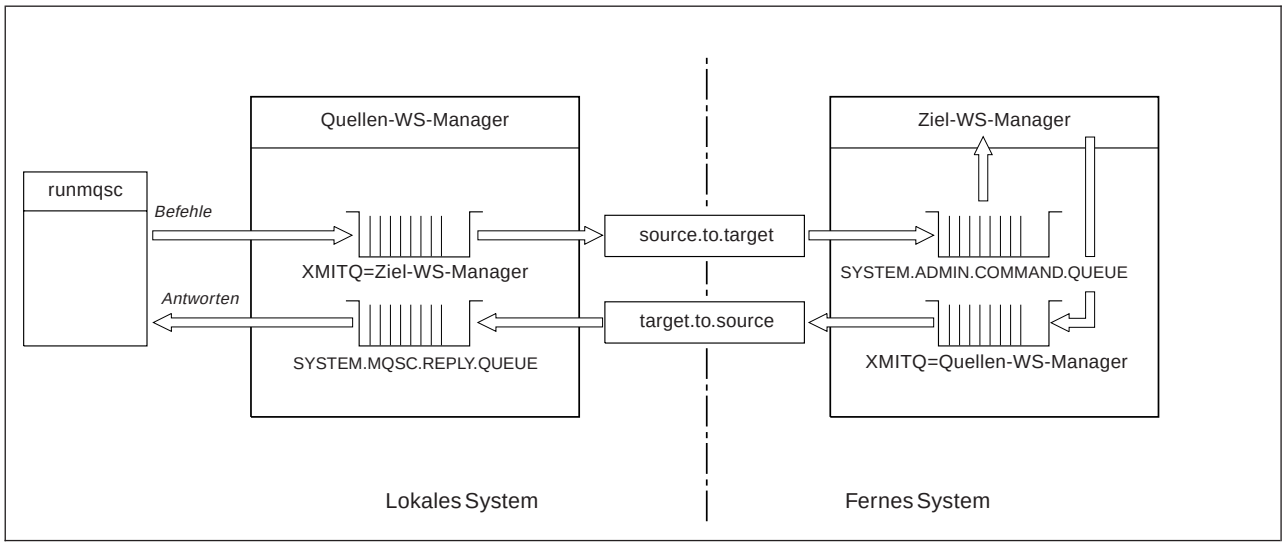


Abbildung 7. Kanäle und Warteschlangen für Fernverwaltung konfigurieren

Weitere Informationen zum Konfigurieren ferner Kanäle finden Sie im Handbuch *MQSeries Intercommunication*.

Kanäle und Übertragungswarteschlangen definieren

Geben Sie auf dem Quellen-WS-Manager die folgenden MQSC-Befehle aus, um die Kanäle und die Übertragungswarteschlange zu definieren:

```
* Sendekanal auf dem Quellen-WS-Manager definieren

DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)

* Empfangskanal auf dem Quellen-WS-Manager definieren

DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)

* Übertragungswarteschlange auf dem Quellen-WS-Manager definieren

DEFINE QLOCAL ('target.queue.manager') +
       USAGE(XMITQ)
```

Geben Sie die folgenden Befehle auf dem Ziel-WS-Manager (target.queue.manager) aus, um dort die Kanäle und die Übertragungswarteschlange zu erstellen:

```
* Sendekanal auf dem Ziel-WS-Manager definieren

DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)

* Empfangskanal auf dem Ziel-WS-Manager definieren

DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)

* Übertragungswarteschlange auf dem Ziel-WS-Manager definieren

DEFINE QLOCAL ('source.queue.manager') +
       USAGE(XMITQ)
```

Anmerkung: Die TCP/IP-Verbindungsnamen, die für das Attribut CONNAME in den Definitionen des Senderkanals angegeben sind, dienen nur der Veranschaulichung. Dabei handelt es sich um den Netznamen der Maschine am *anderen* Ende der Verbindung. Verwenden Sie die entsprechenden Werte für Ihr Netz.

Kanäle starten

In der folgenden Beschreibung wird angenommen, dass beide Ausgänge des Kanals unter MQSeries for Compaq OpenVMS ausgeführt werden. Ist dies nicht der Fall, schlagen Sie in der Dokumentation für den Ausgang des Kanals nach, der nicht unter OpenVMS ausgeführt wird.

Stellen Sie vor dem Starten der beiden Kanäle zunächst sicher, dass die TCP/IP-Services für MQSeries auf beiden Knoten konfiguriert wurden und an beiden Enden der Verbindung aktiv sind.

Starten Sie ein Empfangsprogramm am Empfängerausgang jedes Kanals.

- Geben Sie auf dem Quellen-WS-Manager folgenden Befehl ein:

```
runmqlsr -m "source.queue.manager" -t tcp
```

- Geben Sie auf dem Ziel-WS-Manager folgenden Befehl ein:

```
runmqlsr -m "target.queue.manager" -t tcp
```

Im Befehl **runmqchl** für den Quellen-WS-Manager muss der WS-Manager-Name nicht angegeben werden, weil es sich bei dem Quellen-WS-Manager um den Standard-WS-Manager handeln muss. Im Befehl **runmchl** für den Ziel-WS-Manager muss der WS-Manager-Name angegeben werden, wenn es sich bei dem Ziel-WS-Manager nicht um den Standard-WS-Manager seines Knotens handelt.

Starten Sie dann die Kanäle:

- Geben Sie auf dem Quellen-WS-Manager folgenden Befehl ein:

```
runmqchl -m "source.queue.manager" -c "source.to.target"
```

- Geben Sie auf dem Ziel-WS-Manager folgenden Befehl ein:

```
runmqchl -m "target.queue.manager" -c "target.to.source"
```

Die Befehle **runmqlsr** und **runmqchl** sind MQSeries-Steuerbefehle. Sie können nicht mit **runmqsc** ausgegeben werden. Empfangsprogramme und Kanäle können mit Hilfe von **runmqsc**-Befehlen oder -Skripts ('Kanal starten' und 'Empfangsprogramm starten') gestartet werden.

Automatische Definition von Kanälen

Ein automatische Definition von Kanälen ist nur möglich, wenn der Ziel-WS-Manager unter MQSeries Version 5.1 oder höher ausgeführt wird. Wenn eine eingehende Zuordnungsanforderung empfangen wird und in der Datei mit den Kanaldefinitionen (CDF, Channel Definition File) keine Definition für einen entsprechenden Empfänger oder eine entsprechende Serververbindung gefunden wird, erstellt MQSeries automatisch eine Definition und fügt sie der CDF hinzu. Automatische Definitionen basieren auf zwei von MQSeries bereitgestellten Standarddefinitionen: SYSTEM.AUTO.RECEIVER und SYSTEM.AUTO.SVRCONN.

Sie können die automatische Erstellung von Empfänger- und Serververbindungsdefinitionen aktivieren, indem Sie das WS-Manager-Objekt mit dem MQSC-Befehl ALTER QMGR (oder dem PCF-Befehl 'Change Queue Manager') aktualisieren.

Weitere Informationen zum automatischen Erstellen von Kanaldefinitionen finden Sie im Handbuch *MQSeries Intercommunication*.

Informationen zur automatischen Definition von Kanälen für Cluster finden Sie im Handbuch *Cluster-Unterstützung in MQSeries*.

Ferne Ausführung von MQSC-Befehlen

Der Befehlsserver ***muss*** auf dem Ziel-WS-Manager aktiv sein, wenn er eine Fernverarbeitung von MQSC-Befehlen ausführen soll. (Auf dem Quellen-WS-Manager ist dies nicht erforderlich.)

- Geben Sie auf dem Ziel-WS-Manager folgenden Befehl ein:

```
strmqcsv "target.queue.manager"
```

- Auf dem Quellen-WS-Manager können Sie dann MQSC-Befehle interaktiv im Warteschlangenmodus ausführen, indem Sie folgenden Befehl eingeben:

```
runmqsc -w 30 "target.queue.manager"
```

Bei diesem Format des Befehls **runmqsc** mit der Option -w werden die MQSC-Befehle im Warteschlangenmodus ausgeführt, d. h., Befehle werden (in einem veränderten Format) in die Eingabewarteschlange des Befehlsservers eingereiht und der Reihe nach ausgeführt.

Wenn Sie einen MQSC-Befehl eingeben, wird er an den fernen WS-Manager, in diesem Fall `target.queue.manager`, umgeleitet. Das Zeitlimit wird auf 30 Sekunden festgelegt. Wird innerhalb dieser 30 Sekunden keine Antwort empfangen, wird auf dem lokalen (Quellen-)WS-Manager die folgende Nachricht generiert:

```
AMQ8416: MQSC-Zeitüberschreitung bei Warten auf Antwort vom Befehlsserver.
```

Am Ende der MQSC-Sitzung zeigt der lokale WS-Manager alle eingegangenen Antworten mit Zeitlimitüberschreitung an. Wenn die MQSC-Sitzung beendet wird, werden alle weiteren Antworten gelöscht.

Im indirekten Modus können Sie eine MQSC-Befehlsdatei auch auf einem fernen WS-Manager ausführen. Beispiel:

```
runmqsc -w 60 "target.queue.manager" < mycomds.in > report.out
```

Dabei steht `mycomds.in` für eine Datei mit MQSC-Befehlen und `report.out` für die Berichtsdatei.

Mit WS-Managern unter MVS/ESA arbeiten

Sie können MQSC-Befehle von einem MQSeries for Compaq OpenVMS-WS-Manager an einen MVS/ESA-WS-Manager ausgeben. Dazu müssen Sie jedoch den Befehl **runmqsc** und die Kanaldefinitionen auf dem Sender ändern.

Insbesondere müssen Sie auf einem OpenVMS-Knoten dem Befehl **runmqsc** die Option -x hinzufügen:

```
runmqsc -w 30 -x "target.queue.manager"
```

Setzen Sie auf dem Senderkanal das Attribut CONVERT auf YES. Damit legen Sie fest, dass die erforderliche Datenkonvertierung zwischen den Systemen am OpenVMS-Ausgang des Kanals ausgeführt wird. Der Befehl für die Kanaldefinition lautet jetzt folgendermaßen:

```
* Sendekanal auf dem Quellen-WS-Manager unter OpenVMS definieren

DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP) +
       CONVERT(YES)
```

Auch hier gilt, dass Sie außerdem den Empfängerkanal und die Übertragungswarteschlange für den Quellen-WS-Manager definieren müssen. Es sei noch einmal erwähnt, dass in diesem Beispiel von TCP/IP als Übertragungsprotokoll ausgegangen wird.

Empfehlungen für fernes Queuing

Gehen Sie beim Implementieren des fernen Queuings folgendermaßen vor:

1. Erstellen Sie eine Befehlsdatei mit den MQSC-Befehlen, die auf dem fernen System ausgeführt werden sollen.
2. Überprüfen Sie die MQSC-Befehle lokal, indem Sie im Befehl **runmqsc** die Option -v angeben.

   Sie können mit dem Befehl **runmqsc** keine MQSC-Befehle auf einem anderen WS-Manager überprüfen.
3. Überprüfen Sie so genau wie möglich, ob die Befehlsdatei lokal fehlerfrei ausgeführt wird.
4. Führen Sie dann die Befehlsdatei für das ferne System aus.

Probleme bei der Verwendung ferner MQSC-Befehle

Wenn beim Ausführen ferner MQSC-Befehle Probleme auftreten, überprüfen Sie, ob die folgenden Punkte zutreffen:

- Der Befehlsserver auf dem Ziel-WS-Manager wurde gestartet.
- Eine gültige Übertragungswarteschlange wurde definiert.
- Beide Ausgänge der Nachrichtenkanäle wurden definiert, und zwar für beide Kanäle:
  - Den Kanal, über den die Befehle gesendet werden.
  - Den Kanal, über den die Antworten zurückgegeben werden.
- In der Kanaldefinition wurde der richtige Verbindungsname (CONNAME) angegeben.
- Die Empfangsprogramme wurden vor den Nachrichtenkanälen gestartet.
- Es wurde überprüft, dass das Zeitintervall für die Verbindungsunterbrechung nicht abgelaufen ist, z. B., wenn ein Kanal gestartet, aber nach kurzer Zeit wieder gestoppt wurde. Dies ist besonders wichtig, wenn Sie die Kanäle manuell starten.
- Es wurde sichergestellt, dass von einem Quellen-WS-Manager keine Anforderungen gesendet werden, die der Ziel-WS-Manager nicht verstehen kann (z. B. Anforderungen, die neue Parameter enthalten).

Siehe auch „MQSC-Probleme lösen” auf Seite 43.

Eine lokale Definition für eine ferne Warteschlange erstellen

Sie können die Definition einer fernen Warteschlange als lokale Definition einer fernen Warteschlange verwenden. Sie erstellen ein Objekt für eine ferne Warteschlange auf Ihrem lokalen WS-Manager, das eine lokale Warteschlange auf einem anderen WS-Manager identifiziert.

Funktionsweise von lokalen Definitionen ferner Warteschlangen

Eine Anwendung stellt eine Verbindung mit einem lokalen WS-Manager her und gibt dann einen MQOPEN-Aufruf aus. Der in diesem Aufruf angegebene Warteschlangenname ist der Name der Definition einer fernen Warteschlange auf dem lokalen WS-Manager. Die Definition der fernen Warteschlange liefert die Namen

der Zielwarteschlange, des Ziel-WS-Managers und, optional, einer Übertragungswarteschlange. Um eine Nachricht in die ferne Warteschlange einzureihen, gibt die Anwendung einen MQPUT-Aufruf aus, in dem die vom MQOPEN-Aufruf zurückgegebene Kennung angegeben ist. Der WS-Manager hängt den Namen der fernen Warteschlange und den Namen des fernen WS-Managers an einen Übertragungs-Header in der Nachricht an. Anhand dieser Informationen wird die Nachricht an ihre richtige Zieladresse im Netz weitergeleitet.

Als Administrator können Sie die Zieladresse der Nachricht steuern, indem Sie die Definition der fernen Warteschlange ändern.

Beispiel

Ziel: Eine Anwendung soll eine Nachricht in eine Warteschlange auf einem fernen WS-Manager einreihen.

Funktionsweise: Die Anwendung stellt eine Verbindung mit einem WS-Manager her, z. B. mit `saturn.queue.manager`. Die Zielwarteschlange ist einem anderen WS-Manager zugeordnet.

In ihrem MQOPEN-Aufruf gibt die Anwendung die folgenden Felder an:

Feldwert	Beschreibung
ObjectName CYAN.REMOTE.QUEUE	Gibt den lokalen Namen des Objekts für die ferne Warteschlange an. Dieses Objekt definiert die Zielwarteschlange und den Ziel-WS-Manager.
ObjectType (Warteschlange)	Identifiziert dieses Objekt als Warteschlange.
ObjectQmgrName Leerzeichen oder `saturn.queue.manager`	Dieses Feld ist optional. Ist es leer, wird der Name des lokalen WS-Managers verwendet. (Dabei handelt es sich um den WS-Manager, auf dem die Definition der fernen Warteschlange erstellt wurde und mit dem die Anwendung verbunden ist.) Ist es nicht leer, muss der Name des lokalen WS-Managers angegeben werden.

Als Nächstes gibt die Anwendung einen MQPUT-Aufruf aus, um eine Nachricht in die Warteschlange einzureihen.

Auf dem lokalen WS-Manager können Sie mit folgenden MQSC-Befehlen eine lokale Definition einer fernen Warteschlange erstellen:

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
       DESCR ('Warteschlange für Autoversicherungsanträge aus den Filialen') +
       RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
       RQMNAME ('jupiter.queue.manager') +
       XMITQ (INQUOTE.XMIT.QUEUE)
```

Dabei gilt:

QREMOTE (CYAN.REMOTE.QUEUE)
Gibt den lokalen Namen des Objekts für die ferne Warteschlange an. Diesen Namen müssen Anwendungen, die mit diesem WS-Manager verbunden sind, im MQOPEN-Aufruf angeben, um die Warteschlange AUTOMOBILE.INSURANCE.QUOTE.QUEUE auf dem fernen WS-Manager jupiter.queue.manager zu öffnen.

DESCR ('Warteschlange für Autoversicherungsanträge aus den Filialen')
Zusätzlicher Text, der die Verwendung der Warteschlange beschreibt.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Gibt den Namen der Zielwarteschlange auf dem fernen WS-Manager an. Dies ist die tatsächliche Zielwarteschlange für Nachrichten, die von den Anwendungen, die die Warteschlange CYAN.REMOTE.QUEUE angeben, gesendet werden. Die Warteschlange AUTOMOBILE.INSURANCE.QUOTE-.QUEUE muss als lokale Warteschlange auf dem fernen WS-Manager definiert werden.

RQMNAME ('jupiter.queue.manager')
Gibt den Namen des fernen WS-Managers an, dem die Zielwarteschlange AUTOMOBILE.INSURANCE.QUOTE.QUEUE zugeordnet ist. Dieser Name muss in **einfache** Anführungszeichen gesetzt werden.

XMITQ (INQUOTE.XMIT.QUEUE)
Gibt den Namen der Übertragungswarteschlange an. Diese Angabe ist optional; erfolgt keine Angabe, wird eine Warteschlange mit demselben Namen wie dem des fernen WS-Managers verwendet.

In beiden Fällen muss die jeweilige Übertragungswarteschlange als eine lokale Warteschlange mit einem Attribut *Usage* definiert werden, das angibt, dass es sich um eine Übertragungswarteschlange handelt (USAGE(XMIT) in MQSC).

Ein alternative Möglichkeit, Nachrichten in eine ferne Warteschlange einzureihen

Die Verwendung einer lokalen Definition einer fernen Warteschlange stellt nicht die einzige Möglichkeit dar, Nachrichten in eine ferne Warteschlange einzureihen. Anwendungen können den vollständigen Warteschlangennamen, der den Namen des fernen WS-Managers einschließt, als Teil des MQOPEN-Aufrufs angeben. In diesem Fall wird keine lokale Definition einer fernen Warteschlange benötigt. Diese Alternative setzt jedoch voraus, dass Anwendungen entweder den Namen des fernen WS-Managers kennen oder zur Laufzeit Zugriff darauf haben.

Andere Befehle für ferne Warteschlangen verwenden

Zum Anzeigen oder Ändern der Attribute eines Objekts für eine ferne Warteschlange bzw. zum Löschen des Objekts für eine ferne Warteschlange können Sie die entsprechenden MQSC-Befehle verwenden. Beispiel:

Lokale Definition einer fernen Warteschlange

```
* Attribute der fernen Warteschlange anzeigen
* ALL = Alle Attribute anzeigen

DISPLAY QUEUE (CYAN.REMOTE.QUEUE) ALL


* ALTER = Ferne Warteschlange ändern, um PUT-Aufrufe zu ermöglichen.
* Dies wirkt sich nicht auf die Zielwarteschlange aus, sondern
* nur auf Anwendungen, die diese ferne Warteschlange angeben.

ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)


* Diese ferne Warteschlange löschen.
* Dies wirkt sich nicht auf die Zielwarteschlange aus,
* sondern nur auf ihre lokale Definition

DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Anmerkung: Wenn Sie eine ferne Warteschlange löschen, löschen Sie nur die lokale Darstellung der fernen Warteschlange. Sie löschen nicht die ferne Warteschlange selbst und auch keine darin enthaltenen Nachrichten.

Eine Übertragungswarteschlange erstellen

Eine Übertragungswarteschlange ist eine lokale Warteschlange, die verwendet wird, wenn ein WS-Manager Nachrichten über einen Nachrichtenkanal an einen fernen WS-Manager weiterleitet. Der Kanal stellt eine einseitige Verbindung mit dem fernen WS-Manager zur Verfügung. Nachrichten werden so lange in der Übertragungswarteschlange aufbewahrt, bis der Kanal sie entgegennehmen kann. Beim Definieren eines Kanals müssen Sie den Namen einer Übertragungswarteschlange für den Sendeausgang des Nachrichtenkanals angeben.

Das Attribut *Usage* (USAGE in MQSC) definiert, ob eine Warteschlange eine Übertragungswarteschlange oder eine normale Warteschlange ist.

Standardübertragungswarteschlangen

Optional können Sie mit Hilfe des Attributs *XmitQName* (XMITQ in MQSC) eine Übertragungswarteschlange in einem Objekt für eine ferne Warteschlange angeben. Wenn keine Übertragungswarteschlange definiert ist, wird eine Standardwarteschlange verwendet. Wenn Anwendungen Nachrichten in eine ferne Warteschlange einreihen und zu diesem Zeitpunkt eine Übertragungswarteschlange mit demselben Namen wie dem der Zielwarteschlange existiert, wird diese Warteschlange verwendet. Wenn diese Warteschlange nicht existiert, wird die Warteschlange verwendet, die durch das Attribut *DefaultXmitQ* (DEFXMITQ in MQSC) auf dem lokalen WS-Manager angegeben wurde.

Der folgende MQSC-Befehl erstellt beispielsweise eine Standardübertragungswarteschlange auf dem WS-Manager `source.queue.manager` für Nachrichten, die an den WS-Manager `target.queue.manager` gerichtet sind:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Standardübertragungswarteschlange für Ziel-WS-Manager') +
       USAGE(XMITQ)
```

Anwendungen können Nachrichten direkt in eine Übertragungswarteschlange einreihen (mit einem entsprechenden Header), oder die Nachrichten werden indirekt über eine Definition einer fernen Warteschlange eingereiht. Siehe auch „Eine lokale Definition für eine ferne Warteschlange erstellen" auf Seite 73.

Definitionen ferner Warteschlangen als Aliasnamen verwenden

Sie können eine lokale Definition einer fernen Warteschlange nicht nur als Verweis auf eine Warteschlange auf einem anderen WS-Manager verwenden, sondern auch als:

- Aliasname eines WS-Managers
- Aliasname einer Warteschlange für zu beantwortende Nachrichten

Beide Arten von Aliasnamen werden über die lokale Definition einer fernen Warteschlange aufgelöst.

Wie beim fernen Queuing üblich, müssen die entsprechenden Kanäle konfiguriert werden, wenn die Nachricht ihre Zieladresse erreichen soll.

Aliasname eines WS-Managers

Ein Aliasname ist der Prozess, bei dem der Name des Ziel-WS-Managers (der in einer Nachricht angegeben ist) während der Nachrichtenweiterleitung durch einen WS-Manager geändert wird. Aliasnamen von WS-Managern sind wichtig, weil Sie mit ihrer Hilfe die Zieladresse von Nachrichten in einem Netz von WS-Managern steuern können.

Diese Steuerung erfolgt dadurch, dass Sie die Definition der fernen Warteschlange auf dem WS-Manager am Steuerungspunkt ändern. Die sendende Anwendung weiß nicht, dass es sich bei dem angegebenen WS-Manager-Namen um einen Aliasnamen handelt.

Weitere Informationen zu Aliasnamen von WS-Managern finden Sie im Handbuch *MQSeries Intercommunication*.

Aliasnamen für Warteschlangen für zu beantwortende Nachrichten

Eine Anwendung kann optional den Namen einer Warteschlange für zu beantwortende Nachrichten angeben, wenn sie eine *Anforderungsnachricht* in eine Warteschlange einreiht. Wenn die Anwendung, von der die Nachricht verarbeitet wird, den Namen der Warteschlange für zu beantwortende Nachrichten extrahiert, weiß sie, wohin die *Antwortnachricht* (falls eine erforderlich ist) gesendet werden muss.

Der Aliasname einer Warteschlange für zu beantwortende Nachrichten ist der Prozess, durch den eine Warteschlange für zu beantwortende Nachrichten (die in einer Anforderungsnachricht angegeben ist) während der Nachrichtenweiterleitung durch einen WS-Manager geändert wird. Die sendende Anwendung weiß nicht, dass es sich bei dem angegebenen Namen der Warteschlange für zu beantwortende Nachrichten um einen Aliasnamen handelt.

Mit Hilfe eines Aliasnamens einer Warteschlange für zu beantwortende Nachrichten können Sie den Namen der Warteschlange für zu beantwortende Nachrichten und optional deren WS-Manager ändern. Auf diese Weise haben Sie die Möglichkeit, den Weg für die Weiterleitung von Antwortnachrichten zu steuern.

Weitere Informationen zu Anforderungsnachrichten, Antwortnachrichten und Warteschlangen für zu beantwortende Nachrichten finden Sie im Handbuch *MQSeries Application Programming Reference*. Weitere Informationen zu Aliasnamen von Warteschlangen für zu beantwortende Nachrichten finden Sie im Handbuch *MQSeries Intercommunication*.

Datenkonvertierung

Nachrichtendaten in MQSeries-definierten Formaten (auch als integrierte Formate bekannt) können vom WS-Manager aus einem codierten Zeichensatz in einen anderen konvertiert werden, wenn beide Zeichensätze sich auf eine einzige Sprache oder auf eine Gruppe verwandter Sprachen beziehen.

Zum Beispiel wird eine Konvertierung zwischen codierten Zeichensätzen mit den Kennungen (CCSIDs) 850 und 500 unterstützt, weil beide auf westeuropäische Sprachen angewendet werden.

Informationen zur Konvertierung von EBCDIC-Zeilenvorschubzeichen nach ASCII finden Sie unter „Die Zeilengruppe AllQueueManagers" auf Seite 181.

Unterstützte Konvertierungen sind in der Codepage-Konvertierungstabelle im Handbuch *MQSeries Application Programming Reference* definiert.

Ein WS-Manager kann Nachrichten nicht in integrierte Formate konvertieren

Der WS-Manager kann Nachrichten nicht automatisch in integrierte Formate konvertieren, wenn deren CCSIDs zu verschiedenen Sprachengruppen gehören. Eine Konvertierung zwischen CCSID 850 und CCSID 1025 (ein EBCDIC-codierter Zeichensatz für Sprachen mit kyrillischen Zeichen) wird beispielsweise nicht unterstützt, weil viele Zeichen aus dem einen codierten Zeichensatz im anderen codierten Zeichensatz nicht dargestellt werden können. Wenn in Ihrem Netz WS-Manager mit verschiedenen Landessprachen aktiv sind, eine Konvertierung zwischen einigen der codierten Zeichensätze aber nicht unterstützt wird, können Sie eine Standardkonvertierung aktivieren. Eine Beschreibung der Standarddatenkonvertierung finden Sie unter „Standarddatenkonvertierung" auf Seite 79.

Datei ccsid.tbl

Die Datei ccsid.tbl enthält folgende Angaben:

- zusätzliche codierte Zeichensätze. Um zusätzliche codierte Zeichensätze anzugeben, müssen Sie die Datei ccsid.tbl editieren (siehe Anleitung in der Datei selbst).
- Standarddatenkonvertierungen

Sie können die Daten in der Datei ccsid.tbl aktualisieren. Dies kann zum Beispiel erforderlich sein, wenn ein späteres Release Ihres Betriebssystems zusätzliche codierte Zeichensätze unterstützt.

MQSeries for Compaq OpenVMS enthält eine ccsid.tbl-Beispieldatei mit dem Namen

```
MQS_EXAMPLES:CCSID.TBL
```

Die aktive Datei ccsid.tbl befindet sich im Verzeichnis

```
MQS_ROOT:[MQM.CONV.TABLE]
```

Standarddatenkonvertierung: Editieren Sie zum Implementieren der Standarddatenkonvertierung die Datei ccsid.tbl, und geben Sie eine standardmäßige EBCDIC-CCSID und eine standardmäßige ASCII-CCSID sowie die Standardeigenschaften für jede dieser CCSIDs an. Anweisungen dazu finden Sie in der Datei.

Wenn Sie die Datei ccsid.tbl aktualisieren, um die Standarddatenkonvertierung zu implementieren, müssen Sie den WS-Manager erneut starten, damit die Änderung wirksam wird.

Die Standarddatenkonvertierung läuft folgendermaßen ab:

- Wenn zwischen der Quellen- und der Ziel-CCSID keine Konvertierung unterstützt wird, es sich bei den CCSIDs der Quellen- und Zielumgebung aber um zwei EBCDIC- bzw. zwei ASCII-Kennungen handelt, werden die Zeichendaten ohne Konvertierung an die Zielanwendung übergeben.
- Handelt es sich bei einer CCSID um die Kennung eines ASCII-codierten Zeichensatzes und bei der anderen um die eines EBCDIC-codierten Zeichensatzes, konvertiert MQSeries die Daten mit Hilfe der CCSIDs für die Standarddatenkonvertierung, die in der Datei ccsid.tbl definiert wurden.

Anmerkung: Es wird empfohlen, die Konvertierung nur für die Zeichen zuzulassen, für die im codierten Zeichensatz der Nachricht und im codierten Zeichensatz für die Standardkonvertierung dieselben Codewerte stehen. Wenn Sie nur den Zeichensatz verwenden, der für MQSeries-Objektnamen gültig ist, ist diese Anforderung in der Regel erfüllt. Ausnahmen gelten für die in Japan verwendeten EBCDIC-CCSIDs 290, 930, 1279 und 5026, weil sich die Codes für Kleinbuchstaben in diesen Zeichensätzen von denen in anderen EBCDIC-CCSIDs unterscheiden.

Konvertierung von Nachrichten in benutzerdefinierte Formate

Nachrichten in benutzerdefinierten Formaten können vom WS-Manager nicht von einem codierten Zeichensatz in einen anderen konvertiert werden. Wenn Daten in einem benutzerdefinierten Format konvertiert werden müssen, müssen Sie für jedes benutzerdefinierte Format einen Datenkonvertierungs-Exit bereitstellen. Es wird nicht empfohlen, Standard-CCSIDs für die Konvertierung von Daten in benutzerdefinierten Formaten zu verwenden, obwohl dies möglich ist. Weitere Informationen zum Konvertieren von Daten in benutzerdefinierten Formaten und zum Erstellen von Datenkonvertierung-Exits finden Sie im Handbuch *MQSeries Application Programming Guide*.

CCSID des WS-Managers ändern

Es wird empfohlen, den WS-Manager nach einer Änderung der CCSID des WS-Managers mit Hilfe des Attributs CCSID im Befehl ALTER QMGR zu stoppen und erneut zu starten. Dadurch wird sichergestellt, dass alle aktiven Anwendungen, einschließlich Befehlsserver und Kanalprogramme, gestoppt und erneut gestartet werden.

Dies ist notwendig, weil alle Anwendungen, die während der Änderung der CCSID des WS-Managers aktiv sind, sonst weiter die vorhandene CCSID verwenden.

Kapitel 7. MQSeries-Objekte schützen

Dieses Kapitel beschreibt die Funktionen der Zugriffssteuerung in MQSeries for Compaq OpenVMS und wie Sie diese Steuerung implementieren können.

Es enthält die folgenden Abschnitte:

Gründe für den Schutz von MQSeries-Ressourcen

Da WS-Manager von MQSeries mit der Übertragung von Daten befasst sind, die möglicherweise einen hohen Wert darstellen, benötigen Sie den Schutz durch ein Berechtigungssystem. Dieses System stellt sicher, dass die Ressourcen, die einem WS-Manager zugeordnet sind und die er verwaltet, vor unbefugten Zugriffen geschützt sind, die den Verlust oder die unerwünschte Veröffentlichung der Daten zur Folge haben können. Für ein sicheres System ist es entscheidend, dass die folgenden Ressourcen vor einem Zugriff oder einer Änderung durch unbefugte Benutzer oder Anwendungen geschützt sind:

- Verbindungen mit einem WS-Manager
- MQSeries-Objekte wie Warteschlangen, Cluster, Kanäle und Prozesse
- Befehle für die WS-Manager-Verwaltung, einschließlich MQSC- und PCF-Befehle
- MQSeries-Nachrichten
- Kontextinformationen von Nachrichten

Erstellen Sie eine eigene Sicherheitsrichtlinie, in der Sie festlegen, welche Benutzer auf welche Ressourcen zugreifen dürfen.

Vorbereitungen

Alle WS-Manager-Ressourcen werden mit der folgenden VMS-Berechtigungs-ID ausgeführt:

```
MQM
```

Diese Berechtigungs-ID wird bei der MQSeries-Installation erstellt. Sie **müssen** dieses Ressourcenattribut allen Benutzern erteilen, die den Zugriff auf MQSeries-Ressourcen steuern.

Benutzer-IDs in MQSeries for Compaq OpenVMS mit Ressourcen-ID MQM

Wenn Ihre Benutzer-ID die OpenVMS-Berechtigungs-ID MQM einschließt, verfügen Sie über eine Gesamtberechtigung für alle MQSeries-Ressourcen. Ihre Benutzer-ID ***muss*** die OpenVMS-Berechtigungs-ID MQM einschließen, damit Sie alle Steuerbefehle von MQSeries for Compaq OpenVMS außer **crtmqcvx** verwenden können. Vor allem benötigen Sie diese Berechtigung für:

- den Befehl **runmqsc** zum Ausführen von MQSC-Befehlen,
- die Verwaltung von Berechtigungen in MQSeries for Compaq OpenVMS mit Hilfe des Befehls **setmqaut**.

Wenn Sie Kanalbefehle an WS-Manager auf einem fernen System senden, müssen Sie sicherstellen, dass Ihre Benutzer-ID die OpenVMS-Berechtigungs-ID 'MQM' für das Zielsystem einschließt. Eine Liste der PCF- und MQSC-Kanalbefehle finden Sie unter „Sicherheit für Kanalbefehle" auf Seite 91.

Bei der Installation von MQSeries wird außerdem eine ID mit dem Namen MQS_SERVER erstellt. Diese ID besitzt das Eigentumsrecht für die Ressourcendomäne, in der VMS Informationen über Sperren für MQSeries verwaltet. Zugriffsberechtigungen für diese ID werden standardmäßig an Benutzer erteilt, die:

- zu derselben Benutzergruppe wie der Benutzer MQM gehören oder
- bei denen es sich um Systembenutzer handelt oder
- die über die Berechtigungsgruppe SYSPRV, SYSLCK oder BYPASS verfügen.

Um anderen Benutzern Zugriff auf die MQ-Ressourcen zu ermöglichen, müssen Sie sicherstellen, dass die ID MQS_SERVER über die erforderliche WORLD-Berechtigung verfügt, indem Sie folgenden Befehl ausführen:

```
SET SECURITY/CLASS=RESOURCE [MQS_SERVER] /PROTECTION=(W:RWL)
```

Anmerkung: Ihre Benutzer-ID muss nicht die Berechtigungs-ID MQM einschließen, damit Sie Folgendes ausführen können:

- Ausgeben von PCF-Befehlen, einschließlich PCF-Escape-Befehlen, aus einem Verwaltungsprogramm
- Ausgeben von MQI-Aufrufen aus einem Anwendungsprogramm

Weitere Informationen

Weitere Informationen zu:

- MQSeries for Compaq OpenVMS-Befehlssätzen siehe „Kapitel 2. Eine Einführung in die MQSeries-Verwaltung" auf Seite 19,
- MQSeries for Compaq OpenVMS-Steuerbefehlen siehe „Kapitel 17. MQSeries-Steuerbefehle" auf Seite 253,
- PCF-Befehlen und PCF-Escape-Befehlen siehe Handbuch *MQSeries Programmable System Management*,
- MQI-Aufrufen siehe Handbücher *MQSeries Application Programming Guide* und *MQSeries Application Programming Reference*.

Der Objektberechtigungsmanager (OAM)

Der Zugriff auf WS-Manager-Ressourcen wird standardmäßig durch eine installierbare Komponente für die Berechtigungsverwaltung gesteuert. Bei dieser Komponente handelt es sich um den so genannten Objektberechtigungsmanager (Object Authority Manager, OAM) für MQSeries for Compaq OpenVMS. Er ist standardmäßig in MQSeries for Compaq OpenVMS enthalten und wird automatisch installiert und für jeden von Ihnen erstellten WS-Manager aktiviert, sofern Sie dies nicht anders festlegen. In diesem Kapitel wird für den im Produkt enthaltenen Objektberechtigungsmanager die Abkürzung OAM (Object Authority Manager) verwendet.

Der OAM ist eine *installierbare Komponente* des Berechtigungsservices. Dadurch, dass der OAM als installierbarer Service bereitgestellt wird, können Sie ihn flexibel einsetzen:

- Sie können den standardmäßigen OAM über die zu diesem Zweck bereitgestellte Schnittstelle durch eine eigene Berechtigungsservicekomponente ersetzen.
- Sie können die Funktionalität des standardmäßigen OAM um Funktionen aus Ihrer eigenen Berechtigungsservicekomponente erweitern (ebenfalls über die bereitgestellte Schnittstelle).
- Sie können den OAM entfernen oder löschen und ohne Berechtigungsservice arbeiten.

Weitere Informationen zu installierbaren Services finden Sie im Handbuch *MQSeries Programmable System Management*.

Der OAM verwaltet Benutzerberechtigungen zum Bearbeiten von MQSeries-Objekten einschließlich Warteschlangen, Prozessdefinitionen und Kanälen. Er stellt außerdem eine Befehlsschnittstelle zur Verfügung, über die Sie einer bestimmten Gruppe oder bestimmten Benutzern Zugriffsberechtigung für ein Objekt erteilen oder sie widerrufen können. Die Entscheidung, ob Zugriff auf eine Ressource ermöglicht wird, trifft der OAM. Der WS-Manager richtet sich nach dieser Entscheidung. Kann der OAM keine Entscheidung treffen, verhindert der WS-Manager den Zugriff auf diese Ressource.

Funktionsweise des OAM

Der OAM nutzt die Sicherheitseinrichtungen des zu Grunde liegenden Betriebssystems OpenVMS. Vor allem verwendet der OAM die Benutzer-, Gruppen- und Berechtigungs-IDs von OpenVMS. Benutzer können nur auf WS-Manager-Objekte zugreifen, wenn sie über die erforderliche Berechtigung verfügen.

Zugriff über Berechtigungs-IDs verwalten

Die Befehlsschnittstelle verwendet statt des Begriffs Benutzer-ID den Begriff *Principal*. Der Grund dafür ist, dass Berechtigungen, die einer Benutzer-ID erteilt werden, auch anderen Definitionseinheiten erteilt werden können, z. B. einem Anwendungsprogramm, das MQI-Aufrufe ausgibt, oder einem Verwaltungsprogramm, das PCF-Befehle ausgibt. In diesen Fällen entspricht der Principal, der dem Programm zugeordnet wurde, nicht notwendigerweise der Benutzer-ID, die beim Starten des Programms verwendet wurde. Im Folgenden beziehen sich die Begriffe Principal und Benutzer-ID immer auf OpenVMS-Benutzer-IDs.

Berechtigungs-IDs und die primäre Berechtigungs-ID

Die Verwaltung von Zugriffsberechtigungen für MQSeries-Ressourcen basiert auf OpenVMS-*Berechtigungs-IDs*, d. h. auf IDs, über die Principals verfügen. Ein Principal kann über eine oder mehrere OpenVMS-Berechtigungs-IDs verfügen. Eine Gruppe ist die Menge aller Principals, denen eine bestimmte Berechtigungs-ID erteilt wurde.

Der OAM verwaltet Berechtigungen auf der Ebene von Berechtigungs-IDs und nicht auf der einzelner Principals. Die Zuordnung von Principals zu ID-Namen erfolgt innerhalb des OAM, und die Ausführung von Operationen erfolgt auf der Ebene der Berechtigungs-IDs. Sie können jedoch die Berechtigungen eines einzelnen Principals anzeigen.

Ein Principal verfügt über mehrere Berechtigungs-IDs

Die Berechtigungen eines Principals ergeben sich aus der Zusammenfassung der einzelnen Berechtigungen aller Berechtigungs-IDs, über die er verfügt, d. h. aus seinen Verarbeitungsrechten. Immer wenn ein Principal Zugriff auf eine Ressource anfordert, erstellt der OAM eine Zusammenfassung seiner Berechtigungen und überprüft dann, ob er über die erforderliche Berechtigung verfügt. Mit dem Befehl **setmqaut** können Sie die Berechtigungen für einen bestimmten Principal oder eine ID festlegen.

Anmerkung: Alle Änderungen über den Befehl **setmqaut** werden sofort wirksam, es sei denn, das Objekt wird gerade verwendet. In diesem Fall wird die Änderung erst wirksam, wenn das Objekt zum nächsten Mal geöffnet wird. Änderungen an der Liste der Berechtigungs-IDs eines Principals werden jedoch erst wirksam, wenn ein WS-Manager zurückgesetzt wird, d. h., wenn er gestoppt und erneut gestartet wird.

Die Zusammenfassung der Berechtigungen eines Principals wird zwischengespeichert, nachdem sie vom OAM erstellt wurde. Änderungen der Berechtigungen einer ID, die nach dem Zwischenspeichern erfolgen, werden erst erkannt, nachdem der WS-Manager erneut gestartet wurde. Vermeiden Sie Änderungen von Berechtigungen, während der WS-Manager aktiv ist.

Standardmäßige Berechtigungs-ID

Der OAM erkennt eine Standardberechtigungsgruppe an, der alle Benutzer nominell zugeordnet werden. Diese Gruppe ist durch die Pseudo-Berechtigungs-ID 'NOBODY' definiert. 'NOBODY' kann wie eine gültige Berechtigungs-ID dazu verwendet werden, Berechtigungen mit Hilfe von MQSeries-Befehlen zuzuordnen. Diese ID verfügt standardmäßig über keine Berechtigungen. Benutzern ohne Berechtigungen kann über diese Berechtigungs-ID Zugriff auf MQSeries-Ressourcen erteilt werden.

Ressourcen, die mit dem OAM geschützt werden können

Mit dem OAM können Sie Folgendes steuern:

- Zugriff auf MQSeries-Objekte über MQI. Wenn ein Anwendungsprogramm versucht, auf ein Objekt zuzugreifen, überprüft der OAM, ob die anfordernde Benutzer-ID über die Berechtigung (auf Grund der Berechtigungs-ID) für diese Operation verfügt.

  Dies bedeutet vor allem, dass Warteschlangen und die Nachrichten in Warteschlangen vor unbefugtem Zugriff geschützt werden können.

- Berechtigung zum Ausführen von MQSC-Befehlen. Nur Principals mit der Berechtigungs-ID MQM können Verwaltungsbefehle für WS-Manager, z. B. zum Erstellen einer Warteschlange, ausführen.
- Berechtigung zum Ausführen von Steuerbefehlen. Nur Principals mit der Berechtigungs-ID MQM können Steuerbefehle ausführen, z. B. zum Erstellen eines WS-Managers, Starten eines Befehlsservers oder Verwenden des Befehls **runmqsc**.
- Berechtigung zum Ausführen von PCF-Befehlen

Verschiedenen Benutzern können unterschiedliche Arten von Zugriffsberechtigungen für dasselbe Objekt erteilt werden. Zum Beispiel kann es Benutzern mit der einen ID erlaubt werden, sowohl PUT- als auch GET-Operationen für eine bestimmte Warteschlange auszuführen, während Benutzern mit einer anderen ID nur erlaubt wird, die Warteschlange zu durchsuchen (MQGET mit Option 'browse'). Entsprechend verfügen Benutzer mit IDs vielleicht über GET- und PUT-Berechtigungen für eine Warteschlange, sie sind jedoch nicht berechtigt, die Warteschlange zu ändern oder zu löschen.

Berechtigungs-IDs für die Berechtigungsverwaltung verwenden

Durch die Verwendung von IDs anstelle einzelner Principals für die Berechtigungsverwaltung kann der Aufwand für die Berechtigungsverwaltung verringert werden. In der Regel wird eine einzelne Zugriffsart von mehreren Principals angefordert. Sie können zum Beispiel eine ID für Endbenutzer definieren, die eine bestimmte Anwendung ausführen wollen. Neuen Benutzern kann dann sehr einfach Zugriff erteilt werden, indem die betreffende ID der OpenVMS-Benutzer-ID dieser Benutzer zugeordnet wird.

Versuchen Sie, die Anzahl der IDs so niedrig wie möglich zu halten. Beginnen Sie zum Beispiel damit, dass Sie die Principals in eine Gruppe für Anwendungsbenutzer und eine andere Gruppe für Administratoren aufteilen.

Objektberechtigungsmanager inaktivieren

Der OAM ist standardmäßig aktiviert. Sie können ihn inaktivieren, indem Sie folgendermaßen den logischen Namen MQSNOAUT festlegen, bevor der WS-Manager erstellt wird:

```
$ DEFINE/SYSTEM MQSNOAUT TRUE
```

In diesem Fall ist es jedoch normalerweise nicht mehr möglich, den OAM zu einem späteren Zeitpunkt erneut zu starten. Die bessere Lösung besteht darin, den OAM nicht zu inaktivieren, sondern stattdessen sicherzustellen, dass alle Benutzer Zugriff über eine entsprechende Benutzer-ID haben.

Sie können den OAM auch nur zu Testzwecken inaktivieren, indem Sie die Zeilengruppe für den Berechtigungsservice aus der Konfigurationsdatei des WS-Managers (qm.ini) entfernen.

OAM-Befehle verwenden

Der OAM stellt eine Befehlsschnittstelle zum Erteilen und Widerrufen von Berechtigungen zur Verfügung. Um diese Befehle verwenden zu können, benötigen Sie eine entsprechende Berechtigung, d. h., Ihre Benutzer-ID muss die OpenVMS-Berechtigungs-ID MQM einschließen. Diese ID muss bei der Installation des Produkts definiert werden.

Wenn Ihre Benutzer-ID die ID MQM einschließt, verfügen Sie über die Berechtigung zum Verwalten des WS-Managers. Das heißt, Sie sind berechtigt, MQI-Aufrufe und -Befehle unter Ihrer Benutzer-ID auszugeben.

Der OAM stellt zwei Befehle zur Verfügung, die Sie in der DCL-Eingabeaufforderung von OpenVMS eingeben können, um die Berechtigungen von Benutzern zu verwalten. Dabei handelt es sich um folgende Befehle:
- **setmqaut** (Berechtigung setzen oder zurücksetzen)
- **dspmqaut** (Berechtigung anzeigen)

Eine Berechtigungsprüfung erfolgt bei folgenden Aufrufen: MQCONN, MQOPEN, MQPUT1 und MQCLOSE.

Eine Berechtigungsprüfung wird nur für das jeweils erste Exemplar dieser Aufrufe ausgeführt, und die Berechtigung wird frühestens dann geändert, wenn Sie das Objekt zurücksetzen (d. h. schließen oder öffnen).

Deshalb werden Änderungen der Berechtigung für ein Objekt mit Hilfe des Befehls **setmqaut** erst wirksam, wenn Sie das Objekt zurücksetzen.

Angaben bei der Verwendung von OAM-Befehlen

Die Berechtigungsbefehle gelten für den angegebenen WS-Manager. Wenn Sie keinen WS-Manager angeben, wird der Standard-WS-Manager verwendet. Bei diesen Befehlen müssen Sie das Objekt eindeutig angeben, d. h., Sie müssen den Objektnamen und die Objektart angeben. Außerdem müssen Sie den Principal oder ID-Namen angeben, für den die Berechtigung angewendet wird.

Berechtigungslisten

Im Befehl **setmqaut** geben Sie eine Berechtigungsliste an. Dies ist eine einfache Methode, um anzugeben, ob eine Berechtigung erteilt oder widerrufen werden soll und für welche Ressourcen die Berechtigung angewendet wird. Jede Berechtigung in der Liste wird als Schlüsselwort in Kleinbuchstaben mit einem Pluszeichen (**+**) oder Minuszeichen (**–**) als Präfix angegeben. Mit dem Pluszeichen (**+**) fügen Sie die angegebene Berechtigung hinzu, mit dem Minuszeichen (**–**) entfernen Sie die Berechtigung. Sie können in einem einzigen Befehl beliebig viele Berechtigungen angeben. Beispiel:

```
+browse -get +put
```

Den Befehl setmqaut verwenden

Sofern Sie über die erforderliche Berechtigung verfügen, können Sie mit dem Befehl **setmqaut** einem Principal oder einer Berechtigungs-ID die Berechtigung für den Zugriff auf ein bestimmtes Objekt erteilen oder die Berechtigung widerrufen. Das folgende Beispiel zeigt die Verwendung des Befehls **setmqaut**:

```
setmqaut -m "saturn.queue.manager" -t queue -n RED.LOCAL.QUEUE -g GROUPA +browse -get +put
```

In diesem Beispiel gilt Folgendes:

Term	Bedeutung
saturn.queue.manager	Gibt den WS-Manager an.
queue	Gibt die Objektart an.
RED.LOCAL.QUEUE	Gibt den Objektnamen an.
GROUPA	Gibt die ID der Gruppe an, der die Berechtigungen erteilt werden.
+browse -get +put	Gibt die Berechtigungsliste für die angegebene Warteschlange an. Zwischen dem Pluszeichen (‘+’) bzw. dem Minuszeichen (‘–’) und dem Schlüsselwort darf kein Leerzeichen stehen.

Die Berechtigungsliste enthält die Berechtigungen, die hinzugefügt oder entfernt werden, wobei Folgendes gilt:

Term	Wirkung
+browse	Fügt die Berechtigung zum Durchsuchen (MQGET mit Option ’browse’) von Nachrichten in der Warteschlange hinzu.
-get	Entfernt die Berechtigung zum Abrufen (MQGET) von Nachrichten aus der Warteschlange.
+put	Fügt die Berechtigung zum Einreihen (MQPUT) von Nachrichten in die Warteschlange hinzu.

Dies bedeutet, dass Anwendungen, die unter Benutzer-IDs gestartet werden, die die OpenVMS-Berechtigungs-ID GROUPA einschließen, über die genannten Berechtigungen verfügen.

Sie können einen oder mehrere Principals und gleichzeitig eine oder mehrere IDs angeben. Mit dem folgenden Befehl widerrufen Sie zum Beispiel die PUT-Berechtigung für die Warteschlange MyQueue für den Principal FVUSER und die IDs GROUPA und GROUPB.

```
setmqaut -m "saturn.queue.manager" -t queue -n "MyQueue" -p FVUSER -g GROUPA -g GROUPB -put
```

Anmerkung: Mit diesem Befehl widerrufen Sie außerdem die PUT-Berechtigung für alle Berechtigungs-IDs, die in der ID FVUSER eingeschlossen sind, also für alle Gruppen, denen FVUSER zugeordnet ist.

Eine formale Definition des Befehls und seiner Syntax finden Sie unter „setmqaut (Berechtigung setzen/zurücksetzen)" auf Seite 309.

Berechtigungsbefehle und installierbare Services

Der Befehl **setmqaut** verfügt über einen zusätzlichen Parameter, der den Namen der installierbaren Servicekomponente angibt, auf die die Aktualisierung angewendet wird. Sie müssen diesen Parameter angeben, wenn auf Ihrem System mehrere installierbare Komponenten gleichzeitig aktiv sind. Dies ist normalerweise nicht der Fall. Wird der Parameter nicht angegeben, wird die Aktualisierung auf den ersten installierbaren Service des betreffenden Typs (sofern vorhanden) angewendet. Dies ist standardmäßig der mitgelieferte OAM.

Zugriffsberechtigungen

Berechtigungen, die in der Berechtigungsliste des Befehls **setmqaut** definiert werden, können in folgende Kategorien eingeteilt werden:

- Berechtigungen für MQI-Aufrufe
- Berechtigungen für Verwaltungsbefehle
- Kontextberechtigungen
- allgemeine Berechtigungen, d. h. für MQI-Aufrufe, Befehle oder beides

Für jede Berechtigung gibt es ein Schlüsselwort, das in den Befehlen **setmqaut** und **dspmqaut** verwendet wird. Diese Schlüsselwörter werden unter „setmqaut (Berechtigung setzen/zurücksetzen)" auf Seite 309 beschrieben.

Berechtigungsbefehl anzeigen

Mit dem Befehl **dspmqaut** können Sie die Berechtigungen anzeigen, über die ein bestimmter Principal oder eine bestimmte ID für ein bestimmtes Objekt verfügt. Die Optionen haben dieselbe Bedeutung wie im Befehl **setmqaut**. Es kann immer nur die Berechtigung für eine ID oder einen Principal angezeigt werden. Eine formale Spezifikation dieses Befehls finden Sie unter „dspmqaut (Berechtigung anzeigen)" auf Seite 268.

Der folgende Befehl zeigt zum Beispiel die Berechtigungen an, über die die Gruppe `GpAdmin` für eine Prozessdefinition mit dem Namen `Rentenansprüche` auf dem WS-Manager `WS-Man1` verfügt.

```
dspmqaut -m "WS-Man1" -t process -n "Rentenansprüche" -g "GpAdmin"
```

Die Schlüsselwörter, die als Ergebnis dieses Befehls angezeigt werden, geben die aktiven Berechtigungen an.

OAM-Richtlinien

Einige Operationen sind besonders kritisch und sollten deshalb auf privilegierte Benutzer beschränkt sein. Beispiele:

- Starten und Stoppen von WS-Managern
- Zugreifen auf bestimmte Warteschlangen mit Sonderfunktionen, z. B. Übertragungswarteschlangen oder die Befehlwarteschlange SYSTEM.ADMIN-.COMMAND.QUEUE
- Programme, die volle MQI-Kontextoptionen verwenden
- generell das Erstellen und Kopieren von Anwendungswarteschlangen

Benutzer-IDs

Die Benutzer-ID mit Sonderfunktion (MQM), die Sie bei der Produktinstallation erstellt haben, darf nur von dem Produkt selbst verwendet werden. Sie sollte nur privilegierten Benutzern zur Verfügung gestellt werden.

Die Benutzer-ID für Berechtigungsprüfungen in Verbindung mit einem MQ-Prozess ist die OpenVMS-Benutzer-ID.

WS-Manager-Verzeichnisse

Das Verzeichnis mit den Warteschlangen und anderen WS-Manager-Daten ist ein produktinternes Verzeichnis. Objekte in diesem Verzeichnis verfügen über OpenVMS-Benutzerberechtigungen, die in Beziehung zu ihren OAM-Berechtigungen stehen. Verwenden Sie jedoch aus folgenden Gründen keine OpenVMS-Standardbefehle zum Erteilen und Widerrufen von Berechtigungen für MQI-Ressourcen:

- MQSeries-Objektnamen stimmen nicht notwendigerweise mit den entsprechenden Systemobjektnamen überein. Weitere Informationen hierzu finden Sie unter „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21.
- Alle Objekte sind Eigentum der Ressourcen-ID MQM.

Warteschlangen

Die Berechtigung für eine dynamische Warteschlange basiert auf der Modellwarteschlange, aus der sie abgeleitet wurde, ist aber nicht notwendigerweise dieselbe. Weitere Informationen finden Sie unter Hinweis 1 auf Seite 96.

Für Aliaswarteschlangen und ferne Warteschlangen gilt die Berechtigung des Objekts selbst, und nicht die Berechtigung der Warteschlange, in deren Name der Aliasname oder der Name der fernen Warteschlange aufgelöst wird. Es ist daher möglich, einer Benutzer-ID die Zugriffsberechtigung für eine Aliaswarteschlange zu erteilen, die in eine lokale Warteschlange aufgelöst wird, für die diese Benutzer-ID über keine Zugriffsberechtigungen verfügt.

Begrenzen Sie die Berechtigung zum Erstellen von Warteschlangen auf privilegierte Benutzer. Andernfalls können Benutzer die normale Zugriffssteuerung umgehen, indem Sie einfach eine Aliaswarteschlange erstellen.

Alternative Benutzerberechtigung

Über die alternative Benutzerberechtigung wird gesteuert, ob eine Benutzer-ID beim Zugriff auf ein MQSeries-Objekt die Berechtigung einer anderen Benutzer-ID verwenden kann. Dies ist besonders wichtig, wenn ein Server Anforderungen von einem Programm empfängt und sicherstellen will, dass dieses Programm über die erforderliche Berechtigung für diese Anforderung verfügt. Der Server selbst verfügt möglicherweise über die erforderliche Berechtigung, er muss jedoch wissen, ob das Programm zur Ausführung der angeforderten Aktionen berechtigt ist.

Beispiel:

- Ein Serverprogramm, das unter der Benutzer-ID PAYSERV ausgeführt wird, ruft aus einer Warteschlange eine Anforderungsnachricht ab, die von der Benutzer-ID USER1 in diese Warteschlange eingereiht wurde.
- Nachdem das Serverprogramm die Anforderungsnachricht abgerufen hat, verarbeitet es die Anforderung und reiht die Antwort in die Warteschlange für zu beantwortende Nachrichten ein, die mit der Anforderungsnachricht angegeben wurde.

- Statt die eigene Benutzer-ID (PAYSERV) zu verwenden, um seine Berechtigung zum Öffnen der Warteschlange für zu beantwortende Nachrichten nachzuweisen, kann der Server eine andere Benutzer-ID angeben, in diesem Fall USER1. In diesem Beispiel können Sie mit Hilfe der alternativen Benutzerberechtigung steuern, ob PAYSERV zum Öffnen der Warteschlange für zu beantwortende Nachrichten die Benutzer-ID USER1 als alternative Benutzer-ID angeben darf.

Die alternative Benutzer-ID wird im Feld *AlternateUserId* des Objektdeskriptors angegeben.

Anmerkung: Sie können alternative Benutzer-IDs für alle MQSeries-Objekte verwenden. Alternative Benutzer-IDs haben keine Auswirkung auf die Benutzer-ID, die von einem anderen Ressourcenmanager verwendet wird.

Kontextberechtigung

Als Kontext werden die Informationen bezeichnet, die einer einzelnen Nachricht zugeordnet sind und im Nachrichtendeskriptor (MQMD) als einem Teil der Nachricht enthalten sind. Die Kontextinformationen bestehen aus zwei Abschnitten:

Identität
Dieser Abschnitt gibt an, von wem die Nachricht gesendet wurde. Er besteht aus folgenden Feldern:
- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

Ursprung
Dieser Abschnitt gibt an, woher die Nachricht kommt und wann sie in die Warteschlange eingereiht wurde. Er besteht aus folgenden Feldern:
- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

Anwendungen können die Kontextdaten entweder bei einem MQOPEN- oder einem MQPUT-Aufruf angeben. Die Daten können von der Anwendung generiert, von einer anderen Nachricht übernommen oder standardmäßig vom WS-Manager generiert werden. Kontextdaten können zum Beispiel von Serverprogrammen verwendet werden, um die Identität des Requesters zu überprüfen, indem sie testen, ob die Nachricht von einer Anwendung kommt, die unter der ID eines berechtigten Benutzers ausgeführt wird.

Ein Serverprogramm kann über das Feld *UserIdentifier* die Benutzer-ID eines alternativen Benutzers feststellen.

Mit der Kontextberechtigung steuern Sie, ob der Benutzer eine der Kontextoptionen bei einem MQOPEN- oder MQPUT1-Aufruf angeben kann. Informationen zu Kontextoptionen finden Sie im Handbuch *MQSeries Application Programming Guide*. Eine Beschreibung der Felder des Nachrichtendeskriptors, die im Zusammenhang mit Kontextinformationen stehen, finden Sie im Handbuch *MQSeries Application Programming Reference*.

Sicherheitsüberlegungen bei fernen WS-Managern

In Bezug auf die Sicherheit bei fernen WS-Managern sollten Sie Folgendes beachten:

PUT-Berechtigung
Um Sicherheit zwischen WS-Managern herzustellen, können Sie die PUT-Berechtigung angeben, die verwendet wird, wenn ein Kanal eine Nachricht von einem anderen WS-Manager empfängt.

Geben Sie das Kanalattribut PUTAUT wie folgt an:

DEF Standard-Benutzer-ID. Dies ist die Benutzer-ID, unter der der Nachrichtenkanalagent ausgeführt wird.

CTX Die Benutzer-ID im Nachrichtenkontext.

Übertragungswarteschlangen
WS-Manager reihen Nachrichten an ferne WS-Manager automatisch in eine Übertragungswarteschlange ein; dafür ist keine spezielle Berechtigung erforderlich. Um eine Nachricht direkt in eine Übertragungswarteschlange einzureihen, wird jedoch eine spezielle Berechtigung benötigt (siehe Tabelle 2 auf Seite 93).

Kanal-Exits
Kanal-Exits können für eine zusätzliche Sicherheit verwendet werden.

Weitere Informationen finden Sie im Handbuch *MQSeries Intercommunication*.

Sicherheit für Kanalbefehle

Kanalbefehle können als PCF-Befehle ausgegeben werden, und zwar über die Schnittstelle MQAI, MQSC-Befehle und Steuerbefehle.

PCF-Befehle

Sie können PCF-Kanalbefehle ausgeben, indem Sie eine PCF-Nachricht an die Warteschlange SYSTEM.ADMIN.COMMAND.QUEUE auf einem fernen OpenVMS-System senden. Die Benutzer-ID, die im Nachrichtendeskriptor der PCF-Nachricht angegeben ist, muss die Berechtigungs-ID MQM auf dem Zielsystem einschließen. Hierbei handelt es sich um folgende Befehle:

- *ChangeChannel*
- *CopyChannel*
- *CreateChannel*
- *DeleteChannel*
- *PingChannel*
- *ResetChannel*
- *StartChannel*
- *StartChannelInitiator*
- *StartChannelListener*
- *StopChannel*
- *ResolveChannel*

Informationen zu PCF-Sicherheitsanforderungen finden Sie im Handbuch *MQSeries Programmable System Management*.

MQSC-Kanalbefehle

Sie können MQSC-Kanalbefehle an ein fernes OpenVMS-System ausgeben, indem Sie den Befehl entweder direkt in einer PCF-Escape-Nachricht senden oder ihn über **runmqsc** im indirekten Modus eingeben. Die Benutzer-ID, die im Nachrichtendeskriptor der zugeordneten PCF-Nachricht angegeben ist, muss die

Berechtigungs-ID MQM auf dem Zielsystem einschließen. (PCF-Befehle sind implizit in MQSC-Befehlen enthalten, die über **runmqsc** im indirekten Modus ausgegeben werden.) Hierbei handelt es sich um folgende Befehle:
- ALTER CHANNEL
- DEFINE CHANNEL
- DELETE CHANNEL
- PING CHANNEL
- RESET CHANNEL
- START CHANNEL
- START CHINIT
- START LISTENER
- STOP CHANNEL
- RESOLVE CHANNEL

Bei MQSC-Befehlen, die über den Befehl **runmqsc** ausgeführt werden, entspricht die Benutzer-ID in der PCF-Nachricht normalerweise der des aktuellen Benutzers.

Steuerbefehle für Kanäle

Bei Steuerbefehlen für Kanäle muss die Benutzer-ID, unter der die Befehle ausgegeben werden, die Berechtigungs-ID MQM einschließen. Hierbei handelt es sich um folgende Befehle:
- **runmqchi** (Kanalinitiator ausführen)
- **runmqchl** (Kanal ausführen)

Erläuterungen zu den Tabellen mit den Berechtigungsspezifikationen

In den Tabellen mit den Berechtigungsspezifikationen ab auf Seite 93 werden die Funktionen der Berechtigungen und die jeweiligen Einschränkungen genau definiert. Die Tabellen gelten für:
- Anwendungen, die MQI-Aufrufe ausgeben
- Verwaltungsprogramme, die MQSC-Befehle als PCF-Escape-Befehle ausgeben
- Verwaltungsprogramme, die PCF-Befehle ausgeben

Die Informationen in diesem Abschnitt sind in mehreren Tabellen zusammengefasst, die folgende Angaben enthalten:

Auszuführende Aktion
MQI-Option, MQSC-Befehl oder PCF-Befehl

Zugriffssteuerungsobjekt
Warteschlange, Prozess oder WS-Manager

Erforderliche Berechtigung
Dargestellt als eine 'MQZAO_'-Konstante

In den Tabellen entsprechen die Konstanten mit dem Präfix MQZAO_ den Schlüsselwörtern in der Berechtigungsliste des Befehls **setmqaut** für die jeweilige Definitionseinheit. Zum Beispiel entspricht MQZAO_BROWSE dem Schlüsselwort `+browse`, MQZAO_SET_ALL_CONTEXT dem Schlüsselwort `+setall` usw. Diese Konstanten sind in der Header-Datei cmqzc.h definiert, die mit dem Produkt geliefert wird. Weitere Informationen finden Sie unter „Inhalt der Berechtigungsdateien" auf Seite 101.

MQI-Berechtigungen

Eine Anwendung kann nur dann bestimmte MQI-Aufrufe und -Optionen ausgeben, wenn der Benutzer-ID, unter der sie ausgeführt wird (oder deren Berechtigungen sie übernehmen kann) die entsprechende Berechtigung erteilt wurde.

Bei vier MQI-Aufrufen sind Berechtigungsprüfungen erforderlich: MQCONN, MQOPEN, MQPUT1 und MQCLOSE.

Bei MQOPEN und MQPUT1 erfolgt die Berechtigungsprüfung für den Namen des Objekts, das geöffnet wird, und nicht für den Namen (oder die Namen), in den ein Name aufgelöst wurde. Eine Anwendung kann zum Beispiel über die Berechtigung verfügen, eine Aliaswarteschlange zu öffnen, ohne jedoch berechtigt zu sein, die Basiswarteschlange, in deren Name der Aliasname aufgelöst wird, zu öffnen. Die Regel besagt, dass die Überprüfung für die erste Definition ausgeführt wird, die bei der Namensauflösung gefunden wird und bei der es sich nicht um den Aliasnamen eines WS-Managers handelt, es sei denn, die Definition des WS-Manager-Alias wird direkt geöffnet; d. h., ihr Name steht im Feld *ObjectName* des Objektdeskriptors. Eine Berechtigung ist immer für das jeweilige Objekt erforderlich, das geöffnet wird; in einigen Fällen sind zusätzliche, warteschlangenabhängige Berechtigungen erforderlich, die durch eine Berechtigung für das WS-Manager-Objekt erlangt werden.

Tabelle 2 enthält eine Zusammenfassung der für die einzelnen Aufrufe benötigten Berechtigungen.

Tabelle 2. Erforderliche Sicherheitsberechtigungen für MQI-Aufrufe

Erforderliche Berechtigung für:	Warteschlangen-objekt (1)	Prozess-objekt	Warteschlangen-manager-objekt	Namenslisten
Option MQCONN	Nicht zutreffend	Nicht zutreffend	MQZAO_ CONNECT	Nicht zutreffend
Option MQOPEN				
MQOO_INQUIRE	MQZAO_INQUIRE (2)	MQZAO_INQUIRE (2)	MQZAO_INQUIRE (2)	MQZAO_INQUIRE (2)
MQOO_BROWSE	MQZAO_BROWSE	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_INPUT_*	MQZAO_INPUT	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_SAVE_ ALL_CONTEXT (3)	MQZAO_INPUT	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_OUTPUT (normale Warteschlange) (4)	MQZAO_OUTPUT	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_PASS_ IDENTITY_CONTEXT (5)	MQZAO_PASS_ IDENTITY_ CONTEXT	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_PASS_ ALL_CONTEXT (5, 6)	MQZAO_PASS _ALL_CONTEXT	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_SET_ IDENTITY_CONTEXT (5, 6)	MQZAO_SET_ IDENTITY_ CONTEXT	Nicht zutreffend	MQZAO_SET_ IDENTITY_ CONTEXT (7)	Nicht zutreffend
MQOO_SET_ ALL_CONTEXT (5, 8)	MQZAO_SET_ ALL_CONTEXT	Nicht zutreffend	MQZAO_SET_ ALL_CONTEXT (7)	Nicht zutreffend

Tabellen mit Berechtigungsspezifikationen

Tabelle 2. Erforderliche Sicherheitsberechtigungen für MQI-Aufrufe (Forts.)

Erforderliche Berechtigung für:	Warteschlangen-objekt (1)	Prozess-objekt	Warteschlangen-manager-objekt	Namenslisten
MQOO_OUTPUT (Übertragungswarteschlange) (9)	MQZAO_SET_ ALL_CONTEXT	Nicht zutreffend	MQZAO_SET_ ALL_CONTEXT (7)	Nicht zutreffend
MQOO_SET	MQZAO_SET	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQOO_ALTERNATE_ USER_AUTHORITY	(10)	(10)	MQZAO_ ALTERNATE_ USER_ AUTHORITY (10, 11)	(10)
Option MQPUT1				
MQPMO_PASS_ IDENTITY_CONTEXT	MQZAO_PASS_ IDENTITY_ CONTEXT (12)	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQPMO_PASS_ ALL_CONTEXT	MQZAO_PASS_ ALL_CONTEXT (12)	Nicht zutreffend	Keine Überprüfung	Nicht zutreffend
MQPMO_SET_ IDENTITY_CONTEXT	MQZAO_SET_ IDENTITY_ CONTEXT (12)	Nicht zutreffend	MQZAO_SET_ IDENTITY_ CONTEXT (7)	Nicht zutreffend
MQPMO_SET_ ALL_CONTEXT	MQZAO_SET_ ALL_CONTEXT (12)	Nicht zutreffend	MQZAO_SET_ ALL_CONTEXT (7)	Nicht zutreffend
(Übertragungswarteschlange) (9)	MQZAO_SET_ ALL_CONTEXT	Nicht zutreffend	MQZAO_SET_ ALL_CONTEXT (7)	Nicht zutreffend
MQPMO_ALTERNATE_ USER_AUTHORITY	(13)	Nicht zutreffend	MQZAO _ALTERNATE_ USER_ AUTHORITY (11)	Nicht zutreffend
Option MQCLOSE				
MQCO_DELETE	MQZAO_DELETE (14)	Nicht zutreffend	Nicht zutreffend	Nicht zutreffend
MQCO_DELETE_PURGE	MQZAO_DELETE (14)	Nicht zutreffend	Nicht zutreffend	Nicht zutreffend

Besondere Hinweise:

1. Beim Öffnen einer Modellwarteschlange gilt Folgendes:
   - Für die Modellwarteschlange wird die Berechtigung MQZAO_DISPLAY benötigt, und zwar zusätzlich zu allen anderen Berechtigungen (ebenfalls für die Modellwarteschlange), die möglicherweise für die angegebenen Öffnungsoptionen erforderlich sind.
   - Zum Erstellen der dynamischen Warteschlange wird die Berechtigung MQZAO_CREATE nicht benötigt.
   - Der Benutzer-ID, unter der die Modellwarteschlange geöffnet wird, werden automatisch alle warteschlangenspezifischen Berechtigungen (entspricht MQZAO_ALL) für die dynamische Warteschlange erteilt.
2. Abhängig von der Art des Objekts, das geöffnet wird, wird entweder das Warteschlangen-, Prozess-, Namenslisten- oder WS-Manager-Objekt überprüft.
3. MQOO_INPUT_* muss ebenfalls angegeben werden. Dies gilt für eine lokale, Modell- oder Aliaswarteschlange.
4. Diese Überprüfung wird für alle Ausgaben ausgeführt (Ausnahme siehe Hinweis 9).
5. MQOO_OUTPUT muss ebenfalls angegeben werden.
6. MQOO_PASS_IDENTITY_CONTEXT ist in dieser Option enthalten.
7. Diese Berechtigung ist sowohl für das WS-Manager-Objekt als auch für die betreffende Warteschlange erforderlich.
8. MQOO_PASS_IDENTITY_CONTEXT, MQOO_PASS_ALL_CONTEXT und MQOO_SET_IDENTITY_CONTEXT sind in dieser Option enthalten.
9. Diese Überprüfung wird für eine lokale oder Modellwarteschlange ausgeführt, für die das Warteschlangenattribut *Usage* auf MQUS_TRANSMISSION gesetzt ist und die direkt für eine Ausgabe geöffnet wurde. Dies gilt nicht, wenn eine ferne Warteschlange geöffnet wurde (indem entweder die Namen des fernen WS-Managers und der fernen Warteschlange oder der Name einer lokalen Definition der fernen Warteschlange angegeben wird).
10. Eine der folgenden Berechtigungen muss ebenfalls angegeben werden: MQOO_INQUIRE (für jede Objektart) oder (für Warteschlangen) MQOO_BROWSE, MQOO_INPUT_*, MQOO_OUTPUT bzw. MQOO_SET. Die ausgeführte Überprüfung entspricht der für die anderen angegebenen Optionen, wobei die bereitgestellte alternative Benutzer-ID für die Überprüfung der speziell angegebenen Objektberechtigung und die aktuelle Anwendungsberechtigung für die Überprüfung der Berechtigung MQZAO_ALTERNATE_USER_IDENTIFIER verwendet wird.

11. Diese Berechtigung lässt die Angabe einer beliebigen alternativen Benutzer-ID (*AlternateUserId*) zu.
12. Eine MQZAO_OUTPUT-Überprüfung wird ebenfalls durchgeführt, wenn das Warteschlangenattribut *Usage* für die Warteschlange nicht auf MQUS_TRANSMISSION gesetzt ist.
13. Die ausgeführte Überprüfung entspricht der für die anderen angegebenen Optionen, wobei die bereitgestellte alternative Benutzer-ID für die Überprüfung der speziell angegebenen Warteschlangenberechtigung und die aktuelle Anwendungsberechtigung für die Überprüfung der Berechtigung MQZAO_ALTERNATE_USER_IDENTIFIER verwendet wird.
14. Die Überprüfung wird nur ausgeführt, wenn die beiden folgenden Bedingungen zutreffen:
    - Eine permanente dynamische Warteschlange wird geschlossen und gelöscht.
    - Die Warteschlange wurde nicht von dem MQOPEN-Aufruf geöffnet, der die verwendete Objektkennung zurückgegeben hat.

    Andernfalls findet keine Überprüfung statt.

Allgemeine Hinweise:

1. Die Sonderberechtigung MQZAO_ALL_MQI schließt alle folgenden ein, die sich auf die Objektart beziehen:
   - MQZAO_CONNECT
   - MQZAO_INQUIRE
   - MQZAO_SET
   - MQZAO_BROWSE
   - MQZAO_INPUT
   - MQZAO_OUTPUT
   - MQZAO_PASS_IDENTITY_CONTEXT
   - MQZAO_PASS_ALL_CONTEXT
   - MQZAO_SET_IDENTITY_CONTEXT
   - MQZAO_SET_ALL_CONTEXT
   - MQZAO_ALTERNATE_USER_AUTHORITY
2. MQZAO_DELETE (siehe Hinweis 14 auf Seite 96) und MQZAO_DISPLAY sind als Verwaltungsberechtigungen klassifiziert. Sie sind daher nicht in MQZAO_ALL_MQI eingeschlossen.
3. 'Keine Überprüfung' bedeutet, dass keine Berechtigungsprüfung durchgeführt wird.
4. 'Nicht zutreffend' bedeutet, dass Berechtigungsprüfungen für diese Operation nicht relevant sind. Sie können zum Beispiel keinen MQPUT-Aufruf für ein Prozessobjekt ausgeben.

Verwaltungsberechtigungen

Benutzer mit diesen Berechtigungen können Verwaltungsbefehle ausführen. Dabei kann es sich um einen MQSC-Befehl in Form einer PCF-Escape-Nachricht oder eines PCF-Befehls handeln. Mit Hilfe dieser Methoden kann ein Programm einen Verwaltungsbefehl in Form einer Nachricht an einen WS-Manager senden, der dann im Namen des betreffenden Benutzers ausgeführt wird.

Berechtigungen für MQSC-Befehle in PCF-Escape-Befehlen

Tabelle 3 enthält eine Zusammenfassung der benötigten Berechtigungen für einzelne MQSC-Befehle, die in PCF-Escape-Befehlen enthalten sind.

Tabelle 3. MQSC-Befehle und erforderliche Sicherheitsberechtigungen

(2)Erforderliche Berechtigung für:	Warteschlangenobjekt	Prozess-objekt	Warteschlangen-manager-objekt	Namenslisten
MQSC-Befehl				
ALTER Objekt	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE
CLEAR QLOCAL	MQZAO_CLEAR	Nicht zutreffend	Nicht zutreffend	Nicht zutreffend
DEFINE Objekt NOREPLACE (3)	MQZAO_CREATE (4)	MQZAO_CREATE (4)	Nicht zutreffend	MQZAO_CREATE (4)
DEFINE Objekt REPLACE (3, 5)	MQZAO_CHANGE	MQZAO_CHANGE	Nicht zutreffend	MQZAO_CHANGE
DELETE Objekt	MQZAO_DELETE	MQZAO_DELETE	Nicht zutreffend	MQZAO_DELETE
DISPLAY Objekt	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY

Besondere Hinweise:

1. Die Benutzer-ID, unter der das Programm (z. B. **runmqsc**) ausgeführt wird, das den Befehl übergibt, muss außerdem über die Berechtigung MQZAO_CONNECT für den WS-Manager verfügen.
2. Abhängig von der Art des Objekts wird entweder das Warteschlangen-, Prozess-, Namenslisten- oder WS-Manager-Objekt überprüft.
3. Für DEFINE-Befehle wird außerdem die Berechtigung MQZAO_DISPLAY für das LIKE-Objekt (falls eins angegeben ist) oder für das entsprechende Objekt SYSTEM.DEFAULT.xxx (falls LIKE nicht angegeben ist) benötigt.
4. Die Berechtigung MQZAO_CREATE bezieht sich nicht auf ein bestimmtes Objekt oder eine bestimmte Objektart. Die Berechtigung zum Erstellen wird für alle Objekte (für einen angegebenen WS-Manager) erteilt, indem im Befehl SETMQAUT die Objektart QMGR angegeben wird.
5. Dies gilt, wenn das zu ersetzende Objekt tatsächlich bereits vorhanden ist. Wenn nicht, entspricht die Überprüfung der für 'DEFINE Objekt NOREPLACE'.

Tabellen mit Berechtigungsspezifikationen

Allgemeine Hinweise:

1. Um einen PCF-Befehl ausführen zu können, müssen Sie über die Berechtigung DISPLAY für den WS-Manager verfügen.
2. Die Berechtigung zum Ausführen eines PCF-Escape-Befehls ist von dem MQSC-Befehl abhängig, der im Text der PCF-Escape-Nachricht enthalten ist.
3. 'Nicht zutreffend' bedeutet, dass Berechtigungsprüfungen für diese Operation nicht relevant sind. Sie können zum Beispiel keinen Befehl CLEAR QLOCAL für ein WS-Manager-Objekt ausgeben.

Berechtigungen für PCF-Befehle

Tabelle 4 enthält eine Zusammenfassung der für die einzelnen PCF-Befehle benötigten Berechtigungen.

Tabelle 4. PCF-Befehle und erforderliche Sicherheitsberechtigungen

(2) Erforderliche Berechtigung für:	Warteschlangenobjekt	Prozessobjekt	Warteschlangen-manager-objekt	Namenslisten
PCF-Befehl				
Change object	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE	MQZAO_CHANGE
Clear Queue	MQZAO_CLEAR	Nicht zutreffend	Nicht zutreffend	Nicht zutreffend
Copy object (ohne 'replace') (3)	MQZAO_CREATE (4)	MQZAO_CREATE (4)	Nicht zutreffend	MQZAO_CREATE (4)
Copy object (mit 'replace') (3, 6)	MQZAO_CHANGE	MQZAO_CHANGE	Nicht zutreffend	MQZAO_CHANGE
Create object (ohne 'replace') (5)	MQZAO_CREATE (4)	MQZAO_CREATE (4)	Nicht zutreffend	MQZAO_CREATE (4)
Create object (mit 'replace') (5, 6)	MQZAO_CHANGE	MQZAO_CHANGE	Nicht zutreffend	MQZAO_CHANGE
Delete object	MQZAO_DELETE	MQZAO_DELETE	Nicht zutreffend	MQZAO_DELETE
Inquire object	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY	MQZAO_DISPLAY
Inquire object names	Keine Überprüfung	Keine Überprüfung	Keine Überprüfung	Keine Überprüfung
Reset queue statistics	MQZAO_DISPLAY und MQZAO_CHANGE	Nicht zutreffend	Nicht zutreffend	Nicht zutreffend

Besondere Hinweise:

1. Die Benutzer-ID, unter der das Programm, das den Befehl übergibt, ausgeführt wird, muss außerdem berechtigt sein, eine Verbindung mit seinem lokalen WS-Manager herzustellen und die Admin-Befehlswarteschlange für Ausgaben zu öffnen.
2. Abhängig von der Art des Objekts wird entweder das Warteschlangen-, Prozess-, Namenslisten- oder WS-Manager-Objekt überprüft.
3. Für Copy-Befehle wird außerdem die Berechtigung MQZAO_DISPLAY für das From-Objekt benötigt.
4. Die Berechtigung MQZAO_CREATE bezieht sich nicht auf ein bestimmtes Objekt oder eine bestimmte Objektart. Die Berechtigung zum Erstellen wird für alle Objekte (für einen angegebenen WS-Manager) erteilt, indem im Befehl SETMQAUT die Objektart QMGR angegeben wird.
5. Für Create-Befehle wird außerdem die Berechtigung MQZAO_DISPLAY für das entsprechende SYSTEM.DEFAULT.*-Objekt benötigt.
6. Dies gilt, wenn das zu ersetzende Objekt bereits vorhanden ist. Wenn nicht, entspricht die Überprüfung der für 'Copy' oder 'Create' ohne 'replace'.

Allgemeine Hinweise:

1. Um einen PCF-Befehl ausführen zu können, müssen Sie über die Berechtigung DISPLAY für den WS-Manager verfügen.
2. Die Sonderberechtigung MQZAO_ALL_ADMIN schließt alle folgenden ein, die sich auf die Objektart beziehen:
   - MQZAO_CHANGE
   - MQZAO_CLEAR
   - MQZAO_DELETE
   - MQZAO_DISPLAY

   MQZAO_CREATE ist nicht eingeschlossen, weil diese Berechtigung sich nicht speziell auf ein bestimmtes Objekt oder eine bestimmte Objektart bezieht.
3. 'Keine Überprüfung' bedeutet, dass keine Berechtigungsprüfung durchgeführt wird.
4. 'Nicht zutreffend' bedeutet, dass Berechtigungsprüfungen für diese Operation nicht relevant sind. Sie können zum Beispiel den Befehl **Clear Queue** nicht für ein Prozessobjekt verwenden.

Erläuterung der Berechtigungsdateien

Anmerkung: Die Informationen in diesem Abschnitt können für die Fehlerbestimmung herangezogen werden. Verwenden Sie unter normalen Umständen Berechtigungsbefehle zum Anzeigen und Ändern von Berechtigungsinformationen.

MQSeries for Compaq OpenVMS verwendet eine besondere Dateistruktur zum Implementieren der Sicherheit. Das Einzige, was Sie in Bezug auf diese Dateien tun müssen, besteht darin sicherzustellen, dass alle Berechtigungsdateien selbst gesichert sind.

Sicherheit wird durch Berechtigungsdateien implementiert. Unter Sicherheitsaspekten betrachtet, gibt es drei Arten von Berechtigungen:

- Berechtigungen für einzelne Objekte, z. B. die Berechtigung zum Einreihen einer Nachricht in eine Warteschlange.
- Berechtigungen für einzelne Objektklassen, z. B. die Berechtigung zum Erstellen einer Warteschlange.
- Berechtigungen für alle Objektklassen, z. B. die Berechtigung zum Ausführen von Operationen im Namen verschiedener Benutzer.

Pfade für Berechtigungsdateien

Der Pfad für eine Berechtigungsdatei ist von ihrem Typ abhängig. Wenn Sie zum Beispiel eine Berechtigung für ein Objekt angeben, erstellt der WS-Manager die entsprechenden Berechtigungsdateien. Er stellt diese Dateien in ein Unterverzeichnis, dessen Pfad durch den Namen des WS-Managers, die Art der Berechtigung und, sofern hilfreich, den Objektnamen definiert wird.

Nicht alle Berechtigungen gelten direkt für Objektexemplare. Die Berechtigung zum Erstellen eines Objekts gilt zum Beispiel für die Objektklasse, und nicht für ein einzelnes Exemplar. Einige Berechtigungen gelten zudem für den gesamten WS-Manager, so bedeutet zum Beispiel die Berechtigung als alternativer Benutzer, dass ein Benutzer die Berechtigungen, die einem anderen Benutzer zugeordnet sind, übernehmen kann.

Berechtigungsverzeichnisse

Die Berechtigungsverzeichnisse für einen WS-Manager mit dem Namen `saturn` lauten standardmäßig wie folgt:

MQS_ROOT:[MQM.QMGRS.SATURN.AUTH.QUEUES]
Berechtigungsdateien für Warteschlangen

MQS_ROOT:[MQM.QMGRS.SATURN.AUTH.PROCDEF]
Berechtigungsdateien für Prozessdefinitionen

MQS_ROOT:[MQM.QMGRS.SATURN.AUTH.QMANAGER]
Berechtigungsdateien für WS-Manager

MQS_ROOT:[MQM.QMGRS.SATURN.AUTH]$ACLASS
Berechtigungen für alle Klassen

MQS_ROOT:[MQM.QMGRS.SATURN.AUTH.NAMELIST]
Berechtigungen für alle Namenslisten

Die $CLASS-Dateien in den Objektverzeichnissen enthalten die Berechtigungen für die gesamte Klasse.

Anmerkung: Es besteht ein Unterschied zwischen $CLASS (die Berechtigungsdatei mit den Berechtigungen für eine einzelne Klasse) und $ACLASS (die Berechtigungsdatei mit den Berechtigungen für alle Klassen).

Die Pfade der Objektberechtigungsdateien basieren auf den Pfaden der Objekte selbst, wobei AUTH vor das Objektartverzeichnis eingefügt wird. Mit dem Befehl **dspmqfls** können Sie den Pfad für ein bestimmtes Objekt anzeigen.

Der Name und der Pfad für SYSTEM.DEFAULT.LOCAL.QUEUE lauten zum Beispiel:

```
MQS_ROOT:[MQM.QMGRS.SATURN.QUEUES.SYSTEM$DEFAULT$LOCAL$QUEUE]
```

Dann lauten der Name und Pfad für die entsprechende Berechtigungsdatei wie folgt:

```
MQS_ROOT:[MQM.QMGRS.SATURN.AUTH.QUEUES.SYSTEM$DEFAULT$LOCAL$QUEUE]
```

Anmerkung: In diesem Fall stimmen die tatsächlichen Namen der Dateien, die der Warteschlange zugeordnet sind, nicht mit dem Namen der Warteschlange selbst überein. Detaillierte Informationen finden Sie unter „Erläuterungen zu MQSeries-Dateinamen” auf Seite 21.

Inhalt der Berechtigungsdateien

Die Berechtigungen einer einzelnen ID werden durch eine Reihe von Zeilengruppen in der Berechtigungsdatei definiert. Weitere Informationen finden Sie unter „Erläuterung der Berechtigungsdateien” auf Seite 100. Die Berechtigungen gelten für das dieser Datei zugeordnete Objekt. Beispiel:

```
groupb:
   Authority=0x0040007
```

Diese Zeilengruppe definiert die Berechtigung für die ID GROUPB. Die Berechtigungsspezifikation ergibt sich aus der Zusammenfassung der einzelnen Bitmuster auf Basis der folgenden Zuordnungen:

Authorization keyword	Formal name	Hexadecimal Value
connect	MQZAO_CONNECT	0x00000001
browse	MQZAO_BROWSE	0x00000002
get	MQZAO_INPUT	0x00000004
put	MQZAO_OUTPUT	0x00000008
inq	MQZAO_INQUIRE	0x00000010
set	MQZAO_SET	0x00000020
passid	MQZAO_PASS_IDENTITY_CONTEXT	0x00000040
passall	MQZAO_PASS_ALL_CONTEXT	0x00000080
setid	MQZAO_SET_IDENTITY_CONTEXT	0x00000100
setall	MQZAO_SET_ALL_CONTEXT	0x00000200
altusr	MQZAO_ALTERNATE_USER_AUTHORITY	0x00000400
allmqi	MQZAO_ALL_MQI	0x000007FF
crt	MQZAO_CREATE	0x00010000
dlt	MQZAO_DELETE	0x00020000
dsp	MQZAO_DISPLAY	0x00040000
chg	MQZAO_CHANGE	0x00080000
clr	MQZAO_CLEAR	0x00100000
chgaut	MQZAO_AUTHORIZE	0x00800000
alladm	MQZAO_ALL_ADMIN	0x009E0000
none	MQZAO_NONE	0x00000000
all	MQZAO_ALL	0x009E07FF

Diese Definitionen erfolgen in der Header-Datei cmqzc.h. Im folgenden Beispiel werden der ID GROUPB Berechtigungen auf Basis der Hexadezimalzahl 0x40007 erteilt. Dies entspricht:

```
   MQZAO_CONNECT                  0x00000001
   MQZAO_BROWSE                   0x00000002
   MQZAO_INPUT                    0x00000004
   MQZAO_DISPLAY                  0x00040000
                                  ----------
  Authority is:                   0x00040007
```

Diese Zugriffsrechte bedeuten, dass jedes Mitglied von GROUPB die folgenden MQI-Aufrufe ausgeben kann:

MQCONN
MQGET (mit 'browse')

Alle Mitglieder verfügen außerdem über die Berechtigung DISPLAY für das Objekt, das dieser Berechtigungsdatei zugeordnet ist.

Berechtigungsdateien für Klassen

Die *Berechtigungsdateien für Klassen* enthalten Berechtigungen für die gesamte Klasse. Diese Dateien, die so genannten „$CLASS"-Dateien, befinden sich in demselben Verzeichnis wie die Dateien für einzelne Objekte. Der Eintrag MQZAO_CRT in der $CLASS-Datei erteilt die Berechtigung zum Erstellen eines Objekts in der Klasse. Dies ist die einzige Klassenberechtigung.

Berechtigungsdateien für alle Klassen

Die *Berechtigungsdatei für alle Klassen* enthält Berechtigungen, die für einen gesamten WS-Manager gelten. Diese Datei, die so genannte $ACLASS-Datei, befindet sich im Unterverzeichnis 'auth' des WS-Managers.

Die folgenden Berechtigungen gelten für den gesamten WS-Manager und sind in der Berechtigungsdatei für alle Klassen definiert:

Der Eintrag... Erteilt die Berechtigung zum...

MQZAO_ALTERNATE_USER_AUTHORITY
Übernehmen der Identität eines anderen Benutzers bei der Bearbeitung von MQSeries-Objekten

MQZAO_SET_ALL_CONTEXT
Angeben des Kontextes einer Nachricht beim Ausgeben eines MQPUT-Aufrufs

MQZAO_SET_IDENTITY_CONTEXT
Angeben des Identitätskontextes einer Nachricht beim Ausgeben eines MQPUT-Aufrufs

Berechtigungsdateien verwalten

Bei der Verwaltung Ihrer Berechtigungsdateien müssen Sie die folgenden Punkte beachten:

1. Sie müssen sicherstellen, dass die Berechtigungsdateien gesichert sind und kein Schreibzugriff durch unbefugte Endbenutzer möglich ist (siehe „Berechtigungen für Berechtigungsdateien").
2. Um Ihre Berechtigungsdateien gegebenenfalls wiederherstellen zu können, müssen Sie mindestens eine der folgenden Aktionen ausführen:
   - Sichern des Unterverzeichnisses AUTH nach jeder wichtigen Aktualisierung
   - Aufbewahren von DCL-Befehlsdateien mit den verwendeten Befehlen
3. Sie können Berechtigungsdateien kopieren und editieren. Es sollte jedoch im Normalfall nicht erforderlich sein, sie manuell zu erstellen oder zu reparieren. Sollte ein Notfall eintreten, können Sie verloren gegangene oder beschädigte Berechtigungsdateien mit Hilfe dieser Informationen wiederherstellen.

Berechtigungen für Berechtigungsdateien

Berechtigungsdateien müssen von jedem Principal gelesen werden können. Es sollte jedoch nur der Systemverwalter und der Benutzer mit der ID MQM berechtigt sein, diese Dateien zu aktualisieren.

Die Berechtigungen für Berechtigungsdateien, die vom OAM erstellt wurden, lauten:

```
S:RWD, O:RWD, G:RWD, W:R   (ID=MQM, ACCESS=R+W+E+D+C)
```

Ändern Sie diese Berechtigungen nicht, ohne vorher genau zu prüfen, ob durch die Änderungen möglicherweise Sicherheitslücken entstehen.

Um Berechtigungen mit dem in MQSeries for Compaq OpenVMS bereitgestellten Befehl ändern zu können, muss Ihr Prozess über die Berechtigungs-ID MQM verfügen.

Kapitel 8. MQSeries-Steuerroutine der Warteschlange für nicht zustellbare Nachrichten

Eine *Warteschlange für nicht zustellbare Nachrichten* (Dead-Letter Queue, DLQ), manchmal auch als *Warteschlange für nicht zugestellte Nachrichten* bezeichnet, ist eine Zwischenwarteschlange für Nachrichten, die nicht an ihre Zielwarteschlange übermittelt werden können. Jedem WS-Manager in einem Netz sollte eine DLQ zugeordnet sein.

Nachrichten können von WS-Managern, Nachrichtenkanalagenten (Message Channel Agents, MCAs) und Anwendungen in die DLQ eingereiht werden. Alle Nachrichten in der DLQ sollten mit einem *DLQ-Header* (MQDLH) als Präfix versehen werden. Nachrichten, die von einem WS-Manager oder einem Nachrichtenkanalagenten in die DLQ eingereiht werden, verfügen immer über einen MQDLH; es wird dringend empfohlen, dass Anwendungen, die Nachrichten in die DLQ einreihen, die Nachrichten mit einem MQDLH versehen. Das Feld *Reason* in der MQDLH-Struktur enthält einen Ursachencode mit einer Information, warum sich die Nachricht in der DLQ befindet.

In allen MQSeries-Umgebungen sollte eine Routine existieren, die in regelmäßigen Abständen Nachrichten aus der DLQ verarbeitet. MQSeries stellt eine Standardroutine, die so genannte *Steuerroutine der Warteschlange für nicht zustellbare Nachrichten* (DLQ-Steuerroutine) zur Verfügung, die Sie mit dem Befehl **runmqdlq** aufrufen können.

Anweisungen für die Verarbeitung von Nachrichten aus der DLQ werden über eine benutzerdefinierte *Regeltabelle* an die DLQ-Steuerroutine übergeben. Das heißt, die DLQ-Steuerroutine gleicht Nachrichten in der DLQ gegen Einträge in der Regeltabelle ab. Stimmt eine DLQ-Nachricht mit einem Eintrag in der Regeltabelle überein, führt die DLQ-Steuerroutine die Aktion aus, die dem betreffenden Eintrag zugeordnet ist.

Dieses Kapitel enthält die folgenden Abschnitte:

DLQ-Steuerroutine aufrufen

Die DLQ-Steuerroutine wird mit dem Befehl **runmqdlq** aufgerufen. Sie haben zwei Möglichkeiten, die Namen der zu verarbeitenden DLQ und des zu verwendenden WS-Managers anzugeben:

- Als Parameter des Befehls **runmqdlq** in der Eingabeaufforderung. Beispiel:

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER < qrule.rul
```

- In der Regeltabelle. Beispiel:

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

Diese Beispiele betreffen die DLQ mit dem Namen ABC1.DEAD.LETTER.QUEUE, die von dem WS-Manager ABC1.QUEUE.MANAGER verwaltet wird.

Wenn Sie keine DLQ oder keinen WS-Manager angeben, werden der bei der Installation definierte Standard-WS-Manager und die diesem zugeordnete DLQ verwendet.

Der Befehl **runmqdlq** erhält seine Eingabe aus SYS$INPUT. Sie können dem Befehl **runmqdlq** die Regeltabelle zuordnen, indem Sie statt SYS$INPUT die Regeltabelle als Eingabequelle angeben (Eingabe umleiten).

Achtung: Wenn Sie die DLQ-Steuerroutine ausführen, ohne statt SYS$INPUT eine Regeldatei als Eingabequelle zu verwenden, läuft die DLQ-Steuerroutine in eine Schleife.

Um die DLQ-Steuerroutine ausführen zu können, müssen Sie über Zugriffsberechtigungen sowohl für die DLQ selbst als auch für alle Nachrichtenwarteschlangen verfügen, an die Nachrichten aus der DLQ weitergeleitet werden. Wenn die DLQ-Steuerroutine in der Lage ist, Nachrichten in Warteschlangen mit der Berechtigung der im Nachrichtenkontext angegebenen Benutzer-ID einzureihen, müssen Sie darüber hinaus über die Berechtigung verfügen, die Identität anderer Benutzer anzunehmen.

Weitere Informationen zum Befehl **runmqdlq** finden Sie unter „runmqdlq (Steuerroutine der DLQ ausführen)” auf Seite 299.

Die Beispiel-DLQ-Steuerroutine (amqsdlq)

Neben der DLQ-Steuerroutine, die mit dem Befehl **runmqdlq** aufgerufen wird, stellt MQSeries die Quelle für eine Beispiel-DLQ-Steuerroutine (amqsdlq) bereit, deren Funktion der über den Befehl **runmqdlq** aufgerufenen Steuerroutine ähnlich ist. Die Quellen werden lediglich als Schablonen bereitgestellt und müssen so angepasst werden, dass daraus eine DLQ-Steuerroutine entsteht, die die spezifischen lokalen Anforderungen erfüllt. Es kann zum Beispiel sein, dass Sie eine DLQ-Steuerroutine benötigen, die Nachrichten ohne DLQ-Header verarbeiten kann. (Sowohl die Standard-DLQ-Steuerroutine als auch die Beispiel-Steuerroutine amqsdlq verarbeiten nur diejenigen Nachrichten in der DLQ, die mit einem DLQ-Header (MQDLH) beginnen. Nachrichten, die nicht mit einem MQDLH beginnen, werden als fehlerhaft betrachtet und verbleiben auf unbestimmte Zeit in der DLQ.)

Die Quelle für amqsdlq befindet sich in folgendem Verzeichnis:

```
 [.DLQ]
unter MQS_EXAMPLES
```

Die kompilierte Version befindet sich in folgendem Verzeichnis:

```
 [.BIN]
unter MQS_EXAMPLES
```

Die Regeltabelle der DLQ-Steuerroutine

Die Regeltabelle der DLQ-Steuerroutine definiert, wie die DLQ-Steuerroutine Nachrichten, die in der DLQ ankommen, verarbeitet. Eine Regeltabelle enthält zwei Arten von Einträgen:

- Der erste Eintrag in der Tabelle ist optional und enthält *Steuerdaten*.
- Bei allen anderen Einträgen in der Tabelle handelt es sich um *Regeln* für die DLQ-Steuerroutine. Jede Regel besteht aus einem *Muster* (einer Gruppe von Nachrichtenmerkmalen), gegen das eine Nachricht abgeglichen wird, und einer *Aktion*, die ausgeführt wird, wenn eine Nachricht in der DLQ mit dem angegebenen Muster übereinstimmt. Eine Regeltabelle muss mindestens eine Regel enthalten.

Jeder Eintrag in der Regeltabelle besteht aus einem oder mehreren Schlüsselwörtern.

Steuerdaten

Dieser Abschnitt beschreibt die Schlüsselwörter, die ein Steuerdateneintrag in einer Regeltabelle der DLQ-Steuerroutine enthalten darf. Bitte beachten Sie Folgendes:

- Der Standardwert für ein Schlüsselwort ist unterstrichen.
- Die vertikale Linie (|) trennt alternative Werte, d. h., es kann jeweils nur einer der Werte angegeben werden.
- Alle Schlüsselwörter sind optional.

INPUTQ (*Name der Warteschlange***|'_')**

Ermöglicht die Angabe des Namens der zu verarbeitenden DLQ:

1. Wenn Sie einen INPUTQ-Wert als Parameter des Befehls **runmqdlq** angeben, überschreibt er einen eventuell vorhanden INPUTQ-Wert in der Regeltabelle.
2. Wenn Sie keinen INPUTQ-Wert als Parameter des Befehls **runmqdlq**, aber einen Wert in der Regeltabelle angeben, wird der INPUTQ-Wert aus der Regeltabelle verwendet.
3. Wenn Sie keine DLQ oder INPUTQ(' ') in der Regeltabelle angeben, wird der Name der DLQ des WS-Managers, dessen Name als Parameter des Befehls **runmqdlq** übergeben wird, verwendet.
4. Wenn Sie keinen INPUTQ-Wert als Parameter des Befehls **runmqdlq** oder als Wert in der Regeltabelle angeben, wird die DLQ des WS-Managers, dessen Name im Schlüsselwort INPUTQM in der Regeltabelle angegeben ist, verwendet.

INPUTQM (*WS-Manager-Name***|'_')**

Ermöglicht die Angabe des Namens des WS-Managers, der die DLQ verwaltet, die im Schlüsselwort INPUTQ angegeben ist:

1. Wenn Sie einen INPUTQM-Wert als Parameter des Befehls **runmqdlq** angeben, überschreibt er einen eventuell vorhanden INPUTQM-Wert in der Regeltabelle.
2. Wenn Sie keinen INPUTQM-Wert als Parameter des Befehls **runmqdlq** angeben, wird der INPUTQ-Wert aus der Regeltabelle verwendet.
3. Wenn Sie keinen WS-Manager oder nicht INPUTQM(' ') in der Regeltabelle angeben, wird der bei der Installation definierte Standard-WS-Manager verwendet.

RETRYINT (*Intervall* **| 60)**
Dies ist das Intervall in Sekunden, in dem die DLQ-Steuerroutine versuchen soll, Nachrichten in der DLQ, die beim ersten Versuch nicht verarbeitet werden konnten und für die entsprechende Wiederholungen angefordert wurden, erneut zu verarbeiten. Das Wiederholungsintervall ist standardmäßig auf 60 Sekunden eingestellt.

WAIT (YES | NO | *nnn***)**
Gibt an, ob die DLQ-Steuerroutine auf den Eingang weiterer Nachrichten in der DLQ warten soll, wenn sie feststellt, dass keine weiteren Nachrichten vorliegen, die sie verarbeiten könnte.

YES Die DLQ-Steuerroutine wartet auf unbestimmte Zeit.

NO Die DLQ-Steuerroutine wird beendet, wenn sie feststellt, dass die DLQ entweder leer ist oder keine Nachrichten enthält, die sie verarbeiten kann.

nnn Die DLQ-Steuerroutine wartet *nnn* Sekunden lang auf neue zu verarbeitende Nachrichten, bevor sie beendet wird, nachdem sie festgestellt hat, dass die Warteschlange entweder leer ist oder keine Nachrichten enthält, die sie verarbeiten kann.

Es wird empfohlen, WAIT (YES) für stark ausgelastete DLQs und WAIT (NO) oder WAIT (*nnn*) für DLQs mit einer geringen Auslastung zu verwenden. Wenn die DLQ-Steuerroutine auf diese Weise beendet wird, sollte sie mit Hilfe der Auslösefunktion erneut aufgerufen werden.

Als Alternative zum Einfügen von Steuerdaten in der Regeltabelle können Sie die Namen der DLQ und des WS-Managers als Eingabeparameter des Befehls **runmqdlq** angeben. Wenn sowohl in der Regeltabelle als auch mit dem Befehl **runmqdlq** ein Wert angegeben wird, hat der mit dem Befehl **runmqdlq** angegebene Wert Vorrang.

Anmerkung: Wenn in der Regeltabelle ein Steuerdateneintrag eingefügt wird, ***muss*** es sich um den ersten Eintrag in der Tabelle handeln.

Regeln (Muster und Aktionen)

Abb. 8 zeigt eine Beispielregel aus einer Regeltabelle der DLQ-Steuerroutine.

```
PERSIST(MQPER_PERSISTENT) REASON(MQRC_PUT_INHIBITED) +
 ACTION(RETRY) RETRY(3)
```

Abbildung 8. Beispielregel aus einer Regeltabelle der DLQ-Steuerroutine. Diese Regel weist die DLQ-Steuerroutine an, drei Versuche zu unternehmen, um permanente Nachrichten, die auf Grund inaktivierter MQPUT- und MQPUT1-Aufrufe in die DLQ eingereiht wurden, an ihre Zielwarteschlangen zuzustellen.

Alle Schlüsselwörter, die Sie in einer Regel verwenden können, werden später in diesem Abschnitt beschrieben. Bitte beachten Sie Folgendes:

- Der Standardwert für ein Schlüsselwort ist unterstrichen. Der Standardwert für die meisten Schlüsselwörter ist der Stern (*), der für einen beliebigen Wert steht.
- Die vertikale Linie (|) trennt alternative Werte, d. h., es kann jeweils nur einer der Werte angegeben werden.
- Alle Schlüsselwörter mit Ausnahme von ACTION sind optional.

Dieser Abschnitt beginnt mit einer Beschreibung der Schlüsselwörter für die Mustererkennung (gegen die Nachrichten aus der DLQ abgeglichen werden), gefolgt von einer Beschreibung der Aktionsschlüsselwörter (die festlegen, wie die DLQ-Steuerroutine eine übereinstimmende Nachricht verarbeiten soll).

Schlüsselwörter für Mustererkennung

Die Schlüsselwörter für die Mustererkennung, mit denen Sie die Werte angeben, gegen die Nachrichten aus der DLQ abgeglichen werden, werden im Folgenden beschrieben. Alle Schlüsselwörter für die Mustererkennung sind optional.

APPLIDAT (*ApplIdentityData* **| *)**
Dies ist der Wert *ApplIdentityData*, der im Nachrichtendeskriptor MQMD der Nachricht in der DLQ angegeben ist.

APPLNAME (*PutApplName* **| *)**
Dies ist der Name der Anwendung, die den MQPUT- oder MQPUT1-Aufruf ausgibt, wie im Feld *PutApplName* des Nachrichtendeskriptors MQMD der Nachricht in der DLQ angegeben.

APPLTYPE (*PutApplType* **| *)**
Dies ist der Wert *PutApplType*, der im Nachrichtendeskriptor MQMD der Nachricht in der DLQ angegeben ist.

DESTQ (*Name der Warteschlange* **| *)**
Dies ist der Name der Nachrichtenwarteschlange, an die die Nachricht gerichtet ist.

DESTQM (*WS-Manager-Name* **| *)**
Dies ist der Name des WS-Managers der Nachrichtenwarteschlange, an die die Nachricht gerichtet ist.

FEEDBACK (*Rückmeldung* **| *)**
Wenn der Wert für *MsgType* auf MQFB_REPORT gesetzt wird, beschreibt *Rückmeldung* die Art des Berichts.

Es können symbolische Namen verwendet werden. Sie können zum Beispiel den symbolischen Namen MQFB_COA verwenden, um diejenigen Nachrichten in der DLQ zu identifizieren, für die eine Bestätigung des Eingangs in ihren Zielwarteschlangen angefordert wird.

FORMAT (*Format* **| *)**
Dies ist der Name, den der Sender der Nachricht zur Beschreibung des Formats der Nachrichtendaten verwendet.

MSGTYPE (*Nachrichtenart* **| *)**
Dies ist die Nachrichtenart der Nachricht in der DLQ.

Es können symbolische Namen verwendet werden. Sie können zum Beispiel den symbolischen Namen MQMT_REQUEST verwenden, um diejenigen Nachrichten in der DLQ zu identifizieren, für die Antworten angefordert werden.

PERSIST (*Permanenz* **| *)**
Dies ist der Wert für die Permanenz der Nachricht. (Die Permanenz einer Nachricht legt fest, ob die Nachricht bei Neustarts des WS-Managers erhalten bleibt.)

Es können symbolische Namen verwendet werden. Sie können zum Beispiel den symbolischen Namen MQPER_PERSISTENT verwenden, um diejenigen Nachrichten in der DLQ zu identifizieren, die permanent sind.

REASON (*Ursachencode* **| *)**
Dies ist der Ursachencode, der beschreibt, warum die Nachricht in die DLQ eingereiht wurde.

Es können symbolische Namen verwendet werden. Sie können zum Beispiel den symbolischen Namen MQRC_Q_FULL verwenden, um diejenigen Nachrichten zu identifizieren, die in die DLQ eingereiht wurden, weil ihre Zielwarteschlangen voll waren.

REPLYQ (*Name der Warteschlange* **| *)**
Dies ist der Name der Warteschlange für zu beantwortende Nachrichten, der im Nachrichtendeskriptor MQMD der Nachricht in der DLQ angegeben ist.

REPLYQM (*WS-Manager-Name* **| *)**
Dies ist der Name des WS-Managers der Warteschlange für zu beantwortende Nachrichten, der im Nachrichtendeskriptor MQMD der Nachricht in der DLQ angegeben ist.

USERID (*Benutzer-ID* **| *)**
Dies ist die Benutzer-ID des Benutzers, der die Nachricht in der DLQ gesendet hat, so wie im Nachrichtendeskriptor MQMD angegeben.

Aktionsschlüsselwörter

Die Aktionsschlüsselwörter, mit denen Sie angeben, wie eine übereinstimmende Nachricht verarbeitet werden soll, werden im Folgenden beschrieben.

ACTION (DISCARD | IGNORE | RETRY | FWD)
Dies ist die Aktion, die für jede Nachricht in der DLQ ausgeführt wird, die mit dem in dieser Regel definierten Muster übereinstimmt.

DISCARD
Die Nachricht wird aus der DLQ gelöscht.

IGNORE
Die Nachricht bleibt in der DLQ stehen.

RETRY
Die DLQ-Steuerroutine versucht erneut, die Nachricht in ihre Zielwarteschlange einzureihen.

FWD Die DLQ-Steuerroutine leitet die Nachricht an die Warteschlange weiter, die im Schlüsselwort FWDQ angegeben ist.

Das Schlüsselwort ACTION muss angegeben werden. Die Anzahl der Versuche zum Implementieren einer Aktion wird durch das Schlüsselwort RETRY festgelegt. Das Intervall zwischen Versuchen wird durch das Schlüsselwort RETRYINT in den Steuerdaten festgelegt.

FWDQ (*Name der Warteschlange* **| &DESTQ | &REPLYQ)**
Dies ist der Name der Nachrichtenwarteschlange, an die die Nachricht weitergeleitet werden soll, wenn ACTION (FWD) angefordert wird.

Name der Warteschlange
Dies ist der Name einer Nachrichtenwarteschlange. FWDQ(' ') ist nicht gültig.

&DESTQ
Der Name der Warteschlange wird aus dem Feld *DestQName* in der MQDLH-Struktur übernommen.

&REPLYQ
Der Name wird aus dem Feld *ReplyToQ* im Nachrichtendeskriptor MQMD übernommen.

Um Fehlermeldungen zu vermeiden, wenn eine Regel, in der FWDQ (&REPLYQ) angegeben ist, mit einer Nachricht übereinstimmt, die ein leeres Feld *ReplyToQ* enthält, können Sie REPLYQ (?*) im Nachrichtenmuster angeben.

FWDQM (*WS-Manager-Name* **| &DESTQM | &REPLYQM | '_')**
Gibt den Namen des WS-Managers der Warteschlange an, an die eine Nachricht weitergeleitet werden soll.

Name des WS-Managers
Dies ist der Name des WS-Managers der Nachrichtenwarteschlange, an die eine Nachricht weitergeleitet werden soll, wenn ACTION (FWD) angefordert wird.

&DESTQM
Der Name des WS-Managers wird aus dem Feld *DestQMgrName* in der MQDLH-Struktur übernommen.

&REPLYQM
Der Name wird aus dem Feld *ReplyToQMgr* des Nachrichtendeskriptors MQMD übernommen.

' ' Der Standardwert FWDQM(' ') identifiziert den lokalen WS-Manager.

HEADER (YES | NO)
Gibt an, ob der Header MQDLH in einer Nachricht stehen bleiben soll, für die ACTION (FWD) angefordert wird. Der Header MQDLH bleibt standardmäßig in der Nachricht stehen. Das Schlüsselwort HEADER ist nur für die Aktion FWD gültig.

PUTAUT (DEF | CTX)
Definiert die Berechtigung, mit der Nachrichten von der DLQ-Steuerroutine eingereiht werden:

DEF Nachrichten werden mit der Berechtigung der DLQ-Steuerroutine eingereiht.

CTX Nachrichten werden mit der Berechtigung der im Nachrichtenkontext angegebenen Benutzer-ID eingereiht. Wenn Sie PUTAUT (CTX) angeben, müssen Sie über die Berechtigung verfügen, die Identität anderer Benutzer anzunehmen.

RETRY (*Wiederholungszähler* **| 1)**
Gibt an, wie oft (ein Wert von 1 bis 999.999.999) eine Aktion wiederholt werden soll (in dem Intervall, das mit dem Schlüsselwort RETRYINT in den Steuerdaten angegeben ist).

Anmerkung: Die Anzahl der Wiederholungen, die von der DLQ-Steuerroutine zum Implementieren einer bestimmten Regel ausgeführt werden, gilt nur für das aktuelle Exemplar der DLQ-Steuerroutine. Der Zähler bleibt bei Neustarts nicht erhalten. Wenn die DLQ-Steuerroutine erneut gestartet wird, wird der Zähler für die Anzahl der Wiederholungen zum Anwenden einer Regel auf null zurückgesetzt.

Konventionen für die Regeltabelle

Die Regeltabelle muss in Bezug auf ihre Syntax, Struktur und ihren Inhalt den folgenden Konventionen entsprechen:

- Eine Regeltabelle muss mindestens eine Regel enthalten.
- Die Reihenfolge der Schlüsselwörter ist beliebig.
- Ein Schlüsselwort darf nur ein Mal in einer Regel vorkommen.
- Bei Schlüsselwörtern wird die Groß-/Kleinschreibung nicht beachtet.
- Ein Schlüsselwort und seine Parameterwerte müssen durch mindestens ein Leerzeichen oder Komma von anderen Schlüsselwörtern getrennt werden.
- Vor und nach einer Regel sowie zwischen Schlüsselwörtern, Satzzeichen und Werten können beliebig viele Leerzeichen stehen.
- Jede Regel muss in einer neuen Zeile beginnen.
- Aus Gründen der Portierbarkeit sollte die signifikante Länge einer Zeile 72 Zeichen nicht überschreiten.
- Verwenden Sie das Pluszeichen (+) als letztes belegtes Zeichen in einer Zeile, um anzugeben, dass die Regel in der nächsten Zeile mit dem ersten Zeichen, bei dem es sich nicht um ein Leerzeichen handelt, fortgesetzt wird. Verwenden Sie das Minuszeichen (–) als letztes belegtes Zeichen in einer Zeile, um anzugeben, dass die Regel am Beginn der nächsten Zeile fortgesetzt wird. Fortsetzungszeichen können zwischen Schlüsselwörtern und Parametern stehen.
- Kommentarzeilen, die mit einem Stern (*) beginnen, können an jeder beliebigen Stelle in der Regeltabelle stehen.
- Leerzeilen werden ignoriert.
- Jeder Eintrag in der Regeltabelle der DLQ-Steuerroutine besteht aus einem oder mehreren Schlüsselwörtern und den zugehörigen Parametern. Für die Parameter gelten die folgenden Syntaxregeln:
  - Jeder Parameterwert muss mindestens ein signifikantes Zeichen enthalten. Die Anführungszeichen, die als Begrenzungszeichen für Werte verwendet werden, gelten nicht als signifikante Zeichen. Bei den folgenden Beispielen handelt es sich um gültige Parameter:
    FORMAT('ABC') 3 signifikante Zeichen
    FORMAT(ABC) 3 signifikante Zeichen
    FORMAT('A') 1 signifikantes Zeichen
    FORMAT(A) 1 signifikantes Zeichen
    FORMAT(' ') 1 signifikantes Zeichen

Diese Parameter sind ungültig, weil sie keine signifikanten Zeichen enthalten:
FORMAT('')
FORMAT( )
FORMAT()
FORMAT

- Platzhalterzeichen werden unterstützt: Das Fragezeichen (?) kann anstelle eines einzelnen Zeichens, außer für ein abschließendes Leerzeichen, und der Stern (*) anstelle einer Null oder mehrerer aufeinanderfolgender Zeichen verwendet werden. Der Stern (*) und das Fragezeichen (?) werden in Parameterwerten ***immer*** als Platzhalterzeichen interpretiert.
- Platzhalterzeichen sind in Parametern der folgenden Schlüsselwörter nicht zulässig: ACTION, HEADER, RETRY, FWDQ, FWDQM und PUTAUT.
- Abschließende Leerzeichen in Parameterwerten und in den entsprechenden Feldern der Nachricht in der DLQ gelten nicht als signifikante Zeichen, wenn Platzhalterzeichenabgleiche ausgeführt werden. Führende und eingebettete Leerzeichen innerhalb von Zeichenfolgen in Anführungszeichen gelten bei Platzhalterzeichenabgleichen jedoch als signifikante Zeichen.
- Numerische Parameter dürfen nicht das Fragezeichen (?) als Platzhalterzeichen enthalten. Der Stern (*) kann anstelle eines vollständigen numerischen Parameters verwendet werden, darf jedoch nicht Teil eines numerischen Parameters sein. Bei den folgenden Beispielen handelt es sich um gültige numerische Parameter:
  MSGTYPE(2) Es sind nur Antwortnachrichten auswählbar.
  MSGTYPE(*) Alle Nachrichtenarten sind auswählbar.
  MSGTYPE('*') Alle Nachrichtenarten sind auswählbar.

  `MSGTYPE('2*')` ist jedoch nicht gültig, da es einen Stern (*) als Teil eines numerischen Parameters enthält.
- Numerische Parameter müssen im Bereich 0 bis 999.999.999 liegen. Ein Parameterwert, der in diesem Bereich liegt, wird akzeptiert, und zwar auch dann, wenn er in dem Feld, auf das sich das Schlüsselwort bezieht, eigentlich nicht gültig ist. Für numerische Parameter können symbolische Namen verwendet werden.
- Wenn ein Zeichenfolgewert kleiner als das Feld im Header MQDLH oder MQMD ist, auf das sich das Schlüsselwort bezieht, wird das Feld bis zum Ende mit Leerzeichen aufgefüllt. Ist der Wert (ohne Sterne) länger als das Feld, wird ein Fehler erkannt. Bei den folgenden Beispielen handelt es sich um gültige Zeichenfolgewerte für ein acht Zeichen langes Feld:
  'ABCDEFGH' 8 Zeichen
  'A*C*E*G*I' 5 Zeichen (ohne die Sterne)
  '*A*C*E*G*I*K*M*O*' 8 Zeichen (ohne die Sterne)
- Zeichenfolgen mit Leerzeichen, Kleinbuchstaben oder Sonderzeichen, außer Punkt (.), Schrägstrich (/), Unterstreichung (_) und Prozentzeichen (%), müssen in einfachen Anführungszeichen stehen. Kleinbuchstaben, die nicht in Anführungszeichen stehen, werden in Großbuchstaben umgesetzt. Wenn die Zeichenfolge ein Zitat enthält, müssen sowohl am Anfang als auch am Ende des Zitats zwei einfache Anführungszeichen stehen. Bei der Berechnung der Zeichenfolgelänge wird jedes Vorkommen von doppelten Anführungszeichen als ein einfaches Anführungszeichen gezählt.

Verarbeitung der Regeltabelle

Die DLQ-Steuerroutine durchsucht die Regeltabelle nach einer Regel, deren Muster mit einer Nachricht in der DLQ übereinstimmt. Die Suche beginnt mit der ersten Regel in der Tabelle und geht dann die Regeln sequenziell bis zum Ende der Tabelle durch. Wenn eine Regel mit einem übereinstimmenden Muster gefunden wird, wird die Aktion der betreffenden Regel ausgeführt. Die DLQ-Steuerroutine erhöht den Wiederholungszähler für eine Regel um 1, wenn sie versucht, die Regel anzuwenden. Schlägt der erste Versuch fehl, wird der Versuch so lange wiederholt, bis die Anzahl der Versuche mit dem im Schlüsselwort RETRY angegebenen Wert übereinstimmt. Wenn alle Versuche fehlschlagen, sucht die DLQ-Steuerroutine nach der nächsten übereinstimmenden Regel in der Tabelle.

Dieser Prozess wird so lange für weitere übereinstimmende Regeln wiederholt, bis eine Aktion erfolgreich ausgeführt werden kann. Wenn jede übereinstimmende Regel so oft versucht wurde, wie im jeweils zugehörigen Schlüsselwort RETRY angegeben, aber kein Versuch erfolgreich war, wird ACTION (IGNORE) angenommen. ACTION (IGNORE) wird auch angenommen, wenn keine übereinstimmende Regel gefunden wird.

Anmerkungen:

1. Übereinstimmende Regelmuster werden nur für Nachrichten in der DLQ gesucht, die mit einem Header MQDLH beginnen. Nachrichten, die nicht mit einem MQDLH beginnen, werden in regelmäßigen Abständen als fehlerhaft gemeldet und verbleiben auf unbestimmte Zeit in der DLQ.
2. Für alle Musterschlüsselwörter können Standardwerte verwendet werden, so dass eine Regel nur aus einer Aktion bestehen kann. Beachten Sie jedoch, dass diese nur aus einer Aktion bestehenden Regeln auf alle Nachrichten in der Warteschlange angewendet werden, die über MQDLHs verfügen und die noch nicht nach anderen Regeln in der Tabelle verarbeitet wurden.
3. Die Gültigkeit der Regeltabelle wird beim Starten der DLQ-Steuerroutine überprüft, und Fehler werden markiert. Sie können die Regeltabelle jederzeit ändern, die Änderungen werden jedoch erst wirksam, wenn die DLQ-Steuerroutine erneut gestartet wird.
4. Die DLQ-Steuerroutine ändert weder den Inhalt von Nachrichten noch den des Headers MQDLH oder des Nachrichtendeskriptors. Die DLQ-Steuerroutine reiht Nachrichten immer mit der Nachrichtenoption MQPMO_PASS_ALL-_CONTEXT in andere Warteschlangen ein.
5. Die DLQ-Steuerroutine öffnet die DLQ mit der Option MQOO_INPUT_AS_Q-_DEF.
6. Mehrere Exemplare der DLQ-Steuerroutine können gleichzeitig unter Verwendung derselben Regeltabelle für dieselbe Warteschlange ausgeführt werden. Im Normalfall besteht jedoch eine Eins-zu-eins-Beziehung zwischen einer DLQ und einer DLQ-Steuerroutine.

Verarbeitung aller DLQ-Nachrichten sicherstellen

Die DLQ-Steuerroutine führt ein Protokoll über die Nachrichten in der DLQ, die gelesen, aber nicht entfernt wurden. Wenn Sie die DLQ-Steuerroutine als ein Filter zum Extrahieren einer kleinen Untermenge von Nachrichten aus der DLQ verwenden, muss die DLQ-Steuerroutine für die Nachrichten in der DLQ, die nicht von ihr verarbeitet wurden, weiter ein Protokoll führen. Die DLQ-Steuerroutine kann auch nicht garantieren, dass neu in der DLQ eingehende Nachrichten gelesen werden, nicht einmal dann, wenn die DLQ als Warteschlange im FIFO (First In/First Out)-Modus definiert wurde. Wenn die Warteschlange nicht leer ist, wird die DLQ deshalb in regelmäßigen Abständen neu durchsucht, um alle Nachrichten zu überprüfen. Aus diesen Gründen sollten Sie möglichst sicherstellen, dass die DLQ so wenig Nachrichten wie möglich enthält. Wenn sich Nachrichten, die nicht gelöscht oder an andere Warteschlangen weitergeleitet werden können (aus welchen Gründen auch immer), in der Warteschlange ansammeln, erhöht sich die Arbeitsbelastung der DLQ-Steuerroutine, und es besteht die Gefahr, dass die DLQ vollständig gefüllt wird.

Sie können besondere Maßnahmen ergreifen, um der DLQ-Steuerroutine zu ermöglichen, die DLQ zu leeren. Vermeiden Sie zum Beispiel ACTION (IGNORE), da diese Aktion lediglich Nachrichten in der DLQ belässt. (Denken Sie daran, dass ACTION (IGNORE) für Nachrichten verwendet wird, auf die keine andere Regel in der Tabelle angewendet werden kann.) Verwenden Sie für Nachrichten, die ignoriert werden sollen, eine Aktion, mit der die Nachrichten in eine andere Warteschlange verschoben werden. Beispiel:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Die letzte Regel in der Tabelle sollte so definiert werden, dass sie alle Nachrichten erfasst, auf die keine der vorausgehenden Regeln in der Tabelle angewendet werden konnte. Die letzte Regel in der Tabelle könnte zum Beispiel folgendermaßen lauten:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

Mit dieser Aktion werden Nachrichten, auf die keine der vorausgehenden Regeln in der Tabelle angewendet werden konnte, an die Warteschlange `REALLY.DEAD.QUEUE` weitergeleitet, wo sie manuell verarbeitet werden können. Wenn eine solche letzte Regel nicht vorhanden ist, bleiben Nachrichten auf unbestimmte Zeit in der DLQ stehen.

Beispiel einer Regeltabelle der DLQ-Steuerroutine

Es folgt ein Beispiel für eine Regeltabelle, die einen Steuerdateneintrag und mehrere Regeln enthält:

```
*************************************************************************
*        An example rules table for the runmqdlq command                *
*************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ')  inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.

* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
  action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.
```

```
REPLYQM(CCCC.*) +
  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message.

PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)

* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We don't have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

DLQ-Steuerroutine

Kapitel 9. Instrumentierungsereignisse

Mit Hilfe der MQSeries-Instrumentierungsereignisse können Sie den Betrieb von WS-Managern überwachen. Dieses Kapitel gibt eine kurze Einführung in Instrumentierungsereignisse. Eine ausführlichere Beschreibung finden Sie im Abschnitt über Instrumentierungsereignisse im Handbuch *MQSeries Programmable System Management*.

Funktion von Instrumentierungsereignissen

Instrumentierungsereignisse bilden die Ursache für die Generierung spezieller Nachrichten, so genannter *Ereignisnachrichten*, sobald ein WS-Manager erkennt, dass eine vordefinierte Kombination von Bedingungen eingetreten ist. Zum Beispiel verursachen folgende Bedingungen das Ereignis *Warteschlange voll*:

- 'Warteschlange voll'-Ereignisse sind für eine bestimmte Warteschlange aktiviert.
- Eine Anwendung gibt einen MQPUT-Aufruf aus, um eine Nachricht in diese Warteschlange einzureihen, der Aufruf schlägt jedoch fehl, weil die Warteschlange voll ist.

Andere Bedingungen, die Instrumentierungsereignisse verursachen können:

- Ein Schwellenwert für die Anzahl der Nachrichten in einer Warteschlange wird erreicht.
- Eine Warteschlange wird innerhalb eines bestimmten Zeitraums nicht verwendet.
- Ein Kanalexemplar wird gestartet oder gestoppt.
- Auf einem MQSeries for Compaq OpenVMS-System versucht eine Anwendung, eine Warteschlange zu öffnen, und gibt dabei eine nicht berechtigte Benutzer-ID an.

Mit Ausnahme von Kanalereignissen müssen alle Instrumentierungsereignisse aktiviert werden, bevor sie generiert werden können.

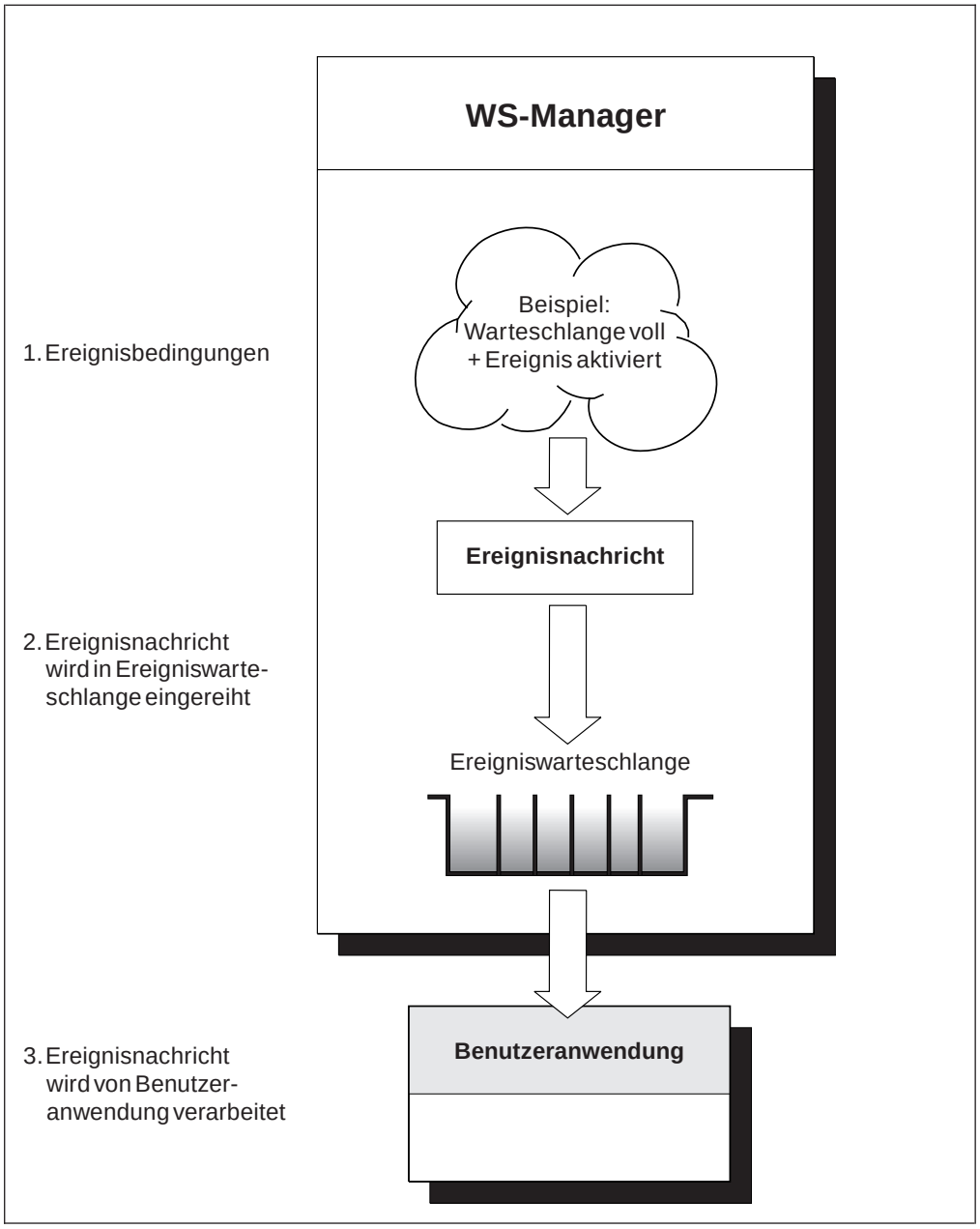


Abbildung 9. Erläuterung zu Instrumentierungsereignissen. Wenn ein WS-Manager erkennt, dass die Bedingungen für ein Ereignis eingetreten sind, reiht er eine Ereignisnachricht in die entsprechende Ereigniswarteschlange ein.

Die Ereignisnachricht, die Informationen zu den Bedingungen enthält, die das Ereignis verursacht haben, wird in eine *Ereigniswarteschlange* eingereiht. Die Ereignisnachricht kann von einer Anwendung zu Analysezwecken aus dieser Warteschlange abgerufen werden.

Gründe für die Verwendung von Ereignissen

Wenn Sie Ereigniswarteschlangen als ferne Warteschlangen definieren, können Sie alle Ereigniswarteschlangen einem einzigen WS-Manager zuordnen (für diejenigen Knoten, die Instrumentierungsereignisse unterstützen). Mit Hilfe der generierten Ereignisse können Sie dann ein Netz von WS-Managern von einem einzigen Knoten aus überwachen. Dies wird in Abb. 10 gezeigt.

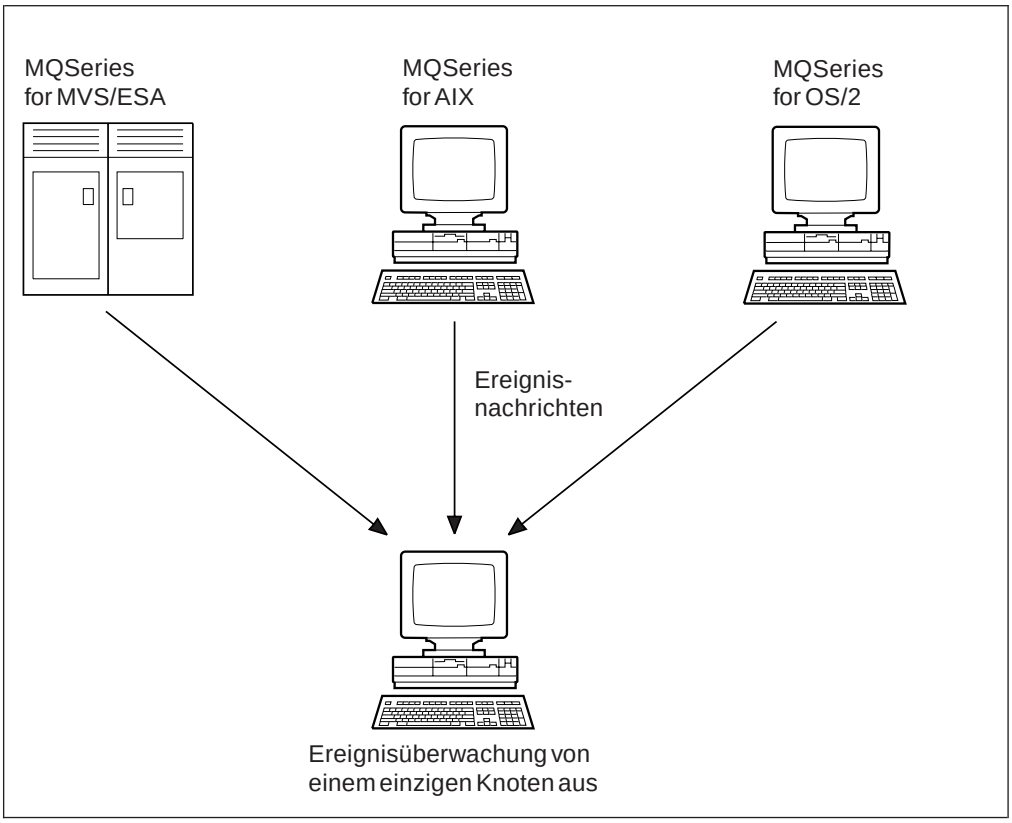


Abbildung 10. Überwachen von WS-Managern über verschiedene Plattformen hinweg von einem einzelnen Knoten aus

Ereignisarten

MQSeries-Ereignisse können in folgende Kategorien eingeteilt werden:

WS-Manager-Ereignisse
Diese Ereignisse beziehen sich auf die Definitionen von Ressourcen in WS-Managern. Eine Anwendung versucht zum Beispiel eine Nachricht in eine Warteschlange einzureihen, die nicht existiert.

Leistungsereignisse
Diese Ereignisse sind Hinweise, dass für eine Ressource ein Schwellenwert erreicht wurde. Beispiele: Der Grenzwert für die Länge einer Warteschlange wurde erreicht, oder die Warteschlange wurde nach einem GET-Aufruf nicht innerhalb einer vordefinierten Zeit bedient.

Kanalereignisse

Diese Ereignisse werden von Kanälen berichtet, wenn sie während ihres Betriebs entsprechende Bedingungen erkennen. Dies ist zum Beispiel der Fall, wenn ein Kanalexemplar gestoppt wird.

Auslöseereignisse

Wenn in diesem oder anderen MQSeries-Büchern von 'Auslösen' gesprochen wird, bezieht sich das in einigen Fällen auf ein *Auslöseereignis*. Dies tritt ein, wenn ein WS-Manager erkennt, dass die Bedingungen für ein Auslöseereignis erfüllt sind. Eine Warteschlange kann zum Beispiel so konfiguriert werden, dass sie bei jedem Eingang einer Nachricht ein Auslöseereignis generiert. (Es bestehen erhebliche Unterschiede zwischen den Bedingungen für Auslöseereignisse und denen für Instrumentierungsereignisse.)

Nach einem Auslöseereignis wird eine Auslösenachricht in eine Initialisierungswarteschlange eingereiht und, optional, ein Anwendungsprogramm gestartet.

Ereignisaufzeichnung durch Ereigniswarteschlangen

Wenn ein Ereignis eintritt, reiht der WS-Manager eine Ereignisnachricht in die entsprechende Ereigniswarteschlange (falls definiert) ein. Die Ereignisnachricht enthält Informationen zu dem Ereignis, die Sie abrufen können, indem Sie ein geeignetes MQI-Anwendungsprogramm mit folgenden Funktionen erstellen:

- Abrufen der Nachricht aus der Warteschlange.
- Verarbeiten der Nachricht zum Extrahieren der Ereignisdaten. Eine Beschreibung der Ereignisnachrichtenformate finden Sie im Handbuch *MQSeries Programmable System Management*.

Für jede Ereigniskategorie gibt es eine eigene Ereigniswarteschlange. Alle Ereignisse einer Kategorie haben zur Folge, dass eine Ereignisnachricht in die zugehörige Warteschlange eingereiht wird.

Diese Ereigniswarteschlange...	**enthält Nachrichten zu...**
SYSTEM.ADMIN.QMGR.EVENT	WS-Manager-Ereignissen
SYSTEM.ADMIN.PERFM.EVENT	Leistungsereignissen
SYSTEM.ADMIN.CHANNEL.EVENT	Kanalereignissen

Sie können Ereigniswarteschlangen als lokale oder ferne Warteschlangen definieren. Wenn Sie alle Ereigniswarteschlangen als ferne Warteschlangen auf demselben WS-Manager definieren, können Sie Ihre Überwachungsfunktionen zentral ausführen.

Ereigniswarteschlangen mit Auslösern verwenden

Sie können die Ereigniswarteschlangen mit Auslösern konfigurieren, so dass die Ereignisnachricht, die beim Generieren eines Ereignisses in die Ereigniswarteschlange eingereiht wird, eine (benutzerdefinierte) Überwachungsanwendung startet. Diese Anwendung kann die Ereignisnachricht verarbeiten und entsprechende Aktionen ausführen. Zum Beispiel kann es bei bestimmten Ereignissen erforderlich sein, dass ein Bediener informiert wird, während andere Ereignisse möglicherweise eine Anwendung starten, die automatisch bestimmte Verwaltungs-Tasks ausführt.

Ereignisse aktivieren und inaktivieren

Sie können Ereignisse aktivieren und inaktivieren, indem Sie die entsprechenden Werte für den WS-Manager, Warteschlangenattribute oder beides angeben, je nachdem, um welche Ereignisart es sich handelt. Verwenden Sie dazu entweder:

- MQSC-Befehle. Weitere Informationen finden Sie im Handbuch *MQSeries MQSC-Befehle*.
- PCF-Befehle für WS-Manager auf UNIX-Systemen, OpenVMS-Systemen und unter OS/2. Weitere Informationen finden Sie im Handbuch *MQSeries Programmable System Management*.
- MQAI-Befehle. Weitere Informationen finden Sie im Handbuch *MQSeries Administration Interface Programming Guide and Reference*.

Die Aktivierung eines Ereignisses ist von der Kategorie des Ereignisse abhängig:

- WS-Manager-Ereignisse werden über Attribute des WS-Managers aktiviert.
- Leistungsereignisse müssen als Ganzes auf dem WS-Manager aktiviert werden, andernfalls können keine Leistungsereignisse eintreten. Sie können dann die einzelnen Leistungsereignisse aktivieren, indem Sie das entsprechende Warteschlangenattribut setzen. Darüber hinaus müssen Sie die Bedingungen angeben, die das Ereignis verursachen, z. B. einen Wert für die maximale Warteschlangenlänge.
- Kanalereignisse treten automatisch ein, müssen also nicht aktiviert werden. Wenn Sie Kanalereignisse nicht überwachen wollen, können Sie PUT-Aufrufe für die Kanalereigniswarteschlange inaktivieren.

Ereignisnachrichten

Ereignisnachrichten enthalten Informationen zum Ursprung eines Ereignisses, einschließlich Ereignisart und Name der Anwendung, die das Ereignis verursacht hat; bei Leistungsereignissen wird außerdem eine kurze Statistik für die Warteschlange erstellt.

Das Format von Ereignisnachrichten ist dem von PCF-Antwortnachrichten ähnlich. Die Nachrichtendaten können von benutzerdefinierten Verwaltungsprogrammen mit Hilfe der im Handbuch *MQSeries Programmable System Management* beschriebenen Datenstrukturen aus den Nachrichten abgerufen werden.

Kapitel 10. Transaktionsunterstützung

Das Handbuch *MQSeries Application Programming Guide* enthält eine umfassende Einführung zum Thema dieses Kapitels. Deshalb enthält dieses Kapitel nur eine kurze Zusammenfassung.

Ein Anwendungsprogramm kann ein Gruppe von Aktualisierungen zu einer *Arbeitseinheit* zusammenfassen. Diese Aktualisierungen sind in der Regel logisch miteinander verbunden und müssen alle erfolgreich ausgeführt werden, damit die Datenintegrität erhalten bleibt. Wenn eine Aktualisierung erfolgreich ist, eine andere aber nicht, geht die Datenintegrität verloren.

Wurde eine Arbeitseinheit erfolgreich abgeschlossen, wird eine **COMMIT**-Operation ausgeführt, d. h., die Aktualisierungen werden festgeschrieben. Ab diesem Zeitpunkt sind alle Aktualisierungen, die in dieser Arbeitseinheit ausgeführt wurden, permanent bzw. nicht mehr rückgängig zu machen. Kann die Arbeitseinheit nicht erfolgreich abgeschlossen werden, werden alle Aktualisierungen *zurückgesetzt*. Der Prozess, von dem Arbeitseinheiten entweder festgeschrieben (COMMIT-Operation) oder zurückgesetzt (ROLLBACK-Operation) werden, heißt *Synchronisationspunktkoordinierung*.

Eine *lokale* Arbeitseinheit ist eine Arbeitseinheit, in der nur die Ressourcen des MQSeries-WS-Managers aktualisiert werden. In diesem Fall wird die Synchronisationspunktkoordinierung vom WS-Manager selbst mit Hilfe einer einphasigen Festschreibung bereitgestellt.

Eine *globale* Arbeitseinheit ist eine Arbeitseinheit, in der Ressourcen, die anderen Ressourcenmanagern zugeordnet sind, z. B. XA-kompatible Datenbanken, aktualisiert werden. In diesem Fall muss eine zweiphasige Festschreibung verwendet werden, und die Arbeitseinheit kann vom WS-Manager selbst koordiniert werden.

Zusammengefasst heißt dies, dass WS-Manager-Ressourcen als Teil von lokalen oder globalen Arbeitseinheiten aktualisiert werden können:

Lokale Arbeitseinheit
: Verwenden Sie lokale Arbeitseinheiten, wenn nur die Ressourcen des MQSeries-WS-Managers aktualisiert werden sollen. Aktualisierungen werden mit Hilfe des Verbs MQCMIT festgeschrieben oder mit Hilfe des Verbs MQBACK zurückgesetzt.

Globale Arbeitseinheit
: Verwenden Sie globale Arbeitseinheiten, wenn die Aktualisierungen auch XA-kompatible Datenbankmanager betreffen. In diesem Fall kann die Koordinierung intern oder extern vom WS-Manager erfolgen.

  WS-Manager-Koordinierung
  : Globale Arbeitseinheiten werden mit dem Verb MQBEGIN gestartet und mit MQCMIT festgeschrieben bzw. mit MQBACK zurückgesetzt. Es wird eine zweiphasige Festschreibung verwendet, wobei zuerst alle XA-kompatiblen Ressourcenmanager wie Oracle aufgefordert werden, die Festschreibung vorzubereiten. Erst wenn sich alle erfolgreich vorbereitet haben, werden sie aufgefordert, die Festschreibung durchzuführen. Wenn einer der Ressourcenmanager

meldet, dass seine Vorbereitung für die Festschreibung nicht erfolgreich war, werden alle aufgefordert, eine Zurücksetzung durchzuführen.

Externe Koordinierung
In diesem Fall wird die Koordinierung von einem XA-kompatiblen Transaktionsmanager, z. B. IBM CICS, Transarc Encina oder BEA Tuxedo, ausgeführt. Arbeitseinheiten werden unter der Aufsicht des Transaktionsmanagers gestartet und festgeschrieben. Die Verben MQBEGIN, MQCMIT und MQBACK stehen nicht zur Verfügung.

Dieses Kapitel beschreibt, wie Unterstützung für globale Arbeitseinheiten aktiviert wird (Unterstützung für lokale Arbeitseinheiten muss nicht extra aktiviert werden).

Es enthält die folgenden Abschnitte:

Datenbankkoordinierung

Wenn der WS-Manager globale Arbeitseinheiten selbst koordiniert, ist es möglich, Datenbankaktualisierungen in MQ-Arbeitseinheiten zu integrieren. Das heißt, es kann eine gemischte MQI-/SQL-Anwendung geschrieben werden, und die Verben MQCMIT und MQBACK können zum Festschreiben oder Zurücksetzen der Änderungen an Warteschlangen und Datenbanken gleichzeitig verwendet werden.

Der WS-Manager erreicht dies durch ein zweiphasiges COMMIT-Protokoll. Wenn eine Arbeitseinheit festgeschrieben werden soll, fragt der WS-Manager zunächst alle beteiligten Datenbankmanager, ob sie auf die Festschreibung ihrer Aktualisierungen vorbereitet sind. Nur wenn alle Beteiligten, einschließlich des WS-Managers selbst, ihre Vorbereitungen für die Festschreibung getroffen haben, werden alle Warteschlangen- und Datenbankaktualisierungen festgeschrieben. Wenn nur ein Beteiligter seine Aktualisierungen nicht vorbereiten kann, wird die Arbeitseinheit stattdessen zurückgesetzt.

Für den Fall, dass der WS-Manager während der COMMIT-Operation die Verbindung mit einem der Datenbankmanager verliert, wird eine vollständige Wiederherstellung unterstützt. Wenn ein Datenbankmanager plötzlich nicht mehr verfügbar ist, während sein Status noch unbestätigt ist, d. h., er wurde zur Vorbereitung aufgefordert, hat jedoch noch keine Festschreibungs- oder Zurücksetzungsentscheidung empfangen, speichert der WS-Manager das Ergebnis der Arbeitseinheit so lange, bis es erfolgreich zugestellt werden kann. Entsprechend bleiben beim Beenden des WS-Managers die Ergebnisse noch nicht abgeschlossener COMMIT-Operationen über den Neustart des WS-Managers hinaus erhalten.

Mit dem MQI-Verb MQBEGIN müssen Arbeitseinheiten bezeichnet werden, die auch Datenbankaktualisierungen enthalten. Das Handbuch *MQSeries Application Programming Guide* enthält Beispielprogramme, mit denen MQSeries- und Datenbankaktualisierungen in derselben Arbeitseinheit ausgeführt werden.

Der WS-Manager kommuniziert mit dem Datenbankmanager über die XA-Schnittstelle, so wie im Dokument *X/Open Distributed Transaction Processing: The XA Specification* (ISBN 1 872630 24 3) beschrieben. Das heißt, der WS-Manager kann

mit Datenbankmanagern kommunizieren, die ebenfalls diesen Standard unterstützen. Solche Datenbankmanager werden als *XA-kompatible* Datenbankmanager bezeichnet.

Tabelle 5 zeigt, welche XA-kompatiblen Datenbankmanager von den MQSeries-Produkten der Version 5.1 unterstützt werden.

Tabelle 5. XA-kompatible relationale Datenbanken

MQSeries-Produkt	DB2	Oracle	Sybase
MQSeries for AIX	Ja	Ja	Ja
MQSeries for HP-UX	Ja	Ja	Nein
MQSeries for Sun Solaris	Ja	Ja	Ja
MQSeries for Compaq OpenVMS Alpha	Nein	Ja	Nein
MQSeries for Windows NT and Windows 2000	Ja	Nein	Ja

Einschränkungen

Für die Unterstützung der Datenbankkoordination gelten folgende Einschränkungen:

- Die Möglichkeit zum Koordinieren von Datenbankaktualisierungen innerhalb von MQSeries-Arbeitseinheiten wird in einer MQI-Client-Anwendung **nicht** unterstützt.
- Die MQI- und Datenbankaktualisierungen müssen auf derselben WS-Manager-Servermaschine erfolgen.
- Der Datenbankserver kann sich auf einer anderen Maschine als der WS-Manager-Server befinden. In diesem Fall muss über eine XA-kompatible Client-Einrichtung, die vom Datenbankmanager selbst zur Verfügung gestellt wird, auf die Datenbank zugegriffen werden.
- Auch wenn der WS-Manager selbst XA-kompatibel ist, ist es nicht möglich, einen anderen WS-Manager als Teilnehmer an globalen Arbeitseinheiten zu konfigurieren. Dies liegt daran, dass zu einem Zeitpunkt immer nur eine Verbindung unterstützt werden kann.

Datenbankverbindungen

Eine Anwendung, die eine Standardverbindung mit dem WS-Manager herstellt, wird einem Thread in einem separaten Agenten des lokalen WS-Managers zugeordnet. Wenn die Anwendung MQBEGIN ausgibt, müssen sowohl die Anwendung als auch der Agent Verbindungen mit den Datenbanken herstellen, die von der Arbeitseinheit betroffen sein werden. Die Datenbankverbindungen werden aufrechterhalten, solange die Anwendung mit dem WS-Manager verbunden ist. Dies ist dann besonders zu berücksichtigen, wenn die Datenbank nur eine begrenzte Zahl von Benutzern oder Verbindungen unterstützt.

Eine Möglichkeit, die Anzahl der Verbindungen zu verringern, besteht darin, dass die Anwendung den Aufruf MQCONNX verwendet, um eine FASTPATH-Bindung anzufordern. In diesem Fall werden die Anwendung und der Agent des lokalen WS-Managers zu ein und demselben Prozess und können infolgedessen eine einzige Datenbankverbindung gemeinsam benutzen. Bevor Sie diese Lösung wählen, lesen Sie im Handbuch *MQSeries Application Programming Guide* nach. Es enthält eine Liste mit Einschränkungen für FASTPATH-Anwendungen.

Datenbankmanager konfigurieren

Bevor ein Datenbankmanager an globalen Arbeitseinheiten, die vom WS-Manager koordiniert werden, teilnehmen kann, müssen Sie eine Reihe von Tasks ausführen:

1. Erstellen Sie eine *XA-Switch-Ladedatei*[1] für den Datenbankmanager.
2. Definieren Sie den Datenbankmanager in der Konfigurationsdatei des WS-Managers (`qm.ini`).

   In der Datei `qm.ini` müssen verschiedene Einträge definiert werden, darunter der Name der Switch-Ladedatei.

Switch-Ladedateien erstellen

Anweisungen zum Erstellen von Switch-Ladedateien für die unterstützten Datenbankmanager finden Sie unter „Oracle-Switch-Ladedatei erstellen" auf Seite 130.

Weitere Informationen zum Installationsverfahren finden Sie in Ihrem MQSeries-Installationshandbuch.

Die Beispielquellenmodule, die zum Erzeugen der Switch-Ladedateien verwendet werden, enthalten alle eine einzige Funktion mit dem Namen MQStart. Beim Laden der Switch-Ladedatei ruft der WS-Manager diese Funktion auf und erhält als Rückmeldung die Adresse einer Struktur, die als XA-Switch bezeichnet wird. Die Switch-Ladedatei wird mit einer vom Datenbankmanager bereitgestellten Bibliothek verknüpft, über die MQSeries den Datenbankmanager aufrufen kann.

Die Beispielquellenmodule zum Erstellen der Switch-Ladedateien für Oracle befinden sich in 'oraswit.c'.

Datenbankmanager definieren

Nachdem Sie eine Switch-Ladedatei für Ihren Datenbankmanager erstellt haben, müssen Sie deren Standort Ihrem WS-Manager mitteilen. Dies geschieht in der Zeilengruppe `XAResourceManager` in der Datei `qm.ini` des WS-Managers.

Sie müssen für jeden Datenbankmanager, den Ihr WS-Manager koordinieren soll, eine `XAResourceManager`-Zeilengruppe hinzufügen.

Die Zeilengruppe `XAResourceManager` enthält folgende Attribute:

Name=Name

Dies ist eine benutzerdefinierte Zeichenfolge, die das Datenbankmanagerexemplar identifiziert.

Der Name muss angegeben werden und darf maximal 31 Zeichen lang sein. Er muss eindeutig sein. Es kann sich dabei einfach um den Namen des Datenbankmanagers handeln, er kann jedoch außerdem zum Beispiel den Namen der zu aktualisierenden Datenbank einschließen, um auch in komplexeren Konfigurationen seine Eindeutigkeit zu gewährleisten.

Der ausgewählte Name sollte aussagekräftig sein, weil der WS-Manager ihn sowohl in Nachrichten als auch in Ausgaben des Befehls **dspmqtrn** für Verweise auf dieses Datenbankmanagerexemplar verwendet.

Nachdem Sie einmal einen Namen festgelegt haben, **lassen Sie dieses Attribut unverändert**. Informationen zum Ändern von Konfigurationsdaten finden Sie unter „Konfigurationsdaten ändern" auf Seite 139.

1. Eine XA-Switch-Ladedatei ist ein dynamisch geladenes Objekt, das die Kommunikation zwischen dem WS-Manager und dem Datenbankmanager ermöglicht.

SwitchFile=Name
Dies ist der vollständig qualifizierte Name der XA-Switch-Ladedatei des Datenbankmanagers. Dieses Attribut muss angegeben werden.

XAOpenString=Zeichenfolge
Die Daten dieser Zeichenfolge werden in Aufrufen an den xa_open-Eingangspunkt des Datenbankmanagers übergeben. Das Format dieser Zeichenfolge ist vom jeweiligen Datenbankmanager abhängig, sie sollte jedoch in der Regel den Namen der zu aktualisierenden Datenbank angeben.

Dieses Attribut ist optional; wird es nicht angegeben, wird eine Zeichenfolge mit Leerzeichen angenommen.

XACloseString=Zeichenfolge
Die Daten dieser Zeichenfolge werden in Aufrufen an den xa_close-Eingangspunkt des Datenbankmanagers übergeben. Das Format dieser Zeichenfolge ist vom jeweiligen Datenbankmanager abhängig.

Dieses Attribut ist optional; wird es nicht angegeben, wird eine Zeichenfolge mit Leerzeichen angenommen.

ThreadOfControl=THREAD | PROCESS
Als Wert für *ThreadOfControl* kann THREAD oder PROCESS angegeben werden. Der WS-Manager verwendet diesen Wert für serielle Anordnungen.

Wenn der Datenbankmanager 'Thread-sicher' ist, kann für *ThreadOfControl* der Wert THREAD angegeben werden, und der WS-Manager kann den Datenbankmanager über mehrere Threads gleichzeitig aufrufen.

Ist der Datenbankmanager nicht Thread-sicher, muss für *ThreadOfControl* der Wert PROCESS angegeben werden. Der WS-Manager serialisiert alle Aufrufe an den Datenbankmanager, so dass aus einem einzelnen Prozess heraus immer nur ein Aufruf erfolgt.

Eine ausführliche Beschreibung dieser Attribute finden Sie unter „Die Zeilengruppe XAResourceManager" auf Seite 190.

Unter „Oracle-Konfiguration" finden Sie weitere Informationen zu den speziellen Tasks, die Sie ausführen müssen, um MQSeries mit den einzelnen unterstützten Datenbankmanagern zu konfigurieren.

Oracle-Konfiguration

Sie müssen folgende Tasks ausführen:

- Oracle-Version überprüfen und Programmkorrekturen anwenden, sofern noch nicht geschehen
- Einstellungen von Umgebungsvariablen überprüfen
- Oracle XA-Unterstützung aktivieren
- Oracle-Switch-Ladedatei erstellen
- Konfigurationsdaten in Zeilengruppe XAResourceManager in der Datei `qm.ini` hinzufügen
- Oracle-Konfigurationsparameter bei Bedarf ändern

Mindestens erforderliche Oracle-Version

Die mindestens erforderliche Oracle-Version unter OpenVMS ist 8.1.6.

Einstellungen von Umgebungsvariablen überprüfen

Stellen Sie sicher, dass die Oracle-Umgebungsvariablen für WS-Manager-Prozesse sowie in den Anwendungsprozessen gesetzt sind. Insbesondere die folgenden Umgebungsvariablen sollten gesetzt werden, **bevor** der WS-Manager gestartet wird:

ORACLE_HOME
Dies ist das Oracle-Ausgangsverzeichnis.

ORACLE_SID
Dies ist die verwendete Oracle-System-ID.

Oracle XA-Unterstützung aktivieren

Sie müssen sicherstellen, dass die Oracle XA-Unterstützung aktiviert ist. Vor allen muss eine gemeinsam benutzte Oracle-Bibliothek erstellt worden sein. Dies geschieht während der Installation der Oracle XA-Bibliothek.

Bei der Installation von Oracle8 wird die Bibliothek automatisch erstellt. Wenn Sie die Bibliothek wiederherstellen müssen, schauen Sie in der Veröffentlichung *Oracle8 Administrator's Reference* für Ihre Plattform nach.

Oracle-Switch-Ladedatei erstellen

Der einfachste Weg zum Erstellen der Oracle-Switch-Ladedatei besteht darin, die Beispieldatei zu verwenden. Der Quellcode zum Erstellen der Oracle-Switch-Ladedatei wird in Abb. 11 gezeigt.

```
#include <cmqc.h>
#include "xa.h"

extern struct xa_switch_t xaosw;

struct xa_switch_t * MQENTRY MQStart(void)
{
   return(&xaosw);
}
```

Abbildung 11. Quellcode für Oracle-Switch-Ladedatei oraswit.c

Die in der include-Anweisung angegebene Header-Datei xa.h wird von MQSeries in demselben Verzeichnis wie oraswit.c bereitgestellt.

Oracle-Switch-Ladedatei auf OpenVMS-Systemen erstellen

Um die Oracle-Switch-Ladedatei auf OpenVMS-Systemen zu erstellen, muss die Datei oraswit.c kompiliert und mit der Oracle-Client-Bibliothek ″oraclient_v816.exe″ verknüpft werden.

1. Erstellen Sie das Verzeichnis, in dem die Oracle-Switch-Ladedatei oraswit erstellt werden soll.
2. Kopieren Sie die folgenden Dateien aus `mqs_examples:[xatm]` in dieses Verzeichnis:
   - xa.h
   - oraswit.c
3. Kompilieren Sie die kopierte Quellendatei (oraswit.c). Beispiel:

```
$ cc oraswit0.c
```

4. Generieren Sie die Switch-Ladedatei:

```
$ link/share oraswit0.obj, sys$input/options
ora_root:[util]oraclient_v816.exe/share
SYMBOL_VECTOR=(MQStart=PROCEDURE)
```

5. Die Prozedur MQStart wird zur Laufzeit dynamisch aus dem generierten Image geladen. Daher muss entweder ein logischer Name in der Tabelle mit logischen Namen des Systems definiert werden, der auf die generierte Datei ohne die Erweiterung `".exe"` zeigt, oder die Datei muss nach sys$share kopiert werden. Wenn der vollständige Pfadname der erstellten Datei zum Beispiel disk$a_device:[a_directory]oraswit0.exe lautet, verwenden Sie einen der beiden folgenden Befehle:

```
$ define/sys/exec oraswit0 disk$a_device:[a_directory]oraswit0
```

   oder

```
$ copy disk$a_device:[a_directory]oraswit0.exe sys$share:oraswit0.exe
```

XAResourceManager-Konfigurationsdaten für Oracle hinzufügen

Im nächsten Schritt wird die Konfigurationsdatei `qm.ini` des WS-Managers geändert, um Oracle als Teilnehmer an globalen Arbeitseinheiten zu definieren. Sie müssen eine Zeilengruppe XAResourceManager mit folgenden Attributen hinzufügen:

Name=Name
Dieses Attribut muss angegeben werden. Wählen Sie einen geeigneten Namen für diesen Teilnehmer aus. Sie können den Namen der zu aktualisierenden Datenbank einschließen.

SwitchFile=Name
Dieses Attribut muss angegeben werden. Dies ist der vollständig qualifizierte Name der Oracle-Switch-Ladedatei.

XAOpenString=Zeichenfolge
Die XAOpen-Zeichenfolge für Oracle hat folgendes Format:

```
Oracle_XA+Acc=P//|P/Benutzername/Kennwort
         +SesTm=Sitzungszeitlimit
         [+DB=Datenbankname]
         [+GPwd=P/Gruppenkennwort]
         [+LogDir=Protokollverzeichnis]
         [+MaxCur=MaxAnzOpenCursor]
         [+SqlNet=Verbindungszeichenfolge]
```

Dabei gilt:

Acc= Ist erforderlich und gibt Benutzerzugriffsinformationen an. **P//** bedeutet, dass keine expliziten Benutzer- oder Kennwortdaten übergeben werden und das Formular **ops$login** verwendet werden soll. **P**/Benutzername/Kennwort gibt eine gültige ORACLE-Benutzer-ID und das zugehörige Kennwort an.

SesTm=
Ist erforderlich und gibt die maximale Zeit an, die eine Transaktion inaktiv sein darf, bevor sie automatisch vom System gelöscht wird. Die Zeit wird in Sekunden angegeben.

DB= Gibt den Datenbanknamen an, wobei 'Datenbankname' für den Namen steht, den Oracle-Vorcompiler zum Identifizieren der Datenbank verwenden. Dieses Feld ist nur erforderlich, wenn Anwendungen den Datenbanknamen explizit angeben (d. h. eine AT-Klausel in ihren SQL-Anweisungen verwenden).

GPwd=
GPwd gibt das Serversicherheitskennwort an, wobei P/Gruppenkennwort für das Kennwort der Serversicherheitsgruppe steht. Serversicherheitsgruppen bieten eine zusätzliche Sicherheitsstufe, wenn verschiedene Anwendungen auf dasselbe ORACLE-Exemplar zugreifen. Der Standardwert ist eine ORACLE-definierte Serversicherheitsgruppe.

LogDir=
LogDir gibt das Verzeichnis auf einer lokalen Maschine an, in dem die Oracle XA-Bibliotheksfehlernachrichten und Trace-Daten protokolliert werden können. Wenn kein Wert angegeben wird, wird das aktuelle Verzeichnis verwendet. Stellen Sie sicher, dass der Benutzer MQM auf dieses Verzeichnis zugreifen kann.

MaxCur=
MaxCur gibt die Anzahl der Cursor an, die beim Öffnen der Datenbank angelegt werden sollen. Dies entspricht der Vorcompiler-Option 'maxopencursors'.

SqlNet=
SqlNet gibt die SQL*Net-Verbindungszeichenfolge an, die für die Anmeldung am System verwendet wird. Die Verbindungszeichenfolge kann eine SQL*Net V1-Zeichenfolge, eine SQL*Net V2-Zeichenfolge oder ein SQL*Net V2-Aliasname sein. Dieses Feld ist erforderlich, wenn Sie Oracle auf einer Maschine getrennt vom WS-Manager installieren.

Weitere Informationen finden Sie im *Oracle8 Server Application Developer's Guide* (Part Number A54642-01).

XACloseString=Zeichenfolge
Für Oracle muss keine XAClose-Zeichenfolge angegeben werden.

ThreadOfControl=THREAD | PROCESS
Dieser Parameter muss auf MQSeries for Compaq OpenVMS-Plattformen nicht angegeben werden.

Eine ausführliche Beschreibung dieser Attribute finden Sie unter „Die Zeilengruppe XAResourceManager" auf Seite 190.

In Abb. 12 hat die zu aktualisierende Datenbank den Namen MQBankDB. Es wird empfohlen, der XAOpen-Zeichenfolge einen Parameter `LogDir` hinzuzufügen, damit alle Fehler- und Trace-Protokolle in demselben Verzeichnis gespeichert werden. Es wird davon ausgegangen, dass die Oracle-Switch-Ladedatei nach ihrer Erstellung in das Verzeichnis sys$share kopiert wurde.

```
XAResourceManager:
  Name=Oracle MQBankDB
  SwitchFile=sys$share:oraswit0
  XAOpenString=Oracle_XA+Acc=P/jim/tiger+SesTm=35+LogDir=/tmp/ora.log+DB=MQBankDB
```

Abbildung 12. Beispiel eines XAResourceManager-Eintrags für Oracle

Oracle-Konfigurationsparameter ändern

Der WS-Manager und Benutzeranwendungen verwenden die in der XAOpen-Zeichenfolge angegebene Benutzer-ID, wenn sie eine Verbindung mit Oracle herstellen.

- **Datenbankberechtigungen** Die in der Open-Zeichenfolge angegebene Oracle-Benutzer-ID muss über die Zugriffsberechtigungen für die Ansicht DBA_PENDING_TRANSACTIONS verfügen. Die erforderliche Berechtigung kann mit dem folgenden Befehl erteilt werden, wobei `UserID` für die Benutzer-ID steht, der Zugriff erteilt wird.

```
grant select on DBA_PENDING_TRANSACTIONS to userID;
```

Weitere Informationen zur Sicherheit finden Sie in „Kapitel 7. MQSeries-Objekte schützen” auf Seite 81.

Verwaltungs-Tasks

Im Normalbetrieb ist nach Abschluss der Konfigurationsschritte der Verwaltungsaufwand sehr gering. Die Verwaltungsaufgabe wurde erleichtert, weil der WS-Manager tolerant reagiert, wenn Datenbankmanager nicht verfügbar sind. Dies heißt vor allen Folgendes:

- Der WS-Manager kann jederzeit gestartet werden, ohne dass vorher alle Datenbankmanager gestartet werden müssen.
- Der WS-Manager muss nicht gestoppt und erneut gestartet werden, wenn einer der Datenbankmanager nicht mehr verfügbar ist.

Dadurch können Sie den WS-Manager unabhängig von den Datenbankmanagern starten und stoppen und umgekehrt, sofern der Datenbankmanager dies unterstützt.

Immer wenn die Verbindung zwischen dem WS-Manager und einem Datenbankmanager verloren geht, müssen sie erneut synchronisiert werden, sobald beide wieder verfügbar sind.

Resynchronisation ist der Prozess, bei dem alle unbestätigten Arbeitseinheiten, die diese Datenbank betreffen, abgeschlossen werden. Dies geschieht im Allgemeinen automatisch, ohne dass ein Benutzereingriff erforderlich ist. Der WS-Manager fordert vom Datenbankmanager eine Liste der Arbeitseinheiten an, die sich aus Sicht des Datenbankmanagers in einem unbestätigten Status befinden. Anschließend weist er den Datenbankmanager an, die betreffenden unbestätigten Arbeitsein- heiten entweder festzuschreiben oder zurückzusetzen. Wenn der WS-Manager gestoppt wird, muss er beim Neustart eine Resynchronisation mit allen Datenbankmanagerexemplaren durchführen. Wenn ein einzelner Datenbankmanager ausfällt, muss der WS-Manager nur mit diesem Datenbankmanager eine Resynchronisation durchführen, sobald er feststellt, dass der betreffende Datenbankmanager wieder verfügbar ist.

Der WS-Manager versucht automatisch, die Verbindung mit einem nicht verfügbaren Datenbankmanager wiederherzustellen, wenn neue globale Arbeitseinheiten gestartet werden. Alternativ kann der Befehl **rsvmqtrn** ausgeführt werden, um alle unbestätigten Arbeitseinheiten explizit aufzulösen.

Unbestätigte Arbeitseinheiten

Ein Datenbankmanager bleibt möglicherweise mit unbestätigten Arbeitseinheiten zurück, wenn die Verbindung mit dem WS-Manager verloren geht, nachdem der Datenbankmanager die PREPARE-Anweisung erhalten hat. Solange der Datenbankmanager kein COMMIT- oder ROLLBACK-Ergebnis vom WS-Manager erhalten hat, muss er die für die Aktualisierungen eingerichteten Datenbanksperren aufrechterhalten.

Da andere Anwendungen wegen dieser Sperren keine Datenbanksätze aktualisieren, möglicherweise nicht einmal lesen können, muss so schnell wie möglich eine Resynchronisation durchgeführt werden.

Wenn Sie aus bestimmten Gründen nicht darauf warten können, bis der WS-Manager eine automatische Resynchronisation mit dem Datenbankmanager durchführt, können Sie von dem Datenbankmanager bereitgestellte Funktionen verwenden, um die Datenbankaktualisierungen manuell festzuschreiben oder zurückzusetzen. Dies wird als eine *heuristische* Entscheidung bezeichnet und sollte nur als letzte Möglichkeit in Frage kommen, weil die Gefahr besteht, dass die Datenintegrität verletzt wird. Gegebenenfalls müssen Sie die Datenbankaktualisierungen festschreiben, wenn alle anderen Teilnehmer sie zurücksetzen, oder umgekehrt. Besser ist es, nach dem Neustart der Datenbank den WS-Manager erneut zu starten oder den Befehl **rsvmqtrn** zu verwenden, um eine automatische Resynchronisation einzuleiten.

Den Befehl dspmqtrn verwenden

Solange ein Datenbankmanager nicht verfügbar ist, kann mit dem Befehl **dspmqtrn** der Status nicht abgeschlossener Arbeitseinheiten, die diese Datenbank betreffen, überprüft werden.

Wenn ein Datenbankmanager ausfällt, bevor mit der zweiphasigen Festschreibung begonnen wird, werden alle noch unbestätigten Arbeitseinheiten, an denen der Datenbankmanager beteiligt war, zurückgesetzt. Der WS-Manager selbst setzt seine noch unbestätigten Arbeitseinheiten bei seinem nächsten Neustart zurück.

Der **dspmqtrn** zeigt nur die Arbeitseinheiten an, in denen für einen oder mehrere Teilnehmer ein unbestätigter Status gilt, d. h., es wird auf die COMMIT- oder ROLLBACK-Operation des WS-Managers gewartet.

Für jede dieser Arbeitseinheiten wird der Status jedes einzelnen Teilnehmers angezeigt. Wenn die Arbeitseinheit die Ressourcen eines bestimmten Ressourcenmanagers nicht aktualisiert hat, wird dieser nicht angezeigt.

Von einer unbestätigten Arbeitseinheit wird dann gesprochen, wenn für einen Ressourcenmanager einer der folgenden Zustände zutrifft:

Prepared Vorbereitet - Der Ressourcenmanager ist darauf vorbereitet, seine Aktualisierungen festzuschreiben.

Committed Festgeschrieben - Der Ressourcenmanager hat seine Aktualisierungen festgeschrieben.

Rolled-back Zurückgesetzt - Der Ressourcenmanager hat seine Aktualisierungen zurückgesetzt.

Participated Teilgenommen - Der Ressourcenmanager ist Teilnehmer, hat seine Aktualisierungen aber nicht vorbereitet, festgeschrieben oder zurückgesetzt.

Beachten Sie, dass der WS-Manager den jeweiligen Status der einzelnen Teilnehmer bei einem Neustart nicht beibehält. Wenn der WS-Manager erneut gestartet wird, aber keine Verbindung mit einem Datenbankmanager herstellen kann, werden die unbestätigten Arbeitseinheiten, an denen dieser Datenbankmanager beteiligt war, beim Neustart nicht aufgelöst. In diesem Fall gilt für den Datenbankmanager der Status *Prepared*, bis eine Resynchronisation durchgeführt wird.

Wenn mit dem Befehl **dspmqtrn** eine unbestätigte Arbeitseinheit angezeigt wird, werden zunächst alle Ressourcenmanager, die teilnehmen können, angezeigt. Diesen wird eine eindeutige ID (*RMId*) zugeordnet, die anstelle des *Namens* der Ressourcenmanager verwendet wird, wenn über ihren Status in Bezug auf eine unbestätigte Arbeitseinheit berichtet wird.

Abb. 13 zeigt das Ergebnis des folgenden Befehls:

```
dspmqtrn -m MY_QMGR
```

```
AMQ7107: Ressourcenmanager 0 ist MQSeries.
AMQ7107: Ressourcenmanager 1 ist Oracle MQBankDB
AMQ7107: Ressourcenmanager 2 ist Oracle MQFeeDB

AMQ7056: Transaktionsnummer 0,1.
    XID: formatID 5067085, gtrid_length 12, bqual_length 4
         gtrid [3291A50600000201374657374]
         bqual [00000001]
AMQ7105: Festschreibung durch Ressourcenmanager 0 ist erfolgt.
AMQ7104: Vorbereitung durch Ressourcenmanager 1 ist erfolgt.
AMQ7104: Vorbereitung durch Ressourcenmanager 2 ist erfolgt.
```

Abbildung 13. Beispielausgabe des Befehls dspmqtrn

Die Ausgabe in Abb. 13 zeigt, dass dem WS-Manager drei Ressourcenmanager zugeordnet sind. Der erste ist der Ressourcenmanager `0`, bei dem es sich um den WS-Manager selbst handelt. Die anderen beiden Ressourcenmanagerexemplare sind die Oracle-Datenbanken `MQBankDB` und `MQFeeDB`.

In diesem Beispiel gibt es nur eine unbestätigte Arbeitseinheit. Für alle drei Ressourcenmanager wird eine Nachricht ausgegeben, d. h., für den WS-Manager und beide Oracle-Datenbanken in der Arbeitseinheit wurden Aktualisierungen durchgeführt.

Die Aktualisierungen für den WS-Manager und den Ressourcenmanager **0** wurden *festgeschrieben* (committed). Die Aktualisierungen für die Oracle-Datenbanken befinden sich im Status *Prepared*, d. h., die Verbindung mit Oracle wurde unterbrochen, bevor es aufgerufen wurde, die Aktualisierungen für die Datenbanken *MQBankDB* und *MQFeeDB* festzuschreiben.

Die unbestätigte Arbeitseinheit verfügt über eine externe ID (XID). Diese ID ordnet Oracle den Aktualisierungen zu.

Den Befehl rsvmqtrn verwenden

Die Ausgabe in Abb. 13 auf Seite 136 zeigt eine einzelne unbestätigte Arbeitseinheit, in der die COMMIT-Entscheidung noch an beide Oracle-Datenbanken zugestellt werden muss.

Um diese Arbeitseinheit abzuschließen, müssen der WS-Manager und Oracle eine Resynchronisation durchführen, sobald Oracle wieder verfügbar ist. Der WS-Manager nutzt den Start neuer Arbeitseinheiten als Gelegenheit für einen Versuch, die Verbindung mit Oracle wiederherzustellen. Alternativ können Sie den WS-Manager mit dem Befehl **rsvmqtrn** explizit anweisen, die Resynchronisation durchzuführen. Dies sollten Sie sofort nach dem Neustart von Oracle tun, damit alle Datenbanksperren, die auf Grund der unbestätigten Arbeitseinheit bestehen, so schnell wie möglich aufgehoben werden.

Verwenden Sie dazu die Option **-a**, mit der der WS-Manager angewiesen wird, alle unbestätigten Arbeitseinheiten aufzulösen. Im folgenden Beispiel wurde Oracle erneut gestartet, so dass der WS-Manager die unbestätigte Arbeitseinheit auflösen konnte:

```
rsvmqtrn -m MY_QMGR -a
```

Alle unbestätigten Transaktionen wurden aufgelöst.

Gemischte Ergebnisse und Fehler

Obwohl der WS-Manager ein zweiphasiges COMMIT-Protokoll verwendet, schließt dies nicht völlig aus, dass einige Arbeitseinheiten mit gemischten Ergebnissen abgeschlossen werden. Dies ist der Fall, wenn einige Teilnehmer ihre Aktualisierungen festschreiben und andere ihre Aktualisierungen zurücksetzen.

Werden Arbeitseinheiten mit einem gemischten Ergebnis abgeschlossen, hat dies ernste Auswirkungen, weil sich gemeinsam benutzte Ressourcen nicht mehr in einem konsistenten Status befinden.

Gemischte Ergebnisse entstehen hauptsächlich dann, wenn heuristische Entscheidungen für Arbeitseinheiten getroffen werden, statt dem WS-Manager selbst zu ermöglichen, unbestätigte Arbeitseinheiten aufzulösen.

Immer wenn der WS-Manager eine Beschädigung infolge einer heuristischen Entscheidung erkennt, erstellt er eine FFST-Information und dokumentiert den Fehler in seinen Fehlerprotokollen, und zwar mit einer der beiden folgenden Nachrichten:

- Ein Datenbankmanager hat eine Rücksetzung statt einer Festschreibung durchgeführt:

```
AMQ7606 Eine Transaktion wurde festgeschrieben, doch mindestens ein
        Ressourcenmanager hat eine Rücksetzung vorgenommen.
```

- Ein Datenbankmanager hat eine Festschreibung statt einer Rücksetzung durchgeführt:

```
AMQ7607 Eine Transaktion wurde zurückgesetzt, doch mindestens ein
        Ressourcenmanager hat eine Festschreibung vorgenommen.
```

Es werden weitere Nachrichten ausgegeben, in denen die Datenbanken identifiziert werden, die durch heuristische Entscheidungen beschädigt wurden. Danach ist es Ihre Aufgabe, die Fehler in den betroffenen Datenbanken lokal zu beheben, um die Konsistenz wiederherzustellen. Dabei handelt es sich um eine schwierige Prozedur, bei der Sie zunächst die Aktualisierung, die fälschlicherweise festgeschrieben oder zurückgesetzt wurde, isolieren und dann die Datenbankänderung manuell widerrufen oder erneut bestätigen müssen.

Beschädigungen infolge von Softwarefehlern treten seltener auf. Wenn Arbeitseinheiten auf diese Weise betroffen sind, wird ihre Transaktionsnummer mit der Nachricht AMQ7112 ausgegeben. Die Teilnehmer können sich in einem inkonsistenten Status befinden.

```
rsvmqtrn -m MY_QMGR

AMQ7107: Ressourcenmanager 0 ist MQSeries.
AMQ7107: Ressourcenmanager 1 ist Oracle MQBankDB
AMQ7107: Ressourcenmanager 2 ist Oracle MQFeeDB

AMQ7112: Transaktion Nummer 0,1 hat einen Fehler festgestellt.
    XID: formatID 5067085, gtrid_length 12, bqual_length 4
         gtrid [3291A5060000201374657374]
         bqual [00000001]
AMQ7105: Festschreibung durch Ressourcenmanager 0 ist erfolgt.
AMQ7104: Vorbereitung durch Ressourcenmanager 1 ist erfolgt.
AMQ7104: Zurücksetzung durch Ressourcenmanager 2 ist erfolgt.
```

Abbildung 14. Beispielausgabe des Befehls dspmqtrn für eine fehlerhafte Transaktion

Der WS-Manager versucht, solche Fehler erst bei seinem nächsten Neustart zu beheben. Für das Beispiel in Abb. 14 würde dies bedeuten, dass die Aktualisierungen für Ressourcenmanager **1**, die Datenbank MQBankDB, auch dann den Status *Prepared* beibehalten, wenn der Befehl **rsvmqtrn** zum Auflösen der Arbeitseinheit ausgeführt wurde.

Konfigurationsdaten ändern

Nachdem der WS-Manager erfolgreich zum Koordinieren von globalen Arbeitseinheiten gestartet wurde, sollten Sie sich davor hüten, Änderungen in den XAResourceManager-Zeilengruppen in der Datei `qm.ini` vorzunehmen.

Wenn Sie die Datei `qm.ini` ändern müssen, können Sie das jederzeit tun, die Änderungen werden jedoch erst wirksam, nachdem der WS-Manager erneut gestartet wurde. Wenn Sie zum Beispiel die XAOpen-Zeichenfolge, die an einen Datenbankmanager übergeben wird, ändern müssen, müssen Sie danach den WS-Manager erneut starten, damit die Änderung wirksam wird.

Beachten Sie, dass das Entfernen einer XAResourceManager-Zeilengruppe tatsächlich dazu führt, dass der WS-Manager keine Verbindung mehr mit dem betreffenden Datenbankmanager herstellen kann.

Ändern Sie ***niemals*** das Attribut *Name* in einer der XAResourceManager-Zeilengruppen. Dieses Attribut identifiziert das jeweilige Datenbankmanagerexemplar eindeutig gegenüber dem WS-Manager. Wenn diese eindeutige ID geändert wird, geht der WS-Manager davon aus, dass das Datenbankmanagerexemplar entfernt und ein völlig neues Exemplar hinzugefügt wurde. Der WS-Manager ordnet nicht abgeschlossene Arbeitseinheiten weiter dem alten *Namen* zu, wodurch die Datenbank möglicherweise in einem unbestätigten Status verbleibt.

Datenbankmanagerexemplare entfernen

Wenn Sie eine Datenbank oder einen Datenbankmanager endgültig aus Ihrer Konfiguration entfernen müssen, sollten Sie zunächst sicherstellen, dass sich die Datenbank nicht in einem unbestätigten Status befindet. Überprüfen Sie dies, bevor Sie den WS-Manager erneut starten. Die meisten Datenbankmanager stellen Befehle zur Verfügung, mit denen eine Liste unbestätigter Transaktionen angezeigt werden kann. Sind unbestätigte Transaktionen vorhanden, lassen Sie zunächst den WS-Manager eine Resynchronisation mit dem Datenbankmanager durchführen, bevor Sie dessen XAResourceManager-Zeilengruppe entfernen.

Wenn Sie diese Prozedur nicht ausführen, speichert der WS-Manager weiter alle unbestätigten Arbeitseinheiten, die diese Datenbank betreffen. Bei jedem Neustart des WS-Managers wird eine Warnung (AMQ7623) ausgegeben. Wenn Sie sicher sind, dass diese Datenbank nicht noch einmal für den WS-Manager konfiguriert werden muss, können Sie den WS-Manager mit der Option -r des Befehls **rsvmqtrn** anweisen, die Datenbank als Teilnehmer an den unbestätigten Transaktionen zu vergessen.

Anmerkung: Der WS-Manager vergisst Transaktionen nur dann endgültig, wenn eine Synchronisationspunktverarbeitung mit allen Teilnehmern abgeschlossen wurde.

In bestimmten Fällen kann es erforderlich sein, eine XAResourceManager-Zeilengruppe vorübergehend zu entfernen. Dies geschieht am besten dadurch, dass die Zeilengruppe 'auskommentiert' wird, so dass sie zu einem späteren Zeitpunkt leicht wieder aktiviert werden kann. Sie können diese Aktion zum Beispiel dann ausführen, wenn es bei jedem Versuch, eine Verbindung mit einer bestimmten Datenbank oder einem bestimmten Datenbankmanager herzustellen, zu einem Fehler kommt. Ein vorübergehendes Entfernen des fraglichen XAResourceManager-Eintrags ermöglicht es dem WS-Manager, globale Arbeitseinheiten für alle anderen Teilnehmer zu starten.

Im Folgenden ein Beispiel für eine 'auskommentierte' *XAResourceManager*-Zeilengruppe:

```
# Diese Datenbank wurde vorübergehend entfernt.
#XAResourceManager:
#  Name=Oracle MQBankDB
#  SwitchFile=sys$share:oraswit0
#  XAOpenString=MQBankDB
```

Abbildung 15. Auskommentierte XAResourceManager-Zeilengruppe

Kapitel 11. Wiederherstellung und Neustart

Ein Nachrichtenübertragungssystem stellt sicher, dass eingehende Nachrichten an ihre Zieladressen zugestellt werden. Dies bedeutet, dass das System eine Methode zum Verfolgen der Nachrichten im System und zum Wiederherstellen von Nachrichten im Falle eines Systemausfalls bereitstellen muss.

MQSeries stellt sicher, dass Nachrichten nicht verloren gehen, indem es die Aktivitäten der WS-Manager, die für den Empfang, die Übertragung und die Zustellung von Nachrichten zuständig sind, kontinuierlich aufzeichnet (protokolliert). Diese Protokolle werden für drei Arten der Wiederherstellung verwendet:

1. *Wiederherstellung nach Neustart*, wenn Sie MQSeries geplant stoppen.
2. *Wiederherstellung nach Absturz*, wenn MQSeries durch einen unerwarteten Fehler gestoppt wurde.
3. *Wiederherstellung über Datenträger*, um beschädigte Objekte wiederherzustellen.

In allen Fällen wird der WS-Manager in dem Status wiederhergestellt, in dem er sich befand, als er gestoppt wurde. Alle nicht abgeschlossenen Transaktionen werden zurückgesetzt, wobei aus den Warteschlangen alle Nachrichten entfernt werden, die zu dem Zeitpunkt, an dem der WS-Manager gestoppt wurde, nicht festgeschrieben waren. Alle permanenten Nachrichten werden wiederhergestellt; nicht permanente Nachrichten gehen dabei jedoch verloren.

Dieses Kapitel gibt eine Einführung in die Konzepte der Wiederherstellung und des Neustarts; anschließend werden Hinweise zur Fehlerbehandlung gegeben. Es enthält die folgenden Abschnitte:

Verlust von Nachrichten ausschließen (Protokollieren)

MQSeries zeichnet alle wichtigen Änderungen von Daten, die vom WS-Manager verwaltet werden, in einem Protokoll auf. Dazu gehören das Erstellen und Löschen von Objekten, alle Aktualisierungen von permanenten Nachrichten, der Status von Transaktionen, Änderungen von Objektattributen sowie Kanalaktivitäten. Das Protokoll enthält damit die Informationen, die Sie benötigen, um alle Aktualisierungen von Nachrichtenwarteschlangen wiederherzustellen, indem es:

- Datensätze mit WS-Manager-Änderungen bereitstellt,
- Datensätze mit Warteschlangenaktualisierungen für den Neustart bereitstellt,
- die Wiederherstellung von Daten nach einem Hardware- oder Softwarefehler ermöglicht.

Protokollformate

Ein MQSeries-Protokoll besteht aus zwei Komponenten:
1. einer oder mehreren Dateien mit Protokolldaten
2. einer Protokollsteuerdatei

Es gibt mehrere Protokolldateien mit den aufgezeichneten Daten. Sie können die Anzahl und Größe definieren (siehe Erläuterung unter „Protokollgröße berechnen" auf Seite 147) oder den Systemstandardwert (3 Dateien mit je 4 MB) übernehmen.

Beim Erstellen eines WS-Managers definieren Sie mit der Anzahl der Protokolldateien auch die Anzahl der anzulegenden *primären* Protokolldateien. Wenn Sie keine Anzahl angeben, wird der Standardwert verwendet. Wenn Sie den Protokollpfad nicht geändert haben, werden die Protokolldateien in folgendem Verzeichnis erstellt:

```
MQS_ROOT:[MQM.LOG.WS-Manager-Name.ACTIVE]
```

MQSeries verwendet zunächst die primären Protokolldateien. Wenn der Platz in den Dateien nicht mehr ausreicht, legt es *sekundäre* Protokolldateien an. Dies geschieht dynamisch; die Dateien werden wieder gelöscht, wenn weniger Speicherbereich für die Protokolle benötigt wird. Standardmäßig können maximal zwei sekundäre Protokolldateien angelegt werden, das entspricht einem zusätzlichen Plattenspeicherplatz von 8 MB. Auch der Standardwert für die Anzahl kann geändert werden (siehe „Kapitel 13. MQSeries konfigurieren" auf Seite 179).

Protokollsteuerdatei

Die Protokollsteuerdatei enthält die Informationen, die benötigt werden, um die Protokolldateien selbst zu überwachen: ihre Größe und Adresse, den Namen der nächsten verfügbaren Datei und so weiter.

Anmerkung: Stellen Sie sicher, dass die Protokolle, die Sie beim Starten eines WS-Managers erstellen, groß genug sind, um die von ihren Anwendungen erstellten Nachrichten aufzunehmen. Die Standardwerte für die Anzahl und Größe der Protokolle müssen an die jeweiligen Anforderungen angepasst werden. Wie Sie die Standardwerte ändern, wird auf Seite 147 beschrieben.

Protokolltypen

In MQSeries ist die Anzahl der für die Protokollierung verwendeten Dateien von der Dateigröße, der Anzahl der empfangenen Nachrichten und der Länge der Nachrichten abhängig. Es gibt zwei Methoden zum Aufzeichnen von WS-Manager-Aktivitäten: zyklische Protokollierung und lineare Protokollierung.

Zyklische Protokollierung

Verwenden Sie die zyklische Protokollierung für die Wiederherstellung nach einem Neustart, um Transaktionen, die gerade verarbeitet wurden, als das System gestoppt wurde, mit Hilfe des Protokolls zurückzusetzen.

Bei der zyklischen Protokollierung werden alle Daten für den Neustart in einem Protokolldateiring aufgezeichnet. Zunächst wird die erste Datei im Ring gefüllt, dann die zweite und so weiter, bis alle Dateien voll sind. Wenn dies der Fall ist, wird die Protokollierung in der ersten Datei innerhalb des Rings fortgesetzt. Dieses Verfahren wird angewendet, solange das Produkt aktiv ist, und hat den Vorteil, dass immer Protokolldateien zur Verfügung stehen.

Dies ist eine einfache Beschreibung der Funktionsweise der zyklischen Protokollierung. Dabei gibt es jedoch eine Schwierigkeit. Die Protokolle, die für einen Neustart des WS-Managers ohne Datenverlust erforderlich sind, werden so lange aufbewahrt, bis sie nicht mehr benötigt werden, um eine Wiederherstellung der WS-Manager-Daten sicherzustellen. Der Mechanismus zum Freigeben von Protokolldateien für eine erneute Verwendung wird unter „Prüfpunktverfahren – Vollständige Wiederherstellung sicherstellen" auf Seite 144 beschrieben. Die wichtige Information für Sie bis hierher ist, dass MQSeries sekundäre Protokolldateien verwendet, um die Protokollkapazität bei Bedarf zu erweitern.

Lineare Protokollierung

Verwenden Sie die lineare Protokollierung sowohl für die Wiederherstellung nach einem Neustart als auch für die Wiederherstellung über Datenträger bzw. Vorwärtswiederherstellung (erneute Erstellung verloren gegangener oder beschädigter Daten durch erneute Ausführung der protokollierten Aktionen).

Bei der linearen Protokollierung werden die Protokolldaten in einer fortlaufenden Folge von Dateien aufgezeichnet. Speicherbereich wird nicht mehrfach benutzt, so dass Sie jeden Datensatz, der seit der Erstellung des WS-Managers protokolliert wurde, jederzeit abrufen können.

Ist kein Plattenspeicherplatz mehr verfügbar, müssen Sie gegebenenfalls über eine Archivierung nachdenken. Die Verwaltung des Plattenspeicherplatzes für das Protokoll und damit die Frage, ob Speicherbereich bei Bedarf erneut benutzt oder erweitert wird, ist Teil der allgemeinen Verwaltungs-Tasks.

Die Anzahl der Protokolldateien, die bei der linearen Protokollierung erstellt werden, kann sehr groß werden, abhängig vom Umfang des Nachrichtenflusses und davon, wie lange der WS-Manager aktiv ist. Ein Teil der Dateien wird als aktiv bezeichnet. Aktive Dateien enthalten die Protokolleinträge, die für den Neustart des WS-Managers erforderlich sind. Die Anzahl der aktiven Protokolldateien entspricht in der Regel der Anzahl der primären Protokolldateien, die in den Konfigurationsdateien definiert ist. (Weitere Informationen zum Definieren der Anzahl finden Sie unter „Protokollgröße berechnen" auf Seite 147.)

Das Schlüsselereignis, über das gesteuert wird, ob eine Protokolldatei als aktiv bezeichnet wird oder nicht, ist ein *Prüfpunkt*. Ein MQSeries-Prüfpunkt ist eine Gruppe von Protokollsätzen mit Informationen, die einen erfolgreichen Neustart des WS-Managers ermöglichen. Alle vor einem Prüfpunkt aufgezeichneten Informationen werden nicht mehr für den Neustart des WS-Managers benötigt und können deshalb als inaktiv bezeichnet werden. (Weitere Informationen zu Prüfpunkten finden Sie unter „Prüfpunktverfahren – Vollständige Wiederherstellung sicherstellen" auf Seite 144).

Sie müssen entscheiden, wann inaktive Protokolldateien nicht mehr benötigt werden. Sie haben die Möglichkeit, sie zu archivieren, oder können Sie löschen, da sie für den Betrieb nicht mehr von Bedeutung sind. Weitere Informationen zur Disposition von Protokolldateien finden Sie unter „Protokolle verwalten" auf Seite 148.

Wenn in der zweiten oder einer späteren primären Protokolldatei ein neuer Prüfpunkt aufgezeichnet wird, wird die erste Datei inaktiv; gleichzeitig wird eine neue primäre Datei formatiert und am Ende des Pools mit den primären Dateien hinzugefügt, so dass die Anzahl der primären Dateien für die Protokollierung unverändert bleibt. Auf diese Weise bildet der Pool mit den primären Protokolldateien eine umlaufende Dateigruppe in einer sich ständig erweiternden Liste von

Protokolldateien. Auch hier der Hinweis, dass es Teil der allgemeinen Verwaltungs-Tasks ist, die inaktiven Dateien den betrieblichen Anforderungen entsprechend zu verwalten.

Obwohl sekundäre Protokolldateien für eine lineare Protokollierung definiert werden, werden sie im normalen Betrieb nicht verwendet. In einer Situation, in der es auf Grund besonders lang andauernder Transaktionen nicht möglich ist, eine Datei aus dem Pool aktiver Dateien freizugeben, weil sie möglicherweise noch für einen Neustart benötigt wird, werden sekundäre Dateien formatiert und dem Pool mit den aktiven Protokolldateien hinzugefügt.

Wenn die Anzahl verfügbarer sekundärer Dateien verbraucht ist, werden Anforderungen für weitere Operationen, die eine Protokollierung erfordern, zurückgewiesen, und an die Anwendung wird eine MQRC_RESOURCE_PROBLEM-Fehlernachricht zurückgegeben.

Beide Protokolltypen werden mit einem unerwarteten Netzausfall fertig, vorausgesetzt, es liegt kein Hardwarefehler vor.

Prüfpunktverfahren – Vollständige Wiederherstellung sicherstellen

Permanente Aktualisierungen von Nachrichtenwarteschlangen erfolgen in zwei Schritten. Zuerst werden die Datensätze mit der Aktualisierung in das Protokoll geschrieben, anschließend wird die Warteschlangendatei aktualisiert. Die Protokolldateien können daher aktueller sein als die Warteschlangendateien. Um sicherzustellen, dass die Verarbeitung nach einem Neustart an einem konsistenten Punkt beginnt, verwendet MQSeries Prüfpunkte. Ein Prüfpunkt ist ein Zeitpunkt, an dem der Datensatz, der im Protokoll beschrieben ist, mit dem Datensatz in der Warteschlange übereinstimmt. Der Prüfpunkt selbst besteht aus der Folge derjenigen Protokollsätze, die für den Neustart des WS-Managers benötigt werden; die Sätze können zum Beispiel den Status aller Transaktionen (d. h. Arbeitseinheiten) enthalten, die bei der Festlegung des Prüfpunktes aktiv waren.

Prüfpunkte werden automatisch von MQSeries generiert. Sie werden beim Starten des WS-Managers, beim Abschließen des WS-Managers, im Falle eines Speicherengpasses für die Protokollierung sowie nach jeweils 1000 protokollierten Operationen festgelegt. Da die Warteschlangen weitere Nachrichten bearbeiten, entsteht eine Inkonsistenz zwischen dem Prüfpunktsatz und dem aktuellen Status der Warteschlangen.

Wenn MQSeries gestartet wird, sucht es nach der Adresse des letzten Prüfpunktsatzes im Protokoll. Diese Information befindet sich in der Prüfpunktdatei, die am Ende jeder Prüfpunktfestlegung aktualisiert wird. Der Prüfpunktsatz gibt den letzten Zeitpunkt wieder, an dem Konsistenz zwischen dem Protokoll und den Daten bestand. Die Daten dieses Prüfpunktes werden verwendet, um die Warteschlangen so wiederherzustellen, wie sie zum Zeitpunkt der Prüfpunkterstellung existierten. Nachdem die Warteschlangen erneut erstellt wurden, werden die im Protokoll aufgezeichneten Aktionen der Reihe nach ausgeführt, um die Warteschlangen wieder in den Status zu versetzen, in dem sie sich vor dem Systemfehler oder Systemabschluss befanden.

MQSeries verwaltet interne Zeiger auf den Anfang und das Ende des Protokolls. Der Anfangszeiger wird auf den letzten Prüfpunkt gesetzt, an dem eine Konsistenz mit den wiederherzustellenden Nachrichtendaten besteht.

Prüfpunkte ermöglichen eine effektivere Wiederherstellung und die Steuerung der Wiederverwendung primärer und sekundärer Protokolldateien.

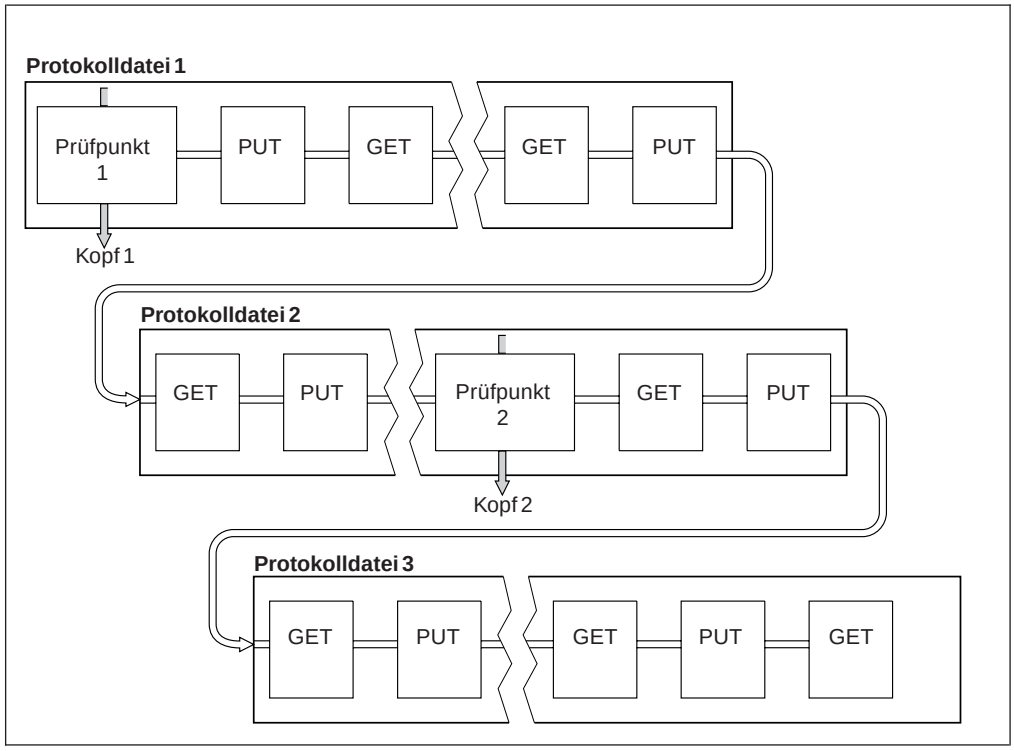


Abbildung 16. Prüfpunktverfahren. Der Übersichtlichkeit wegen sind nur Anfang und Ende der Protokolldateien dargestellt.

In Abb. 16 werden alle Datensätze vor dem letzten Prüfpunkt (Prüfpunkt 2) nicht mehr von MQSeries benötigt. Die Warteschlangen können aus den Informationen des Prüfpunktes und allen darauf folgenden Einträgen wiederhergestellt werden. Bei einer zyklischen Protokollierung können alle Dateien vor dem Prüfpunkt wiederverwendet werden, nachdem sie freigegeben wurden. Bei einer linearen Protokollierung muss auf die freigegebenen Dateien im normalen Betrieb nicht mehr zugegriffen werden, so dass sie inaktiv werden. Im Beispiel wird der Zeiger für den Warteschlangenanfang auf den letzten Prüfpunkt (Prüfpunkt 2) gesetzt, der damit zum neuen Warteschlangenanfang wird (Kopf 2). Protokolldatei 1 kann anschließend wiederverwendet werden.

Prüfpunktverfahren

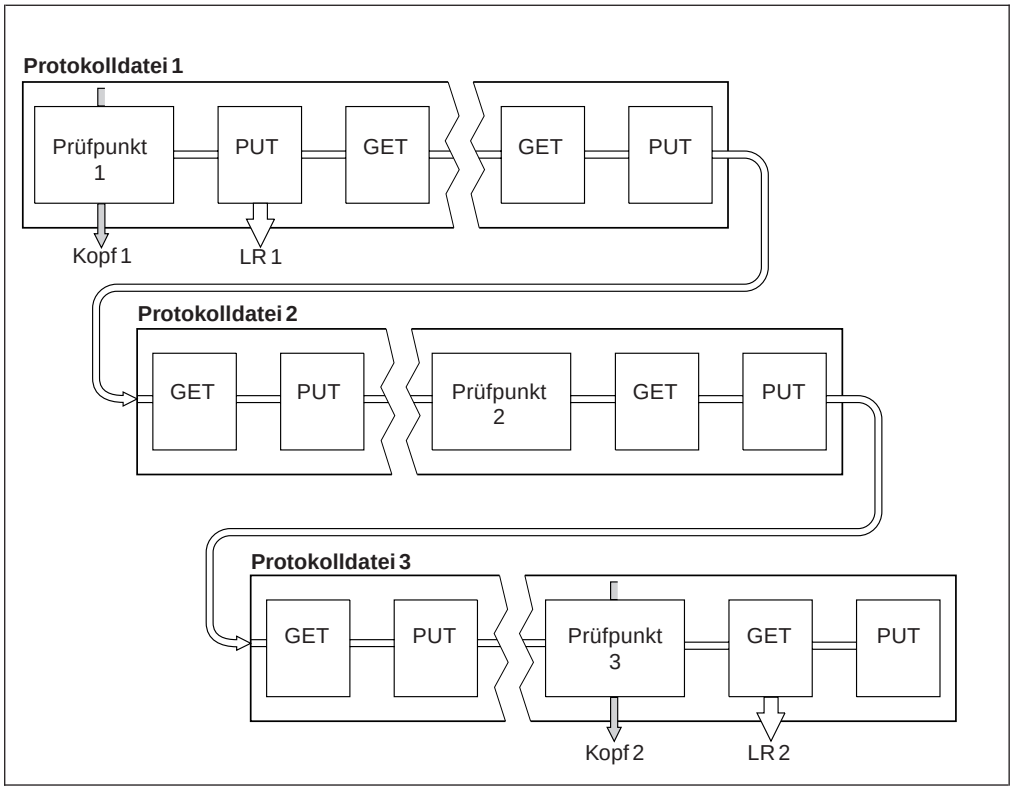


Abbildung 17. Prüfpunktverfahren bei einer lang andauernden Transaktion. Der Übersichtlichkeit wegen sind nur Anfang und Ende der Protokolldateien dargestellt.

Abb. 17 zeigt, wie sich eine lang andauernde Transaktion auf die Wiederverwendung von Protokolldateien auswirkt. In diesem Beispiel hat eine lang andauernde Transaktion einen Eintrag im Protokoll direkt hinter dem ersten Prüfpunkt verursacht (LR 1). Die Transaktion wird erst nach dem dritten Prüfpunkt abgeschlossen (LR 2). Alle Informationen ab LR 1 müssen bis zum Abschließen der Transaktion verfügbar bleiben, um gegebenenfalls eine Wiederherstellung der Transaktion zu ermöglichen.

Nachdem die lang andauernde Transaktion abgeschlossen wurde (bei LR 2), wird der Anfang des Protokolls auf Prüfpunkt 3 gesetzt, den letzten protokollierten Prüfpunkt. Die Dateien mit Protokollsätzen, die vor Prüfpunkt 3 (Kopf 2) liegen, werden nicht mehr benötigt. Bei einer zyklischen Protokollierung kann der Speicherbereich wiederverwendet werden.

Wenn die primären Protokolldateien voll sind, bevor die lang andauernde Transaktion abgeschlossen ist, werden sekundäre Protokolldateien verwendet, um das Risiko, das durch ein volles Protokoll entstehen kann, möglichst zu vermeiden.

Wenn der Protokollkopfsatz versetzt wird und Sie eine zyklische Protokollierung verwenden, können die primären Protokolldateien möglicherweise für eine Wiederverwendung ausgewählt werden, und die Protokollfunktion verwendet die erste verfügbare primäre Datei, nachdem sie die aktuelle Datei vollständig gefüllt hat. Wenn Sie aber eine lineare Protokollierung verwenden, wird der Protokollkopfsatz im Pool mit den aktiven Dateien weiter nach vorn versetzt und die erste Datei wird inaktiv. Eine neue primäre Datei wird formatiert und am Ende des Pools hinzugefügt; sie steht sofort für zukünftige Protokollierungsvorgänge zur Verfügung.

Protokollgröße berechnen

Nachdem Sie entschieden haben, ob der WS-Manager zyklische oder lineare Protokollierung verwenden soll, müssen Sie als Nächstes die Protokollgröße berechnen, die für den WS-Manager benötigt wird. Die Größe des Protokolls wird durch die folgende Protokollkonfiguration bestimmt:

LogFilePages
Die Größe jeder primären und sekundären Protokolldatei in Einheiten von 4-KB-Seiten.

LogPrimaryFiles
Die Anzahl der standardmäßig angelegten primären Protokolldateien.

LogSecondaryFiles
Die Anzahl der sekundären Protokolldateien, die erstellt werden können, wenn die primären Protokolldateien voll sind.

Tabelle 6 zeigt die Datenmenge, die der WS-Manager für verschiedene Operationen protokolliert. Die meisten der vom WS-Manager ausgeführten Operationen erfordern einen minimalen Speicherbereich für das Protokoll. Wenn jedoch eine permanente Nachricht in eine Warteschlange eingereiht wird, müssen die gesamten Nachrichtendaten in das Protokoll geschrieben werden, um eine Wiederherstellung der Nachricht zu ermöglichen. Daher ist die Protokollgröße in der Regel von der Anzahl und der Größe der permanenten Nachrichten abhängig, die der WS-Manager bearbeiten muss.

Tabelle 6. Gesamtgröße des Protokolls (alle Werte sind Näherungswerte)

Operation	Größe
Permanente Nachricht einreihen	750 Bytes + Nachrichtenlänge. Ist die Nachricht lang, wird sie in Segmente von 15.700 Bytes mit je 300 Bytes Verwaltungsdaten aufgeteilt.
Nachricht abrufen	260 Bytes
Synchronisationspunkt, COMMIT-Operation	750 Bytes
Synchronisationspunkt, ROLLBACK-Operation	1000 Bytes + 12 Bytes pro GET oder PUT, der zurückgesetzt wird
Objekt erstellen	1500 Bytes
Objekt löschen	300 Bytes
Attribute ändern	1024 Bytes
Datenträger-Image aufzeichnen	800 Bytes + Image. Das Image wird in Segmente von 15.700 Bytes mit je 300 Bytes Verwaltungsdaten aufgeteilt.
Prüfpunkt	750 Bytes + 200 Bytes für jede aktive Arbeitseinheit. Für jeden nicht festgeschriebenen PUT oder GET werden möglicherweise zusätzliche Daten protokolliert, die aus Leistungsgründen in einem Puffer zwischengespeichert werden.

Anmerkungen:

1. Die Anzahl der primären und sekundären Protokolldateien kann bei jedem Start des WS-Managers geändert werden.

2. Die Größe der Protokolldatei kann nicht geändert werden und muss festgelegt werden, **bevor** der WS-Manager erstellt wird.
3. Die Anzahl der primären Protokolldateien und die Größe der Protokolldatei bestimmen den Protokollspeicherbereich, der beim Erstellen des WS-Managers reserviert wird. Es wird empfohlen, diesen Speicherbereich so zu organisieren, dass er aus wenigen großen Protokolldateien und nicht aus vielen kleinen Protokolldateien besteht.
4. Die Gesamtzahl der primären und sekundären Protokolldateien darf 63 nicht überschreiten, wodurch die maximale Größe des Protokollspeicherbereichs, der dem WS-Manager für die Wiederherstellung nach einem Neustart zur Verfügung gestellt werden kann, im Falle von lang andauernden Transaktionen begrenzt wird. Diese Begrenzung gilt nicht für die Größe des Speicherbereichs, den der WS-Manager möglicherweise für die Wiederherstellung über Datenträger benötigt.
5. Bei der *zyklischen* Protokollierung wird der Speicherbereich für die primären Protokolldateien vom WS-Manager wiederverwendet. Das heißt, dass das Protokoll des WS-Managers kleiner sein kann als die Datenmenge, von der Sie bei der Berechnung der vom WS-Manager zu erstellenden Protokolldaten ausgegangen sind. Der WS-Manager legt, im Rahmen einer festgelegten Begrenzung, eine sekundäre Protokolldatei an, wenn eine Protokolldatei voll ist und die nächste primäre Protokolldatei im Dateiring nicht verfügbar ist.
6. Primäre Protokolldateien werden beim Festlegen eines Prüfpunktes zur Wiederverwendung freigegeben. Der WS-Manager berücksichtigt bei der Festlegung eines Prüfpunktes sowohl den Speicherbereich für primäre als auch für sekundäre Protokolldateien, weil die Größe des Protokollspeicherbereichs abnimmt.

   Wenn Sie nicht mehr primäre als sekundäre Protokolldateien definieren, legt der WS-Manager möglicherweise sekundäre Protokolldateien an, bevor ein Prüfpunkt festgelegt wird. Dadurch werden die primären Protokolldateien für eine Wiederverwendung frei.

Protokolle verwalten

Nach einer gewissen Zeit werden einige der erstellten Protokollsätze nicht mehr für einen Neustart des WS-Managers benötigt, und der WS-Manager fordert freien Speicherbereich in den Protokolldateien an. Dieser Vorgang ist für den Benutzer transparent, allerdings sehen Sie im Normalfall nicht, wenn die Größe des verwendeten Plattenspeicherplatzes verringert wird, weil der angelegte Speicherbereich sofort wiederverwendet wird.

Von den Protokollsätzen werden nur diejenigen, die seit dem Start des letzten vollständigen Prüfpunktes und die von aktiven Transaktionen erstellt wurden, für einen Neustart des WS-Managers benötigt. Das kann dazu führen, dass das Protokoll voll wird, wenn längere Zeit kein Prüfpunkt festgelegt wurde oder eine lang andauernde Transaktion schon vor sehr langer Zeit einen Protokollsatz erstellt hat. Der WS-Manager versucht Prüfpunkte so häufig festzulegen, dass das erste Problem vermieden wird.

Wenn das Protokoll durch Datensätze von einer lang andauernden Transaktion gefüllt ist, führen Versuche zum Schreiben von Protokollsätzen zu Fehlern, und einige MQI-Aufrufe geben MQRC_RESOURCE_PROBLEM zurück. (Für das Festschreiben oder Zurücksetzen nicht abgeschlossener Transaktionen wird Speicherbereich reserviert, so dass MQCMIT- oder MQBACK-Aufrufe nicht fehlschlagen sollten.)

Der WS-Manager setzt Transaktionen zurück, die zu viel Protokollspeicherbereich belegen. Eine Anwendung, deren Transaktion auf diese Weise zurückgesetzt wird, kann anschließend keine MQPUT- oder MQGET-Operation zwischen den Synchronisationspunkten derselben Transaktion ausführen. Bei einem Versuch, in diesem Status zwischen Synchronisationspunkten eine Nachricht einzureihen oder abzurufen, wird MQRC_BACKED_OUT zurückgegeben. Die Anwendung kann dann einen MQCMIT-Aufruf, der MQRC_BACKED_OUT zurückgibt, oder einen MQBACK-Aufruf ausgeben und eine neue Transaktion starten. Wenn die Transaktion, die zu viel Protokollspeicherbereich belegt hat, zurückgesetzt wird, wird der betreffende Protokollspeicherbereich freigegeben und der WS-Manager setzt seinen Betrieb normal fort.

Wenn das Protokoll voll ist, wird eine Nachricht (AMQ7463) ausgegeben. Zusätzlich wird die Nachricht AMQ7465 ausgegeben, wenn das Protokoll voll ist, weil eine lang andauernde Transaktion verhindert, das Speicherbereich freigegeben wird.

Schließlich wird die Nachricht AMQ7466 ausgegeben, wenn Datensätze schneller in das Protokoll geschrieben werden, als sie von asynchronen Systemverwaltungsprozessen bearbeitet werden können. Wenn diese Nachricht angezeigt wird, sollten Sie die Anzahl der Protokolldateien erhöhen oder die vom WS-Manager zu verarbeitende Datenmenge verringern.

Der Datenträger ist voll

Die Protokollierungskomponente des WS-Managers ist darauf vorbereitet, wenn ein Datenträger voll ist oder wenn Protokolldateien voll sind. Wenn der Datenträger mit den Protokolldateien voll ist, gibt der WS-Manager die Nachricht AMQ6708 aus, und es wird ein Fehlersatz erstellt.

Die Protokolldateien werden in ihrer maximalen Größe erstellt und nicht nach und nach erweitert, wenn Protokollsätze hineingeschrieben werden. Das heißt, MQSeries kann nur dann Plattenspeicherplatz fehlen, wenn eine neue Datei erstellt werden muss. Es steht daher immer Speicherbereich zur Verfügung, um einen Datensatz in das Protokoll zu schreiben. MQSeries weiß also immer, wie viel Speicherbereich in den vorhandenen Protokolldateien verfügbar ist, und verwaltet den Speicherbereich in den Dateien entsprechend.

Wenn das Laufwerk mit den Protokolldateien voll ist, haben Sie vielleicht die Möglichkeit, Plattenspeicherplatz freizugeben. Bei einer linearen Protokollierung enthält das Protokollverzeichnis möglicherweise einige inaktive Protokolldateien, die Sie auf ein anderes Laufwerk oder eine andere Einheit kopieren können. Wenn der Speicherbereich auch dann noch nicht ausreicht, überprüfen Sie, ob die Konfiguration des Protokolls in der Konfigurationsdatei des WS-Managers korrekt ist. Möglicherweise können Sie die Anzahl der primären oder sekundären Protokolldateien verringern, so dass das Protokoll in dem verfügbaren Speicherbereich Platz findet. Beachten Sie, dass die Größe der Protokolldateien für einen vorhandenen WS-Manager nicht geändert werden kann. Der WS-Manager geht davon aus, dass alle Protokolldateien dieselbe Größe haben.

Protokolldateien verwalten

Stellen Sie bei einer zyklischen Protokollierung sicher, dass genügend Speicherbereich für die Protokolldateien vorhanden ist. Dies geschieht beim Konfigurieren Ihres Systems (siehe „Die Zeilengruppe LogDefaults” auf Seite 184 und „Die Zeilengruppe Log” auf Seite 188). Die Größe des vom Protokoll verwendeten

Plattenspeicherplatzes, einschließlich des Speicherbereichs für sekundäre Dateien, die bei Bedarf erstellt werden müssen, ist durch die konfigurierte Größe des Datenträgers begrenzt.

Bei einer linearen Protokollierung werden durch das kontinuierliche Protokollieren von Daten weitere Protokolldateien hinzugefügt, so dass der belegte Plattenspeicherplatz ständig größer wird. Wenn besonders viele Daten protokolliert werden, wird der verfügbare Plattenspeicherplatz sehr schnell von neuen Protokolldateien belegt.

Nach einer gewissen Zeit werden ältere Protokolldateien nicht mehr für einen Neustart des WS-Managers oder für eine Wiederherstellung von beschädigten Objekten über Datenträger benötigt. Der WS-Manager gibt in regelmäßigen Abständen zwei Nachrichten aus, um anzuzeigen, welche Protokolldateien weiter benötigt werden:

- Die Nachricht AMQ7467 gibt den Namen der ältesten Protokolldatei an, die für einen Neustart des WS-Managers benötigt wird. Diese Protokolldatei und alle jüngeren Protokolldateien müssen beim Neustart des WS-Managers verfügbar sein.
- Die Nachricht AMQ7468 gibt den Namen der ältesten Protokolldatei an, die für eine Wiederherstellung über Datenträger benötigt wird.

Alle Protokolldateien, die älter als die genannten Dateien sind, müssen nicht mehr online verfügbar sein. Sie können sie auf einen Archivdatenträger, z. B. ein Band für eine Wiederherstellung nach einer Katastrophe, kopieren, und anschließend aus dem aktiven Protokollverzeichnis löschen. Alle Protokolldateien, die für eine Wiederherstellung über Datenträger, aber nicht für einen Neustart benötigt werden, können ebenfalls in ein Archiv ausgelagert werden.

Wenn eine benötigte Protokolldatei nicht gefunden wird, wird die Bedienernachricht AMQ6767 ausgegeben. Stellen Sie dem WS-Manager die Protokolldatei und alle darauf folgenden Protokolldateien zur Verfügung, und wiederholen Sie die Operation.

Anmerkung: Bei einer Wiederherstellung über Datenträger müssen alle benötigten Protokolldateien gleichzeitig im Protokolldateiverzeichnis verfügbar sein. Stellen Sie sicher, dass regelmäßig Datenträger-Images von allen Objekten erstellt werden, die möglicherweise wiederhergestellt werden müssen, um zu vermeiden, dass nicht genügend Plattenspeicherplatz für die benötigten Protokolldateien verfügbar ist.

Adresse der Protokolldatei

Denken Sie beim Auswählen eines Standortes für die Protokolldateien daran, dass der Betrieb ernsthaft beeinträchtigt wird, wenn MQSeries kein neues Protokoll formatieren kann, weil nicht genügend Plattenspeicherplatz vorhanden ist.

Stellen Sie bei einer zyklischen Protokollierung sicher, dass auf dem Laufwerk mindestens genügend Speicherbereich für die konfigurierten primären Protokolldateien vorhanden ist. Außerdem sollte genügend Speicherbereich für mindestens eine sekundäre Protokolldatei, die bei einer erforderlichen Erweiterung des Protokolls benötigt wird, verfügbar sein.

Bei einer linearen Protokollierung sollte erheblich mehr Speicherbereich vorhanden sein; der vom Protokoll belegte Speicherbereich wächst durch die kontinuierliche Protokollierung von Daten ständig an.

Idealerweise werden die Protokolldateien auf einem Plattenlaufwerk getrennt von den WS-Manager-Daten angelegt. Dies hat Vorteile in Bezug auf die Leistung. Es ist auch möglich, die Protokolldateien auf mehreren gespiegelten Plattenlaufwerken anzulegen. Dadurch besteht ein Schutz vor Fehlern auf dem Laufwerk mit dem Protokoll. Ohne Spiegelungsverfahren sind Sie gegebenenfalls gezwungen, bis zur letzten Sicherung Ihres MQSeries-Systems zurückzugehen.

Das Protokoll für eine Wiederherstellung verwenden

Es kann mehrere Ursachen für eine Beschädigung ihrer Daten geben. MQSeries for Compaq OpenVMS unterstützt Sie in folgenden Fällen bei der Wiederherstellung:

- Ein Datenobjekt ist beschädigt.
- Es ist ein Spannungsverlust im System aufgetreten.
- Es ist ein Übertragungsfehler aufgetreten.
- Ein Protokolldatenträger ist beschädigt.

Dieser Abschnitt beschreibt, wie die Protokolle nach diesen Fehlern für eine Wiederherstellung verwendet werden.

Wiederherstellung nach Fehlern

MQSeries kann sowohl nach Übertragungsfehlern als auch nach Spannungsverlusten wiederhergestellt werden. Darüber hinaus ist in einigen Fällen auch bei anderen Fehlerarten, z. B. nach dem versehentlichen Löschen einer Datei, eine Wiederherstellung möglich.

Im Falle eines Übertragungsfehlers bleiben Nachrichten in Warteschlangen erhalten, bis sie von einer empfangenden Anwendung entfernt werden. Wenn die Nachricht übertragen werden muss, bleibt sie so lange in der Übertragungswarteschlange, bis sie erfolgreich übertragen werden kann. Für eine Wiederherstellung nach einem Übertragungsfehler ist es normalerweise lediglich erforderlich, die Kanäle, von denen die fehlerhafte Verbindung verwendet wird, erneut zu starten.

Nach einem Spannungsverlust stellt MQSeries beim Neustart des WS-Managers die Warteschlangen in dem Status wieder her, in dem sie sich zum Zeitpunkt des Fehlers befanden. Dadurch wird sichergestellt, dass keine permanenten Nachrichten verloren gehen. Nicht permanente Nachrichten werden gelöscht; sie gehen verloren, wenn MQSeries gestoppt wird.

Es kann passieren, dass ein MQSeries-Objekt unbrauchbar wird, z. B. durch eine versehentliche Beschädigung. Sie müssen dann entweder das gesamte System oder einen Teil davon wiederherstellen. Die erforderliche Aktion ist davon abhängig, wann die Beschädigung entdeckt wurde, ob das ausgewählte Protokollverfahren eine Wiederherstellung über Datenträger unterstützt und welche Objekte beschädigt sind.

Wiederherstellung über Datenträger

Wiederherstellung über Datenträger bedeutet das erneute Erstellen von Objekten aus Informationen, die durch eine lineare Protokollierung aufgezeichnet wurden. Diese Art der Wiederherstellung ist bei einer zyklischen Protokollierung nicht möglich. Wenn zum Beispiel eine Objektdatei versehentlich gelöscht oder aus anderen Gründen unbrauchbar wird, kann sie mit Hilfe der Wiederherstellung über Datenträger erneut erstellt werden. Die Informationen in der Protokolldatei, die für die Wiederherstellung eines Objekts erforderlich sind, werden als *Datenträger-Image* bezeichnet. Datenträger-Images können manuell mit dem Befehl **rcdmqimg** und unter bestimmten Bedingungen auch automatisch aufgezeichnet werden.

Ein Datenträger-Image besteht aus einer Folge von Protokollsätzen, die das Image eines Objekts enthalten, aus dem das Objekt selbst erneut erstellt werden kann.

Der erste Protokollsatz, der zum erneuten Erstellen eines Objekts benötigt wird, ist der *Objektwiederherstellungssatz*; mit diesem Satz beginnt das zuletzt gespeicherte Datenträger-Image für das Objekt. Der Objektwiederherstellungssatz für jedes Objekt gehört zu den Informationen, die an einem Prüfpunkt aufgezeichnet werden.

Beim erneuten Erstellen eines Objekts aus seinem Datenträger-Image müssen außerdem alle Einträge aus den Protokollsätzen erneut ausgeführt werden, in denen Aktualisierungen für das Objekt beschrieben sind, die nach der letzten Image-Aufzeichnung ausgeführt wurden.

Nehmen wir zum Beispiel eine lokale Warteschlange, für das ein Image des Warteschlangenobjekts erstellt wurde, bevor eine permanente Nachricht in die Warteschlange eingereiht wird. Um das zuletzt gespeicherte Image des Objekts erneut zu erstellen, müssen zusätzlich zu dem Image selbst auch alle Protokolleinträge erneut ausgeführt werden, in denen das Einreihen der Nachricht in die Warteschlange aufgezeichnet wurde.

Nachdem ein Objekt erstellt wurde, enthalten die zugehörigen Protokollsätze genug Informationen, um das Objekt vollständig wiederherzustellen. Diese Datensätze bilden das erste Datenträger-Image des Objekts. Im Anschluss daran zeichnet der WS-Manager in folgenden Fällen automatisch Datenträger-Images auf:

- Bei jedem Systemabschluss werden Images aller Prozessobjekte und nicht lokalen Warteschlangen erstellt.
- Images von lokalen Warteschlangen werden erstellt, sobald die Warteschlange leer wird.

Datenträger-Images können auch manuell mit dem Befehl **rcdmqimg** aufgezeichnet werden (siehe Beschreibung unter „rcdmqimg (Datenträger-Image aufzeichnen)” auf Seite 290).

Datenträger-Images wiederherstellen

MQSeries stellt einige Objekte automatisch aus ihrem Datenträger-Image wieder her, wenn es feststellt, dass die Objekte zerstört oder beschädigt sind. Dies gilt insbesondere für Objekte, die während des normalen WS-Manager-Starts als beschädigt erkannt werden. Wenn eine Transaktion zum Zeitpunkt des letzten WS-Manager-Abschlusses noch nicht beendet war, werden auch alle betroffenen Warteschlangen automatisch wiederhergestellt, um den Startvorgang korrekt abzuschließen.

Andere Objekte müssen Sie mit dem Befehl **rcrmqobj** manuell wiederherstellen. Dieser Befehle führt die Datensätze im Protokoll noch einmal aus, um das MQSeries-Objekt erneut zu erstellen. Das Objekt wird aus seinem zuletzt gespeicherten Image, das im Protokoll gefunden wird, sowie aus allen gültigen Protokollereignissen, die zwischen dem Zeitpunkt der Speicherung des Images und dem der Ausgabe des Befehls zum erneuten Erstellen liegen, erneut erstellt. Wenn ein MQSeries-Objekt beschädigt wurde, gibt es nur zwei Möglichkeiten: Das Objekt wird gelöscht, oder es wird mit der oben beschriebenen Methode wiederhergestellt. Beachten Sie jedoch, dass nicht permanente Nachrichten auf diese Weise nicht wiederhergestellt werden können.

Weitere Informationen zum Befehl **rcrmqobj** finden Sie unter „rcrmqobj (Objekt erneut erstellen)” auf Seite 292.

Es ist wichtig, noch einmal daran zu erinnern, dass die Protokolldatei mit dem Objektwiederherstellungssatz und alle darauf folgenden Protokolldateien im Protokolldateiverzeichnis zur Verfügung stehen müssen, wenn Sie versuchen, ein Objekt auf diese Weise wiederherzustellen. Wenn eine erforderliche Datei nicht gefunden wird, wird die Bedienernachricht AMQ6767 ausgegeben und die Wiederherstellung schlägt fehl. Wenn Sie nicht regelmäßig Datenträger-Images der Objekte aufzeichnen, die Sie gegebenenfalls erneut erstellen möchten, kann die Situation eintreten, dass nicht mehr genügend Plattenspeicherplatz verfügbar ist, um die für das erneute Erstellen eines Objekts erforderlichen Protokolldateien zu speichern.

Beschädigte Objekte beim Start wiederherstellen

Wenn der WS-Manager während des Starts ein beschädigtes Objekt erkennt, ist die von ihm ausgeführte Aktion von der Objektart abhängig und davon, ob der WS-Manager so konfiguriert ist, dass er die Wiederherstellung über Datenträger unterstützt.

Wenn das WS-Manager-Objekt beschädigt ist, kann der WS-Manager nur gestartet werden, wenn er das Objekt wiederherstellen kann. Wurde der WS-Manager mit einer linearen Protokollierung konfiguriert, so dass er die Wiederherstellung über Datenträger unterstützt, versucht MQSeries automatisch, das MQSeries-Objekt aus seinen Datenträger-Images wiederherzustellen. Wenn das ausgewählte Protokollierungsverfahren die Wiederherstellung über Datenträger nicht unterstützt, können Sie entweder eine Sicherung des WS-Managers zurückschreiben oder den WS-Manager löschen.

Wenn beim Stoppen des WS-Managers Transaktionen aktiv waren, werden für ein erfolgreiches Starten des WS-Managers auch die lokalen Warteschlangen mit den permanenten, nicht festgeschriebenen Nachrichten, die innerhalb dieser Transaktionen ausgegeben oder abgerufen wurden, benötigt. Ist eine dieser lokalen Warteschlangen beschädigt und unterstützt der WS-Manager die Wiederherstellung über Datenträger, versucht er automatisch, sie aus ihren Datenträger-Images wiederherzustellen. Wenn eine der Warteschlangen nicht wiederhergestellt werden kann, kann MQSeries nicht gestartet werden.

Wenn während des Startvorgangs eines WS-Managers, der die Wiederherstellung über Datenträger nicht unterstützt, beschädigte lokale Warteschlangen mit nicht festgeschriebenen Nachrichten erkannt werden, werden die Warteschlangen als beschädigte Objekte markiert und die darin enthaltenen nicht festgeschriebenen Nachrichten ignoriert. Dies ist so, weil es auf einem solchen WS-Manager nicht möglich ist, eine Wiederherstellung von beschädigten Objekten über Datenträger auszuführen, und deshalb keine andere Möglichkeit besteht, als die Objekte zu löschen. Die Nachricht AMQ7472 mit einem Bericht über die Beschädigungen wird ausgegeben.

Beschädigte Objekte zu anderen Zeiten wiederherstellen

Eine Wiederherstellung von Objekten über Datenträger wird nur beim Start automatisch ausgeführt. Zu allen anderen Zeiten wird die Bedienernachricht AMQ7472 ausgegeben, sobald ein beschädigtes Objekt erkannt wird, und die meisten Operationen, die das Objekt betreffen, schlagen fehl. Wenn das WS-Manager-Objekt irgendwann nach dem Starten des WS-Managers beschädigt wird, führt der WS-Manager einen erzwungenen Abschluss aus. Nachdem ein Objekt beschädigt wurde, können Sie es löschen; oder Sie können versuchen, es mit dem Befehl **rcrmqobj** (siehe „rcrmqobj (Objekt erneut erstellen)” auf Seite 292 für weitere Informationen) aus seinem Datenträger-Image wiederzuerstellen, sofern der WS-Manager eine lineare Protokollierung verwendet.

MQSeries-Protokolldateien schützen

Entfernen Sie die Protokolldateien auf keinen Fall manuell, während ein MQSeries-WS-Manager aktiv ist. Wenn ein Benutzer versehentlich (oder vorsätzlich) die Protokolldateien löscht, die für den Neustart eines WS-Managers benötigt werden, gibt MQSeries keine Fehlernachrichten aus und setzt die Verarbeitung von Daten, einschließlich permanenter Nachrichten, fort. Der WS-Manager wird normal abgeschlossen, kann aber nicht erneut gestartet werden. Eine Wiederherstellung von Nachrichtenobjekten ist dann nicht mehr möglich.

Jeder Benutzer mit der Berechtigung, Protokolle eines aktiven WS-Managers zu entfernen, verfügt auch über die Berechtigung, andere wichtige WS-Manager-Ressourcen zu löschen (z. B. Berechtigungsdateien, Warteschlangendateien, den Objektkatalog und ausführbare MQSeries-Dateien). Er kann daher, sei es aus Unerfahrenheit oder möglicherweise sogar absichtlich, einen aktiven oder ruhenden WS-Manager auf eine Weise beschädigen, gegen die sich MQSeries selbst nicht schützen kann.

Prüfen Sie also genau, bevor Sie privilegierte Berechtigungen oder die Berechtigungs-ID MQM erteilen.

Sichern und Zurückschreiben

Im Normalfall werden Sie in regelmäßigen Abständen eine Sicherung Ihrer WS-Manager-Daten erstellen, um bei einer möglichen Beschädigung infolge von Hardwarefehlern geschützt zu sein. Da Nachrichtedaten jedoch oft sehr kurzlebig sind, verzichten Sie möglicherweise auf eine regelmäßige Sicherung.

MQSeries sichern

Um die Daten eines WS-Managers zu sichern, müssen Sie Folgendes tun:

1. Stellen Sie sicher, das der WS-Manager nicht aktiv ist.

   Wenn der WS-Manager aktiv ist, stoppen Sie ihn mit dem Befehl **endmqm**.

   Anmerkung: Wenn Sie versuchen, einen aktiven WS-Manager zu sichern, ist die Sicherung möglicherweise nicht konsistent, weil Aktualisierungen stattfinden können, während die Dateien kopiert werden.

2. Suchen Sie die Verzeichnisse, in denen der WS-Manager seine Daten und Protokolldateien ablegt.

   Informationen zu diesen Verzeichnissen finden Sie zum Beispiel in den Konfigurationsdateien. Weitere Informationen hierzu finden Sie in „Kapitel 13. MQSeries konfigurieren” auf Seite 179.

   Anmerkung: Sie werden möglicherweise die Namen, die in dem Verzeichnis angezeigt werden, nicht wiedererkennen. Dies liegt daran, dass die Namen umgewandelt wurden, um sicherzustellen, dass sie mit der Plattform, auf der Sie MQSeries einsetzen, kompatibel sind. Weitere Informationen zu Namensumwandlungen finden Sie unter „Erläuterungen zu MQSeries-Dateinamen” auf Seite 21.

3. Erstellen Sie Kopien von allen Daten- und Protokolldateiverzeichnissen des WS-Managers, einschließlich aller Unterverzeichnisse.

   Achten Sie darauf, wirklich alle Dateien zu erfassen, insbesondere die Protokollsteuerdatei und die Konfigurationsdateien. Einige der Verzeichnisse

sind möglicherweise leer, es sind jedoch alle erforderlich, um die Sicherung zu einem späteren Zeitpunkt zurückzuschreiben. Daher ist es ratsam, auch die leeren Verzeichnisse zu speichern.

4. Stellen Sie sicher, dass die Eigentumsrechte der Dateien erhalten bleiben. Verwenden Sie dazu den Befehl BACKUP mit dem Parameter /BY_OWNER.

MQSeries zurückschreiben

Um die Sicherung der Daten eines WS-Managers zurückzuschreiben, müssen Sie Folgendes tun:

1. Stellen Sie sicher, das der WS-Manager nicht aktiv ist.
2. Suchen Sie die Verzeichnisse, in denen der WS-Manager seine Daten und Protokolldateien ablegt. Diese Information finden Sie in der Konfigurationsdatei.
3. Löschen Sie den Inhalt der Verzeichnisse, in die Sie die gesicherten Daten kopieren wollen.
4. Kopieren Sie die gesicherten WS-Manager-Daten und Protokolldateien in die vorgesehenen Verzeichnisse.

Überprüfen Sie die erstellte Verzeichnisstruktur, um sicherzustellen, dass alle erforderlichen Verzeichnisse vorhanden sind.

Weitere Informationen zu MQSeries-Verzeichnissen und Unterverzeichnissen finden Sie in „Anhang C. Verzeichnisstruktur" auf Seite 329.

Stellen Sie sicher, dass sowohl eine Protokollsteuerdatei als auch die Protokolldateien vorhanden sind. Überprüfen Sie außerdem, ob die Konfigurationsdateien von MQSeries und des WS-Managers konsistent sind, damit MQSeries in den richtigen Verzeichnissen nach den zurückgeschriebenen Daten sucht.

Wenn die Daten korrekt gesichert und zurückgeschrieben wurden, kann der WS-Manager gestartet werden.

Anmerkung: Auch wenn sich die Daten und Protokolldateien des WS-Managers in unterschiedlichen Verzeichnissen befinden, sollten Sie die Verzeichnisse zur selben Zeit sichern bzw. zurückschreiben. Wenn die Daten und Protokolldateien des WS-Managers unterschiedlich alt sind, befindet sich der WS-Manager nicht in einem gültigen Status und kann voraussichtlich nicht gestartet werden. Wird er trotzdem gestartet, ist die Wahrscheinlichkeit groß, dass die Daten beschädigt sind.

Wiederherstellungsszenarios

Dieser Abschnitt behandelt eine Reihe möglicher Probleme und zeigt, wie sie behoben werden können.

Plattenlaufwerkfehler

Möglicherweise treten Probleme mit dem Plattenlaufwerk auf, auf dem sich die WS-Manager-Daten, -Protokolldateien oder beides befinden. Die Probleme können Datenverluste oder -beschädigungen einschließen. Die drei genannten Fälle unterscheiden sich nur durch den Teil der Daten, der erhalten bleibt (sofern überhaupt Daten erhalten bleiben).

In ***allen*** Fällen müssen Sie zuerst die Verzeichnisstruktur auf Beschädigungen überprüfen und gegebenenfalls reparieren. Wenn Sie WS-Manager-Daten verloren

haben, besteht die Gefahr, dass die Verzeichnisstruktur des WS-Managers beschädigt wurde. In diesem Fall müssen Sie die Verzeichnisstruktur manuell neu aufbauen, bevor Sie versuchen, den WS-Manager erneut zu starten. Nachdem Sie die Verzeichnisstrukturen überprüft haben, gibt es verschiedene Dinge, die Sie tun können, je nachdem, welchen Protokolltyp Sie verwenden.

- **Wenn die Beschädigungen der Verzeichnisstruktur sehr stark sind oder das Protokoll beschädigt ist**, entfernen Sie alle alten Dateien unter dem Verzeichnis mit dem `Namen des WS-Managers`, einschließlich der Konfigurationsdateien, des Protokolls und des WS-Manager-Verzeichnisses, schreiben Sie die letzte Sicherung zurück, und versuchen Sie dann, den WS-Manager erneut zu starten.
- **Wenn Sie eine lineare Protokollierung mit Wiederherstellung über Datenträger verwenden**, stellen Sie sicher, dass die Verzeichnisstruktur intakt ist, und versuchen Sie, den WS-Manager erneut zu starten. Wenn sich der WS-Manager nicht erneut starten lässt, schreiben Sie eine Sicherung zurück. Wenn der WS-Manager gestartet wird, überprüfen Sie mit Hilfe von MQSC-Befehlen, ob andere Objekte beschädigt wurden. Stellen Sie die Objekte zum Beispiel mit dem folgenden **rcrmqobj**-Befehl wieder her:

```
rcrmqobj -m QMgrName -t * *
```

  Dabei steht `QMgrName` für den WS-Manager, der wiederhergestellt wird. -t * * gibt an, dass alle Objekte und alle Objektarten wiederhergestellt werden. Wenn nur ein oder zwei Objekte als beschädigt gemeldet werden, können Sie an dieser Stelle auch direkt den Namen und die Art dieser Objekte eingeben.

  Anmerkung: Diese Befehle können nicht für Kanäle verwendet werden.
- **Wenn Sie eine lineare Protokollierung mit Wiederherstellung über Datenträger verwenden und das Protokoll unbeschädigt ist**, können Sie möglicherweise eine Sicherung der WS-Manager-Daten zurückschreiben und dabei die vorhandenen Protokolldateien und die Protokollsteuerdatei unverändert lassen. Beim Start des WS-Managers werden die Änderungen aus dem Protokoll angewendet, um den WS-Manager in den Status zurückzuversetzen, in dem er sich zum Zeitpunkt des Fehlers befand.

  Diese Methode basiert auf zwei Voraussetzungen. Erstens ist es wichtig, dass die Prüfpunktdatei als Teil der WS-Manager-Daten zurückgeschrieben wird. Diese Datei enthält die Informationen, die festlegen, wie viele Daten aus dem Protokoll angewendet werden müssen, um einen konsistenten Status des WS-Managers herzustellen.

  Zweitens müssen die älteste Protokolldatei, die zum Zeitpunkt der Sicherung zum Starten des WS-Managers benötigt wurde, und alle darauf folgenden Protokolldateien im Protokolldateiverzeichnis zur Verfügung stehen.

  Wenn dies nicht möglich ist, müssen Sie eine Sicherung sowohl der WS-Manager-Daten als auch des Protokolls zurückschreiben, die beide zur selben Zeit erstellt worden sein müssen.
- **Wenn Sie eine zyklische Protokollierung oder eine lineare Protokollierung ohne Wiederherstellung über Datenträger verwenden**, müssen Sie die letzte Sicherung des WS-Managers zurückschreiben, über die Sie verfügen. Nachdem Sie die Sicherung zurückgeschrieben haben, starten Sie den WS-Manager erneut und überprüfen Sie wie oben beschrieben, ob Objekte beschädigt sind. Da die Wiederherstellung über Datenträger nicht zur Verfügung steht, müssen Sie jedoch einen anderen Weg finden, um die beschädigten Objekte erneut zu erstellen.

WS-Manager-Objekt ist beschädigt

Wenn das WS-Manager-Objekt während des normalen Betriebs als beschädigt gemeldet wird, führt der WS-Manager einen erzwungenen Abschluss aus. Abhängig vom verwendeten Protokolltyp gibt es zwei Möglichkeiten, das Objekt unter diesen Umständen wiederherzustellen:

- **Wenn Sie die lineare Protokollierung verwenden (und nur dann)**, löschen Sie manuell die Datei mit dem beschädigten Objekt, und starten Sie den WS-Manager erneut. Die Wiederherstellung des beschädigten Objekts erfolgt automatisch.
- **Wenn Sie die zyklische oder lineare Protokollierung verwenden**, schreiben Sie die letzte Sicherung der WS-Manager-Daten und des Protokolls zurück, und starten Sie den WS-Manager erneut.

Einzelnes Objekt ist beschädigt

Wenn ein einzelnes Objekt während des normalen Betriebs als beschädigt gemeldet wird, gibt es abhängig vom verwendeten Protokolltyp zwei Möglichkeiten, das Objekt wiederherzustellen:

- **Wenn Sie die lineare Protokollierung verwenden**, erstellen Sie das Objekt erneut aus seinem Datenträger-Image.
- **Wenn Sie die zyklische Protokollierung verwenden**, schreiben Sie die letzte Sicherung der WS-Manager-Daten und des Protokolls zurück, und starten Sie den WS-Manager erneut.

Fehler bei automatischer Wiederherstellung eines Datenträgerobjekts

Wenn eine lokale Warteschlange, die für einen WS-Manager-Start mit einer linearen Protokollierung benötigt wird, beschädigt ist und die automatische Wiederherstellung des Datenträgerobjekts fehlschlägt, schreiben Sie die letzte Sicherung der WS-Manager-Daten und des Protokolls zurück, und starten Sie den WS-Manager erneut.

Protokollauszug mit dem Befehl dmpmqlog erstellen

Mit dem Befehl **dmpmqlog** können Sie einen Auszug des WS-Manager-Protokolls erstellen. Standardmäßig enthält ein Auszug alle aktiven Protokollsätze, d. h., er beginnt mit dem Kopfsatz des Protokolls. Dies entspricht in der Regel dem Anfang des letzten festgelegten Prüfpunktes.

Ein Protokollauszug kann nur erstellt werden, wenn der WS-Manager nicht aktiv ist. Da der WS-Manager beim Abschluss einen Prüfpunkt festlegt, enthält der aktive Teil des Protokolls normalerweise nur wenige Protokollsätze. Sie können den Befehl **dmpmqlog** jedoch anweisen, mehr Protokollsätze in den Auszug aufzunehmen, indem Sie eine der folgenden Optionen angeben, um den Anfangspunkt für den Auszug zu ändern:

- Die einfachste Option ist, am *Basispunkt* des Protokolls mit dem Auszug zu beginnen. Der Basispunkt des Protokolls ist der erste Protokollsatz in der Protokolldatei, die den Kopfsatz des Protokolls enthält. Die Menge der zusätzlichen Daten im Auszug ist davon abhängig, an welcher Position in der Protokolldatei sich der Kopfsatz des Protokolls befindet. Befindet er sich nahe am Anfang der Protokolldatei, enthält der Auszug nur wenige zusätzliche Daten. Befindet er sich nahe am Ende der Protokolldatei, enthält der Auszug wesentlich mehr Daten.

- Eine andere Option ermöglicht es, einen einzelnen Protokollsatz als Anfangspunkt für den Auszug anzugeben. Jeder Protokollsatz wird durch eine eindeutige *Protokollfolgenummer (Log Sequence Number, LSN)* identifiziert. Bei einer zyklischen Protokollierung kann der Protokollsatz, der als Anfangspunkt dient, nicht vor dem Basispunkt des Protokolls liegen; diese Einschränkung gilt nicht für die lineare Protokollierung. Inaktive Protokolldateien müssen gegebenenfalls wieder aktiviert werden, bevor der Befehl ausgeführt wird. Bei dieser Option muss eine gültige Protokollfolgenummer als Anfangspunkt angegeben werden. Diese muss einer vorhergehenden **dmpmqlog**-Ausgabe entnommen werden.

  Bei einer linearen Protokollierung können Sie zum Beispiel den Wert von `nextlsn` aus der letzten **dmpmqlog**-Ausgabe angeben. Der Wert von `nextlsn` wird im `Log File Header` angezeigt und gibt die Protokollfolgenummer des nächsten zu schreibenden Protokollsatzes an. Dieser Wert kann daher als Anfangspunkt zum Formatieren aller Protokollsätze, die seit dem letzten Protokollauszug geschrieben wurden, verwendet werden.
- Die dritte Option gilt nur für die lineare Protokollierung. Die Protokollauszugsfunktion kann angewiesen werden, in einem vorgegebenen Protokolldatei-Speicherbereich mit dem Formatieren von Protokollsätzen zu beginnen. In diesem Fall erwartet die Protokollauszugsfunktion, dass sich diese Protokolldatei und alle darauf folgenden Dateien in demselben Verzeichnis wie die aktiven Protokolldateien befinden. Diese Option gilt nicht für die zyklische Protokollierung, weil die Protokollauszugsfunktion in diesem Fall nicht auf Protokollsätze zugreifen kann, die vor dem Basispunkt des Protokolls liegen.

Die Ausgabe des Befehls **dmpmqlog** besteht aus dem `Log File Header` und einer Reihe formatierter Protokollsätze. Der WS-Manager verwendet verschiedene Protokollsätze, um Änderungen an seinen Daten aufzuzeichnen.

Einige der formatierten Informationen werden nur intern verwendet. Die folgende Liste enthält die nützlichsten Protokolldatensätze:

Log File Header
Jedes Protokoll verfügt über einen Protokolldatei-Header, der immer als erstes vom Befehl **dmpmqlog** formatiert wird. Er enthält die folgenden Felder:

logactive	Die Anzahl der primären Protokollspeicherbereiche.
loginactive	Die Anzahl der sekundären Protokollspeicherbereiche.
logsize	Die Anzahl der 4-KB-Seiten pro Bereich.
baselsn	Die erste Protokollfolgenummer im Protokollspeicherbereich, der den Kopfsatz des Protokolls enthält.
nextlsn	Die Protokollfolgenummer des nächsten zu schreibenden Protokollsatzes.
headlsn	Die Protokollfolgenummer des Protokollsatzes, der dem Kopfsatz des Protokolls entspricht.
tailsn	Die Protokollfolgenummer, die den Endpunkt des Protokolls identifiziert.
hflag1	Gibt an, ob das Protokoll zyklisch (CIRCULAR) oder linear (LOG RETAIN) ist.
HeadExtentID	Der Protokollspeicherbereich, in dem sich der Kopfsatz des Protokolls befindet.

Log Record Header
Jeder Protokollsatz im Protokoll verfügt über einen unveränderbaren Header mit folgenden Informationen:

LSN Die Protokollfolgenummer (Log Sequence Number).

LogRecdType Die Art des Protokollsatzes.

XTranid Die Transaktions-ID, die diesem Protokollsatz zugeordnet ist (falls vorhanden).

Enthält *TranType* den Wert MQI, handelt es sich ausschließlich um eine MQ-Transaktion. Enthält *TranType* den Wert XA, sind andere Ressourcenmanager beteiligt. Alle Aktualisierungen innerhalb derselben Arbeitseinheit haben dieselbe *XTranid*.

QueueName Die Warteschlange, die diesem Protokollsatz zugeordnet ist (falls vorhanden).

Qid Die eindeutige interne ID der Warteschlange.

PrevLSN Die Protokollfolgenummer des vorherigen Protokollsatzes innerhalb derselben Transaktion (falls vorhanden).

Start Queue Manager
Hier wird protokolliert, dass der WS-Manager gestartet wurde.

StartDate Das Datum, an dem der WS-Manager gestartet wurde.

StartTime Die Uhrzeit, zu der der WS-Manager gestartet wurde.

Stop Queue Manager
Hier wird protokolliert, dass der WS-Manager gestoppt wurde.

StopDate Das Datum, an dem der WS-Manager gestoppt wurde.

StopTime Die Uhrzeit, zu der der WS-Manager gestoppt wurde.

ForceFlag Die Abschlussart, die verwendet wurde.

Start Checkpoint
Dies bezeichnet den Anfang eines WS-Manager-Prüfpunktes.

End Checkpoint
Dies bezeichnet das Ende eines WS-Manager-Prüfpunktes.

ChkPtLSN Die Protokollfolgenummer des Protokollsatzes, der diesen Prüfpunkt gestartet hat.

Put Message
Hier wird das Einreihen einer permanenten Nachricht in die Warteschlange protokolliert. Wurde die Nachricht zwischen Synchronisationspunkten eingereiht, enthält der Protokollsatz-Header eine *XTranid* ungleich NULL. Der Rest des Datensatz enthält Folgendes:

SpcIndex Eine ID für die Nachricht in der Warteschlange. Über diese ID kann der zugehörige MQGET-Aufruf zugeordnet werden, mit dem diese Nachricht aus der Warteschlange abgerufen wurde. In diesem Fall schließt sich ein *Get Message*-Protokollsatz mit denselben Werten für `QueueName` und `SpcIndex` an. An dieser Stelle kann die ID in `SpcIndex` erneut verwendet werden, um eine Folgenachricht in die Warteschlange einzureihen.

Data Der hexadezimale Auszug dieses Protokollsatzes enthält ver-

schiedene interne Daten gefolgt vom Nachrichtendeskriptor (mit der Strukturkennung 'MD') und den Nachrichtendaten selbst.

Put Part
Permanente Nachrichten, die zu lang für einen einzigen Protokollsatz sind, werden als ein einziger *Put Message*-Datensatz gefolgt von mehreren *Put Part*-Protokollsätzen protokolliert.

Data Hier werden die Nachrichtendaten an der Stelle fortgesetzt, an der sie am Ende des vorhergehenden Protokollsatzes unterbrochen wurden.

Get Message
Nur GET-Aufrufe für permanente Nachrichten werden protokolliert. Wenn die Nachricht zwischen Synchronisationspunkten abgerufen wird, enthält der Protokollsatz-Header eine *XTranid* ungleich NULL. Der Rest des Datensatz enthält Folgendes:

SpcIndex Identifiziert die Nachricht, die aus der Warteschlange abgerufen wurde. Der jüngste *Put Message*-Protokollsatz mit denselben Werten in *QueueName* und *SpcIndex* identifiziert die abgerufene Nachricht.

QPriority Die Priorität der Nachricht, die aus der Warteschlange abgerufen wurde.

Start Transaction
Zeigt den Start einer neuen Transaktion an. Enthält TranType den Wert MQI, handelt es sich ausschließlich um eine MQ-Transaktion. Enthält TranType den Wert XA, sind andere Ressourcenmanager beteiligt. Alle Aktualisierungen durch diese Transaktion haben dieselbe *XTranid*.

Prepare Transaction
Gibt an, dass der WS-Manager darauf vorbereitet ist, die Aktualisierungen, die der angegebenen *XTranid* zugeordnet sind, festzuschreiben. Dieser Protokollsatz wird als Teil einer zweiphasigen Festschreibung, an der andere Ressourcenmanager beteiligt sind, geschrieben.

Commit Transaction
Gibt an, dass der WS-Manager alle Aktualisierungen durch die Transaktion festgeschrieben hat.

Rollback Transaction
Dieser Protokollsatz gibt an, dass der WS-Manager eine Transaktion zurücksetzen will.

End Transaction
Dieser Protokollsatz bezeichnet das Ende einer zurückgesetzten Transaktion.

Transaction Table
Dieser Datensatz wird während der Synchronisationspunktverarbeitung geschrieben. Er enthält den Status aller Transaktionen, die permanente Aktualisierungen ausgeführt haben. Für jede Transaktion werden die folgenden Informationen aufgezeichnet:

XTranid Transaktions-ID

FirstLSN Protokollfolgenummer des ersten Protokollsatzes einer Transaktion.

LastLSN Protokollfolgenummer des letzten Protokollsatzes einer Transaktion.

Transaction Participants
Dieser Protokollsatz wird vom XA-Transaktionsmanager des WS-Managers geschrieben. Er enthält die externen Ressourcenmanager, die an einer Transaktion beteiligt sind. Für jeden Teilnehmer werden die folgenden Informationen aufgezeichnet:

RMName Der Name des Ressourcenmanagers.

RMId Die ID des Ressourcenmanagers. Sie ist auch in nachfolgenden *Transaction Prepared*-Protokollsätzen enthalten, in denen globale Transaktionen, an denen der Ressourcenmanager beteiligt ist, aufgezeichnet werden.

SwitchFile Die Switch-Ladedatei für diesen Ressourcenmanager.

XAOpenString Die XAClose-Zeichenfolge für diesen Ressourcenmanager.

XACloseString Die XAClose-Zeichenfolge für diesen Ressourcenmanager.

Transaction Prepared
Dieser Protokollsatz wird vom XA-Transaktionsmanager des WS-Managers geschrieben. Er zeigt an, dass die angegebene globale Transaktion erfolgreich vorbereitet wurde. Alle teilnehmenden Ressourcenmanager werden angewiesen, die Transaktion festzuschreiben. Die *RMId* jedes einzelnen Ressourcenmanagers wird im Protokollsatz aufgezeichnet. Wenn der WS-Manager selbst an der Transaktion teilnimmt, ist ein *Participant Entry*-Protokollsatz mit einer *RMID* gleich NULL vorhanden.

Transaction Forget
Dieser Protokollsatz wird vom XA-Transaktionsmanager des WS-Managers geschrieben. Er folgt dem *Transaction Prepared*-Protokollsatz, wenn die COMMIT-Entscheidung an alle Teilnehmer zugestellt wurde.

Purge Queue
Hier wird protokolliert, dass alle Nachrichten in einer Warteschlange gelöscht wurden, z. B. mit dem Befehl RUNMQSC CLEAR.

Queue Attributes
Hier wird die Initialisierung oder Änderung der Attribute einer Warteschlange protokolliert.

Create Object
Protokolliert die Erstellung eines MQSeries-Objekts.

ObjName Der Name des erstellten Objekts.

UserId Die Benutzer-ID, unter der das Objekt erstellt wurde.

Delete Object
Protokolliert das Löschen eines MQSeries-Objekts.

ObjName Des Name des gelöschten Objekts.

Abb. 18 auf Seite 162 zeigt eine Beispielausgabe des Befehls **dmpmqlog**. Der Auszug, der mit der Protokollfolgenummer eines angegebenen Protokollsatzes beginnt, wurde mit folgendem Befehl erstellt:

```
dmpmqlog -m "testqm" -s 0:0:0:44162
```

Befehl dmpmqlog verwenden

```
AMQ7701: Befehl DMPMQLOG startet.
LOG FILE HEADER
***************

counter1  . . . : 23                  counter2  . . . : 23
FormatVersion . : 2                   logtype . . . . : 10
logactive . . . : 3                   loginactive . . : 2
logsize . . . . : 1024       pages
baselsn . . . . : <0:0:0:0>
nextlsn . . . . : <0:0:0:60864>
lowtranlsn  . . : <0:0:0:0>
minbufflsn  . . : <0:0:0:58120>
headlsn . . . . : <0:0:0:58120>
taillsn . . . . : <0:0:0:60863>
logfilepath . . : ""
hflag1  . . . . : 1
               -> CONSISTENT
               -> CIRCULAR
HeadExtentID  . : 1                   LastEID . . . . : 846249092
LogId . . . . . : 846249061           LastCommit  . . : 0
FirstArchNum  . : 4294967295          LastArchNum . . : 4294967295
nextArcFile . . : 4294967295          firstRecFile  . : 4294967295
firstDlteFile . : 4294967295          lastDeleteFile  : 4294967295
RecHeadFile . . : 4294967295          FileCount . . . : 3
frec_trunclsn . : <0:0:0:0>
frec_readlsn  . : <0:0:0:0>
frec_extnum . . : 0                   LastCId . . . . : 0
onlineBkupEnd . : 0                   softmax . . . . : 4194304


LOG RECORD - LSN <0:0:0:44162>
**********

HLG Header: lrecsize 212, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ALM Start Checkpoint (1025)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 192                 LogRecdOwnr . . : 1024   (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

No data for Start Checkpoint Record
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 1 von 13)

```
LOG RECORD - LSN <0:0:0:44374>
**********

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Transaction Table (773)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 200                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
TranCount . . . : 0

LOG RECORD - LSN <0:0:0:44594>
**********

HLG Header: lrecsize 1836, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : Transaction Participants (1537)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 1816                LogRecdOwnr . . : 1536   (T)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id. . . . . . . : TLPH
Version . . . . : 1                   Flags . . . . . : 3
Count . . . . . : 2

Participant Entry 0
RMName  . . . . : DB2 MQBankDB
RMId  . . . . . : 1
SwitchFile  . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString  . :
XACloseString . :

Participant Entry 1
RMName  . . . . : DB2 MQBankDB
RMId  . . . . . : 2
SwitchFile  . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString  . :
XACloseString . :
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 2 von 13)

```
LOG RECORD - LSN <0:0:0:46448>
**********

HLG Header: lrecsize 236, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ALM End Checkpoint (1026)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 216                 LogRecdOwnr . . : 1024   (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

ChkPtLSN  . . . : <0:0:0:44162>
OldestLSN . . . : <0:0:0:0>
MediaLSN  . . . : <0:0:0:0>


LOG RECORD - LSN <0:0:0:52262>
**********

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Start Transaction (769)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 200                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: MQI    TranNum{High 0, Low 1}
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
SoftLogLimit  . : 10000
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 3 von 13)

```
LOG RECORD - LSN <0:0:0:52482>
**********

HLG Header: lrecsize 730, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : AQM Put Message (257)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 710                 LogRecdOwnr . . : 256    (AQM)
XTranid . . . . : TranType: MQI    TranNum{High 0, Low 1}
QueueName . . . : Queue1
Qid . . . . . . : {Hash 196836031, Counter: 0}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:52262>

Version . . . . : 3
SpcIndex  . . . : 1
PrevLink.Locn . : 36                  PrevLink.Length : 8
PrevDataLink  . : {High 0, Low 2048}
Data.Locn . . . : 2048                Data.Length . . : 486
Data  . . . . . :
00000:  41 51 52 48 00 00 00 04 FF FF FF FF FF FF FF FF    AQRH............
00016:  00 00 00 00 00 00 00 00 00 00 00 01 00 01 01 C0    ...............&#192;
00032:  00 00 00 00 00 00 00 01 00 00 00 22 00 00 00 00    ...........".....
00048:  00 00 00 00 41 4D 51 20 74 65 73 74 71 6D 20 20    ....AMQ testqm
00064:  20 20 20 20 33 80 2D D2 00 00 10 13 00 00 00 00        3€-&#30;....
00080:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00096:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00112:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01    ................
00128:  00 00 00 00 00 00 00 22 00 00 00 00 00 00 00 00    .......".........
00144:  00 00 00 00 00 00 00 C9 2C B5 C0 25 FF FF FF FF    ...&#26;,&#181;&#192;%.
00160:  4D 44 20 20 00 00 00 01 00 00 00 00 00 00 00 08    MD  ............
00176:  00 00 00 00 00 00 01 11 00 00 03 33 20 20 20 20    ...........3
00192:  20 20 20 20 00 00 00 00 00 00 00 01 20 20 20 20        ........
00208:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00224:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00240:  20 20 20 20 20 20 20 20 20 20 20 74 65 73 74                test
00256:  71 6D 20 20 20 20 20 20 20 20 20 20 20 20 20 20    qm
00272:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00288:  20 20 20 20 20 20 20 20 20 20 20 73 62 6F 6C                sbol
00304:  61 6D 20 20 20 20 20 20 04 37 34 38 30 00 00 00    am      .7480...
00320:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00336:  00 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20    ........
00352:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00368:  20 20 20 20 20 20 20 20 00 00 00 06 75 74 7A 61            ....utza
00384:  70 69 20 20 20 20 20 20 20 20 20 20 20 20 20 20    pi
00400:  20 20 20 20 20 20 20 20 31 39 39 37 30 35 31 39            19970519
00416:  31 30 34 32 31 35 32 30 20 20 20 20 00 00 00 00    10421520    ....
00432:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00448:  50 65 72 73 69 73 74 65 6E 74 20 6D 65 73 73 61    Persistent messa
00464:  67 65 20 70 75 74 20 75 6E 64 65 72 20 73 79 6E    ge put under syn
00480:  63 70 6F 69 6E 74                                  cpoint
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 4 von 13)

```
LOG RECORD - LSN <0:0:0:53458>
**********

HLG Header: lrecsize 734, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : AQM Put Message (257)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 714                 LogRecdOwnr . . : 256    (AQM)
XTranid . . . . : TranType: NULL
QueueName . . . : Queue2
Qid . . . . . . : {Hash 184842943, Counter: 2}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 3
SpcIndex  . . . : 1
PrevLink.Locn . : 36                  PrevLink.Length : 8
PrevDataLink  . : {High 0, Low 2048}
Data.Locn . . . : 2048                Data.Length . . : 490
Data  . . . . . :
00000:  41 51 52 48 00 00 00 04 FF FF FF FF FF FF FF FF    AQRH............
00016:  00 00 00 00 00 00 00 00 00 00 00 01 00 01 01 C0    ...............&#192;
00032:  00 00 00 00 00 00 00 01 00 00 00 26 00 00 00 00    ...........&;...
00048:  00 00 00 00 41 4D 51 20 74 65 73 74 71 6D 20 20    ....AMQ testqm
00064:  20 20 20 20 33 80 2D D2 00 00 10 13 00 00 00 00        3€-&#30;......
00080:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00096:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00112:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01    ................
00128:  00 00 00 00 00 00 00 26 00 00 00 00 00 00 00 00    .......&;.......
00144:  00 00 00 00 00 00 00 C9 2C B6 D8 DD FF FF FF FF    ...&#26;,.&#216;...
00160:  4D 44 20 20 00 00 00 01 00 00 00 00 00 00 00 08    MD  ............
00176:  00 00 00 00 00 00 01 11 00 00 03 33 20 20 20 20    ...........3
00192:  20 20 20 20 00 00 00 00 00 00 00 01 20 20 20 20        ........
00208:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00224:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00240:  20 20 20 20 20 20 20 20 20 20 20 20 74 65 73 74                test
00256:  71 6D 20 20 20 20 20 20 20 20 20 20 20 20 20 20    qm
00272:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00288:  20 20 20 20 20 20 20 20 20 20 20 20 73 62 6F 6C                sbol
00304:  61 6D 20 20 20 20 20 20 04 37 34 38 30 00 00 00    am      .7480...
00320:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00336:  00 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20    ........
00352:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00368:  20 20 20 20 20 20 20 20 00 00 00 06 75 74 7A 61            ....utza
00384:  70 69 20 20 20 20 20 20 20 20 20 20 20 20 20 20    pi
00400:  20 20 20 20 20 20 20 20 31 39 39 37 30 35 31 39            19970519
00416:  31 30 34 33 32 37 30 36 20 20 20 20 00 00 00 00    10432706    ....
00432:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00448:  50 65 72 73 69 73 74 65 6E 74 20 6D 65 73 73 61    Persistent messa
00464:  67 65 20 6E 6F 74 20 70 75 74 20 75 6E 64 65 72    ge not put under
00480:  20 73 79 6E 63 70 6F 69 6E 74                       syncpoint
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 5 von 13)

```
LOG RECORD - LSN <0:0:0:54192>
**********

HLG Header: lrecsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Commit Transaction (774)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 196                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: MQI    TranNum{High 0, Low 1}
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:52482>

Version . . . . : 1
LOG RECORD - LSN <0:0:0:54408>
**********

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Start Transaction (769)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 200                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: MQI    TranNum{High 0, Low 3}
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
SoftLogLimit  . : 10000

LOG RECORD - LSN <0:0:0:54628>
**********

HLG Header: lrecsize 240, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : AQM Get Message (259)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 220                 LogRecdOwnr . . : 256    (AQM)
XTranid . . . . : TranType: MQI    TranNum{High 0, Low 3}
QueueName . . . : Queue1
Qid . . . . . . : {Hash 196836031, Counter: 0}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:54408>

Version . . . . : 2
SpcIndex  . . . : 1                   QPriority . . . : 0
PrevLink.Locn . : 36                  PrevLink.Length : 8
PrevDataLink  . : {High 4294967295, Low 4294967295}
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 6 von 13)

```
LOG RECORD - LSN <0:0:0:54868>
**********

HLG Header: lrecsize 240, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : AQM Get Message (259)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 220                 LogRecdOwnr . . : 256    (AQM)
XTranid . . . . : TranType: NULL
QueueName . . . : Queue2
Qid . . . . . . : {Hash 184842943, Counter: 2}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 2
SpcIndex  . . . : 1                   QPriority . . . : 0
PrevLink.Locn . : 36                  PrevLink.Length : 8
PrevDataLink  . : {High 4294967295, Low 4294967295}
LOG RECORD - LSN <0:0:0:55108>
**********

HLG Header: lrecsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Commit Transaction (774)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 196                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: MQI    TranNum{High 0, Low 3}
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:54628>

Version . . . . : 1

LOG RECORD - LSN <0:0:0:55324>
**********

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Start Transaction (769)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 200                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: XA
   XID: formatID 5067085, gtrid_length 14, bqual_length 4
        gtrid [3270BDB4000010237465737471 6D]
         bqual [00000001]
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
SoftLogLimit  . : 10000
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 7 von 13)

```
LOG RECORD - LSN <0:0:0:55544>
**********

HLG Header: lrecsize 738, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : AQM Put Message (257)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 718                 LogRecdOwnr . . : 256    (AQM)
XTranid . . . . : TranType: XA
   XID: formatID 5067085, gtrid_length 14, bqual_length 4
        gtrid [3270BDB4000010237465737 4716D]
        bqual [00000001]
QueueName . . . : Queue2
Qid . . . . . . : {Hash 184842943, Counter: 2}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:55324>

Version . . . . : 3
SpcIndex  . . . : 1
PrevLink.Locn . : 36                  PrevLink.Length : 8
PrevDataLink  . : {High 0, Low 2048}
Data.Locn . . . : 2048                Data.Length . . : 494
Data  . . . . . :
00000:  41 51 52 48 00 00 00 04 FF FF FF FF FF FF FF FF    AQRH............
00016:  00 00 00 00 00 00 00 00 00 00 00 01 00 01 01 C0    ...............&#192;
00032:  00 00 00 00 00 00 00 01 00 00 00 2A 00 00 00 00    ...........*....
00048:  00 00 00 01 41 4D 51 20 74 65 73 74 71 6D 20 20    ....AMQ testqm
00064:  20 20 20 20 33 80 2D D2 00 00 10 13 00 00 00 00        3€-&#30;........
00080:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00096:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00112:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01    ................
00128:  00 00 00 00 00 00 00 2A 00 00 00 00 00 00 00 00    .......*........
00144:  00 00 00 00 00 00 00 C9 2C B8 3E E8 FF FF FF FF    ...&#26;,&#184;>....
00160:  4D 44 20 20 00 00 00 01 00 00 00 00 00 00 00 08    MD  ............
00176:  00 00 00 00 00 00 01 11 00 00 03 33 20 20 20 20    ...........3
00192:  20 20 20 20 00 00 00 00 00 00 00 01 20 20 20 20        ........
00208:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00224:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00240:  20 20 20 20 20 20 20 20 20 20 20 20 74 65 73 74                test
00256:  71 6D 20 20 20 20 20 20 20 20 20 20 20 20 20 20    qm
00272:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00288:  20 20 20 20 20 20 20 20 20 20 20 20 73 62 6F 6C                sbol
00304:  61 6D 20 20 20 20 20 20 04 37 34 38 30 00 00 00    am      .7480...
00320:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00336:  00 00 00 00 00 00 00 00 20 20 20 20 20 20 20 20    ........
00352:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00368:  20 20 20 20 20 20 20 20 00 00 00 06 75 74 7A 61            ....utza
00384:  70 69 20 20 20 20 20 20 20 20 20 20 20 20 20 20    pi
00400:  20 20 20 20 20 20 20 20 31 39 39 37 30 35 31 39            19970519
00416:  31 30 34 34 35 38 37 32 20 20 20 20 00 00 00 00    10445872    ....
00432:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00448:  41 6E 6F 74 68 65 72 20 70 65 72 73 69 73 74 65    Another persiste
00464:  6E 74 20 6D 65 73 73 61 67 65 20 70 75 74 20 75    nt message put u
00480:  6E 64 65 72 20 73 79 6E 63 70 6F 69 6E 74          nder syncpoint
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 8 von 13)

```
LOG RECORD - LSN <0:0:0:56282>
**********

HLG Header: lrecsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Prepare Transaction (770)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 196                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: XA
   XID: formatID 5067085, gtrid_length 14, bqual_length 4
        gtrid [3270BDB4000010237465737471 6D]
         bqual [00000001]
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:55544>

Version . . . . : 1


LOG RECORD - LSN <0:0:0:56498>
**********

HLG Header: lrecsize 708, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : Transaction Prepared (1538)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 688                 LogRecdOwnr . . : 1536   (T)
XTranid . . . . : TranType: XA
   XID: formatID 5067085, gtrid_length 14, bqual_length 4
        gtrid [3270BDB4000010237465737471 6D]
         bqual [00000001]
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id. . . . . . . : TLPR
Version . . . . : 1                   Flags . . . . . : 1
Count . . . . . : 3

Participant Entry 0
RMId  . . . . . : 0                   State . . . . . : 2

Participant Entry 1
RMId  . . . . . : 1                   State . . . . . : 2

Participant Entry 2
RMId  . . . . . : 2                   State . . . . . : 2
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 9 von 13)

```
LOG RECORD - LSN <0:0:0:57206>
**********

HLG Header: lrecsize 216, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Commit Transaction (774)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 196                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: XA
   XID: formatID 5067085, gtrid_length 14, bqual_length 4
        gtrid [3270BDB40000102374657374716D]
        bqual [00000001]
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:56282>

Version . . . . : 1
LOG RECORD - LSN <0:0:0:57440>
**********

HLG Header: lrecsize 224, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : Transaction Forget (1539)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 204                 LogRecdOwnr . . : 1536   (T)
XTranid . . . . : TranType: XA
   XID: formatID 5067085, gtrid_length 14, bqual_length 4
        gtrid [3270BDB40000102374657374716D]
        bqual [00000001]
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id. . . . . . . : TLFG
Version . . . . : 1                   Flags . . . . . : 0
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 10 von 13)

```
LOG RECORD - LSN <0:0:0:58120>
**********

HLG Header: lrecsize 212, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ALM Start Checkpoint (1025)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 192                 LogRecdOwnr . . : 1024   (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

No data for Start Checkpoint Record

LOG RECORD - LSN <0:0:0:58332>
**********

HLG Header: lrecsize 220, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ATM Transaction Table (773)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 200                 LogRecdOwnr . . : 768    (ATM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
TranCount . . . : 0
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 11 von 13)

```
LOG RECORD - LSN <0:0:0:58552>
**********

HLG Header: lrecsize 1836, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : Transaction Participants (1537)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 1816                LogRecdOwnr . . : 1536   (T)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Id. . . . . . . : TLPH
Version . . . . : 1                   Flags . . . . . : 3
Count . . . . . : 2

Participant Entry 0
RMName  . . . . : DB2 MQBankDB
RMId  . . . . . : 1
SwitchFile  . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString  . :
XACloseString . :

Participant Entry 1
RMName  . . . . : DB2 MQFeeDB
RMId  . . . . . : 2
SwitchFile  . . : /Development/sbolam/build/devlib/tstxasw
XAOpenString  . :
XACloseString . :

LOG RECORD - LSN <0:0:0:60388>
**********

HLG Header: lrecsize 236, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ALM End Checkpoint (1026)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 216                 LogRecdOwnr . . : 1024   (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

ChkPtLSN  . . . : <0:0:0:58120>
OldestLSN . . . : <0:0:0:0>
MediaLSN  . . . : <0:0:0:0>
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 12 von 13)

```
LOG RECORD - LSN <0:0:0:60624>
**********

HLG Header: lrecsize 240, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : ALM Stop Queue Manager (1028)
Eyecatcher  . . : ALRH                Version . . . . : 1
LogRecdLen  . . : 220                 LogRecdOwnr . . : 1024   (ALM)
XTranid . . . . : TranType: NULL
QueueName . . . : NULL
Qid . . . . . . : {NULL_QID}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:0:0>

Version . . . . : 1
StopDate  . . . : 19970519            StopTime  . . . : 10490868
SessionNumber . : 0                   ForceFlag . . . : Quiesce

AMQ7702: DMPMQLOG command has finished successfully.
```

Abbildung 18. Beispiel einer dmpmqlog-Ausgabe (Teile- 13 von 13)

Hinweise zu Abb. 18 auf Seite 162:

1. Das Feld *headlsn* im Protokollsatz *Log File Header* enthält den Wert <0:0:0:58120>. An diesem Punkt wäre der Auszug gestartet worden, wenn nicht eine andere Protokollfolgenummer als Anfangspunkt angegeben worden wäre.
2. Das Feld *nextlsn* enthält den Wert <0:0:0:60864>; das ist die Protokollfolgenummer des ersten Protokollsatzes, den der WS-Manager schreiben wird, wenn er das nächste Mal neu gestartet wird.
3. Das Feld *HeadExtentID* enthält den Wert 1, der anzeigt, dass der Kopfsatz des Protokolls sich zurzeit in Protokolldatei S0000001.LOG befindet.
4. Der erste Protokollsatz, der formatiert wird, ist ein *Start Checkpoint*-Protokollsatz. Der Prüfpunkt umfasst alle Protokollsätze bis zum Protokollsatz *End CheckPoint* mit der Nummer <0:0:0:46448>.
5. Einer der Sätze, die bei der Prüfpunktverarbeitung protokolliert wurden, ist der Protokollsatz *Transaction Participants* mit der Nummer <0:0:0:44594>. Er enthält die Ressourcenmanager, die an globalen Transaktionen, die vom WS-Manager koordiniert werden, teilnehmen.
6. Der Protokollsatz *Start Transaction* mit der Nummer <0:0:0:52262> bezeichnet den Start einer Transaktion. Das Feld *XTranid* zeigt für *TranType* den Wert MQI an, der angibt, dass es sich um eine lokale Transaktion handelt, die nur MQ-Aktualisierungen enthält.
7. Der nächste Protokollsatz ist ein *Put Message*-Protokollsatz, in dem der MQPUT-Aufruf für eine permanente Nachricht an dem Synchronisationspunkt aufgezeichnet wird, der die Transaktion gestartet hat. Der MQPUT-Aufruf richtete sich an die Warteschlange *Queue1* und die Nachricht wurde als `Persistent message put under syncpoint` (Zwischen Synchronisationspunkten eingereihte permanente Nachricht) protokolliert. Dieser Nachricht wurde ein *SpcIndex* von 1 zugeordnet, der dem weiter unten erwähnten MQGET für diese Nachricht übereinstimmt.

8. Der nächste Protokollsatz mit der Protokollfolgenummer <0:0:0:53458> ist wieder ein *Put Message*-Satz. Diese permanente Nachricht wurde in eine andere Warteschlange (*Queue2*) eingereiht; sie wurde jedoch nicht zwischen Synchronisationspunkten eingereiht, denn *XTranid* ist gleich *NULL*. Ihr ist ebenfalls ein *SpcIndex* von 1 zugeordnet; dieser Wert ist eine eindeutige ID für diese spezielle Warteschlange.
9. Der nächste Protokollsatz mit der Protokollfolgenummer <0:0:0:54192> schreibt die Nachricht fest, die zwischen Synchronisationspunkten eingereiht wurde.
10. In den Protokollsätzen <0:0:0:54408> und <0:0:0:54628> wird zwischen Synchronisationspunkten von einem MQGET für Warteschlange *Queue1* eine neue Transaktion gestartet. Das Feld *SpcIndex* im Protokollsatz *Get Message* enthält den Wert 1, der anzeigt, dass es sich um dieselbe Nachricht handelt, die in <0:0:0:52262> in Warteschlange *Queue1* eingereiht wurde.
11. Der nächste Protokollsatz ruft die Nachricht ab, die von dem anderen *Put Message*-Protokollsatz in die Warteschlange *Queue2* eingereiht wurde.
12. Der MQGET-Aufruf zwischen Synchronisationspunkten wurde festgeschrieben, wie vom Protokollsatz *Commit Transaction* mit der Nummer <0:0:0:55108> angezeigt.
13. Schließlich wird im Protokollsatz *Start Transaction* mit der Nummer <0:0:0:55324> mit einem MQBEGIN-Aufruf eine globale Transaktion gestartet. Das Feld *XTranid* in diesem Protokollsatz enthält für *TranType* den Wert XA.
14. Der folgende *Put Message*-Protokollsatz zeichnet eine permanente Nachricht auf, die in Warteschlange *Queue2* eingereiht wurde. Der Wert für die *XTranid* in diesem Satz ist derselbe wie im vorhergehenden Protokollsatz.
15. Wenn für diese *Xtranid* ein *Transaction Prepared*-Protokollsatz geschrieben wird, muss die Transaktion als Ganzes festgeschrieben werden. Wenn dieser Protokollsatz fehlt, kann das als Hinweis darauf verstanden werden, dass die Transaktion zurückgesetzt wurde. In diesem Fall gibt es einen *Transaction Prepared*-Protokollsatz mit der Nummer <0:0:0:56498>. Dieser zeichnet den WS-Manager selbst als Teilnehmer mit einer *RMId* gleich NULL auf. Es gibt zwei weitere Teilnehmer, deren *RMIds* 1 und 2 dem vorhergehenden *Transaction Participants*-Protokollsatz zugeordnet werden können.
16. Während der COMMIT-Phase protokolliert der XA-Transaktionsmanagers des WS-Managers keine einzelnen Antworten der Teilnehmer. Das Protokoll zeigt nur an, ob die WS-Manager-Aktualisierungen festgeschrieben wurden oder nicht. Der *Commit Transaction*-Protokollsatz mit der Nummer <0:0:0:57206> zeigt an, dass die Nachricht tatsächlich für Warteschlange *Queue2* festgeschrieben wurde.
17. Der *Transaction Forget*-Protokollsatz mit der Nummer <0:0:0:57440> zeigt an, dass die COMMIT-Entscheidung auch an die beiden anderen Ressourcenmanager zugestellt wurde. Jeder Fehler bei der Festschreibung der Aktualisierungen durch diese Ressourcenmanager wird in den Fehlerprotokollen des WS-Managers diagnostiziert.

Befehl dmpmqlog verwenden

Kapitel 12. Namensservice verwenden

Der Namensservice ist ein installierbarer Service, mit dessen Hilfe Anwendungen, die mit einem WS-Manager verbunden sind, Warteschlangen öffnen können, bei denen es sich aus Sicht der Anwendung um lokale Warteschlangen handelt. Diese Warteschlangen sind tatsächlich jedoch Warteschlangen, die auf einem anderen WS-Manager, d. h. einer anderen Maschine, definiert sind, wobei das Attribut SCOPE auf den Wert CELL gesetzt ist.

Die Anwendung kann für Warteschlangen, die auf diese Weise geöffnet wurden, alle Operationen ausführen, die für ferne Warteschlangen zulässig sind. Die bereitgestellte Implementierung verwendet DCE (Distributed Computing Environment), Sie können jedoch auch eine eigene Komponente erstellen, die DCE nicht verwendet.

Um die bereitgestellte Namensservice-Komponente zu verwenden, müssen Sie den Namensservice und dessen installierte Komponente auf dem WS-Manager definieren. Fügen Sie dazu die entsprechende Zeilengruppe in die Konfigurationsdatei (qm.ini) des WS-Managers ein. Detaillierte Informationen finden Sie im Handbuch *MQSeries Programmable System Management*. Außerdem sind Anpassungen der DCE-Konfiguration erforderlich.

DCE für eine gemeinsame Benutzung von Warteschlangen auf verschiedenen WS-Managern verwenden

Wenn sich Ihre WS-Manager auf Knoten innerhalb einer DCE-Zelle befinden, können Sie sie so konfigurieren, dass sie Warteschlangen gemeinsam benutzen. Anwendungen können dann eine Verbindung mit einem der WS-Manager herstellen und eine Warteschlange auf einem *anderen* WS-Manager auf einem anderen Knoten öffnen. Dies ist für die Anwendung transparent; sie weiß nicht, dass sich die Warteschlange in Wirklichkeit auf einem anderen WS-Manager befindet. (Normalerweise weist der WS-Manager Anforderungen einer lokalen Anwendung zum Öffnen einer Warteschlange zurück, wenn es sich nicht um eine Warteschlange auf dem WS-Manager selbst handelt.)

Konfigurations-Tasks für gemeinsame Warteschlangen

Dieser Abschnitt beschreibt, wie gemeinsame Warteschlangen auf WS-Managern definiert werden, die sich auf Knoten innerhalb der DCE-Zelle befinden.

Führen Sie für jeden WS-Manager die folgenden Tasks aus:

1. Konfigurieren Sie den Namensservice, indem Sie die erforderliche Zeilengruppe für den Namensservice in der Konfigurationsdatei des WS-Managers hinzufügen. Der Inhalt dieser Zeilengruppe wird im Handbuch *MQSeries Programmable System Management* beschrieben. Um den Namensservice aufzurufen, müssen Sie den WS-Manager erneut starten.
2. Wenn der WS-Manager aktiv ist, stoppen Sie ihn mit dem Befehl **endmqm**.
3. Starten Sie den WS-Manager mit dem Befehl **strmqm** erneut.
4. Definieren Sie Kanäle für Nachrichtenübertragungen zwischen den WS-Managern (siehe „Kanäle und Übertragungswarteschlangen für Fernverwaltung vorbereiten” auf Seite 68).

Warteschlangen gemeinsam benutzen

Geben Sie für jede Warteschlange, die gemeinsam benutzt werden soll, für das Attribut SCOPE den Wert CELL an. Verwenden Sie dazu zum Beispiel die folgenden MQSC-Befehle:

```
DEFINE QLOCAL(GREY.PUBLIC.QUEUE) SCOPE(CELL)
```

oder

```
ALTER QLOCAL(PINK.LOCAL.QUEUE) SCOPE(CELL)
```

Die erstellte oder geänderte Warteschlange muss einem WS-Manager auf einem Knoten innerhalb der DCE-Zelle zugeordnet sein.

DCE-Konfiguration

Um die bereitgestellte Namensservice-Komponente verwenden zu können, muss OSF Distributed Computing Environment (DCE) installiert sein. Über diesen Service können Anwendungen, die mit einem der WS-Manager verbunden sind, Warteschlangen öffnen, die einem anderen WS-Manager innerhalb derselben DCE-Zelle zugeordnet sind.

Ein Beispiel für ein DCL-Shell-Script, mit dem der bereitgestellte Namensservice ausgeführt werden kann, finden Sie in der Datei `mqs_examples:dcesetu.com`.

Kapitel 13. MQSeries konfigurieren

Dieses Kapitel erläutert, wie Sie die Funktionsweise eines einzelnen WS-Managers oder eines Knotens an die Anforderungen Ihrer Installation anpassen können.

Sie ändern MQSeries-Konfigurationsdaten, indem Sie die Werte für eine Gruppe von Konfigurationsattributen (oder Parametern), über die MQSeries gesteuert wird, ändern.

Die Art und Weise, wie diese Konfigurationsdaten geändert werden, und wo MQSeries die Änderungen speichert, ist von der jeweiligen Plattform abhängig. Benutzer von MQSeries for Compaq OpenVMS ändern die Konfigurationsdaten, indem sie die **MQSeries-Konfigurationsdateien** editieren.

Diese Kapitel enthält folgende Abschnitte:

- „Attribute für die Änderung von MQSeries-Konfigurationsdaten" auf Seite 181 mit einer Beschreibung der Attribute, über die Sie MQSeries-Konfigurationsdaten ändern.
- „Konfigurationsdaten des WS-Managers ändern" auf Seite 187 mit einer Beschreibung der Attribute, über die Sie WS-Manager-Konfigurationsdaten ändern.
- „Beispiele für die Dateien mqs.ini und qm.ini" auf Seite 196 mit Beispielen für die Dateien `mqs.ini` und `qm.ini` für MQSeries for Compaq OpenVMS.

MQSeries-Konfigurationsdateien

Benutzer von MQSeries for Compaq OpenVMS müssen zum Ändern von MQSeries-Konfigurationsattributen folgende Dateien editieren:

- eine MQSeriesKonfigurationsdatei (`mqs.ini`), um MQSeries-Änderungen auf dem Knoten als Ganzes durchzuführen. Es gibt für jeden Knoten eine Datei `mqs.ini`.
- eine WS-Manager-Konfigurationsdatei (`qm.ini`), um Änderungen für einzelne WS-Manager durchzuführen. Es gibt für jeden WS-Manager auf dem Knoten eine Datei `qm.ini`.

Eine Konfigurationsdatei (die auch als ***Zeilengruppendatei*** oder ***INI***-Datei bezeichnet wird) enthält eine oder mehrere Zeilengruppen, die nichts anderes sind, als eine Gruppen von Zeilen in der Datei, die zusammen eine bestimmte Funktion haben oder einen Teil eines Systems definieren, z. B. Protokollfunktionen, Kanalfunktionen oder installierbare Services.

Änderungen, die Sie in einer Konfigurationsdatei vornehmen, werden erst wirksam, wenn der WS-Manager das nächste Mal gestartet wird.

Konfigurationsdateien editieren

Bevor Sie mit dem Editieren einer Konfigurationsdatei beginnen, sollten Sie unbedingt eine Sicherung erstellen, damit Sie über eine Kopie verfügen, auf die Sie gegebenenfalls zurückgreifen können!

Sie haben zwei Möglichkeiten zum Editieren von Konfigurationsdateien:

- automatisch, indem Sie Befehle verwenden, mit denen Sie die Konfiguration von WS-Managern auf dem Knoten ändern.

- manuell, indem Sie einen Standard-Texteditor verwenden.

Sie können die Standardwerte in den MQSeries-Konfigurationsdateien nach der Installation ändern.

Wenn Sie für ein Attribut in einer Konfigurationsdatei einen ungültigen Wert angeben, wird der Wert ignoriert und eine Bedienernachricht mit einem Hinweis auf den Fehler ausgegeben. (Im Ergebnis bedeutet dies dasselbe, als hätten Sie das Attribut gar nicht angegeben.)

Führen Sie beim Erstellen eines neuen WS-Managers Folgendes aus:
- Sichern der MQSeries-Konfigurationsdatei
- Sichern der Konfigurationsdatei des neuen WS-Managers

Gründe für das Editieren einer Konfigurationsdatei

Eine Konfigurationsdatei muss zum Beispiel in folgenden Fällen editiert werden:

- Eine Konfigurationsdatei ist verloren gegangen; sie kann gegebenenfalls aus einer Sicherung zurückgeschrieben werden.
- Ein oder mehrere WS-Manager müssen in ein neues Verzeichnis versetzt werden.
- Der Standard-WS-Manager muss geändert werden; dies kann erforderlich sein, wenn der vorhandene WS-Manager versehentlich gelöscht wurde.
- Sie haben eine entsprechende Anweisung vom IBM Support Center erhalten.

Prioritäten innerhalb der Konfigurationsdatei

Für die Festlegung der Attributwerte in einer Konfigurationsdatei gelten folgende Prioritäten:

- Parameter, die in der Befehlszeile eingegeben werden, haben Vorrang vor Werten, die in der Konfigurationsdatei definiert werden.
- Werte, die in den `qm.ini`-Dateien definiert werden, haben Vorrang vor Werten, die in der Datei `mqs.ini` definiert werden.

Änderungen in Konfigurationsdateien implementieren

Wenn Sie eine Konfigurationsdatei ändern, werden die Änderungen nicht sofort vom WS-Manager implementiert. Änderungen in der MQSeries-Konfigurationsdatei werden nur beim Start von MQSeries implementiert. Änderungen in einer WS-Manager-Konfigurationsdatei werden beim Start des WS-Managers implementiert. Ist der WS-Manager aktiv, während Sie die Änderungen vornehmen, müssen Sie den WS-Manager stoppen und erneut starten, damit das System die Änderungen erkennt.

Die MQSeries-Konfigurationsdatei mqs.ini

Die MQSeries-Konfigurationsdatei `mqs.ini` enthält Informationen, die alle WS-Manager auf dem Knoten betreffen. Sie wird bei der Installation automatisch erstellt. Die Datei `mqs.ini` enthält vor allem die Daten zu den einzelnen WS-Managern.

Die Datei `mqs.ini` befindet sich standardmäßig im Verzeichnis **MQS_ROOT:[MQM]**.

Die Datei `mqs.ini` enthält Folgendes:
- die Namen der WS-Manager.
- den Namen des Standard-WS-Managers.
- die Adressen der Dateien, die den einzelnen WS-Managern zugeordnet sind.

Weitere Informationen zum Inhalt der Datei mqs.ini finden Sie unter „Attribute für die Änderung von MQSeries-Konfigurationsdaten".

Konfigurationsdateien (qm.ini) des WS-Managers

Eine WS-Manager-Konfigurationsdatei (qm.ini) enthält wichtige Informationen zu einem einzelnen WS-Manager. Es gibt für jeden WS-Manager eine WS-Manager-Konfigurationsdatei. Die Datei qm.ini wird automatisch mit dem ihr zugeordneten WS-Manager erstellt.

Eine qm.ini-Datei wird im Stammverzeichnis der Verzeichnisstruktur des WS-Managers gespeichert.

In MQSeries for Compaq OpenVMS lautet der vollständige Name (Pfad und Dateiname) für die Konfigurationsdatei eines WS-Managers mit dem Namen QMNAME zum Beispiel wie folgt:

```
MQS_ROOT:[MQM.QMGRS.QMNAME]QM.INI
```

Anmerkung: Der Name des WS-Managers darf maximal 48 Zeichen lang sein. Das garantiert jedoch nicht, dass der Name gültig oder eindeutig ist. Deshalb wird ein Verzeichnisname auf Basis des WS-Manager-Namens generiert. Dieser Prozess wird als **Namensumwandlung** bezeichnet. Eine Beschreibung finden Sie unter „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21.

Weitere Informationen zur Datei qm.ini finden Sie unter „Konfigurationsdaten des WS-Managers ändern" auf Seite 187.

Attribute für die Änderung von MQSeries-Konfigurationsdaten

Die Datei mqs.ini enthält folgende Attributgruppen:

- Die Zeilengruppe AllQueueManagers
- „Die Zeilengruppe ClientExitPath" auf Seite 183
- „Die Zeilengruppe DefaultQueueManager" auf Seite 183
- „Die Zeilengruppe ExitProperties" auf Seite 183
- „Die Zeilengruppe LogDefaults" auf Seite 184
- „Die Zeilengruppe QueueManager" auf Seite 186

Ein Beispiel für die Datei mqs.ini finden Sie unter „Beispiele für die Dateien mqs.ini und qm.ini" auf Seite 196.

Die Zeilengruppe AllQueueManagers

Die Zeilengruppe AllQueueManagers kann folgende Angaben enthalten:

- den Pfad für das Verzeichnis 'qmgrs', in dem die einem WS-Manager zugeordneten Dateien gespeichert werden.
- die Methode zum Konvertieren von Daten aus dem EBCDIC-Format in ASCII-Format.

DefaultPrefix=*Verzeichnisname*

Dieses Attribut gibt den Pfad für das Verzeichnis 'qmgrs' an, in dem die WS-Manager-Daten gespeichert werden.

Wenn Sie das Standardpräfix für den WS-Manager ändern, müssen Sie die Verzeichnisstruktur replizieren, die bei der Installation erstellt wurde (siehe „Anhang C. Verzeichnisstruktur" auf Seite 329).

Vor allem muss die Struktur 'qmgrs' erstellt werden. Bevor Sie das Standardpräfix ändern, müssen Sie MQSeries stoppen. Starten Sie es erst wieder, nachdem Sie die Strukturen in das neue Verzeichnis versetzt und das Standardpräfix geändert haben.

Eine andere Möglichkeit zum Ändern des Standardpräfixes besteht darin, das Attribut `DefaultPrefix`, das von dem Befehl **crtmqm** gelesen wird, mit Hilfe des logischen Namens MQSPREFIX zu überschreiben.

ConvEBCDICNewline=NL_TO_LF|TABLE|ISO

EBCDIC-Codepages enthalten ein Zeilenumbruchzeichen (NL, New Line), das von ASCII-Codepages nicht unterstützt wird (obwohl einige ISO-Varianten des ASCII-Codes ein entsprechendes Zeichen enthalten).

Über das Attribut ConvEBCDICNewline können Sie die Methode angeben, die MQSeries für die Konvertierung des EBCDIC-Zeilenumbruchzeichens in ASCII-Format verwenden soll.

NL_TO_LF

Geben Sie NL_TO_LF an, wenn das EBCDIC-Zeilenumbruchzeichen (X'15') bei allen Konvertierungen von EBCDIC nach ASCII in das ASCII-Zeichen für Zeilenvorschub, LF (X'0A'), konvertiert werden soll.

NL_TO_LF ist der Standardwert.

TABLE

Geben Sie TABLE an, wenn das EBCDIC-Zeilenumbruchzeichen bei allen Konvertierungen von EBCDIC nach ASCII gemäß den auf Ihrer Plattform verwendeten Konvertierungstabellen konvertiert werden soll.

Beachten Sie, dass das Ergebnis bei dieser Art der Konvertierung von Plattform zu Plattform und von Sprache zu Sprache unterschiedlich sein kann; auch auf derselben Plattform kann das Ergebnis unterschiedlich sein, wenn Sie unterschiedliche CCSIDs verwenden.

ISO

Geben Sie ISO an, wenn:

- ISO-CCSIDs mit der Methode TABLE konvertiert werden sollen,
- alle anderen CCSIDs mit der Methode NL_TO_LF konvertiert werden sollen.

Tabelle 7 enthält eine Auflistung möglicher ISO-CCSIDs.

Tabelle 7. Liste möglicher ISO-CCSIDs

CCSID	Codierter Zeichensatz
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

Wenn es sich bei der ASCII-CCSID nicht um eine ISO-Untergruppe handelt, enthält ConvEBCDICNewline den Standardwert NL_TO_LF.

Weitere Informationen zur Datenkonvertierung finden Sie im Handbuch *MQSeries Application Programming Guide* bzw. unter „Datenkonvertierung" auf Seite 78.

Die Zeilengruppe ClientExitPath

Die Zeilengruppe `ClientExitPath` gibt den Standardpfad für die Adresse des Kanal-Exits auf dem Client an.

ExitsDefaultPath=*Standardpräfix*
Das Attribut ExitsDefaultPath gibt das Standardpräfix für die Plattform an.

Die Zeilengruppe DefaultQueueManager

Die Zeilengruppe `DefaultQueueManager` gibt den Standard-WS-Manager für den Knoten an.

Name=*Standard-WS-Manager*
Der Standard-WS-Manager verarbeitet alle Befehle, für die nicht explizit ein WS-Manager-Name angegeben wurde. Das Attribut `DefaultQueueManager` wird automatisch aktualisiert, wenn ein neuer Standard-WS-Manager erstellt wird. Wenn Sie versehentlich einen neuen Standard-WS-Manager erstellt haben und zum ursprünglichen zurückkehren wollen, müssen Sie das Attribut `DefaultQueueManager` manuell ändern.

Die Zeilengruppe ExitProperties

Die Zeilengruppe `ExitProperties` gibt Konfigurationsoptionen an, die von Exit-Programmen des WS-Managers verwendet werden.

CLWLMode=SAFE|FAST
Über den Exit für die Cluster-Auslastung (CLWL, Cluster Workload Exit) können Sie angeben, welche Cluster-Warteschlange innerhalb des Clusters als Antwort auf einen MQAPI-Aufruf (MQOPEN oder MQPUT usw.) geöffnet werden soll. Der CLWL-Exit wird entweder im FAST- oder SAFE-Modus ausgeführt, abhängig von dem Wert, den Sie für das Attribut CLWLMode angeben. Wenn Sie das Attribut CLWLMode nicht angeben, wird der Exit für die Cluster-Auslastung im SAFE-Modus ausgeführt.

SAFE
Die Option SAFE gibt an, dass der CLWL-Exit in einem vom WS-Manager getrennten Prozess ausgeführt wird. Dies ist der Standardwert.

Wenn bei der Ausführung des benutzerdefinierten CLWL-Exits im SAFE-Modus ein Fehler auftritt, geschieht Folgendes:

- Der CLWL-Serverprozess (amqzlwa0) schlägt fehl.
- Der WS-Manager startet den CLWL-Serverprozess erneut.
- Der Fehler wird in das Fehlerprotokoll geschrieben. Wenn gerade ein MQAPI-Aufruf verarbeitet wird, erhalten Sie einen Hinweis in Form eines negativen Rückkehrcodes.

Die Integrität des WS-Managers bleibt erhalten.

Anmerkung: Bei der Ausführung des CLWL-Exits in einem getrennten Prozess entsteht eine zusätzliche Systembelastung, die die Leistung beeinträchtigen kann.

FAST
Geben Sie FAST an, wenn der Cluster-Exit in den WS-Manager-Prozess integriert werden soll.

Bei Angabe dieser Option wird die Leistung verbessert, weil die zusätzlichen Systembelastungen, die bei einer Ausführung im SAFE-Modus entstehen, vermieden werden; dies geschieht allerdings zum Nachteil der Integrität des WS-Managers. Daher sollten Sie den CLWL-Exit nur im FAST-Modus ausführen, wenn Sie sicher sind, dass es **keine** Probleme im Zusammenhang mit dem CLWL-Exit gibt, und Sie eine ernsthafte Verschlechterung der Leistung feststellen.

Wenn bei der Ausführung des CLWL-Exits im FAST-Modus ein Fehler auftritt, schlägt der WS-Manager fehlt, und es besteht die Gefahr, dass die Integrität des WS-Managers beschädigt wird.

Die Zeilengruppe LogDefaults

Die Zeilengruppe `LogDefaults` gibt die Standardprotokollattribute für den Knoten an. Die Protokollattribute werden beim Erstellen eines WS-Managers als Standardwerte verwendet, können aber überschrieben werden, indem Sie die Protokollattribute im Befehl **crtmqm** angeben. Eine Beschreibung dieses Befehls finden Sie unter „crtmqm (WS-Manager erstellen)” auf Seite 260.

Nachdem ein WS-Manager erstellt wurde, werden die Protokollattribute für diesen WS-Manager aus seiner Protokollzeilengruppe in der Datei `qm.ini` gelesen.

Das Attribut *DefaultPrefix* in der Zeilengruppe `AllQueueManagers` und das Attribut *LogPath* in der Zeilengruppe `LogDefaults` ermöglichen es, für den WS-Manager und für sein Protokoll unterschiedliche physische Laufwerke anzugeben. Diese Methode wird empfohlen, obwohl sie sich standardmäßig auf demselben Laufwerk befinden.

Informationen zur Berechnung der Protokollgröße finden Sie unter „Protokollgröße berechnen” auf Seite 147.

Anmerkung: Bei den in der folgenden Parameterliste genannten Begrenzungen handelt es sich um MQSeries-Begrenzungen. Die maximal mögliche Protokollgröße kann durch Betriebssystembegrenzungen verringert werden.

LogPrimaryFiles=3|*2-62*
Primäre Protokolldateien sind die Protokolldateien, die bei der Erstellung für eine spätere Verwendung angelegt werden.

Es müssen mindestens zwei primäre Protokolldateien vorhanden sein, ihre maximale Anzahl beträgt 62. Der Standardwert ist 3.

Die Gesamtzahl der primären und sekundären Protokolldateien darf 63 nicht überschreiten und nicht kleiner als 3 sein.

Dieser Wert kann beim Erstellen des WS-Managers durch den Parameter -lp des Befehls **crtmqm** überschrieben werden.

LogSecondaryFiles=2|*1-61*
Sekundäre Protokolldateien werden angelegt, wenn der Speicherbereich in den primären Protokolldateien erschöpft ist.

Es muss mindestens eine sekundäre Protokolldatei vorhanden sein, die maximale Anzahl beträgt 61. Der Standardwert ist 2.

Die Gesamtzahl der primären und sekundären Protokolldateien darf 63 nicht überschreiten und nicht kleiner als 3 sein.

Dieser Wert kann beim Erstellen des WS-Managers durch den Parameter -ls des Befehls **crtmqm** überschrieben werden.

LogFilePages=*Anzahl*
Die Protokolldaten werden in einer Folge von Dateien, den Protokolldateien, gespeichert. Die Protokolldateigröße wird in Einheiten von 4-KB-Seiten angegeben.

In MQSeries for Compaq OpenVMS beträgt die Standardanzahl von Protokolldateien 1024, was einer Protokolldateigröße von 4 MB entspricht. Die minimale Anzahl von Protokolldateiseiten beträgt 64 und die maximale Anzahl 16384.

Dieser Wert kann beim Erstellen des WS-Managers durch den Parameter -lf des Befehls **crtmqm** überschrieben werden.

LogType=CIRCULAR|LINEAR
Über das Attribut `LogType` wird der zu verwendende Protokolltyp (zyklisch oder linear) definiert. Der Standardwert ist CIRCULAR (zyklisch).

CIRCULAR
Geben Sie diesen Wert an, wenn eine Wiederherstellung nach Neustart gestartet werden soll, bei der das Protokoll verwendet wird, um Transaktionen zurückzusetzen, die noch nicht abgeschlossen waren, als das System gestoppt wurde.

Eine ausführliche Erläuterung der zyklischen Protokollierung finden Sie unter „Zyklische Protokollierung" auf Seite 142.

LINEAR
Geben Sie diesen Wert an, wenn Sie beides verwenden wollen, sowohl die Wiederherstellung nach Neustart als auch die Wiederherstellung über Datenträger bzw. Vorwärtswiederherstellung (Erstellung verloren gegangener oder beschädigter Daten durch erneute Ausführung der protokollierten Aktionen).

Eine ausführliche Erläuterung der linearen Protokollierung finden Sie unter „Lineare Protokollierung" auf Seite 143.

Um den Standardprotokolltyp zu ändern, können Sie das Attribut `LogType` in der Datei `mqs.ini` editieren. Alternativ dazu können Sie den Standardwert überschreiben, indem Sie im Befehl **crtmqm** den Parameter -ll für lineare Protokollierung angeben. Sie können die Protokollierungsmethode nicht ändern, nachdem ein WS-Manager erstellt wurde.

LogBufferPages=17|*4-32*
Die Speicherkapazität, die als Puffer zum Schreiben von Datensätzen zugeordnet wird, ist konfigurierbar. Die Größe der Puffer wird in Einheiten von 4-KB-Seiten angegeben.

Die minimale Anzahl an Pufferseiten beträgt 4 und die maximale Anzahl 32. Größere Puffer führen zu einem höheren Durchsatz, insbesondere bei sehr langen Nachrichten.

Die Standardanzahl an Pufferseiten beträgt 17, was 68 KB entspricht.

Der Wert wird beim Erstellen bzw. Starten des WS-Managers überprüft und kann bei jedem dieser Vorgänge erhöht oder verkleinert werden. Eine Änderung des Wertes wird jedoch erst wirksam, wenn der WS-Manager erneut gestartet wird.

LogDefaultPath=*Verzeichnisname*
Sie können das Verzeichnis für die Protokolldateien eines WS-Managers angeben. Das Verzeichnis muss sich auf einer lokalen Einheit befinden, für das der WS-Manager über Schreibzugriff verfügt; dabei sollte es sich vorzugsweise um ein anderes Laufwerk handeln als das, auf dem sich die Nachrichtenwarteschlangen befinden. Durch die Auswahl eines anderen Laufwerks erhöhen Sie den Schutz im Falle eines Systemfehlers.

Der Standardwert in MQSeries for Compaq OpenVMS ist `MQS_ROOT:[MQM.LOG]`.

Alternativ können Sie den Namen eines Verzeichnisses auch im Befehl **crtmqm** mit der Option -ld angeben. Beim Erstellen eines WS-Managers wird auch im WS-Manager-Verzeichnis ein Verzeichnis für die Speicherung der Protokolldateien erstellt. Der Name dieses Verzeichnisses basiert auf dem Namen des WS-Managers. Dadurch wird sichergestellt, dass der Protokolldateipfad eindeutig ist und dass der Name den Begrenzungen für die Länge von Verzeichnisnamen entspricht.

Wenn Sie im Befehl **crtmqm** nicht die Option -ld angeben, wird standardmäßig der Wert des Attributs `LogDefaultPath` in der Datei `mqs.ini` verwendet, also `MQS_ROOT:[MQM.LOG]`.

Der Name des WS-Managers wird an den Namen des Protokolldateiverzeichnisses angehängt, um sicherzustellen, dass mehrere WS-Manager auch unterschiedliche Protokollverzeichnisse verwenden.

Nachdem der WS-Manager erstellt wurde, wird in der Zeilengruppe Log in der Datei `qm.ini` ein `LogPath`-Wert erstellt, der dem vollständigen Verzeichnisnamen für die Protokolldateien des WS-Managers entspricht. Über diesen Wert werden die Protokolldateien lokalisiert, wenn der WS-Manager gestartet oder gelöscht wird.

Die Zeilengruppe QueueManager

Es gibt für jeden WS-Manager eine `QueueManager`-Zeilengruppe. Die Attribute geben den Namen des WS-Managers und den Namen des Verzeichnisses mit den Dateien, die dem WS-Manager zugeordnet sind, an. Der Name des Verzeichnisses basiert auf dem Namen des WS-Managers, wird aber umgewandelt, wenn der WS-Manager-Name kein gültiger Dateiname ist.

Weitere Informationen zur Namensumwandlung finden Sie unter „Erläuterungen zu MQSeries-Dateinamen” auf Seite 21.

Name=*WS-Manager-Name*
Dieses Attribut gibt den Namen des WS-Managers an.

Prefix=*Präfix*
Dieses Attribut gibt an, wo die WS-Manager-Dateien gespeichert werden. Standardmäßig entspricht dieser Wert dem des Attributs DefaultPrefix der Zeilengruppe `AllQueueManager` in der Datei `mqs.ini`.

Directory=*Name*
Dieses Attribut gibt den Namen des Unterverzeichnisses an, in dem die WS-Manager-Dateien gespeichert werden. In der Regel ist dies das Verzeichnis `MQS_ROOT:[MQM.QMGRS]`, es sei denn, es wurde ein anderes Präfix angegeben. Dieser Name basiert auf dem Namen des WS-Managers, kann aber umgewandelt werden, wenn der Name bereits existiert oder der WS-Manager-Name kein gültiger Dateiname ist.

Konfigurationsdaten des WS-Managers ändern

Die folgenden Attributgruppen können in einer qm.ini-Datei eines einzelnen WS-Managers stehen oder zum Überschreiben der in der Datei mqs.ini festgelegten Werte verwendet werden.

- „Die Zeilengruppe Service"
- „Die Zeilengruppe ServiceComponent"
- „Die Zeilengruppe Log" auf Seite 188
- „Die Zeilengruppe XAResourceManager" auf Seite 190
- „Die Zeilengruppe Channels" auf Seite 192
- „Die Zeilengruppen LU62 und TCP" auf Seite 194
- „Die Zeilengruppe ExitPath" auf Seite 195

Die Zeilengruppe Service

Die Zeilengruppe Service gibt den Namen eines installierbaren Services und die Anzahl der Eingangspunkte dieses Services an. Für jeden verwendeten Service muss es eine Service-Zeilengruppe geben.

Für jede Komponente eines Services muss es eine ServiceComponent-Zeilengruppe geben, die den Namen und Pfad des Moduls mit dem Code für die Komponente enthält. Weitere Informationen finden Sie unter „Die Zeilengruppe ServiceComponent".

Name=AuthorizationService | NameService
Gibt den Namen des erforderlichen Services an.

AuthorizationService
In MQSeries handelt es sich bei dem Berechtigungsservice um den so genannten Objektberechtigungsmanager (OAM, Object Authority Manager).

In MQSeries for Compaq OpenVMS werden die Zeilengruppe AuthorizationService und die zugehörige Zeilengruppe ServiceComponent beim Erstellen des WS-Managers automatisch hinzugefügt, können aber überschrieben werden, indem vor dem Erstellen des WS-Managers der logische Name mqsnoaut definiert wird. (Weitere Informationen finden Sie unter „Objektberechtigungsmanager inaktivieren" auf Seite 85.) Jede andere ServiceComponent-Zeilengruppe muss manuell hinzugefügt werden.

NameService
Die Zeilengruppe NameService muss manuell in der Datei qm.ini hinzugefügt werden, um den bereitgestellten Namensservice zu aktivieren.

EntryPoints=*Anzahl der Eingangspunkte*
Gibt die Anzahl der Eingangspunkte an, die für den Service definiert werden. Dazu gehören die Eingangspunkte für die Initialisierung und die Beendigung.

Weitere Informationen zu installierbaren Services und Komponenten finden Sie im Handbuch *MQSeries Programmable System Management*.

Weitere Informationen zu Sicherheitsservices im Allgemeinen finden Sie in „Kapitel 7. MQSeries-Objekte schützen" auf Seite 81.

Die Zeilengruppe ServiceComponent

Die Zeilengruppe ServiceComponent gibt den Namen und Pfad des Moduls mit dem Code für die Komponente an.

Für jeden Service kann es mehrere ServiceComponent-Zeilengruppen geben, aber jede ServiceComponent-Zeilengruppe muss mit der entsprechenden Service-Zeilengruppe übereinstimmen.

In MQSeries for Compaq OpenVMS ist die Zeilengruppe AuthorizationService standardmäßig vorhanden, und die zugeordnete Komponente, der Objektberechtigungsmanager (OAM), ist aktiv.

Service=*Servicename*
Gibt den Namen des erforderlichen Services an. Dieser Name muss mit dem Wert des Attributs `Name` in der Zeilengruppe `Service` übereinstimmen.

Name=*Komponentenname*
Gibt den beschreibenden Namen der Servicekomponente an. Dieser Name muss eindeutig sein und darf nur gültige Zeichen für Namen von MQSeries-Objekten (z. B. Warteschlangen) enthalten. Dieser Name wird in Bedienernachrichten, die von dem Service generiert werden, angezeigt. Daher wird empfohlen, einen Namen zu wählen, der mit einem Firmennamen oder einer ähnlich eindeutigen Zeichenfolge beginnt.

Module=*Modulname*
Gibt den Namen des Moduls an, das den Code für diese Komponente enthält.

Anmerkung: Geben Sie einen vollständigen Pfadnamen an.

ComponentDataSize=*Größe*
Gibt die Größe des Komponentendatenbereichs in Bytes an, der bei jedem Aufruf an die Komponente übermittelt wird. Geben Sie null an, wenn keine Komponentendaten erforderlich sind.

Weitere Informationen zu installierbaren Services und Komponenten finden Sie im Handbuch *MQSeries Programmable System Management*.

Die Zeilengruppe Log

Die Zeilengruppe `Log` gibt die Protokollattribute für einen einzelnen WS-Manager an. Die Attribute werden beim Erstellen des WS-Managers standardmäßig aus den Einstellungen der Zeilengruppe `LogDefaults` in der Datei `mqs.ini` übernommen, sofern sie nicht durch Parameter im Befehl **crtmqm** überschrieben werden. Weitere Informationen finden Sie unter „Die Zeilengruppe LogDefaults" auf Seite 184 und unter „crtmqm (WS-Manager erstellen)" auf Seite 260.

Ändern Sie Attribute in dieser Zeilengruppe nur, wenn der betreffende WS-Manager anders als Ihre übrigen WS-Manager konfiguriert werden muss.

Die Werte der Attribute in der Datei `qm.ini` werden beim Start des WS-Managers gelesen. Die Datei wird zusammen mit dem WS-Manager erstellt.

Informationen zur Berechnung der Protokollgröße finden Sie unter „Protokollgröße berechnen" auf Seite 147.

Anmerkung: Bei den in der folgenden Parameterliste genannten Begrenzungen handelt es sich um MQSeries-Begrenzungen. Die maximal mögliche Protokollgröße kann durch Betriebssystembegrenzungen verringert werden.

LogPrimaryFiles=3|*2-62*
Primäre Protokolldateien sind die Protokolldateien, die bei der Erstellung für eine spätere Verwendung angelegt werden.

Es müssen mindestens zwei primäre Protokolldateien vorhanden sein, ihre maximale Anzahl beträgt 62. Der Standardwert ist 3.

Die Gesamtzahl der primären und sekundären Protokolldateien darf 63 nicht überschreiten und nicht kleiner als 3 sein.

Der Wert wird beim Erstellen bzw. Starten des WS-Managers überprüft. Sie können ihn ändern, nachdem der WS-Manager erstellt wurde. Eine Änderung des Wertes wird jedoch erst wirksam, wenn der WS-Manager erneut gestartet wird, und ist nicht sofort erkennbar.

LogSecondaryFiles=2|*1-61*
Sekundäre Protokolldateien werden angelegt, wenn der Speicherbereich in den primären Protokolldateien erschöpft ist.

Es muss mindestens eine sekundäre Protokolldatei vorhanden sein, die maximale Anzahl beträgt 61. Der Standardwert ist 2.

Die Gesamtzahl der primären und sekundären Protokolldateien darf 63 nicht überschreiten und nicht kleiner als 3 sein.

Der Wert wird beim Starten des WS-Managers überprüft. Sie können diesen Wert ändern, Änderungen werden jedoch erst wirksam, wenn der WS-Manager erneut gestartet wird, und sind möglicherweise auch dann nicht sofort erkennbar.

LogFilePages=*Anzahl*
Die Protokolldaten werden in einer Folge von Dateien, den Protokolldateien, gespeichert. Die Protokolldateigröße wird in Einheiten von 4-KB-Seiten angegeben.

In MQSeries for Compaq OpenVMS beträgt die Standardanzahl von Protokolldateien 1024, was einer Protokolldateigröße von 4 MB entspricht. Die minimale Anzahl von Protokolldateiseiten beträgt 64 und die maximale Anzahl 16384.

Anmerkung: Die Größe der Protokolldateien, die beim Erstellen des WS-Managers angegeben werden, können für einen vorhandenen WS-Manager nicht geändert werden.

LogType=CIRCULAR|LINEAR
Das Attribut `LogType` definiert den Protokolltyp (zyklisch oder linear), der vom WS-Manager verwendet werden soll. Sie können den zu verwendenden Protokolltyp nicht mehr ändern, nachdem der WS-Manager gestartet wurde. Informationen zum Erstellen eines WS-Managers mit dem gewünschten Protokolltyp finden Sie in der Beschreibung des Attributs LogType unter „Die Zeilengruppe LogDefaults" auf Seite 184.

CIRCULAR
Geben Sie diesen Wert an, wenn eine Wiederherstellung nach Neustart gestartet werden soll, bei der das Protokoll verwendet wird, um Transaktionen zurückzusetzen, die noch nicht abgeschlossen waren, als das System gestoppt wurde.

Eine ausführliche Erläuterung der zyklischen Protokollierung finden Sie unter „Zyklische Protokollierung" auf Seite 142.

LINEAR
Geben Sie diesen Wert an, wenn Sie beides verwenden wollen, sowohl die Wiederherstellung nach Neustart als auch die Wiederherstellung über

Datenträger bzw. Vorwärtswiederherstellung (Erstellung verloren gegangener oder beschädigter Daten durch erneute Ausführung der protokollierten Aktionen).

Eine ausführliche Erläuterung der linearen Protokollierung finden Sie unter „Lineare Protokollierung" auf Seite 143.

LogBufferPages=17|*4-32*
Die Speicherkapazität, die als Puffer zum Schreiben von Datensätzen zugeordnet wird, ist konfigurierbar. Die Größe der Puffer wird in Einheiten von 4-KB-Seiten angegeben.

Die minimale Anzahl an Pufferseiten beträgt 4 und die maximale Anzahl 32. Größere Puffer führen zu einem höheren Durchsatz, insbesondere bei sehr langen Nachrichten.

Die Standardanzahl an Pufferseiten beträgt 17, was 68 KB entspricht.

Der Wert wird beim Starten des WS-Managers überprüft und kann dabei jedes Mal erhöht oder verkleinert werden. Eine Änderung des Wertes wird jedoch erst wirksam, wenn der WS-Manager erneut gestartet wird.

LogPath=*Verzeichnisname*
Sie können das Verzeichnis für die Protokolldateien eines WS-Managers angeben. Das Verzeichnis muss sich auf einer lokalen Einheit befinden, für das der WS-Manager über Schreibzugriff verfügt, dabei sollte es sich vorzugsweise um ein anderes Laufwerk handeln als das, auf dem sich die Nachrichtenwarteschlangen befinden. Durch die Auswahl eines anderen Laufwerks erhöhen Sie den Schutz im Falle eines Systemfehlers.

Der Standardwert ist `MQS_ROOT:[MQM.LOG]`.

Sie können den Namen eines Verzeichnisses im Befehl **crtmqm** mit der Option -ld angeben. Beim Erstellen eines WS-Managers wird auch im WS-Manager-Verzeichnis ein Verzeichnis für die Speicherung der Protokolldateien erstellt. Der Name dieses Verzeichnisses basiert auf dem Namen des WS-Managers. Dadurch wird sichergestellt, dass der Protokolldateipfad eindeutig ist und der Name den Begrenzungen für die Länge von Verzeichnisnamen entspricht.

Wenn Sie im Befehl **crtmqm** nicht die Option -ld angeben, wird der Wert des Attributs `LogDefaultPath` in der Datei `mqs.ini` verwendet.

Anmerkung: In MQSeries for Compaq OpenVMS müssen die Benutzer-ID MQM und die Gruppe MQM über alle Berechtigungen für die Protokolldateien verfügen. Wenn Sie die Verzeichnisse dieser Dateien ändern, müssen Sie sich selbst diese Berechtigungen erteilen. Dies ist nicht erforderlich, wenn sich die Protokolldateien in den Standardverzeichnissen des Produkts befinden.

Die Zeilengruppe XAResourceManager

Die Zeilengruppe `XAResourceManager` gibt die Ressourcenmanager an, die an globalen Arbeitseinheiten, die vom WS-Manager koordiniert werden, teilnehmen.

Die Datei `qm.ini` muss eine `XAResourceManager`-Zeilengruppe für jedes Ressourcenmanagerexemplar enthalten, das an globalen Arbeitseinheiten teilnimmt; es werden keine Standardwerte aus `mqs.ini` übergeben.

Informationen zum Hinzufügen von `XAResourceManager`-Attributen zur Datei `qm.ini` finden Sie unter„Datenbankkoordinierung" auf Seite 126.

Name=*Name* **(obligatorisch)**
Dieses Attribut identifiziert das Exemplar des WS-Managers.

Der Wert `Name` darf maximal 31 Zeichen lang sein und muss innerhalb der Datei `qm.ini` eindeutig sein. Sie können den Namen des Ressourcenmanagers verwenden, so wie er in der zugehörigen XA-Switch-Strukur definiert wurde. Wenn Sie jedoch mehrere Exemplare desselben WS-Managers verwenden, müssen Sie für jedes Exemplar einen eigenen eindeutigen Namen wählen. Sie können die Eindeutigkeit sicherstellen, indem Sie zum Beispiel den Namen der Datenbank in die Zeichenfolge für das Attribut `Name` einschließen.

MQSeries verwendet den Wert `Name` in Nachrichten und in Ausgaben des Befehls **dspmqtrn**.

Es wird davon abgeraten, den Namen eines Ressourcenmanagerexemplars zu ändern oder dessen Eintrag aus der Datei `qm.ini` zu löschen, nachdem der zugehörige WS-Manager gestartet wurde und der Name des Ressourcenmanagers in Kraft ist.

SwitchFile=*Name* **(obligatorisch)**
Dieses Attribut gibt den vollständig qualifizierten Namen der Ladedatei mit der XA-Switch-Struktur des Ressourcenmanagers an.

XAOpenString=*Zeichenfolge* **(optional)**
Dieses Attribut gibt die Datenzeichenfolge an, die an den xa_open-Eingangspunkt des Ressourcenmanagers übergeben werden soll. Der Inhalt der Zeichenfolge ist vom Ressourcenmanager selbst abhängig. Die Zeichenfolge kann zum Beispiel die Datenbank identifizieren, auf die dieses Exemplar des Ressourcenmanagers zugreifen soll. Weitere Informationen zum Definieren dieses Attributs finden Sie unter „XAResourceManager-Konfigurationsdaten für Oracle hinzufügen" auf Seite 132 sowie in der Beschreibung der entsprechenden Zeichenfolge in der Dokumentation zu Ihrem Ressourcenmanager.

XACloseString=*Zeichenfolge* **(optional)**
Dieses Attribut gibt die Datenzeichenfolge an, die an den xa_close-Eingangspunkt des Ressourcenmanagers übergeben werden soll. Der Inhalt der Zeichenfolge ist vom Ressourcenmanager selbst abhängig. Weitere Informationen zum Definieren dieses Attributs finden Sie unter „XAResourceManager-Konfigurationsdaten für Oracle hinzufügen" auf Seite 132 sowie in der Beschreibung der entsprechenden Zeichenfolge in der Dokumentation zu Ihrer Datenbank.

ThreadOfControl=THREAD|PROCESS
Der Wert des Attributs ThreadOfControl wird vom WS-Manager für serielle Anordnungen verwendet, wenn er den Ressourcenmanager von einem seiner eigenen Multithread-Prozesse aufrufen muss.

THREAD
Bedeutet, dass der Ressourcenmanager vollständig 'Thread-sicher' ist. In einem MQSeries-Multithread-Prozess können XA-Funktionsaufrufe an den externen Ressourcenmanager von mehreren Threads gleichzeitig ausgegeben werden.

PROCESS
Bedeutet, dass der Ressourcenmanager nicht 'Thread-sicher' ist. In einem MQSeries-Multithread-Prozess kann zu einem Zeitpunkt immer nur ein XA-Funktionsaufruf an den Ressourcenmanager ausgegeben werden.

Das Attribut `ThreadOfControl` gilt nicht für XA-Funktionsaufrufe, die vom WS-Manager in einem Multithread-Anwendungsprozess ausgegeben werden.

Generell muss diese Betriebsart für eine Anwendung mit gleichzeitig in verschiedenen Threads ablaufenden Arbeitseinheiten von jedem einzelnen Ressourcenmanager unterstützt werden.

Die Zeilengruppe Channels

Die Zeilengruppe `Channels` enthält Informationen zu den Kanälen.

MaxChannels=100 | *Anzahl*
Dieses Attribut gibt die maximal zulässige Anzahl der Kanäle an. Der Standardwert ist 100.

MaxActiveChannels=*MaxChannels-Wert*
Dieses Attribut gibt die maximal zulässige Anzahl der Kanäle an, die gleichzeitig aktiv sein dürfen. Der Standardwert ist der für das Attribut `MaxChannels` angegebene Wert.

MaxInitiators=3 | *Anzahl*
Dieses Attribut gibt die maximale Anzahl der Initiatoren an.

MQIBINDTYPE=FASTPATH | STANDARD
Dieses Attribut gibt die Bindung für Anwendungen an.

FASTPATH
Kanäle verwenden MQCONNX FASTPATH für Verbindungen. Das heißt, es gibt keinen Agenten.

STANDARD
Kanäle verwenden die STANDARD-Bindung für Verbindungen.

AdoptNewMCA=NO | SVR | SDR | RCVR | CLUSRCVR | ALL | FASTPATH
Wenn MQSeries eine Anforderung zum Starten eines Kanals empfängt, aber feststellt, dass für denselben Kanal bereits ein amqcrsta-Prozess aktiv ist, muss dieser Prozess gestoppt werden, bevor der neue gestartet werden kann. Über das Attribut `AdoptNewMCA` können Sie die Beendigung eines aktiven Prozesses und den Start eines neuen Prozesses für einen angegebenen Kanaltyp steuern.

Wenn Sie das Attribut `AdoptNewMCA` für einen bestimmten Kanaltyp angeben, der neue Kanal aber nicht gestartet werden kann, weil der Kanal bereits aktiv ist, geschieht Folgendes:

1. Der neue Kanal versucht, den bereits aktiven Kanal zu stoppen, indem er ihn freundlich auffordert, sich selbst zu beenden.
2. Wenn das Warteintervall des Attributs AdoptNewMCATimeout abläuft, ohne dass der bereits aktive Kanalserver auf die Aufforderung reagiert hat, wird die Beendigung des Prozesses (oder Threads) für den bereits aktiven Kanalserver erzwungen.
3. Wenn der bereits aktive Kanalserver nach Schritt 2 und nach dem erneuten Ablauf des Warteintervalls von AdoptNewMCATimeout immer noch nicht beendet ist, schließt MQSeries den Kanal mit einer 'Kanal wird verwendet'-Fehlermeldung.

Sie können einen oder mehrere Werte (getrennt durch Komma oder Leerzeichen) aus der folgenden Liste angeben:
NO Das Attribut `AdoptNewMCA` ist nicht erforderlich. Dies ist die Standardeinstellung.
SVR Serverkanäle übernehmen
SDR Senderkanäle übernehmen
RCVR Empfängerkanäle übernehmen
CLUSRCVR
Cluster-Empfängerkanäle übernehmen

ALL Alle Kanaltypen außer FASTPATH-Kanäle übernehmen

FASTPATH
Den Kanal übernehmen, wenn es sich um einen FASTPATH-Kanal handelt. Dies ist nur der Fall, wenn auch der entsprechende Kanaltyp angegeben wurde, z. B. AdoptNewMCA=RCVR,SVR,FASTPATH.

> **Achtung!**
> Aufgrund der internen Struktur des WS-Managers kann das Attribut AdoptNewMCA in Verbindung mit FASTPATH-Kanälen zu unvorhersehbaren Ereignissen führen. Prüfen Sie daher genau, bevor Sie das Attribut `AdoptNewMCA` für FASTPATH-Kanäle aktivieren.

AdoptNewMCATimeout=60|1—3600
Dieses Attribut gibt die Zeitdauer in Sekunden an, die der neue Prozess auf die Beendigung des alten Prozesses warten soll. Geben Sie einen Wert (in Sekunden) von 1 bis 3600 an. Der Standardwert ist 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL
Über das Attribut `AdoptNewMCACheck` können Sie die Typüberprüfung angeben, die beim Aktivieren des Attributs `AdoptNewMCA` erforderlich ist. Es ist wichtig, dass Sie möglichst alle drei der unten genannten Überprüfungen ausführen, um die Kanäle vor einem versehentlichen oder beabsichtigten Beenden zu schützen. Sie sollten mindestens überprüfen, ob die Kanalnamen übereinstimmen.

Geben Sie mindestens einen der folgenden Werte ein (durch Komma oder Leerzeichen getrennt):

QM Das Empfangsprogramm überprüft, ob die WS-Manager-Namen übereinstimmen.

ADDRESS
Das Empfangsprogramm überprüft die Kommunikationsadresse (z. B. die TCP/IP-Adresse).

NAME
Das Empfangsprogramm überprüft, ob die Kanalnamen übereinstimmen.

ALL Das Empfangsprogramm überprüft die Kommunikationsadresse und ob die WS-Manager-Namen und die Kanalnamen übereinstimmen.

`AdoptNewMCACheck=NAME,ADDRESS` ist der Standardwert für FAP1, FAP2 und FAP3, während `AdoptNewMCACheck=NAME,ADDRESS,QM` der Standardwert für FAP4 und höher ist.

Die Zeilengruppen LU62 und TCP

Diese Zeilengruppen geben Konfigurationsparameter für Netzprotokolle an. Diese Parameter können zum Überschreiben der Standardwerte für Kanäle verwendet werden.

Anmerkung: Es müssen nur Attribute angegeben werden, die von den Standardwerten abweichen.

LU62

Folgende Attribute können angegeben werden:

TPName

Dieses Attribut gibt den TP-Namen an, der auf dem fernen System gestartet werden soll.

LocalLU

Dies ist der Name der logischen Einheit, die auf lokalen Systemen verwendet werden soll.

TCP

Folgende Attribute können angegeben werden:

Port=1414 | *Port-Nummer*

Dieses Attribut gibt die Standard-Port-Nummer (in Dezimalschreibweise) für TCP/IP-Sitzungen an. Die 'normale' Port-Nummer für MQSeries ist 1414.

KeepAlive=YES | **NO**

Über dieses Attribut schalten Sie die Funktion KeepAlive ein oder aus. KeepAlive=YES bewirkt, dass TCP/IP in regelmäßigen Abständen überprüft, ob das andere Ende der Verbindung noch aktiv ist. Wenn nicht, wird der Kanal geschlossen.

ListenerBacklog=Anzahl

Wenn der Empfang über TCP/IP erfolgt, wird eine maximale Anzahl an unbeantworteten Verbindungsanforderungen festgelegt. Dies kann dazu führen, dass ein *Rückstand* von Anforderungen vermutet wird, die am TCP/IP-Anschluss darauf warten, vom Empfangsprogramm abgeholt zu werden. Die Standardwerte für den Rückstand des Empfangsprogramms werden in Tabelle 8 gezeigt.

Tabelle 8. Standardwerte für unbeantwortete Verbindungsanforderungen (TCP)

Plattform	Standardwert für ListenerBacklog
OS/390	255
OS/2 Warp	10
Windows NT Server	100
Windows NT Workstation	5
AS/400	255
Sun Solaris	100
HP-UX	20
AIX V4.2 oder höher	100
AIX V4.1 oder früher	10
Alle anderen Plattformen	5

Wenn der Rückstand die in Tabelle 8 auf Seite 194 gezeigten Werte erreicht, wird die TCP/IP-Verbindung zurückgewiesen, und der Kanal kann nicht gestartet werden.

Bei MCA-Kanälen führt dies dazu, dass der Kanal in den Status RETRY übergeht und zu einem späteren Zeitpunkt versucht, die Verbindung wiederherzustellen.

Bei Client-Verbindungen empfängt der Client einen Ursachencode MQRC_Q_MGR_NOT_AVAILABLE von MQCONN und versucht in der Regel zu einem späteren Zeitpunkt, die Verbindung wiederherzustellen.

Über das Attribut `ListenerBacklog` können Sie den Standardwert für die Anzahl der unbeantworteten Anforderungen für das TCP/IP-Empfangsprogramm überschreiben.

Anmerkung: Einige Betriebssysteme unterstützen größere Werte als die angegebenen Standardwerte. Bei Bedarf kann auf diese Weise verhindert werden, dass die Anzahl der maximalen Verbindungen erreicht wird.

Die Zeilengruppe ExitPath

ExitDefaultPath=*Zeichenfolge*

Das Attribut ExitDefaultPath gibt die Adresse folgender Exits an:

- Kanal-Exits für Clients
- Kanal-Exits und Datenkonvertierungs-Exits für Server

Der Exit-Pfad für Clients wird aus der Zeilengruppe ClientExitPath in der Datei `mqs.ini` und für Server aus dieser Zeilengruppe (ExitPath) gelesen.

Beispiele für die Dateien mqs.ini und qm.ini

Abb. 19 zeigt ein Beispiel für eine Datei `mqs.ini` in MQSeries for Compaq OpenVMS.

```
#********************************************************************#
#* Modulname  : mqs.ini                                             *#
#* Typ        : MQSeries-Konfigurationsdatei                        *#
#* Funktion   : MQSeries-Ressourcen für den Knoten definieren       *#
#*                                                                  *#
#********************************************************************#
#* Hinweise   :                                                     *#
#* 1) Dies ist ein Beispiel für eine MQSeries-Konfigurationsdatei.  *#
#*                                                                  *#
#********************************************************************#
AllQueueManagers:
   #****************************************************************#
   #* Der Pfad für das Verzeichnis qmgrs, in dem WS-Manager-Daten   *#
   #* gespeichert sind                                             *#
   #****************************************************************#
   DefaultPrefix=mqs_root:[mqm]

ClientExitPath:
   ExitsDefaultPath=mqs_root:[mqm.exits]

LogDefaults:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=1024
   LogType=CIRCULAR
   LogBufferPages=17
   LogDefaultPath=mqs_root:[mqm.log]
QueueManager:
   Name=saturn.queue.manager
   Prefix=mqs_root:[mqm]
   Directory=saturn$queue$manager
DefaultQueueManager:
   Name=saturn.queue.manager
QueueManager:
   Name=pluto.queue.manager
   Prefix=mqs_root:[mqm]
   Directory=pluto$queue$manager
```

Abbildung 19. Beispiel für eine MQSeries-Konfigurationsdatei für MQSeries for Compaq OpenVMS-Systeme

Abb. 20 auf Seite 197 zeigt, wie Attributgruppen in einer WS-Manager-Konfigurationsdatei in MQSeries for Compaq OpenVMS angeordnet werden können.

```
#*******************************************************************#
#* Modulname  : qm.ini                                             *#
#* Typ        : Konfigurationsdatei für MQSeries-WS-Manager        *#
#  Funktion   : Konfiguration eines WS-Managers definieren         *#
#*                                                                 *#
#*******************************************************************#
#* Hinweise   :                                                    *#
#* 1) Diese Datei definiert die Konfiguration des WS-Managers.     *#
#*                                                                 *#
#*******************************************************************#
ExitPath:
   ExitsDefaultPath=mqm_root:[mqm.exits]

Service:
   Name=AuthorizationService
   EntryPoints=9

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=amqzfu
   ComponentDataSize=0

Service:
   Name=NameService
   EntryPoints=5

ServiceComponent:
   Service=NameService
   Name=MQSeries.DCE.name.service
   Module=amqzfa
   ComponentDataSize=0

Log:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=1024
   LogType=CIRCULAR
   LogBufferPages=17
   LogPath=mqm_root:[mqm.log.saturn$queue$manager]

XAResourceManager:
  Name=Oracle Resource Manager Bank
  SwitchFile=sys$share:oraswit0.exe
  XAOpenString=MQBankDB
  XACloseString=
  ThreadOfControl=PROCESS


CHANNELS:
  MaxChannels = 20         ; Maximal zulässige Anzahl der Kanäle
                           ; Standardwert ist 100
  MaxActiveChannels = 10   ; Maximal zulässige Anzahl gleichzeitig
                           ; aktiver Kanäle. Der Standardwert ist
                           ; der Wert für MaxChannels.

TCP:                       ; TCP/IP-Einträge
  KeepAlive = Yes          ; KeepAlive einschalten
```

Abbildung 20. Beispiel für eine WS-Manager-Konfigurationsdatei für MQSeries for Compaq OpenVMS

MQSeries-Konfigurationsdatei ändern

Hinweise:

MQSeries auf dem Knoten verwendet die Standardverzeichnisse für WS-Manager und für die Protokolle.

Der WS-Manager saturn.queue.manager ist der Standard-WS-Manager für den Knoten. Der Name des Verzeichnisses für Dateien dieses WS-Managers wurde automatisch in einen gültigen Dateinamen für das OpenVMS-Dateisystem umgewandelt.

Da die MQSeries-Konfigurationsdatei zum Lokalisieren der Daten des WS-Managers verwendet wird, kann eine fehlende oder falsche Konfigurationsdatei zur Folge haben, dass einige oder alle MQSeries-Befehle fehlschlagen. Auch können Anwendungen nur Verbindungen mit WS-Managern herstellen, die in der MQSeries-Konfigurationsdatei definiert sind.

Kapitel 14. Fehlerbestimmung

Dieses Kapitel enthält Vorschläge zur Behandlung von Fehlern, die möglicherweise bei der Verwendung von MQSeries for Compaq OpenVMS auftreten.

Nicht alle Fehler können sofort behoben werden, so können Leistungsprobleme zum Beispiel durch Einschränkungen der vorhandenen Hardware verursacht werden. Wenn Sie glauben, dass der Fehler durch den MQSeries-Code verursacht wird, wenden Sie sich an das IBM Support Center. Dieses Kapitel enthält folgende Abschnitte:

Erste Überprüfungen

Fehler in Verbindung mit MQSeries werden in den meisten Fällen durch folgende Komponenten verursacht:

- MQSeries
- das Netzwerk
- die Anwendung
- das zu Grunde liegende Betriebssystem

In den folgenden Abschnitten werden einige grundlegende Fragen behandelt, die Sie bei der Fehlersuche berücksichtigen sollten.

Lief MQSeries bisher fehlerfrei?

Wenn MQSeries auch vorher nicht fehlerfrei lief, wurde es möglicherweise nicht korrekt installiert und konfiguriert. Lesen Sie im Handbuch *MQSeries for Compaq OpenVMS Alpha, Version 5.1 Einstieg* nach, um zu überprüfen, ob MQSeries korrekt installiert und konfiguriert wurde.

Werden Fehlernachrichten angezeigt?

MQSeries erstellt Fehlerprotokolle mit Nachrichten über den Betrieb von MQSeries selbst, über alle gestarteten WS-Manager und über fehlerhafte Daten von den verwendeten Kanälen. Überprüfen Sie die Fehlerprotokolle auf Nachrichten, die dem jeweiligen Fehler zugeordnet werden können.

Informationen zum Inhalt der Fehlerprotokolle und zu den Protokollverzeichnissen finden Sie unter „Fehlerprotokolle" auf Seite 210.

Werden Rückkehrcodes mit Fehlererläuterungen angezeigt?

Wenn die Anwendung einen Rückkehrcode erhält, der auf einen Fehler eines MQI-Aufrufs (Message Queue Interface) hinweist, lesen Sie die Beschreibung des betreffenden Rückkehrcodes im Handbuch *MQSeries Application Programming Reference*.

Können Sie den Fehler reproduzieren?

Wenn Sie den Fehler reproduzieren können, überprüfen Sie die Bedingungen, unter denen Sie ihn reproduzieren konnten:

- Wurde er durch einen Befehl oder eine ähnliche Verwaltungsanforderung verursacht?

  Wird die Operation ausgeführt, wenn sie auf eine andere Weise eingegeben wird? Wird der Befehl nur dann ausgeführt, wenn Sie ihn über die Befehlszeile eingeben, überprüfen Sie, ob der Befehlsserver noch aktiv ist und ob die Warteschlangendefinition SYSTEM.ADMIN.COMMAND.QUEUE unverändert ist.
- Wird er durch ein Programm verursacht? Tritt der Fehler auf allen MQSeries-Systemen und WS-Managern auf oder nur auf einigen?
- Gibt es Anwendungen, die immer dann auf dem System aktiv sind, wenn der Fehler auftritt? Wenn ja, überprüfen Sie, ob die Anwendung fehlerfrei läuft.

Trat der Fehler erst auf, nachdem Änderungen vorgenommen wurden?

Berücksichtigen Sie beim Überprüfen der Änderungen, die seit dem letzten fehlerfreien Betrieb vorgenommen wurden, das MQSeries-System einschließlich der anderen Programme, mit denen es kommuniziert, sowie die Hardware und alle neuen Anwendungen. Überprüfen Sie auch, ob möglicherweise ohne Ihr Wissen eine neue Anwendung auf dem System ausgeführt wurde.

- Haben Sie Warteschlangendefinitionen geändert, hinzugefügt oder gelöscht?
- Haben Sie Kanaldefinitionen geändert oder hinzugefügt? Änderungen können an MQSeries-Kanaldefinitionen oder an zu Grunde liegenden Kommunikationsdefinitionen, die für eine Anwendung benötigt werden, vorgenommen worden sein.
- Bearbeiten Ihre Anwendungen Rückkehrcodes, die sie möglicherweise als Ergebnis von Änderungen, die Sie vorgenommenen haben, erhalten?

Lief die Anwendung bisher fehlerfrei?

Wenn der Fehler offenbar nur in Verbindung mit einer bestimmten Anwendung auftritt, überprüfen Sie, ob die Anwendung vorher fehlerfrei lief.

Bevor Sie diese Frage mit **Ja** beantworten, überprüfen Sie Folgendes:

- Trat der Fehler erst auf, nachdem die Anwendung geändert wurde?

  Wenn ja, liegt die Ursache vermutlich in dem neuen oder geänderten Teil der Anwendung. Überprüfen Sie, ob die Änderungen eine erkennbare Ursache für den Fehler enthalten. Tritt der Fehler auch dann auf, wenn Sie einen früheren Stand der Anwendung verwenden?
- Wurden alle Funktionen der Anwendung vorher im vollen Umfang benutzt?

  Trat der Fehler möglicherweise erst auf, als ein bisher nie benutzter Teil der Anwendung zum ersten Mal aufgerufen wurde? Wenn ja, wird der Fehler vermutlich durch diesen Teil der Anwendung verursacht. Finden Sie heraus, was die Anwendung machte, als der Fehler auftrat, und überprüfen Sie den Quellcode im betreffenden Teil des Programms auf Fehler.

Wenn ein Programm vorher bei vielen Gelegenheiten fehlerfrei ausgeführt wurde, überprüfen Sie den Status der aktuellen Warteschlange sowie die Dateien, die verarbeitet wurden, als der Fehler auftrat. Möglicherweise enthalten Sie ungewöhnliche Datenwerte, durch die eine nur selten verwendete Verzweigung innerhalb des Programms aufgerufen wird.

- Überprüft die Anwendung alle Rückkehrcodes?

  Möglicherweise wurde Ihr MQSeries-System geändert, vielleicht nur ganz geringfügig, aber mit der Folge, dass Ihre Anwendung die Rückkehrcodes, die sie als Ergebnis der Änderung empfängt, nicht überprüft. Geht Ihre Anwendung zum Beispiel davon aus, dass die Warteschlangen, auf die sie zugreift, gemeinsam benutzt werden können? Wenn eine Warteschlange erneut definiert wurde, diesmal aber als exklusive Warteschlange, kann Ihre Anwendung Rückkehrcodes verarbeiten, die ihr anzeigen, dass ein Zugriff auf diese Warteschlange nicht mehr möglich ist.
- Kann die Anwendung auf anderen MQSeries-Systemen ausgeführt werden?

  Kann es sein, dass sich die Konfiguration des MQSeries-Systems, auf dem der Fehler auftritt, von der Konfiguration des anderen Systems unterscheidet? Wurde für die Warteschlangen zum Beispiel dieselbe Nachrichtenlänge oder -priorität definiert?

Die Anwendung lief bisher nicht fehlerfrei

Wenn die Anwendung auch bisher nicht fehlerfrei lief, müssen Sie sie genau auf mögliche Fehler überprüfen.

Bevor Sie den Code überprüfen und je nachdem, in welcher Programmiersprache der Code geschrieben wurde, überprüfen Sie zunächst die Ausgabe des Umsetzungsprogramms bzw. Compilers und Verbindungseditors (sofern zutreffend), um zu sehen, ob Fehler gemeldet wurden.

Wenn die Anwendung nicht von einem Umsetzungsprogramm, Compiler oder Verbindungseditor in eine Ladebibliothek gestellt wurde, können Sie es nicht erfolgreich aufrufen. Informationen zum Erstellen einer Anwendung finden Sie im Handbuch *MQSeries Application Programming Reference*.

Wenn die Dokumentation belegt, dass alle diese Schritte ohne Fehler ausgeführt wurden, sollten Sie die Codierlogik der Anwendung überprüfen. Geben die Fehlersymptome einen Hinweis auf die fehlerhafte Funktion und damit auf den fehlerhaften Code der Anwendung? Unter „Typische Programmierfehler" auf Seite 202 finden Sie einige Beispiele für typische Fehler, die Probleme in Verbindung mit MQSeries-Anwendungen verursachen.

Betrifft der Fehler bestimmte Teile des Netzes?

Möglicherweise können Sie erkennen, ob bestimmte Teile des Netzes von dem Fehler betroffen sind (z. B. ferne Warteschlangen). Wenn die Verbindung mit dem WS-Manager einer fernen Warteschlange nicht aktiv ist, können die Nachrichten nicht an die ferne Warteschlange übertragen werden.

Überprüfen Sie, ob die Verbindung zwischen den beiden Systemen verfügbar ist und die Datenübertragungskomponente von MQSeries gestartet wurde.

Überprüfen Sie, ob Nachrichten in der Übertragungswarteschlange ankommen, sowie die lokale Warteschlangendefinition der Übertragungswarteschlange und aller fernen Warteschlangen.

Haben Sie irgendwelche das Netz betreffende Änderungen vorgenommen oder irgendwelche MQSeries-Definitionen geändert, die den Fehler verursacht haben könnten?

Tritt der Fehler zu einer bestimmten Tageszeit auf?

Wenn der Fehler zu einer bestimmten Tageszeit auftritt, können Ladezeiten des Systems die Ursache sein. In der Regel ist ein System in der Mitte des Vormittags und in der Mitte des Nachmittags besonders stark durch Ladezeiten ausgelastet, so dass dies die Zeiten sind, in denen am häufigsten ladezeitabhängige Fehler auftreten. (Wenn sich Ihr MQSeries-Netz über mehrere Zeitzonen erstreckt, können sich diese Hauptladezeiten auf eine andere Zeit des Tages verschieben.)

Tritt der Fehler unregelmäßig auf?

Ein unregelmäßig auftretender Fehler kann dadurch verursacht werden, dass nicht berücksichtigt wird, dass Prozesse unabhängig voneinander ablaufen können. Zum Beispiel gibt ein Programm möglicherweise einen MQGET-Aufruf ohne Angabe einer WAIT-Option aus, bevor ein älterer Prozess beendet wurde. Ein unregelmäßiger Fehler kann auch dann auftreten, wenn Ihre Anwendung versucht, eine Nachricht aus einer Warteschlange abzurufen, obwohl der Aufruf zum Einreihen dieser Nachricht noch nicht bestätigt wurde (d. h., er wurde noch nicht festgeschrieben oder zurückgesetzt).

Haben Sie Funktionsaktualisierungen installiert?

Wenn eine Funktionsaktualisierung für MQSeries installiert wurde, überprüfen Sie, ob der Aktualisierungsvorgang erfolgreich abgeschlossen wurde und ob dabei ein Fehler aufgetreten ist.

- Gab es für die Aktualisierung irgendwelche besonderen Anweisungen?
- Wurden Tests ausgeführt, um zu überprüfen, ob die Aktualisierung korrekt und vollständig installiert wurde?
- Tritt der Fehler auch dann noch auf, wenn die vorherige Servicestufe von MQSeries wiederhergestellt wurde?
- Wenn die Installation erfolgreich war, wenden Sie sich an das IBM Support Center, um zu überprüfen, ob es Programmkorrekturen gibt.
- Wenn für ein anderes Programm eine Programmkorrektur installiert wurde, überprüfen Sie, ob dies Auswirkungen auf die Kommunikation zwischen MQSeries und dem betreffenden Programm haben kann.

Müssen Sie sonstige Aktualisierungen installieren?

MQSeries setzt auf das zu Grunde liegende Betriebssystem (OpenVMS) und auf verschiedene Netzprodukte, z. B. TCP/IP, auf. Wenden Sie sich an den jeweiligen Lieferanten, um sicherzustellen, dass Sie alle erforderlichen Funktionsaktualisierungen für diese Produkte installiert haben.

Typische Programmierfehler

In der folgenden Liste sind die am häufigsten vorkommenden Ursachen für Fehler bei der Ausführung von MQSeries-Programmen aufgeführt. Überprüfen Sie, ob der Fehler auf Ihrem MQSeries-System möglicherweise auf eine oder mehrere der hier aufgelisteten Ursachen zurückgeführt werden kann:

- Warteschlangen werden als gemeinsame Warteschlangen behandelt, obwohl es sich in Wirklichkeit um exklusive Warteschlangen handelt.
- In einem MQI-Aufruf werden falsche Parameter übergeben.

- In einem MQI-Aufruf werden zu wenige Parameter übergeben. Dies kann bedeuten, dass MQI keine Beendigungs- oder Ursachencodes definieren kann, die von der Anwendung verarbeitet werden können.
- Rückkehrcodes von MQI-Anforderungen werden nicht überprüft.
- Es werden Variablen mit falschen Längenangaben übergeben.
- Parameter werden in einer falschen Reihenfolge übergeben.
- *MsgId* (Nachrichten-ID) und *CorrelId* (Korrelations-ID) werden nicht korrekt initialisiert.

Die nächsten Schritte

Möglicherweise haben Sie die Ursache des Fehlers bereits bei den ersten Überprüfungen gefunden. In diesem Fall sollten Sie den Fehler jetzt beheben können, gegebenenfalls mit Hilfe anderer Bücher aus der MQSeries-Bibliothek (siehe Bibliographie) und aus den Bibliotheken anderer Lizenzprogramme.

Wenn Sie die Ursache noch nicht gefunden haben, müssen Sie den Fehler genauer untersuchen.

Dieser Abschnitt soll Ihnen dabei helfen, die Fehlerursache zu finden, wenn die ersten Überprüfungen ohne Ergebnis geblieben sind.

Nachdem Sie festgestellt haben, dass Ihr System nicht geändert wurde und es keine Probleme mit Ihren Anwendungsprogrammen gibt, wählen Sie jetzt die Option aus, deren Beschreibung am besten auf die Symptome des Fehlers zutrifft.
- „Haben Sie falsche Ausgaben erhalten?"
- „Haben Sie von einem PCF-Befehl keine Antwort erhalten?"
- „Betrifft der Fehler nur ferne Warteschlangen?" auf Seite 205

Sollte keines der beschriebenen Symptome auf den Fehler zutreffen, überprüfen Sie, ob er möglicherweise durch eine andere Komponente auf Ihrem System verursacht wurde.

Haben Sie falsche Ausgaben erhalten?

Mit „falsche Ausgaben" ist in diesem Buch Folgendes gemeint:
- Die Anwendung hat nicht die erwartete Nachricht empfangen.
- Die Anwendung hat eine Nachricht mit unerwarteten oder beschädigten Informationen empfangen.
- Die Anwendung hat eine Nachricht empfangen, die sie nicht erwartet hatte, z. B. eine Nachricht, die an eine andere Anwendung gerichtet war.

Überprüfen Sie in allen Fällen, ob die von der Anwendung verwendeten Aliasnamen von Warteschlangen oder WS-Managern richtig angegeben wurden und ob sie den Änderungen, die im Netz vorgenommen wurden, entsprechen.

Wenn MQSeries-Fehlernachrichten (beginnen alle mit dem Präfix „AMQ") generiert werden, sollten Sie im Fehlerprotokoll nachschauen. Weitere Informationen finden Sie unter „Fehlerprotokolle" auf Seite 210.

Haben Sie von einem PCF-Befehl keine Antwort erhalten?

Wenn Sie einen Befehl ausgegeben, darauf aber keine Antwort erhalten haben, überprüfen Sie die folgenden Fragen:
- Ist der Befehlsserver aktiv?

  Überprüfen Sie mit Hilfe des Befehls **dspmqcsv** den Status des Befehlsservers.

  - Wenn die Antwort auf diesen Befehl anzeigt, dass der Befehlsserver nicht aktiv ist, starten Sie ihn mit dem Befehl **strmqcsv**.
  - Wenn die Antwort auf den Befehl anzeigt, dass die Warteschlange SYSTEM.ADMIN.COMMAND.QUEUE nicht für MQGET-Anforderungen aktiviert ist, aktivieren Sie sie für MQGET-Anforderungen.
- Wurde eine Antwort in die Warteschlange für nicht zustellbare Nachrichten (DLQ) gestellt?

  Die Header-Struktur der Warteschlange für nicht zustellbare Nachrichten enthält einen Ursachen- bzw. Rückgabecode mit einer Beschreibung des Fehlers. Informationen zur Header-Struktur (MQDLH) der Warteschlange für nicht zustellbare Nachrichten finden Sie im Handbuch *MQSeries Application Programming Reference*.

  Wenn sich Nachrichten in der Warteschlange für nicht zustellbare Nachrichten befinden, können Sie die Nachrichten mit Hilfe der bereitgestellten Beispielanwendung (amqsbcg) und des Aufrufs MQGET durchsuchen. Die Beispielanwendung durchsucht alle Nachrichten in einer angegebenen Warteschlange auf einem angegebenen WS-Manager und zeigt den Nachrichtendeskriptor und die Felder mit dem Nachrichtenkontext aller Nachrichten in der angegebenen Warteschlange an.
- Wurde eine Nachricht in das Fehlerprotokoll gestellt?

  Weitere Informationen finden Sie unter „Fehlerprotokolle" auf Seite 210.
- Wurden die Warteschlangen für PUT- und GET-Operationen aktiviert?
- Ist das Warteintervall (*WaitInterval*) lang genug?

  Wenn Ihr MQGET-Aufruf das zulässige Zeitlimit überschritten hat, werden der Beendigungscode MQCC_FAILED und der Ursachencode MQRC_NO_MSG_AVAILABLE zurückgegeben. (Informationen zum Feld *WaitInterval* sowie zu Beendigungs- und Ursachencodes des Aufrufs MQGET finden Sie im Handbuch *MQSeries Application Programming Reference*).
- Wenn Sie ein eigenes Anwendungsprogramm verwenden, um Befehle in die Warteschlange SYSTEM.ADMIN.COMMAND.QUEUE einzureihen, müssen Sie dann einen Synchronisationspunkt festlegen?

  Sofern Sie Ihre Anforderungsnachricht nicht explizit vom Synchronisationspunkt ausgeschlossen haben, müssen Sie einen Synchronisationspunkt festlegen, bevor Sie versuchen, Antwortnachrichten zu empfangen.
- Sind die Werte für die Attribute MAXDEPTH und MAXMSGL für Ihre Warteschlangen hoch genug?
- Verwenden Sie die Felder *CorrelId* und *MsgId* richtig?

  Geben Sie die Werte *MsgId* und *CorrelId* in Ihrer Anwendung an, um sicherzustellen, dass Sie alle Nachrichten aus der Warteschlange empfangen.

Versuchen Sie, den Befehlsserver zu stoppen und dann erneut zu starten, wobei Sie auf alle erstellten Fehlernachrichten antworten.

Wenn das System immer noch nicht antwortet, kann der Fehler an einem einzelnen WS-Manager oder aber am gesamten MQSeries-System liegen. Stoppen Sie zunächst die einzelnen WS-Manager, um einen fehlerhaften WS-Manager zu finden und zu isolieren. Wird der Fehler dadurch nicht behoben, versuchen Sie, MQSeries zu stoppen und erneut zu starten, wobei Sie auf alle Nachrichten, die im Fehlerprotokoll erstellt werden, antworten.

Bleibt der Fehler auch nach dem Neustart bestehen, fordern Sie Hilfe vom zuständigen IBM Support Center an.

Sind einige Ihrer Warteschlangen fehlerhaft?

Wenn Sie vermuten, dass der Fehler nur einen Teil der Warteschlangen betrifft, überprüfen Sie diejenigen lokalen Warteschlangen, die Ihrer Meinung nach fehlerhaft sind:

1. Zeigen Sie die Informationen zu jeder einzelnen Warteschlange an. Verwenden Sie dazu den MQSC-Befehl DISPLAY QUEUE.
2. Überprüfen Sie Folgendes anhand der angezeigten Daten:
   - Wenn der Wert für CURDEPTH den Wert für MAXDEPTH erreicht hat, bedeutet dies, dass die Warteschlange nicht verarbeitet werden kann. Überprüfen Sie, ob alle Anwendungen normal laufen.
   - Wenn der Wert für CURDEPTH den Wert für MAXDEPTH noch nicht erreicht hat, überprüfen Sie, ob die folgenden Warteschlangenattribute korrekt sind:
     – Bei Verwendung der Auslösefunktion:
       - Ist der Auslösemonitor aktiv?
       - Ist die Auslöseschwelle zu hoch? Das heißt, werden Auslöseereignisse häufig genug generiert?
       - Ist der Prozessname richtig?
       - Ist der Prozess verfügbar und betriebsbereit?
     – Kann die Warteschlange gemeinsam benutzt werden? Wenn nicht, wurde sie möglicherweise bereits von einer anderen Anwendung für eine Eingabe geöffnet.
     – Wurde die Warteschlange wie erforderlich für GET- und PUT-Aufrufe aktiviert?
   - Wenn keine Anwendungsprozesse Nachrichten aus der Warteschlange abrufen können, stellen Sie den Grund dafür fest. Es kann daran liegen, dass die Anwendungen gestartet werden müssen, eine Verbindung unterbrochen wurde oder der Aufruf MQOPEN aus irgendeinem Grund fehlgeschlagen ist.

     Überprüfen Sie die Attribute IPPROCS und OPPROCS. Diese Attribute zeigen an, ob die Warteschlange für Ein- bzw. Ausgaben geöffnet wurde. Wenn ein Attribut den Wert null hat, bedeutet dies, dass Operationen des betreffenden Typs nicht ausgeführt werden können. Beachten Sie, dass die Werte sich geändert haben können und die Warteschlange geöffnet war, aber inzwischen wieder geschlossen wurde.

     Sie müssen den Status zu dem Zeitpunkt überprüfen, an dem Sie eine Nachricht einreihen oder abrufen wollen.

Wenn Sie den Fehler nicht beheben können, fordern Sie Hilfe vom zuständigen IBM Support Center an.

Betrifft der Fehler nur ferne Warteschlangen?

Wenn der Fehler nur ferne Warteschlangen betrifft, überprüfen Sie Folgendes:

- Überprüfen Sie, ob erforderliche Kanäle gestartet wurden und ausgelöst werden können und ob alle erforderlichen Initiatoren aktiv sind.
- Überprüfen Sie, ob die Programme, die Nachrichten in die ferne Warteschlange einreihen sollen, Fehler gemeldet haben.
- Wenn Sie die Auslösefunktion zum Starten des verteilten Queuing verwenden, überprüfen Sie, ob die Auslösefunktion für die Übertragungswarteschlange aktiviert ist. Überprüfen Sie außerdem, ob der Kanalinitiator aktiv ist.
- Überprüfen Sie die Fehlerprotokolle auf Nachrichten, die auf Kanalfehler oder -probleme hinweisen.

Nächste Schritte

- Starten Sie den Kanal gegebenenfalls manuell. Informationen hierzu finden Sie im Handbuch *MQSeries Intercommunication*.

Informationen zum Definieren von Kanälen finden Sie im Handbuch *MQSeries Intercommunication*.

Überlegungen zum Anwendungsdesign

Es gibt eine Reihe von Möglichkeiten, wie sich ein schlechter Programmentwurf auf die Leistung auswirken kann. Dies ist nicht immer sofort erkennbar, weil das Programm scheinbar problemlos läuft, gleichzeitig aber die Leistung anderer Tasks beeinträchtigen kann. Einige Probleme, die Programme mit MQSeries-Aufrufen betreffen, werden in den folgenden Abschnitten behandelt.

Weitere Informationen zum Anwendungsdesign finden Sie im Handbuch *MQSeries Application Programming Guide*.

Auswirkung der Nachrichtenlänge

Nachrichten in MQSeries können zwar bis zu 100 MB Daten umfassen, die Datenmenge in einer Nachricht hat jedoch Auswirkungen auf die Leistung der Anwendung, von der die Nachricht verarbeitet wird. Um die Leistung Ihrer Anwendung zu optimieren, sollten Sie nur die wirklich benötigten Daten in einer Nachricht senden; so müssen zum Beispiel in einer Anforderung zur Belastung eines Kontos möglicherweise nur die Kontonummer und der Belastungsbetrag als wirklich benötigte Informationen von der Client- an die Serveranwendung übergeben werden.

Auswirkung der Nachrichtenpermanenz

Permanente Nachrichten werden protokolliert. Die Protokollierung von Nachrichten wirk sich negativ auf die Leistung Ihrer Anwendung aus, daher sollten Sie permanente Nachrichten nur für wichtige Daten verwenden. Wenn die Daten in einer Nachricht nach einem Stopp oder einer Fehlfunktion des WS-Managers gelöscht werden können, verwenden Sie eine nicht permanente Nachricht.

Nach einer bestimmten Nachricht suchen

Der Aufruf MQGET ruft in der Regel die erste Nachricht in einer Warteschlange ab. Wenn Sie im Nachrichtendeskriptor die Nachrichten-ID und die Korrelations-ID (*MsgId* und *CorrelId*) für eine bestimmte Nachricht angeben, muss der WS-Manager die Warteschlange nach dieser Nachricht durchsuchen. Ein solche Verwendung des Aufrufs MQGET wirkt sich auf die Leistung Ihrer Anwendung aus.

Warteschlangen mit Nachrichten in unterschiedlicher Länge

Wenn die Nachrichten in einer Warteschlange unterschiedlich lang sind, kann die Anwendung die Länge einer Nachricht feststellen, indem sie den Aufruf MQGET mit dem Feld *BufferLength* gleich null verwendet; der Aufruf schlägt zwar fehl, er gibt aber die Größe der Nachrichtendaten zurück. Die Anwendung kann den Aufruf anschließend wiederholen, wobei sie die ID der Nachricht, die mit dem ersten Aufruf 'gemessen' wurde, und einen Puffer mit der passenden Größe angibt. Wenn andere Anwendungen auf dieselbe Warteschlange zugreifen, stellen Sie jedoch möglicherweise fest, dass die Leistung Ihrer Anwendung nachlässt, weil der zweite MQGET-Aufruf damit beschäftigt ist, nach einer Nachricht zu suchen, die zwischenzeitlich (zwischen dem ersten und zweiten Aufruf) von einer anderen Anwendung abgerufen wurde.

Wenn Ihre Anwendung keine Nachrichten mit fester Länge verwenden kann, können Sie das Problem beheben, indem Sie mit dem Aufruf MQINQ nach der maximalen Größe von Nachrichten in der Warteschlange suchen und diesen Wert dann in Ihrem MQGET-Aufruf verwenden. Die maximale Größe von Nachrichten in einer Warteschlange wird im Attribut *MaxMsgLength* der Warteschlange gespeichert. Bei dieser Methode werden jedoch sehr große Speicherbereiche belegt, weil dieses Warteschlangenattribut den maximal in MQSeries for Compaq OpenVMS zulässigen Wert von 100 MB enthalten kann.

Häufigkeit von Synchronisationspunkten

Programme, die zwischen Synchronisationspunkten zahlreiche MQPUT-Aufrufe ausgeben, ohne sie festzuschreiben, können Leistungsprobleme verursachen. Davon betroffene Warteschlangen können mit Nachrichten gefüllt werden, auf die in dieser Zeit nicht zugegriffen werden kann, die aber gleichzeitig möglicherweise von anderen Tasks dringend erwartet werden. Dies hat zum einen Auswirkungen auf den Hauptspeicher und zum anderen auf Threads, die durch Tasks blockiert werden, die vergeblich versuchen, Nachrichten abzurufen.

Den Aufruf MQPUT1 verwenden

Verwenden Sie den Aufruf MQPUT1 nur, wenn Sie eine einzelne Nachricht in eine Warteschlange einreihen müssen. Verwenden Sie für mehrere Nachrichten den Aufruf MQOPEN, gefolgt von einer Serie von MQPUT-Aufrufen und einem einzelnen MQCLOSE-Aufruf.

Falsche Ausgabe

Der Begriff 'Falsche Ausgabe' kann auf unterschiedliche Weise interpretiert werden. Seine Bedeutung im Rahmen der Fehlerbestimmung in diesem Buch wird unter „Haben Sie falsche Ausgaben erhalten?" auf Seite 203 erklärt.

In diesem Abschnitt werden zwei Arten der falschen Ausgabe behandelt:

- Nachrichten erscheinen wider Erwarten nicht in der Warteschlange.
- Nachrichten enthalten falsche Informationen oder Informationen, die beschädigt wurden.

Darüber hinaus werden weitere Fehler behandelt, die in einer Anwendung auftreten können, die verteilte Warteschlangen verwendet.

Nachrichten erscheinen nicht in der Warteschlange

Wenn Nachrichten wider Erwarten nicht in einer Warteschlange erscheinen, überprüfen Sie Folgendes:

- Wurde die Nachricht erfolgreich in die Warteschlange eingereiht?
  - Wurde die Warteschlange korrekt definiert? Ist zum Beispiel MAXMSGL groß genug?
  - Ist die Warteschlange für PUT-Aufrufe aktiviert?
  - Ist die Warteschlange bereits voll? Dies kann bedeuten, dass eine Anwendung nicht in der Lage war, die gesuchte Nachricht in die Warteschlange einzureihen.
- Können Sie überhaupt eine Nachricht aus der Warteschlange abrufen?
  - Müssen Sie einen Synchronisationspunkt festlegen?

    Wenn Nachrichten zwischen Synchronisationspunkten eingereiht oder abgerufen werden, stehen sie anderen Tasks erst zur Verfügung, nachdem die Arbeitseinheit mit Wiederherstellung festgeschrieben wurde.

- Ist das Warteintervall lang genug?

  Das Warteintervall kann als eine Option für den Aufruf MQGET angegeben werden. Stellen Sie sicher, dass der Aufruf lange genug auf eine Antwort wartet.
- Warten Sie auf eine bestimmte Nachricht, die durch eine Nachrichten-ID oder eine Korrelations-ID (*MsgId* oder *CorrelId*) identifiziert wurde?

  Überprüfen Sie, ob Sie auf eine Nachricht mit der richtigen *MsgId* bzw. *CorrelId* warten. Nach einem erfolgreichen MQGET-Aufruf enthalten diese beiden Werte die IDs der abgerufenen Nachricht, so dass Sie diese Werte möglicherweise zurücksetzen müssen, um eine andere Nachricht erfolgreich abzurufen.

  Überprüfen Sie außerdem, ob Sie andere Nachrichten aus der Warteschlange abrufen können.
- Können andere Anwendungen Nachrichten aus der Warteschlange abrufen?
- Wurde die erwartete Nachricht als permanente Nachricht definiert?

  Wenn nicht, ist die Nachricht nach einem erneuten Start von MQSeries verloren gegangen.
- Besitzt eine andere Anwendung einen exklusiven Zugriff auf die Warteschlange?

Wenn Sie in Bezug auf die Warteschlange keinen Fehler feststellen können und MQSeries aktiv ist, überprüfen Sie wie folgt den Prozess, von dem Sie annehmen, dass er die Nachricht in die Warteschlange einreiht:

- Wurde die Anwendung gestartet?

  Wenn die Anwendung durch einen Auslöser gestartet werden sollte, überprüfen Sie, ob die richtigen Auslöseoptionen angegeben wurden.
- Wurde die Anwendung gestoppt?
- Ist ein Auslösemonitor aktiv?
- Wurde der Auslöseprozess korrekt definiert?
- Wurde die Anwendung ordnungsgemäß beendet?

  Überprüfen Sie im Jobprotokoll, ob möglicherweise eine abnormale Beendigung stattfand.
- Hat die Anwendung ihre Änderungen festgeschrieben, oder wurden sie zurückgesetzt?

Wenn mehrere Transaktionen auf die Warteschlange zugreifen, kann es zwischen ihnen zu Konflikten kommen. Nehmen wir zum Beispiel an, dass eine Transaktion einen MQGET-Aufruf mit einer Pufferlänge gleich null ausgibt, um die Länge einer Nachricht festzustellen, und anschließend einen gezielten MQGET-Aufruf mit Angabe der *MsgId* der Nachricht ausgibt. Inzwischen wird aber von einer anderen Transaktion erfolgreich ein MQGET-Aufruf für dieselbe Nachricht ausgegeben, so dass an die erste Anwendung der Ursachencode MQRC_NO_MSG_AVAILABLE zurückgegeben wird. Anwendungen, die in einer Multiserverumgebung ausgeführt werden sollen, müssen eine solche Situation verarbeiten können.

Prüfen Sie die Möglichkeit, dass die Nachricht empfangen wurde, Ihre Anwendung sie aber aus irgendeinem Grund nicht verarbeiten konnte. Zum Beispiel kann ein Fehler in dem erwarteten Format der Nachricht die Anwendung veranlasst haben, die Nachricht zurückzuweisen. Wie Sie in einem solchen Fall weiter vorgehen können, ist unter „Nachrichten mit unerwarteten oder beschädigten Informationen” auf Seite 209 beschrieben.

Nachrichten mit unerwarteten oder beschädigten Informationen

Wenn die Nachricht andere Informationen enthält, als von Ihrer Anwendung erwartet, oder die Informationen auf irgendeine Art und Weise beschädigt sind, überprüfen Sie folgende Punkte:

- Wurde Ihre Anwendung oder die Anwendung, von der die Nachricht in die Warteschlange eingereiht wurde, geändert?

  Stellen Sie sicher, dass alle Änderungen auf allen Systemen, die von den Änderungen betroffen sind, gleichermaßen vorgenommen wurden.

  Zum Beispiel kann sich das Format der Nachrichtendaten geändert haben, was zur Folge hat, dass beide Anwendungen erneut kompiliert werden müssen, damit sie die Änderungen berücksichtigen. Wenn eine der Anwendungen nicht erneut kompiliert wurde, werden ihre Daten von der anderen Anwendung als beschädigt betrachtet.
- Sendet eine Anwendung Nachrichten an die falsche Warteschlange?

  Überprüfen Sie, ob die von Ihrer Anwendung empfangenen Nachrichten nicht möglicherweise für eine Anwendung bestimmt sind, die auf eine andere Warteschlange zugreift. Ändern Sie gegebenenfalls Ihre Sicherheitsdefinitionen, um zu verhindern, dass Nachrichten von dazu nicht berechtigten Anwendungen in die falschen Warteschlangen eingereiht werden.

  Wenn Ihre Anwendung eine Aliaswarteschlange verwendet hat, überprüfen Sie, ob der Aliasname auf die richtige Warteschlange zeigt.
- Wurde die Auslöseinformation für diese Warteschlange richtig angegeben?

  Überprüfen Sie, ob tatsächlich Ihre und nicht möglicherweise eine andere Anwendung gestartet werden sollte.

Wenn Sie den Fehler auch nach diesen Überprüfungen nicht beheben konnten, sollten Sie Ihre Anwendungslogik überprüfen, und zwar sowohl die des Programms zum Senden der Nachricht als auch die des Programms zum Empfangen der Nachricht.

Probleme mit falschen Ausgaben bei der Verwendung verteilter Warteschlangen

Wenn Ihre Anwendung verteilte Warteschlangen verwendet, sollten Sie zudem folgende Punkte überprüfen:

- Wurde MQSeries sowohl auf dem sendenden als auch auf dem empfangenden System ordnungsgemäß installiert und korrekt für verteiltes Queuing konfiguriert?
- Sind die Verbindungen zwischen den beiden Systemen verfügbar?

  Überprüfen Sie, ob beide Systeme verfügbar und mit MQSeries verbunden sind. Überprüfen Sie, ob die Verbindung zwischen den beiden Systemen und die Kanäle zwischen den beiden WS-Managern aktiv sind.
- Ist auf dem sendenden System die Auslösefunktion aktiviert?
- Handelt es sich bei der Nachricht, auf die Sie warten, um eine Antwortnachricht von einem fernen System?

  Überprüfen Sie, ob die Auslösefunktion auf dem fernen System aktiviert ist.
- Ist die Warteschlange bereits voll?

Falsche Ausgabe

Dies kann bedeuten, dass eine Anwendung nicht in der Lage war, die gesuchte Nachricht in die Warteschlange einzureihen. Überprüfen Sie in diesem Fall, ob die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde.

Der Header der Warteschlange für nicht zustellbare Nachrichten enthält einen Ursachen- oder Rückgabecode, der Auskunft darüber gibt, warum die Nachricht nicht in die Zielwarteschlange eingereiht werden konnte. Informationen zur Struktur des Headers der Warteschlange für nicht zustellbare Nachrichten finden Sie im Handbuch *MQSeries Application Programming Reference*.

- Gibt es eine Abweichung zwischen dem sendenden und dem empfangenden WS-Manager?

  Es ist zum Beispiel möglich, dass der empfangende WS-Manager die Nachrichtenlänge nicht verarbeiten kann.
- Sind die Kanaldefinitionen der Sende- und Empfangskanäle kompatibel?

  Zum Beispiel wird die Komponente für verteiltes Queuing gestoppt, wenn in der Folgenummernserie eine Abweichung vorliegt. Weitere Informationen zum verteilten Queuing finden Sie im Handbuch *MQSeries Intercommunication*.
- Findet eine Datenkonvertierung statt? Wenn die sendenden und empfangenden Anwendungen mit unterschiedlichen Datenformaten arbeiten, wird eine Konvertierung erforderlich. Bei der Ausgabe des Aufrufs MQGET wird eine automatische Konvertierung durchgeführt, wenn es sich bei dem erkannten Format um eines der integrierten Formate handelt.

  Wenn der zu konvertierende Datensatz nicht erkannt wird, wird der Datenkonvertierungs-Exit verwendet, so dass Sie die Umwandlung mit Ihren eigenen Routinen ausführen können.

  Dies gilt nicht für Daten, die an MQSeries for MVS/ESA gesendet werden.

  Weitere Informationen zur Datenkonvertierung finden Sie im Handbuch *MQSeries Intercommunication*.

Fehlerprotokolle

MQSeries erstellt eine Reihe von Fehlerprotokollen mit Nachrichten über den Betrieb von MQSeries selbst, über alle gestarteten WS-Manager und über fehlerhafte Daten von den verwendeten Kanälen.

Das Verzeichnis der Fehlerprotokolle hängt davon ab, ob der Name des WS-Managers bekannt ist und ob der Fehler einem Client zugeordnet ist.

- Der Name des WS-Managers ist bekannt und der WS-Manager verfügbar:

  `MQS_ROOT:[MQM.QMGRS.WS-Manager-Name.ERRORS]AMQERR01.LOG`
- Der WS-Manager ist nicht verfügbar:

  `MQS_ROOT:[MQM.QMGRS.$SYSTEM.ERRORS]AMQERR01.LOG`
- In einer Client-Anwendung ist ein Fehler aufgetreten:

  `MQS_ROOT:[MQM.ERRORS]AMQERR01.LOG`
- Informationen zu FFST (First Failure Support Technology) finden Sie unter „FFST-Daten überprüfen” auf Seite 217.

Anmerkung: Bei Clients werden die Fehler im Stammverzeichnis des Clients gespeichert.

Protokolldateien

Bei der Installation wird im Dateipfad QMGRS ein Verzeichnis mit dem Namen [MQM.QMGRS.$SYSTEM.ERRORS] erstellt. Das Unterverzeichnis 'errors' kann bis zu drei Fehlerprotokolldateien enthalten:

- AMQERR01.LOG
- AMQERR02.LOG
- AMQERR03.LOG

Nachdem Sie einen WS-Manager erstellt haben, werden drei Fehlerprotokolldateien erstellt, sobald sie vom WS-Manager benötigt werden. Diese Dateien haben dieselben Namen wie die $SYSTEM-Dateien, also AMQERR01, AMQERR02 und AMQERR03, jede mit einer Kapazität von 256 KB. Die Dateien befinden sich im Unterverzeichnis 'errors' jedes WS-Managers, den Sie erstellen.

Sobald Fehlernachrichten generiert werden, werden sie in der Datei AMQERR01 gespeichert. Wenn AMQERR01 größer als 256 KB wird, wird sie in die Datei AMQERR02 kopiert. Vorher wird AMQERR02 in die Datei AMQERR03.LOG kopiert. Der bisherige Inhalt von AMQERR03 wird gegebenenfalls vorher gelöscht.

Die neuesten Fehlernachrichten befinden sich daher immer in AMQERR01, während die anderen Dateien dazu dienen, Fehlernachrichten für eine längere Zeit aufzubewahren.

Alle Nachrichten, die Kanäle betreffen, werden ebenfalls in den entsprechenden Fehlerdateien des WS-Managers gespeichert, es sei denn, der Name ihres WS-Managers ist nicht bekannt oder der WS-Manager ist nicht verfügbar. Wenn der Name des WS-Managers nicht verfügbar ist oder sein Name nicht festgestellt werden kann, werden kanalbezogene Nachrichten im Unterverzeichnis [MQM.QMGRS.$SYSTEM.ERRORS] gespeichert.

Sie können den Inhalt einer Fehlerprotokolldatei in Ihrem normalen OpenVMS-Editor anzeigen.

Fehler in der Startphase

Es gibt eine Reihe von besonderen Fällen, in denen Fehler auftreten, bevor die oben genannten Fehlerprotokolle vollständig eingerichtet sind. MQSeries versucht, solche Fehler in einem Fehlerprotokoll aufzuzeichnen. Das Verzeichnis dieses Protokoll ist davon abhängig, wie weit ein WS-Manager erstellt ist.

Wenn zum Beispiel auf Grund einer beschädigten Konfigurationsdatei keine Verzeichnisinformation verfügbar ist, werden Fehler in einem Fehlerverzeichnis protokolliert, das bei der Installation im Stammverzeichnis 'mqm' erstellt wird.

Wenn die MQSeries-Konfigurationsdatei und das Attribut DefaultPrefix in der Zeilengruppe AllQueueManagers gelesen werden kann, werden Fehler im Verzeichnis DefaultPrefix[.errors] protokolliert.

Weitere Information zu Konfigurationsdateien finden Sie in „Kapitel 13. MQSeries konfigurieren" auf Seite 179.

Bedienernachrichten

In MQSeries for Compaq OpenVMS beziehen sich Bedienernachrichten auf normale Fehler, die in der Regel direkt von Benutzern verursacht werden, z. B. indem sie in einem Befehl Parameter verwenden, die dort nicht zulässig sind. Bedienernachrichten werden in der jeweiligen Landessprache ausgegeben, d. h., es gibt Nachrichtenkataloge, die in Standardverzeichnissen installiert werden.

Diese Nachrichten werden im zugeordneten Fenster, sofern vorhanden, angezeigt und darüber hinaus in das Fehlerprotokoll AMQERR01.LOG im Verzeichnis des WS-Managers geschrieben. Beispiel:

```
MQS_ROOT:[MQM.QMGRS.QUEUE$MANAGER.ERRORS]
```

Einige Fehler werden in der Datei AMQERR01.LOG im Verzeichnis des WS-Managers und andere in der $SYSTEM-Verzeichniskopie des Fehlerprotokolls protokolliert.

Beispiel für ein Fehlerprotokoll

Dieses Beispiel zeigt einen Teil eines MQSeries for Compaq OpenVMS-Fehlerprotokolls:

```
 ...
06/29/00  09:41:39 AMQ7467: Die älteste erforderliche Protokolldatei zum Starten des
Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
-----------------------------------------------                                --
06/29/00  09:41:39 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
-------------------------------------------                                  --
06/29/00  09:42:05 AMQ7467: Die älteste erforderliche Protokolldatei zum
Starten des Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
---------------------------------------                                      --
06/29/00  09:42:05 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist
S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
-------------------------------------                                        --
06/29/00  09:42:06 AMQ8003: MQSeries-Warteschlangenmanager gestartet.
```

```
ERKLÄRUNG: Der MQSeries-Warteschlangenmanager BKM1 wurde gestartet.
AKTION: Keine.
----------------------------------------                                    --
06/29/00  09:42:06 AMQ7467: Die älteste erforderliche Protokolldatei zum
Starten des Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
----------------------------------------                                    --
06/29/00  09:42:06 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist
S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
----------------------------------------                                    --
06/29/00  09:46:27 AMQ7030: Anforderung zum Stillegen des Warteschlangenmanagers akzeptiert.
Der Warteschlangenmanager wird gestoppt, wenn keine weiteren Aufgaben mehr für
ihn anstehen.

ERKLÄRUNG: Sie haben gefordert, dass der WS-Manager beendet wird, wenn keine
Arbeit mehr für ihn vorliegt. In der Zwischenzeit wird er den Start neuer
Anwendungen ablehnen, jedoch zulassen, dass alle aktiven Anwendungen ihre
Arbeit beenden können.
AKTION: Keine.
----------------------------------------                                    --
06/29/00  09:46:43 AMQ7467: Die älteste erforderliche Protokolldatei zum
Starten des Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
-----------------------------------                                         --
06/29/00  09:46:43 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist
S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
-----------------------------------                                         --
06/29/00  09:46:44 AMQ8004: MQSeries-Warteschlangenmanager beendet.

ERKLÄRUNG: Der MQSeries-Warteschlangenmanager BKM1 wurde beendet.
AKTION: Keine.
-----------------------------------                                         --
06/29/00  09:46:59 AMQ7467: Die älteste erforderliche Protokolldatei zum
Starten des Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
```

```
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
------------------------------------                                        --
06/29/00  09:47:00 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist
S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
-------------------------------------                                       --
06/29/00  09:47:08 AMQ7472: Objekt TEST1 der Art queue ist beschädigt.

ERKLÄRUNG: Objekt TEST1 der Art queue wurde als beschädigt markiert. Dies
bedeutet, dass der WS-Manager entweder nicht auf das Objekt im Dateisystem
zugreifen konnte oder dass eine Inkonsistenz mit den Daten im Objekt erkannt
wurde.
AKTION: Wenn ein beschädigtes Objekt erkannt wird, hängt die auszuführende
Aktion davon ab, ob der Warteschlangenmanager die Wiederherstellung über
Datenträger unterstützt und wann die Beschädigung festgestellt wurde.
Unterstützt der WS-Manager keine Datenträgerwiederherstellung, müssen Sie das
Objekt löschen, da eine Wiederherstellung nicht möglich ist. Wenn der
Warteschlangenmanager die Wiederherstellung über Datenträger unterstützt und
die Beschädigung während der Verarbeitung beim Starten des
Warteschlangenmanagers festgestellt wurde, initiiert der Warteschlangenmanager
die Wiederherstellung des Objekts über Datenträger automatisch. Wenn der
Warteschlangenmanager die Wiederherstellung über Datenträger unterstützt und die
Beschädigung nach dem Starten des Warteschlangenmanagers festgestellt wurde,
kann das Objekt von einem Datenträgerabbild unter Verwendung des Befehls
rcrmqobj wiederhergestellt werden, oder es kann gelöscht werden.
-------------------------------------                                       --
06/29/00  09:47:09 AMQ8003: MQSeries-Warteschlangenmanager gestartet.

ERKLÄRUNG: Der MQSeries-Warteschlangenmanager BKM1 wurde gestartet.
AKTION: Keine.
-------------------------------------                                       --
06/29/00  09:47:09 AMQ7467: Die älteste erforderliche Protokolldatei zum
Starten des Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
------------------------------------                                        --
06/29/00  09:47:10 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist
S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
-------------------------------------                                       --
06/29/00  09:47:47 AMQ7081: Objekt TEST1 der Art queue neu erstellt.

ERKLÄRUNG: Das Objekt TEST1 der Art queue wurde von dem entsprechenden
Datenträgerabbild neu erstellt.
AKTION: Keine.
-------------------------------------                                       --
06/29/00  11:22:10 AMQ7467: Die älteste erforderliche Protokolldatei zum
Starten des Warteschlangenmanagers BKM1 ist S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Starten des Warteschlangenmanagers erforderlich ist.
Möglicherweise sind ältere Protokollsätze für die Wiederherstellung über
Datenträger erforderlich.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben. Wenn Protokolldateien verschoben werden,
```

```
die zum erneuten Erstellen von Objekten aus Datenträgerabbildern erforderlich
sind, müssen diese zum erneuten Erstellen der Objekte zurückgeschrieben werden.
----------------------------------                                          --
06/29/00  11:22:10 AMQ7468: Die älteste Protokolldatei, die zur Wiederherstellung
des Warteschlangenmanagers BKM1 über Datenträger erforderlich ist, ist
S0000000.LOG.

ERLÄUTERUNG: Die Protokolldatei S0000000.LOG enthält den ältesten Protokollsatz,
der zum erneuten Erstellen eines der Objekte aus dem entsprechenden
Datenträgerabbild erforderlich ist. Protokolldateien, die älter sind als diese,
werden bei der Wiederherstellung über Datenträger nicht verwendet.
AKTION: Protokolldateien, die älter als S0000000.LOG sind, können auf einen
Archivierungsdatenträger gespeichert werden, um Speicherplatz im
Protokollverzeichnis freizugeben.
--------------------------------                                            --
06/29/00  11:22:11 AMQ8004: MQSeries-Warteschlangenmanager beendet.

ERKLÄRUNG: Der MQSeries-Warteschlangenmanager BKM1 wurde beendet.
AKTION: Keine.
--------------------------------                                            --
 ...
```

Warteschlangen für nicht zustellbare Nachrichten

Nachrichten, die aus irgendeinem Grund nicht zugestellt werden können, werden in die Warteschlange für nicht zustellbare Nachrichten gestellt. Mit dem MQSC-Befehl DISPLAY QUEUE können Sie überprüfen, ob die Warteschlange Nachrichten enthält. Wenn sich Nachrichten in der Warteschlange befinden, können Sie die Nachrichten mit Hilfe der bereitgestellten Beispielanwendung (amqsbcg) und des Aufrufs MQGET durchsuchen. Die Beispielanwendung durchsucht alle Nachrichten in einer angegebenen Warteschlange auf einem angegebenen WS-Manager und zeigt den Nachrichtendeskriptor und die Felder mit dem Nachrichtenkontext aller Nachrichten in der angegebenen Warteschlange an.

Sie müssen entscheiden, was mit Nachrichten, die in der Warteschlange für nicht zustellbare Nachrichten gefunden werden, weiter geschehen soll; dies ist von den Gründen abhängig, aus denen die Nachrichten in die Warteschlange eingereiht wurden.

Probleme können dann entstehen, wenn Sie nicht auf jedem von Ihnen verwendeten WS-Manager eine Warteschlange für nicht zustellbare Nachrichten definiert haben. Wenn der WS-Manager mit dem Befehl **crtmqm** erstellt wird, wird automatisch eine Warteschlange für nicht zustellbare Nachrichten mit dem Namen SYSTEM.DEAD.LETTER.QUEUE als Standardobjekt erstellt. Diese Warteschlange ist jedoch nicht als Warteschlange für nicht zustellbare Nachrichten für den WS-Manager definiert (siehe „Eine Warteschlange für nicht zustellbare Nachrichten definieren” auf Seite 46).

Konfigurationsdateien und Fehlerbestimmung

Fehler in Konfigurationsdateien führen häufig dazu, dass WS-Manager nicht gefunden werden und Fehler des Typs „WS-Manager nicht verfügbar" auftreten.

Sie haben mehrere Möglichkeiten, die Konfigurationsdateien zu überprüfen:

- Stellen Sie sicher, dass die Konfigurationsdateien vorhanden sind.
- Stellen Sie sicher, dass sie die richtigen Berechtigungen enthalten, z. B.:

```
      MQS.INI;1   MQM    (RWED, RWED, RW, R)
(ID=MQM, Zugriffsrechte=READ+WRITE+EXECUTE+DELETE+CONTROL)
```

- Stellen Sie sicher, dass die MQSeries-Konfigurationsdatei auf die richtigen WS-Manager- und Protokollverzeichnisse verweist.

MQSeries-Trace verwenden

MQSeries for Compaq OpenVMS verwendet die folgenden Befehle für die Trace-Einrichtung:

- **strmqtrc** – siehe „strmqtrc (MQSeries-Trace starten)" auf Seite 319
- **dspmqtrc** – siehe „dspmqtrc (Formatierte MQSeries-Trace-Ausgabe anzeigen)" auf Seite 275
- **endmqtrc** – siehe „endmqtrc (MQSeries-Trace beenden)" auf Seite 285

Die Trace-Einrichtung verwendet für jede Definitionseinheit, für die ein Trace durchgeführt wird, eine eigene Datei, in der die Trace-Informationen aufgezeichnet werden.

Trace-Dateien werden im Verzeichnis MQS_ROOT:[MQM.TRACE] erstellt.

Die Dateien in diesem Verzeichnis enthalten Einzelangaben zu WS-Managern sowie Informationen zu allen Traces während der Startphase und allen $SYSTEM-Traces.

Namen von Trace-Dateien

Namen von Trace-Dateien haben folgende Struktur:

AMQ*pppppppp*.TRC

Dabei steht *pppppppp* für die Prozess-ID (PID) des Prozesses, der den Trace erstellt hat.

Anmerkungen:

1. In MQSeries for Compaq OpenVMS hat der Wert für die Prozess-ID immer eine Länge von acht Zeichen.
2. Es gibt eine eigene Trace-Datei für jeden Prozess, der als Teil der Definitionseinheit, für die ein Trace durchgeführt wird, aktiv ist.

Beispiel für Trace-Daten

Das folgende Beispiel ist ein Auszug aus einer OpenVMS-Trace-Datei:

```
 ...
ID     ELAPSED_MSEC     DELTA_MSEC      APPL    SYSCALL KERNEL  INTERRUPT

30d     0 0 MQS CEI Exit!. 12484.1 xcsWaitEventSem rc=10806020
30d     0 0 MQS CEI Exit! 12484.1 zcpReceiveOnLink rc=20805311
30d     0 0 MQS FNC Entry 12484.1 zxcProcessChildren
30d     0 0 MQS CEI Entry. 12484.1 xcsRequestMutexSem
30d     1 0 MQS CEI Entry.. 12484.1 xcsHSHMEMBtoPTR
30d     1 0 MQS CEI Exit... 12484.1 xcsHSHMEMBtoPTR rc=00000000
30d     1 0 MQS FNC Entry.. 12484.1 xllSemGetVal
30d     1 0 MQS FNC Exit... 12484.1 xllSemGetVal rc=00000000
30d     1 0 MQS FNC Entry.. 12484.1 xllSemReq
30d     1 0 MQS FNC Exit... 12484.1 xllSemReq rc=00000000
30d     1 0 MQS CEI Exit.. 12484.1 xcsRequestMutexSem rc=00000000
30d     2 0 MQS CEI Entry. 12484.1 xcsReleaseMutexSem
30d     2 0 MQS CEI Entry.. 12484.1 xcsHSHMEMBtoPTR
30d     2 0 MQS CEI Exit... 12484.1 xcsHSHMEMBtoPTR rc=00000000
30d     2 0 MQS FNC Entry.. 12484.1 xllSemRel
30d     2 0 MQS FNC Exit... 12484.1 xllSemRel rc=00000000
30d     2 0 MQS CEI Exit.. 12484.1 xcsReleaseMutexSem rc=00000000
30d     2 0 MQS CEI Entry. 12484.1 xcsHSHMEMBtoPTR
 ...
```

Abbildung 21. Beispiel für einen MQSeries for Compaq OpenVMS-Trace

Anmerkungen:

1. In diesem Beispiel sind nicht alle Daten dargestellt. Ein tatsächlicher Trace enthält die vollständigen Funktionsnamen und Rückkehrcodes.
2. Die Rückkehrcodes werden als Werte angegeben, nicht als Literale.

First Failure Support Technology (FFST)

Informationen, die normalerweise in FFST-Protokollen aufgezeichnet werden, werden in MQSeries for Compaq OpenVMS in einer Datei im Verzeichnis MQS_ROOT:[MQM.ERRORS] aufgezeichnet.

Dabei handelt es sich in der Regel um schwer wiegende, nicht behebbare Fehler, die entweder auf einen Konfigurationsfehler im System oder einen internen MQSeries-Fehler hinweisen.

FFST-Daten überprüfen

Die Namen der Dateien haben folgende Struktur: AMQ*nnnnnnnn*_*mm*.FDC, wobei:

nnnnnnnn
: für die ID des Prozesses steht, der den Fehler meldet,

mm
: für eine Folgenummer (normalerweise 0) steht.

FFST

Wenn ein Prozess eine FFST-Datei erstellt, macht er gleichzeitig einen Eintrag in das Systemfehlerprotokoll. Der Eintrag enthält den Namen der FFST-Datei, um eine automatische Fehlerverfolgung zu unterstützen.

```
+------------------------------------------------------------------------------+
|                                                                              |
| MQSeries First Failure Symptom Report                                        |
| =====================================                                        |
|                                                                              |
| Date/Time         :- Monday January 29 21:32:03 GMT 2001                     |
| Host Name         :- CELERY (Unknown)                                        |
| PIDS              :- 5697175                                                 |
| LVLS              :- 510                                                     |
| Product Long Name :- MQSeries for OpenVMS Alpha                              |
| Vendor            :- IBM                                                     |
| Probe Id          :- ZX005025                                                |
| Application Name  :- MQM                                                     |
| Component         :- zxcProcessChildren                                      |
| Build Date        :- Jan  8 2001                                             |
| Userid            :- [400,400] (SJACKSON)                                    |
| Program Name      :- AMQZXMA0.EXE                                            |
| Process           :- 202001DA                                                |
| Thread            :- 00000001                                                |
| QueueManager      :- JJJH                                                    |
| Major Errorcode   :- zrcX_PROCESS_MISSING                                    |
| Minor Errorcode   :- OK                                                      |
| Probe Type        :- MSGAMQ5008                                              |
| Probe Severity    :- 2                                                       |
| Probe Description :- AMQ5008: Ein wichtiger MQSeries-Prozess 538968541 ist   |
|   unauffindbar. Es wird angenommen, dass er beendet ist.                     |
| Arith1            :- 538968541 202001dd                                      |
| VMS Errorcode     :- -SYSTEM-W-NONEXPR, nonexistent process (000008E8)       |
|                                                                              |
| JPI Quota information:                                                       |
| ======================                                                       |
| ASTCNT=247/250(98%) *                    BIOCNT=500/500(100%) *              |
| BYTCNT=183616/183616(100%) *             DIOCNT=250/250(100%) *              |
| ENQCNT=4885/5000(97%) *                  FILCNT=241/250(96%) *               |
| PAGFILCNT=975280/1000000(97%) *          TQCNT=246/250(98%) *                |
| FREPTECNT=2147483647                     APTCNT=0                            |
| GPGCNT=5808                              PPGCNT=5872                         |
| VIRTPEAK=203264                          DFWSCNT=1392                        |
| WSAUTH=2784                              WSAUTHEXT=65536                     |
| WSEXTENT=65536                           WSPEAK=11680                        |
| WSQUOTA=2784                             WSSIZE=15792                        |
| CPULIM=0                                 MAXDETACH=0                         |
| MAXJOBS=0                                JOBPRCCNT=2                         |
| PAGEFLTS=2895                            PRCCNT=2/100(2%) +                  |
| (*) - % resource remaining, (+) - % resource used                            |
|                                                                              |
| Privilege and rights information:                                            |
| =================================                                            |
| CURPRIV=bugchk detach netmbx prmgbl sysgbl sysprv tmpmbx world               |
| IMAGPRIV=bugchk prmgbl sysgbl world                                          |
| AUTHPRIV=bugchk detach netmbx prmgbl sysgbl sysprv tmpmbx world              |
| SJACKSON                                 INTERACT                            |
| REMOTE                                   MQM                                 |
| SYS                                                                          |
| IMAGE_RIGHTS=                                                                |
| SYS$NODE_CELERY                                                              |
|                                                                              |
| SYI information:                                                             |
| ================                                                             |
| ACTIVE CPU=1/1(100%) +                   CLUSTER NODES=1                     |
| FREE_GBLPAGES=16000528/16174643(98%) *   GBLPAGFIL=1000000                   |
| FREE_GBLSECTS=936/1550(60%) *            MEMSIZE=16384                       |
| PAGEFILE_FREE=16888/16888(100%) *        PAGE_SIZE=8192                      |
| SWAPFILE_FREE=936/936(100%) *            MAXPROCESSCNT=102                   |
| PROCSECTCNT=64                           BALSETCNT=100                       |
| WSMAX=65536                              NPAGEDYN=2269184                    |
| NPAGEVIR=9437184                         PAGEDYN=1597440                     |
| VIRTUALPAGECNT=2147483647                LOCKIDTBL_MAX=109437                |
| PQL_DASTLM=24                            PQL_MASTLM=100                      |
| PQL_DBIOLM=32                            PQL_MBIOLM=100                      |
| PQL_DBYTLM=65536                         PQL_MBYTLM=100000                   |
| PQL_DCPULM=0                             PQL_MCPULM=0                        |
| PQL_DDIOLM=32                            PQL_MDIOLM=100                      |
| PQL_DFILLM=128                           PQL_MFILLM=100                      |
| PQL_DPGFLQUOTA=65536                     PQL_MPGFLQUOTA=32768                |
```

```
| PQL_DPRCLM=32                          PQL_MPRCLM=10                          |
| PQL_DTQELM=16                          PQL_MTQELM=0                           |
| PQL_DWSDEFAULT=1392                    PQL_MWSDEFAULT=1392                    |
| PQL_DWSQUOTA=2784                      PQL_MWSQUOTA=2784                      |
| PQL_DWSEXTENT=65536                    PQL_MWSEXTENT=65536                    |
| PQL_DENQLM=128                         PQL_MENQLM=300                         |
| PQL_DJTQUOTA=4096                      PQL_MJTQUOTA=0                         |
| CLISYMTBL=750                          DEFMBXMXMSG=256                        |
| DEFMBXBUFQUO=1056                      CHANNELCNT=5000                        |
| DLCKEXTRASTK=2560                      PIOPAGES=575                           |
| CTLPAGES=256                           CTLIMGLIM=35                           |
| (*) - % resource remaining, (+) - % resource used                             |
|                                                                               |
+-------------------------------------------------------------------------------+

MQM Function Stack
zxcProcessChildren
xcsFFST

MQM Trace History
                     --> xllFreeSem
                     <-- xllFreeSem rc=OK
                     --> xcsFreeQuickCell
                      --> xllSpinLockRequest
                      <-- xllSpinLockRequest rc=OK
                      --> xstFreeCell
                      <-- xstFreeCell rc=OK
                      --> xllSpinLockRelease
                      <-- xllSpinLockRelease rc=OK
                     <-- xcsFreeQuickCell rc=OK
                    <-- xcsCloseEventSem rc=OK
                    --> xcsFreeMemBlock
                     --> xstFreeMemBlock
                      --> xcsRequestThreadMutexSem
                      <-- xcsRequestThreadMutexSem rc=OK
                      --> xcsReleaseThreadMutexSem
                      <-- xcsReleaseThreadMutexSem rc=OK
                      --> xstFreeBlockFromSharedMemSet
                       --> xllSpinLockSlowRequest
                       <-- xllSpinLockSlowRequest rc=OK
                       --> xllSpinLockRelease
                       <-- xllSpinLockRelease rc=OK
                       --> xstFreeBlockInExtent
                        --> xcsQueryMutexSem
                        <-- xcsQueryMutexSem rc=OK
                        --> xcsRequestMutexSem
                         --> xllSemReq
                          --> vms_mtx
                           --> vms_get_lock
                           <-- vms_get_lock rc=OK
                          <-- vms_mtx rc=OK
                         <-- xllSemReq rc=OK
                        <-- xcsRequestMutexSem rc=OK
                        --> xclDeleteMutexMem
                         --> xllCSCloseMutex
                          --> xihHANDLEtoSUBPOOLFn
                           --> xihGetConnSPDetailsFromList
                            --> xihGetConnSPDetails
                            <-- xihGetConnSPDetails rc=OK
                           <-- xihGetConnSPDetailsFromList rc=OK
                          <-- xihHANDLEtoSUBPOOLFn rc=OK
                          --> xllSpinLockSlowRequest
                          <-- xllSpinLockSlowRequest rc=OK
                          --> xllSpinLockRelease
                          <-- xllSpinLockRelease rc=OK
                          --> xllFreeSem
                          <-- xllFreeSem rc=OK
                          --> vms_mtx
                           --> vms_get_lock
                           <-- vms_get_lock rc=OK
                          <-- vms_mtx rc=OK
                          --> xcsFreeQuickCell
                           --> xllSpinLockRequest
                           <-- xllSpinLockRequest rc=OK
                           --> xstFreeCell
                           <-- xstFreeCell rc=OK
                           --> xllSpinLockRelease
                           <-- xllSpinLockRelease rc=OK
                          <-- xcsFreeQuickCell rc=OK
                         <-- xllCSCloseMutex rc=OK
```

```
     <-- xclDeleteMutexMem rc=OK
     --> xstSerialiseExtent
      --> xllSpinLockRequest
      <-- xllSpinLockRequest rc=OK
     <-- xstSerialiseExtent rc=OK
     --> xstFreeChunk
      --> xstDeleteChunk
      <-- xstDeleteChunk rc=OK
      --> xstInsertChunk
      <-- xstInsertChunk rc=OK
     <-- xstFreeChunk rc=OK
     --> xstReleaseSerialisationOnExtent
      --> xllSpinLockRelease
      <-- xllSpinLockRelease rc=OK
     <-- xstReleaseSerialisationOnExtent rc=OK
    <-- xstFreeBlockInExtent rc=OK
   <-- xstFreeBlockFromSharedMemSet rc=OK
  <-- xstFreeMemBlock rc=OK
 <-- xcsFreeMemBlock rc=OK
<-- zcpDeleteIPC rc=OK
--> xcsReleaseMutexSem
 --> xllSemRel
  --> vms_mtx
   --> vms_get_lock
   <-- vms_get_lock rc=OK
  <-- vms_mtx rc=OK
 <-- xllSemRel rc=OK
<-- xcsReleaseMutexSem rc=OK
--> xcsFreeMemBlock
 --> xstFreeMemBlock
  --> xcsRequestThreadMutexSem
  <-- xcsRequestThreadMutexSem rc=OK
  --> xcsReleaseThreadMutexSem
  <-- xcsReleaseThreadMutexSem rc=OK
  --> xstFreeBlockFromSharedMemSet
   --> xllSpinLockSlowRequest
   <-- xllSpinLockSlowRequest rc=OK
   --> xllSpinLockRelease
   <-- xllSpinLockRelease rc=OK
   --> xstFreeBlockInExtent
    --> xcsQueryMutexSem
    <-- xcsQueryMutexSem rc=OK
    --> xcsRequestMutexSem
     --> xllSemReq
      --> vms_mtx
       --> vms_get_lock
       <-- vms_get_lock rc=OK
      <-- vms_mtx rc=OK
     <-- xllSemReq rc=OK
    <-- xcsRequestMutexSem rc=OK
    --> xclDeleteMutexMem
     --> xllCSCloseMutex
      --> xihHANDLEtoSUBPOOLFn
       --> xihGetConnSPDetailsFromList
        --> xihGetConnSPDetails
        <-- xihGetConnSPDetails rc=OK
       <-- xihGetConnSPDetailsFromList rc=OK
      <-- xihHANDLEtoSUBPOOLFn rc=OK
      --> xllSpinLockSlowRequest
      <-- xllSpinLockSlowRequest rc=OK
      --> xllSpinLockRelease
      <-- xllSpinLockRelease rc=OK
      --> xllFreeSem
      <-- xllFreeSem rc=OK
      --> vms_mtx
       --> vms_get_lock
       <-- vms_get_lock rc=OK
      <-- vms_mtx rc=OK
      --> xcsFreeQuickCell
       --> xllSpinLockRequest
       <-- xllSpinLockRequest rc=OK
       --> xstFreeCell
       <-- xstFreeCell rc=OK
       --> xllSpinLockRelease
       <-- xllSpinLockRelease rc=OK
      <-- xcsFreeQuickCell rc=OK
     <-- xllCSCloseMutex rc=OK
    <-- xclDeleteMutexMem rc=OK
    --> xstSerialiseExtent
     --> xllSpinLockRequest
```

```
                          <-- xllSpinLockRequest rc=OK
                         <-- xstSerialiseExtent rc=OK
                         --> xstFreeChunk
                          --> xstDeleteChunk
                          <-- xstDeleteChunk rc=OK
                          --> xstInsertChunk
                          <-- xstInsertChunk rc=OK
                         <-- xstFreeChunk rc=OK
                         --> xstReleaseSerialisationOnExtent
                          --> xllSpinLockRelease
                          <-- xllSpinLockRelease rc=OK
                         <-- xstReleaseSerialisationOnExtent rc=OK
                        <-- xstFreeBlockInExtent rc=OK
                       <-- xstFreeBlockFromSharedMemSet rc=OK
                      <-- xstFreeMemBlock rc=OK
                     <-- xcsFreeMemBlock rc=OK
                    <-- zxcCleanupAgent rc=OK
                    --> xcsReleaseMutexSem
                     --> xllSemRel
                      --> vms_mtx
                       --> vms_get_lock
                       <-- vms_get_lock rc=OK
                      <-- vms_mtx rc=OK
                     <-- xllSemRel rc=OK
                    <-- xcsReleaseMutexSem rc=OK
                    --> xcsCheckProcess
                     --> kill
                     <-- kill rc=OK
                    <-- xcsCheckProcess rc=OK
                    --> xcsCheckProcess
                     --> kill
                     <-- kill rc=OK
                    <-- xcsCheckProcess rc=OK
                    --> xcsRequestMutexSem
                     --> xllSemReq
                      --> vms_mtx
                       --> vms_get_lock
                       <-- vms_get_lock rc=OK
                      <-- vms_mtx rc=OK
                     <-- xllSemReq rc=OK
                    <-- xcsRequestMutexSem rc=OK
                    --> xcsReleaseMutexSem
                     --> xllSemRel
                      --> vms_mtx
                       --> vms_get_lock
                       <-- vms_get_lock rc=OK
                      <-- vms_mtx rc=OK
                     <-- xllSemRel rc=OK
                    <-- xcsReleaseMutexSem rc=OK
                    --> xcsCheckProcess
                     --> kill
                     <-- kill rc=Unknown(FFFF)
                    <-- xcsCheckProcess rc=xecP_E_INVALID_PID
                    --> xcsBuildDumpPtr
                     --> xcsGetMem
                     <-- xcsGetMem rc=OK
                    <-- xcsBuildDumpPtr rc=OK
                   <-- xcsBuildDumpPtr rc=OK
                  <-- xcsBuildDumpPtr rc=Unknown(4B)
                  --> xcsFFST

ECAnchor
6A91B0                                    5A584541              ZXEA
6A91C0   03000000  E8030000  03220000  0100DA01    ....&#232;...."....&#218;.
6A91D0   2C05A500  D4040000  0A00DA01  01000000    ,.¥.&#212;.....&#218;.....
6A91E0   B0000000  0A00DA01  03000000  E8030000    °.....&#218;.....&#232;...
6A91F0   03220000  0100DA01  07000000  00000000    ."....&#218;.........
6A9200   283F9700  03000000  F0030000  08410000    (?—..........A..
6A9210   0300DA01  03000000  F0030000  08410000    ..&#218;.....,....A..
6A9220   0300DA01  01000000  B4000000  0200DA01    ..&#218;.....´.....&#218;.
6A9230   03000000  E8030000  03220000  0100DA01    ....&#232;...."....&#218;.
6A9240   00000000  00000000  00000000  00000000    ................
6A9250   00000000  00000000  00000000  00000000    ........,.......
6A9260   00000000  01000000  B4000000  0200DA01    ........ .....&#218;.
6A9270   03000000  E8030000  03220000  0100DA01    ,...&#232;...."..,.&#218;.
6A9280   B41F0000  0200DA01  01000000  B4000000     .....&#218;.....´...
6A9290   0200DA01  03000000  E8030000  03220000    ..&#218;.....&#232;...."..
6A92A0   0100DA01  DA012020  00000000  EFCD0300    ..&#218;.&#218;.  .....&#205;..
6A92B0   00000000  00000000  00000000  00000000    ................
6A92C0   00000000  00000000  01000000  00000000    ................
```

```
6A92D0   44454641  554C5400  00000000  00000000    DEFAULT.........
6A92E0   00000000  00000000  00000000  00000000    ................
6A92F0   00000000  00000000  00000000  00000000    ................
6A9300   2F6D7173  5F726F6F  742F6D71  6D000000    /mqs_root/mqm...
6A9310   00000000  00000000  00000000  00000000    ................
6A9320 to 6A93F0 suppressed, lines same as above
6A9400   5A435048  01000000  B8000000  0400DA01    ZCPH....&#184;.....&#218;.
6A9410   03000000  F0030000  08410000  0300DA01    .........A....&#218;.
6A9420   DC040000  0400DA01  01000000  B8000000    &#220;.....&#218;.....&#184;...
6A9430   0400DA01  03000000  F0030000  08410000    ..&#218;..........A..
6A9440   0300DA01  00000000  00000000  00000000    ..&#218;.............
6A9450   00000000  00000000  00000000  00000000    ................
6A9460 to 6A9470 suppressed, lines same as above
6A9480   00000000  00000000  5A435048  01000000    ........ZCPH....
6A9490   B8000000  0500DA01  03000000  F0030000    &#184;.....&#218;.........
6A94A0   08410000  0300DA01  DC040000  0500DA01    .A....&#218;.&#220;.....&#218;.
6A94B0   01000000  B8000000  0500DA01  03000000    ....&#184;.....&#218;.....
6A94C0   F0030000  08410000  0300DA01  4C4B0000    .....A....&#218;.LK..
6A94D0   0500DA01  01000000  B8000000  0500DA01    ..&#218;.....&#184;.....&#218;.
6A94E0   03000000  F0030000  08410000  0300DA01    .........A....&#218;.
6A94F0   07000000  00090000  50140000  0100DA01    ........P.....&#218;.
6A9500   01000000  B0000000  0100DA01  03000000    ....°.....&#218;.....
6A9510   E8030000  03220000  0100DA01  98170000    &#232;...."..,.&#218;.˜...
6A9520   0200DA01  01000000  B4000000  0200DA01    ..&#218;.....´.....&#218;.
6A9530   03000000  E8030000  03220000  0100DA01    ....&#232;...."....&#218;.
6A9540   00000000  00000000  08000000  3C0A0000    ............<...
6A9550   50140000  0100DA01  01000000  B0000000    P.....&#218;.....°...
6A9560   0100DA01  03000000  E8030000  03220000    ..&#218;.....&#232;...."..
6A9570   0100DA01  08180000  0200DA01  01000000    ,.&#218;.......&#218;.....
6A9580   B4000000  0200DA01  03000000  E8030000    ´.....&#218;.....&#232;...
6A9590   03220000  0100DA01  00000000  00000000    ."....&#218;.........
6A95A0   00000000  00000000  00000000  00000000    ................
6A95B0 to 6A9650 suppressed, lines same as above
6A9660   01000000  00000000  78180000  0200DA01    ....,...x.....&#218;.
6A9670   01000000  B4000000  0200DA01  03000000    .... ´.....&#218;.....
6A9680   E8030000  03220000  0100DA01  09000000    &#232;...."....&#218;.....
6A9690   780B0000  50140000  0100DA01  01000000    x...P.....&#218;.....
6A96A0   B0000000  0100DA01  03000000  E8030000    °.....&#218;.....&#232;...
6A96B0   03220000  0100DA01  00000000  00000000    ."....&#218;.........
6A96C0   00000000  00000000  00000000  00000000    ................
6A96D0 to 6A9750 suppressed, lines same as above
6A9760   00000000  00000000  DD012020  00000000    ........&#221;.  ....
6A9770   00000000  00000000  01000000  09000000    ................
6A9780   00000000                                  ....
```

Funktions-Stack und Trace-Protokoll werden von IBM zur Unterstützung der Fehlerbestimmung verwendet. In den meisten Fällen kann der Systemadministrator nur wenig tun, wenn eine FFST-Datei generiert wird, außer dass er Fehler an die Support Center weiterleitet.

Es gibt jedoch eine Gruppe von Fehlern, die er beheben kann. Wenn die FFST-Datei beim Aufrufen einer der internen Funktionen die Beschreibung „quota exceeded" (Quota überschritten) oder „out of space on device" (Speicherengpass auf Einheit) enthält, liegt das häufig daran, dass der betreffende SYSGEN-Parametergrenzwert überschritten wurde.

Sie können den Fehler beheben, indem Sie die Systemparameter anpassen und die internen Grenzwerte erhöhen. Weitere Informationen finden Sie in „Kapitel 13. MQSeries konfigurieren" auf Seite 179.

Fehlerbestimmung bei Clients

Eine MQI-Client-Anwendung empfängt MQRC_*-Ursachencodes auf dieselbe Weise wie jede andere MQI-Anwendung. Allerdings gibt es jetzt zusätzliche Ursachencodes für Fehlerbedingungen in Verbindung mit Clients. Beispiel:

- ferne Maschine antwortet nicht
- Fehler auf der Übertragungsleitung
- ungültige Maschinenadresse

Fehler treten in den meisten Fällen dann auf, wenn eine Anwendung einen Aufruf MQCONN ausgibt und die Antwort MQRC_Q_MQR_NOT_AVAILABLE erhält. Die Fehlerursache wird in einer Fehlernachricht, die in die Client-Protokolldatei geschrieben wird, erläutert. Nachrichten können auch auf dem Server protokolliert werden; das ist von der Art des Fehlers abhängig.

Clients beenden

Auch nachdem ein Client beendet wurde, ist es möglich, dass der Prozess auf dem Server seine Warteschlangen weiter geöffnet hält. Dies ist in der Regel nur kurze Zeit der Fall, bis die Übertragungsschicht erkennt, dass der Partner nicht mehr vorhanden ist.

Fehlernachrichten bei Clients

Wenn auf einem Client-System ein Fehler auftritt, werden Fehlernachrichten, falls möglich, in die dem Server zugeordneten Fehlerdateien geschrieben. Kann eine Fehlernachricht dort nicht abgelegt werden, versucht der Client-Code, die Fehlernachricht in ein Fehlerprotokoll im Stammverzeichnis der Client-Maschine zu schreiben.

OS/2-, UNIX- und OpenVMS-Clients

Fehlernachrichten für OS/2-, UNIX- und OpenVMS-Clients werden in die Fehlerprotokolle der jeweiligen MQSeries-Serversysteme geschrieben. Diese Dateien stehen in der Regel auf OpenVMS-Systemen im Verzeichnis `MQS_ROOT:[MQM.ERRORS]` und auf UNIX-Systemen im Verzeichnis `/var/mqm/errors`.

DOS- und Windows-Clients

Das Verzeichnis der Protokolldatei AMQERR01.LOG wird durch die Umgebungsvariable MQDATA festgelegt. Das Standardverzeichnis, sofern es nicht von MQDATA überschrieben wird, ist:

```
C:\
```

In der DOS-Umgebung wird die Umgebungsvariable MQDATA verwendet.

Dies ist die Standardbibliothek, die vom Client-Code zum Speichern von Trace- und Fehlerinformationen verwendet wird; sie enthält außerdem den Namen des Verzeichnisses, in dem die Datei qm.ini gespeichert wird (wird für NetBIOS-Konfiguration benötigt). Ist kein Verzeichnis angegeben, wird standardmäßig Laufwerk C verwendet.

Die Namen der Standarddateien in diesem Verzeichnis lauten:

AMQERR01.LOG
: Für Fehlernachrichten

AMQERR01.FDC
: Für FFDC-Nachrichten (First Failure Data Capture)

Client-Fehlerbestimmung

Kapitel 15. Leistungsoptimierung

Dieses Kapitel erläutert, wie Sie die Leistung Ihres OpenVMS-Systems verbessern können, um MQSeries optimal zu nutzen.

Es ist für ein Produkt wie MQSeries nicht möglich, Werte für die verschiedenen OpenVMS-Optimierungsparameter zu definieren, die unter allen Bedingungen immer korrekt sind. Die effektivsten Werte werden durch die Auslastung für MQSeries selbst und für das OpenVMS-System als Ganzes bestimmt. Obwohl die Parametereinstellungen, die im Handbuch *MQSeries for Compaq OpenVMS Alpha, Version 5.1 Einstieg* beschrieben werden, sinnvolle Mindest- bzw. Anfangswerte angeben, kann es erforderlich sein, diese Werte zu erhöhen, wenn die Auslastung der WS-Manager zunimmt. Dieser Vorgang wird 'Optimierung' genannt.

Die Leistungsoptimierung von OpenVMS-Systemen wird im Handbuch *OpenVMS Performance Management* beschrieben. Verwenden Sie die Informationen in diesem Buch sowie die im Folgenden beschriebenen Informationen, die sich speziell auf MQSeries beziehen:

- Einige Optimierungsparameter betreffen das System als Ganzes (z. B. GBLPAGES). Diese Parameter werden vom Dienstprogramm SYSGEN gesteuert und deshalb manchmal auch als SYSGEN-Parameter bezeichnet. Bei einer normalen Verwendung überwacht der Mechanismus AUTOGEN FEEDBACK die vom System verwendeten Ressourcen und passt SYSGEN-Parameter automatisch an die sich verändernde Auslastung an. Dadurch können manuelle Eingriffe erheblich reduziert werden, die erforderlich sind, um eine optimale Systemleistung aufrechtzuerhalten und Fehler infolge mangelnder Ressourcen zu vermeiden. Die folgenden SYSGEN-Parameter sind für MQSeries relevant:

  GBLPAGES, GBLSECTIONS und GBLPAGFIL
  Der WS-Manager wird als eine Gruppe kooperierender Prozesse implementiert, die über einen gemeinsamen (globalen) Speicher miteinander kommunizieren. Daher ist es wichtig sicherzustellen, dass die SYSGEN-Parameter zur Steuerung des globalen Speichers (GBLPAGES, GBLSECTIONS und GBLPAGFIL) ausreichend große Werte enthalten. Im Handbuch *MQSeries for Compaq OpenVMS Alpha, Version 5.1 Einstieg* sind sinnvolle Anfangswerte für diese Parameter angegeben. Je größer die Anzahl der Benutzer von MQSeries wird, desto mehr globaler Speicher wird benötigt, und Sie müssen gegebenenfalls die Werte für die entsprechenden SYSGEN-Parameter erhöhen.

  CHANNELCNT
  Die WS-Manager-Prozesse verwenden die OpenVMS-Mailboxes für die Interprozesskommunikation und als Synchronisationsmechanismus. Auf diese Mailboxes wird über Kanäle zugegriffen, so dass sichergestellt werden muss, dass der Wert für den SYSGEN-Parameter CHANNELCNT groß genug ist. In den meisten Fällen ist der Wert, der bei der Installation festgelegt wird, ausreichend. Bei stark ausgelasteten Systemen mit vielen aktiven MQSeries-Prozessen muss der Wert aber möglicherweise erhöht werden.

  Um SYSGEN-Parameter explizit festzulegen, müssen Sie die Datei MODPARAMS.DAT wie im Handbuch *OpenVMS Performance Management* beschrieben ändern.

- Einige OpenVMS-Optimierungsparameter betreffen einzelne Benutzernamen oder Prozesse (z. B. PGFLQUOTA). Diese Parameter werden in der Regel (aber nicht immer) vom Dienstprogramm AUTHORIZE gesteuert. Es gibt kein automatisches Verfahren zum Anpassen dieser Parameter. Da diese Parameter jedoch die Größe einiger Ressourcen, die von einem einzelnen Prozess verwendet werden können, begrenzen, möchten Sie deren Werte möglicherweise höher festlegen als eigentlich nötig, um Zusatzkapazitäten für gelegentliche Spitzenbelastungen bereitzustellen. Die folgenden prozessspezifischen Parameter sind für MQSeries relevant:

  PGFLQUOTA
  Dieser Parameter steuert die Größe des Seitendateispeichers, den ein Prozess verwenden darf. Da MQSeries-Prozesse in der Regel sehr große und/oder sehr viele Nachrichten verarbeitet, benötigen Sie möglicherweise große Seitendateispeicher.

  PRCLM
  Dieser Parameter steuert die Anzahl der Unterprozesse, die ein angegebener Prozess erstellen kann. Da die meisten MQSeries-Prozesse als Unterprozesse des Ausführungs-Controllers erstellt werden, ist für PRCLM ein hoher Wert erforderlich.

  ENQLM, ASTLM, TQELM
  Wie bereits erwähnt, wird der WS-Manager als eine Gruppe kooperierender Prozesse implementiert. Diese Prozesse synchronisieren ihre Aktivitäten über den OpenVMS-Sperrenmanager, asynchrone Systemabfangpositionen (ASTs, Asynchronous System Traps) und Zeitgeber. Die Werte für diese drei Parameter, mit denen die Verwendung dieser Ressourcen begrenzt wird, müssen groß genug sein, um die Anforderungen des WS-Managers zu erfüllen.

Die Werte für prozessspezifische Parameter festlegen

Diese Parameter werden normalerweise mit dem Dienstprogramm AUTHORIZE festgelegt, über das der Wert für den jeweiligen Benutzernamen angepasst wird (für MQSeries ist dies in der Regel der Wert MQM).

Unter OpenVMS werden einige Prozess-Quotas jedoch von allen Prozessen desselben Jobs gemeinsam benutzt, d. h. von einem Elternprozess und allen anderen Prozessen, die Unterprozesse dieses Elternprozesses sind. Prozess-Quotas, die in diese Kategorie gehören, sind BYTLM, FILLM, PGFLQUOTA, PRCLM, TQELM und ENQLM, die als *Quota-Pool* bezeichnet werden.

Da die meisten WS-Manager-Prozesse als Unterprozesse des Ausführungs-Controllers erstellt werden, folgt daraus, dass die Quota-Pools von allen WS-Manager-Prozessen gemeinsam benutzt werden. Deshalb ist es normalerweise erforderlich, für diese Quotas Werte festzulegen, die für einen einzelnen Prozess eigentlich zu hoch erscheinen. Dies gilt insbesondere für PGFLQUOTA, da dieser Parameter die Größe des virtuellen Speichers begrenzt, den die WS-Manager-Prozesse **insgesamt** erstellen können. Aus diesem Grund übernimmt der Ausführungs-Controller bei seinem Start nicht die Quota-Anfangswerte aus der Berechtigungsdatei, die über das Dienstprogramm AUTHORIZE verwaltet wird, sondern legt selbst geeignete Werte fest. Dies hat zur Folge, dass AUTHORIZE nicht mehr zum Ändern dieser Werte verwendet werden kann.

Stattdessen können Sie die explizit festgelegten Quota-Werte mit Hilfe der folgenden logischen Namen überschreiben:

```
MQS_ASTLM
MQS_BIOLM
MQS_BYTLM
MQS_DIOLM
MQS_ENQLM
MQS_FILLM
MQS_PGFLQUOTA
MQS_PRCLM
MQS_TQELM
```

Diese logischen Namen können Sie in der Datei SYS$MANAGER:MQS_SYSTARTUP.COM festlegen. MQSeries stellt eine Datei mit dem Namen SYS$MANAGER:MQS_SYSTARTUP.TEMPLATE zur Verfügung, die Sie editieren und umbenennen können. So können Sie zum Beispiel einen anderen Wert für den Parameter PGFLQUOTA angeben:

1. Kopieren Sie die TEMPLATE-Datei MQS_SYSTARTUP.TEMPLATE in eine COM-Datei.
2. Editieren Sie die Datei MQS_SYSTARTUP.COM, die Sie auf diese Weise erstellt haben, indem Sie die Kommentarzeichen in den Zeilen entfernen, in denen die logischen Namen für die Prozess-Quotas, z. B. mqs_pgflquota, definiert sind.
3. Definieren Sie neue Werte.

   Beispiel:

```
$! DEFINE/SYSTEM MQS_PGFLQUOTA 1000000
```

   kann geändert werden in

```
$  DEFINE/SYSTEM MQS_PGFLQUOTA 5000000
```

4. Rufen Sie die Datei mqs_systartup auf, um die logischen Namen zu definieren. Beispiel:

```
$ @sys$manager:mqs_systartup
```

   Dies geschieht in der Regel während des Systemstarts.

Dass zu wenig Quota-Pools verfügbar sind, wird in der Regel dann erkennbar, wenn eine neue Client-Anwendung keine Verbindung mit einem WS-Manager herstellen kann oder eine andere Anwendung fehlschlägt, weil die neue Verbindung zu viele Ressourcen verbraucht.

Beachten Sie auch, dass die SYSGEN PQL_D*-Parameter nicht für den Ausführungs-Controller gelten, da dieser mit expliziten Einstellungen für viele seiner Quotas gestartet wird.

Kapitel 16. MQSeries für OpenVMS und Cluster-Umgebungen

OpenVMS-Cluster und *MQSeries-WS-Manager-Cluster* sind zwei verschiedene Dinge, die voneinander unabhängig sind.

Anmerkung: Der Begriff *Cluster* bezieht sich hier auf einen MQSeries-WS-Manager-Cluster. Ein OpenVMS-Cluster wird auch immer als *OpenVMS-Cluster* bezeichnet.

MQSeries-WS-Manager-Cluster verwenden nicht notwendigerweise Interkommunikationsprotokolle für OpenVMS-Cluster, den verteilten Sperrenmanager für OpenVMS-Cluster oder das Dateisystem für OpenVMS-Cluster. Die gesamte Kommunikation zwischen WS-Managern in einem MQSeries-Cluster erfolgt über MQSeries-Kanäle und eines der unterstützten Protokolle. Dadurch ist es möglich, MQSeries-WS-Manager-Cluster mit WS-Managern zu konfigurieren, die auf OpenVMS-Systemen ausgeführt werden, die nicht Teil desselben OpenVMS-Clusters sind.

Wenn ein MQSeries-WS-Manager in einem OpenVMS-Cluster konfiguriert wird, kann der MQSeries-WS-Manager zu einer Zeit nur auf einem OpenVMS-Knoten (wird im Verlaufe dieses Kapitels einfach als Knoten bezeichnet) innerhalb des OpenVMS-Clusters ausgeführt werden. Die Funktion eines einzelnen MQSeries-WS-Managers kann nicht über mehrere OpenVMS-Knoten innerhalb eines OpenVMS-Clusters verteilt werden. Bei dem Versuch, einen MQSeries-WS-Manager auf mehreren OpenVMS-Knoten zu starten, wird eine Fehlernachricht zurückgegeben. Wenn jedoch mehrere MQSeries-WS-Manager in einem OpenVMS-Cluster konfiguriert sind, können sie auf verschiedenen OpenVMS-Knoten innerhalb des OpenVMS-Clusters ausgeführt werden.

Um eine bessere Verfügbarkeit von MQSeries-WS-Managern in einem OpenVMS-Cluster bereitzustellen, wurde mit MQSeries V5.1 eine neue Funktion, die so genannten Überbrückungsgruppen, eingeführt. Sie machen es möglich, einen WS-Manager nach einem Fehler automatisch auf einem anderen OpenVMS-Cluster-Knoten erneut zu starten. Diese Funktion kann mit und ohne MQSeries-WS-Manager-Cluster verwendet werden (siehe „Überbrückungsgruppen für OpenVMS-Cluster" auf Seite 230).

MQSeries in einem OpenVMS-Cluster installieren

Die Installation von MQSeries for Compaq OpenVMS Alpha V5.1 in einem OpenVMS-Cluster unterscheidet sich kaum von der Installation von MQSeries auf einem standalone OpenVMS-System. Bevor Sie mit der Installation beginnen, sollten Sie jedoch die folgenden Punkte überprüfen:

- Wenn es in dem OpenVMS-Cluster mehrere Systemdatenträger gibt, muss MQSeries auf jedem Systemdatenträger installiert werden, von dem ein Knoten gebootet wird *und* auf dem MQSeries ausgeführt werden soll. MQSeries muss nur ein Mal pro Systemdatenträger installiert werden, und nicht ein Mal pro Knoten.
- Der Datenträger mit der Verzeichnisstruktur MQS_ROOT muss systemweit bei den OpenVMS-Knoten angemeldet sein, auf denen die WS-Manager, die in der Verzeichnisstruktur enthalten sind, ausgeführt werden sollen. Es kann für jeden Knoten verschiedene MQS_ROOT-Verzeichnisstrukturen geben. Wenn jedoch

Überbrückungsgruppen konfiguriert werde sollen, muss jeder OpenVMS-Knoten in einer Überbrückungsgruppe auf dieselbe MQS_ROOT-Verzeichnisstruktur verweisen. Wenn Sie bei der Installation von MQSeries zum Eingeben des Stammdatenträgers für die MQSeries-Datendateien aufgefordert werden, müssen Sie das Verzeichnis MQS_ROOT für jede Installation angeben.

- Wenn der Datenträger mit den Protokolldateien für einen WS-Manager nicht derselbe ist wie der mit MQS_ROOT, muss der Datenträger mit den Protokolldateien systemweit bei allen Knoten in einer Überbrückungsgruppe angemeldet werden.
- MQSeries verwendet das Konto MQM, dem das Standardverzeichnis SYS$SPECIFIC:[MQS_SERVER] zugeordnet ist. Dieses Verzeichnis wird nur für den Knoten erstellt, auf dem MQSeries installiert ist. Das Verzeichnis muss für jeden zusätzlichen Knoten erstellt werden, der von demselben Datenträger gebootet wird *und* auf dem MQSeries ausgeführt werden soll. Dies geschieht dadurch, dass auf jedem zusätzlichen Knoten die folgenden DCL-Befehle ausgeführt werden:

```
$create/directory sys$specific:[mqs_server]/owner=[mqs_server] -
/protection=(s:rwed,o:rwed,g,w)
$set sec/acl=(identifier=mqm,options=default,access=r+w+e+d+c) -
sys$specific:[000000]mqs_server.dir
$set sec/acl=(identifier=mqm,access=r+w+e+d+c) -
sys$specific:[000000]mqs_server.dir
```

Überbrückungsgruppen für OpenVMS-Cluster

Überbrückungsgruppen für OpenVMS-Cluster - Übersicht

Überbrückungsgruppen für OpenVMS-Cluster sind als neue Funktion in MQSeries for Compaq OpenVMS V5.1 verfügbar. Mit ihrer Hilfe können MQSeries-WS-Manager nach einem Fehler des MQSeries-WS-Managers automatisch auf einem anderen OpenVMS-Knoten in einem OpenVMS-Cluster erneut gestartet werden. Von den Überbrückungsgruppen für OpenVMS-Cluster werden die folgenden Arten von Fehlern unterstützt:

- Anhalten eines OpenVMS-Knotens, auf dem ein MQSeries-WS-Manager aktiv ist.
- Systemabsturz eines OpenVMS-Knotens, auf dem ein MQSeries-WS-Manager aktiv ist.
- Systemabschluss eines OpenVMS-Knotens, auf dem ein MQSeries-WS-Manager aktiv ist, ohne dass der MQSeries-WS-Manager ordnungsgemäß beendet wurde.
- Fehler in einem Prozess des Ausführungs-Controllers eines MQSeries-WS-Managers.

Von den Überbrückungsgruppen für OpenVMS-Cluster werden die folgenden Arten von Fehlern nicht unterstützt:

- eine Störung auf einem OpenVMS-Knoten, auf dem ein MQSeries-WS-Manager aktiv ist, die jedoch nicht zu einem Fehler des Knotens oder des MQSeries-WS-Managers führt.
- ein Fehler im Prozess eines MQSeries-WS-Managers, mit Ausnahme des Ausführungs-Controllers. Ein MQSeries-WS-Manager wird niemals auf demselben Knoten automatisch erneut gestartet.
- ein Software- oder Hardwarefehler auf dem Datenträger mit den Warteschlangendateien und Protokolldaten des MQSeries-WS-Managers.

- Beschädigung der Warteschlangendateien oder Protokolldaten des MQSeries-WS-Managers.

Überbrückungsgruppen für OpenVMS-Cluster werden nur für WS-Manager unterstützt, die für MQSeries-Kanäle das TCP/IP-Protokoll verwenden. Die folgenden TCP/IP-Stacks werden unterstützt:

- Digital TCP/IP Services für OpenVMS V5.0A
- Porcess Software's TCPware für OpenVMS V5.4
- Process Software's Multinet für OpenVMS 4.3

Überbrückungsgruppen für OpenVMS-Cluster - Konzepte

Eine Überbrückungsgruppe für OpenVMS-Cluster ist eine Gruppe aus OpenVMS-Knoten, die in der Lage sind, einen MQSeries-WS-Manager auszuführen. Eine Überbrückungsgruppe für OpenVMS-Cluster kann aus ein bis vier OpenVMS-Knoten bestehen, wobei alle OpenVMS-Knoten Mitglied desselben OpenVMS-Clusters sein müssen. Eine Überbrückungsgruppe für OpenVMS-Cluster ist spezifisch für einen einzigen MQSeries-WS-Manager. In einem OpenVMS-Cluster können mehrere Überbrückungsgruppen für OpenVMS-Cluster konfiguriert werden. Beachten Sie, dass die maximale Länge eines WS-Manager-Namens, die von Überbrückungsgruppen für OpenVMS-Cluster unterstützt wird, 25 Zeichen beträgt.

Überbrückung ist der Prozess, bei dem ein MQSeries-WS-Manager auf einem anderen OpenVMS-Knoten erneut gestartet wird, nachdem ein unterstützter Fehler aufgetreten ist. Nach Abschluss dieses Prozesses wird davon gesprochen, dass für den MQSeries-WS-Manager eine Überbrückung durchgeführt wurde.

Zurückübertragung ist der Prozess, bei dem ein MQSeries-WS-Manager auf seinem ursprünglichen OpenVMS-Knoten erneut gestartet wird, nachdem ein Fehler behoben wurde. Überbrückungsgruppen für OpenVMS-Cluster unterstützen keine automatische Zurückübertragung, sie kann jedoch manuell ausgeführt werden. Nach Abschluss dieses Prozesses wird davon gesprochen, dass für den MQSeries-WS-Manager eine Zurückübertragung durchgeführt wurde.

Ein *Überbrückungsmonitor* ist ein Prozess, der auf jedem Mitglied einer Überbrückungsgruppe für OpenVMS-Cluster ausgeführt wird. Die Überbrückungsmonitore sind für die Ausführung aller Funktionen der Überbrückungsgruppen zuständig. Die Überbrückungsmonitore in einer Überbrückungsgruppe für OpenVMS-Cluster arbeiten zusammen, um diese Funktionen bereitzustellen. Ein Überbrückungsmonitor wird mit dem Befehl **runmqfm** gestartet. (Weitere Informationen zu diesem Befehl finden Sie unter „runmqfm (Überbrückungsmonitor starten)" auf Seite 301.)

Einer der Überbrückungsmonitore wird als *Beobachtungsmonitor* ernannt, und dieser Überbrückungsmonitor befindet sich in einem Status der *Beobachtung* (watching). Der erste Überbrückungsmonitor, der in einer Überbrückungsgruppe gestartet wird, ist gleichzeitig der erste Beobachtungsmonitor. Eine Überbrückungsgruppe wird *aktiv*, sobald der erste Überbrückungsmonitor gestartet wird. Wenn der Beobachtungsmonitor oder der OpenVMS-Knoten, auf dem er ausgeführt wird, fehlschlägt, wir automatisch ein anderer Überbrückungsmonitor zum Beobachtungsmonitor ernannt. Der Beobachtungsmonitor überprüft, ob der MQSeries-WS-Manager aktiv ist, und leitet eine Überbrückungsoperation ein, wenn ein unterstützter Fehler auftritt. Operationen, die auf einem anderen OpenVMS-Knoten ausgeführt werden müssen, werden vom Beobachtungsmonitor an den Überbrückungsmonitor auf dem OpenVMS-Knoten weitergeleitet, von dem die Operation tatsächlich ausgeführt wird.

Überbrückungsgruppen für OpenVMS-Cluster werden mit dem DCL-Befehl **failover** verwaltet. Der Befehl **failover** kann auf jedem Knoten in der Überbrückungsgruppe für OpenVMS-Cluster verwendet werden. Alle Befehle werden an den Beobachtungsmonitor gesendet, der dann entscheidet, welcher Überbrückungsmonitor den Befehl verarbeiten soll, und ihn gegebenenfalls an einen anderen Überbrückungsmonitor weiterleitet.

Die Konfigurationsdatei für die Überbrückungsgruppe für OpenVMS-Cluster enthält die Details zur Überbrückungsgruppe für OpenVMS-Cluster, einschließlich Anzahl und Namen der OpenVMS-Knoten. Die Datei hat den Namen `FAILOVER.INI` und befindet sich im Verzeichnis `MQS_ROOT:[MQM.QMGRS.WS-Manager-Name]`. Es handelt sich um eine Textdatei, die in einem Texteditor geändert werden kann und vor dem Start des ersten Überbrückungsmonitors erstellt werden muss. Eine Schablone für die Konfigurationsdatei mit dem Namen `FAILOVER.TEMPLATE` wird im Verzeichnis `MQS_EXAMPLES` bereitgestellt. Die Parameter in der Konfigurationsdatei können nicht dynamisch geändert werden. Damit eine Änderung wirksam wird, müssen alle Überbrückungsmonitore gestoppt und erneut gestartet werden. Dabei ist Sorgfalt geboten, weil eine automatische Überbrückung des MQSeries-WS-Managers nur stattfinden kann, wenn die Überbrückungsmonitore gestartet wurden.

Für einen MQSeries-WS-Manager in einer Überbrückungsgruppe können alle MQSeries-Befehle normal ausgeführt werden, mit Ausnahme der Befehle **strmqm** und **endmqm**. Diese beiden Befehle geben eine Fehlernachricht zurück, wenn ein MQSeries-WS-Manager Mitglied einer aktiven Überbrückungsgruppe ist. Zum Starten und Beenden des MQSeries-WS-Managers muss der Befehl **failover** verwendet werden.

Für jeden OpenVMS-Knoten in der Überbrückungsgruppe für OpenVMS-Cluster gilt die Priorität des OpenVMS-Knotens; über diese Priorität wird bestimmt, auf welchem OpenVMS-Knoten der WS-Manager nach einem Fehler gestartet werden soll. Der OpenVMS-Knoten mit dem niedrigsten numerischen Prioritätswert besitzt die höchste Priorität.

Die TCP/IP-Adresse der Überbrückungsgruppe für OpenVMS-Cluster ist die TCP/IP-Adresse, die der Überbrückungsgruppe zugewiesen wird. Alle Kanäle, die auf den WS-Manager der Überbrückungsgruppe verweisen, müssen so konfiguriert werden, dass der Verbindungsname die TCP/IP-Adresse enthält. Jede Überbrückungsgruppe für OpenVMS-Cluster muss eine eindeutige TCP/IP-Adresse verwenden. Alle OpenVMS-Knoten in der Überbrückungsgruppe für OpenVMS-Cluster müssen über eine TCP/IP-Schnittstellenadresse in demselben Teilnetz verfügen, und die TCP/IP-Adresse der Überbrückungsgruppe für OpenVMS-Cluster muss sich in demselben Teilnetz befinden.

Konfiguration einer Überbrückungsgruppe für OpenVMS-Cluster vorbereiten

Vor der Konfiguration einer Überbrückungsgruppe für OpenVMS-Cluster müssen Sie folgende Schritte ausführen:

1. Erstellen Sie den WS-Manager mit dem Befehl **crtmqm**, sofern er nicht bereits vorhanden ist.
2. Verschaffen Sie sich eine TCP/IP-Adresse für die Überbrückungsgruppe für OpenVMS-Cluster.

3. Erstellen oder ändern Sie MQSeries-Kanäle, so dass sie die TCP/IP-Adresse der Überbrückungsgruppe für OpenVMS-Cluster verwenden.
4. Entscheiden Sie, welche OpenVMS-Knoten Mitglied der Überbrückungsgruppe für OpenVMS-Cluster sein sollen, und legen Sie deren Prioritäten fest.
5. Stellen Sie sicher, dass der logische Name MQS_ROOT auf dasselbe Verzeichnis auf allen OpenVMS-Knoten in der Überbrückungsgruppe für OpenVMS-Cluster verweist und dass der Datenträger systemweit bei allen Knoten angemeldet ist. Die Datenträger mit dem Verzeichnis `MQS_ROOT` und den Protokolldateien sollten nicht über MSCP (Mass Storage Control Protocol) von einem Knoten in der Überbrückungsgruppe für einen anderen Knoten bereitgestellt werden, denn wenn der Knoten, der den Datenträger bereitstellt, ausfällt, kann der Knoten, für den der Datenträger bereitgestellt wird, nicht mehr auf den Datenträger zugreifen.

Eine Überbrückungsgruppe für OpenVMS-Cluster konfigurieren

Gehen Sie folgendermaßen vor, um eine Überbrückungsgruppe für OpenVMS-Cluster zu konfigurieren:

1. Kopieren Sie die Datei `MQS_EXAMPLES:FAILOVER.TEMPLATE` in die Datei `MQS_ROOT:[MQM.QMGRS.`*`WS-Manager-Name`*`]FAILOVER.INI`.
2. Editieren Sie die Datei `MQS_ROOT:[MQM.QMGRS.`*`WS-Manager-Name`*`]FAILOVER.INI`, und ändern Sie sie für diese Konfiguration einer Überbrückungsgruppe für OpenVMS-Cluster (siehe „Die Konfigurationsdatei FAILOVER.INI editieren” auf Seite 234).
3. Editieren Sie die Befehlsprozeduren **START_QM.COM, END_QM.COM** und **TIDY_QM.COM** (siehe „Von Überbrückungsgruppen verwendete Befehlsprozeduren” auf Seite 235).
4. Definieren Sie die ICC-Sicherheit für die ICC-Zuordnung, die von den Überbrückungsmonitoren verwendet wird (siehe „Sicherheit für ICC-Zuordnungen einstellen” auf Seite 243.)
5. Starten Sie mit dem Befehl **runmqfm -m** *WS-Manager-Name* einen Überbrückungsmonitor auf jedem Knoten in der Überbrückungsgruppe für OpenVMS-Cluster.
6. Starten Sie den WS-Manager mit dem Befehl **failover -m** *WS-Manager-Name* **-n** *Knotenname* **-s**.
7. Ändern Sie den standortspezifischen Systemabschluss, damit:
   - der WS-Manager beendet oder versetzt wird, wenn er auf einem Knoten aktiv ist, für den ein Systemabschluss durchgeführt wird.
   - der Überbrückungsmonitor angehalten wird.

Tasks nach Abschluss der Konfiguration der Überbrückungsgruppe für OpenVMS-Cluster

Die folgenden Tasks können Sie ausführen, nachdem Sie die Cluster-Überbrückungsgruppe konfiguriert haben:

- Editieren Sie die Systemstartdatei, so dass die Überbrückungsmonitore mit dem Befehl **runmqfm** auf allen Knoten in der Überbrückungsgruppe für OpenVMS-Cluster gestartet werden. Der Befehl **runmqfm** sollte hinter dem Befehl zum Starten von MQSeries stehen.
- Wenn der WS-Manager automatisch beim Systemstart gestartet werden soll, fügen Sie einen Befehl in die Systemstartdatei auf dem betreffenden Knoten ein, so dass der WS-Manager gleich nach dem Überbrückungsmonitor gestartet wird. Der Befehl zum Starten des WS-Managers auf einem Knoten lautet wie folgt: **failover -m** *WS-Manager-Name* **-n** *Knotenname* **-s**.
- Ändern Sie den standortspezifischen Systemabschluss, damit die Überbrückungsmonitore beim Systemabschluss beendet werden. Auch muss der WS-Manager beendet oder versetzt werden, wenn er auf einem Knoten aktiv ist, für den ein Systemabschluss durchgeführt wird.

Die Konfigurationsdatei FAILOVER.INI editieren

Die Datei `FAILOVER.INI` muss für jede Überbrückungsgruppe für OpenVMS-Cluster angepasst werden. Die Bedeutung der einzelnen Felder ist in Tabelle 9 auf Seite 234 beschrieben.

Die Schablone für die im Verzeichnis `MQS_EXAMPLES` bereitgestellte Konfigurationsdatei finden Sie in „Anhang F. Schablonen für OpenVMS-Cluster-Überbrückungsgruppen" auf Seite 341. Alle Zeilen in der Datei, die mit dem Zeichen '#' beginnen, werden beim Lesen vom Überbrückungsmonitor ignoriert. Die Groß-/Kleinschreibung der Feldnamen muss so sein, wie in der Schablonendatei angegeben. Hinter jedem Feldnamen muss das Zeichen '=' stehen, gefolgt vom jeweiligen Wert. Alle Felder in der Schablonendatei sind obligatorisch, d. h., es dürfen keine Felder entfernt werden.

Tabelle 9. Beschreibung der Felder in der Datei FAILOVER.INI

Feldname	Beschreibung
IpAddress	Die TCP/IP-Adresse, die von der Überbrückungsgruppe verwendet wird.
PortNumber	Die TCP/IP-Port-Nummer, die vom Empfangsprogramm des WS-Managers verwendet wird.
TimeOut	Dieser Zeitlimitwert wird an die Prozedur EndCommand übergeben (siehe „Von Überbrückungsgruppen verwendete Befehlsprozeduren" auf Seite 235).
StartCommand	Die Befehlsprozedur, über die der WS-Manager gestartet wird.
EndCommand	Die Befehlsprozedur, über die der WS-Manager beendet wird.
TidyCommand	Die Befehlsprozedur zum Aufräumen eines Knotens nach einem WS-Manager-Fehler, den der OpenVMS-Knoten unbeschadet überstanden hat.
LogDirectory	Das Verzeichnis mit den Protokolldateien, die von den Prozeduren StartCommand, EndCommand und TidyCommand erstellt werden.

Tabelle 9. Beschreibung der Felder in der Datei FAILOVER.INI (Forts.)

Feldname	Beschreibung
NodeCount	Die Anzahl der Knoten in der Überbrückungsgruppe. Für jeden Knoten müssen die drei letzten in dieser Tabelle aufgeführten Felder definiert werden, d. h., die Anzahl dieser Dreiergruppen muss der hier angegebenen Anzahl der Knoten entsprechen. Die maximale Anzahl der unterstützten Knoten ist vier.
NodeName	Der Name des Knotens. Dies ist der Wert, der für den OpenVMS-Systemparameter SCSNODE angegeben wurde.
Interface	Der Name der TCP/IP-Schnittstelle für den Knoten bei Verwendung des 'Digital TCP/IP Services für OpenVMS TCP/IP'-Stacks. Diesen Namen können Sie der Ausgabe des Schnittstellenbefehls $tcpip entnehmen. Dieses Feld wird bei Verwendung des 'TCPware für OpenVMS TCP/IP'- oder 'Multinet für OpenVMS TCP/IP'-Stacks nicht benötigt, es sollte jedoch der Standardwert 'we0' angegeben werden. (Entfernen Sie dieses Feld nicht aus der Konfigurationsdatei.)
Priority	Dies ist die Priorität des Knotens in der Überbrückungsgruppe. Der Wert muss im Bereich von 1 bis 10 liegen. Der Wert 1 bezeichnet die höchste Priorität. Mehrere Knoten können dieselbe Priorität haben. Bei einem Fehler oder wenn in einem **failover**-Befehl mit der Option -s oder -f kein bestimmter Knoten angegeben ist, wird der WS-Manager auf dem verfügbaren Knoten mit der höchsten Priorität gestartet.

Von Überbrückungsgruppen verwendete Befehlsprozeduren

Überbrückungsgruppen verwenden drei Befehlsprozeduren, um einige ihrer Funktionen zu implementieren. Die Verzeichnisse dieser Befehlsprozeduren werden in der Konfigurationsdatei `FAILOVER.INI` in den Feldern StartCommand, EndCommand und TidyCommand angegeben. Schablonendateien für diese Befehle mit den Namen **START_QM.TEMPLATE**, **END_QM.TEMPLATE** und **TIDY_QM.TEMPLATE** werden im Verzeichnis `MQS_EXAMPLES` bereitgestellt. Diese Dateien sind in „Anhang F. Schablonen für OpenVMS-Cluster-Überbrückungsgruppen" auf Seite 341 abgebildet.

Den Befehlsprozeduren werden fünf oder sechs Parameter übergeben. Diese Parameter werden in Tabelle 10 auf Seite 235 aufgelistet:

Tabelle 10. Parameter, die an Befehlsprozeduren übergeben werden

Parameter	Wert
P1	Name des WS-Managers
P2	Name des WS-Manager-Verzeichnisses
P3	Cluster-TCP/IP-Adresse
P4	Name der Knotenschnittstelle
P5	Port-Nummer des Empfangsprogramms
P6	Zeitlimit zum Beenden des WS-Managers (nur Prozedur EndCommand)

Die Prozedur StartCommand wird in folgenden Situationen zum Starten des WS-Managers verwendet:

- Es erfolgt eine explizite Angabe mit der Option -s des Befehls **failover**.
- Der WS-Manager wird mit der Option -f des Befehls **failover** auf einen anderen OpenVMS-Knoten versetzt.

- Es erfolgt ein erneuter Start nach einem WS-Manager-Fehler.

Standardmäßig konfiguriert die Prozedur StartCommand die TCP/IP-Adresse der Überbrückungsgruppe auf dem Knoten, um den WS-Manager auszuführen, und startet anschließend den WS-Manager mit dem Befehl **strmqm -m** *WS-Manager-Name*. Abhängig von den Systemvoraussetzungen kann die Befehlsprozedur folgendermaßen geändert werden:
- Ändern des Befehls **strmqm**
- Hinzufügen von Befehlen zum Starten zusätzlicher MQSeries-Prozesse, z. B. des Empfangsprogramms
- Hinzufügen von Befehlen zum Starten von Anwendungsprozessen

Die Prozedur StartCommand muss mit dem Status 1 enden, damit der WS-Manager nach seinem Start überwacht werden kann.

Die Prozedur EndCommand wird in folgenden Situationen zum Beenden des WS-Managers verwendet:
- Es erfolgt eine explizite Angabe mit der Option -e des Befehls **failover**.
- Der WS-Manager wird mit der Option -f des Befehls **failover** auf einen anderen OpenVMS-Knoten versetzt.

Standardmäßig versucht die Prozedur EndCommand den WS-Manager mit dem Befehl **endmqm -i** *WS-Manager-Name* zu beenden. Wenn der WS-Manager nicht innerhalb des in der Konfigurationsdatei angegebenen Zeitlimits beendet wird, versucht die Prozedur, den WS-Manager mit dem Befehl **endmqm -p** *WS-Manager-Name* zu beenden. Wenn der WS-Manager nach einem erneuten Ablauf des Zeitlimits immer noch aktiv ist, wird er beendet, indem der Ausführungs-Controller gelöscht wird. Nachdem der WS-Manager beendet wurde, wird die TCP/IP-Adresse der Überbrückungsgruppe dekonfiguriert. Wenn der WS-Manager erfolgreich mit dem Befehl **endmqm** beendet wurde, wird der Status SS$_NORMAL zurückgegeben. Wurde der WS-Manager durch Löschen des Ausführungs-Controllers beendet, wird der Status SS$_ABORT zurückgegeben. Wenn der WS-Manager auch nach Ablauf eines dritten Zeitlimits noch nicht beendet ist, wird der Status SS$_TIMEOUT zurückgegeben. Anhand dieser Statusrückmeldungen erkennt der Beobachtungsmonitor das Ergebnis der Prozedur EndCommand und legt den Status der Überbrückungsgruppe entsprechend fest. Abhängig von den Systemvoraussetzungen kann die Befehlsprozedur folgendermaßen geändert werden:
- Hinzufügen von Befehlen zum Beenden zusätzlicher MQSeries-Prozesse, z. B. des Empfangsprogramms
- Hinzufügen von Befehlen zum Beenden von Anwendungsprozessen

Die Prozedur TidyCommand wird zum Aufräumen eines OpenVMS-Knotens verwendet, wenn der WS-Manager mit einem Fehler beendet wurde, der OpenVMS-Knoten jedoch unbeschadet weiter aktiv ist.

Standardmäßig dekonfiguriert die Prozedur TidyCommand die TCP/IP-Adresse der Überbrückungsgruppe. Abhängig von den Systemvoraussetzungen kann die Befehlsprozedur folgendermaßen geändert werden:
- Hinzufügen von Befehlen zum Beenden aller noch aktiven MQSeries-Prozesse, z. B. des Empfangsprogramms
- Hinzufügen von Befehlen zum Beenden weiterhin aktiver Anwendungsprozesse

Die Schablonendateien sind standardmäßig so definiert, dass 'Digital TCP/IP Services für OpenVMS'-Befehle zum Konfigurieren und Dekonfigurieren der TCP/IP-Adresse verwendet werden. Wenn Sie 'TCPware für OpenVMS' oder 'MultiNet für OpenVMS' verwenden, fügen Sie Kommentarzeichen vor den 'Digital TCP/IP Services für OpenVMS'-Befehlen ein (inaktivieren), und entfernen Sie die Kommentarzeichen vor den 'TCPware für OpenVMS'- bzw. 'MultiNet für OpenVMS'-Befehlen (aktivieren).

Verwaltung von Überbrückungsgruppen

Überbrückungsgruppen müssen über das Konto SYSTEM oder über ein MQSeries-Verwaltungskonto verwaltet werden. Die Verwaltung der Überbrückungsgruppen erfolgt mit Hilfe der DCL-Befehle **runmqfm** und **failover**. Der Befehl **runmqfm** wird zum Starten der Überbrückungsmonitore und der Befehl **failover** zum Ausführen aller anderen Verwaltungs-Tasks verwendet. Diese Befehle werden unter „runmqfm (Überbrückungsmonitor starten)" auf Seite 301 und unter „failover (Überbrückungsgruppe verwalten)" auf Seite 286 beschrieben.

Überbrückungsmonitore starten

Überbrückungsmonitore werden gestartet, indem auf dem OpenVMS-Knoten, auf dem der jeweilige Überbrückungsmonitor gestartet werden soll, der Befehl **runmqfm** ausgeführt wird. Verwenden Sie zum Beispiel zum Starten des Überbrückungsmonitors für den WS-Manager `TESTQM` den folgenden Befehl:

```
$ runmqfm -m TESTQM
```

Dieser Befehl erstellt einen Hintergrundprozess mit einem Namen, der auf dem des WS-Managers basiert und der mit `_FM` endet. In diesem Beispiel hat der Prozess den Namen `TESTQM_FM`. Dieser Prozess wird in einer aktiven Anzeige des Dienstprogramms MONMQ aufgelistet.

Wenn eine Protokolldatei erforderlich ist, kann sie angegeben werden, indem die Ausgabe des Befehls **runmqfm** umgeleitet wird; in der Protokolldatei können durch Angabe der Option -d zusätzliche Debug-Informationen angezeigt werden. Beispiel:

```
$ runmqm -m TESTQM -d > sys$manager:fm.log
```

Beachten Sie, dass mit dem Befehl **runmqfm** nur Überbrückungsmonitore gestartet werden; es wird nicht der WS-Manager gestartet.

WS-Manager in einer Überbrückungsgruppe starten

Um einen WS-Manager in einer Überbrückungsgruppe zu starten, muss mindestens ein Überbrückungsmonitor aktiv sein, und auf dem Knoten, auf dem der WS-Manager gestartet werden soll, muss ebenfalls ein Überbrückungsmonitor aktiv sein. Ein WS-Manager wird mit der Option -s des Befehls **failover** gestartet. Der Befehl kann auf jedem OpenVMS-Knoten in der Überbrückungsgruppe ausgeführt werden. Verwenden Sie zum Beispiel zum Starten des WS-Managers `TESTQM` auf dem Knoten `BATMAN` folgenden Befehl:

```
$ failover -m TESTQM -n BATMAN -s
```

Wenn der WS-Manager auf dem OpenVMS-Knoten mit der höchsten Priorität gestartet werden soll, verwenden Sie den Befehl ohne die Option -n. Beispiel:

```
$ failover -m TESTQM -s
```

Beachten Sie, dass jeder Versuch, einen WS-Manager mit dem Befehl **strmqm** zu starten, fehlschlägt, nachdem ein Überbrückungsmonitor für einen WS-Manager (egal auf welchem Knoten) gestartet wurde. Sobald jedoch alle Überbrückungsmonitore für einen WS-Manager gestoppt wurden, kann der Befehl **strmqm** wieder normal verwendet werden.

WS-Manager in einer Überbrückungsgruppe beenden

Um einen WS-Manager in einer Überbrückungsgruppe zu beenden, muss auf dem Knoten, auf dem der WS-Manager ausgeführt wird, ein Überbrückungsmonitor aktiv sein. Ein WS-Manager wird mit der Option -e des Befehls **failover** beendet. Der Befehl kann auf jedem OpenVMS-Knoten in der Überbrückungsgruppe ausgeführt werden. Verwenden Sie zum Beispiel zum Beenden des WS-Managers TESTQM den folgenden Befehl:

```
$ failover -m TESTQM -e
```

Beachten Sie, dass jeder Versuch, einen WS-Manager mit dem Befehl **endmqm** zu beenden, fehlschlägt, nachdem ein Überbrückungsmonitor für einen WS-Manager (egal auf welchem Knoten) gestartet wurde. Sobald jedoch alle Überbrückungsmonitore für einen WS-Manager gestoppt wurden, kann der Befehl **endmqm** wieder normal verwendet werden.

WS-Manager innerhalb einer Überbrückungsgruppe versetzen

Einen WS-Manager innerhalb einer Überbrückungsgruppe zu versetzen, bedeutet, dass der WS-Manager auf dem Knoten, auf dem er gerade ausgeführt wird, gestoppt und anschließend auf einem anderen Knoten innerhalb der Überbrückungsgruppe erneut gestartet wird. Um einen WS-Manager innerhalb einer Überbrückungsgruppe zu versetzen, muss auf dem Knoten, auf dem der WS-Manager gerade ausgeführt wird, ein Überbrückungsmonitor aktiv sein und auf dem Knoten, auf den der WS-Manager versetzt werden soll, muss ebenfalls ein Überbrückungsmonitor aktiv sein.

Ein WS-Manager wird mit der Option -f des Befehls **failover** versetzt. Der Befehl kann auf jedem OpenVMS-Knoten in der Überbrückungsgruppe ausgeführt werden. Verwenden Sie zum Beispiel zum Versetzten des WS-Managers TESTQM auf den Knoten ROBIN den folgenden Befehl:

```
$ failover -m TESTQM -n ROBIN -f
```

Wenn der WS-Manager auf den OpenVMS-Knoten mit der höchsten Priorität versetzt werden soll, verwenden Sie den Befehl ohne die Option -n. Beispiel:

```
$ failover -m TESTQM -f
```

Status einer Überbrückungsgruppe anzeigen

Es gibt drei Statusarten, die den Gesamtstatus der Überbrückungsgruppe beschreiben:

- Status des WS-Managers in der Überbrückungsgruppe
- Status des WS-Manager-Knotens in der Überbrückungsgruppe (einer pro Knoten)
- Status des Überbrückungsmonitors auf dem Knoten (einer pro Knoten)

Die möglichen Statuswerte werden in den folgenden drei Tabellen beschrieben:

Tabelle 11. Status des WS-Managers in der Überbrückungsgruppe

Status	Beschreibung
STOPPED	Der WS-Manager wurde niemals in der Überbrückungsgruppe gestartet oder wurde ordnungsgemäß abgeschlossen.
STARTED	Der WS-Manager wurde in der Überbrückungsgruppe gestartet. Die Überbrückungsgruppe wird versuchen, den WS-Manager nach einem Fehler erneut zu starten.

Tabelle 12. Status des WS-Manager-Knotens in der Überbrückungsgruppe

Status	Beschreibung
AVAILABLE	Der Knoten ist frei, d. h., auf ihm kann ein WS-Manager gestartet werden, wenn auf einem anderen Knoten ein Fehler auftritt.
RUNNING	Der WS-Manager ist auf diesem Knoten aktiv.
EXCLUDED	Der WS-Manager wurde auf diesem Knoten auf eine nicht ordnungsgemäße Art gestoppt, der Knoten selbst läuft jedoch nach wie vor fehlerfrei. Wenn ein WS-Manager auf einem anderen Knoten fehlschlägt, wird er auf diesem Knoten nicht erneut gestartet.

Tabelle 13. Status des Überbrückungsmonitors auf dem Knoten

Status	Beschreibung
STARTED	Auf diesem Knoten ist ein Überbrückungsmonitor aktiv, es ist jedoch nicht der Beobachtungsmonitor.
WATCHING	Auf diesem Knoten ist ein Überbrückungsmonitor aktiv, dabei handelt es sich um den Beobachtungsmonitor.
STOPPED	Auf diesem Knoten ist kein Überbrückungsmonitor aktiv.

Der Status der Überbrückungsgruppe wird mit der Option -q des Befehls **failover** angezeigt. Es muss mindestens ein Überbrückungsmonitor aktiv sein, und der Befehl kann auf jedem Knoten in der Überbrückungsgruppe ausgeführt werden. Geben Sie zum Beispiel folgenden Befehl ein, um den Status der Überbrückungsgruppe des WS-Managers TESTQM anzuzeigen:

```
$ failover -m TESTQM -q
```

Es folgt eine Beispielausgabe dieses Befehls:

```
83H8439, 5697-270 (C) Copyright IBM Corp. 1996.  ALLE RECHTE VORBEHALTEN.

OpenVMS Cluster-Überbrückungsgruppe - Konfiguration und Status.

Name des WS-Managers                              : TESTQM
Sortierungsnummer                                 : 11
TCP/IP-Adresse                                    : 10.20.30.40
Empfangsprogramm Port-Nummer                      : 1414
Zeitlimit für das Beenden des WS-Managers         : 30
Status des WS-Managers in der Überbrückungsgruppe  : STARTED

OpenVMS-Knoten - Konfiguration und Status

Knotenname                        : BATMAN
Priorität                         : 2
TCP/IP-Schnittstelle              : we0
Status des WS-Managers            : RUNNING
Status des Überbrückungsmonitors : WATCHING

Knotenname                        : ROBIN
Priorität                         : 1
TCP/IP-Schnittstelle              : we0
Status des WS-Managers            : EXCLUDED
Status des Überbrückungsmonitors : STARTED
```

Über DCL-Symbole den Status einer Überbrückungsgruppe anzeigen

In bestimmten Fällen kann es erforderlich sein, DCL-Befehlsprozeduren zum Steuern von Überbrückungsgruppen zu schreiben. Mit der Option -l des Befehls **failover** kann über drei lokale DCL-Symbole der Status der Überbrückungsgruppe angezeigt werden. Über diese Symbole können dann bedingte Aktionen auf Basis des Status des WS-Managers ausgeführt werden. Es muss mindestens ein Überbrückungsmonitor aktiv sein, und der Befehl kann auf jedem Knoten in der Überbrückungsgruppe ausgeführt werden. Die Symbole für die Statusangaben sind in Tabelle 14 auf Seite 241 aufgeführt.

Tabelle 14. DCL-Symbole und Beschreibung

Name des DCL-Symbols	Beschreibung
MQS$QMGR_NODE	Enthält den Namen des OpenVMS-Knotens, auf dem der WS-Manager aktiv ist, oder eine Nullzeichenfolge, wenn kein WS-Manager aktiv ist.
MQS$AVAILABLE_NODES	Enthält die Liste der OpenVMS-Knoten, die für die Ausführung des WS-Managers zur Verfügung stehen. Das sind alle Knoten, die sich im WS-Manager-Status AVAILABLE befinden und auf denen ein Überbrückungsmonitor aktiv ist.
MQS$MONITOR_NODES	Enthält die Liste der OpenVMS-Knoten, auf denen ein Überbrückungsmonitor aktiv ist.

Geben Sie zum Beispiel folgenden Befehl ein, damit die Symbole den Status der Überbrückungsgruppe für den WS-Managers TESTQM angeben:

```
$ failover -m TESTQM -l
```

Im Folgenden werden Beispielergebnisse für die Angaben der Symbole gezeigt:

```
MQS$AVAILABLE_NODES = ""
MQS$MONITOR_NODES = "BATMAN,ROBIN"
MQS$QMGR_NODE = "BATMAN"
```

Einen Überbrückungsmonitor anhalten

Der Überbrückungsmonitor auf einem OpenVMS-Knoten kann mit der Option -h des Befehls **failover** angehalten werden. Der Befehl kann auf jedem Knoten in der Überbrückungsgruppe ausgeführt werden. Verwenden Sie zum Beispiel folgenden Befehl, um den Überbrückungsmonitor für WS-Manager TESTQM auf dem Knoten BATMAN anzuhalten:

```
$ failover -m TESTQM -n BATMAN -h
```

Wenn es sich bei dem angehaltenen Überbrückungsmonitor um den Beobachtungsmonitor handelt, übernimmt ein anderer Überbrückungsmonitor diese Aufgabe, sofern noch einer vorhanden ist. Wenn es sich bei dem angehaltenen Überbrückungsmonitor um den letzten Überbrückungsmonitor für die Überbrückungsgruppe handelt, ist die Überbrückungsgruppe nicht weiter aktiv. In diesem Fall kann der WS-Manager wieder mit den Befehlen **strmqm** und **endmqm** gestartet bez. beendet werden. Mit der Option -h des Befehls **failover** wird ein WS-Manager niemals beendet. Wenn der WS-Manager auf dem OpenVMS-Knoten aktiv ist, auf dem der Überbrückungsmonitor angehalten wird, bleibt der WS-Manager weiter aktiv.

Befehle während einer Aktualisierung ausführen

failover-Befehle mit den Optionen -s, -e, -f und -c beinhalten Aktualisierungen. Während der Ausführung dieser Befehle wird vom Beobachtungsmonitor eine Markierung gesetzt, dass eine Aktualisierung im Gange ist. Solange diese Markierung gesetzt ist, schlagen alle weiteren Befehle zum Aktualisieren und Anhalten von Überbrückungsmonitoren fehl, weil simultane Aktualisierungen nicht zulässig sind. Befehle, die keine Aktualisierung beinhalten, z. B. mit den Optionen -q und -l, werden während einer Aktualisierung weiter ausgeführt.

In seltenen Fällen kann es vorkommen, dass die Markierung nach einer fehlgeschlagenen Aktualisierung gesetzt bleibt. Mit der Option -u des Befehls **failover** kann die Aktualisierungsmarkierung entfernt werden. Dieser Befehl sollte mit Vorsicht verwendet werden. Verwenden Sie zum Beispiel folgenden Befehl, um die Aktualisierungsmarkierung für WS-Manager `TESTQM` zu entfernen:

```
$ failover -m TESTQM -u
```

Status einer Überbrückungsgruppe ändern

In bestimmten Situationen kann es nötig sein, den Status einer Überbrückungsgruppe zu ändern. Dazu kann die Option -c des Befehls **failover** verwendet werden. Dies ist in den meisten Fällen dann nötig, wenn ein WS-Manager auf einem Knoten nach einem Fehler den Status EXCLUDED aufweist und Sie den Status wieder auf AVAILABLE zurücksetzen wollen, nachdem Sie den Knoten bereinigt haben. Verwenden Sie zum Beispiel folgenden Befehl, um den Status des WS-Managers `TESTQM` auf dem Knoten `BATMAN` in AVAILABLE zu ändern:

```
$ failover -m TESTQM -n BATMAN -c -qmgr available
```

Sie haben auch die Möglichkeit, einen Knoten vorübergehend als möglichen Knoten für die Ausführung des WS-Manager auszuschließen, indem Sie den Status des WS-Manager-Knotens von AVAILABLE in EXCLUDED ändern. Verwenden Sie zum Beispiel folgenden Befehl, um den Status für den WS-Manager `TESTQM` auf dem Knoten `BATMAN` in EXCLUDED zu ändern:

```
$ failover -m TESTQM - n BATMAN -c -qmgr excluded
```

Es ist darüber hinaus möglich, alle anderen Status zu ändern, allerdings wird eine Änderung nur wirksam, wenn sie mit dem aktiven System konsistent ist. Wenn zum Beispiel auf einem Knoten ein Überbrückungsmonitor aktiv ist und Sie versuchen, den Status des Monitors in STOPPED zu ändern, bleibt diese Änderung unwirksam. Abgesehen von der Änderung des Status des WS-Manager-Knotens von EXCLUDED in AVAILABLE (und umgekehrt) wird der Befehl zum Ändern des Status normalerweise nicht benötigt, weil der Beobachtungsmonitor alle 30 Sekunden eine Integritätsprüfung ausführt und die Statusangaben ändert, wenn er eine Abweichung vom aktiven Systemstatus feststellt.

Sicherheit für ICC-Zuordnungen einstellen

Die Überbrückungsgruppenmonitore und Client-Programme verwenden OpenVMS Intra Cluster Communication (ICC)-Aufrufe, um Nachrichten zu übergeben. Um zu verhindern, dass unbefugte Benutzer Nachrichten an die Überbrückungsgruppenmonitore senden, müssen die ICC-Zuordnungen in der Befehlsprozedur **SYS$STARTUP:ICC$SYSTARTUP.COM** konfiguriert werden.

Jede Überbrückungsgruppe verwendet zwei Zuordnungsnamen: einen mit dem Namen des WS-Managers, der für die Kommunikation mit dem Beobachtungsmonitor verwendet wird, und einen anderen mit dem Namen des WS-Managers, der für die Kommunikation mit den einzelnen Überbrückungsmonitoren verwendet wird; an den zweiten Namen wird `_MQ_FM` angehängt.

Abb. 22 auf Seite 243 zeigt ein Beispiel für die Einträge, die in der Datei **ICC$SYSTARTUP.COM** für jeden Knoten in einer Überbrückungsgruppe erforderlich sind. Die Überbrückungsgruppe enthält zwei Knoten mit den Namen BATMAN und ROBIN, und der WS-Manager heißt `TESTQM`.

```
$! ---------------------  List Nodes with Special Actions -------------------
$!
$ nodeactions = "/BATMAN/ROBIN/"
$ if f$locate("/"+nodename+"/",nodeactions) .eq. f$length(nodeactions) -
then goto exit ! No action for this node
$ goto 'nodename' ! Go to action code for this node
$!
$! --------------------  Major Nodes ----------------------
$BATMAN:
$ROBIN:
$!
$! Place in here calls to @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT and
$! @SYS$MANAGER:ICC$ADD_REGISTRY_TABLE that apply to FAilover odes in the
$! cluster
$!
$!
$  @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT ICC$::"TESTQM" -
"/owner=MQM/acl=((id=MQM,access=open+access),(id=*,access=none))"
$!
$  @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT 'nodename'::"TESTQM" -
"/owner=MQM/acl=((id=MQM,access=open+access),(id=*,access=none))"
$!
$  @SYS$MANAGER:ICC$CREATE_SECURITY_OBJECT 'nodename'::"TESTQM_MQ_FM" -
"/owner=MQM/acl=((id=MQM,access=open+access),(id=*,access=none))"
$!
$  set security/class=logical_name_table icc$registry_table -
        /acl=(id=MQM,access=read+write)
$!
$  GOTO EXIT
$!
```

Abbildung 22. Erforderlicher Bespieleintrag für ICC$SYSTARTUP.COM

Beachten Sie, dass ICC-Zuordnungsnamen maximal 31 Zeichen lang sein dürfen, so dass für Namen von MQSeries-WS-Managern, die in einer Überbrückungsgruppe verwendet werden, eine maximale Länge von 25 Zeichen unterstützt wird. Weitere Informationen zum Einstellen der Sicherheit von ICC-Zuordnungen finden Sie im Handbuch *OpenVMS System Manager's Manual*.

Fehlerbehebung bei Überbrückungsgruppen

Beim Ausführen der Prozeduren **start_qm.com**, **end_qm.com** und **tidy_qm.com** wird im Protokollverzeichnis (LogDirectory), das in der Konfigurationsdatei `failover.ini` angegeben ist, eine Protokolldatei erstellt. Die Namen der Protokolldateien lauten *WSMgrName_Prozedurname*`.log`. Wenn zum Beispiel für den WS-Manager `TESTQM` die Befehlsprozedur **start_qm.com** ausgeführt wird, wird eine Protokolldatei mit dem Namen `testqm_start_qm.log` erstellt.

Der Überbrückungsmonitor erstellt standardmäßig keine Protokolldatei, es kann jedoch über den Umleitungsparameter des Befehls **runmqfm** eine Protokolldatei angegeben werden. In die Datei werden zusätzliche Debug-Informationen geschrieben, wenn mit dem Befehl **runmqfm** die Option -d angegeben wird.

Überprüfen Sie, ob FDC-Dateien im Verzeichnis `MQS_ROOT:[MQM.ERRORS]` generiert wurden.

MultiNet für OpenVMS mit Überbrückungsgruppen verwenden

Um Multinet für OpenVMS mit Überbrückungsgruppen zu verwenden, muss der Cluster-Aliasnamenservice aktiviert werden. Der Cluster-Aliasnamenservice wird mit folgendem Befehl aktiviert:

```
$ MULTINET CONFIGURE/SERVERS
SERVER-CONFIG> ENABLE CLUSTERALIAS
SERVER-CONFIG> EXIT
```

Die Schablonenbefehlsdateien gehen davon aus, dass nur eine Cluster-Aliasnamenadresse vorhanden ist und diese von der Überbrückungsgruppe verwendet wird. Wenn jedoch mehrere Cluster-Aliasnamenadressen vorhanden sind, müssen die Befehlsprozeduren geändert werden, damit die anderen Adressen in dem logischen Namen MULTINET_CLUSTER_IP_ALIASES erhalten bleiben.

Beispiel für die Verwendung von Überbrückungsgruppen

Im folgenden Beispiel werden in einem OpenVMS-Cluster in einer Überbrückungsgruppe für den WS-Manager `TESTQM` zwei Knoten, BATMAN und ROBIN, konfiguriert. Die TCP/IP-Adresse der Überbrückungsgruppe lautet 10.20.30.40, und die Port-Nummer des TCP/IP-Empfangsprogramms ist 1414. Der Knoten BATMAN wird als primärer Knoten und ROBIN als sekundärer Knoten definiert. Der WS-Manager wird zunächst auf dem Knoten BATMAN gestartet und nach einem Fehler des WS-Managers auf dem Knoten ROBIN erneut gestartet. Wenn der WS-Manager auf dem Knoten ROBIN aktiv ist, wird der WS-Manager nicht wieder auf den Knoten BATMAN zurückversetzt, wenn ROBIN erneut gebootet wird. Wenn auf dem Knoten mit dem WS-Manager ein Systemabschluss durchgeführt wird, wird der WS-Manager beendet und der Überbrückungsmonitor angehalten. Ist der Knoten nicht aktiv, wird für den WS-Manager kein Abschluss durchgeführt und nur der Überbrückungsmonitor angehalten.

Schablonendatei failover.template anpassen

Die Datei failover.template wird wie folgt geändert und nach `mqs_root:[mqm.qmgrs.testqm]failover.ini` kopiert.

```
# FAILOVER.TEMPLATE
# Template for creating a FAILOVER.INI configuration file
# All lines beginning with a '#' are treated as comments
#
# OpenVMS Cluster Failover Set Configuration information
# ------------------------------------------------------
#
# The TCP/IP address used by the OpenVMS Cluster Failover Set
#
IpAddress=10.20.30.40
#
# The TCP/IP port number used by the MQSeries Queue Manager
#
PortNumber=1414
#
# The timeout used by the EndCommand command procedure
#
TimeOut=30
#
# The command procedure used to start the Queue Manager
#
StartCommand=@sys$manager:start_qm
#
# The command procedure used to end the Queue Manager
#
EndCommand=@sys$manager:end_qm
#
# The command procedure used to tidy up on a node after a
# Queue Manager failure but the OpenVMS node did not fail
#
TidyCommand=@sys$manager:tidy_qm
#
# The directory in which the log files for the start, end and
# tidy commands are written
#
LogDirectory=mqs_root:[mqm.errors]
#
# The number of nodes in the OpenVMS Cluster Failover Set. The
# number of nodes defined below must agree with this number
#
NodeCount=2
#
# The Name of the OpenVMS node
#
NodeName=BATMAN
#
# The TCP/IP interface name for the node
#Interface=we0
#
# The priority of the node
#
Priority=1
#
# The Name of the OpenVMS node
#
NodeName=ROBIN
#
# The TCP/IP interface name for the node
#
Interface=we0
#
# The priority of the node
#
Priority=2
```

Abbildung 23. Schablone failover.template zum Erstellen einer Konfigurationsdatei FAILOVER.INI

Änderung von Befehlsprozeduren für Überbrückungsgruppen

Die Befehlsprozeduren werden aus der Schablonendatei kopiert. Die einzigen Änderungen bestehen darin, dass die Kommentarzeichen für den Start des Empfangsprogramms in **start_qm.com** und das Ende des Empfangsprogramms in **end_qm.com** entfernt werden (aktivieren). Wenn Anwendungen gestoppt oder gestartet werden sollen, können den Befehlsprozeduren die entsprechenden Befehle hinzugefügt werden.

Beispiel für Befehlsprozedur zum Starten der Überbrückungsgruppe (start_failover_set.com)

Mit der Befehlsprozedur **start_failover_set.com** werden der Überbrückungsmonitor auf jedem Knoten und unter bestimmten Bedingungen der WS-Manager gestartet. Die Prozedur wird beim Systemstart aufgerufen, nachdem die Befehlsprozedur **MQS_STARTUP.COM** ausgeführt wurde. An die Prozedur werden zwei Parameter übergeben: der Name des WS-Managers und der Name des primären Knotens. In diesem Fall lautet der Aufruf wie folgt:

```
$@start_failover_set testqm batman
```

Die Befehlsprozedur **start_failover_set.com** startet den Überbrückungsmonitor und ermittelt dann mit Hilfe der Option -l des Befehls **failover** den Status der Überbrückungsgruppe. Beachten Sie, dass der Überbrückungsmonitor möglicherweise noch nicht vollständig gestartet wurde, wenn der Befehl **failover** ausgeführt wird, so dass der Befehl im Abstand von einer Sekunde bis zu drei Mal wiederholt wird. Wenn es sich bei dem Knoten um den primären Knoten handelt und der WS-Manager nicht gestartet wurde, wird er mit der Option -s des Befehls **failover** gestartet.

```
$on error then exit
$@sy$manager:mqs_symbols
$!
$! start_failover_set.com
$! ----------------------
$! Command procedure to start a Failover Set Queue Manager during startup
$!
$! p1 = Queue Manager name
$! p2 = Primary Node name
$!
$! Check that the Queue Manager has been specified
$!
$if p1 .eqs ""
$then
$ Write sys$output "Queue Manager name omitted"
$ exit
$else
$ qmgr_name = p1
$endif
$!
$! Check that the primary node name has been specified
$!
$if p2 .eqs ""
$then
$ Write sys$output "Node name omitted"
$ exit
$else
$ primary_node = p2
$endif
$!
$! Get the node name of this node
$!
$this_node=f$getsyi("nodename")
$!
$! Start the Failover Monitor on this node
$!
$runmqfm -m 'qmgr_name'
$!
$! Check that the Failover Monitor has fully started
$! Wait up to 3 seconds
$!
$count = 0
$check_start:
$on error then continue
$!
$! Set the MQS$* symbols to the state of Failover Set
$! Wait up to 3 seconds
$!
```

Abbildung 24. Befehlsprozedur start_failover_set (Teile- 1 von 2)

Beispiel für Überbrückungsgruppe

```
$failover -m 'qmgr_name' -l
$!
$! If an error is returned wait a second and try again
$!
$if ( ($status/8) .and %xfff ) .ne. 0 then goto wait
$!
$! If this node is not listed as running a monitor wait a second and try again
$!
$if f$locate( this_node, mqs$monitor_nodes ) .ne. f$length( mqs$monitor_nodes )
$then
$ goto start_qm
$endif
$wait:
$on error then exit
$count = count + 1
$!
$! If we have waited 3 seconds display an error and exit
$!
$if count .ge. 3
$then
$ write sys$output "Failover Monitor not started"
$ exit
$else
$ wait 00:00:01
$ goto check_start
$endif
$start_qm:
$!
$! Only start the Queue Manager on the primary node
$!
$if this_node .nes. primary_node
$then
$ write sys$output "Queue Manager not started on Secondary node"
$ exit
$endif
$!
$! Start the Queue Manager on the primary node if it is not already running.
$!
$if mqs$qmgr_node .eqs. ""
$then
$ failover -m 'qmgr_name' -n 'this_node' -s
$else
write sys$output "Queue Manager already started"
$endif
$exit
```

Abbildung 24. Befehlsprozedur start_failover_set (Teile- 2 von 2)

Beispiel für Befehlsprozedur zum Beenden der Überbrückungsgruppe (end_failover_set.com)

Mit der Befehlsprozedur **end_failover_set.com** werden der WS-Manager unter bestimmten Bedingungen und dann der Überbrückungsmonitor auf jedem Knoten beendet. Die Prozedur wird beim standortspezifischen Systemabschluss aufgerufen, bevor die Befehlsprozedur **MQS_SHUTDOWN.COM** ausgeführt wird. An die Prozedur wird nur ein Parameter übergeben: der Name des WS-Managers. In diesem Fall lautet der Aufruf wie folgt:

```
$@start_failover_set testqm
```

Die Befehlsprozedur **end_failover_set.com** ruft mit Hilfe der Option -l des Befehls **failover** den Status der Überbrückungsgruppe ab. Wenn der WS-Manager auf diesem Knoten aktiv ist, wird er beendet. Anschließend wird der Überbrückungsmonitor angehalten.

```
on error then exit
$@sys$manager:mqs_symbols
$!
$! end_failover_set.com
$! -------------------------
$! Command procedure to end a Failover Set Queue Manager during shutdown
$!
$! p1 = Queue Manager name
$!
$! Check that the Queue Manager has been specified
$!
$if p1 .eqs ""
$then
$ Write sys$output "Queue Manager name omitted"
$ exit
$else
$ qmgr_name = p1
$endif
$!
$! Get the node name of this node
$!
$this_node=f$getsyi("nodename")
$!
$! Set the MQS$* symbols to the state of the Failover Set
$!
$failover -m 'qmgr_name' -l
$!
$! If an error then exit
$!
$if ( ($status/8) .and %xfff ) .ne. 0
$then
$ write sys$output "Error querying Failover Set"
$ exit
$endif
$!
$! If the Queue Manager is not running on this node then exit
$!
$ if mqs$qmgr_node .nes. this_node
$then
$ write sys$output "Queue Manager not running on this node"
$ goto halt_fm
$endif
$!
$! End the Queue Manager
$!
$failover -m gjtest -e
$halt_fm:
$!
$! Halt the Failover Monitor
$!
$failover -m gjtest -n 'this_node' -h
```

Abbildung 25. Befehlsprozedur end_failover_set

Teil 2. Referenz

Kapitel 17. MQSeries-Steuerbefehle

Dieses Kapitel enthält Referenzinformationen zu den in MQSeries for Compaq OpenVMS verwendeten Steuerbefehlen. Alle Befehle in diesem Kapitel können in der DCL-Eingabeaufforderung von OpenVMS eingegeben werden.

Befehlsnamen und -optionen können in Großbuchstaben, in Kleinbuchstaben oder in einer Kombination aus Groß- und Kleinbuchstaben eingegeben werden. Bei Parametern von Steuerbefehlen (z. B. Namen von Warteschlangen) muss jedoch gegebenenfalls die Groß-/Kleinschreibung beachtet werden. Weitere Informationen finden Sie unter „Groß-/Kleinschreibung in Steuerbefehlen" auf Seite 23.

Bevor Steuerbefehle verwendet werden können, muss nach dem letzten Neustart ein Mal der Befehl 'mqs_startup' ausgeführt werden.

Regeln für die Benennung von MQSeries-Objekten

In der Regel dürfen Namen von MQSeries-Objekten bis zu 48 Zeichen lang sein. Dies gilt für alle folgenden Objekte:

- Warteschlangenmanager (Wenn der WS-Manager von einer Überbrückungsgruppe für OpenVMS-Cluster unterstützt wird, beträgt die maximale Länge jedoch 25 Zeichen.)
- Warteschlangen
- Prozessdefinitionen
- Namenslisten
- Cluster

Die maximale Länge von Kanalnamen beträgt 20 Zeichen.

Die folgenden Zeichen können in allen MQSeries-Namen verwendet werden:

- Großbuchstaben A–Z
- Kleinbuchstaben a–z
- Numerische Zeichen 0–9
- Punkt (.)
- Unterstreichung (_)
- Schrägstrich (/) (siehe Anmerkung 1)
- Prozentzeichen (%) (siehe Anmerkung 1)

Anmerkungen:

1. Schrägstrich und Prozentzeichen sind Sonderzeichen. Sie dürfen diese beiden Sonderzeichen jedoch nicht als erste Zeichen eines Namens verwenden. Wenn in einem Namen eines dieser beiden Sonderzeichen steht, muss der Name immer in doppelten Anführungszeichen stehen.
2. Die Verwendung führender oder eingebetteter Leerzeichen ist nicht zulässig.
3. Landessprachliche Zeichen sind nicht zulässig.
4. Namen können in doppelten Anführungszeichen stehen, dies ist jedoch nur wichtig, wenn der Name Sonderzeichen enthält oder die Groß-/Kleinschreibung erhalten bleiben soll.

Ein Blick auf Objektdateien

Jede MQSeries-Warteschlange, jeder WS-Manager und jedes Prozessobjekt wird durch eine Datei repräsentiert. Da es sich bei diesen Objektnamen nicht notwendigerweise um gültige Dateinamen handelt, konvertiert der WS-Manager den Objektnamen bei Bedarf in einen gültigen Dateinamen. Informationen hierzu finden Sie unter „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21.

Informationen, wie Sie den eigentlichen Dateinamen eines Objekts anzeigen, finden Sie unter „dspmqfls (MQSeries-Dateien anzeigen)" auf Seite 273.

Syntaxdiagramme lesen

Dieses Kapitel enthält Syntaxdiagramme.

Jedes Syntaxdiagramm beginnt mit einem doppelten Rechtspfeil und endet mit einem Paar aus Rechtspfeil und Linkspfeil. Bei Zeilen, die mit einem einfachen Rechtspfeil beginnen, handelt es sich um Fortsetzungszeilen. Syntaxdiagramme werden von links nach rechts und von oben nach unten in Richtung der Pfeile gelesen.

Weitere Konventionen für Syntaxdiagramme:

Tabelle 15. Syntaxdiagramme lesen

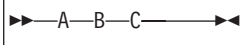
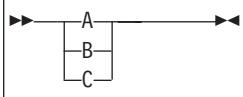
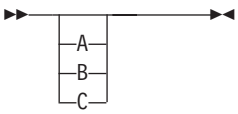
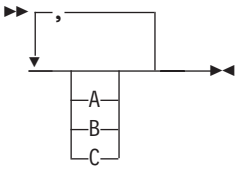
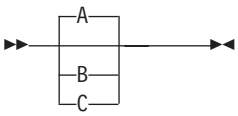
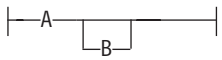
Konvention	Erläuterung
	Sie müssen die Werte `A`, `B` und `C` angeben. Erforderliche Werte werden in der Hauptzeile eines Syntaxdiagramms angezeigt.
	Sie können den Wert `A` angeben. Optionale Werte werden unterhalb der Hauptzeile eines Syntaxdiagramms angezeigt.
	Die Werte `A`, `B` und `C` sind alternativ. Einer der Werte muss angegeben werden.

Tabelle 15. Syntaxdiagramme lesen (Forts.)

Konvention	Erläuterung
	Die Werte `A`, `B` und `C` sind alternativ. Einer der Werte kann angegeben werden.
	Sie können einen der Werte `A`, `B` oder `C` angeben. Erforderliche Trennzeichen bei der Angabe mehrerer Werte oder bei Wiederholungen (in diesem Beispiel das Komma (,)) werden auf der Pfeillinie angezeigt.
	Die Werte `A`, `B` und `C`) sind alternativ. Einer der Werte muss angegeben werden. Wird keiner der gezeigten Werte angegeben, wird standardmäßig der Wert `A` (der Wert wird oberhalb Hauptzeile angezeigt) verwendet.
 **Name:** 	Das Syntaxfragment `Name` wird getrennt vom Hauptsyntaxdiagramm aufgeführt.

Syntaxhilfe

Sie erhalten einen Hilfetext zur Syntax der in diesem Kapitel beschriebenen Befehle, wenn Sie den Befehl gefolgt von einem Fragezeichen eingeben. MQSeries zeigt dann die für den jeweiligen Befehl erforderliche Syntax an. Die Syntax umfasst alle Parameter und Variablen des Befehls. Unterschiedliche Klammern zeigen an, ob ein Parameter erforderlich ist oder nicht. Beispiel:

```
Befehlsname [-x OptParam ] ( -c | -b ) { -p Principal } Argument
```

Dabei gilt:

Befehlsname	Dies ist der Name des Befehls, für den Hilfe angefordert wurde.
[-x OptParam]	Die eckigen Klammern zeigen an, dass dieser Parameter optional ist.
(-c | -b)	Diese Angabe ist erforderlich. In diesem Fall müssen Sie entweder die Option -c oder -b angeben.
{ -p Principal }	Hier können Sie optional eine Liste mit Variablen angeben. Bei einer Anzeige wie dieser müssen Sie jedoch mindestens eine Variable bei der Eingabe des Befehls angeben.
Argument	Ein Argument, das mit diesem Befehl angegeben werden muss. Die Angabe ist obligatorisch, wenn dies in der Antwort auf die Abfrage angezeigt wird.

Beispiele

1. Ergebnis der Eingabe des Befehls `endmqm ?`

```
endmqm [-z][-c | -i | -p] WS-Manager-Name
```

2. Ergebnis der Eingabe des Befehls `rcdmqimg ?`

```
rcdmqimg [-z] [-m WS-Manager-Name] -t Objekttyp [generischer Objektname]
```

MQSeries-Rückkehrcodes

Die meisten MQSeries-Befehle, z. B. **crtmqm**, geben bei ihrer Beendigung eine Statuszeile aus, in der sie anzeigen, ob sie erfolgreich ausgeführt wurden.

Wenn der Status eines Befehls in einer DCL-Befehlsdatei getestet werden soll, kann es notwendig sein, den von einem MQSeries-Programm zurückgegebenen Statuswert zu interpretieren.

Die MQSeries-Rückkehrcodes sind in einer Nachrichtendatei mit dem Namen SYS$MESSAGE:MQS_MSG.EXE definiert.

Um auf den Nachrichtentext zu einem Rückkehrcode in der Datei zuzugreifen, ***müssen*** Sie den DCL-Befehl SET MESSAGE verwenden. Dieser Befehl lädt die Nachrichtencodes in die Nachrichtentabelle Ihres Prozesses. Beispiel:

```
$ SET MESSAGE SYS$MESSAGE:MQS_MSG.EXE
```

Anschließend können Sie mit Hilfe der lexikalischen Funktion F$MESSAGE den Text eines MQSeries-Rückkehrcodes ausgeben. Beispiel:

```
$ strmqm )(*falscher WS-Manager-Name&%$#
Der Name des Warteschlangenmanagers ist entweder ungültig oder nicht bekannt.
$ WRITE SYS$OUTPUT F$MESSAGE($STATUS)
%MQS-F-CSPRC_Q_MGR_NAM, Falscher WS-Manager-Name
```

Mit Hilfe der folgenden DCL-Gleichung können Sie den OpenVMS-Rückkehrcode in einen Rückkehrcode konvertieren, der in MQSeries für OS/2- oder UNIX-Systeme verwendet wird:

```
$ RC = $STATUS / 8 .AND. %xFFF
```

Beispiel:

```
$ crtmqm &*)*(
Der Name des Warteschlangenmanagers ist entweder ungültig oder nicht bekannt.
$ RC = $STATUS / 8 .AND. %xFFF
$ SHOW SYMBOL RC
  RC = 72   Hex = 00000048  Octal = 00000000110
```

crtmqcvx (Datenkonvertierung)

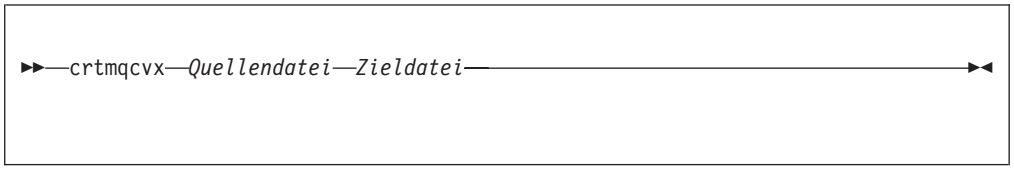
Funktion

Mit dem Befehl **crtmqcvx** erstellen Sie ein Codefragment, das eine Datenkonvertierung von Datentypstrukturen ausführt. Der Befehl generiert eine C-Funktion, die in einem Exit zum Konvertieren von C-Strukturen verwendet werden kann.

Der Befehl liest eine Eingabedatei mit einer oder mehreren Strukturen, die konvertiert werden soll(en). Anschließend wird eine Ausgabedatei mit einem Codefragment oder Fragmenten zum Konvertieren dieser Strukturen erstellt.

Weitere Informationen zu diesem Befehl und zu seiner Verwendung finden Sie im Handbuch *MQSeries Application Programming Guide*.

Syntax



Erforderliche Parameter

Quellendatei
Gibt die Eingabedatei mit den zu konvertierenden C-Strukturen an.

Zieldatei
Gibt die Ausgabedatei mit den Codefragmenten an, die zum Konvertieren der Strukturen generiert wurden.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

Das folgende Beispiel zeigt die Ergebnisse des Datenkonvertierungsbefehls bei einer Anwendung auf eine C-Quellenstruktur. Der ausgegebene Befehl lautet wie folgt:

```
crtmqcvx quelle.tmp ziel.c
```

Die Eingabedatei quelle.tmp hat folgenden Inhalt:

```
/* Dies ist eine C-Teststruktur, die mit dem Dienstprogramm   */
/* crtmqcvx konvertiert werden kann.                          */

struct my_structure
{
    int    code;
    MQLONG value;
};
```

Die von dem Befehl generierte Ausgabedatei ziel.c ist unten abgebildet. Sie können diese Codefragmente in Ihren Anwendungen zum Konvertieren von Datenstrukturen verwenden. Dabei sollten Sie jedoch berücksichtigen, dass das Fragment Makros aus der MQSeries-Header-Datei amqsvmha.h. verwendet.

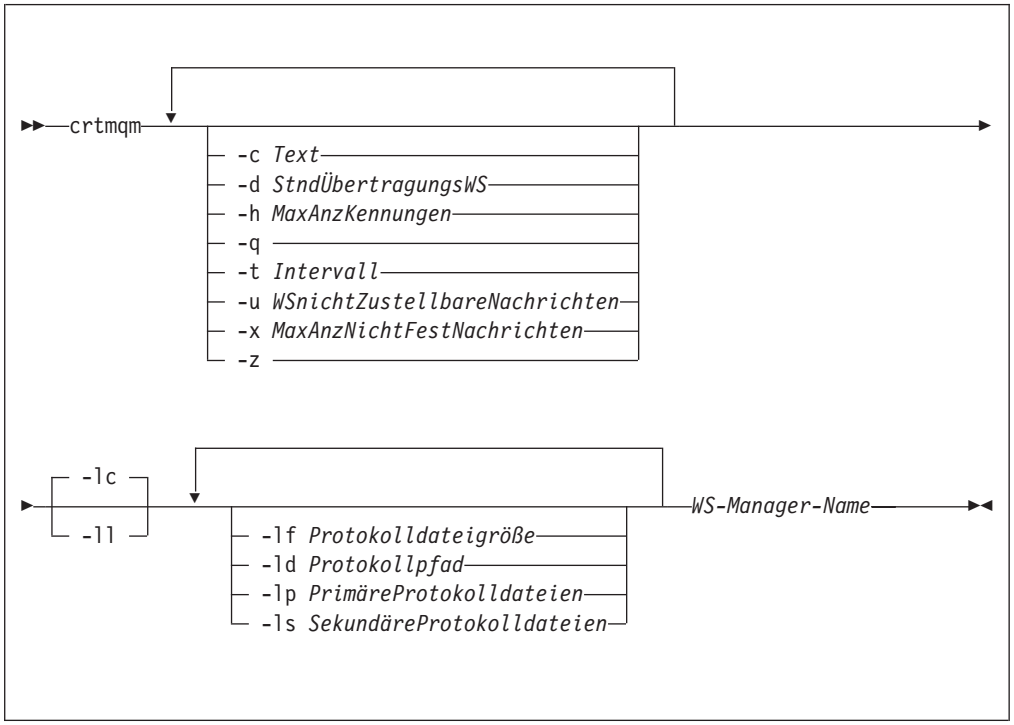
```
MQLONG Convertmy_structure(
             PMQBYTE *in_cursor,
             PMQBYTE *out_cursor,
             PMQBYTE in_lastbyte,
             PMQBYTE out_lastbyte,
             MQHCONN hConn,
             MQLONG  opts,
             MQLONG  MsgEncoding,
             MQLONG  ReqEncoding,
             MQLONG  MsgCCSID,
             MQLONG  ReqCCSID,
             MQLONG  CompCode,
             MQLONG  Reason)
{
    MQLONG ReturnCode = MQRC_NONE;

    ConvertLong(1); /* code */

    AlignLong();
    ConvertLong(1); /* value */

Fail:
    return(ReturnCode);
}
```

crtmqm (WS-Manager erstellen)

Funktion

Mit dem Befehl **crtmqm** erstellen Sie einen lokalen WS-Manager. Nachdem Sie einen WS-Manager erstellt haben, können Sie ihn mit dem Befehl **strmqm** starten.

Syntax



Erforderliche Parameter

WS-Manager-Name
Gibt den Namen des zu erstellenden WS-Managers an. Der Name darf maximal 48 Zeichen lang sein. Diese Angabe muss am Ende des Befehls stehen.

Optionale Parameter

-c *Text*
Dies ist ein beschreibender Text für diesen WS-Manager. Standardmäßig stehen hier Leerzeichen.

Der Text darf maximal 64 Zeichen lang sein. Wenn zwischen Groß-/Kleinschreibung unterschieden werden soll, muss die Beschreibung in doppelten Anführungszeichen stehen.

-d *StndÜbertragungsWS*
Gibt den Namen der lokalen Übertragungswarteschlange an, in die ferne Nachrichten gestellt werden, für die keine spezielle Übertragungswarteschlange als Zieladresse angegeben wurde. Es gibt keinen Standardwert.

-h *MaxAnzKennungen*
Gibt die maximale Anzahl von Kennungen an, die eine einzelne Anwendung gleichzeitig öffnen darf.

Geben Sie einen Wert im Bereich von 1 bis 999 999 999 an. Der Standardwert ist 256.

-q Gibt an, dass dieser WS-Manager der Standard-WS-Manager werden soll. Der neue WS-Manager ersetzt den eventuell vorhandenen Standard-WS-Manager.

Wenn Sie diese Option versehentlich angeben und zum vorherigen Standard-WS-Manager zurückkehren möchten, können Sie die Zeilengruppe *DefaultQueueManager* in der MQSeries-Konfigurationsdatei editieren. Weitere Informationen zu Konfigurationsdateien finden Sie in „Kapitel 13. MQSeries konfigurieren" auf Seite 179.

-t *Intervall*
Gibt das Auslösezeitintervall in Millisekunden für alle von diesem WS-Manager gesteuerten Warteschlangen an. Dieser Wert gibt die Zeit nach dem Empfang einer Nachricht mit einer Auslöserfunktion an, nach deren Ablauf die Auslösefunktion ausgesetzt wird. Das heißt, wenn beim Eingang einer Nachricht in einer Warteschlange eine Auslösenachricht in die Initialisierungswarteschlange eingereiht wird, wird bei weiteren Nachrichten, die innerhalb des angegebenen Intervalls in derselben Warteschlange ankommen, keine weitere Auslösenachricht generiert.

Über das Auslösezeitintervall können Sie sicherstellen, dass Ihre Anwendung genügend Zeit zur Bearbeitung einer Auslösebedingung hat, bevor sie durch eine Alarmmeldung veranlasst wird, sich um eine weitere Auslösebedingung in derselben Warteschlange zu kümmern. Wenn Sie über alle auftretenden Auslöseereignisse informiert werden möchten, geben Sie in diesem Feld einen niedrigen Wert oder null an.

Geben Sie einen Wert im Bereich von 0 bis 999 999 999 an. Der Standardwert ist 999 999 999 Millisekunden, eine Zeit von mehr als 11 Tagen. Wenn Sie den Standardwert übernehmen, bedeutet dies nichts anderes, als dass die Auslösefunktion nach der ersten Auslösenachricht inaktiviert wird. Die Auslösefunktion kann jedoch von einer Anwendung reaktiviert werden, die für die Warteschlange einen ALTER QUEUE-Befehl ausgibt, um die Auslöseattribute zurückzusetzen.

-u *WSnichtZustellbareNachrichten*
Gibt den Namen der lokalen Warteschlange an, die als Warteschlange für nicht zustellbare Nachrichten verwendet werden soll. In diese Warteschlange werden Nachrichten eingereiht, die nicht an ihre korrekte Zieladresse weitergeleitet werden können.

Wird kein Name angegeben, gilt standardmäßig, dass keine Warteschlange für nicht zustellbare Nachrichten vorhanden ist.

-x *MaxAnzNichtFestNachrichten*
Gibt die maximale Anzahl nicht festgeschriebener Nachrichten an einem Synchronisationspunkt an. Die Anzahl ergibt sich aus der Summe der folgenden Nachrichten:
- alle Nachrichten, die aus Warteschlangen abgerufen werden können.
- alle Nachrichten, die in Warteschlangen eingereiht werden können.
- alle Auslösenachrichten, die in dieser Arbeitseinheit generiert werden.

Diese Begrenzung gilt nicht für Nachrichten, die außerhalb eines Synchronisationspunktes abgerufen oder eingereiht werden.

Geben Sie einen Wert im Bereich von 1 bis 10 000 an. Der Standardwert sind 1000 nicht festgeschriebene Nachrichten.

-z Diese Option unterdrückt Fehlernachrichten.

Sie wird in MQSeries normalerweise verwendet, um unerwünschte Fehlernachrichten zu unterdrücken. Da bei Angabe dieser Option Informationen verloren gehen können, wird empfohlen, sie nicht beim Eingeben von Befehlen in einer Befehlszeile zu verwenden.

Mit der folgenden Gruppe von Optionen wird definiert, welche Protokollierung der erstellte WS-Manager verwenden soll. Weitere Information zu Protokollen finden Sie unter „Das Protokoll für eine Wiederherstellung verwenden" auf Seite 151.

-lc Es soll die zyklische Protokollierung verwendet werden. Dies ist die Standardprotokollierungsmethode.

-ll Es soll die lineare Protokollierung verwendet werden.

-lf *Protokolldateigröße*
Gibt die Größe der Protokolldateien in Einheiten von 4 KB an. Der Wert darf nicht kleiner als 64 und nicht größer als 16384 sein. Der Standardwert ist 1024, was einer Standardprotokollgröße von 4 MB entspricht.

-ld *Protokollpfad*
Gibt das Verzeichnis für die Protokolldateien an. Das Standardverzeichnis heißt `MQS_ROOT:[MQM.LOG]`. Der Standardwert kann bei der Anpassung von MQSeries geändert werden.

Die Benutzer-ID MQM und die Gruppe MQM müssen über alle Berechtigungen für die Protokolldateien verfügen. Wenn Sie die Verzeichnisse dieser Dateien ändern, müssen Sie sich selbst diese Berechtigungen erteilen. Dies geschieht automatisch, wenn sich die Protokolldateien in ihren Standardverzeichnissen befinden.

-lp *PrimäreProtokolldateien*
Gibt die Anzahl der primären Protokolldateien an, die angelegt werden sollen. Der Standardwert ist 3. Der Wert darf nicht kleiner als 2 und nicht größer als 62 sein.

-ls *SekundäreProtokolldateien*
Gibt die Anzahl der sekundären Protokolldateien an, die angelegt werden sollen. Der Standardwert ist 2. Der Wert darf nicht kleiner als 1 und nicht größer als 61 sein.

Anmerkung: Die Gesamtzahl der Protokolldateien ist auf 63 beschränkt, unabhängig von der angeforderten Anzahl.

Bei den oben in den Parameterbeschreibungen genannten Begrenzungen handelt es sich um MQSeries-Begrenzungen. Die maximal mögliche Protokollgröße kann durch Betriebssystembegrenzungen verringert werden.

Rückkehrcodes

0 WS-Manager wurde erstellt.
8 WS-Manager ist bereits vorhanden.
49 WS-Manager wird gestoppt.
69 Kein Speicher verfügbar.
70 Kein Speicher für Warteschlange verfügbar.
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name
100 Ungültiges Protokollverzeichnis
111 Der WS-Manager wurde erstellt. allerdings gab es einen Fehler bei der Verarbeitung der Definition für den Standard-WS-Manager in der Konfigurationsdatei des Produkts. Die Spezifikation für den Standard-WS-Manager ist möglicherweise nicht korrekt.
115 Ungültige Protokollgröße

Beispiele

1. Dieser Befehl erstellt einen Standard-WS-Manager mit dem Namen `Paint.WS.Manager` und der Beschreibung `Paint Shop`. Außerdem wird angegeben, dass lineare Protokollierung verwendet werden soll.

```
crtmqm -c "Paint Shop" -ll -q "Paint.WS.Manager"
```

2. In diesem Beispiel werden mehrere Protokolldateien angefordert. Es werden zwei primäre und drei sekundäre Protokolldateien angegeben.

```
crtmqm -c "Paint Shop" -ll -lp 2 -ls 3 -q "Paint.WS.Manager"
```

3. In diesem Beispiel wird ein anderer WS-Manager mit dem Namen `Reisen` erstellt. Das Auslöseintervall wird mit 5000 Millisekunden (oder 5 Sekunden) angegeben, und die Warteschlange für nicht zustellbare Nachrichten für diesen WS-Manager heißt SYSTEM.DEAD.LETTER.QUEUE.

```
crtmqm -t 5000 -u SYSTEM.DEAD.LETTER.QUEUE "Reisen"
```

   Nach der Generierung eines Auslöseereignisses werden weitere Auslöseereignisse für fünf Sekunden inaktiviert.

Zugehörige Befehle

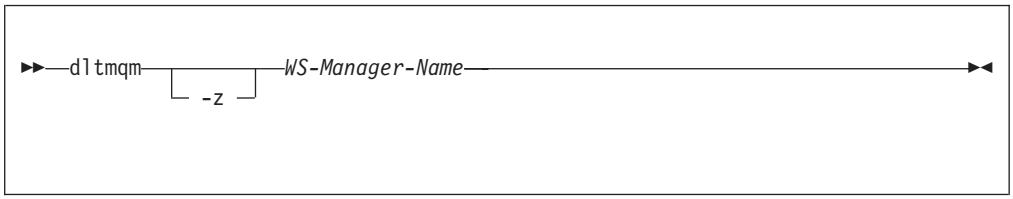
strmqm WS-Manager starten
endmqm WS-Manager beenden
dltmqm WS-Manager löschen

dltmqm (WS-Manager löschen)

Funktion

Mit dem Befehl **dltmqm** löschen Sie einen angegebenen WS-Manager. Alle diesem WS-Manager zugeordneten Objekte werden ebenfalls gelöscht. Bevor Sie einen WS-Manager löschen können, müssen Sie ihn mit dem Befehl **endmqm** beenden.

Syntax



Erforderliche Parameter

WS-Manager-Name
Gibt den Namen des zu löschenden WS-Managers an.

Optionale Parameter

-z Diese Option unterdrückt Fehlernachrichten.

Rückkehrcodes

0 WS-Manager gelöscht
3 WS-Manager wird erstellt
5 WS-Manager aktiv
16 WS-Manager nicht vorhanden
49 WS-Manager wird gestoppt
69 Kein Speicher verfügbar
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name
100 Ungültiges Protokollverzeichnis
112 WS-Manager wurde gelöscht. Allerdings gab es einen Fehler bei der Verarbeitung der Definition für den Standard-WS-Manager in der Konfigurationsdatei des Produkts. Die Spezifikation für den Standard-WS-Manager ist möglicherweise nicht korrekt.

Beispiele

1. Der folgende Befehl löscht den WS-Manager `saturn.queue.manager`.

```
dltmqm "saturn.queue.manager"
```

2. Der folgende Befehl löscht den WS-Manager `Reisen` und unterdrückt alle durch den Befehl verursachten Fehlernachrichten.

```
dltmqm -z "Reisen"
```

Zugehörige Befehle

crtmqm	WS-Manager erstellen
strmqm	WS-Manager starten
endmqm	WS-Manager beenden

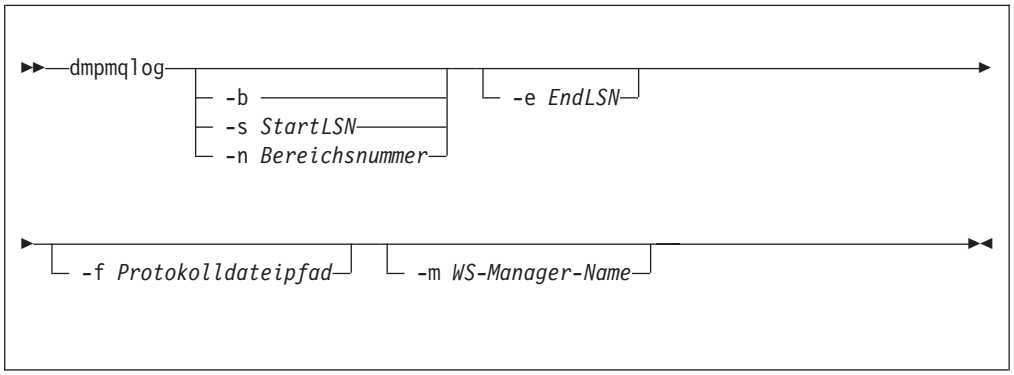
dmpmqlog (Protokollauszug erstellen)

Funktion

Mit dem Befehl **dmpmqlog** erstellen Sie eine formatierte Version des MQSeries-Systemprotokolls.

Das betreffende Protokoll muss auf demselben Betriebssystemtyp erstellt worden sein wie das, auf dem der Befehl ausgegeben wird.

Syntax



Optionale Parameter

Startpunkt für den Auszug
Geben Sie einen der folgenden Parameter an, um die Protokollfolgenummer (LSN, Log Sequence Number) auszuwählen, mit der der Auszug beginnen soll. Wenn kein Startpunkt angegeben wird, beginnt der Auszug standardmäßig mit der Protokollfolgenummer des ersten Satzes im aktiven Teil des Protokolls.

-b Gibt an, dass der Auszug mit der Basis-Protokollfolgenummer beginnen soll. Die Basis-Protokollfolgenummer identifiziert den Protokollspeicherbereich, in dem sich der Anfang des aktiven Teils des Protokolls befindet.

-s *StartLSN*
Gibt an, dass der Auszug mit der angegebenen Protokollfolgenummer beginnen soll. Die Protokollfolgenummer wird im Format `nnnn:nnnn:nnnn:nnnn` angegeben.

Wenn Sie eine zyklische Protokollierung verwenden, muss die angegebene Protokollfolgenummer gleich oder größer als die Basis-Protokollfolgenummer des Protokolls sein.

-n *Bereichsnummer*
Gibt an, dass der Auszug mit der angegebenen Protokollspeicherbereichsnummer beginnen soll. Die Nummer des Protokollspeicherbereichs muss im Bereich 0–9 999 999 liegen.

Dieser Parameter ist nur für WS-Manager gültig, für die als *LogType* (in der Konfigurationsdatei qm.ini) der Wert LINEAR angegeben ist.

-e *EndLSN*
Gibt an, dass der Auszug mit der angegebenen Protokollfolgenummer enden soll. Die Protokollfolgenummer wird im Format `nnnn:nnnn:nnnn:nnnn` angegeben.

-f *Protokolldateipfad*
Dies ist der absolute, nicht der relative, Verzeichnispfadname für die Protokolldateien. Das angegebene Verzeichnis muss die Protokoll-Header-Datei (`amqhlctl.lfh`) und ein Unterverzeichnis mit dem Namen `active` enthalten. Im Unterverzeichnis 'active' müssen sich die Protokolldateien befinden. Standardmäßig wird davon ausgegangen, dass sich die Protokolldateien in den Verzeichnissen befinden, die in den Dateien `mqs.ini` und `qm.ini` angegeben sind. Bei Angabe dieser Option werden Namen von Warteschlangen, denen Warteschlangen-IDs zugeordnet sind, nur dann im Auszug angezeigt, wenn für die Option -m explizit ein WS-Manager-Name angegeben wird und sich die Objektkatalogdatei im Verzeichnispfad dieses WS-Managers befindet.

Auf Systemen, die lange Dateinamen unterstützen, hat diese Datei den Namen `qmqmobjcat`. Damit die Warteschlangen-IDs den Warteschlangennamen zugeordnet werden können, muss diese Datei beim Erstellen der Protokolldateien verwendet worden sein. Zum Beispiel befindet sich die Objektkatalogdatei für den WS-Manager QM1 im Verzeichnis `MQS_ROOT:[MQM.QMGRS.QM1.QMANAGER]`. Um diese Zuordnung zu erreichen, kann es nötig sein, einen temporären WS-Manager zu erstellen (z. B. mit dem Namen tmpq), dessen Objektkatalog durch den zu ersetzen, der den speziellen Protokolldateien zugeordnet ist, und dann den Befehl dmpmqlog mit den Optionen `-m tmpq` und `-f` mit dem absoluten Verzeichnispfadnamen der Protokolldateien auszugeben.

-m *WS-Manager-Name*
Dies ist der Name des WS-Managers. Wird dieser Parameter nicht angegeben, wird der Name des Standard-WS-Managers verwendet.

Der angegebene WS-Manager bzw. der verwendete Standard-WS-Manager dürfen nicht aktiv sein, wenn der Befehl **dmpmqlog** ausgegeben wird. Ebenso gilt, dass der WS-Manager nicht gestartet werden darf, solange der Befehle **dmpmqlog** ausgeführt wird.

dspmqaut (Berechtigung anzeigen)

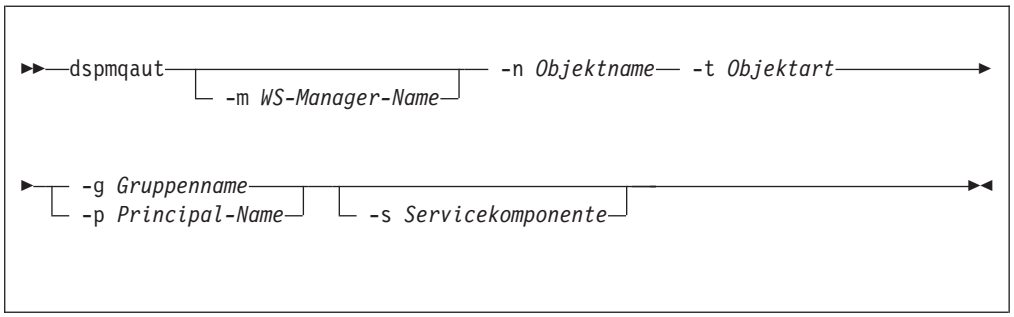
Funktion

Mit dem Befehl **dspmqaut** zeigen Sie die aktuellen Berechtigungen für ein bestimmtes Objekt an.

Es kann nur eine einzige Gruppe angegeben werden.

Wenn eine Benutzer-ID Mitglied mehrerer Gruppen ist, zeigt dieser Befehl die kombinierten Berechtigungen aller Gruppen an.

Syntax



Erforderliche Parameter

-n *Objektname*
Gibt den Namen des Objekts an, das abgefragt wird.

Dieser Parameter ist erforderlich, ***außer*** wenn sich die Abfrage an den WS-Manager selbst richtet. In diesem Fall muss er nicht angegeben werden.

Sie müssen den Namen des WS-Managers, der Warteschlange oder der Prozessdefinition angeben.

-t *Objektart*
Gibt die Art des Objekts an, das abgefragt wird. Gültige Werte:

queue oder q Eine Warteschlange oder Warteschlangen, die dem Parameter für die Objektart entsprechen.

qmgr Ein WS-Manager-Objekt

process oder prcs
Ein Prozess

namelist oder nl
Eine Namensliste

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, der abgefragt werden soll.

-g *Gruppenname*
Gibt den Namen der Benutzergruppe an, die abgefragt werden soll. Sie können nur ***einen*** Namen angeben, wobei es sich um den Namen einer vorhandenen Berechtigungs-ID handeln muss.

-p *Principal-Name*
Gibt den Namen eines Benutzers an, dessen Berechtigungen für das angegebenen Objekt angezeigt werden sollen.

-s *Servicekomponente*
Dieser Parameter gilt nur, wenn Sie installierbare Berechtigungsservices verwenden, andernfalls wird er ignoriert.

Wenn installierbare Berechtigungsservices unterstützt werden, gibt dieser Parameter den Namen des Berechtigungsservices an, für den die Berechtigungen gelten. Dieser Parameter ist optional; wenn er nicht angegeben wird, richtet sich die Abfrage der Berechtigung an die erste installierbare Komponente für den Service.

Zurückgegebene Parameter

Dieser Befehl gibt eine Berechtigungsliste zurück, die keinen, einen oder mehrere Berechtigungsparameter enthalten kann. Für jeden zurückgegebenen Berechtigungsparameter gilt, dass jede Benutzer-ID in der angegebenen Gruppe über die Berechtigung verfügt, die durch diesen Parameter definierte Operation auszuführen.

Tabelle 16 zeigt die Berechtigungen, die an die unterschiedlichen Objektarten erteilt werden können.

Tabelle 16. Sicherheitsberechtigungen mit dem Befehl dspmqaut

Berechtigung	queue	process	qmgr	namelist
all	✓	✓	✓	✓
alladm	✓	✓	✓	✓
allmqi	✓	✓	✓	✓
altusr			✓	
browse	✓			
chg	✓	✓	✓	✓
clr	✓			
connect			✓	
cpy	✓	✓	✓	✓
crt	✓	✓	✓	✓
dlt	✓	✓	✓	✓
dsp	✓	✓	✓	✓
get	✓			
inq	✓	✓	✓	✓
passall	✓			
passid	✓			
put	✓			
SET	✓	✓	✓	
setall	✓		✓	
setid	✓		✓	

dspmqaut

In der folgenden Liste werden die Berechtigungen definiert, die den einzelnen Parametern zugeordnet sind:

all Ausführen aller Operationen für das Objekt.

alladm Ausführen aller Verwaltungsoperationen für das Objekt.

allmqi Ausführen aller MQI-Aufrufe für das Objekt.

altusr Angeben einer alternativen Benutzer-ID in einem MQI-Aufruf.

browse Abrufen einer Nachricht aus einer Warteschlange mit dem Aufruf MQGET und der Option BROWSE.

chg Ändern der Attribute eines angegebenen Objekts mit Hilfe des entsprechenden Befehlssatzes.

clr Löschen des Inhalts einer Warteschlange (nur PCF-Befehl 'Clear queue').

connect Verbinden der Anwendung mit dem angegebenen WS-Manager durch Ausgeben eines MQCONN-Aufrufs.

cpy Kopieren der Definition eines Objekts, z. B. PCF-Befehl 'Copy queue'.

crt Erstellen von Objekten der angegebenen Objektart mit Hilfe des entsprechenden Befehlssatzes.

dlt Löschen des angegebenen Objekts mit Hilfe des entsprechenden Befehlssatzes.

dsp Anzeigen der Attribute des angegebenen Objekts mit Hilfe des entsprechenden Befehlssatzes.

get Abrufen einer Nachricht aus einer Warteschlange durch Ausgeben eines MQGET-Aufrufs.

inq Abfragen einer angegebenen Warteschlange durch Ausgeben eines MQINQ-Aufrufs.

passall Übergeben des gesamten Kontextes.

passid Übergeben des Identitätskontextes.

put Einreihen einer Nachricht in eine angegebene Warteschlange durch Ausgeben eines MQPUT-Aufrufs.

set Setzen von Attributen für eine Warteschlange über die MQI durch Ausgeben eines MQSET-Aufrufs.

setall Setzen des gesamten Kontextes für eine Warteschlange.

setid Setzen des Identitätskontextes für eine Warteschlange.

Die Berechtigungen für Verwaltungsoperationen, sofern unterstützt, gelten für die folgenden Befehlssätze:

- Steuerbefehle
- MQSC-Befehle
- PCF-Befehle

Rückkehrcodes

0 Operation war erfolgreich
36 Ungültige Argumente übergeben
40 WS-Manager ist nicht verfügbar
49 WS-Manager wird gestoppt
69 Kein Speicher verfügbar
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name
133 Unbekannter Objektname
145 Unerwarteter Objektname
146 Objektname fehlt
147 Objektart fehlt
148 Ungültige Objektart
149 Name der Definitionseinheit fehlt

Beispiele

Das folgende Beispiel zeigt einen Befehl zum Anzeigen der Berechtigungen für den WS-Manager `saturn.queue.manager` für die Benutzergruppe `Personal`:

```
dspmqaut -m "saturn.queue.manager" -t qmgr -g Personal
```

Das Ergebnis dieses Befehls lautet wie folgt:

```
Definitionseinheit Personal verfügt über die folgenden Berechtigungen für Objekt:
        get
        browse
        put
        inq
        set
        connect
        altusr
        passid
        passall
        setid
```

Zugehörige Befehle

setmqaut Berechtigung setzen oder zurücksetzen

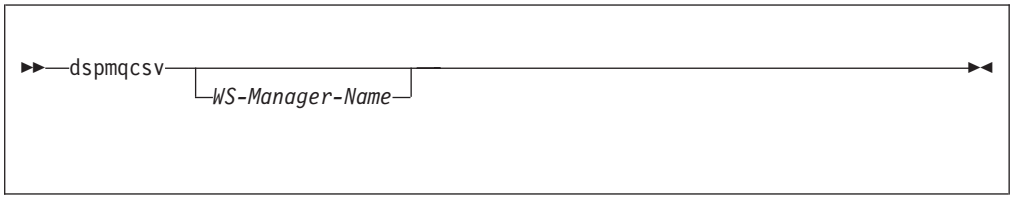
dspmqcsv (Befehlsserver anzeigen)

Funktion

Mit dem Befehl **dspmqcsv** zeigen Sie den Status des Befehlsservers für den angegebenen WS-Manager an.

Folgende Statusangaben sind möglich:
- Starten
- Aktiv
- Aktiv (ohne dass SYSTEM.ADMIN.COMMAND.QUEUE für GET-Aufrufe aktiviert ist)
- Stoppen
- Gestoppt

Syntax



Optionale Parameter

WS-Manager-Name
Gibt den Namen des lokalen WS-Managers an, für den der Status des Befehlsservers abgefragt wird.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

Der folgenden Befehl zeigt den Status des Befehlsservers, der dem WS-Managers `venus.q.mgr` zugeordnet ist, an:

```
dspmqcsv "venus.q.mgr"
```

Zugehörige Befehle

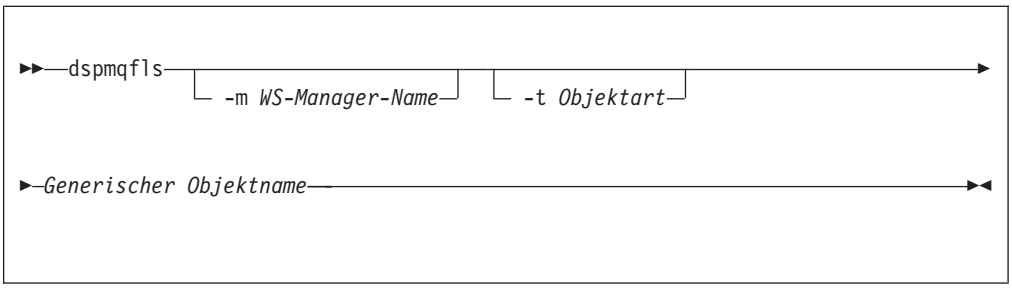
strmqcsv Befehlsserver starten
endmqcsv Befehlsserver beenden

dspmqfls (MQSeries-Dateien anzeigen)

Funktion

Mit dem Befehl **dspmqfls** zeigen Sie die tatsächlichen Dateisystemnamen für alle MQSeries-Objekte an, die einem bestimmten Kriterium entsprechen. Mit Hilfe dieses Befehls können Sie die Dateien identifizieren, die einem bestimmten MQSeries-Objekt zugeordnet sind. Dies ist bei der Sicherung bestimmter Objekte hilfreich. Weitere Informationen zur Namensumwandlung finden Sie unter „Erläuterungen zu MQSeries-Dateinamen” auf Seite 21.

Syntax



Erforderliche Parameter

Generischer Objektname
Gibt den Namen des MQSeries-Objekts an. Der Name besteht aus einer Zeichenfolge ohne Markierungen; die Angabe dieses Parameters ist erforderlich. Wenn kein Name angegeben wird, wird ein Fehler zurückgegeben.

Dieser Parameter unterstützt das Platzhalterzeichen * am Ende der Zeichenfolge.

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, für den Dateien überprüft werden. Wird kein Name angegeben, gilt der Befehl für den Standard-WS-Manager.

-t *Objektart*
Gibt die Art des MQSeries-Objekts an. In der folgenden Liste werden die gültigen Objektarten aufgeführt. Zunächst wird die Abkürzung genannt, daneben steht der vollständiger Name der Objektart.

*** oder all** Alle Objektarten; dies ist der Standardwert.

q oder queue Eine Warteschlange oder Warteschlangen, die dem Parameter für den Objektnamen entsprechen.

ql oder qlocal Eine lokale Warteschlange

qa oder qalias Eine Aliaswarteschlange

qr oder qremote
Eine ferne Warteschlange

qm oder qmodel
Eine Modellwarteschlange

qmgr Ein WS-Manager-Objekt

prcs oder process
Ein Prozess

ctlg oder catalog
Ein Objektkatalog

nl oder namelist
Eine Namensliste

Anmerkung: Der Befehl **dspmqfls** zeigt das Verzeichnis an, in dem sich die Warteschlange befindet, und ***nicht*** den Namen der Warteschlange.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde beendet, aber nicht vollständig wie erwartet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

1. Der folgende Befehl zeigt die Details zu allen Objekten an, deren Namen mit `SYSTEM.ADMIN` beginnen und die auf dem Standard-WS-Manager definiert sind.

```
dspmqfls SYSTEM.ADMIN*
```

2. Der folgende Befehl zeigt Dateiinformationen für alle Prozesse an, deren Namen mit `PROC` beginnen und die auf dem WS-Manager `RADIUS` definiert sind.

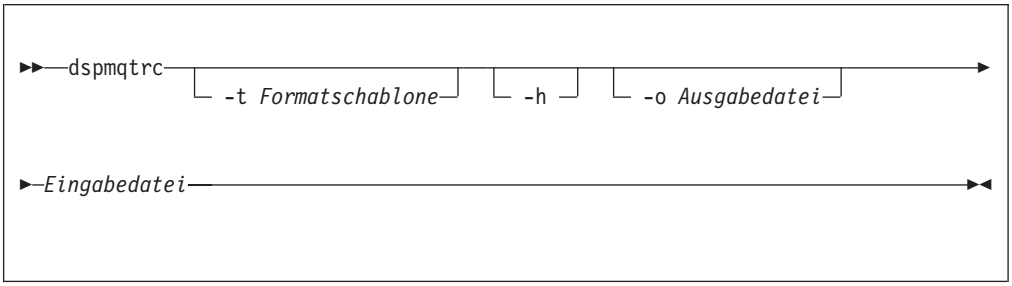
```
dspmqfls -m RADIUS -t prcs PROC*
```

dspmqtrc (Formatierte MQSeries-Trace-Ausgabe anzeigen)

Funktion

Mit dem Befehl **dspmqtrc** zeigen Sie eine formatierte MQSeries-Trace-Ausgabe an.

Syntax



Erforderliche Parameter

Eingabedatei
Gibt den Namen der Datei an, die den unformatierten Trace enthält. Beispiel: `MQS_ROOT:[MQM.TRACE]AMQ20202345.TRC`.

Optionale Parameter

-t *Formatschablone*
Gibt den Namen der Schablonendatei an, die beschreibt, wie der Trace angezeigt werden soll. Der Standardwert ist `SYS$SHARE:AMQTRC.FMT`.

-h Header-Daten des Berichts werden nicht angezeigt.

-o *Ausgabedatei*
Der Name der Datei, in die die formatierten Daten geschrieben werden sollen.

Beispiele

1. Der folgende Befehl zeigt die Umleitung der Ausgabe:

```
dspmqtrc mqs_root:[mqm.trace]amq20202345.trc > mqs_root:[mqm.trace]amq20202345.fmt
```

Zugehörige Befehle

endmqtrc	MQSeries-Trace beenden
strmqtrc	MQSeries-Trace starten

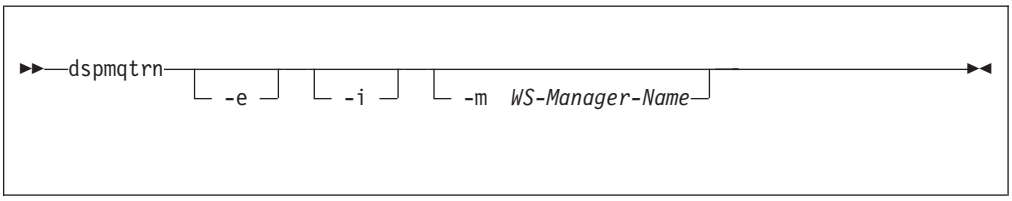
dspmqtrn (MQSeries-Transaktionen anzeigen)

Funktion

Mit dem Befehl **dspmqtrn** zeigen Sie Transaktionen an, die sich in einem zweiphasigen Festschreibungsprozess im PREPARED-Status befinden und dem WS-Manager bekannt sind (siehe unten den Hinweis unter 'Achtung').

Zu jeder Transaktion wird die Transaktionsnummer (eine lesbare Vorgangsnummer), der Transaktionsstatus und die Transaktions-ID angezeigt. Transaktions-IDs können maximal 128 Zeichen lang sein, woraus sich die Notwendigkeit einer Vorgangsnummer ergibt.

Syntax



Achtung: Diesen Befehl werden Sie wahrscheinlich nur benötigen, wenn Sie einen externen Transaktionsmanager verwenden und an zweiphasigen Festschreibungsprozessen teilnehmen. Wenn Sie keine zweiphasige Festschreibung durchführen, verwenden Sie diesen Befehl nicht. Der Befehl sollte nur verwendet werden, wenn der Synchronisationspunktmanager eine Transaktion nicht auflösen konnte.

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, dessen Transaktionen überprüft werden sollen. Wird kein Name angegeben, gilt der Befehl für den Standard-WS-Manager.

-e Abfragedetails extern koordinierter, unbestätigter Transaktionen. Dabei handelt es sich um Transaktionen, für die MQSeries zur Vorbereitung der Festschreibung aufgefordert wurde, zu denen MQSeries jedoch noch keine Transaktionsergebnisse vorliegen.

-i Abfragedetails intern koordinierter, unbestätigter Transaktionen. Dabei handelt es sich um Transaktionen, für die jeder Ressourcenmanager zur Vorbereitung der Festschreibung aufgefordert wurde, die Ressourcenmanager aber von MQSeries noch nicht über die Transaktionsergebnisse informiert wurden.

Es werden Informationen zum abgeleiteten Status der Transaktion in jedem teilnehmenden Ressourcenmanager angezeigt. Mit Hilfe dieser Informationen können Sie die Auswirkungen von Fehlern in einem einzelnen Ressourcenmanager beurteilen.

Anmerkung: Wenn Sie weder -e noch -i angeben, werden Details sowohl der intern als auch der extern koordinierten, unbestätigten Transaktionen angezeigt.

Rückkehrcodes

0	Operation war erfolgreich
36	Ungültige Argumente übergeben
40	WS-Manager ist nicht verfügbar
49	WS-Manager wird gestoppt
69	Kein Speicher verfügbar
71	Unerwarteter Fehler
72	Falscher WS-Manager-Name
102	Keine Transaktionen gefunden

Zugehörige Befehle

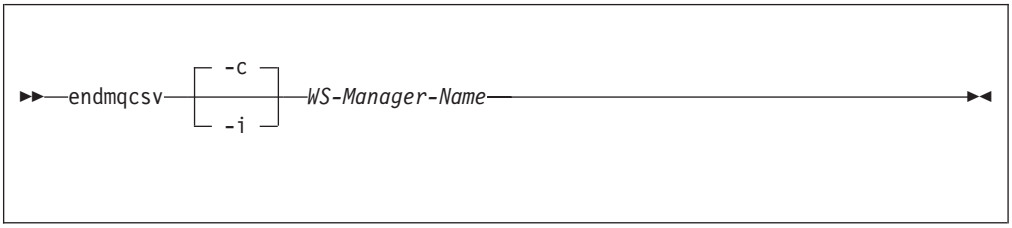
rsvmqtrn	MQSeries-Transaktion auflösen

endmqcsv (Befehlsserver beenden)

Funktion

Mit dem Befehl **endmqcsv** stoppen Sie den Befehlsserver auf dem angegebenen WS-Manager.

Syntax



Erforderliche Parameter

WS-Manager-Name
Gibt den Namen des WS-Managers an, für den der Befehlsserver beendet werden soll.

Optionale Parameter

-c Gibt an, dass der Befehlsserver auf kontrollierte Weise gestoppt werden soll. Der Befehlsserver darf die Verarbeitung aller bereits gestarteten Befehlsnachrichten beenden. Es werden keine neuen Nachrichten aus der Befehlswarteschlange gelesen.

Dies ist die Standardeinstellung.

-i Gibt an, dass der Befehlsserver sofort gestoppt wird. Aktionen in Verbindung mit einer zu dem Zeitpunkt ausgeführten Befehlsnachricht werden möglicherweise nicht beendet.

Rückkehrcodes

0 Befehl wurde normal beendet.

10 Befehl wurde mit unerwarteten Ergebnissen beendet.

20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

1. Der folgende Befehl stoppt den Befehlsserver auf WS-Manager `saturn.queue-.manager`:

```
endmqcsv  -c "saturn.queue.manager"
```

   Der Befehlsserver kann die Verarbeitung der bereits gestarteten Befehle beenden, bevor er stoppt. Alle neu empfangenen Befehle verbleiben unbearbeitet in der Befehlswarteschlange, bis der Befehlsserver erneut gestartet wird.
2. Der folgende Befehl stoppt den Befehlsserver auf WS-Manager `pluto` sofort:

   ```
   endmqcsv -i "pluto"
   ```

Zugehörige Befehle

strmqcsv Befehlsserver starten
dspmqcsv Status eines Befehlsservers anzeigen

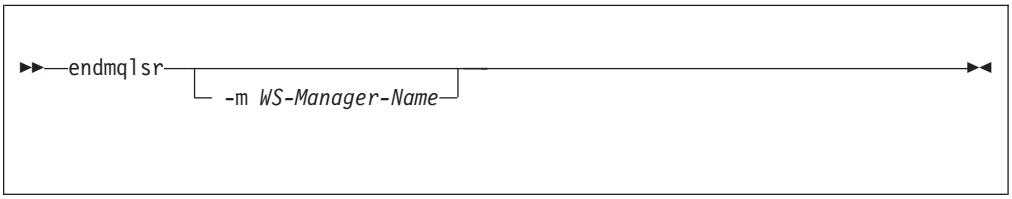
endmqlsr (Empfangsprogramm beenden)

Funktion

Mit dem Befehl **endmqlsr** beenden Sie alle Empfangsprogramme für den angegebenen WS-Manager.

Der WS-Manager muss gestoppt werden, bevor der Befehl **endmqlsr** ausgegeben wird.

Syntax



Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an. Wird kein Name angegeben, erfolgt die Verarbeitung für den Standard-WS-Manager.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

endmqm (WS-Manager beenden)

Funktion

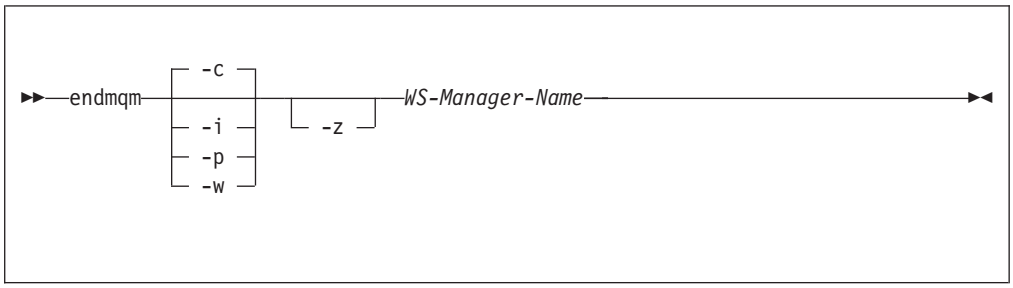
Mit dem Befehl **endmqm** beenden (stoppen) Sie einen angegebenen lokalen WS-Manager. Dieser Befehl stoppt den WS-Manager auf eine von drei Arten:

- normaler bzw. gesteuerter Abschluss
- sofortiger Abschluss
- erzwungener Abschluss

Die Attribute des WS-Managers und die ihm zugeordneten Objekte sind nicht betroffen. Sie können den WS-Manager mit dem Befehl **strmqm** (WS-Manager starten) erneut starten.

Um einen WS-Manager zu löschen, müssen Sie ihn zunächst stoppen, bevor Sie dann den Befehl **dltmqm** (WS-Manager löschen) verwenden.

Syntax



Erforderliche Parameter

WS-Manager-Name
Gibt den Namen des Nachrichten-WS-Managers an, der gestoppt werden soll.

Optionale Parameter

-c Kontrollierter (bzw. gesteuerter) Abschluss. Der WS-Manager stoppt erst, nachdem alle Anwendungen ihre Verbindungen getrennt haben. Alle zu dem Zeitpunkt verarbeiteten MQI-Aufrufe werden beendet. Dies ist die Standardeinstellung.

Die Steuerung wird sofort an Sie zurückgegeben, und Sie werden nicht darüber benachrichtigt, wann der WS-Manager gestoppt wurde.

-w Bestätigter Abschluss

Diese Abschlussart entspricht dem gesteuerten Abschluss, außer dass die Steuerung erst an Sie zurückgegeben wird, nachdem der WS-Manager gestoppt wurde. Sie erhalten während des Abschlussvorgangs die Nachricht 'Warte auf Beendigung von WS-Manager *WS-Manager-Name*'.

-i Sofortiger Abschluss. Der WS-Manager stoppt, nachdem er alle zu dem Zeitpunkt verarbeiteten MQI-Aufrufe beendet hat. Alle MQI-Anforderungen, die nach Ausgabe des Befehls eingehen, schlagen fehl. Nicht beendete Arbeitseinheiten werden zurückgesetzt, wenn der WS-Manager das nächste Mal gestartet wird.

-p Erzwungener Abschluss.

Verwenden Sie diese Abschlussart nur in Ausnahmesituationen. Zum Beispiel, wenn ein WS-Manager nach Ausgabe eines normalen **endmqm**-Befehls nicht gestoppt wird.

Der WS-Manager stoppt, ohne darauf zu warten, dass Anwendungen ihre Verbindung beenden oder MQI-Aufrufe beendet werden. Dies kann unvorhersehbare Folgen für MQSeries-Anwendungen haben. Alle Prozesse des WS-Managers, der nicht gestoppt werden konnte, werden 30 Sekunden nach Ausgabe des Befehls beendet.

Anmerkung: Nach einem erzwungenen Abschluss oder einem Fehler des WS-Managers wurde der WS-Manager möglicherweise beendet, ohne dass der von ihm verwaltete gemeinsam benutzte Speicher bereinigt wurde. Dies kann zu Problemen beim Neustart führen. Informationen zur Verwendung des Dienstprogramms MONMQ zum Bereinigen des Speichers nach einer abrupten Beendigung dieser Art finden Sie unter „Gemeinsam benutzten Speicher mit MONMQ verwalten" auf Seite 375.

-z Unterdrückt Fehlermeldungen für diesen Befehl.

Rückkehrcodes

0 WS-Manager wurde beendet
3 WS-Manager wird erstellt
16 WS-Manager nicht vorhanden
40 WS-Manager ist nicht verfügbar
49 WS-Manager wird gestoppt
69 Kein Speicher verfügbar
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name

Beispiele

Das folgende Beispiel zeigt Befehle, mit denen der angegebene WS-Manager beendet (gestoppt) wird.

1. Dieser Befehl beendet den WS-Manager `mercury.queue.manager` auf kontrollierte Weise. Alle Anwendungen, mit denen eine Verbindung besteht, können ihre Verbindungen trennen.

```
endmqm "mercury.queue.manager"
```

2. Dieser Befehl beendet den WS-Manager `saturn.queue.manager` sofort. Alle aktuellen MQI-Aufrufe werden beendet, neue Aufrufe sind jedoch nicht zulässig.

```
endmqm -i "saturn.queue.manager"
```

Zugehörige Befehle

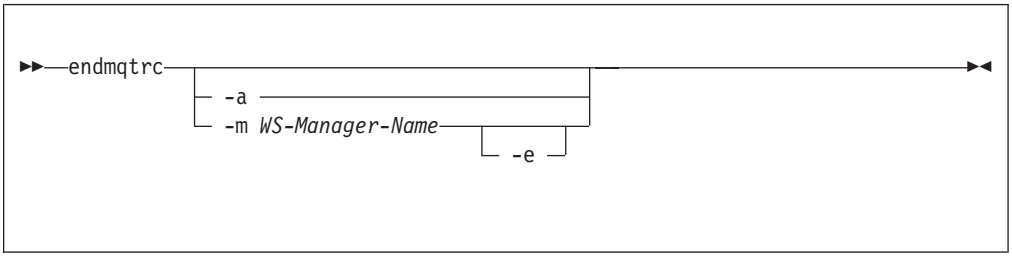
crtmqm	WS-Manager erstellen
strmqm	WS-Manager starten
dltmqm	WS-Manager löschen

endmqtrc (MQSeries-Trace beenden)

Funktion

Mit dem Befehl **endmqtrc** beenden Sie den Trace für die angegebene Definitionseinheit bzw. für alle Definitionseinheiten.

Syntax



Optionale Parameter

-m *WS-Manager-Name*
Dies ist der Name des WS-Managers, für den der Trace beendet werden soll.

In diesem Befehl kann nur eine Option -m und ein zugehöriger WS-Manager-Name angegeben werden.

Ein WS-Manager-Name und eine Option -m kann in demselben Befehl wie die Option -e angegeben werden.

-e Bei Angabe dieser Option wird der Trace in der Startphase beendet.

-a Bei Angabe dieser Option werden alle Traces beendet.

Diese Option ***muss*** allein angegeben werden.

Rückkehrcodes

AMQ5611 Diese Nachricht wird ausgegeben, wenn im Befehl ungültige Argumente angegeben wurden.

Beispiele

Dieser Befehle beendet den Daten-Trace für den WS-Manager `QM1`.

```
endmqtrc -m QM1
```

Zugehörige Befehle

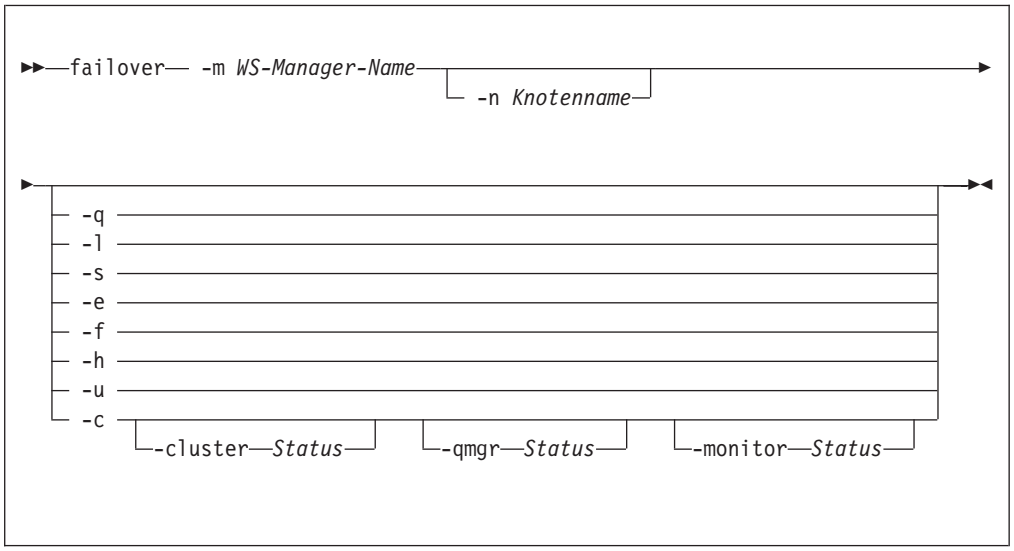
dspmqtrc Formatierte Trace-Ausgabe anzeigen
strmqtrc MQSeries-Trace starten

failover (Überbrückungsgruppe verwalten)

Funktion

Mit dem Befehl **failover** verwalten Sie eine Überbrückungsgruppe. Der Befehl **failover** enthält sowohl Aktualisierungs- als auch Abfrageparameter. Der Befehl **failover** kann auf jedem OpenVMS-Knoten in der Überbrückungsgruppe ausgeführt werden.

Syntax



Erforderliche Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, auf den der Befehl **failover** angewendet werden soll. Der Name für *WS-Manager-Name* darf maximal 25 Zeichen lang sein.

-n *Knotenname*
Gibt den Namen des OpenVMS-Knotens an, auf den der Befehl angewendet wird. Dieser Parameter ist für die Optionen -h und -c erforderlich.

Optionale Parameter

-q Fragt den Status der Überbrückungsgruppe ab und zeigt die Ausgabe an.

-l Fragt den Status der Überbrückungsgruppe ab und übergibt Werte an die folgenden DCL-Symbole:

DCL-Symbolname	Beschreibung
MQS$QMGR_NODE	Enthält den Namen des OpenVMS-Knotens, auf dem der WS-Manager aktiv ist, oder eine Nullzeichenfolge, wenn kein WS-Manager aktiv ist.
MQS$AVAILABLE_NODES	Enthält die Liste der OpenVMS-Knoten, die für die Ausführung des WS-Managers zur Verfügung stehen. Das sind alle Knoten, die sich im WS-Manager-Status AVAILABLE befinden und auf denen ein Überbrückungsmonitor aktiv ist.
MQS$MONITOR_NODES	Enthält die Liste der OpenVMS-Knoten, auf denen ein Überbrückungsmonitor aktiv ist.

-s Startet den WS-Manager in der Überbrückungsgruppe. Wenn der Parameter -n angegeben, wird der WS-Manager auf dem angegebenen OpenVMS-Knoten gestartet; andernfalls wird er auf dem Knoten mit der höchsten Priorität gestartet.

-e Beendet den WS-Manager in der Überbrückungsgruppe.

-f Versetzt den WS-Manager auf einen anderen Knoten in der Überbrückungsgruppe. Wenn der Parameter -n angegeben wird, wird der WS-Manager auf den angegebenen Knoten versetzt; andernfalls wird er auf den Knoten mit der höchsten Priorität versetzt.

-h Hält den Überbrückungsmonitor an, der auf dem mit dem Parameter -n angegebenen Knoten aktiv ist.

-u Entfernt die Markierung für Aktualisierungen.

-c Ändert den Status der Überbrückungsgruppe. Der jeweils zu ändernde Status wird durch die folgenden drei Parameter festgelegt. Änderungen werden nur durchgeführt, wenn sie mit dem aktuellen Status der Überbrückungsgruppe konsistent sind.

-cluster started | stopped
Wird mit dem Parameter -c angegeben und ändert den Status der gesamten Überbrückungsgruppe.

-qmgr available | running | excluded
Wird mit dem Parameter -c angegeben und ändert den WS-Manager-Status des Knotens, der mit dem Parameter -n angegeben wird.

-monitor started | stopped | watching
Wird mit dem Parameter -c angegeben und ändert den Status des Überbrückungsmonitors für den Knoten, der mit dem Parameter -n angegeben ist.

Rückkehrcodes

0 Befehl wurde normal beendet.
5 WS-Manager ist aktiv
36 In einem Befehl angegebene Argumente sind ungültig.
326 MQseries-WS-Manager ist nicht aktiv.
1925 Für den WS-Manager wurde kein Überbrückungsmonitor gestartet.
1926 Für die Überbrückungsgruppe wird eine Aktualisierungsoperation ausgeführt.
1937 Kein Knoten verfügbar, auf dem der WS-Manager gestartet werden kann.
1939 Beendigung des WS-Managers wurde erzwungen.
1940 Beendigung des WS-Managers hat zulässiges Zeitlimit überschritten.

OpenVMS-Fehlercodes
36 %SYSTEM-F-NOPRIV, keine ausreichende Berechtigung bzw. Versuch, auf geschützte Objekte zuzugreifen
652 %SYSTEM-F-NOSUCHNODE, ferner Knoten unbekannt
660 %SYSTEM-F-REJECT, Verbindung mit Netzobjekt zurückgewiesen

Beispiele

1. Diese Beispiel fragt den Status einer Überbrückungsgruppe für den WS-Manager `testqm` ab.

```
failover -m "testqm" -q
```

2. Dieses Beispiel startet den WS-Manager `testqm` auf dem Knoten `batman`.

```
failover -m "testqm" -n batman -s
```

3. Dieses Beispiel versetzt den WS-Manager `testqm` auf den Knoten mit der höchsten Priorität.

```
failover -m "testqm" -f
```

Zugehörige Befehle

runmqfm Überbrückungsmonitor starten

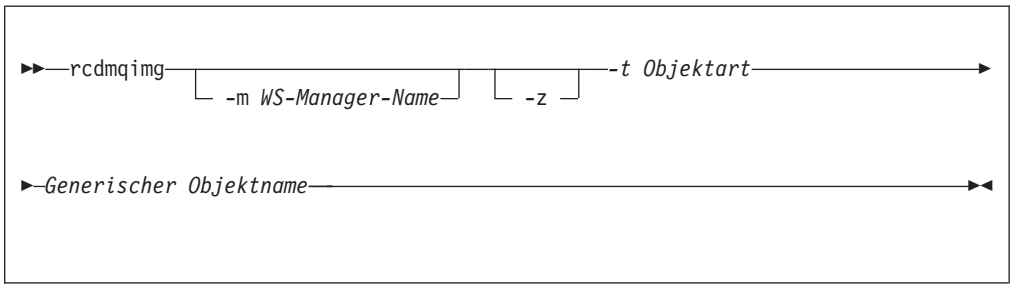
rcdmqimg (Datenträger-Image aufzeichnen)

Funktion

Mit dem Befehl **rcdmqimg** schreiben Sie ein Image eines MQSeries-Objekts oder einer Objektgruppe in das Protokoll, um es bei der Wiederherstellung über Datenträger zu verwenden. Mit dem Befehl **rcrmqobj** können Sie das Objekt aus dem Image erneut erstellen.

Dieser Befehl wird mit einem aktiven WS-Manager verwendet. Weitere Aktivitäten auf dem WS-Manager werden protokolliert, so dass die Protokollsätze alle Änderungen des Objekts wiedergeben, obwohl das Image nicht mehr aktuell ist.

Syntax



Erforderliche Parameter

Generischer Objektname
Gibt den Namen des Objekts an, das aufgezeichnet werden soll. Am Ende dieses Parameters kann ein Stern stehen, so dass alle Objekte, deren Namen mit dem Teil des Namens vor dem Stern übereinstimmen, aufgezeichnet werden.

Dieser Parameter ist erforderlich, *außer* wenn Sie ein WS-Manager-Objekt oder die Kanalsynchronisationsdatei aufzeichnen. Wenn Sie einen Objektnamen für die Kanalsynchronisationsdatei angeben, wird er ignoriert.

-t *Objektart*
Gibt die Art der Objekte an, deren Images aufgezeichnet werden sollen. Folgende Objektarten sind gültig:

prcs oder process
Prozesse

q oder queue Alle Warteschlangenarten

ql oder qlocal Lokale Warteschlangen

qa oder qalias Aliaswarteschlangen

qr oder qremote
Ferne Warteschlangen

qm oder qmodel
Modellwarteschlangen

qmgr WS-Manager-Objekt

syncfile Kanalsynchronisationsdatei

nl oder namelist
Namenslisten

ctlg oder catalog
Ein Objektkatalog

*** oder all** Alle oben genannten Objektarten

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, für den Images aufgezeichnet werden sollen. Wird kein Name angegeben, gilt der Befehl für den Standard-WS-Manager.

-z Diese Option unterdrückt Fehlernachrichten.

Rückkehrcodes

0	Operation war erfolgreich
36	Ungültige Argumente übergeben
40	WS-Manager ist nicht verfügbar
49	WS-Manager wird gestoppt
68	Wiederherstellung über Datenträger wird nicht unterstützt
69	Kein Speicher verfügbar
71	Unerwarteter Fehler
72	Falscher WS-Manager-Name
119	Benutzer nicht berechtigt
128	Keine Objekte verarbeitet
131	Ressourcenfehler
132	Objekt ist beschädigt
135	Temporäres Objekt kann nicht aufgezeichnet werden

Beispiele

Der folgende Befehl zeichnet ein Image des WS-Manager-Objekts `saturn.queue.manager` im Protokoll auf.

```
rcdmqimg -t qmgr -m "saturn.queue.manager"
```

Zugehörige Befehle

rcrmqobj WS-Manager-Objekt erneut erstellen

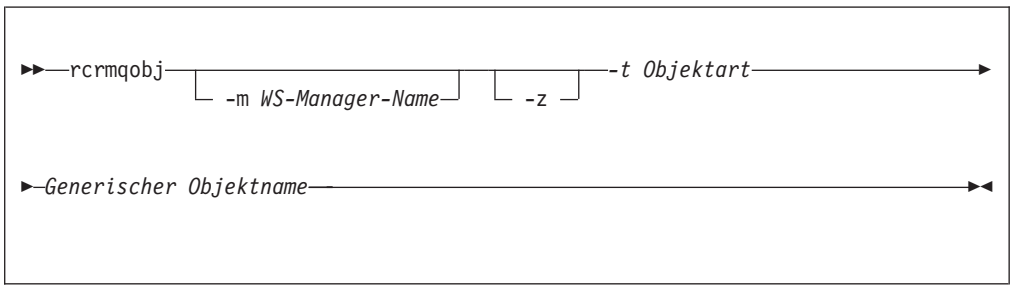
rcrmqobj (Objekt erneut erstellen)

Funktion

Mit dem Befehl **rcrmqobj** erstellen Sie ein Objekt oder eine Objektgruppe erneut aus ihren im Protokoll gespeicherten Images. Verwenden Sie den zugehörigen Befehl **rcdmqimg**, um die Objekt-Images im Protokoll aufzuzeichnen.

Dieser Befehl muss auf einem aktiven WS-Manager ausgegeben werden. Alle Aktivitäten auf dem WS-Manager nach der Aufzeichnung des Images werden protokolliert. Um das Objekt erneut zu erstellen, müssen Sie die im Protokoll gespeicherten Ereignisse, die nach dem Aufzeichnen des Images eingetreten sind, erneut ausführen.

Syntax



Erforderliche Parameter

Generischer Objektname
Gibt den Namen des Objekts an, das erneut erstellt werden soll. Am Ende dieses Parameters kann ein Stern stehen, so dass alle Objekte, deren Namen mit dem Teil des Namens vor dem Stern übereinstimmen, erneut erstellt werden.

Dieser Parameter ist erforderlich, *außer* wenn es sich bei der Objektart um die Kanalsynchronisationsdatei handelt; wenn für diese Objektart ein Name angegeben wird, wird er ignoriert.

-t *Objektart*
Gibt die Art der Objekte an, die erneut erstellt werden sollen. Folgende Objektarten sind gültig:

prcs oder process
Prozesse

q oder queue Alle Warteschlangenarten

ql oder qlocal Lokale Warteschlangen

qa oder qalias Aliaswarteschlangen

qr oder qremote
Ferne Warteschlangen

qm oder qmodel
Modellwarteschlangen

nl oder namelist
Namenslisten

ctlg oder catalog
Ein Objektkatalog

*** oder all** Alle oben genannten Objektarten

syncfile Die Kanalsynchronisationsdatei

Anmerkung: Bei Angabe dieser Option wird die Kanalsynchronisationsdatei für den angegebenen WS-Manager erneut generiert. Dies ist notwendig, weil die Datei nicht mit dem Befehl **rcdmqimg** gespeichert wird.

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, für den Objekte erneut erstellt werden sollen. Wird kein Name angegeben, gilt der Befehl für den Standard-WS-Manager.

-z Diese Option unterdrückt Fehlernachrichten.

Rückkehrcodes

0 Operation war erfolgreich
36 Ungültige Argumente übergeben
40 WS-Manager ist nicht verfügbar
49 WS-Manager wird gestoppt
66 Datenträger-Image nicht verfügbar
68 Wiederherstellung über Datenträger wird nicht unterstützt
69 Kein Speicher verfügbar
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name
119 Benutzer nicht berechtigt
128 Keine Objekte verarbeitet
135 Temporäres Objekt kann nicht wiederhergestellt werden
136 Objekt wird verwendet

Beispiele

1. Der folgende Befehl erstellt alle lokalen Warteschlangen für den Standard-WS-Manager erneut:

```
rcrmqobj -t ql *
```

2. Der folgende Befehl erstellt alle fernen Warteschlangen, die dem WS-Manager `Lager` zugeordnet sind, erneut:

```
rcrmqobj -m "Lager" -t qr *
```

Zugehörige Befehle

rcdmqimg MQSeries-Objekt im Protokoll aufzeichnen

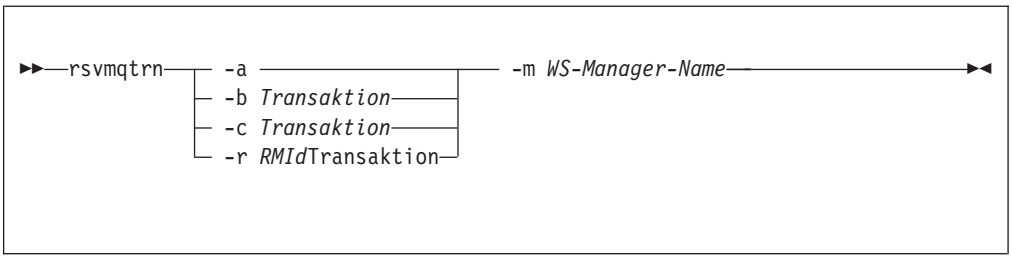
rsvmqtrn (MQSeries-Transaktionen auflösen)

Funktion

Mit dem Befehl **rsvmqtrn** schreiben Sie intern oder extern koordinierte unbestätigte Transaktionen fest oder setzen sie zurück.

Verwenden Sie diesen Befehl nur, wenn Sie sicher sind, dass Transaktionen nicht von den normalen Protokollen aufgelöst werden können. Das Ausgeben dieses Befehls kann dazu führen, dass die Transaktionsintegrität zwischen Ressourcenmanagern und einer verteilten Transaktion verloren geht.

Syntax



Erforderliche Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an. Dieser Parameter ist obligatorisch.

Optionale Parameter

-a Gibt an, dass der WS-Manager versuchen soll, alle intern koordinierten unbestätigten Transaktionen (d. h. alle globalen Arbeitseinheiten) aufzulösen.

-b Gibt an, dass die angegebene Transaktion zurückgesetzt werden soll. Diese Option ist nur für extern koordinierte Transaktionen (d. h. für externe Arbeitseinheiten) gültig.

-c Gibt an, dass die angegebene Transaktion festgeschrieben werden soll. Diese Option ist nur für extern koordinierte Transaktionen (d. h. für externe Arbeitseinheiten) gültig.

-r *RMId*
Identifiziert den Ressourcenmanager, für den die Festschreibungs- bzw. Zurücksetzungsentscheidung gilt. Diese Option ist nur für intern koordinierte Transaktionen gültig sowie für Ressourcenmanager, die nicht mehr in der Datei `qm.ini` des WS-Managers konfiguriert sind. Die übergebene Ausgabe ist mit der Entscheidung von MQSeries für die Transaktion konsistent.

Transaktion
Dies ist die Vorgangsnummer der festgeschriebenen oder zurückgesetzten Transaktion. Sie können die richtige Vorgangsnummer mit dem Befehl **dspmqtrn** herausfinden. Dieser Parameter ist für die *RMId*-Parameter -b, -c und -r erforderlich.

Rückkehrcodes

0	Operation war erfolgreich
32	Transaktionen konnten nicht aufgelöst werden
34	Ressourcenmanager nicht erkannt
35	Ressourcenmanager nicht permanent verfügbar
36	Ungültige Argumente übergeben
40	WS-Manager ist nicht verfügbar
49	WS-Manager wird gestoppt
69	Kein Speicher verfügbar
71	Unerwarteter Fehler
72	Falscher WS-Manager-Name
85	Transaktionen unbekannt

Zugehörige Befehle

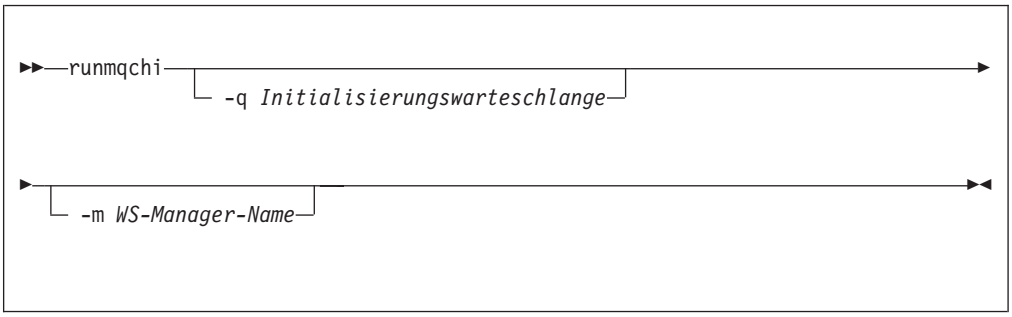
dspmqtrn	Liste mit vorbereiteten Transaktionen anzeigen

runmqchi (Kanalinitiator ausführen)

Funktion

Mit dem Befehl **runmqchi** führen Sie einen Kanalinitiatorprozess aus. Weitere Informationen zu diesem Befehl finden Sie im Handbuch *MQSeries Intercommunication*.

Syntax



Optionale Parameter

-q *Initialisierungswarteschlange*
Gibt den Namen der Initialisierungswarteschlange an, die von diesem Kanalinitiator verarbeitet werden soll. Erfolgt keine Angabe, wird die Warteschlange SYSTEM.CHANNEL.INITQ verwendet.

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, auf dem sich die Initialisierungswarteschlange befindet. Wird kein Name angegeben, wird der Standard-WS-Manager verwendet.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Wenn Fehler auftreten, die zu einem Rückkehrcode 10 oder 20 führen, überprüfen Sie die Protokolldatei des WS-Managers, dem der Kanal zugeordnet ist, auf Fehlernachrichten. Überprüfen Sie außerdem das Fehlerprotokoll $SYSTEM, denn dort werden Fehler aufgezeichnet, die auftreten, bevor der Kanal dem WS-Manager zugeordnet wird. Weitere Informationen zu Fehlerprotokollen finden Sie unter „Fehlerprotokolle” auf Seite 210.

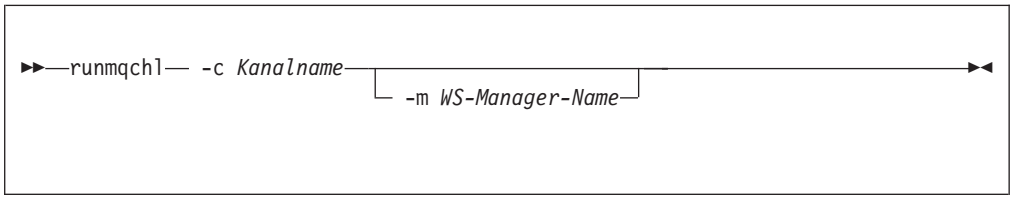
runmqchl (Kanal ausführen)

Funktion

Mit dem Befehl **runmqchl** führen Sie einen Sender (SDR) bzw. einen Requester (RQSTR) aus.

Der Kanal wird synchron ausgeführt. Geben Sie zum Stoppen des Kanals MQSC-Befehl STOP CHANNEL aus.

Syntax



Erforderliche Parameter

-c *Kanalname*
Gibt den Namen des auszuführenden Kanals an.

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, dem dieser Kanal zugeordnet ist. Wird kein Name angegeben, wird der Standard-WS-Manager verwendet.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Wenn die Rückkehrcodes 10 oder 20 generiert werden, überprüfen Sie die Fehlerprotokolle des zugeordneten WS-Managers auf Fehlernachrichten. Überprüfen Sie außerdem das Fehlerprotokoll $SYSTEM, denn dort werden Fehler aufgezeichnet, die auftreten, bevor der Kanal dem WS-Manager zugeordnet wird.

runmqdlq (Steuerroutine der DLQ ausführen)

Funktion

Mit dem Befehl **runmqdlq** starten Sie die Steuerroutine der Warteschlange für nicht zustellbare Nachrichten (DLQ, Dead Letter Queue), ein Dienstprogramm, mit dem Sie Nachrichten in einer Warteschlange für nicht zustellbare Nachrichten überwachen und bearbeiten können.

Über die Steuerroutine der Warteschlange für nicht zustellbare Nachrichten können verschiedenen Aktionen für ausgewählte Nachrichten ausgeführt werden, indem eine Regelgruppe angegeben wird, mit der sowohl eine Nachricht ausgewählt als auch die Aktion für diese Nachricht definiert wird.

Der Befehl **runmqdlq** erhält seine Eingabe von `SYS$INPUT`. Nach der Verarbeitung des Befehls wird ein Bericht mit den Ergebnissen und einer Zusammenfassung erstellt und an `SYS$OUTPUT` gesendet.

Indem Sie die Eingabe von `SYS$INPUT` auf die Tastatur umleiten, können Sie Regeln für den Befehl **runmqdlq** interaktiv eingeben.

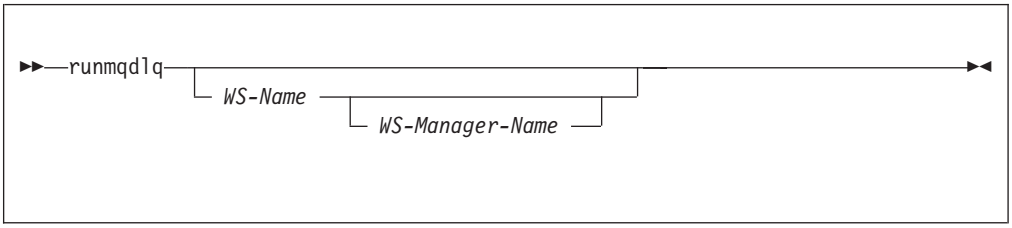
Indem Sie als Eingabequelle eine Datei angeben, können Sie eine Regeltabelle auf die angegebene Warteschlange anwenden. Die Regeltabelle muss mindestens eine Regel enthalten.

Wenn die Steuerroutine der Warteschlange für nicht zustellbare Nachrichten im Vordergrund läuft, ohne dass statt `SYS$INPUT` eine Datei (mit der Regeltabelle) als Eingabequelle verwendet wird, macht die Steuerroutine Folgendes:

- Sie erwartet Eingaben über die Tastatur.
- Sie beginnt erst mit der Verarbeitung der angegebenen Warteschlange, wenn sie ein Dateiendezeichen (Strg-Z) empfängt.

Weitere Informationen zu Regeltabellen und zu deren Erstellung finden Sie unter „Die Regeltabelle der DLQ-Steuerroutine" auf Seite 107.

Syntax



Optionale Parameter

Die MQSC-Regeln für Kommentarzeilen und das Verknüpfen von Zeilen gelten auch für die Eingabeparameter der Steuerroutine der Warteschlange für nicht zustellbare Nachrichten.

WS-Name
Gibt den Namen der zu verarbeitenden Warteschlange an.

Wird kein Name angegeben, wird die Warteschlange für nicht zustellbare Nachrichten, die für den lokalen WS-Manager definiert wurde, verwendet. Wenn Leerzeichen (' ') angegeben werden, wird die Warteschlange für nicht zustellbare Nachrichten des lokalen WS-Managers explizit zugeordnet.

Mit Hilfe einer Steuerroutine der Warteschlange für nicht zustellbare Nachrichten können einzelne Nachrichten in einer Warteschlange für nicht zustellbare Nachrichten für eine spezielle Verarbeitung ausgewählt werden. Sie können die Nachrichten zum Beispiel an unterschiedliche Warteschlangen für nicht zustellbare Nachrichten umleiten. Für die anschließende Verarbeitung der Nachrichten durch ein anderes Exemplar der Steuerroutine kann dann eine andere Regeltabelle verwendet werden.

WS-Manager-Name
Dies ist der Name des WS-Managers, der die zu verarbeitende Warteschlange steuert.

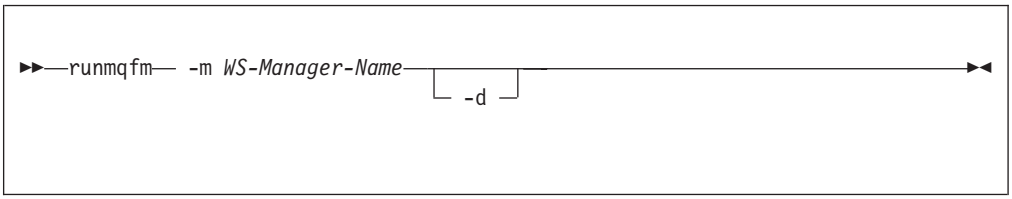
Wird kein Name angegeben, wird der Standard-WS-Manager der Installation verwendet. Wenn Leerzeichen (' ') angegeben werden, wird der Standard-WS-Manager der Installation explizit zugeordnet.

runmqfm (Überbrückungsmonitor starten)

Funktion

Mit dem Befehl **runmqfm** starten Sie einen Überbrückungsmonitor auf einem OpenVMS-Knoten. Der Überbrückungsmonitor wird auf dem OpenVMS-Knoten ausgeführt, auf dem der Befehl **runmqfm** ausgegeben wird.

Syntax



Erforderliche Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, für den der Befehl **runmqfm** gestartet werden soll. Der Name für *WS-Manager-Name* darf maximal 25 Zeichen lang sein.

Optionale Parameter

-d Gibt an, dass zusätzliche Debug-Informationen in die Protokolldatei geschrieben werden sollen.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

Der folgende Befehl startet einen Überbrückungsmonitor für den WS-Manager `testqm` und schreibt Debug-Informationen in die Protokolldatei `test.log`.

```
runmqfm -m "testqm" -d > test.log
```

Zugehörige Befehle

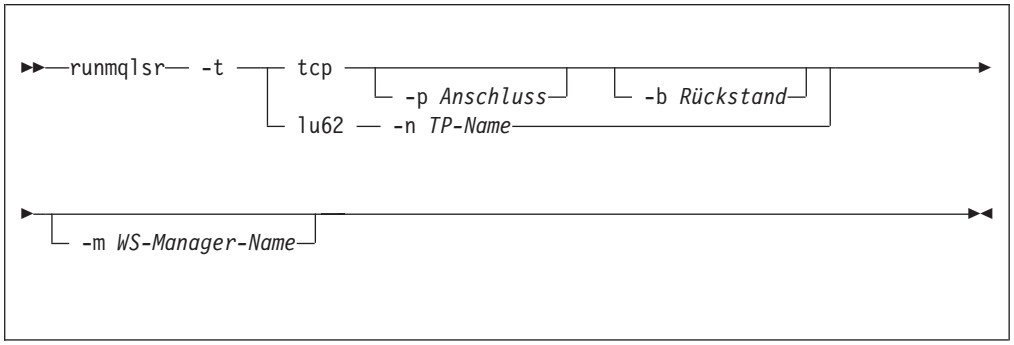
failover Überbrückungsgruppe verwalten

runmqlsr (Empfangsprogramm ausführen)

Funktion

Mit dem Befehl **runmqlsr** führen Sie ein Empfangsprogramm aus.

Syntax



Erforderliche Parameter

-t Gibt das zu verwendende Übertragungsprotokoll an:

tcp Transmission Control Protocol/Internet Protocol (TCP/IP)

lu62 SNA LU 6.2. (Die neuesten Informationen zur Verwendung dieses Parameters finden Sie in den Release-Hinweisen in `sys$help:mqseries0510.release_notes`.)

Optionale Parameter

-p *Anschluss*
Gibt die Port-Nummer für TCP/IP an. Diese Option ist für TCP und UDP gültig. Ist kein Wert angegeben, wird der Wert aus der Konfigurationsdatei des WS-Managers oder aus Standardwerten im Programm übernommen. Der Standardwert ist 1414.

-n *TP-Name*
Gibt das LU 6.2-Transaktionsprogramm an. Diese Option ist nur für das LU 6.2-Übertragungsprotokoll gültig. Ist kein Wert angegeben, wird der Wert aus der Konfigurationsdatei des WS-Managers übernommen. Wenn dort kein Wert enthalten ist, wird der Befehl nicht ausgeführt.

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an. Wird kein Name angegeben, gilt der Befehl für den Standard-WS-Manager.

-b *Rückstand*
Gibt die Anzahl der gleichzeitigen Verbindungsanforderungen an, die das Empfangsprogramm unterstützt. Eine Liste der Standardwerte und weitere Informationen finden Sie unter „Die Zeilengruppen LU62 und TCP" auf Seite 194.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

Der folgende Befehl führt ein Empfangsprogramm auf dem Standard-WS-Manager über das TCP/IP-Protokoll aus. Der Befehl gibt an, dass das Empfangsprogramm die Port-Nummer 4321 verwenden soll.

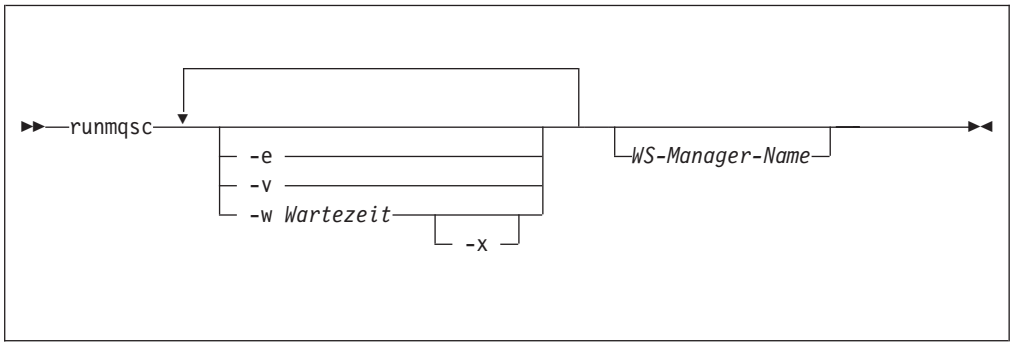
```
runmqlsr -t tcp -p 4321
```

runmqsc (MQSeries-Befehle ausführen)

Funktion

Mit dem Befehl **runmqsc** geben Sie MQSC-Befehle an einen WS-Manager aus. Mit Hilfe von MQSC-Befehlen können Sie Verwaltungs-Tasks ausführen, z. B. ein lokales Warteschlangenobjekt definieren, ändern oder löschen. MQSC-Befehle und ihre Syntax werden im Handbuch *MQSeries Application Programming Guide* beschrieben.

Syntax



Beschreibung

Sie können den Befehl **runmqsc** in drei Modi aufrufen:

Prüfmodus
MQSC-Befehle werden geprüft, aber nicht ausgeführt. Es wird ein Bericht generiert, der für jeden einzelnen Befehl angibt, ob er erfolgreich ausgeführt werden kann oder nicht. Dieser Modus ist nur auf einem lokalen WS-Manager verfügbar.

Direkter Modus
MQSC-Befehle werden direkt an einen lokalen WS-Manager gesendet.

Indirekter Modus
MQSC-Befehle werden auf einem fernen WS-Manager ausgeführt. Diese Befehle werden in die Befehlswarteschlange auf einem fernen WS-Manager eingereiht und in der Reihenfolge ausgeführt, in der sie eingereiht wurden. Berichte des Befehls werden an den lokalen WS-Manager zurückgegeben.

Anmerkung: Die Benutzer-ID, unter der der ferne WS-Manager ausgeführt wird, muss lokal vorhanden sein und über die richtigen Berechtigungen verfügen.

Der Befehl **runmqsc** erhält seine Eingabe von `SYS$INPUT`. Nach Verarbeitung der Befehle wird ein Bericht mit den Ergebnissen und einer Zusammenfassung an `SYS$OUTPUT` gesendet.

Indem Sie die Eingabe von `SYS$INPUT` auf die Tastatur umleiten, können Sie MQSC-Befehle interaktiv eingeben.

Indem Sie als Eingabequelle eine Datei angeben, können Sie häufig verwendete Befehlsfolgen in der Datei eingeben und so leichter ausführen. Sie können auch den Ausgabebericht in eine Datei umleiten.

Optionale Parameter

-e Verhindert, dass Quellentext für die MQSC-Befehle in einen Bericht kopiert wird. Dies ist hilfreich, wenn Sie Befehle interaktiv eingeben.

-v Gibt an, das der Prüfmodus verwendet wird, d. h., die angegebenen Befehl werden geprüft, die Aktionen jedoch nicht ausgeführt. Dieser Modus ist nur lokal verfügbar. Die Optionen -w und -x werden ignoriert, wenn sie gleichzeitig angegeben werden.

-w *Wartezeit*
Gibt an, dass der indirekte Modus verwendet wird, d. h., die MQSC-Befehle werden auf einem anderen WS-Manager ausgeführt. Dazu müssen Sie den erforderlichenne Kanal und die erforderlichen Übertragungswarteschlangen definiert haben.

Wartezeit
Gibt die Zeit in Sekunden an, die **runmqsc** auf Antworten wartet. Alle nach Ablauf dieser Zeit eingehenden Antworten werden gelöscht, die MQSC-Befehle jedoch weiter ausgeführt. Geben Sie eine Zeit von 1 bis 999 999 Sekunden an.

Jeder Befehl wird als PCF-Escape-Befehl an die Befehlswarteschlange (SYSTEM.ADMIN.COMMAND.QUEUE) auf dem Ziel-WS-Manager gesendet.

Die Antworten werden in der Warteschlange SYSTEM.MQSC.REPLY-.QUEUE empfangen, und die Ausgabe wird dem Bericht hinzugefügt. Diese kann entweder als lokale Warteschlange oder als Modellwarteschlange definiert werden.

Operationen im indirekten Modus werden über den Standard-WS-Manager ausgeführt.

Diese Option wird ignoriert, wenn die Option -v angegeben ist.

-x Gibt an, dass der Ziel-WS-Manager unter MVS/ESA aktiv ist. Diese Option kann nur im indirekten Modus angegeben werden. Die Option -w muss ebenfalls angegeben werden. Im indirekten Modus werden die MQSC-Befehle in einem Format geschrieben, dass mit dem der MQSeries for MVS/ESA-Befehlswarteschlange kompatibel ist.

WS-Manager-Name
Gibt den Namen des Ziel-WS-Managers an, auf dem die MQSC-Befehle ausgeführt werden sollen. Wird kein Name angegeben, werden die MQSC-Befehle auf dem Standard-WS-Manager ausgeführt.

Rückkehrcodes

0 MQSC-Befehlsdatei wurde erfolgreich ausgeführt.
10 MQSC-Befehlsdatei wurde mit Fehlern ausgeführt - Fehlerursachen sind im Bericht beschrieben.
20 Fehler - MQSC-Befehlsdatei nicht ausgeführt.

Beispiele

1. Geben Sie diesen Befehl in der OpenVMS-Eingabeaufforderung ein:

```
runmqsc
```

   Jetzt können Sie MQSC-Befehle direkt in der OpenVMS-Eingabeaufforderung eingeben. Da kein WS-Manager angegeben wurde, werden die MQSC-Befehle auf dem Standard-WS-Manager ausgeführt.

2. Geben Sie diesen Befehl ein, wenn die MQSC-Befehle nur geprüft werden sollen:

```
runmqsc -v BANK < DKA0:[USERS]COMMFILE.IN
```

   Mit diesem Befehl wird die MQSC-Befehlsdatei `COMMFILE.IN` im Verzeichnis `DKA0:[USERS]` geprüft. Der Name des WS-Managers lautet `BANK`. Die Ausgabe wird im aktuellen Fenster angezeigt.

3. Dieser Befehl führt die MQSC-Befehlsdatei `MQS_ROOT:[MQM.MQSC]MQSCFILE.IN` auf dem Standard-WS-Manager aus.

```
runmqsc < MQS_ROOT:[MQM.MQSC]MQSCFILE.IN > MQS_ROOT:[MQM.MQSC]MQSCFILE.OUT
```

   In diesem Beispiel wird die Ausgabe in die Datei `MQS_ROOT:[MQM.MQSC]MQSCFILE.OUT` umgeleitet.

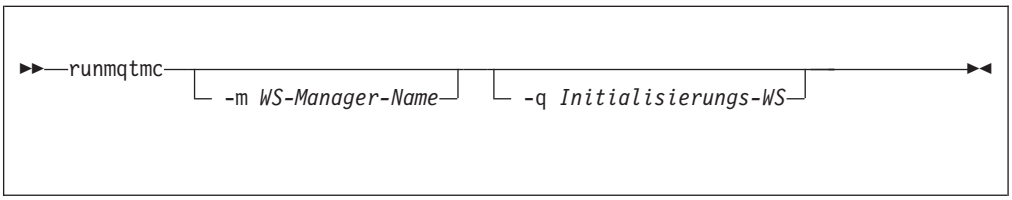
runmqtmc (Client-Auslösemonitor starten)

Funktion

Mit dem Befehl **runmqtmc** rufen Sie einen Auslösemonitor für einen Client auf. Weitere Informationen zur Verwendung von Auslösemonitoren finden Sie im Handbuch *MQSeries Application Programming Guide*.

Anmerkung: Dieser Befehl ist ***nur*** auf OpenVMS-, OS/2- und AIX-Clients verfügbar.

Syntax



Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, auf dem der Client-Auslösemonitor ausgeführt werden soll. Wird kein Name angegeben, wird der Client-Auslösemonitor auf dem Standard-WS-Manager ausgeführt.

-q *Initialisierungs-WS*
Gibt den Namen der zu verarbeitenden Initialisierungswarteschlange an. Wird kein Name angegeben, wird die Warteschlange SYSTEM.DEFAULT.INITIATION.QUEUE verwendet.

Rückkehrcodes

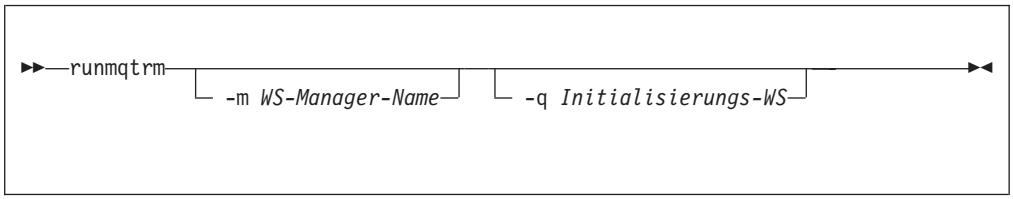
0 Nicht verwendet. Der Client-Auslösemonitor ist so konfiguriert, dass er ständig ausgeführt wird; er wird also nicht beendet. Der Wert ist reserviert.
10 Client-Auslösemonitor wurde durch einen Fehler unterbrochen.
20 Fehler - Client-Auslösemonitor nicht aktiv.

runmqtrm (Auslösemonitor starten)

Funktion

Mit dem Befehl **runmqtrm** rufen Sie einen Auslösemonitor auf. Weitere Informationen zur Verwendung von Auslösemonitoren finden Sie im Handbuch *MQSeries Application Programming Guide*.

Syntax



Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, auf dem der Auslösemonitor ausgeführt werden soll. Wird kein Name angegeben, wird der Auslösemonitor auf dem Standard-WS-Manager ausgeführt.

-q *Initialisierungs-WS*
Gibt den Namen der zu verarbeitenden Initialisierungswarteschlange an. Wird kein Name angegeben, wird die Warteschlange SYSTEM.DEFAULT.INITIATION.QUEUE verwendet.

Rückkehrcodes

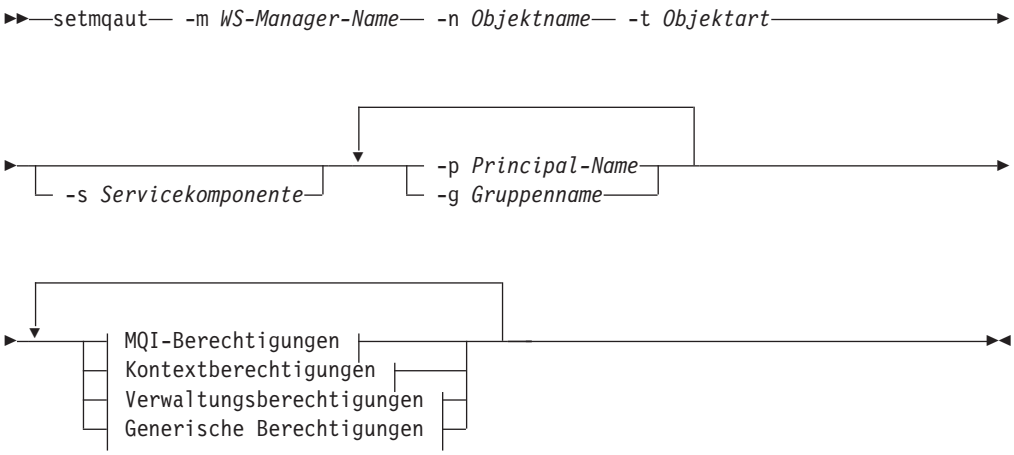
0 Nicht verwendet. Der Auslösemonitor ist so konfiguriert, dass er ständig ausgeführt wird; er wird also nicht beendet. Der Wert ist reserviert.
10 Auslösemonitor wurde durch einen Fehler unterbrochen.
20 Fehler - Auslösemonitor nicht aktiv.

setmqaut (Berechtigung setzen/zurücksetzen)

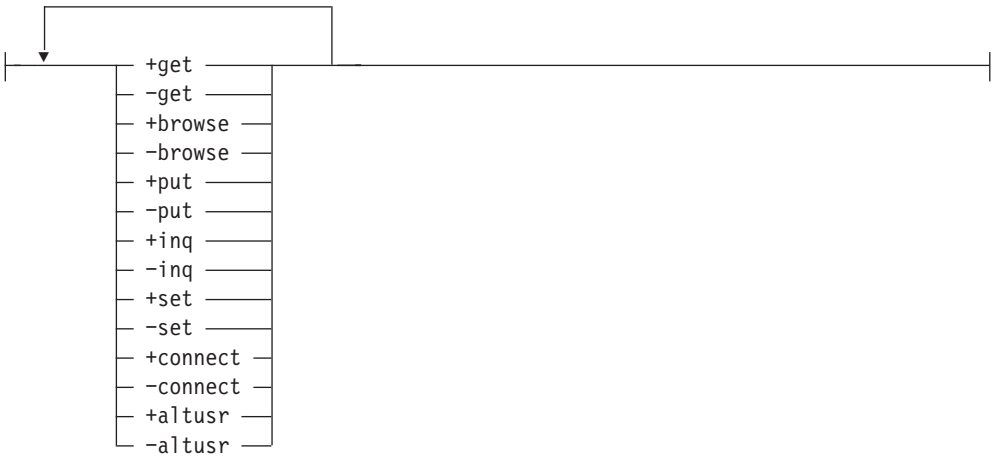
Funktion

Mit dem Befehl **setmqaut** ändern Sie die Berechtigungen für ein Objekt oder eine Objektklasse. Berechtigungen können beliebig vielen Principals oder Gruppen erteilt bzw. ihnen entzogen werden.

Syntax



MQI-Berechtigungen:



Kontextberechtigungen:

Verwaltungsberechtigungen:

Generische Berechtigungen:

Beschreibung

Mit diesem Befehl können Sie sowohl eine Berechtigung *setzen*, d. h. einer Benutzergruppe oder einem Principal Berechtigung zum Ausführen einer Operation erteilen, als auch eine Berechtigung *zurücksetzen*, d. h. die Berechtigung zum Ausführen einer Operation entziehen. Sie müssen zum einen die Benutzergruppen und Principals, für die die Berechtigungen gelten, und zum anderen den WS-Manager, die Objektart und den Objektnamen des Objekts angeben. Sie können beliebig viele Gruppen und Principals angeben.

Achtung: Wenn Sie einem Principal eine Berechtigungsgruppe zuordnen, gelten dieselben Berechtigungen für alle Principals in derselben primären Gruppe.

Die zu erteilenden Berechtigungen können in folgende Kategorien eingeteilt werden:

- Berechtigungen für das Ausgeben von MQI-Aufrufen

- Berechtigungen für MQI-Kontext
- Berechtigungen für das Ausgeben von Befehlen für Verwaltungs-Tasks
- generische Berechtigungen

Jede Berechtigung, die geändert werden soll, wird in einer Berechtigungsliste als Teil des Befehls angegeben. Jeder Eintrag in der Liste besteht aus einer Zeichenfolge mit einem Pluszeichen ('+') oder einem Minuszeichen ('−') als Präfix. Wenn Sie zum Beispiel den Eintrag '+put' in die Liste aufnehmen, erteilen Sie die Berechtigung zum Ausgeben von MQPUT-Aufrufen für eine Warteschlange. Wenn Sie dagegen den Eintrag '-put' in die Berechtigungsliste aufnehmen, entziehen Sie die Berechtigung zum Ausgeben von MQPUT-Aufrufen.

Berechtigungen können in jeder beliebigen Reihenfolge angegeben werden, solange sie sich nicht widersprechen. Wenn Sie zum Beispiel gleichzeitig `allmqi` und `set` angeben, führt dies zu einem Widerspruch.

Sie können in einem einzigen Befehl so viele Gruppen oder Berechtigungen angeben wie erforderlich.

Wenn eine Benutzer-ID Mitglied mehrerer Gruppen ist, entsprechen die gültigen Berechtigungen der Zusammenfassung der Berechtigungen aller Gruppen, denen die Benutzer-ID angehört.

Erforderliche Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers des Objekts an, für das die Berechtigungen geändert werden sollen. Der Name darf maximal 48 Zeichen lang sein.

-n *Objektname*
Gibt den Namen des Objekts an, für das die Berechtigungen geändert werden sollen.

Dieser Parameter ist erforderlich, ***außer*** wenn es sich bei dem Objekt um den WS-Manager selbst handelt. Sie müssen den Namen des WS-Managers, der Warteschlange oder der Prozessdefinition angeben, dürfen aber keinen generischen Namen verwenden.

-t *Objektart*
Gibt die Art des Objekts an, für das die Berechtigungen geändert werden sollen.

Gültige Werte:
- **q** oder **queue**
- **prcs** oder **process**
- **qmgr**

Optionale Parameter

-p *Principal-Name*
Gibt den Namen des Principals an, für den die Berechtigungen geändert werden sollen.

Sie müssen mindestens einen Principal oder eine Gruppe angeben.

-g *Gruppenname*
Gibt den Namen der Berechtigungs-ID der Benutzergruppe an, deren Berechtigungen geändert werden sollen. Sie können mehrere Berechtigungs-IDs angeben, vor jedem Namen muss jedoch die Option -g stehen.

-s *Servicekomponente*
Dieser Parameter gilt nur, wenn Sie installierbare Berechtigungsservices verwenden, andernfalls wird er ignoriert.

Wenn installierbare Berechtigungsservices unterstützt werden, gibt dieser Parameter den Namen des Berechtigungsservices an, für den die Berechtigungen gelten. Dieser Parameter ist optional; wenn er nicht angegeben wird, gilt die Aktualisierung für die erste installierbare Komponente für den Service.

Berechtigungen
Gibt die Berechtigungen an, die erteilt oder entzogen werden sollen. Vor jedem Eintrag in der Liste steht entweder ein Pluszeichen ('+'), d. h., die Berechtigung wird erteilt, oder ein Minuszeichen ('–'), d. h., die Berechtigung wird entzogen. Geben Sie in der Liste zum Beispiel '+put' ein, um die Berechtigung zum Ausgeben eines MQPUT-Aufrufs über die MQI zu erteilen. Geben Sie '-put' ein, um die Berechtigung zum Ausgeben eines MQPUT-Aufrufs zu entziehen.

Tabelle 17 zeigt die Berechtigungen, die für die unterschiedlichen Objektarten erteilt werden können.

Tabelle 17. Berechtigungen für verschiedene Objektarten erteilen

Berechtigung	Warteschlange	Prozess	WS-Manager	Namensliste
all	✓	✓	✓	✓
alladm	✓	✓	✓	✓
allmqi	✓	✓	✓	✓
altusr			✓	
browse	✓			
chg	✓	✓	✓	✓
clr	✓			
connect			✓	
crt	✓	✓	✓	✓
dlt	✓	✓	✓	✓
dsp	✓	✓	✓	✓
get	✓			
inq	✓	✓	✓	✓
passall	✓			
passid	✓			
put	✓			
set	✓	✓	✓	
setall	✓		✓	
setid	✓		✓	

Berechtigungen für MQI-Aufrufe

altusr Verwenden einer alternativen Benutzer-ID in einer Nachricht.

Weitere Informationen zu alternativen Benutzer-IDs finden Sie im Handbuch *MQSeries Application Programming Guide*.

browse
Abrufen einer Nachricht aus einer Warteschlange durch Ausgeben eines MQGET-Aufrufs mit der Option BROWSE.

connect
Verbinden der Anwendung mit dem angegebenen WS-Manager durch Ausgeben eines MQCONN-Aufrufs.

get Abrufen einer Nachricht aus einer Warteschlange durch Ausgeben eines MQGET-Aufrufs.

inq Abfragen einer angegebenen Warteschlange durch Ausgeben eines MQINQ-Aufrufs.

put Einreihen einer Nachricht in eine angegebene Warteschlange durch Ausgeben eines MQPUT-Aufrufs.

set Setzen von Attributen für eine Warteschlange über die MQI durch Ausgeben eines MQSET-Aufrufs.

Anmerkung: Wenn Sie eine Warteschlange für mehrere Optionen öffnen, müssen Sie über die Berechtigung für jede einzelne verfügen.

Berechtigungen für Kontext

passall
Übergeben des gesamten Kontextes an die angegebene Warteschlange. Alle Kontextfelder werden aus der ursprünglichen Anforderung kopiert.

passid Übergeben des Identitätskontextes an die angegebene Warteschlange. Der Identitätskontext ist derselbe wie der in der Anforderung.

setall Setzen des gesamten Kontextes für die angegebene Warteschlange. Dies wird von bestimmten Systemdienstprogrammen verwendet.

setid Setzen des Identitätskontextes für die angegebene Warteschlange. Dies wird von bestimmten Systemdienstprogrammen verwendet.

Berechtigungen für Befehle

chg Ändern der Attribute des angegebenen Objekts.

clr Inhalt der angegebenen Warteschlange löschen (nur PCF-Befehl 'Clear queue').

crt Erstellen von Objekten der angegebenen Objektart.

dlt Löschen des angegebenen Objekts.

dsp Anzeigen der Attribute des angegebenen Objekts.

Berechtigungen für generische Operationen

all Verwenden aller auf das Objekt anwendbaren Operationen.

alladm
Ausführen aller auf das Objekt anwendbaren Verwaltungsoperationen.

allmqi Verwenden aller auf das Objekt anwendbaren MQI-Aufrufe.

Rückkehrcodes

0 Befehl wurde normal beendet
36 Ungültige Argumente übergeben
40 WS-Manager ist nicht verfügbar
49 WS-Manager wird gestoppt
69 Kein Speicher verfügbar
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name
133 Unbekannter Objektname
145 Unerwarteter Objektname
146 Objektname fehlt
147 Objektart fehlt
148 Ungültige Objektart
149 Name der Definitionseinheit fehlt
150 Berechtigungsspezifikation fehlt
151 Ungültige Berechtigungsspezifikation

Beispiele

1. Dieses Beispiel zeigt einen Befehl, in dem als Objekt, für das Berechtigungen erteilt werden, die Warteschlange `orange.queue` auf dem WS-Manager `saturn.queue.manager` angegeben ist.

```
setmqaut -m "saturn.queue.manager" -n "orange.queue" -t queue -g "tango" +inq +alladm
```

   Die Berechtigungen werden an die Benutzergruppe `tango` erteilt, wobei die zugehörige Berechtigungsliste die folgenden Festlegungen für die Benutzergruppe `tango` enthält:
   - Sie erhält die Berechtigung zum Ausgeben von MQINQ-Aufrufen.
   - Sie erhält die Berechtigung zum Ausführen aller Verwaltungsoperationen für das Objekt.

2. In diesem Befehl enthält die Berechtigungsliste folgende Festlegungen für die Benutzergruppe `foxy`:
   - Sie erhält keine Berechtigung zum Ausgeben von Aufrufen über die MQI für die angegebene Warteschlange.
   - Sie erhält Berechtigung zum Ausführen aller Verwaltungsoperationen für die angegebene Warteschlange.

```
setmqaut -m "saturn.queue.manager" -n "orange.queue" -t queue -g "foxy" -allmqi +alladm
```

3. In diesem Befehl gibt die Berechtigungsliste an, dass die Benutzergruppe `waltz` über die Berechtigung zum Erstellen und Löschen des WS-Managers `saturn.queue.manager` verfügt:

```
setmqaut -m "saturn.queue.manager" -t qmgr -g "waltz" +crt +dlt
```

Zugehörige Befehle

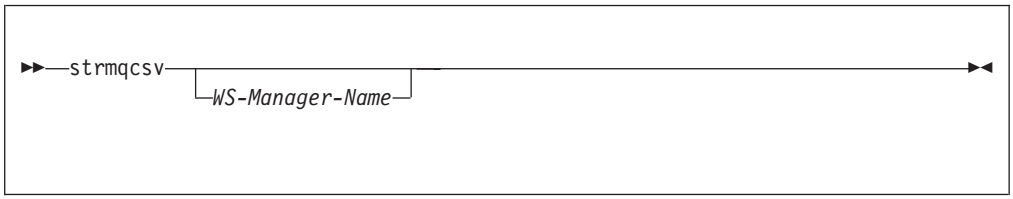
dspmqaut Berechtigung anzeigen

strmqcsv (Befehlsserver starten)

Funktion

Mit dem Befehl **strmqcsv** starten Sie den Befehlsserver für den angegebenen WS-Manager. Dadurch ist es MQSeries möglich, Befehle an die Befehlswarteschlange zu senden.

Syntax



Optionale Parameter

WS-Manager-Name
Gibt den Namen des WS-Managers an, für den der Befehlsserver gestartet werden soll.

Rückkehrcodes

0 Befehl wurde normal beendet.
10 Befehl wurde mit unerwarteten Ergebnissen beendet.
20 Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiele

Der folgende Befehl startet einen Befehlsserver für den WS-Manager `Erde`:

```
strmqcsv "Erde"
```

Zugehörige Befehle

endmqcsv Befehlsserver beenden

dspmqcsv Status eines Befehlsservers anzeigen

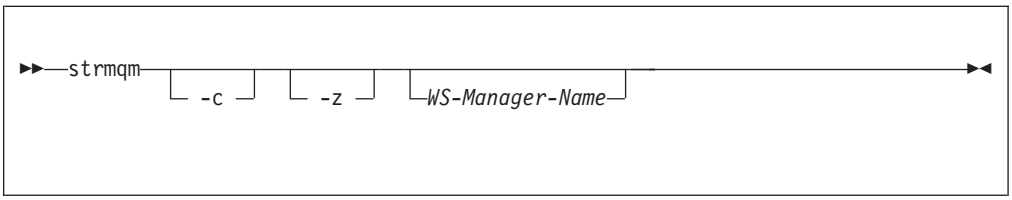
strmqm (WS-Manager starten)

Funktion

Mit dem Befehl **strmqm** starten Sie einen lokalen WS-Manager.

Anmerkung: Bevor der Befehl **strmqm** oder ein anderer Steuerbefehl verwendet wird, muss nach dem letzten Neustart ein Mal der Befehl `mqs_startup` ausgeführt worden sein, bevor der WS-Manager gestartet und erfolgreich ausgeführt werden kann.

Syntax



Optionale Parameter

-c Startet den WS-Manager, definiert die Standard- und Systemobjekte erneut und stoppt dann den WS-Manager. (Die Standard- und Systemobjekte für einen WS-Manager werden ursprünglich mit dem Befehl **crtmqm** erstellt.) Alle vorhandenen System- und Standardobjekte, die dem WS-Manager zugeordnet sind, werden ersetzt, wenn Sie diese Option angeben.

WS-Manager-Name
Gibt den Namen eines lokalen WS-Managers an, der gestartet werden soll. Wird kein Name angegeben, wird der Standard-WS-Manager gestartet.

-z Diese Option unterdrückt Fehlernachrichten.

Sie wird in MQSeries verwendet, um unerwünschte Fehlernachrichten zu unterdrücken. Da bei Angabe dieser Option Informationen verloren gehen können, sollte sie nicht beim Eingeben von Befehlen in einer Befehlszeile verwendet werden.

Rückkehrcodes

0 WS-Manager gestartet
3 WS-Manager wird erstellt
5 WS-Manager aktiv
16 WS-Manager nicht vorhanden
23 Protokoll nicht verfügbar
49 WS-Manager wird gestoppt
69 Kein Speicher verfügbar
71 Unerwarteter Fehler
72 Falscher WS-Manager-Name
100 Ungültiges Protokollverzeichnis

Beispiele

Der folgende Befehl startet den WS-Manager `Konto`:

```
strmqm "Konto"
```

Zugehörige Befehle

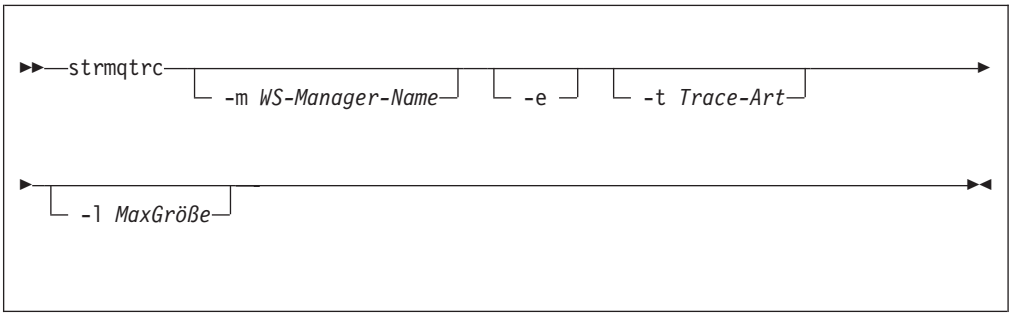
crtmqm WS-Manager erstellen
dltmqm WS-Manager löschen
endmqm WS-Manager beenden

strmqtrc (MQSeries-Trace starten)

Funktion

Mit dem Befehl **strmqtrc** aktivieren Sie die Trace-Einrichtung. Dieser Befehl kann unabhängig davon ausgeführt werden, ob die Trace-Einrichtung aktiviert ist oder nicht. Wenn die Trace-Einrichtung bereits aktiviert ist, werden die aktuellen Trace-Optionen durch diejenigen überschrieben, die mit dem zuletzt aufgerufenen Befehl angegeben wurden.

Syntax



Beschreibung

Es können verschiedene Trace-Informationen angefordert werden. Für jeden von Ihnen angegebenen 'Trace-Art'-Wert für die Trace-Ausgabe, einschließlich '-t all', können Sie weitere '-t'-Optionen mit zusätzlichen Parametern angeben, damit für die betreffende Trace-Art standardmäßig nur bestimmte Detailinformationen generiert werden.

Beispiele für Trace-Daten, die mit diesem Befehl generiert wurden, finden Sie unter „MQSeries-Trace verwenden" auf Seite 216.

Optionale Parameter

-m *WS-Manager-Name*
Gibt den Namen des WS-Managers an, für den der Trace durchgeführt werden soll.

Ein WS-Manager-Name und die Option -m können in demselben Befehl wie die Option -e angegeben werden. Wenn auf eine angegebene Definitionseinheit, für die ein Trace durchgeführt wird, mehrere Spezifikationen angewendet werden, umfasst der tatsächlich ausgeführte Trace alle angegebenen Optionen.

Die Option -m und den WS-Manager-Namen wegzulassen, führt zu einem Fehler, außer wenn die Option -e angegeben ist.

-e Wenn diese Option angegeben wird, wird ein Trace in der Startphase durchgeführt. Demzufolge ist es möglich, für die Erstellung oder den Start eines WS-Managers einen Trace durchzuführen. Das heißt, es werden Trace-Informationen geschrieben, bevor die Prozesse wissen, welcher MQSeries-Komponente sie zugeordnet sind. Jeder Prozess, der einer Komponente eines WS-Managers zugeordnet ist, führt während seiner eigenen Startphase einen Trace durch, wenn diese Option angegeben wird. Ist sie nicht angegeben, gilt die Standardeinstellung, dass kein Trace in der Startphase durchgeführt wird.

-t *Trace-Art*

Definiert, für welche Punkte während der Verarbeitung ein Trace durchgeführt wird. Wenn diese Option nicht angegeben wird, sind alle Trace-Punkte aktiviert, und ein vollständiger Trace wird durchgeführt.

Alternativ können Optionen (eine oder mehrere) aus der folgenden Liste angegeben werden.

Anmerkung: Wenn mehrere Trace-Arten angegeben werden, ***muss*** jede über eine eigene Option -t verfügen. Es können beliebig viele '-t'-Optionen angegeben werden, solange jeder eine gültige Trace-Art zugeordnet ist.

Es ist zulässig, für mehrere '-t'-Optionen dieselbe Trace-Art anzugeben.

all Für jeden Trace-Punkt im System werden Daten ausgegeben. Dies ist die Standardeinstellung, wenn die Option -t nicht angegeben wird.

api Für Trace-Punkte, die der MQI und Hauptkomponenten des WS-Managers zugeordnet sind, werden Daten ausgegeben.

comms Für Trace-Punkte, die Datenflüssen in Kommunikationsnetzen zugeordnet sind, werden Daten ausgegeben.

csflows Für Trace-Punkte, die Verarbeitungsabläufen in allgemeinen Services zugeordnet sind, werden Daten ausgegeben.

lqmflows Für Trace-Punkte, die Verarbeitungsabläufen im lokalen WS-Manager zugeordnet sind, werden Daten ausgegeben.

remoteflows Für Trace-Punkte, die Verarbeitungsabläufen in der DFV-Komponente zugeordnet sind, werden Daten ausgegeben.

otherflows Für Trace-Punkte, die Verarbeitungsabläufen in anderen Komponenten zugeordnet sind, werden Daten ausgegeben.

csdata Für Trace-Punkte, die internen Datenpuffern in allgemeinen Services zugeordnet sind, werden Daten ausgegeben.

lqmdata Für Trace-Punkte, die internen Datenpuffern im lokalen WS-Manager zugeordnet sind, werden Daten ausgegeben.

remotedata Für Trace-Punkte, die internen Datenpuffern in der DFV-Komponente zugeordnet sind, werden Daten ausgegeben.

otherdata Für Trace-Punkte, die internen Datenpuffern in anderen Komponenten zugeordnet sind, werden Daten ausgegeben.

versiondata Für Trace-Punkte, die der aktiven Version von MQSeries zugeordnet sind, werden Daten ausgegeben.

commentary Für Trace-Punkte, die Kommentaren in den MQSeries-Komponenten zugeordnet sind, werden Daten ausgegeben.

-l *MaxGröße*

Der Wert *MaxGröße* gibt die maximale Größe einer Trace-Datei (AMQnnnn.TRC) in Millionen Bytes an. Wenn Sie zum Beispiel MaxGröße = 1 angeben, ist die Größe der Trace-Datei auf 1 Million Bytes begrenzt.

Wenn eine Trace-Datei die angegebene maximale Größe erreicht, wird sie von AQnnnn.TRC in AMQnnnn.TRS umbenannt und dann eine neue Datei AMQnnnn.TRC gestartet. Alle Trace-Dateien werden erneut gestartet, wenn die maximale Größe erreicht wird. Wenn eine ältere Version der Datei AMQnnnn-.TRS vorhanden ist, wird sie gelöscht.

Rückkehrcodes

AMQ7024 Diese Nachricht wird ausgegeben, wenn im Befehl ungültige Argumente angegeben wurden.

AMQ8304 Es ist bereits die maximale Anzahl von neun Traces aktiv.

Beispiele

Dieser Befehl aktiviert den Trace für Daten von allgemeinen Services und vom lokalen WS-Mmanager für den WS-Manager `QM1`.

```
strmqtrc -m QM1 -t csdata -t lqmdata
```

Zugehörige Befehle

dspmqtrc Formatierte Trace-Ausgabe anzeigen

endmqtrc MQSeries-Trace beenden

Teil 3. Anhänge und Schlussteil

Anhang A. MQSeries for Compaq OpenVMS - Übersicht

Programm und Teilenummer

- 5724–A38 MQSeries for Compaq OpenVMS, Alpha Version 5 Release 1, Teilenummer 0790997.

Hardwarevoraussetzungen

Als MQSeries-Server kann eine beliebige Compaq Alpha-Maschine mit einem Mindestplattenspeicherplatz von 128 MB verwendet werden.

Softwarevoraussetzungen

Soweit nicht anders angegeben, gelten in Compaq OpenVMS-Umgebungen für Server und Clients dieselben Voraussetzungen.

Es wird jeweils die Versionsstufe der Software angegeben, die mindestens erforderlich:

- Compaq OpenVMS Alpha Version 7.2–1.

Konnektivität

MQSeries for Compaq OpenVMS unterstützt die folgenden Netzprotokolle sowie die folgende Übertragungshardware:

Netzprotokolle:

- SNA LU6.2
- TCP/IP
- DECnet Phase V

Außerdem werden alle Übertragungshardwareprodukte unterstützt, die DECnet, TCP/IP oder DIGITAL DECnet/SNA Gateway unterstützen.

Für DECnet-Konnektivität:

- DECnetPLUS Version 7.1 für OpenVMS Version 7.2–1

Für TCP/IP-Konnektivität:

- DIGITAL TCP/IP Services für OpenVMS Alpha Version 5.0.a und 5.1
- Process Software's TCPWare Version 5.4
- Process Software's Multinet Version 4.3

Für SNA-Konnektivität: SNA APPC LU6.2-Software und die entsprechende Lizenz müssen installiert sein. Der Zugriff auf ein entsprechend konfiguriertes SNA-Gateway muss vorhanden sein.

- DECnet SNA Gateway ST Version 1.3
- DECnet SNA LU6.2 API Version 2.4

Sicherheit

MQSeries for Compaq OpenVMS verwendet die Sicherheitseinrichten des OAM (Object Authority Manager) für MQSeries for Compaq OpenVMS.

Alle MQSeries-Ressourcen arbeiten mit der VMS-Berechtigungs-ID MQM. Diese Berechtigungs-ID wird bei der MQSeries-Installation erstellt und muss mit diesem Ressourcenattribut allen Benutzern zuerteilt werden, für die die Steuerung von MQSeries-Ressourcen möglich sein muss.

Verwaltungsfunktionen

Die MQSeries-Verwaltung erfolgt über:

- Die Befehlszeilenschnittstelle **runmqsc**.

Kompatibilität

Die MQI für MQSeries for Compaq OpenVMS Alpha V5.1 ist mit vorhandenen Anwendungen, die mit Version 2.2.1.1 arbeiten, kompatibel.

Unterstützte Compiler

Programme können mit den Programmiersprachen C, C++, COBOL oder Java erstellt werden.

- Für C-Programme kann der DEC C-Compiler verwendet werden.
- Für C++-Programme kann der DEC C++-Compiler verwendet werden.
- Für COBOL-Programme kann der DEC COBOL-Compiler verwendet werden.
- Für Java-Programme kann der Java-Compiler verwendet werden.

Sprachenauswahl

Eine Nachrichtentextdatei, die im Lieferumfang enthalten ist, ist mit dem 7–Bit-Zeichensatz codiert, der auch vom OpenVMS-Betriebssystem verwendet wird.

CCSID-Auswahl

MQSeries for Compaq OpenVMS ermöglicht die Angabe der CCSID bei der Erstellung der WS-Managerinstanz. Standardwert für die CCSID des WS-Managers ist 819. MQSeries for Compaq OpenVMS unterstützt die Zeichensatzkonvertierung in die für den WS-Manager konfigurierte CCSID. Informationen darüber, welche CCSIDs (einschließlich der, die das Euro-Zeichen unterstützen) für WS-Manager in MQSeries for Compaq OpenVMS angegeben werden können, finden Sie im Handbuch *MQSeries Application Programming Reference*.

Anhang B. Systemstandardwerte

Bei Erstellung eines WS-Managers mit dem Steuerbefehl **crtmqm** werden die Systemobjekte sowie die Standardobjekte automatisch erstellt.

- Bei den Systemobjekten handelt es sich um die MQSeries-Objekte, die für den Betrieb eines WS-Managers bzw. Kanals erforderlich sind.
- Über die Standardobjekte sind alle Attribute eines Objekts definiert. Bei Erstellung eines Objekts wie beispielsweise eine lokale Warteschlange werden für alle Attribute, für die Sie keinen Wert angeben, die Werte aus dem Standardobjekt übernommen.

Tabelle 18. System- und Standardobjekte für Warteschlangen

Objektname	**Beschreibung**
SYSTEM.DEFAULT.ALIAS.QUEUE	Standardaliaswarteschlange.
SYSTEM.DEFAULT.LOCAL.QUEUE	Lokale Standardwarteschlange.
SYSTEM.DEFAULT.MODEL.QUEUE	Standardmodellwarteschlange.
SYSTEM.DEFAULT.REMOTE.QUEUE	Ferne Standardwarteschlange.
SYSTEM.DEAD.LETTER.QUEUE	Beispielwarteschlange für nicht zustellbare Nachrichten.
SYSTEM.DEFAULT.INITIATION.QUEUE	Standardstartwarteschlange.
SYSTEM.CICS.INITIATION.QUEUE	CICS®-Standardstartwarteschlange.
SYSTEM.ADMIN.COMMAND.QUEUE	Warteschlange für Verwaltungsbefehle. Wird für ferne MQSC- und PCF-Befehle verwendet.
SYSTEM.MQSC.REPLY.QUEUE	MQSC-Warteschlange für Antwortnachrichten. Hierbei handelt es sich um eine Modellwarteschlange, die eine temporäre dynamische Warteschlange für Antworten auf ferne MQSC-Befehle erstellt.
SYSTEM.ADMIN.QMGR.EVENT	Ereigniswarteschlange für WS-Managerereignisse.
SYSTEM.ADMIN.PERFM.EVENT	Ereigniswarteschlange für leistungsspezifische Ereignisse.
SYSTEM.ADMIN.CHANNEL.EVENT	Ereigniswarteschlange für Kanalereignisse.
SYSTEM.CHANNEL.INITQ	Kanalstartwarteschlange.
SYSTEM.CHANNEL.SYNCQ	Die Warteschlange, die die Synchronisationsdaten für Kanäle enthält.
SYSTEM.CLUSTER.COMMAND.QUEUE	Die Warteschlange, die für die Übertragung von Nachrichten an den Repository-WS-Manager verwendet wird.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	Die Warteschlange, in der alle Repository-Daten gespeichert werden.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	Die Übertragungswarteschlange für alle an Cluster gerichteten Nachrichten.

Standardwerte

Tabelle 19. System- und Standardobjekte für Kanäle

Objektname	Beschreibung
SYSTEM.DEF.SENDER	Standardsenderkanal.
SYSTEM.DEF.SERVER	Standardserverkanal.
SYSTEM.DEF.RECEIVER	Standardempfängerkanal.
SYSTEM.DEF.REQUESTER	Standard-Requester-Kanal.
SYSTEM.DEF.SVRCONN	Standardkanal für Serververbindungen.
SYSTEM.DEF.CLNTCONN	Standardkanal für Client-Verbindungen.
SYSTEM.AUTO.RECEIVER	Dynamischer Empfängerkanal.
SYSTEM.AUTO.SVRCONN	Dynamischer Kanal für Serververbindungen.
SYSTEM.DEF.CLUSRCVR	Standardempfängerkanal für Cluster, über den Standardwerte für alle Attribute zur Verfügung gestellt werden, für die bei Erstellung eines CLUSRCVR-Kanals in einem WS-Manager innerhalb eines Clusters keine Werte angegeben wurden.
SYSTEM.DEF.CLUSSDR	Standardsenderkanal für Cluster, über den Standardwerte für all Attribute zur Verfügung gestellt werden, für die bei Erstellung eines CLUSSDR-Kanals in einem WS-Managers innerhalb eines Clusters keine Werte angegeben wurden.

Tabelle 20. System- und Standardobjekte für Namenslisten

Objektname	Beschreibung
SYSTEM.DEFAULT.NAMELIST	Standardnamensliste.

Tabelle 21. System- und Standardobjekte für Prozesse

Objektname	Beschreibung
SYSTEM.DEFAULT.PROCESS	Standardprozessdefinition.

Anhang C. Verzeichnisstruktur

Tabelle Abb. 26 enthält eine allgemeine Übersicht über die Daten- und Protokollverzeichnisstruktur für einen WS-Manager. Die Verzeichnisse, die Sie hier sehen, entsprechen der Standardinstallation. Wenn Sie Änderungen vornehmen, ändern sich die Adressen der Datei- und Verzeichnispfade entsprechend.

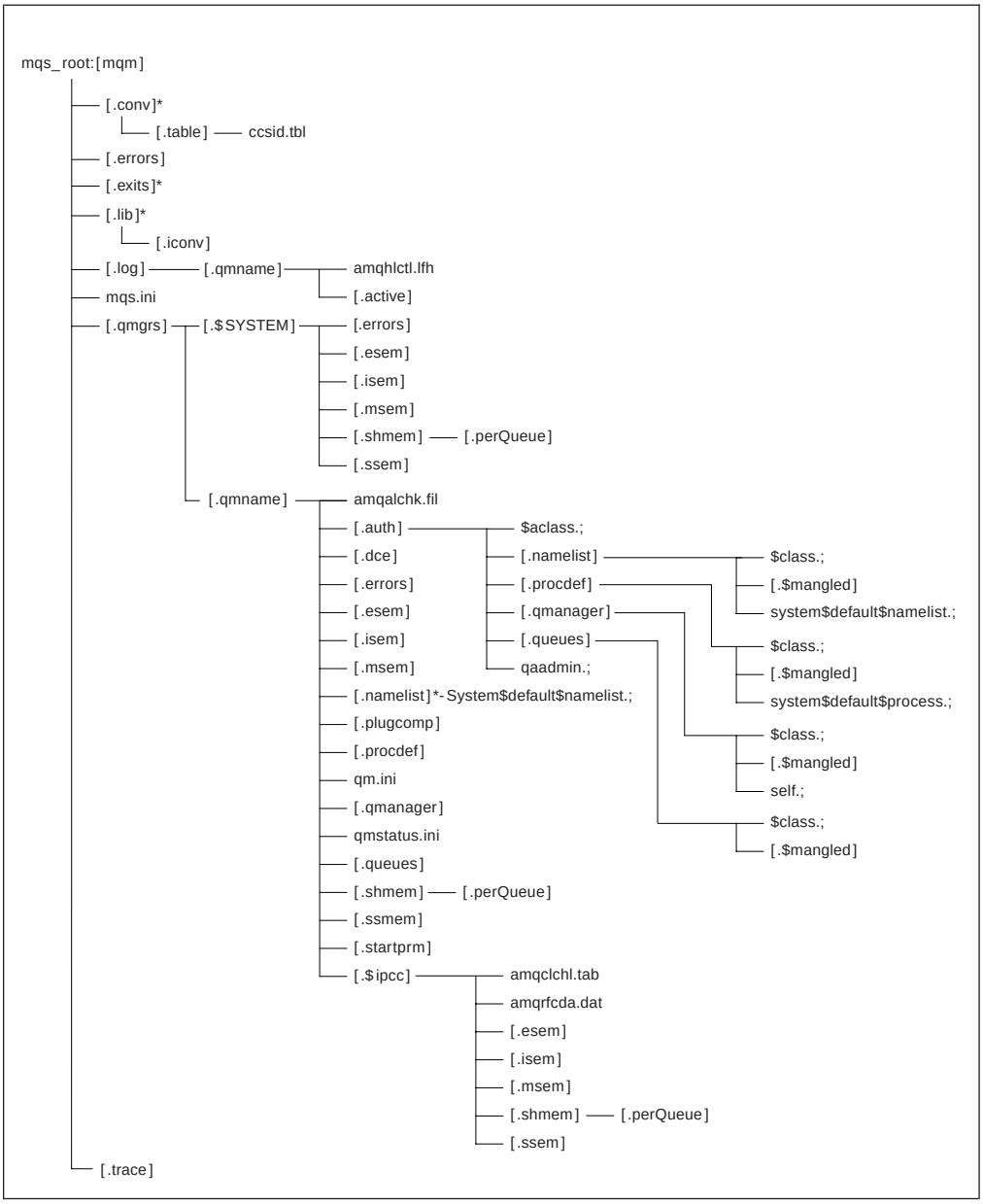


Abbildung 26. Standardverzeichnisstruktur nach dem Start eines WS-Managers

Die Struktur in Abb. 26 ist in MQSeries typisch für einen WS-Manager, der bereits seit einiger Zeit in Betrieb ist. Die tatsächliche Struktur hängt von den Operationen ab, die im WS-Manager ausgeführt wurden.

Verzeichnisse und Dateien in MQS_ROOT:[MQM]

Die folgenden Verzeichnisse und Dateien befinden sich standardmäßig im Verzeichnis MQS_ROOT:[MQM]:

.conv Dieses Verzeichnis enthält alle Dateien, die für die Datenkonvertierung verwendet werden.

.table Dieses Verzeichnis enthält die Datei ccsid.tbl.

.errors Dieses Verzeichnis enthält die Bedienernachrichtendateien, von der ältesten bis zur aktuellen Version:
AMQERR01.LOG
AMQERR02.LOG
AMQERR03.LOG

.exits Ein leeres Verzeichnis für benutzerdefiniertes Exits.

.lib Dieses Verzeichnis enthält das Unterverzeichnis .iconv, das wiederum alle Konvertierungstabellen für die codierten Zeichensätze enthält.

.iconv Ein Verzeichnis mit den Konvertierungstabellen für codierte Zeichensätze (wie beispielsweise Tabelle `002501B5.TBL` bis `44B031A8.TBL`).

.log Wenn Sie MQSeries installiert sowie einen WS-Manager erstellt und gestartet haben und dieser anschließend einige Zeit aktiv war, enthält dieses Verzeichnis das folgende Unterverzeichnis sowie die folgenden Dateien:

amqhlctl.lfh
Protokollsteuerdatei.

active Dieses Verzeichnis enthält die Protokolldateien, die wie folgt numeriert sind:
S0000000.LOG
S0000001.LOG
S0000002.LOG
... usw.

mqs.ini
MQSeries-Konfigurationsdatei.

.qmgrs
Dieses Verzeichnis enthält für jeden WS-Manager die Unterverzeichnisse `.$system` und `.qmname`. Das Verzeichnis `.$system` enthält die intern von MQSeries verwendeten Verzeichnisse und Dateien. Weitere Informationen zum Unterverzeichnis `.qmname` finden Sie unter „Verzeichnisse und Dateien im Unterverzeichnis MQS_ROOT:[MQM.QMGRS.QMNAME]" auf Seite 331.

.trace Dieses Verzeichnis enthält die bei Ausführung des Befehls **strmqtrc** erstellten Trace-Dateien.

Verzeichnisse und Dateien im Unterverzeichnis MQS_ROOT:[MQM.QMGRS.QMNAME]

Die folgenden Verzeichnisse und Dateien befinden sich standardmäßig im Verzeichnis MQS_ROOT:[MQM.QMGRS.QMNAME]. Der WS-Managername (.QMNAME) wird für jeden WS-Manager erstellt, der auf dem System erstellt wurde und dort aktiv ist.

amqalchk.fil
Eine Prüfpunktdatei, die Daten zum letzten Prüfpunkt enthält.

.auth Dieses Verzeichnis enthält berechtigungsspezifische Unterverzeichnisse und Dateien.

$aclass.;
Diese Datei enthält die Berechtigungszeilengruppen für alle Klassen.

.namelist
Dieses Verzeichnis enthält je eine Datei für jede Namensliste. Jede Datei enthält die Berechtigungszeilengruppen für die jeweilige Namensliste.

$class.;
Diese Datei enthält die Berechtigungszeilengruppen für die Namenslistenklasse.

.$mangled
Namenslistennamen, die ungültige OpenVMS-Zeichen enthalten, werden automatisch in gültige OpenVMS-Namen konvertiert. Diese Datei enthält die gültigen OpenVMS-Namen (siehe „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21).

system$default$namelist
Diese Datei enthält die Berechtigungszeilengruppen für die Systemstandardnamensliste.

.procdef
Jeder MQSeries-Prozessdefinition entspricht eine Datei in diesem Verzeichnis.

$class.;
Diese Datei enthält die Berechtigungszeilengruppen für die Prozessdefinitionsklasse.

.$mangled
Prozessdefinitionsnamen, die ungültige OpenVMS-Zeichen enthalten, werden automatisch in gültige OpenVMS-Namen konvertiert. Diese Datei enthält die gültigen OpenVMS-Namen (siehe „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21.)

.system$default$process.;
Diese Datei enthält Berechtigungszeilengruppen für die Systemstandardprozesse.

.qmanager
Dieses Verzeichnis enthält je eine Datei für jeden WS-Manager. Jede Datei enthält die Berechtigungszeilengruppen für den jeweiligen WS-Manager.

$class.;
Diese Datei enthält die Berechtigungszeilengruppen für die WS-Managerklasse.

.$mangled
Namen von WS-Managerdefinitionen, die ungültige OpenVMS-Zeichen enthalten, werden automatisch in gültige OpenVMS-Namen konvertiert. Diese Datei enthält die gültigen OpenVMS-Namen (siehe „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21).

self.; Dieses Datei enthält die Berechtigungszeilengruppen für das WS-Managerobjekt.

.queues
Dieses Verzeichnis enthält je eine Datei für jede Warteschlange. Jede Datei enthält die Berechtigungszeilengruppen für die jeweilige Warteschlange.

$CLASS
Diese Datei enthält die Berechtigungszeilengruppen für die Warteschlangenklasse.

.$mangled
Warteschlangennamen, die ungültige OpenVMS-Zeichen enthalten, werden automatisch in gültige OpenVMS-Namen konvertiert. Diese Datei enthält die gültigen OpenVMS-Namen (siehe „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21).

Definitionsdateien für die Warteschlange
Jede Datei entspricht einem der für den WS-Manager vorab definierten Objekte.

system$admin$channel$event.;
system$admin$command$queue.;
system$admin$perfm$event.;
system$admin$qmgr$event.;
system$channel$initq.;
system$channel$syncq.;
system$cics$initiation$queue.;
system$cluster$command$queue.;
system$cluster$repository$queue.;
system$cluster$transmit$queue.;
system$dead$letter$queue.;
system$default$alias$queue.;
system$default$initiation$queue.;
system$default$local$queue.;
system$default$model$queue.;
system$default$remote$queue.;
system$mqsc$reply$queue.;

.qaadmin.;
Eine Datei, die intern für die Berechtigungssteuerung verwendet wird.

.dce Ein leeres Verzeichnis, das für die DCE-Unterstützung reserviert ist.

.errors Dieses Verzeichnis enthält die Bedienernachrichtendateien, von der ältesten bis zur aktuellen Version:
amqerr01.log
amqerr02.log
amqerr03.log

.esem Ein Verzeichnis, das intern verwendete Dateien enthält.

.isem Ein Verzeichnis, das intern verwendete Dateien enthält.

.msem Ein Verzeichnis, das intern verwendete Dateien enthält.

.namelist
Dieses Verzeichnis enthält Namenslisten für jeden WS-Manager.

.plugcomp
Ein leeres Verzeichnis, das für installierbare Services reserviert ist.

.procdef
Jeder MQSeries-Prozessdefinition entspricht eine Datei in diesem Verzeichnis. Die einzelnen Dateinamen entsprechen dem jeweiligen Namen der Prozessdefinition.

qm.ini Datei mit der WS-Managerkonfiguration.

.qmanager
Das WS-Managerobjekt.

qmstatus.ini
Diese Datei enthält Beschreibung des WS-Managerstatus im Textformat.

.queues
Enthält für jede Warteschlange ein Verzeichnis, das wiederum eine einzige Datei, nämlich 'q' enthält.

Der Dateiname entspricht dem Namen der jeweiligen Warteschlange, wobei bestimmte Einschränkungen gelten (siehe „Erläuterungen zu MQSeries-Dateinamen" auf Seite 21).

.shmem

.perQueue
Ein Verzeichnis, das intern verwendete Dateien enthält.

.ssem Ein Verzeichnis, das intern verwendete Dateien enthält.

.startprm
Ein Verzeichnis, das intern verwendete temporäre Dateien enthält.

Verzeichnisstruktur

.$ipcc

amqclchl.tab
Client-Kanaltabellendatei.

amqrfcda.dat
Kanaltabellendatei.

.esem Ein Verzeichnis, das intern verwendete Dateien enthält.

.isem Ein Verzeichnis, das intern verwendete Dateien enthält.

.msem Ein Verzeichnis, das intern verwendete Dateien enthält.

.shmem

.perQueue
Ein Verzeichnis, das intern verwendete Dateien enthält.

.ssem Ein Verzeichnis, das intern verwendete Dateien enthält.

Anhang D. Befehlssätze - Übersicht

Die folgenden Tabellen enthalten eine Gegenüberstellung der Funktionen der verschiedenen verwaltungsspezifischen Befehlssätze.

- „Befehle für die Verwaltung des WS-Managers"
- „Befehle für die Verwaltung des Befehlsservers"
- „Befehle für die Warteschlangenverwaltung" auf Seite 336
- „Befehle für die Prozessverwaltung" auf Seite 336
- „Befehle für die Kanalverwaltung" auf Seite 337
- „Weitere Steuerbefehle" auf Seite 337

Anmerkung: Es sind nur die MQSC-Befehle aufgeführt, die für MQSeries for Compaq OpenVMS gelten.

Befehle für die Verwaltung des WS-Managers

Tabelle 22. Befehle für die Verwaltung des WS-Managers

PCF-Befehl	MQSC-Befehl	Steuerbefehl
Change Queue Manager	ALTER QMGR	–
(Create queue manager)★	–	crtmqm
(Delete queue manager)★	–	dltmqm
Inquire Queue Manager	DISPLAY QMGR	–
(Stop queue manager)★	–	endmqm
Ping Queue Manager	PING QMGR	–
(Start queue manager)★	–	strmqm
Anmerkung: ★ Nicht als PCF-Befehle verfügbar.		

Befehle für die Verwaltung des Befehlsservers

Tabelle 23. Befehle für die Verwaltung des Befehlsservers

Beschreibung	Steuerbefehl
Befehlsserver anzeigen	dspmqcsv
Befehlsserver starten	strmqcsv
Befehlsserver stoppen	endmqcsv
Anmerkung: Funktionen in dieser Gruppe sind nur als Steuerbefehle verfügbar. Es gibt keine entsprechenden MQSC- oder PCF-Befehle in dieser Gruppe.	

Befehle für die Warteschlangenverwaltung

Tabelle 24. Befehle für die Warteschlangenverwaltung

PCF-Befehl	MQSC-Befehl
Change Queue	ALTER QLOCAL ALTER QALIAS ALTER QMODEL ALTER QREMOTE
Clear Queue	CLEAR QUEUE
Copy Queue	DEFINE QLOCAL(x) LIKE(y) DEFINE QALIAS(x) LIKE(y) DEFINE QMODEL(x) LIKE(y) DEFINE QREMOTE(x) LIKE(y)
Create Queue	DEFINE QLOCAL DEFINE QALIAS DEFINE QMODEL DEFINE QREMOTE
Delete Queue	DELETE QLOCAL DELETE QALIAS DELETE QMODEL DELETE QREMOTE
Inquire Queue	DISPLAY QUEUE
Inquire Queue Names	DISPLAY QUEUE
Anmerkung: Es gibt keine entsprechenden Steuerbefehle in dieser Gruppe.	

Befehle für die Prozessverwaltung

Tabelle 25. Befehle für die Prozessverwaltung

PCF-Befehl	MQSC-Befehl
Change Process	ALTER PROCESS
Copy Process	DEFINE PROCESS(x) LIKE(y)
Create Process	DEFINE PROCESS
Delete Process	DELETE PROCESS
Inquire Process	DISPLAY PROCESS
Inquire Process Names	DISPLAY PROCESS
Anmerkung: Es gibt keine entsprechenden Steuerbefehle in dieser Gruppe.	

Befehle für die Kanalverwaltung

Tabelle 26. Befehle für die Kanalverwaltung

PCF-Befehl	MQSC-Befehl	Steuerbefehl
Change Channel	ALTER CHANNEL	–
Copy Channel	DEFINE CHANNEL(x) LIKE(y)	–
Create Channel	DEFINE CHANNEL	–
Delete Channel	DELETE CHANNEL	–
Inquire Channel	DISPLAY CHANNEL	–
Inquire Channel Names	DISPLAY CHANNEL	–
Ping Channel	PING CHANNEL	–
Reset Channel	RESET CHANNEL	–
Resolve Channel	RESOLVE CHANNEL	–
Start Channel	START CHANNEL	runmqchl
Start Channel Initiator	START CHINIT	runmqchi
Start Channel Listener	–	runmqlsr
Stop Channel	STOP CHANNEL	–

Weitere Steuerbefehle

Tabelle 27. Weitere Steuerbefehle

Beschreibung	Steuerbefehl
MQSeries-Konvertierungs-Exit erstellen	crtmqcvx
Berechtigung anzeigen	dspmqaut
Von Objekten verwendete Dateien anzeigen	dspmqfls
Formatierte MQSeries-Trace-Ausgabe anzeigen	dspmqtrc
MQSeries-Trace beenden	endmqtrc
Überbrückungsgruppe verwalten	failover
Datenträger-Image aufzeichnen	rcdmqimg
Datenträgerobjekt erneut erstellen	rcrmqobj
MQSeries-Transaktionen auflösen	rsvmqtrn
MQSC-Befehle ausführen	runmqsc
Auslösemonitor ausführen	runmqtrm
Client-Auslösemonitor ausführen	runmqtmc
Berechtigung setzen oder zurücksetzen	setmqaut
Überbrückungsmonitor starten	runmqfm
MQSeries-Trace starten	strmqtrc
Anmerkung: Funktionen in dieser Gruppe sind nur als Steuerbefehle verfügbar. Es gibt keine direkten PCF- oder MQSC-Entsprechungen.	

Befehlssätze - Übersicht

Anhang E. MQI-Beispielprogramme und MQSC-Beispieldateien

In MQSeries for Compaq OpenVMS stehen eine Reihe kurzer MQI-Beispielprogramme und MQSC-Befehlsdateien zur Verfügung, anhand derer Sie den Umgang mit diesen Programmen und Dateien üben können. Diese Beispielprogramme und -dateien sind in den folgenden Abschnitten erläutert:

- „MQSC-Beispielbefehlsdateien"
- „C- und COBOL-Beispielprogramme"
- „Verschiedene Tools" auf Seite 340

MQSC-Beispielbefehlsdateien

In Tabelle 28 ist die MQSC-Beispielbefehlsdatei aufgeführt. Dabei handelt es sich einfach um eine ASCII-Textdatei, die MQSC-Befehle enthält. Durch Ausführung des Befehls **runmqsc** für diese Datei werden die in der Datei angegebenen Objekte erstellt. Siehe hierzu „Standardmäßig verfügbare MQSC-Befehlsdateien ausführen" auf Seite 43.

Diese Dateien befinden sich standardmäßig im Verzeichnis `MQS_EXAMPLES:`.

Tabelle 28. MQSC-Befehlsdatei

Dateiname	Funktion
AMQSCOS0.TST	Erstellt eine Gruppe von MQI-Objekten, die für die C- und COBOL-Beispielprogramme verwendet werden können.

C- und COBOL-Beispielprogramme

In Tabelle 29 sind die MQI-Beispielquellendateien aufgeführt. Standardmäßig befinden sich die Quellendateien im Verzeichnis `MQS_EXAMPLES:` und die kompilierten Versionen im Verzeichnis `[.BIN]`, unterhalb von `MQS_EXAMPLES:`. Weitere Informationen zur Funktion und zur Verwendung dieser Programme finden Sie im Handbuch *MQSeries Application Programming Guide*.

Tabelle 29. Beispielprogramme - Quellendateien

C	COBOL	Funktion
AMQSBCG0.C	–	Liest den Nachrichtendeskriptor und die Nachrichtenkontextfelder aller Nachrichten einer bestimmten Warteschlange und gibt diese aus.
AMQSECHA.C	AMQVECHX.COB	Meldet eine Nachricht aus einer Nachrichtenwarteschlange an die Warteschlange für Antwortnachrichten zurück. Diese Datei kann als ausgelöstes Anwendungsprogramm ausgeführt werden.
AMQSGBR0.C	AMQ0GBR0.COB	Schreibt Nachrichten aus einer Warteschlange in SYS$OUTPUT; die Nachrichten sind weiterhin in der Warteschlange vorhanden. Dieses Programm verwendet den MQGET-Aufruf mit der Anzeigeoption.
AMQSGET0.C	AMQ0GET0.COB	Entfernt unter Verwendung des MQGET-Aufrufs Nachrichten aus einer angegebenen Warteschlange und schreibt diese in SYS$OUTPUT.
AMQSINQA.C	AMQVINQX.COB	Liest die ausgelöste Warteschlange; als Anforderung wird jeweils der Name einer Warteschlange angegeben, zu der daraufhin Informationen zurückgegeben werden.

Beispiele

Tabelle 29. Beispielprogramme - Quellendateien (Forts.)

C	COBOL	Funktion
AMQSPUT0.C	AMQ0PUT0.COB	Kopiert SYS$INPUT in eine Nachricht und stellte die Nachricht anschließend in eine angegebene Warteschlange.
AMQSREQ0.C	AMQ0REQ0.COB	Stellt die Anforderungsnachrichten in eine angegebene Warteschlange und zeigt anschließend die entsprechenden Antworten an.
AMQSSETA.C	AMQVSETX.COB	Verhindert PUT-Vorgänge in einer angegebenen Warteschlange und gibt eine Ergebnisnachricht aus. Wird als ausgelöste Anwendung ausgeführt.
AMQSTRG0.C	–	Ein Auslösemonitor, der eine angegebene Initialisierungswarteschlange liest und anschließend das der jeweiligen Auslösenachricht zugeordnete Programm startet. Stellt einen Teil der Auslösefunktionen des mitgelieferten Befehls **runmqtrm** zur Verfügung.
AMQSVFCX.C	–	Eine C-Beispielstruktur für eine Exit-Routine für Datenkonvertierungen.

Verschiedene Tools

Diese Tool-Dateien werden zur Unterstützung des Formatierungsprogramms und der Codekonvertierung bereitgestellt.

Tabelle 30. Verschiedene Dateien

Dateiname	Pfad	Funktion
AMQTRC.FMT	SYS$LIBRARY	Definiert MQSeries-Trace-Formate.
CCSID.TBL	MQS_ROOT:[MQM.CONV.TABLE]	Editieren Sie diese Datei, wenn Ihrem MQSeries-System neue unterstützte CCSID-Werte hinzugefügt werden sollen. Weitere Informationen zu CCSIDs finden Sie in der CDRA-Dokumentation (Character Data Representation Architecture).

Anhang F. Schablonen für OpenVMS-Cluster-Überbrückungsgruppen

Dieser Anhang enthält folgende Schablonen für Überbrückungsgruppen:

Schablonenkonfigurationsdatei FAILOVER.TEMPLATE

```
#********************************************************************#
#*                                                                  *#
#* Statement:     Licensed Materials - Property of IBM              *#
#*                                                                  *#
#*                33H2205, 5622-908                                 *#
#*                33H2267, 5765-623                                 *#
#*                29H0990, 5697-176                                 *#
#*                (C) Copyright IBM Corp. 2000, 2001                *#
#*                                                                  *#
#********************************************************************#
#
# FAILOVER.TEMPLATE
# Template for creating a FAILOVER.INI configuration file
# All lines beginning with a '#' are treated as comments
#
# OpenVMS Cluster Failover Set Configuration information
# ------------------------------------------------------
#
# The TCP/IP address used by the OpenVMS Cluster Failover Set
#
IpAddress=n.n.n.n
#
# The TCP/IP port number used by the MQSeries Queue Manager
#
PortNumber=1414
#
# The timeout used by the EndCommand command procedure
#
TimeOut=30
#
# The command procedure used to start the Queue Manager
#
StartCommand=@sys$manager:start_qm
#
# The command procedure used to end the Queue Manager
#
EndCommand=@sys$manager:end_qm
#
# The command procedure used to tidy up on a node after a
# Queue Manager failure but the OpenVMS node did not fail
#
TidyCommand=@sys$manager:tidy_qm
#
# The directory in which the log files for the start, end and
# tidy commands are written
#
LogDirectory=mqs_root:[mqm.errors]
#
# The number of nodes in the OpenVMS Cluster Failover Set. The
```

FAILOVER.TEMPLATE

```
# number of nodes defined below must agree with this number
#
NodeCount=2
#
# The Name of the OpenVMS node
#
NodeName=BATMAN
#
# The TCP/IP interface name for the node
#
Interface=we0
#
# The priority of the node
#
Priority=1
#
# The Name of the OpenVMS node
#
NodeName=ROBIN
#
# The TCP/IP interface name for the node
#
Interface=we0
#
# The priority of the node
#
Priority=2
```

Abbildung 27. Schablonenkonfigurationsdatei failover.template

Befehlsprozedurschablone START_QM.TEMPLATE zum Starten

```
$on error then exit
$!********************************************************************
$!* Statement:     Licensed Materials - Property of IBM              *
$!*                33H2205, 5622-908                                 *
$!*                33H2267, 5765-623                                 *
$!*                29H0990, 5697-176                                 *
$!*                (C) Copyright IBM Corp. 2000, 2001                *
$!********************************************************************
$! Template command procedure used by Failover Sets to start the
$! queue manager
$! Parameters :
$! P1 = Queue Manager Name
$! P2 = Queue Manager Directory Name
$! P3 = TCP/IP address
$! P4 = TCP/IP interface name
$! P5 = Listener port number
$!
$ @sys$startup:mqs_symbols
$ set def mqs_root:[mqm.qmgrs.'p2'.errors]
$ define sys$scratch mqs_root:[mqm.qmgrs.'p2'.errors]
$!
$! Digital TCP/IP Services for OpenVMS commands
$!
$ @sys$startup:tcpip$define_commands
$!
$! Configure the IP address
$!
$ ifconfig 'p4' alias 'p3'
$!
$! TCPware for OpenVMS commands
$!
$! @tcpware:tcpware_commands
$!
$! Configure the IP address
$!
$! netcu add secondary 'p3'
$!
$! MultiNet for OpenVMS commands
$!
$! Configure the IP address
$!
$! define/sys/exec multinet_ip_cluster_aliases "''p3'"
$!
$! Restart the Multinet server
$!
$! @multinet:start_server
$!$! Start the queue manager
$!
$ strmqm 'p1'
$!
$! Start the listener
$!
$! runmqlsr -t tcp -p 'p5' -m 'p1'
$!
$! Insert commands to start any applications
$!
$exit
```

Abbildung 28. Befehlsprozedur START_QM.TEMPLATE zum Starten

Befehlsprozedurschablone END_QM.TEMPLATE zum Beenden

```
$on error then exit
$!
$!*********************************************************************
$!*                                                                   *
$!* Statement:     Licensed Materials - Property of IBM               *
$!*                                                                   *
$!*                33H2205, 5622-908                                  *
$!*                33H2267, 5765-623                                  *
$!*                29H0990, 5697-176                                  *
$!*                (C) Copyright IBM Corp. 2000, 2001                 *
$!*                                                                   *
$!*********************************************************************
$!
$! Template Command procedure used by Failover Sets to end the
$! queue manager
$!
$! Parameters :
$!
$! P1 = Queue Manager Name
$! P2 = Queue Manager Directory Name
$! P3 = TCP/IP address
$! P4 = TCP/IP interface name
$! P5 = Listener port number
$! P6 = End Queue Manager Timeout
$!
$ @sys$startup:mqs_symbols
$ check_qm:==$sys$system:mqcheckqm
$ set def mqs_root:[mqm.qmgrs.'p2'.errors]
$ define sys$scratch mqs_root:[mqm.qmgrs.'p2'.errors]
$ SS$_NORMAL=1
$ SS$_ABORT=44
$ SS$_TIMEOUT=556
$!
$! Insert commands to shutdown any applications prior to ending MQSeries
$!
$! Get the timeout period for each operation seconds
$!
$ timeout = 'p6'
$!
$! Initialise the outer loop
$!
$ out_count = 0
$!
$! Initialise the complete flag
$!
$ complete  = 0
$!
$ out_next:
$ if (out_count .gt. 2) .or. (complete .eq. 1) then goto out_finish
$!
$ if out_count .eq. 0
$ then
$!
$! End the queue manager gracefully first
$!
$ spawn/nowait $endmqm -i 'p1'
$ else
$ if out_count .eq. 1
$ then
$!
$! End the queue manager abruptly
$!
$ spawn/nowait $endmqm -p 'p1'
$ else
$!
```

```
$! Stop/id the execution controller
$!
$ check_qm -m 'p1'
$ if ( mqs$ec_pid .nes. "") then $stop/id='mqs$ec_pid'
$ endif
$ endif
$!
$ in_start:
$!
$! Initialise the outer loop
$!
$ in_count  = 0
$!
$ in_next:
$!
$! Inner loop
$!
$ if ( ( in_count .ge. timeout) .and. ( timeout .ne. 0 ) ) -
     .or. (complete .eq. 1) then goto in_finish
$!
$! Check if the execution controller is still running
$!
$ check_qm -m 'p1'
$ if mqs$ec_pid .eqs. ""
$ then
$!
$! The Execution controller is no longer running so we are finished
$!
$ complete = 1
$ goto in_finish
$ endif
$!
$! Wait a second and go round again
$!
$ wait 00:00:01
$ in_count = in_count + 1
$ goto in_next
$ in_finish:
$!
$! End of the inner loop
$!
$ out_count = out_count + 1
$ goto out_next
$ out_finish:
$!
$! End of the outer loop
$!
$! Digital TCP/IP Services for OpenVMS commands
$!
$ @sys$startup:tcpip$define_commands
$!
$! De-configure the IP address
$!
$ ifconfig 'p4' -alias 'p3'
$!
$! TCPware for OpenVMS commands
$!
$! @tcpware:tcpware_commands
$!
$! De-configure the IP address
$!
$! netcu remove secondary 'p3'
$!
$! MultiNet for OpenVMS commands
$!
$! De-configure the IP address
$!
```

END_QM.TEMPLATE

```
$! deass/sys/exec multinet_ip_cluster_aliases
$!
$! Restart the Multinet server
$!
$! @multinet:start_server
$!
$!
$! If the Queue Manager was shutdown successfully set the status
$! to SS$_NORMAL. If it was necessary to STOP/ID the Execution
$! controller set the status to SS$_ABORT and if the Execution
$! controller is still running set the status to SS$_TIMEOUT to
$! indicate an error
$!
$ if ( complete .eq. 1 )
$then
$!
$! End the listener process
$!
$!  endmqlsr -m 'p1'
$!
$ if ( out_count .eq. 3 )
$ then
$ exit SS$_ABORT
$ else
$ exit SS$_NORMAL
$ endif
$else
$ exit SS$_TIMEOUT
$endif
```

Abbildung 29. Befehlsprozedurschablone END_QM.TEMPLATE zum Beenden

Befehlsprozedurschablone TIDY_QM.TEMPLATE zum Bereinigen

```
$on error then exit
$!*********************************************************************
$!* Statement:     Licensed Materials - Property of IBM               *
$!*                                                                   *
$!*                33H2205, 5622-908                                  *
$!*                33H2267, 5765-623                                  *
$!*                29H0990, 5697-176                                  *
$!*                (C) Copyright IBM Corp. 2000, 2001                 *
$!*********************************************************************
$! Template Command procedure used by Failover Sets to tidy up after
$! a queue manager failure
$!
$! Parameters :
$! P1 = Queue Manager Name
$! P2 = Queue Manager Directory Name
$! P3 = TCP/IP address
$! P4 = TCP/IP interface name
$! P5 = Listener port number
$!
$ @sys$startup:mqs_symbols
$ set def mqs_root:[mqm.qmgrs.'p2'.errors]
$ define sys$scratch mqs_root:[mqm.qmgrs.'p2'.errors]
$!
$! Insert commands to do any tidying up after a queue manager has failed
$!
$! Digital TCP/IP Services for OpenVMS commands
$!
$ @sys$startup:tcpip$define_commands
$!
$! De-configure the IP address
$!
$ ifconfig 'p4' -alias 'p3'
$!
$! TCPware for OpenVMS commands
$!
$! @tcpware:tcpware_commands
$!
$! De-configure the IP address
$!
$! netcu remove secondary 'p3'
$!
$! MultiNet for OpenVMS commands
$!
$! De-configure the IP address
$!
$! deass/sys/exec multinet_ip_cluster_aliases
$!
$! Restart the Multinet server
$!
$! @multinet:start_server
$!
$exit
```

Abbildung 30. Befehlsprozedurschablone TIDY_QM.TEMPLATE zum Bereinigen

Anhang G. Unterstützte codierte Zeichensätze unter MQSeries for Compaq OpenVMS

MQSeries for Compaq OpenVMS unterstützt die meisten der codierten Codesätze, die in den standardmäßig in MQSeries for Compaq OpenVMS verfügbaren Ländereinstellungen verwendetet werden; bei Letzteren handelt es sich um den Teil der Benutzerumgebung, der die jeweils landesspezifischen Konventionen festlegt.

Ist keine Ländereinstellung angegeben, wird die CCSID 819 verwendet; dabei handelt es sich um den codierten Zeichensatz ISO8859-1.

Die CCSID (Coded Character Set Identifier, ID des codierten Zeichensatzes), die in MQSeries den für die Nachricht und die Nachrichtendaten verwendeten codierten Zeichensatz angibt, wird durch Analyse der Kategorie LC_CTYPE der lokalen Konfiguration ermittelt.

In Tabelle 31 sind die verschiedenen Ländereinstellungen sowie die CCSIDs aufgeführt, die für den in der jeweiligen Ländereinstellung verwendeten codierten Zeichensatz eingetragen sind.

Tabelle 31. Ländereinstellungen und CCSIDs

Ländereinstellung	**Sprache**	**Codierter Zeichensatz**	**CCSID**
C	Englisch	ISO8859-1	819
CS_CZ_ISO8859-2	Tschechisch	ISO8859-2	912
DA_DK_ISO8859-1	Dänisch	ISO8859-1	819
DE_DE_ISO8859-1	Deutsch	ISO8859-1	819
DE_CH_ISO8859-1	Deutsch (Schweiz)	ISO8859-1	819
EL_GR_ISO8859-7	Griechisch	ISO8859-7	813
EN_GB_ISO8859-1	Englisch (Großbritannien)	ISO8859-1	819
EN_US_ISO8859-1	Englisch (USA)	ISO8859-1	819
ES_ES_ISO8859-1	Spanisch	ISO8859-1	819
FI_FI_ISO8859-1	Finnisch	ISO8859-1	819
FR_FR_ISO8859-1	Französisch (Frankreich)	ISO8859-1	819
FR_BE_ISO8859-1	Französisch (Belgien)	ISO8859-1	819
FR_CA_ISO8859-1	Französisch (Kanada)	ISO8859-1	819
FR_CH_ISO8859-1	Französisch (Schweiz)	ISO8859-1	819
HU_HU_ISO8859-2	Ungarisch	ISO8859-2	912
IS_IS_ISO8859-1	Isländisch	ISO8859-1	819
IT_IT_ISO8859-1	Italienisch (Italien)	ISO8859-1	819
IW_IL_ISO8859-8	Hebräisch	ISO8859-8	916
JA_JP_EUCJP	Japanisch	eucJP	954
JA_JP_SDECKANJI	Japanisch	SDECKANJI	954**

Tabelle 31. Ländereinstellungen und CCSIDs (Forts.)

Ländereinstellung	Sprache	Codierter Zeichensatz	CCSID
JA_JP_SJIS	Japanisch	SJIS	932
KO_KR_DECKOREAN	Koreanisch	DECKOREAN	970**
NL_NL_ISO8859-1	Niederländisch (Niederlande)	ISO8859-1	819
NL_BE_ISO8859-1	Niederländisch (Belgien)	ISO8859-1	819
NO_NO_ISO8859-1	Norwegisch	ISO8859-1	819
PL_PL_ISO8859-2	Polnisch	ISO8859-2	912
PT_PT_ISO8859-1	Portugiesisch	ISO8859-1	819
SK_SK_ISO8859-2	Slowakisch	ISO8859-2	912
RU_RU_ISO8859-5	Kyrillisch	ISO8859-5	915
SV_SE_ISO8859-1	Schwedisch	ISO8859-1	819
TR_TR_ISO8859-9	Türkisch	ISO8859-9	920
ZH_CN_DECHANZI	Vereinfachtes Chinesisch	DECHANZI	1383**
ZH_HK_DECHANZI	Vereinfachtes Chinesisch	DECHANZI	1383**
ZH_HK_EUCTW	Traditionelles Chinesisch	eucTW	964
ZH_HK_EUCTW	Traditionelles Chinesisch	eucTW	964
ZH_HK_DECHANYU	Traditionelles Chinesisch	DECHANYU	964**
ZH_TW_DECHANYU	Traditionelles Chinesisch	DECHANYU	964**
ZH_HK_BIG5	Traditionelles Chinesisch	big5	950
ZH_TW_BIG5	Traditionelles Chinesisch	big5	950

Anmerkung:

** Es wird die nächstliegende für IBM registrierte CCSID verwendet.

Weitere Informationen zu plattformübergreifender Unterstützung dieser Ländereinstellungen finden Sie im Handbuch *MQSeries Application Programming Reference*.

Anhang H. Das Diagnose-Tool MONMQ

Bei MONMQ handelt es sich um ein Tool zur Unterstützung bei der Diagnose und Fehlerbehebung in MQSeries for Compaq OpenVMS. Dieses Dienstprogramm erlaubt eine interaktive Verwendung über die Befehlszeile oder über ein DCL-Script.

MONMQ wird in der Regel für folgende Aufgaben verwendet:
- Verwaltung von gemeinsam benutztem Speicher
- Erfassung von Informationen zur OpenVMS-Ressourcennutzung
- Abruf der Trace-Ausgabe von einem aktiven WS-Manager

Das Dienstprogramm MONMQ verfügt über eine Hilfefunktion, die Informationen zu Parametern zur Verfügung stellt, und führt darüber hinaus ein MONMQ-Befehlsscript aus. Beim Start von MONMQ wird das Standardscript `sys$manager:mqs_trace_startup.mqt` ausgeführt, das die Ausgangskonfiguration zur Verfügung stellt.

```
$monmq
ok - trace mailbox 0 opened as default

MQT> help
Die Hilfe kann zur Anzeige von Informationen zu verfügbaren Befehlen
oder Parametern verwendet werden
Help [ <verb> || <parameter/variable name> || commands || parameters || examples]

Gültige Trace-Befehle haben das folgende Format:
Verb [<parameters>] [<variable = expression>][;][optional second command]
```

Übersicht

Die Trace-Funktion für MQSeries unter OpenVMS ist unter Verwendung globaler Abschnitte und Mailboxes implementiert. Bis zu zehn Trace-Abschnitte (LUs) können sich in jedem beliebigen Knoten, auf dem MQSeries installiert ist, befinden. Es wird jedoch dringend empfohlen, Trace-Sitzungen nur für jeweils eine LU auszuführen. Ebenso wird Benutzern davon abgeraten, mehr als eine LU parallel zu öffnen. Eine Missachtung dieser Empfehlungen führt zu unvorhersehbaren Ergebnissen.

Jeder gemeinsame Abschnitt (einer LU) enthält die Kanaldefinitionen und die LU-Definition selbst. Die Kanaldefinitionen wiederum enthalten jeweils ausführliche Angaben zu den verbundenen Threads, das private Thread-Stack und den Thread-Ringpuffer. Darüber hinaus enthält der gemeinsame Abschnitt eine Gruppe von Markierungen (Flags) für die Interprozesskommunikation zwischen MONMQ und den verbundenen Threads.

Jeder LU ist eine Mailbox zugeordnet, in der die Echtzeit-Trace-Nachrichten empfangen werden. Für die Ausführung von Echtzeit-Trace-Vorgängen muss ein Client-Prozess mit dem Befehl **TRACE START** gestartet werden. Dieser dedizierte Hintergrundprozess liest und formatiert die in den LU-Mailboxes eintreffenden Nachrichten und zeigt diese an. Jeder verbundene Thread schreibt in dieselbe Mailbox; dadurch erhalten Sie die Möglichkeit, die Kommunikation zwischen MQSeries-Prozessen und Threads anzuzeigen.

Bei richtiger Ausführung kann MONMQ als umfassendes Verfahren zur Fehlerdiagnose z. B. bei Interprozess-Taktfehlern, bei Engpässen bezüglich Betriebssystemressourcen oder sogar bei Codierungsfehlern eingesetzt werden.

Die MONMQ-Befehle werden in diesem Anhang erläutert.

Variablen in MONMQ

In vielen Befehlen in MONMQ werden Variablen verwendet. Erfolgt im Befehl keine Angabe für eine Variable, wird der über den Befehl **set** definierte Standardwert übernommen. Werden Variablen mit einem anderen Befehl als **set** verwendet, bleibt ihr Standardwert unverändert.

Variablen können folgenden Inhalt haben:

- Integervariablen (dezimal oder hexadezimal).

  Hexadezimale Werte können mit einem führenden '0x' oder über Angabe der Buchstaben A-F eingegeben werden, die bei hexadezimalen Werten verwendet werden.
- Text, der in Anführungszeichen gesetzt werden muss.
- Ein Wertebereich, der im Format Mindestwert:Höchstwert angegeben wird.

  Ein Bereich wird verwendet, damit beispielsweise ein Befehl für mehrere Kanäle gelten kann.

Beispiel:

```
MQT> set lu=2
MQT> set pid=0x223
MQT> set pid=2fa
MQT> set buffile="filename.buf"
MQT> set chl=0:20
```

Der aktuelle Standardwert für die Variablen kann über den Befehl **variables** angezeigt werden.

```
MQT> variables
defined variables
lu=0:0                  nochls=20               buffer=1000
chl=0:20                component=0(HEX)        line=0
mask=0(HEX)             pid=0(HEX)              node=(null)
function=0(HEX)         div=0                   depth=32
resource=0              wait=1(BOOL)            timestamp=0(BOOL)
listfile=(null)         buffile=(null)          step=0(BOOL)
active=0(BOOL)          fname=(null)            delay=100
post=0(BOOL)


defined constants
fent=1(HEX)             fout=2(HEX)             ferr=4(HEX)
fxxx=8(HEX)             dgn=10(HEX)             shm=20(HEX)
spl=40(HEX)             evt=80(HEX)             mtx=100(HEX)
prc=200(HEX)            msc=400(HEX)            inf=800(HEX)
log=2000(HEX)           shl=4000(HEX)           memory=3
mutex=4                 mailbox=5               nanoseconds=1
microseconds=2          milliseconds=3          seconds=4
```

Es wird empfohlen, Standardwerte zu definieren, um die Verwendung der Befehle zu vereinfachen. Beispielsweise sind die folgenden Befehlsfolgen von ihrer Funktion her identisch:

```
MQT> open lu=0 buffer=1000 nochls=20
MQT> open lu=1 buffer=1000 nochls=20
MQT> show channels lu=0 chl=1:10
MQT> show channels lu=1 chl=1:10
```

oder

```
MQT> set nochls=20 chl=0:10 buffer=1000 lu=0
MQT> open
MQT> open lu=1
MQT> show channels
MQT> show channels lu=1
```

MONMQ-Befehle können auf die Mindestzahl von Zeichen gekürzt werden, die noch einen eindeutigen Befehl ergeben. Die im Beispiel oben angegebenen Befehle können beispielsweise wie folgt abgekürzt werden:

```
MQT> se noc=20 ch=0:10 buffe=1000 lu=0
MQT> op
MQT> op lu=1
MQT> sh ch
MQT> sh ch lu=1
```

MONMQ kann darüber hinaus einfache arithmetische Operationen mit Variablen ausführen, so dass Befehle wie **set lu=lu+1** möglich sind.

Neue Variablen können mit dem Befehl **declare** deklariert werden. Dieser Befehl hat folgende Parameter:

- Variablenname
- Variablentyp
- Hilfetext. Dieser kann anschließend über den Befehl **help** abgerufen werden.

```
MQT> declare ec int "channel number for execution controller"
MQT> set ec = 4
MQT> show channel chl=ec
Chl    Pid    Mailbox    Stack   Active  Post    Time   Mask     Process Name
  4    2c1f   7ee70290     4        0      0      0    ffffffff   AMQZXMA0.EXE
MQT> help ec

VARIABLE ec:
channel number for execution controller

MQT>
```

Standardwerte zuordnen

DEFAULT variable=<*expression*> [variable=*expression*] ...

Mit diesem Befehl können allen in MONMQ definierten Variablen Standardwerte zugeordnet werden. Anschließend muss der Variablenname in der Befehlszeile nicht mehr angegeben werden. Beispiel:

```
MQT>default lu=2 chl=3:6
```

Mit diesem Befehl wird für die Variable **lu** der Standardwert 2, für die Variable **chl** der Bereich 3 bis 6 definiert. Bei einem typischen Befehl wie beispielsweise **show channels** werden daraufhin die Kanäle 3 bis 6 (einschließlich) in LU 2 angezeigt.

Standardwerte werden nur übernommen, wenn die Variable nicht in der Befehlszeile angegeben wird. Alle Standardwerte werden in der Startscriptdatei `MQS_TRACE_STARTUP.MQT` festgelegt. Sie können diese Datei nach Bedarf editieren.

Trace-Abschnitt und zugeordnete Mailbox öffnen bzw. erstellen

OPEN [lu=*number*] [nochls=*number*] [buffer=*number*]

Mit diesem Befehl wird ein Trace-Abschnitt und die zugeordnete Mailbox geöffnet bzw. erstellt. Der Befehl **open** erstellt die Basisressource, die für die Ausführung der Trace-Funktion für MQSeries-Prozesse erforderlich ist. Jeder LU ist ein gemeinsamer Abschnitt und eine Mailbox für die Kommunikation mit MQSeries-Prozessen zugeordnet.

In diesem Befehl können drei Parameter angegeben werden. Bei dem ersten Parameter [LU] handelt es sich um die Nummer, die dem Trace-Abschnitt bzw. der Mailbox zugeordnet ist und in den meisten anderen MONMQ-Befehlen als Verweis verwendet wird. In einem Knoten können maximal zehn LUs erstellt werden. Der Standardwert ist null. Ist die angegebene LU bereits vorhanden, stellt MONMQ eine Verbindung zu dem bereits vorhandenen Trace-Abschnitt her. Ist kein Abschnitt vorhanden, wird ein neuer Abschnitt erstellt.

Der zweite Parameter [nochls] gibt die Anzahl der Kanäle an, über die diese LU verfügen wird. Jeder Kanal stellt eine MQSeries-Prozess/Thread-Verbindung dar. Standardwert ist 20.

Der dritte Parameter [buffer] gibt die maximale Größe des Trace-Protokollpuffers für jeden der Kanäle an. Standardwert ist 100.

Andere MONMQ-Befehle können nur ausgeführt werden, wenn mindestens eine LU geöffnet ist.

Definition der logischen Einheit (LU) anzeigen

SHOW SEGMENT [lu=*range*]

Mit diesem Befehl wird die Definition der logischen Einheit (LU) angezeigt. Sie sehen unten eine Beispielausgabe für den Befehl **show segment lu=0** mit einer kurzen Beschreibung neben den einzelnen Feldern.

```
Trace LU           : 0             /* Die für den Parameter OPEN [lu] angegebene LU-Nummer.
Mailbox name       : MQS_TRC_MBX_0 /* Name der dieser LU fest zugeordneten Mailbox.
Device name        : MBA1065:      /* Einheitenname der Mailbox.
Status             : Disabled      /* Der aktuelle Status der Mailbox.
Mailbox channel    : 352           /* Die dem MONMQ-Prozess zugeordnete Mailbox-Kanalnummer
History buffer size: 1000          /* Die max. zulässige Anzahl Nachrichteneinträge im Protokoll-
                                   /* ringpuffer (wie über den Parameter OPEN [buffer] festgelegt)
Threads mapped #   : 1             /* Anzahl der Prozess/Threads, die dem globalen Abschnitt
                                   /* dieser LU zugeordnet sind (MONMQ immer zugeordnet)
Time stamping      : Enabled       /* Globale Zeitmarkenmarkierung (noch nicht implementiert)
Max channels #     : 20            /* Anzahl der für diese LU definierten Kanäle (festgelegt über
                                   /* den Parameter OPEN [nochls])
Display deph       : 0             /* Gibt die Stack-Tiefe an, die angezeigt wird; Standardwert
                                   /* ist 0, d. h., alle Stack-Einträge werden angezeigt.
Text filename      :               /* Die Trace-Textdatei für den Client
Binary filename    :               /* Die binäre Trace-Datei für den Client
Last Status        : 1             /* Letzter Status der QIO-Aktivitäten der Mailbox (hilfreich bei
                                   /* VMS-Ressourcenengpässen und bei Ausfall von MONMQ)
Connection map[0]  : 0             /* Eine Bitmap aller verbundene Kanäle (max. Anzahl Kanäle: 128)
```

LU schließen und löschen

CLOSE [lu=*number*]

Dieser Befehl ist das Gegenstück zum Befehl **OPEN**, d. h., er schließt und löscht die angegebene LU. Der Ablauf erfolgt gesteuert; zunächst wird jedem verbundenen Prozess signalisiert, dass die Verbindung getrennt werden soll, anschließend werden die einzelnen Kanäle zurückgesetzt. Zuletzt wird die Zuordnung zur Trace-Mailbox aufgehoben und der gemeinsame Abschnitt gelöscht. Dieser Befehl sollte erst nach Abschluss einer Trace-Sitzung ausgeführt werden.

Kanalinformationen anzeigen

SHOW CHANNELS [full] [connected] [chl=*range*]

Mit diesem Befehl werden Informationen zu dem angegebenen Kanal angezeigt. Bei Angabe des Parameters **[connected]** werden nur die Kanäle angezeigt, für die ein Thread verbunden ist. Bei Eingabe des Befehls `show channels connected` wird beispielsweise Folgendes angezeigt:

```
Chl Pid/Tid      Mailbox   Stack  History  RTime  Time    Mask     Process Name
0   00000245/1   800b0330    4       0       0      0   fffffff    AMQZLAA0.EXE
1   00000244/1   800b0200    8       0       0      0   ffffffff   RUNMQCHI.EXE
2   00000243/1   800b01e0   10       0       0      0   ffffffff   AMQRRMFA.EXE
3   00000242/1   800b01c0    4       0       0      0   ffffffff   AMQZLLP0.EXE
4   00000241/1   800b0220    5       0       0      0   ffffffff   AMQHASMX.EXE
5   00000240/1   800b03f0    4       0       0      0   ffffffff   AMQZXMA0.EXE
```

Bei Angabe des Parameters **[full]** wird die vollständige Definition der angegebenen Kanäle angezeigt. Bei Eingabe des Befehls **show channels full connected chl=0:3** wird beispielsweise Folgendes angezeigt:

```
Pid/Tid             : 0000024b/1        /* Prozess-ID der verbundnen Threads und Thread-Folgenummer
Status              : *** Connected *** /* Aktueller Kanalstatus (Thread ist verbunden)
Process name        : AMQRRMFA.EXE      /* Prozessname des verbundenen Threads
Assigned LU         : 0                 /* Die diesem Kanal zugeordnete LU
Channel no.         : 2                 /* Die Nummer des zugeordneten Kanals in der LU
Mailbox channel     : 800b01e0          /* Mailbox-Kanalnummer der verbundenen Threads
                                        /* für den Trace-Kanal
Current stack depth : 10                /* Aktuelle Stack-Tiefe der Threads
Circular logging    : Disabled          /* Markierung, die angibt, ob die Protokollierung aktiviert ist
Next log entry      : 0                 /* Nummer des nächsten Slots im Protokollpuffer
Realtime tracing    : Disabled          /* Markierung, die angibt, ob Echtzeitprotokollierung aktiv ist
                                        /* (Client muss die Nachrichten lesen)
Time stamping       : Disabled          /* Aktiviert Zeitmarken für die Nachrichten dieses Threads
Trace mask          : ffffffff          /* Hexadezimales Format für die Trace-Maske dieses Threads (siehe
                                        /* Befehl show mask)
Step mode           : Off               /* Noch nicht implementiert
No Wait             : On                /* Ermöglicht eine Fortsetzung der QIO-Aktivität der Threads erst,
                                        /* wenn Ressource verfügbar ist
Last QIO status     : 0                 /* Letzter QIO-Aufrufstatus der Threads
Mapped address      : 9a2000-c9ffff     /* Der virtuelle zugeordnete Adressbereich des globalen
                                        /* LU-Abschnitts für diesen Thread
```

Aktuelle Trace-Maske für einen Kanal anzeigen

SHOW MASK [chl=*range*]

Mit diesem Befehl wird die aktuelle Trace-Maske für einen Kanal angezeigt. Eine hervorgehobene Linie weist darauf hin, dass das Trace-Maskenbit aktiviert ist. Bei Eingabe des Befehls **show mask chl=1** wird beispielsweise Folgendes angezeigt.

```
Trace Mask for Channel 1

Bit 00 - (fent) function entry          Nachrichten zum Funktionseintritt
Bit 01 - (fout) function exit           Nachrichten zum Funktionsausgang
Bit 02 - (ferr)function exit with error Funktionsaustritt, bei dem ein Fehlerstatus zurückgegeben wird
Bit 03 - (fxx) missing function exit    Nachricht über fehlenden Funktionseintritt/-austritt
                                           (siehe Anmerkung unten)
Bit 04 - (dgn) diagnostic messages      Diagnosenachrichten
Bit 05 - (shm) shared memory            OVMS-Nachrichten zum gemeinsam benutzten Speicher
Bit 06 - (spl) spinlocks                OVMS-Spinlock-Nachrichten
Bit 07 - (evt) events                   OVMS-Ereignisnachrichten
Bit 08 - (mtx) mutexes                  OVMS-Mutex-Nachrichten
Bit 09 - (prc) process msgs             OVMS-Thread-Nachrichten
Bit 10 - (msc) miscellaneous            Verschiedene OVMS-Kernel-Nachrichten
Bit 11 - (inf) informational            Interne Nachrichten, wie sie vom Befehl show angefordert werden
Bit 12 -                                Reserviert für benutzerdefinierte Nachrichten
```

Aus dieser Ausgabe geht hervor, dass für diesen Thread die Funktionsein- und -austrittnachrichten, Spinlock-Nachrichten (eine Synchronisationssperre) sowie Ereignisnachrichten aufgezeichnet werden. Die Aufzeichnung aller anderen Nachrichtenarten wird blockiert.

Inhalt des Ziel-Thread-Stacks anzeigen

SHOW STACK [chl=*range*]

Mit diesem Befehl wird der Inhalt des Ziel-Thread-Stacks angezeigt. Bei Eingabe des Befehls **show stack chl=0:1** wird beispielsweise Folgendes angezeigt:

```
0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 12:36:20.18 03 - 02 --->| zcpReceiveOnLink
0003 - 12:36:20.81 03 - 03 ---->| xcsWaitEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt

0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 12:36:20.18 03 - 02 --->| zcpSendOnLink
0003 - 12:36:20.81 03 - 03 ---->| xcsPostEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt
```

Aktive MQSeries-Prozesse und deren Speicherbelegung anzeigen

SHOW PROCESSES

Mit diesem Befehl werden alle aktiven MQSeries-Prozesse im aktuellen Knoten sowie der von ihnen belegte Speicherplatz angezeigt. Bei Eingabe des Befehls **show process** wird beispielsweise Folgendes angezeigt:

```
PID      Proc_Name  Image     Process       WS_Size WS_Peak Virt_Peak Gbl_Pg_Cnt Prc_Pg_Cnt Total_Mem

0000023D BKM3_AG    AMQZLAA0  Agent          23152  16576   203776     3616       12960      16576
0000023C BKM3_CI    RUNMQCHI  Run Chan Init  8752   6208    180832     1840       4368       6208
0000023B BKM3_RM    AMQRRMFA  Repository Mgr 11152  8144    185360     2224       5920       8144
0000023A BKM3_CP    AMQZLLP0  Checkpoint     8752   6384    185952     1920       4464       6384
00000239 BKM3_LG    AMQHASMX  Logger         8752   6288    182016     2080       4208       6288
00000238 BKM3_EC    AMQZXMA0  EC             20752  15232   203792     3536       11680      15216
00000128 _FTA4:     MONMQ     MONMQ Utility  8400   8528    198736     2224       3584       5808
```

Alle in einem Kanal vorhandenen Nachrichten anzeigen

SHOW HISTORY [chl=*range*]

Mit diesem Befehl werden alle im Ringprotokollpuffer des angegebenen Kanals vorhandenen Nachrichten angezeigt. Jede Nachricht wird formatiert; die Einrückungen in der Ausgabe geben die Stack-Tiefe an, bei der die Nachrichten erstellt wurden. Bei Eingabe des Befehls **show history chl=3** wird beispielsweise Folgendes angezeigt:

```
0215 - 12:35:44.52  03 - 02 ---<| zxcProcessChildren
0216 - 12:35:44.55  03 - 02 --->| zxcStartWLMServer
0217 - 12:35:44.57  03 - 02 ---<| zxcStartWLMServer
0218 - 12:35:44.59  03 - 02 --->| zcpReceiveOnLink
0219 - 12:35:44.61  03 - 03 ---->| xcsRequestMutexSem
0220 - 12:35:44.63  03 - 04 ----->| xllSemReq
0221 - 12:35:44.66  03 - 05 ------>| vms_mtx
0222 - 12:35:44.66  03 - 05 .......| vms_mtx  :- Locking BKM3/@ipcc_m_1_10 - timeout: -1
0223 - 12:35:44.70  03 - 06 ------->| vms_get_lock
0224 - 12:35:44.72  03 - 06 -------<| vms_get_lock
0225 - 12:35:44.74  03 - 05 ------<| vms_mtx
0226 - 12:35:44.76  03 - 04 -----<| xllSemReq
0227 - 12:35:44.79  03 - 03 ----<| xcsRequestMutexSem
0228 - 12:35:44.81  03 - 03 ---->| xcsResetEventSem
0229 - 12:35:44.83  03 - 04 ----->| vms_evt
0230 - 12:35:44.83  03 - 04 ......| vms_evt Reset on mailbox BKM3/@ipcc_e_1_2 : tout = -1
0231 - 12:35:44.87  03 - 05 ------>| vms_get_mbx_chan
0232 - 12:35:44.87  03 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0233 - 12:35:44.87  03 - 05 .......| vms_get_mbx_chan Returning key 1a0
0234 - 12:35:44.94  03 - 05 ------<| vms_get_mbx_chan
0235 - 12:35:44.83  03 - 04 ......| vms_evt rc = 0
0236 - 12:35:44.98  03 - 04 -----<| vms_evt
0237 - 12:35:45.00  03 - 03 ----<| xcsResetEventSem
0238 - 12:35:45.03  03 - 03 ---->| xcsReleaseMutexSem
0239 - 12:35:45.05  03 - 04 ----->| xllSemRel
0240 - 12:35:45.07  03 - 05 ------>| vms_mtx
0241 - 12:35:45.07  03 - 05 .......| vms_mtx  :- Unlocking BKM3/@ipcc_m_1_10 - timeout: -1
0242 - 12:35:45.11  03 - 06 ------->| vms_get_lock
0243 - 12:35:45.13  03 - 06 -------<| vms_get_lock
0244 - 12:35:45.16  03 - 05 ------<| vms_mtx
```

Diese Beispielausgaben enthalten die Zeilennummer innerhalb des Protokollpuffers, den Zeitpunkt, zu dem die Nachricht erstellt wurde, die Kanalnummer, die Stack-Tiefe, bei der die Nachricht erstellt wurde, sowie den Namen der Funktion. Beim Öffnen der LU wurde die maximal zulässige Anzahl von Protokollnachrichteneinträgen definiert. Ist der Puffer voll, springt MONMQ zum ersten Eintrag und überschreibt diesen, dann den nächsten usw. Wird während des Trace-Vorgangs ein FFST generiert, wird der Trace für den fehlerhaften Thread am Fehlerpunkt inaktiviert. Dadurch soll verhindert werden, dass der Puffer mit den von den Fehlerroutinen generierten Nachrichten gefüllt wird. Aus diesem Grund ist in einem solchen Fall die letzte Nachricht im Puffer die, die bei der Generierung des FFST geschrieben wurde.

Alle globalen MQSeries-Abschnitte im aktuellen Knoten anzeigen

SHOW GLOBALS

Mit diesem Befehl werden alle globalen MQSeries-Abschnitte im aktuellen Knoten angezeigt.

```
MQS1_shm_00000000
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=6128/383
MQS1_shm_01300010
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=1904/119
MQS1_shm_012c000f
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=464/87
MQS1_shm_012c000e
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=4112/514
MQS1_shm_012c000d
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=240/30
MQS1_shm_012c000c
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=528/132
MQS1_shm_012c000b
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=272/51
MQS1_shm_012c000a
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=4112/771
MQS1_shm_012c0009
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=272/51
MQS1_shm_012c0008
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=144/27
MQS1_shm_012c0007
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=528/99
MQS1_shm_012c0006
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=16/2
MQS1_shm_012c0005
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=128/24
MQS1_shm_012c0004
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=1968/492
MQS1_shm_012c0003
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=1904/476
MQS1_shm_012c0002
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=304/76
MQS1_shm_012c0001
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=1904/595
MQS1_shm_01280000
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=16/6
MQS1_shm_fffffffe
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=16/0
MQS1_shm_ffffffff
       (00000000) WRT     DZRO PRM SYS            Pgltcnt/Refcnt=144/63
```

Mutex-Tabelle anzeigen

SHOW MUTEX [chl=*range*]

Dieser Befehl weist den Ziel-Thread an, den Inhalt seiner internen Mutex-Tabelle an den Client-Trace-Prozess zu senden. Dabei ist es wichtig, dass die richtigen Trace-Maskenbits gesetzt werden, damit diese Informationen vom Client angezeigt werden können. Für diesen Kanal müssen in der Trace-Maske die Bits INF und DGN aktiviert sein (siehe „Maskenbit aktivieren/inaktivieren" auf Seite 370). Bei Eingabe des Befehls **show mutex chl=2** wird beispielsweise Folgendes angezeigt:

```
Mutex Utilisation for Process AMQZXMA0.EXE - Pid 248 ***

0960 - Lock ID: 0100013c - Name: BKM3/@ipcc_m_1_24
0961 - Lock ID: 020006ca - Name: BKM3/@ipcc_m_1_23
0962 - Lock ID: 0b00061e - Name: BKM3/@ipcc_m_1_22
0963 - Lock ID: 0900068e - Name: BKM3/@ipcc_m_1_21
0964 - Lock ID: 0b00032f - Name: BKM3/@ipcc_m_1_20
0965 - Lock ID: 210006eb - Name: BKM3/@ipcc_m_1_19
0966 - Lock ID: 07000742 - Name: BKM3/@ipcc_m_1_18
0967 - Lock ID: 1e000075 - Name: BKM3/@ipcc_m_1_17
0968 - Lock ID: 0c0004dd - Name: BKM3_m_1_45
0969 - Lock ID: 0d00035a - Name: BKM3/@ipcc_m_1_16
0970 - Lock ID: 190000a1 - Name: BKM3/@ipcc_m_1_15
0971 - Lock ID: 1a0005a3 - Name: BKM3/@ipcc_m_1_14
0972 - Lock ID: 14000628 - Name: BKM3/@ipcc_m_1_13
0973 - Lock ID: 130005f3 - Name: BKM3/@ipcc_m_1_12
0974 - Lock ID: 0f0000dc - Name: BKM3_m_1_43
0975 - Lock ID: 02000095 - Name: BKM3_m_1_42
0976 - Lock ID: 2200053e - Name: BKM3_m_1_41
0977 - Lock ID: 020000fc - Name: BKM3_m_1_40
0978 - Lock ID: 31000113 - Name: BKM3_m_1_39
0979 - Lock ID: 02000555 - Name: BKM3_m_1_38
0980 - Lock ID: 2e000389 - Name: BKM3_m_1_37
0981 - Lock ID: 2300011f - Name: BKM3_m_1_36
0982 - Lock ID: 02000109 - Name: BKM3_m_1_35
0983 - Lock ID: 02000327 - Name: BKM3_m_1_34
0984 - Lock ID: 020004a8 - Name: BKM3_m_1_33
0985 - Lock ID: 02000453 - Name: BKM3_m_1_32
0986 - Lock ID: 260007ad - Name: BKM3_m_1_31
0987 - Lock ID: 0200060c - Name: BKM3_m_1_30
```

Diese Ausgabe enthält die Zeilennummer in der Protokolldatei, den Mutex-Namen und die Systemsperren-ID.

Interne Ereignistabelle anzeigen

SHOW EVENTS [chl=*range*]

Dieser Befehl weist den Ziel-Thread an, den Inhalt seiner internen Ereignistabelle an den Client-Trace-Prozess zu senden. Dabei ist es wichtig, dass die richtigen Trace-Maskenbits gesetzt werden, damit diese Informationen vom Client angezeigt werden können. Für diesen Kanal müssen in der Trace-Maske die Bits INF und DGN aktiviert sein (siehe „Maskenbit aktivieren/inaktivieren" auf Seite 370). Bei Eingabe des Befehls **show events chl=2** wird beispielsweise Folgendes angezeigt:

```
Event Utilisation for Process AMQZXMA0.EXE - Pid 248 ***

1037 - Channel: 000003e0 - Name: BKM3/@ipcc_e_1_19
1038 - Channel: 000003d0 - Name: BKM3/@ipcc_e_1_18
1039 - Channel: 000003c0 - Name: BKM3/@ipcc_e_1_17
1040 - Channel: 000003b0 - Name: BKM3/@ipcc_e_1_14
1041 - Channel: 000003a0 - Name: BKM3/@ipcc_e_1_13
1042 - Channel: 00000390 - Name: BKM3/@ipcc_e_1_12
1043 - Channel: 00000380 - Name: BKM3/@ipcc_e_1_10
1044 - Channel: 00000370 - Name: BKM3/@ipcc_e_1_9
1045 - Channel: 00000360 - Name: BKM3/@ipcc_e_1_8
1046 - Channel: 00000330 - Name: BKM3/@ipcc_e_1_7
1047 - Channel: 00000300 - Name: BKM3/@ipcc_e_1_6
1048 - Channel: 000002f0 - Name: BKM3/@ipcc_e_1_5
1049 - Channel: 000002b0 - Name: BKM3_e_1_11
1050 - Channel: 000002a0 - Name: BKM3_e_1_10
1051 - Channel: 00000290 - Name: BKM3_e_1_9
1052 - Channel: 00000280 - Name: BKM3_e_1_8
1053 - Channel: 00000270 - Name: BKM3_e_1_7
1054 - Channel: 00000260 - Name: BKM3_e_1_6
1055 - Channel: 00000250 - Name: BKM3_e_1_5
1056 - Channel: 00000240 - Name: BKM3_e_1_4
1057 - Channel: 00000230 - Name: BKM3_e_1_3
1058 - Channel: 00000220 - Name: BKM3_e_1_2
1059 - Channel: 00000210 - Name: BKM3_e_1_1
1060 - Channel: 00000200 - Name: BKM3_e_1_0
1061 - Channel: 000001c0 - Name: BKM3/@ipcc_e_1_4
1062 - Channel: 000001b0 - Name: BKM3/@ipcc_e_1_3
1063 - Channel: 000001a0 - Name: BKM3/@ipcc_e_1_2
1064 - Channel: 00000190 - Name: BKM3/@ipcc_e_1_1
1065 - Channel: 00000180 - Name: BKM3/@ipcc_e_1_0
1066 - *** End of data ***
```

Diese Ausgabe enthält die Zeilennummer in der Protokolldatei, die Nummer des Mailbox-Kanals und den Ereignisnamen.

Interne Tabelle mit zugeordnetem gemeinsam benutzten Speicher anzeigen

SHOW MEMORY [chl=*range*]

Dieser Befehl weist den Ziel-Thread an, den Inhalt seiner internen Tabelle mit dem zugeordneten gemeinsam benutzten Speicher an den Client-Trace-Prozess zu senden. Dabei ist es wichtig, dass die richtigen Trace-Maskenbits gesetzt werden, damit diese Informationen vom Client empfangen werden können. Für diesen Kanal müssen in der Trace-Maske die Bits INF und DGN aktiviert sein (siehe „Maskenbit aktivieren/inaktivieren" auf Seite 370). Bei Eingabe des Befehls **show memory chl=2** beispielsweise wird Folgendes angezeigt:

```
    *** Shared Memory Utilisation for Process AMQZXMA0.EXE - pid/tid 248-1 ***

0942 - ShmId: 0248000f - Addr: 011d8000/01211fff - Perm: 950 - Size: 00038530 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/AMQ
0943 - ShmId: 0248000e - Addr: 00fbc000/011bdfff - Perm: 950 - Size: 002005f8 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/AMQ
0944 - ShmId: 0248000d - Addr: 00f1c000/00f39fff - Perm: 950 - Size: 0001d478 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/WLM
0945 - ShmId: 0248000c - Addr: 00cba000/00cfbfff - Perm: 944 - Size: 000405f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/HMEMSET.0
0946 - ShmId: 0248000b - Addr: 00c98000/00cb9fff - Perm: 944 - Size: 000205f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon005.0
0947 - ShmId: 0248000a - Addr: 00a96000/00c97fff - Perm: 944 - Size: 00200584 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon004.0
0948 - ShmId: 02480009 - Addr: 00a74000/00a95fff - Perm: 944 - Size: 000205f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon003.0
0949 - ShmId: 02480008 - Addr: 00a62000/00a73fff - Perm: 944 - Size: 000105f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon002.0
0950 - ShmId: 02480007 - Addr: 00a20000/00a61fff - Perm: 944 - Size: 000405f0 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/Anon001.0
0951 - ShmId: 02480006 - Addr: 00908000/00909fff - Perm: 950 - Size: 00001664 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/PLU
0952 - ShmId: 02480005 - Addr: 008f8000/00907fff - Perm: 950 - Size: 0000fff8 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/IPC
0953 - ShmId: 02480004 - Addr: 00802000/008f7fff - Perm: 950 - Size: 000f4838 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/IPC
0954 - ShmId: 02480003 - Addr: 00714000/00801fff - Perm: 950 - Size: 000ec718 Name: /mqs_root/mqm/qmgrs/BKM3/@ipcc/shmem/SUB
0955 - ShmId: 02480002 - Addr: 006ee000/00713fff - Perm: 944 - Size: 000253a8 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/zutSESSAN
0956 - ShmId: 02480001 - Addr: 00600000/006edfff - Perm: 944 - Size: 000ec710 Name: /mqs_root/mqm/qmgrs/BKM3/shmem/SUBPOOL.0
0957 - ShmId: 01280000 - Addr: 005f8000/005f9fff - Perm: 950 - Size: 00000454 Name:                          /var/mqm/errors
0958 - *** End of data ***
```

Diese Ausgabe enthält die Zeilennummer in der Protokolldatei, die ID des gemeinsam benutzten Speichers, den virtuell zugeordneten Adressenbereich, die Markierungen, die für die Erstellung des Abschnitts bzw. für die Zuordnung zum Abschnitt verwendet werden, sowie den internen MQSeries-Namen, der an den Abschnitt vergeben wurde.

Aktive MQSeries-Komponenten nach Namen/hexadezimaler ID anzeigen

SHOW COMPONENTS

Mit diesem Befehl werden alle aktiven MQSeries-Komponenten nach Namen und nach ihren hexadezimalen IDs angezeigt. Diese hexadezimalen IDs werden in anderen show-Befehlen von MONMQ wie beispielsweise **show functions** oder **show components** verwendet. Bei Eingabe des Befehls **show components** beispielsweise wird Folgendes angezeigt:

```
00000001 - Data hardening
00000002 - Log management
00000003 - Object Catalogue
00000004 - Queue management
00000005 - Transaction Management
00000006 - Mobile Component
00000007 - Mobile Component
00000008 - Communications
0000000a - Object Authority Manager
0000000b - Logger
0000000d - LQM Kernal
0000000f - Administration App
00000010 - Administration App
00000013 - Command Server
00000014 - Remote queue processor
00000015 - XA Transaction Manager
00000016 - Data Conversion
00000017 - Common Services
00000018 - Common Services (overflow)
00000019 - Application Interface
0000001a - IPCC
0000001b - DCE Support
0000001c - Pluggable Services
0000001d - Agent
0000001e - XA Transaction Manager
0000001f - C++ Layer
00000020 - CLI
00000021 - Z Utilities
00000022 - Execution Controller
00000023 - App. Bindings
00000024 - Service Component
00000025 - Publish/Subscribe
00000026 - MMC Snap-in for Admin
00000027 - Web Administration
00000028 - KYG Services
00000029 - OVMS MQ kernel
```

Funktionen innerhalb einer angegebenen Komponente anzeigen

SHOW FUNCTIONS [comp=*hex*]

Mit diesem Befehl werden alle Funktionen in der angegebenen Komponente angezeigt. Dabei muss für die Komponente die hexadezimale ID angegeben werden. Mit dem Befehl **SHOW COMPONENT** werden alle aktiven MQSeries-Komponenten angezeigt. Bei Eingabe des Befehls **show functions component=0x1f** wird beispielsweise Folgendes angezeigt:

```
00000000 - ImqBinary::copyOut
00000001 - ImqBinary::pasteIn
00000002 - ImqCache::operator =
00000003 - ImqCache::moreBytes
00000004 - ImqCache::read
00000005 - ImqCache::resizeBuffer
00000006 - ImqCache::setDataOffset
00000007 - ImqCache::setMessageLength
00000008 - ImqCache::useEmptyBuffer
00000009 - ImqCache::write
0000000a - ImqDeadLetterHeader::pasteIn
0000000b - ImqDistributionList::openInfoPrepare
0000000c - ImqItem::structureIdIs
0000000d - ImqQueueManager::backout
0000000e - ImqQueueManager::begin
0000000f - ImqQueueManager::commit
00000010 - ImqQueueManager::connect
00000011 - ImqQueueManager::disconnect
00000012 - ImqMessageTracker::setAccountingToken
00000013 - ImqMessageTracker::setCorrelationId
00000014 - ImqMessageTracker::setGroupId
00000015 - ImqMessageTracker::setMessageId
00000016 - ImqObject::close
00000017 - ImqObject::closeTemporarily
00000018 - ImqObject::inquire
00000019 - ImqObject::open
0000001a - ImqObject::openFor
.............
```

Trace beim Start eines Prozesses aktivieren

ONSTARTUP [start] [lu=*number*] [chl=*range*]

Mit diesem Befehl kann die Trace-Funktion beim Start eines Prozesses aktiviert werden. Bei Ausführung des Befehls wird in der Systemtabelle mit logischen Namen der logische Name MQS_DEF_TRACE definiert; dieser besteht aus den im Befehl angegebenen Werten und hat folgendes Format *LU-Nummer Kanalnummer*. Beim Start eines MQSeries-Prozesses wird dieser logische Namen während der Initialisierungsroutine des Prozesses überprüft; ist er vorhanden, wird eine Verbindung zu der angegebenen LU und Kanalnummer hergestellt. Ist die Kanal-ID bereits zugeordnet, wird der nächste verfügbare Kanal verwendet. Dieser Befehl ist hilfreich, wenn für die Anfangsphasen bei der Erstellung von MQSeries-Prozessen bzw. MQSeries-Threads ein Trace erforderlich ist.

Trace-Aktivierung beim Start eines MQSeries-Prozesses verhindern

ONSTARTUP [stop]

Mit diesem Befehl wird der logische Name MQS_DEF_TRACE aus der Systemtabelle mit den logischen Namen entfernt, d. h., beim Start eines MQSeries-Prozesses wird nicht sofort ein Trace ausgeführt.

Ziel-Thread mit angegebenem Kanal verbinden

CONNECT pid *number* [tid=*number*] [chl=*range*]

Mit diesem Befehl wird der Ziel-Thread mit dem angegebenen Kanal verbunden. Ist kein Kanal angegeben, wird der zuerst verfügbare Kanal verwendet.

Ziel-Thread von angegebenem Kanal trennen

DISCONNECT [chl=*range*]

Mit diesem Befehl wird die Verbindung zwischen Ziel-Thread und angegebenem Kanal getrennt.

Echtzeit-Trace-Nachrichten in der Trace-Mailbox der LUs anzeigen

TRACE START [node=*string*]

Mit diesem Befehl wird ein Client-Trace-Prozess gestartet, bei dem Echtzeit-Trace-Nachrichten in der Trace-Mailbox der LUs angezeigt werden. Bei Angabe des optionalen Parameters **node** wird ein Fenster in dem angegebenen Knoten erstellt, in das die Ausgabe übertragen wird.

Aktuellen Client-Prozess freigeben und beenden

TRACE STOP

Mit diesem Befehl wird der aktuelle Client-Prozess von der Trace-Mailbox abgehängt und beendet. Dabei werden alle Threads, die momentan in diese Mailbox schreiben, inaktiviert.

```
MQT> trace stop

Circular buffering has been disabled for process 24d thread 1
Circular buffering has been disabled for process 24c thread 1
Circular buffering has been disabled for process 24b thread 1
Circular buffering has been disabled for process 248 thread 1
Disconnecting thread pid : 24d, tid : 1 from channel 0 ..... OK
Disconnecting thread pid : 24c, tid : 1 from channel 1 ..... OK
Disconnecting thread pid : 24b, tid : 1 from channel 2 ..... OK
Disconnecting thread pid : 248, tid : 1 from channel 3 .....OK

*** Trace ended - no processes connected ***
```

Diese Ausgabe wird im Fenster des Trace-Clients angezeigt.

Trace-Daten angeben

SELECT [component] AND/OR [function] OR [*fname*]

Mit diesem Befehl können Sie bis zu 8 Kombinationen von Komponente und Funktionen angeben, für die ein Trace ausgeführt werden soll. Alle anderen Trace-Daten werden herausgefiltert. Sie können eine Funktion und/oder Komponente angeben. Ist die angegebene Komponente/Funktion gültig, wird sie in die Filtertabelle geschrieben. Sind keine Einträge vorhanden, wird ein Trace für ALLE Komponenten/Funktionen ausgeführt.

Bei Eingabe des Befehls **SELECT** ohne Parameter wird der Inhalt der Filtertabelle angezeigt. Jede Ausgabezeile enthält den Tabellenindexeintrag, die hexadezimale ID der Komponente und der Funktion und den Textnamen der Funktion. Wird nur eine Komponente angegeben, wird ein Trace für alle Funktionen in dieser Komponente ausgeführt. In diesem Fall ist für die Funktion der Wert 0xffff angegeben.

Bei Eingabe des Befehls **SELECT** ohne Parameter wird beispielsweise Folgendes angezeigt:

```
Chl:0 - Cmp/fnc selection criteria
ALL component/functions
```

Bei Eingabe der folgenden Befehle:

```
MQT>select fname="kill"
MQT>select comp=0x1f
MQT>select comp=0x20 func=0x3
MQT>select
```

wird Folgendes angezeigt:

```
Chl:0 - Cmp/fnc selection criteria
Idx: 0 - Cmp: 00000029 - Fnc: 00000005 - Name - kill
Idx: 1 - Cmp: 00000019 - Fnc: 0000ffff - Name -
Idx: 2 - Cmp: 00000016 - Fnc: 00000003 - Name - vqiAddCacheEntry
```

Einzelnen Eintrag aus der Trace-Filtertabelle entfernen

DESELECT INDEX=<0:7>

Mit diesem Befehl wird ein einzelner, über den Parameter **table index** angegebener Tabelleneintrag aus der Trace-Filtertabelle entfernt. Sind alle 8 Einträge leer, wird ein Trace für alle Komponenten/Funktionen ausgeführt. Bei Eingabe des Befehls **select** beispielsweise werden die folgenden Einträge mit ihren Indexen angezeigt:

```
Chl:0 - Cmp/fnc selection criteria
Idx: 0 - Cmp: 00000029 - Fnc: 00000005 - Name - kill
Idx: 1 - Cmp: 00000019 - Fnc: 0000ffff - Name -
Idx: 2 - Cmp: 00000016 - Fnc: 00000003 - Name - vqiAddCacheEntry
```

Mit den folgenden **deselect**-Befehlen werden die angegebenen Prozesse bzw. Funktionen entfernt:

```
MQT> desel index=0
MQT> desel index=2

MQT>select
Chl:0 - Cmp/fnc selection criteria
Idx: 1 - Cmp: 00000019 - Fnc: 0000ffff - Name -
-------------------------------------------
MQT> deselect index=1
MQT> select

Chl:0 - Cmp/fnc selection criteria
ALL component/functions
-------------------------------------------
```

Client-Prozess - Trace-Nachrichten in Binärdatei schreiben

OPEN BINARY [filename=*string*]

Mit diesem Befehl wird eine binäre Trace-Nachrichtendatei geöffnet und der Client-Prozess veranlasst, Echtzeit-Trace-Nachrichten in diese Datei zu schreiben. Diese Datei kann später für eine Leistungsanalyse von MQSeries-Anwendungen herangezogen werden. Der Standarddateiname ist `mqs_root:[mqm.errors]mqs_buffer_`*xx*`.bin` (dabei steht *xx* für die LU-Nummer).

Binärdatei für Trace-Nachrichten schließen

CLOSE BINARY

Mit diesem Befehl wird die angegebene binäre Trace-Datei der LU geschlossen.

Client-Prozess - Trace-Nachrichten in Textdatei schreiben

OPEN TEXT [filename=*string*]

Mit diesem Befehl wird eine lesbare Textdatei geöffnet und der Client-Prozess veranlasst, formatierte binäre Trace-Nachrichten in diese Datei zu schreiben. Diese Datei kann später mit dem DCL-Befehl **type** oder **edit** angezeigt werden. Der Standarddateiname ist `mqs_root:[mqm.errors]mqs_buffer_xx.lis` (dabei steht *xx* für die LU-Nummer). Diese Ausgabedatei hat den Vorteil, dass sie ohne weitere Vorverarbeitungen gelesen werden kann. Ein Nachteil hingegen ist, dass sie mehr Plattenspeicherplatz als eine Binärdatei benötigt.

Textdatei mit Trace-Nachrichten schließen

CLOSE TEXT

Mit diesem Befehl wird die Trace-Textdatei der LU geschlossen.

Nachrichten mit Zeitmarken versehen

ENABLE TIMESTAMP [chl=*range*]

Mit diesem Befehl wird in der Kanaldefinitionstabelle die Zeitmarkenmarkierung gesetzt. Mit diesem Befehl werden MQSeries-Prozesse angewiesen, in jeder Nachricht die aktuelle Uhrzeit einzufügen. Diese Markierung muss gesetzt werden, wenn eine binäre Trace-Datei zu Leistungsanalysen herangezogen werden soll.

Nachrichten nicht mehr mit Zeitmarken versehen

DISABLE TIMESTAMP [chl=*range*]

Mit diesem Befehl wird die Zeitmarkenmarkierung in der Kanaldefinitionstabelle inaktiviert (siehe „Nachrichten mit Zeitmarken versehen").

Trace aktivieren

ENABLE TRACE [chl=*range*]

Mit diesem Befehl wird in der Kanaldefinitionstabelle die Trace-Markierung 'RTime' gesetzt. Durch Aktivieren bzw. Inaktivieren dieser Markierung können Sie festlegen, ob Trace-Nachrichten an einen Trace-Client geschickt werden sollen. Ist ein Thread verbunden und ein Trace-Client vorhanden, können Sie mit dem Befehl TRACE START einfach den Kanal ein- oder ausschalten, anstatt diesen Thread zu trennen.

Trace inaktivieren

DISABLE TRACE [chl=*range*]

Mit diesem Befehl wird in der Kanaldefinitionstabelle die Trace-Markierung 'RTime' inaktiviert (siehe „Trace aktivieren").

Nachrichtenprotokoll speichern

ENABLE HISTORY [chl=*range*]

Mit diesem Befehl wird in der Kanaldefinitionstabelle die Protokollmarkierung für die angegebenen Kanäle gesetzt. Er weist den verbundenen Thread an, Trace-Nachrichten in den Ringpuffer der LUs zu schreiben. Da dieser Schreibvorgang von dem Prozess ausgeführt wird, für den der Trace aktiviert ist, ist kein Client-Prozess erforderlich. Die Größe des Trace-Ringpuffers wird während der LU-Erstellung mit dem Befehl **open** definiert. Nachdem der letzte Eintrag in den Puffer geschrieben wurde, wird bei der nächsten Nachricht an den Pufferanfang zurückgekehrt und der erste Eintrag überschrieben. Das Feld **Next Log Record** in der LU-Definitionstabelle gibt an, wo der nächste Protokolleintrag eingetragen werden soll.

Nachrichtenprotokoll inaktivieren

DISABLE HISTORY [chl=*range*]

Mit diesem Befehl wird in der Kanaldefinitionstabelle die Protokollmarkierung für die angegebenen Kanäle inaktiviert (siehe „Nachrichtenprotokoll speichern").

Nachrichtenprotokoll löschen

DELETE HISTORY [chl=*range*]

Mit diesem Befehl werden alle Nachrichten aus dem Protokollringpuffer gelöscht. Da dieser Befehl unabhängig davon, ob Prozesse noch in den Puffer schreiben, ausgeführt werden kann, ist es nicht erforderlich, zuerst die Protokollierung zu inaktivieren.

Protokollumfang festlegen

SET [depth]

Mit diesem Befehl wird die maximale Stack-Tiefe angeben, die im Trace-Client-Fenster ausgegeben wird. Nachrichten unterhalb des angegebenen Wertes werden nicht angezeigt; wenn aktiviert, werden sie jedoch in der binären Trace-Datei und im Protokollpuffer angezeigt. Der Standardwert ist null, d. h., alle Nachrichten im Stack werden ausgegeben. Werden Analysedaten in eine Binärdatei geschrieben, wird Benutzern empfohlen, einen möglichst niedrigen Wert (z. B. 1) anzugeben. Die Anzeige des gesamten Stack-Inhalts wirkt sich nachteilig auf die Leistung von Client-Prozessen aus.

Stack- und Protokolldaten für einen Kanal zurücksetzen

SET [free] [chl=*range*]

Mit diesem Befehl wird der angegebene Kanal zurückgesetzt. Alle vorhandenen Stack- und Protokolldaten werden gelöscht, die Zuordnung des Kanals wird aufgehoben, und er steht wieder zur Verfügung.

Maskenbit aktivieren/inaktivieren

SET [mask=*var*] [chl=*range*]

Mit diesem Befehl wird das Maskenbit im entsprechenden Feld innerhalb des verbundenen Threads aktiviert bzw. inaktiviert. Jedes Bit steht für eine Nachrichtenart, die von einem MQSeries-Prozess erstellt wurde. Mit diesem Befehl können Sie angeben, für welche Nachrichtenart ein Trace ausgeführt werden sollen. Es gibt folgende Nachrichtenarten:

```
MQT>set mask = 0xffffff chl=1
MQT>show mask chl=1
```

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

Sollen mehrere Nachrichtenarten angegeben werden, müssen diese durch ein ODER-Zeichen getrennt werden; Beispiel:

```
 MQT>set mask =0x0 chl=1 MQT>show mask chl=1
```

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

```
MQT>set mask = mtx | evt | fent chl=1
MQT>show mask chl=1
```

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

Jede Maske besteht aus 12 Maskentypen, die ein- oder ausgeschaltet werden können, um einen bestimmte Nachrichtenart zu aktivieren oder zu inaktivieren. Wenn beispielsweise nur Funktionseingangspunkten angezeigt werden solle, müssen Sie den Befehl **set mask = fent** eingeben.

Farbe für Kanal festlegen

SET COLOR [chl=*range*]

Mit diesem Befehl wird dem angegebenen Kanal eine Farbe zugewiesen. Daraufhin werden alle Ausgabedaten in Zusammenhang mit diesem Kanal in der betreffenden Farbe angezeigt, bis eine andere Farbe angegeben oder der Kanal zurückgesetzt wird. Mit diesem Befehl kann innerhalb eines Ausgabedatenstroms optisch zwischen den verschiedenen Nachrichten eines Threads unterschieden werden. Beispielsweise wird bei Eingabe dieser Befehle:

```
MQT> set color=yellow chl=2
MQT> set color=blue chl=0
MQT> sho chan chl=0:3 connected
```

Folgendes angezeigt:

```
Chl  Pid/Tid     Mailbox  Stack  History  RTime  Time  Mask        Process Name
0   00000245/1   800b0330   4      0       0      0    ffffffff    AMQZLAA0.EXE
1   00000244/1   800b0200   8      0       0      0    ffffffff    RUNMQCHI.EXE
2   00000243/1   800b01e0  10      0       0      0    ffffffff    AMQRRMFA.EXE
3   00000242/1   800b01c0   4      0       0      0    ffffffff    AMQZLLP0.EXE
```

Dabei ist Kanal 0 blau und Kanal 2 gelb.

Ausgabe in Datei umleiten

SET OUTPUT [filename=*string*]

Mit diesem Befehl werden die Ausgabedaten nicht mehr angezeigt, sondern an die angegebene Datei übertragen. Bei Verwendung des Befehls **output** als Parameter in anderen Befehlen gilt er nur für den Befehl, in dem er angegeben ist. Fehler werden allerdings weiterhin am Bildschirm ausgegeben, und nicht an die Datei übertragen. Es werden nur gültige Trace-Daten in die angegebene Datei geschrieben.

Binäre Trace-Datei analysieren

ANALYSE [component] [function] [unit=*xx*]

Mit diesem Befehl wird der Inhalt der zuvor in einer Trace-Sitzung verwendeten binären Trace-Datei analysiert. Die zu analysierende Komponente bzw. Funktion kann zwar angegeben werden, dies führt jedoch nur zu einem Ergebnis, wenn die Datei Daten in Zusammenhang mit der betreffenden Komponente bzw. Funktion enthält. War zu dem Zeitpunkt, zu dem die Datei generiert wurde, eine bestimmte Trace-Maske oder eine bestimmte Komponente ausgewählt, enthält die Binärdatei nur Daten für diese ausgewählten Einträge; Komponenten oder Funktionen, die nicht diesen Kriterien entsprechen, können in der Analyse nicht verwendet werden.

Anmerkung: Wenn Sie den Befehl ausschreiben, achten Sie darauf, dass ANALYSE mit einem **S** geschrieben. Dieser MONMQ-Befehl sollte nicht mit dem OpenVMS-Befehl **ANALYZE** verwechselt werden. Es handelt sich hier um zwei verschiedene Befehle.

Der Parameter **unit** gibt die Zeiteinheit für die Analyse an; folgende Werte sind möglich (*xx*): Sekunden, Millisekunden, Mikrosekunden, Nanosekunden. Der Standardwert ist Millisekunden.

Soll die Ausgabe beispielsweise in Mikrosekunden erfolgen, müssen Sie den folgenden Befehl **analyse unit=micro:** eingeben. Im Folgenden sehen Sie ein Beispiel für die Ausgabe auf diesen Befehl hin:

```
================================================================================================
                          COMPONENT :- Common Services
================================================================================================
Calls  Minimum     Average     Maximum        Total            Function
 42      0.00       66.41       311.78      2789.15         xcsRequestMutexSem
 42      0.00       96.98       222.64      4072.98         xcsReleaseMutexSem
  6      0.00      138.50       286.11       831.00         xcsResetEventSem
  5      0.00    10112.60     10159.51     50563.01         xcsWaitEventSem
  6      0.00      564.74      1029.23      3388.45         xcsCheckExtendMemory
 42      0.00       51.32       266.86      2155.41         xllSemReq
 42      0.00       73.05       159.17      3068.16         xllSemRel
================================================================================================
                          COMPONENT :- Common Services (overflow)
================================================================================================
Calls  Minimum     Average     Maximum        Total            Function
 36      0.00       41.15       122.06      1481.35         xcsCheckProcess
  6      0.00       37.92        68.35       227.52         xihGetConnSPDetailsFromList
  6      0.00       65.26       114.25       391.58         xihHANDLEtoSUBPOOLFn
  6      0.00       12.69        23.44      1267.49         xihGetNextSetConnDetailsFromList
  6      0.00       12.53        22.46        75.19         xcsRequestThreadMutexSem
  6      0.00       12.37        22.46        74.21         xcsReleaseThreadMutexSem
================================================================================================
                          COMPONENT :- IPCC
================================================================================================
Calls  Minimum     Average     Maximum        Total            Function
  5      0.00    10589.97     11029.84     52949.85         xcpReceiveOnLink
================================================================================================
                          COMPONENT :- CLI
================================================================================================
Calls  Minimum     Average     Maximum        Total            Function
  6      0.00        1.95        1.95         11.72         zapInquireStatus
================================================================================================
                          COMPONENT :- Execution Controller
================================================================================================
Calls  Minimum     Average     Maximum        Total            Function
  6      0.00      954.46     3275.18       11726.79         zxcProcessChildren
  6      0.00       12.69       22.46          76.17         zxcStartWLMServer
================================================================================================
                          COMPONENT :- OVMS MQ kernel
================================================================================================
Calls  Minimum     Average     Maximum        Total            Function
 36      0.00       15.49       99.60         557.58         kill
  6      0.00        0.65        3.91           3.91         vms_mapgbl
 84      0.00       11.17       88.16         938.68         vms_get_lock
 84      0.00       42.41      221.94        3562.54         vms_mtx
 12      0.00       44.51      149.40         534.14         vms_get_mbx_chan
 11      0.00     4651.77    10137.05       51169.42         vms_evt
  6      0.00        1.63        4.88           9.76         vms_check_health
```

Die Ausgabe enthält folgende Spalten:

Calls (Aufrufe)
: Die Anzahl der Einträge in der Funktion während der Trace-Sitzung.

Minimum (Mindestzeitdauer)
: Die kürzeste Verweildauer in der Funktion.

Average (Mittel)
: Die gesamte Verweildauer in allen Aufrufen an die Funktion geteilt durch die Anzahl der Aufrufe.

Maximum (Max. Verweildauer)
: Die längste Verweildauer in der Funktion.

Total (Summe)
: Die gesamte Verweildauer in der Funktion für alle Aufrufe.

Function (Funktion)
: Der Name der Funktion.

Anmerkung: Der Bereich, den die Analyse abdeckt, entspricht dem Inhalt der binären Trace-Datei. Der Benutzer selbst legt den Umfang fest, indem er eine binäre Trace-Datei öffnet und die Trace-Funktion für den gewünschten Zeitpunkt aktiviert bzw. inaktiviert.

Aktuellen Status von MQSeries-Threads anzeigen

FFST [chl=*range*]

Mit diesem Befehl wird der mit dem Zielkanal verbundene Thread zur Ausgabe eines FFST gezwungen. Der Befehl hat KEINE Auswirkung auf den Ausführungspfad der Ziel-Threads. Mit diesem Befehl erhalten Sie eine Momentaufnahme vom aktuellen Status eines beliebigen MQSeries-Threads. Der FFST enthält hilfreiche Systeminformationen wie beispielsweise die Ressourcennutzung der Threads, Berechtigungen usw. Aus dem FFST-Header ist ersichtlich, dass der FFST von MONMQ (siehe unten) und NICHT auf Grund eines Fehlers erstellt wurde.

Im Folgenden eine Beispielausgabe:

```
MQSeries First Failure Symptom Report
  ====================================

  Date/Time         :- Wednesday November 12  10:59:38 GMT 2000
  Host Name         :- CATWMN (Unknown)
  PIDS              :- 5697175
  LVLS              :- 510
  Product Long Name :- MQSeries for OpenVMS Alpha
  Vendor            :- IBM
  Probe Id          :- VM026000
  Application Name  :- MQM
  Component         :- vms_evt
  Build Date        :- Oct 22 2000 (Collector)
  Userid            :- [400,400] (SYSTEM)
  Program Name      :- AMQZXMA0.EXE
  Process           :- 00000248
  Thread            :- 00000001
  QueueManager      :- BKM3
  Major Errorcode   :- xecF_E_UNEXPECTED_SYSTEM_RC
  Minor Errorcode   :- OK
  Probe Type        :- MSGAMQ6119
  Probe Severity    :- 2
  Probe Description :- AMQ6119: Es ist ein interner MQSeries-Fehler aufgetreten.
                           (***   FORCED FFST BY USER ***)
  Comment1          :- *** FORCED FFST BY USER ***
  Comment2          :- -SYSTEM-S-NORMAL, normal successful completion

etc.....
```

Trace schließen und MONMQ verlassen

EXIT

Mit diesem Befehl wird der Befehl **CLOSE** ausgeführt und MONMQ beendet.

MONMQ beenden, ohne den Trace zu schließen

QUIT

Mit diesem Befehl wird kein CLOSE-Befehl ausgeführt, sondern nur MONMQ beendet. Dieser Befehl ist daher hilfreich, wenn die Trace-Funktion aktiviert bleiben, MONMQ jedoch beendet werden soll. Beim nächsten Start von MONMQ wird die vorherige Trace-Sitzung wieder aufgenommen.

Gemeinsam benutzten Speicher mit MONMQ verwalten

In Ausnahmesituationen wie beispielsweise beim Ausfall eines WS-Managers oder einem erzwungenen Systemabschluss mit dem OpenVMS-Befehl **stop /id** kann es unter Umständen vorkommen, dass gemeinsam benutzte MQSeries-Speichersegmente vom WS-Manager nicht automatisch gelöscht werden. In diesem Fall kann der WS-Manager nicht erneut gestartet werden, da von **strmqm** gemeldet wird, dass der WS-Manager bereits aktiv ist.

MONMQ kann eine Liste der aktuell vorhandenen gemeinsam benutzten MQ-Speichersegmente (globale Abschnitte) ausgeben und diese löschen.

Anmerkung: Stellen Sie sicher, dass vor dem Löschen gemeinsam benutzter Speichersegmente mit dem Dienstprogramm MONMQ alle WS-Manager beendet werden. Wenn der gemeinsam benutzte Speicher eines aktiven WS-Managers gelöscht wird, führt dies zu Fehlern in dem betreffenden WS-Manager und möglicherweise auch in den Warteschlangendateien.

Mit dem Befehl **MONMQ SHOW PROCESS** können Sie sicherstellen, dass keine MQ-Prozesse mehr aktiv sind. Für aktive Prozesse in einem fehlerhaften WS-Manager, die nicht mit dem Befehl **endmqm** gestoppt werden können, können Sie stattdessen den OpenVMS DLC-Befehl **stop /id=<pid>** verwenden.

Stellen Sie sicher, dass keine MQSeries-Prozesse mehr aktiv sind, indem Sie folgenden Befehl eingeben:

```
MQT> show process
```

```
MQ Processes
PID     Proc Name   Image   Process     WS Size    WS Peak    Virt Peak  Gbl Pg Cnt Prc Pg Cnt Total Mem
------- ----------- ------- ----------- ---------- ---------- ---------- ---------- ---------- ----------

List the shared memory global sections that currently exist

MQT> show globals
MQS1_shm_2695000a
                  (00000000) WRT    DZRO PRM SYS Pgltcnt/Refcnt=4112/514
MQS1_shm_26950009
                  (00000000) WRT    DZRO PRM SYS Pgltcnt/Refcnt=272/34
MQS1_shm_ffffffff
                  (00000000) WRT    DZRO TMP SYS Pgltcnt/Refcnt=144/63
MQS1_shm_00000000
                  (00000000) WRT    DZRO TMP SYS Pgltcnt/Refcnt=6304/394
```

Die momentan vorhandenen gemeinsam benutzten Speichersegmente (globale Abschnitte) können mit dem folgenden Befehl aufgeführt werden:

```
MQT> show globals
```

Delete these global sections:

```
MQT> delete
Deleted global section: MQS1_shm_2695000a
Deleted global section: MQS1_shm_26950009
Deleted global section: MQS1_shm_ffffffff
sys$delgbl - unable to delete section MQS1_shm_00000000
```

Bei der Meldung 'unable to delete section MQS_shm_00000000' handelt es sich um eine erwartete Fehlermeldung, da dieser Abschnitt von MONMQ verwendet wird. Anschließend können Sie MONMQ mit folgendem Befehl beenden:

```
MQT> exit
```

Sie können den **delete**-Befehl in einem Script ausführen, vorausgesetzt, es sind keine WS-Managerprozesse mehr aktiv:

```
$ monmq delete
Deleted global section: MQS1_shm_2695000a
Deleted global section: MQS1_shm_26950009
Deleted global section: MQS1_shm_ffffffff
sys$delgbl - unable to delete section MQS1_shm_00000000
```

Scripts und Makros in MONMQ

MONMQ-Befehlsscripts können in MONMQ oder über die Eingabeaufforderung ausgeführt werden. Scripts sind beispielsweise für die Erfassung von Daten oder die Konfiguration der MONMQ-Umgebung hilfreich. Beim Start von MONMQ wird in SYS$MANAGER:MQS_TRACE_STARTUP.MQT ein Script gestartet, das die Trace-Variablen in MONMQ konfiguriert.

Anmerkung: Befindet sich das Script nicht im aktuellen Verzeichnis, muss der vollständige Pfadnamen des Scripts angegeben werden. Beispiel:

```
MQT> ! "sys$manager:test.mqt"
```

Ebenso können auch Makro definiert werden, um gängige oder häufig anfallende Vorgänge abzukürzen. Eine Makrodeklaration besteht aus drei Teilen:

1. Der erste Teil ist der Name des Makros, bei dem es sich um einen eindeutigen Befehlsnamen handeln muss.
2. Der zweite Teil ist der Hauptteil des Makros, der mehrere Zeilen enthalten kann und eine Liste von MQSeries-Befehlen enthält. Anfang und Ende des Makrohauptteils sind durch geschweifte Klammern ({ und }) gekennzeichnet.

   Bei der Deklaration eines mehrzeiligen Makrohauptteils wird **MACRO> als MONMQ-Eingabeaufforderung angezeigt. Alle **$n**-Zeichen (wobei N für eine Ziffer steht), wird in der Makrobefehlszeile durch den Parameter **n** ersetzt.

3. Der dritte Teil einer Makrodefinition ist ein kurzer Hilfetext, der bei Eingabe von help <Makroname> angezeigt wird. Der Hilfetext muss in Anführungszeichen gesetzt werden.

Bei der Makrodeklaration müssen zeitliche Überlegungen berücksichtigt werden. Makroprozesse haben zwar eine kurze Ausführungszeit, einige MONMQ-Befehle weisen jedoch einen fernen Prozess zur Ausführung eines Vorgangs an; in diesem Fall wird der nächste Makrobefehl erst nach Abschluss dieses Vorgangs ausgeführt.

Daher sind manchmal Verzögerungen notwendig. Diese können mit dem Befehl **sleep** implementiert werden, in dem über den Parameter **delay=** Verzögerungen in Zehntelsekunden angegeben werden können.

Mit den folgenden Befehlen kann ein Makro erstellt werden, das die Verbindung eines Kanals aufhebt, die Trace-Maske zurücksetzt und den Kanal freigibt.

Anmerkung: Die Angabe mehrere MONMQ-Befehle in einer Zeile ist möglich, diese werden durch ';' getrennt.

```
MQT> declare tmpchl intrange "variable to hold a chl range temporarily"
MQT> macro remove { set tmpchl = chl ; dis chl= $1 ; sleep delay=5
**MACRO> set mask=0xffffffff chl= $1 ; set free chl = $1
**MACRO> set chl=tmpchl
**MACRO> } "A macro to disconnect and free channels Param: chl number"
MQT> help remove

VERB remove:
Ein Makro zum Trennen und Freigeben von Kanälen, Parameter: chl number

Makrotext:
set tmpchl = chl ; dis chl= $1 ; sleep delay=5 ; set mask=0xffffffff chl=$1
; set free chl = $1 ; set chl=tmpchl
MQT>remove 4
ok - process disconnected process 282 from channel 4
```

Trace-Sitzung - Beispiel

In diesem Abschnitt wird eine typische Trace-Sitzung beschrieben, bei der nacheinander die einzelnen MONMQ-Befehle aufgezeigt werden. In diesem Beispiel wird ein Trace für den aktiven Ausführungs-Controller eines WS-Managers und den Haupt-Thread der zugeordneten Agenten ausgeführt.

Vor Ausführung der Trace-Sitzung müssen Sie Folgendes ausführen:

- Starten Sie den WS-Manager, für den ein Trace ausgeführt werden soll: STRMQM BKM3.
- Stellen Sie sicher, dass in `sys$manager:mqs_trace_startup.mqt` neben den vorinstallierten Standardwerten keine zusätzlichen Befehle enthalten sind.
- Stellen Sie sicher, dass der logische Name `MQS_DEF_TRACE` NICHT definiert wurde. Geben Sie andernfalls in MONMQ den Befehl **ONSTARTUP END** ein.

Starten Sie das Dienstprogramm MONMQ.

```
>monmq
```

Daraufhin wird die MONMQ-Eingabeaufforderung angezeigt:

```
MQT>
```

Öffnen Sie eine einzelne LU mit der ID **0**, mit 10 Kanälen und einem Protokollpuffer mit einer Kapazität für 100 Nachrichten. (Für einen normalen Trace wäre dieser Puffer zu wahrscheinlich zu klein. Angemessen wäre eine Kapazität von 1000 Nachrichten.)

```
MQT> open lu=0 nochls=10 buffer=100
 ok - LU:0 opened
```

Zeigen Sie die Definition der LU1 an.

```
MQT> show seg lu=1
```

```
Trace LU            : 1
Mailbox name         : MQS_TRC_MBX_1
Device name          : MBA431:
Status               : Disabled
Mailbox channel      : 384
History buffer size  : 100
Threads mapped #  : 1
Time stamping        : Enabled
Max channels #       : 10
Display depth        : 0
Text filename        :
Binary filename      :
Last status          : 1
Connection map[0]  : 0
=======================================
```

Zeigen Sie MQSeries-Prozesse an:

```
MQT> show process
```

```
PID        Proc_Name  Image     Process     WS_Size  WS_Peak Virt_Peak  Gbl_Pg_Cnt  Prc_Pg_Cnt Total_Mem

2A00023D   BKM1_AG    AMQZLAA0  Agent         23152  16576   203776     3616         12960     16576
2A00023C   BKM1_CI    RUNMQCHI  Run Chan Init 8752   6208    180832     1840         4368      6208
2A00023B   BKM1_RM    AMQRRMFA  Repository Mgr11152  8144    185360     2224         5920      8144
2A00023A   BKM1_CP    AMQZLLP0  Checkpoint    8752   6384    185952     1920         4464      6384
2A000239   BKM1_LG    AMQHASMX  Logger        8752   6288    182016     2080         4208      6288
2A000238   BKM1_EC    AMQZXMA0  EC            20752  15232   203792     3536         11680     15216
2A000128   _FTA4:     MONMQ     MONMQ Utility 8400   8528    198736     2224         3584      5808
                                                                                    ------------------
                                                                                                52112
```

Geben Sie den Ausführungs-Controller und den Agentenprozess an, und verbinden Sie diese mit Kanal 1 bzw. 2.

```
MQT>connect pid=0x238 tid=1 lu=1 chl=1
MQT>connect pid=0x23D tid=1 lu=1 chl=2
```

Überprüfen Sie die Verbindungsdaten.

```
MQT>show channel full connected lu=1
```

```
Pid/Tid              : 2a000bc/1
Status               : *** Connected ***
Process name         : AMQZXMA0.EXE
Assigned LU          : 1
Channel no.          : 1
Mailbox channel      : 800c03f0
Current stack depth  : 4
Circular logging     : Disabled
Next log entry       : 0
Realtime tracing     : Disabled
Time stamping        : Disabled
Trace mask           : ffffffff
Step mode            : Off
No Wait              : On
Last QIO status      : 0
Mapped address       : 1242000-126bfff
=======================================

Pid/Tid              : 2a000c1/1
Status               : *** Connected ***
Process name         : AMQZLAA0.EXE
Assigned LU          : 1
Channel no.          : 2
Mailbox channel      : 800c03f0
Current stack depth  : 4
Circular logging     : Disabled
Next log entry       : 0
Realtime tracing     : Disabled
Time stamping        : Disabled
Trace mask           : ffffffff
Step mode            : Off
No Wait              : On
Last QIO status      : 0
Mapped address       : 1376000-139ffff
=======================================
```

Setzen Sie Standardwerte für die Parameter **chl** (Kanal), **lu** (LU) und **tid** (Thread-ID), damit diese in den nachfolgenden Befehlen nicht jedes Mal neu gesetzt werden müssen.

```
MQT>default chl=1:2 lu=1 tid=1
```

Legen Sie Kanalfarben fest, um die verschiedenen Trace-Nachrichten unterscheiden zu können. Der Parameter **chl** wird in zwei Befehlen angegeben, da bei Verwendung des Standardwertes (1:2) für beide Kanäle Gelb und anschließend Zyanblau festgelegt worden wäre.

```
MQT>set chl=1 color=yellow
MQT>set chl=2 color=cyan
```

Lassen Sie sich jetzt die Kanäle anzeigen.

Trace-Sitzung - Beispiel

```
MQT>show channels
```

```
Chl  Pid/Tid     Mailbox  Stack  History  RTime  Time  Mask      Process Name
1   2a0000bc/1   800c03f0   4       0      0      0    ffffffff  AMQZXMA.EXE
2   2a0000c1/1   800c0360   4       0      0      0    ffffffff  AMQZLAA0.EXE
```

Nachdem beide Prozesse mit Kanälen verbunden sind, können wir jetzt deren Stacks einsehen.

```
 MQT>show stacks
```

```
0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 00:00:00.00 03 - 02 --->| zcpReceiveOnLink
0003 - 00:00:00.00 03 - 03 ---->| xcsWaitEventSem
0004 - 00:00:00.00 03 - 04 ----->| vms_evt

0001- 00:00:00.00 03 - 01 -->| zlaMain
0002 - 00:00:00.00 03 - 02 --->| zcpReceiveOnLink
0003 - 00:00:00.00 03 - 03 ---->| xcsWaitEventSem
0004 - 00:00:00.00 03 - 04 ----->| vms_evt
```

Durch Aktivierung der Vergabe von Zeitmarken für diese beiden Kanäle können wir feststellen, ob ein Prozess blockiert ist oder nicht.

```
MQT>enable timestamp
MQT>show stack
```

```
0001- 00:00:00.00 03 - 01 -->| ExecCtrlrMain
0002 - 12:36:20.18 03 - 02 --->| zcpReceiveOnLink
0003 - 12:36:20.81 03 - 03 ---->| xcsWaitEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt

0001- 00:00:00.00 03 - 01 -->| zlaMain
0002 - 12:36:20.18 03 - 02 --->| zcpReceiveOnLink
0003 - 12:36:20.81 03 - 03 ---->| xcsWaitEventSem
0004 - 12:36:20.83 03 - 04 ----->| vms_evt
```

Aus der Ausgabe geht hervor, dass einige Nachrichten eine gültige Zeitmarke haben. Dies zeigt, dass beide Prozesse aktiv sind. In diesem Fall befinden sich beide Prozesse in einer Ereignisschleife mit einem Zeitlimit von 10 Sekunden. Dieses Zeitlimit kann mit Hilfe des Befehls **show stack** fortlaufend mit der Nachrichtenzeitmarke verglichen werden, bis sich eine Änderung in den Zeitmarkendaten ergibt.

Durch Aktivierung der Protokollierung können wir erzwingen, dass beide Prozesse ihre Trace-Nachrichten in den Ringpuffer schreiben.

```
MQT>enable history
MQT> show history
```

An diesem Punkt werden alle Trace-Nachrichten in den Puffer geschrieben. Dies lässt sich überprüfen, indem Sie die Trace-Maske und die Tabelle mit den Komponenten/Funktionen abrufen.

```
MQT>show mask
```

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -

Trace Mask for Channel 2
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

```
MQT> select
```

```
Chl:1 - Cmp/fnc selection criteria
ALL component/functions
-------------------------------------------
Chl:2 - Cmp/fnc selection criteria
ALL component/functions
-------------------------------------------
```

Trace-Sitzung - Beispiel

```
Trace Mask for Channel 1
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -

Trace Mask for Channel 2
Bit 00 - (fent) function entry
Bit 01 - (fout) function exit
Bit 02 - (ferr) function exit with error
Bit 03 - (fxx) missing function exit
Bit 04 - (dgn) diagnostic messages
Bit 05 - (shm) shared memory
Bit 06 - (spl) spinlocks
Bit 07 - (evt) events
Bit 08 - (mtx) mutexes
Bit 09 - (prc) process msgs
Bit 10 - (msc) miscellaneous
Bit 11 - (inf) informational
Bit 12 -
```

Wir wollen uns jetzt eine bestimmte Nachrichtenart näher ansehen. Nehmen wir beispielsweise an, wir möchten nur Diagnosenachrichten in Bezug auf den gemeinsam benutzten Speicher sehen.

```
MQT>set mask=shm
MQT>show mask
```

Beide Prozesse schreiben jetzt nur Diagnosenachrichten in Zusammenhang mit dem gemeinsam benutzten Speicher in den Puffer. Jetzt wollen wir den Puffer löschen, einige Sekunden warten und anschließend den Inhalt den Puffers erneut abrufen.

```
MQT>clear history

(wait a few seconds)

MQT> show history
```

```
*** Trace History Chl:1 ***

0990 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0991 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0992 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0993 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0994 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0995 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0996 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0997 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0998 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0999 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff

*** End of buffer ***
*** Trace History Chl: ***


*** End of buffer ***
```

Jetzt sollen auch die Ereignisdiagnosenachrichten in der Trace-Ausgabe aufgeführt werden. Nachdem wir die Maske gesetzt haben, müssen wir einige Sekunden warten.

```
MQT>set mask=evt | shm

(wait a few seconds)

MQT>show history
```

Trace-Sitzung - Beispiel

```
*** Trace History Chl:1 ***

0977 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0978 - 00:00:00.00 00 - 04 ......| vms_evt Event BKM3/@ipcc_e_1_2 TIMEOUT
0979 - 00:00:00.00 00 - 04 ......| vms_evt rc = 1
0980 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0981 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0982 - 00:00:00.00 00 - 04 ......| vms_evt Reset on mailbox BKM3/@ipcc_e_1_2 : tout = -1
0983 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0984 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Returning key 1a0
0985 - 00:00:00.00 00 - 04 ......| vms_evt rc = 0
0986 - 00:00:00.00 00 - 04 ......| vms_evt Wait on mailbox BKM3/@ipcc_e_1_2 : tout = 10000
0987 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0988 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Returning key 1a0
0989 - 00:00:00.00 00 - 04 ......| vms_evt Event BKM3/@ipcc_e_1_2 TIMEOUT
0990 - 00:00:00.00 00 - 04 ......| vms_evt rc = 1
0991 - 00:00:00.00 00 - 04 ......| vms_mapgbl key : fffffffe - addr : 0/0
0992 - 00:00:00.00 00 - 04 ......| vms_mapgbl Section MQS1_shm_fffffffe mapped at 1510000-1511fff
0993 - 00:00:00.00 00 - 04 ......| vms_evt Reset on mailbox BKM3/@ipcc_e_1_2 : tout = -1
0994 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0995 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Returning key 1a0
0996 - 00:00:00.00 00 - 04 ......| vms_evt rc = 0
0997 - 00:00:00.00 00 - 04 ......| vms_evt Wait on mailbox BKM3/@ipcc_e_1_2 : tout = 10000
0998 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_2
0999 - 00:00:00.00 00 - 05 .......| vms_get_mbx_chan Returning key 1a0

*** End of buffer ***

*** Trace History Chl:2 ***

0982 - 00:00:00.00 01 - 04 ......| vms_evt Event BKM3/@ipcc_e_1_7 TIMEOUT
0983 - 00:00:00.00 01 - 04 ......| vms_evt rc = 1
0984 - 00:00:00.00 01 - 04 ......| vms_evt Reset on mailbox BKM3/@ipcc_e_1_7 : tout = -1
0985 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0986 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Returning key 1c0
0987 - 00:00:00.00 01 - 04 ......| vms_evt rc = 0
0988 - 00:00:00.00 01 - 04 ......| vms_evt Wait on mailbox BKM3/@ipcc_e_1_7 : tout = 10000
0989 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0990 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Returning key 1c0
0991 - 00:00:00.00 01 - 04 ......| vms_evt Event BKM3/@ipcc_e_1_7 TIMEOUT
0992 - 00:00:00.00 01 - 04 ......| vms_evt rc = 1
0993 - 00:00:00.00 01 - 04 ......| vms_evt Reset on mailbox BKM3/@ipcc_e_1_7 : tout = -1
0994 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0995 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Returning key 1c0
0996 - 00:00:00.00 01 - 04 ......| vms_evt rc = 0
0997 - 00:00:00.00 01 - 04 ......| vms_evt Wait on mailbox BKM3/@ipcc_e_1_7 : tout = 10000
0998 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Getting mbx BKM3/@ipcc_e_1_7
0999 - 00:00:00.00 01 - 05 .......| vms_get_mbx_chan Returning key 1c0

*** End of buffer ***
```

```
MQT>disable history
MQT>clear history
```

Als nächstes wollen wir einen Trace für eine bestimmte Funktion ausführen. Mit dem Befehl **SHOW COMPONENT** und **SHOW FUNCTION** können wir einen Bereich festlegen, für den wir Trace-Informationen wünschen. In unserem Beispiel soll ein Trace für die allgemeine Servicefunktion **xcsRequestMutexSem** ausgeführt werden. Die Komponente ist **0x17**, der Funktionscode **0x1b**. Wir haben bei der Angabe zwei Möglichkeiten:

```
MQT>select comp=0x17 func=0x1b
```

ODER

```
MQT>select fname="xcsRequestMutexSem"
```

Wenn wir jetzt die Protokollierung aktivieren, wird der Puffer keine Einträge enthalten. Dazu müssen wir zunächst die Trace-Maskenbits so setzen, dass alle Einträge angezeigt werden.

```
MQT>enable history
MQT>show history
```

```
*** Trace History Chl:1 ***

*** End of buffer ***
*** Trace History Ch:2 ***


*** End of buffer ***
```

```
MQT>set mask=0xffffffff
MQT>show history
```

Trace-Sitzung - Beispiel

```
*** Trace History Chl:1 ***

0973 - 00:00:00.00 01 - 02 --->| xcsRequestMutexSem
0974 - 00:00:00.00 01 - 03 ---->| xllSemReq
0975 - 00:00:00.00 01 - 04 ----->| vms_mtx
0976 - 00:00:00.00 01 - 04 ......| vms_mtx  :- Locking BKM3_m_1_45 - timeout: -1
0977 - 00:00:00.00 01 - 05 ------>| vms_get_lock
0978 - 00:00:00.00 01 - 05 ------<| vms_get_lock
0979 - 00:00:00.00 01 - 04 -----<| vms_mtx
0980 - 00:00:00.00 01 - 03 ----<| xllSemReq
0981 - 00:00:00.00 01 - 02 ---<| xcsRequestMutexSem
0982 - 00:00:00.00 01 - 03 ---->| xcsRequestMutexSem
0983 - 00:00:00.00 01 - 04 ----->| xllSemReq
0984 - 00:00:00.00 01 - 05 ------>| vms_mtx
0985 - 00:00:00.00 01 - 05 .......| vms_mtx  :- Locking BKM3_m_1_6 - timeout: -1
0986 - 00:00:00.00 01 - 06 ------->| vms_get_lock
0987 - 00:00:00.00 01 - 06 -------<| vms_get_lock
0988 - 00:00:00.00 01 - 05 ------<| vms_mtx
0989 - 00:00:00.00 01 - 04 -----<| xllSemReq
0990 - 00:00:00.00 01 - 03 ----<| xcsRequestMutexSem
0991 - 00:00:00.00 01 - 03 ---->| xcsRequestMutexSem
0992 - 00:00:00.00 01 - 04 ----->| xllSemReq
0993 - 00:00:00.00 01 - 05 ------>| vms_mtx
0994 - 00:00:00.00 01 - 05 .......| vms_mtx  :- Locking BKM3/@ipcc_m_1_18 - timeout: -1
0995 - 00:00:00.00 01 - 06 ------->| vms_get_lock
0996 - 00:00:00.00 01 - 06 -------<| vms_get_lock
0997 - 00:00:00.00 01 - 05 ------<| vms_mtx
0998 - 00:00:00.00 01 - 04 -----<| xllSemReq
0999 - 00:00:00.00 01 - 03 ----<| xcsRequestMutexSem

*** End of buffer ***
```

Jetzt können wir erkennen, dass für beide Prozesse nur ein Trace für die angegebene Funktion und deren untergeordnete Funktionen ausgeführt wird. Mit dem Befehl **select** kann ein Trace für bis zu 8 Komponenten und Funktionen aktiviert werden. Für einen Echtzeit-Trace (d. h., der während der Ausführung angezeigt wird) müssen wir einen Client-Prozess erstellen, damit die Nachrichten für die angegebene LU angezeigt werden können. Dazu geben wird den Befehl **TRACE** ein.

Dieser Befehl startet in dem angegebenen Knoten einen Client-Prozess und wartet auf eingehende Trace-Nachrichten. Trace-Sitzungen in Client-Fenstern können mit Hilfe von MONMQ gesteuert werden.

```
MQT>trace start node="mihell"
```

Aktivieren Sie jetzt den Client-Prozess, um die Nachrichten anzuzeigen, sobald sie eintreffen.

```
MQT>enable trace
```

MQSeries-Threads können der Trace-Ausgabe nach Belieben hinzugefügt oder aus ihr entfernt werden. Die Threads bleiben weiterhin verbunden, die Trace-Daten werden jedoch inaktiviert, d. h., der Trace hat keine Auswirkungen auf die Leistung.

Trace-Sitzung - Beispiel

Die Trace-Funktion kann sofort beim Start eines Prozesses oder Threads aktiviert werden. Dazu wird der Befehl **ONSTARTUP** verwendet, daraufhin wird beim Start eines jeden neuen MQSeries-Prozesses die Trace-Funktion aktiviert.

Trace-Sitzungen werden beendet, indem alle Kanäle inaktiviert und der Client-Prozess beendet wird. Hierzu wird der **close**-Befehl verwendet.

Soll die Trace-Funktion aktiv bleiben, können Sie in MONMQ den Befehl **quit** eingeben, und die Trace-Sitzung zu einem späteren Zeitpunkt fortsetzen.

Beachten Sie, dass es einen Unterschied zwischen der Auswahl des Trace-Maskenbits und der Auswahl von Komponenten und Funktionen gibt. Die Trace-Maskenbits steuern die Ausgabe der Trace-Nachrichtenarten. So sind z. B. Trace-Eintrag und Trace-Ausgabe Nachrichtenarten. Wenn Sie diese inaktivieren, ist der Befehl **select** unabhängig von den angegebenen Parametern wirkungslos, da die Auswahl der Komponente/Funktion von der Auswahl dieser Maskenbits abhängt.

Anhang I. Benutzer-Exits

MQSeries for Compaq OpenVMS unterstützt Exit-Programme für Kanäle und die Datenkonvertierung. Informationen zu Kanal-Exits finden Sie im Handbuch *MQSeries Intercommunication*. Informationen zu Datenkonvertierung-Exits finden Sie in den Handbüchern *MQSeries Application Programming Guide* und *MQSeries Application Programming Reference*.

Die Informationen in diesem Anhang beziehen sich auf die Verwendung von Exit-Programmen in MQSeries for Compaq OpenVMS.

Kanal- und Workload-Exits

Für MQSeries for Compaq OpenVMS ist es nicht erforderlich, eine Verbindung zu einer eigenen Thread-Version eines Exits herzustellen.

MQSeries-Cluster-Workload-Exits

Beim Verbinden eines Workload-Exits auf OpenVMS muss in der Datei mit den Linker-Optionen Folgendes angegeben werden:

```
sys$share:mqm/share
sys$share:mqutl/share
SYMBOL_VECTOR=(clwlFunction=PROCEDURE,MQStart=PROCEDURE)
```

Für Verweise auf das Exit-Abbild ist ein systemweit gültiger logischer Name erforderlich. Lautet der Exit-Name beispielweise SYS$SHARE:AMQSWLM.EXE, sollte der folgende logische Name definiert werden:

```
$DEFINE/SYSTEM/EXEC AMQSWLM SYS$SHARE:AMQSWLM
```

Die Angabe der Dateierweiterung .EXE ist in der logischen Namensdefinition nicht erforderlich.

Damit dieser Name beim Systemstart definiert wird, muss er in SYS$MANAGER: MQS_SYSTARTUP.COM definiert werden.

Anhang J. Sichere Anwendungen

Wenn Leistung und damit Geschwindigkeit ein wichtiger Faktor in Ihrer Umgebung ist und Ihre Umgebung stabil ist, besteht die Möglichkeit, Benutzeranwendungen, Kanäle und Empfangsprogramme als 'sicher' bzw. 'vertrauenswürdig' zu definieren, d. h., dass sie Direktaufrufbindungen verwenden. Dadurch kann auf OpenVMS-Systemen die Verarbeitungszeit von MQPUT- und MQGET-Aufrufen nicht permanenter Nachrichten um bis zu 400 % reduziert werden.

In einer sicheren Anwendung stellen die MQSeries-Anwendung und der lokale WS-Manageragent ein und denselben Prozess dar. Die Anwendung stellt eine direkte Verbindung zu den WS-Managerressourcen her und wird damit zu einer Erweiterung des WS-Managers. Diese Option kann sich allerdings nachteilig auf die Datenintegrität des WS-Managers auswirken, da kein Schutz vor dem Überschreiben des Speicherinhalts besteht.

Darüber kann für sichere Anwendungen unter Umständen die Erstellung bestimmter Ressourcen wie beispielsweise gemeinsam benutzter Speicher erforderlich sein. Dabei kann es unter Umständen notwendig werden, dass ein anderer WS-Managerprozess Zugriff auf diese Ressourcen benötigt und daher demselben UIC (User Identification Code = Benutzeridentifikationscode) zugeordnet sein muss. Die WS-Managerprozesse laufen alle unter demselben MQM-Account, d. h., die sicheren Anwendungen müssen ebenfalls unter diesem ausgeführt werden.

Diese Punkte sollten vor dem Einsatz sicherer Anwendungen bedacht werden.

Benutzeranwendungen

Ihre Anwendung muss nicht direkt unter dem MQM-Account ausgeführt werden. Nachdem eine Verbindung zu einem WS-Manager erfolgreich hergestellt wurde, ändert MQSeries das Sicherheitsprofil des aktiven Thread automatisch, so dass der Thread nun dem MQM-Account zugeordnet ist. Nach einem Aufruf, der die Verbindung vom Thread zum WS-Manager trennt, wird der Thread automatisch wieder unter dem ursprünglichen UIC ausgeführt.

Solange eine sichere Anwendung mit einem WS-Manager verbunden ist, wird sie unter dem MQM-Account ausgeführt. Ist ein Wechsel des UIC erforderlich, während ein Thread mit einem WS-Manager verbunden ist, müssen Sie sicherstellen, dass vor dem nächsten MQI-Aufruf zunächst ein Wechsel zum MQM-Account erfolgt.

Sichere Anwendungen konfigurieren

Für die Ausführung einer sicheren Anwendung in MQSeries for OpenVMS müssen Sie im Feld **Options** des MQCONNX-Aufrufs **MQCNO_FASTPATH_BINDING** angeben. (Für Standardbindungen wird die Option **MQCNO_STANDARD_BINDING** angegeben.) Wenn Sie keine Option angegeben (**MQCNO_NONE**), wird der Standardwert **STANDARD_BINDING** übernommen.

Zusätzlich kann der logische Name **MQ_CONNECT_TYPE** verwendet werden, um den im MQCONNX-Aufruf angegebenen Bindungstyp zu überschreiben. Wird der logische Name definiert, sollte für diesen der Wert **FASTPATH** oder **STANDARD** angegeben werden, damit der erforderliche Bindungstyp ausgewählt wird. In der

Regel wird jedoch der Bindungstyp **FASTPATH** nur verwendet, wenn im MQCONNX-Aufruf die Verbindungsoption entsprechend gesetzt ist. Mit diesem logischen Namen können Sie eine Anwendung bei Problemen mit **FASTPATH-_BINDING** auch mit dem Bindungstyp **STANDARD_BINDING** ausführen, ohne dass die Anwendung erneut erstellt werden muss.

Insgesamt ist für die Ausführung einer sicheren Anwendung Folgendes erforderlich:

- Geben Sie im MQCONNX-Aufruf die Option **MQCNO_FASTPATH_BINDING** an, und definieren Sie **FASTPATH** für den logischen Namen **MQ_CONNECT-_TYPE**.

ODER

- Geben Sie im MQCONNX-Aufruf die Option **MQCNO_FASTPATH_BINDING** an, ohne den logischen Namen **MQ_CONNECT_TYPE** zu definieren.

Weitere Informationen zur Verwendung sicherer Anwendungen finden Sie im Handbuch *MQSeries Intercommunication*.

Kanäle und Empfangsprogramme als sichere Anwendungen ausführen

Kanalprogramme, die mit dem Befehl **runmqsc start channel** gestartet wurden, werden unter dem MQM-Account ausgeführt. Dasselbe gilt auch für Kanalempfängerprogramme, die von eingehenden TCP-Anforderungen oder DECnet-Verbindungsanforderungen gestartet wurden.

Mit den Befehlen **runmqchl** und **runmqlsr** wird ein Hintergrundprozess erstellt, der unter dem MQM-Account ausgeführt wird. Über den logische Namen **MQ_CONNECT_TYPE** und das Attribut **MQIBindType** in der Kanalzeilengruppe in der Datei `qm.ini` eines WS-Managers wird angegeben, ob ein Kanal- oder Empfangsprogramm als sichere Anwendung ausgeführt werden soll.

Sie haben zwei Möglichkeiten, ein sicheres Kanal- oder Empfangsprogramm zu konfigurieren:

- Geben Sie **MQIBindType=FASTPATH** in der Datei `qm.ini` an, und setzen Sie den logischen Namen auf **FASTPATH**.

ODER

- Geben Sie **MQIBindType=FASTPATH** in der Datei `qm.ini` an, ohne den logischen Namen zu definieren.

Schnelle, nicht permanente Nachrichten

Über Kanalattribut **NPMSPEED** können Sie die Übertragungsgeschwindigkeit für nicht permanente Nachrichten festgelegen. Zulässig sind die Werte 'normal' oder 'schnell'. Der Standardwert ist 'schnell'; damit stehen nicht permanente Nachrichten in einem Kanal bereits vor dem nächsten Synchronisationspunkt zur Verfügung. Solche Nachrichten können daher wesentlich schneller abgerufen werden; im Falle eines Leitungsfehlers oder einer Kanalunterbrechung während der Übertragung gehen sie jedoch unter Umständen verloren. Weitere Informationen zur Ausführung von Kanal- und Empfangsprogrammen als sichere Anwendungen sowie zu schnellen, nicht permanenten Nachrichten finden Sie im Handbuch *MQSeries Intercommunication*.

Anhang K. Zusätzliche Informationen

Dieser Anhang enthält Zusatzinformationen, die für die Konfiguration von MQSeries for Compaq OpenVMS erforderlich sind.

Die Informationen in diesem Abschnitt werden bei der nächsten Aktualisierung des betreffenden Handbuchs eingefügt.

Application Programming Guide

Die Informationen zur Programmierung unter OpenVMS werden um den Hinweis erweitert, dass innerhalb einer AST-Routine keine MQI-Aufrufe (Message Queue Interface) möglich sind.

Dies liegt daran, dass die AST-Routinen von MQSeries selbst verwendet werden und nicht ausgeführt werden können, wenn eine andere AST-Routine aktiv ist.

Auslösen von Anwendungen

Die Befehlsdatei MQTRIGGER.COM wird als Beispielbefehlsdatei zur Verfügung gestellt; sie übernimmt die vom MQSeries-Auslösemonitor (RUNMQTRM) übergebenen Parameter und trennt die Felder in der MQTMC2-Struktur.

Die Befehlsdatei erwartet als ersten Parameter das Image bzw. die Befehlsdatei, das/die mit ausgewählten Feldern aus der MQTMC2-Struktur aufgerufen wird.

MQTRIGGER übergibt die folgenden Felder der MQTMC2-Struktur an das aufgerufene Abbild bzw. die Befehlsdatei:

Parameter	MQTMC2-Feld
1	QName
2	ProcessName
3	TriggerData
4	ApplType
5	UserData
6	QMgrName

Beispiele

1. So wird das Abbild amqsech aufgerufen:

   Das Feld **ApplicId** der Auslöseprozessdefinition wird wie folgt angegeben:

```
APPLICID('@mqs_examples:mqtrigger $mqbin:amqsech')
```

   In diesem Beispiel wird davon ausgegangen, dass das logische Verzeichnis MQBIN wie folgt definiert wurde:

```
SYS$SYSROOT:[SYSHLP.EXAMPLES.MQSERIES.BIN]
```

2. So rufen Sie eine Befehlsdatei (dka200:[user]cmd.com) auf:
   Das Feld **ApplicId** der Auslöseprozessdefinition wird wie folgt angegeben:

```
APPLICID('@mqs_examples:mqtrigger @dka200:[user]cmd')
```

Anhang L. Bemerkungen

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

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

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

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

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

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

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

Bemerkungen

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

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

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

Marken

Folgende Namen sind in gewissen Ländern Marken der IBM Corporation:

AIX	IBM
MQSeries	AS/400
MVS/ESA	NetView
CICS	OS/2
First Failure Support Technology	VSE/ESA
OS/390	BookManager

UNIX ist in gewissen Ländern eine eingetragene Marke und wird ausschließlich von der X/Open Company Limited lizenziert.

DIGITAL, OpenVMS, Compaq, DecNet und Alpha sind Marken der Compaq Corporation.

Intel ist in gewissen Ländern eine eingetragene Marke der Intel Corporation.

Microsoft, Windows und das Windows-Logo sind Marken der Microsoft Corporation.

MultiNet und TCPware sind eingetragene Marken der Process Software.

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

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

Literaturverzeichnis

In diesem Abschnitt wird die für alle aktuellen MQSeries-Produkte verfügbare Dokumentation beschrieben.

Plattformübergreifende MQSeries-Veröffentlichungen

Der Großteil dieser Veröffentlichungen (auch als Handbücher zur MQSeries-„Produktfamilie" bezeichnet), bezieht sich auf alle MQSeries-Produkte der Stufe 2. Hierzu gehören die folgenden Produkte:

- MQSeries for AIX Version 5.2
- MQSeries for AS/400, V5.2
- MQSeries for AT&T GIS UNIX, V2.2
- MQSeries for Compaq OpenVMS Alpha V5.1
- MQSeries for Compaq Tru64 UNIX Version 5.1
- MQSeries for HP-UX Version 5.2
- MQSeries for Linux Version 5.2
- MQSeries for OS/2 Warp Version 5.1
- MQSeries for OS/390, V5.2
- MQSeries for SINIX and DC/OSx Version 2.2
- MQSeries for Sun Solaris, V5.2
- MQSeries for Sun Solaris, Intel Platform Edition Version 5.1
- MQSeries for Tandem NonStop Kernel Version 2.2.0.1
- MQSeries for VSE/ESA, V2.1.1
- MQSeries for Windows, V2.0
- MQSeries for Windows, V2.1
- MQSeries for Windows NT and Windows 2000, V5.2

Folgende plattformübergreifende MQSeries-Veröffentlichungen sind verfügbar:

- *MQSeries Brochure* (G511-1908)
- *An Introduction to Messaging and Queuing* (GC33-0805)
- *MQSeries Intercommunication* (SC33-1872)
- *Cluster-Unterstützung in MQSeries* (SC12-2640)
- *MQSeries Clients* (GC33-1632)
- *MQSeries System Administration* (SC33-1873)
- *MQSeries MQSC-Befehle* (SC12-2645)
- *MQSeries Event Monitoring* (SC34-5760)
- *MQSeries Programmable System Management* (SC33-1482)
- *MQSeries Administration Interface Programming Guide and Reference* (SC34-5390)
- *MQSeries Messages* (GC33-1876)
- *MQSeries Application Programming Guide* (SC33-0807)
- *MQSeries Application Programming Reference* (SC33-1673)
- *MQSeries Programming Interfaces Reference Summary* (SX33-6095)
- *MQSeries Using C++* (SC33-1877)
- *MQSeries Using Java* (SC34-5456)
- *MQSeries Application Messaging Interface* (SC34-5604)

Plattformspezifische MQSeries-Veröffentlichungen

Informationen zu den einzelnen MQSeries-Produkten finden Sie nicht nur in den Handbüchern zur MQSeries-Produktfamilie, sondern auch mindestens in einer der betriebssystemspezifischen Veröffentlichungen.

MQSeries for AIX Version 5.2

MQSeries for AIX Einstieg, GC12-2578

MQSeries for AS/400, V5.2

MQSeries for AS/400 V5.1 Quick Beginnings (GC34-5557)

MQSeries for AS/400 V5.1 System Administration (SC34-5558)

MQSeries for AS/400 V5.1 Application Programming Reference (RPG) (SC34-5559)

MQSeries for AT&T GIS UNIX, V2.2

MQSeries for AT&T GIS UNIX System Management Guide (SC33-1642)

MQSeries for Compaq OpenVMS Alpha V5.1

MQSeries for Compaq OpenVMS Alpha Einstieg (GC12-2997)

MQSeries for Compaq OpenVMS Alpha Systemverwaltung (SC12-2998)

MQSeries for Compaq Tru64 UNIX Version 5.1

MQSeries for Compaq Tru64 UNIX Version 5.1 Einstieg (GC12-2798)

MQSeries for HP-UX Version 5.2

MQSeries for HP-UX Einstieg, GC12-2579

MQSeries for Linux Version 5.2

MQSeries for Linux Einstieg (GC12-2779)

MQSeries for OS/2 Warp Version 5.1

MQSeries for OS/2 Warp Version 5.0 Einstieg, GC12-2513

MQSeries for OS/390, V5.2

MQSeries for OS/390 Concepts and Planning Guide (GC34-5650)

MQSeries for OS/390 System Setup Guide (SC34-5651)

MQSeries for OS/390 System Administration Guide (SC34-5652)

MQSeries for OS/390 System Administration Guide (GC34-5892)

MQSeries for OS/390 Messages and Codes (GC34-5891)

MQSeries for OS/390 Licensed Program Specifications (GC34-5893)

MQSeries for OS/390 Program Directory

MQSeries Link for R/3 Version 1.2

MQSeries Link for R/3 Version 1.2 Benutzerhandbuch (GC12-2533)

MQSeries for SINIX and DC/OSx Version 2.2

MQSeries for SINIX and DC/OSx System Management Guide (GC33-1768)

MQSeries for Sun Solaris, V5.2

MQSeries for Sun Solaris Einstieg, GC12-2580

MQSeries for Sun Solaris, Intel Platform Edition Version 5.1

MQSeries for Sun Solaris, Intel Platform Edition Quick Beginnings, GC12-2927

MQSeries for Tandem NonStop Kernel Version 2.2.0.1

MQSeries for Tandem NonStop Kernel System Management Guide (GC33-1893)

MQSeries for VSE/ESA, V2.1.1

MQSeries for VSE/ESA Licensed Program Specifications (GC34-5365)

MQSeries for VSE/ESA System Management Guide (GC34-5364)

MQSeries for Windows, V2.0

MQSeries for Windows V2.0 Benutzerhandbuch (GC12-2418)

MQSeries for Windows, V2.1

MQSeries for Windows V2.1 Benutzerhandbuch (GC12-2515)

MQSeries for Windows NT and Windows 2000, V5.2

MQSeries for Windows NT Einstieg (GC12-2642)

MQSeries for Windows NT Using the Component Object Model Interface (SC34-5387)

MQSeries LotusScript Extension (SC34-5404)

Softcopy-Bücher

Der Großteil der MQSeries-Bücher ist sowohl als Hard- als auch als Softcopy erhältlich.

HTML-Format

Relevante MQSeries-Veröffentlichungen im HTML-Format sind im Lieferumfang der folgenden MQSeries-Produkte enthalten:

- MQSeries for AIX Version 5.2
- MQSeries for AS/400, V5.2
- MQSeries for Compaq OpenVMS Alpha V5.1
- MQSeries for Compaq Tru64 UNIX Version 5.1
- MQSeries for HP-UX Version 5.2
- MQSeries for Linux Version 5.2
- MQSeries for OS/2 Warp Version 5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.2
- MQSeries for Sun Solaris, Intel Platform Edition Version 5.1
- MQSeries for Windows NT and Windows 2000, V5.2 (kompiliertes HTML-Format)
- MQSeries Link for R/3 V1.2

Die MQSeries-Bücher stehen außerdem im HTML-Format auf der Website der MQSeries-Produktfamilie zur Verfügung:

```
http://www.ibm.com/software/mqseries/
```

Bücher im PDF-Format (Portable Document Format)

PDF-Dateien können mit dem Adobe Acrobat Reader angezeigt und gedruckt werden.

Falls Sie den Adobe Acrobat Reader selbst benötigen oder aktuelle Informationen zu den Betriebssystemen, die den Acrobat Reader unterstützen, besuchen Sie die Website der Adobe Systems Inc. unter:

```
http://www.adobe.com/
```

PDF-Versionen relevanter MQSeries-Bücher sind im Lieferumfang der folgenden MQSeries-Produkte enthalten:

- MQSeries for AIX Version 5.2
- MQSeries for AS/400, V5.2
- MQSeries for Compaq OpenVMS Alpha V5.1
- MQSeries for Compaq Tru64 UNIX Version 5.1
- MQSeries for HP-UX Version 5.2
- MQSeries for Linux Version 5.2
- MQSeries for OS/2 Warp Version 5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.2
- MQSeries for Sun Solaris, Intel Platform Edition Version 5.1
- MQSeries for Windows NT and Windows 2000, V5.2
- MQSeries Link for R/3 V1.2

PDF-Versionen der aktuellen MQSeries-Bücher stehen außerdem auf der Website der MQSeries-Produktfamilie zur Verfügung:

`http://www.ibm.com/software/mqseries/`

BookManager-Format

Die MQSeries-Bibliothek steht im IBM BookManager-Format in einer Reihe von CKITs mit Online-Bibliotheken zur Verfügung, einschließlich der CKITs *Transaction Processing and Data* (SK2T-0730). Diese Softcopy-Bücher im IBM BookManager-Format können mit folgenden IBM Lizenzprogrammen angezeigt werden:

BookManager READ/2
BookManager READ/6000
BookManager READ/DOS
BookManager READ/MVS
BookManager READ/VM
BookManager READ für Windows

Bücher im PostScript-Format

Die MQSeries-Bibliothek im PostScript-Format (.PS) ist im Lieferumfang einer Reihe von MQSeries-Produkten der Version 2 enthalten. Bücher in diesem Format können mit einer entsprechenden Anzeigefunktion angezeigt bzw. mit einem PostScript-Drucker gedruckt werden.

Format der Online-Hilfe von Windows

Das *MQSeries for Windows Benutzerhandbuch* wird zusammen mit MQSeries for Windows Version 2.0 und MQSeries for Windows Version 2.1 im Format der Online-Hilfe von Windows geliefert.

Im Internet verfügbare Informationen zu MQSeries

Die Website der MQSeries-Produktfamilie finden Sie unter:

`http://www.ibm.com/software/mqseries/`

Über die Links auf dieser Website können Sie:

- aktuelle Informationen zur MQSeries-Produktfamilie abrufen
- auf MQSeries-Bücher im HTML- und PDF-Format zugreifen
- MQSeries SupportPacs herunterladen

Referenzliteratur

- *Compaq OpenVMS Performance Management*, January 1999

  Dieses Buch enthält Informationen zur Leistungsoptimierung von OpenVMS-Systemen.
- *Compaq OpenVMS System Management Utilities* 2 volumes, January 1999

  Dieses Buch enthält Referenzinformationen zu Systemverwaltungsdienstprogrammen für OpenVMS.
- *Character Data Representation Library, Character Data Representation Architecture, Reference and Registry*, SC09–2190–00

  Dieses Dokument enthält eine Übersicht über CDRA (Character Data Representation Architecture) und definiert die Elemente der Architektur in Form eines Referenzhandbuch.
- *DecNet SNA Gateway for Synchronous Transport Installation (OpenVMS)*, November 1993

  Dieses Handbuch beschreibt die Installation und Konfiguration von DecNet SNA Gateway.
- *Digital SNA APPC/LU6.2 Programming Interface for OpenVMS*, May 1996

  Dieses Handbuch beschreibt die Installation und Konfiguration von SNA APPC/LU6.2.
- *Digital TCP/IP Services for OpenVMS Installation and Configuration*, January 1999

  Dieses Handbuch enthält Anweisungen für die Installation und Konfiguration von Digital TCP/IP.
- *Guidelines for OpenVMS Cluster Configurations*, January 1999

  Dieses Handbuch beschreibt, wie die Verfügbarkeit und Skalierbarkeit von OpenVMS-Clustern optimiert werden kann.

Referenzliteratur

- *Introduction to Compaq Networking and Data Communications*, (Compaq Part No. 093148)

  Dieses Handbuch enthält eine Übersicht über die Netzwerk- und Datenübertragungskonzepte, Tasks, Produkte und Handbücher von Compaq.

Glossar der Begriffe und Abkürzungen

In diesem Glossar werden die MQSeries-spezifischen Begriffe und Abkürzungen erläutert, die im vorliegenden Handbuch verwendet werden. Bei Begriffen, die hier nicht aufgeführt sind, können Sie entweder im Index nachsehen oder im *IBM Dictionary of Computing*, New York: McGraw-Hill (1994) nachschlagen.

In dieses Glossar wurden Begriffe und Definitionen aus dem *American National Dictionary for Information Systems*, ANSI X3.172-1990 (Copyright 1990 by the American National Standards Institute (ANSI)) aufgenommen. Dieses Wörterbuch kann über das American National Standards Institute, 11 West 42 Street, New York, New York 10036, USA, bezogen werden. Definitionen aus diesem Wörterbuch sind mit (A) gekennzeichnet.

.INI-Datei. Siehe *Konfigurationsdatei*.

A

Adapter. Eine Schnittstelle zwischen MQSeries for OS/390 und TSO, IMS, CICS bzw. Stapeladressräumen. Ein Adaptern ist eine Anschlusseinrichtung, über die Anwendungen auf MQSeries-Services zugreifen können.

Administratorbefehle. MQSeries-Befehle zum Verwalten von MQSeries-Objekten wie beispielsweise Warteschlangen, Prozesse oder Namenslisten.

Adressraum. Der virtuelle Speicherbereich, der für einen bestimmten Job zur Verfügung steht.

Adressraumkennung (ASID). Die eindeutige Kennung für einen Adressraum, die vom System vergeben wird.

Aktives Protokoll. Siehe *Wiederherstellungsprotokoll*.

Alert. Eine Nachricht, die zur Identifizierung eines Problems bzw. eines anstehenden Problems an das zentrale Alert-Verarbeitungssystem für Verwaltungsservices gesendet wird.

Alert-Monitor. In MQSeries for OS/390 eine Komponente des CICS-Adapters für die Verarbeitung außerplanmäßiger Ereignisse, die auf Grund einer Verbindungsanforderung an MQSeries for OS/390 auftreten.

Aliaswarteschlangenobjekt. Ein MQSeries-Objekt, bei dessen Namen es sich um den Aliasnamen einer Basiswarteschlange handelt, die im lokalen WS-Manager definiert ist. Bei Verwendung einer Aliaswarteschlange durch eine Anwendung bzw. einen WS-Manager wird der Aliasname aufgelöst und die angeforderte Operation für die entsprechende Basiswarteschlange ausgeführt.

Alternative Benutzer-ID. Eine Sicherheitseinrichtung, bei der die Berechtigung einer Benutzer-ID von einer anderen Benutzer-ID übernommen werden kann (z. B. zum Öffnen von MQSeries-Objekten).

Anforderungsnachricht. Eine Nachrichtenart, mit der eine Antwort von einem anderen Programm angefordert wird. Vgl. *Antwortnachricht* und *Berichtsnachricht*.

Anstehendes Ereignis. Ein außerplanmäßiges Ereignis, das bei Verbindungsanforderungen von einem CICS-Adapter ausgegeben wird.

Antwortnachricht. Eine Nachrichtenart, die als Antwort auf Anforderungsnachrichten verwendet wird. Vgl. *Anforderungsnachricht* und *Berichtsnachricht*.

Anwendungsprotokoll. Unter Windows NT ein Protokoll, in dem wichtige Anwendungsereignisse aufgezeichnet werden.

Anwendungsumgebung. Die Softwareeinrichtungen, auf die ein Anwendungsprogramm Zugriff hat. CICS und IMS sind Beispiele für Anwendungsumgebungen auf OS/390-Plattformen.

Anwendungswarteschlange. Warteschlangen, die von Anwendungen verwendet werden.

Anzeigecursor. Beim Message-Queuing ein Anzeiger, der beim Durchsuchen einer Warteschlange nach der nächsten Nachricht verwendet wird.

Anzeigen (browse). Beim Message-Queuing der Abruf des Inhalts einer Nachricht durch einen MQGET-Aufruf, ohne dass die Nachricht aus der Warteschlange entfernt wird. Siehe auch *GET (Abrufen)*.

APAR. Siehe *Authorized Program Analysis Report*.

Arbeitsbereich für Systemdiagnose (SDWA). Die in einem SYS1.LOGREC-Eintrag enthaltenen Daten, die einen Programm- bzw. Hardwarefehler beschreiben.

Arbeitseinheit. Eine wiederherstellbare Operationsfolge, die von einer Anwendung zwischen zwei konsistenten Zuständen ausgeführt wird. Eine Einheit beginnt mit dem Start einer Transaktion oder im Anschluss an einen vom Benutzer angeforderten Synchronisationspunkt. Sie endet entweder bei einem vom Benutzer

angeforderten Synchronisationspunkt oder beim Abschluss einer Transaktion. Vgl. *Arbeitseinheit mit Wiederherstellung*.

Arbeitseinheit mit Wiederherstellung. Eine wiederherstellbare Operationsfolge in einem einzelnen Ressourcenmanager. Vgl. *Arbeitseinheit*.

Archivprotokoll. Siehe *Wiederherstellungsprotokoll*.

ASID. Address Space IDentifier (siehe *Adressraumkennung*).

Asynchrone Nachrichtenübertragung. Ein Kommunikationsverfahren zwischen verschiedenen Programmen, bei dem von den einzelnen Programmen Nachrichten in Nachrichtenwarteschlangen eingereiht werden. Bei der asynchronen Nachrichtenübertragung können die sendenden Programme mit der Verarbeitung momentan anstehender Vorgänge fortfahren, ohne erst auf eine Antwort auf die abgeschickte Nachricht warten zu müssen. Vgl. *Synchrone Nachrichtenübertragung*.

Attribut. Eines der verschiedenen Merkmale, die die Eigenschaften eines MQSeries-Objekts festlegen.

Auflösungspfad. Die Warteschlangen, die geöffnet werden, wenn eine Anwendung als Eingabe in einem MQOPEN-Aufruf eine Aliaswarteschlange bzw. eine ferne Warteschlange angibt.

Ausgabeparameter. Ein Parameter eines MQI-Aufrufs, über den der WS-Manager Informationen über die erfolgreiche bzw. nicht erfolgreiche Ausführung des Aufrufs zurückgibt.

Ausgabeprotokollpuffer. In MQSeries for OS/390 ein Puffer, der die Datensätze von Wiederherstellungsprotokollen enthält, bevor diese in das Archivierungsprotokoll geschrieben werden.

Auslagerung. In MQSeries for OS/390 ein automatischer Vorgang, bei dem der Inhalt des aktiven Protokolls in das archivierte Protokoll übertragen wird.

Auslöseereignis. Ein Ereignis (z. B. eine Nachricht, die in einer Warteschlange empfangen wird), auf Grund dessen ein WS-Manager eine Auslösenachricht in einer Initialisierungswarteschlange generiert.

Auslösefunktion (Triggering). In MQSeries eine Einrichtung, mit der ein WS-Manager eine Anwendung automatisch starten kann, sobald die hierfür vorgegebenen Bedingungen in einer Warteschlange erfüllt sind.

Auslösemonitor. Eine ständig aktive Anwendung, die für eine oder mehrere Initialisierungswarteschlangen zuständig ist. Die in eine Initialisierungswarteschlange eingehenden Auslösenachrichten werden vom Auslösemonitor abgerufen. Aufgrund der Informationen in der Auslösenachricht startet der Auslösemonitor einen Prozess für die Warteschlange, in der das Auslöseereignis aufgetreten ist.

Auslösemonitorschnittstelle (Trigger Monitor Interface, TMI). Eine MQSeries-Schnittstelle, mit der benutzerdefinierte oder kommerzielle Auslösemonitorprogramme kompatibel sein müssen. Bestandteil von MQSeries Framework.

Auslösenachricht. Eine Nachricht mit Informationen über das Programm, das von einem Auslösemonitor gestartet werden soll.

Authorized Program Analysis Report (APAR). Ein Bericht zu einem Problem, dessen Ursache auf einen vermutlichen Fehler in einem aktuellen, noch nicht geänderten Programm-Release zurückgeführt wird.

B

Basic Mapping Support (BMS, Anzeigeformatierungsunterstützung). Eine Schnittstelle zwischen CICS und Anwendungsprogrammen, die eine Formatierung der Ein- und Ausgabeanzeigedaten vornimmt und mehrseitige Ausgabenachrichten ungeachtet der von den verschiedenen Terminals verwendeten Steuerzeichen weiterleitet.

Beendigungscode. Ein Rückkehrcode, der das Resultat eines MQI-Aufrufs angibt.

Beendigungsmeldung. Ein anstehendes Ereignis, das bei der erfolgreichen Verbindungsherstellung eines CICS-Subsystems mit MQSeries for OS/390 aktiviert wird.

Befehl. In MQSeries eine verwaltungsspezifische Anweisung, die von einem WS-Manager ausgeführt wird.

Befehlspräfix (CPF). In MQSeries for OS/390 eine Zeichenfolge, die den WS-Manager angibt, an den MQSeries for OS/390-Befehle abgesetzt und von dem MQSeries for OS/390-Bedienernachrichten empfangen werden.

Befehlsprozessor. Die MQSeries-Komponente, von der Befehle verarbeitet werden.

Befehlsserver. Die MQSeries-Komponente, die Befehle aus der Eingabewarteschlange für Systembefehle liest, überprüft und gültige Befehle an den Befehlsprozessor weiterleitet.

Benutzer-ID-Service (User Identifier Service, UIS). Eine Einrichtung in MQSeries for OS/2 Warp, die MQI-Anwendungen die Zuordnungen einer anderen Benutzer-ID als der standardmäßigen zu MQSeries-Nachrichten ermöglicht.

Berechtigungsdatei. In MQSeries auf UNIX-Systemen eine Datei mit Sicherheitsdefinitionen für ein Objekt, eine Objektklasse oder alle Objektklassen.

Berechtigungsprüfungen. Sicherheitsprüfungen, die durchgeführt werden, wenn ein Benutzer Verwaltungsbefehle für ein Objekt (z. B. zum Öffnen einer Warteschlange oder zum Aufbau einer Verbindung zu einem WS-Manager) absetzt.

Berechtigungsservice. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 ein Service, der eine Berechtigungsprüfung der Benutzer-IDs ermöglicht, unter denen Befehle und MQI-Aufrufe abgesetzt werden.

Berichtsnachricht. Eine Nachrichtenart, die Informationen zu einer anderen Nachricht enthält. Eine Berichtsnachricht gibt z. B. an, dass eine Nachricht übertragen wurde, bei der Zieladresse eingegangen ist oder aus bestimmten Gründen nicht verarbeitet werden konnte. Vgl. *Antwortnachricht* und *Anforderungsnachricht*.

BMS. Siehe *Basic Mapping Support*.

Bootstrap Data Set (BSDS). Eine VSAM-Datei, die Folgendes enthält:

- eine Liste aller in MQSeries for OS/390 bekannten aktiven und archivierten Protokolldateien.
- eine aktuelle Übersicht über die jeweils letzten Aktivitäten in MQSeries for OS/390.

Die BSDS ist beim Wiederanlauf des MQSeries for OS/390-Subsystems erforderlich.

BSDS. Siehe *Bootstrap Data Set*.

C

CCF. Channel Control Function (siehe *Kanalsteuerfunktion*).

CCSID. Coded Character Set Identifier (siehe *ID für den codierten Zeichensatz*).

CDF. Channel Definition File (siehe *Kanaldefinitionsdatei*).

CI. Control Interval (siehe *Steuerintervall*).

CL. Siehe *Control Language* (Steuersprache).

Client. Eine Laufzeitkomponente, die lokalen Benutzeranwendungen den Zugriff auf Services zur Steuerung von Warteschlangen ermöglicht, die auf einem Server zur Verfügung stehen. Die von den Anwendungen verwendeten Warteschlangen befinden sich auf dem Server. Siehe auch *MQSeries-Client*.

Client-Anwendung. Eine Anwendung, die auf Workstations zur Ausführung kommt und mit einem Client verbunden ist, über den sie Zugriff auf Services zur Steuerung von Warteschlangen erhält, die auf einem Server zur Verfügung stehen.

Cluster. Ein Verbund von WS-Managern, zwischen denen logische Beziehungen bestehen.

connect. Der Vorgang, bei dem die Kennung für eine WS-Managerverbindung zur Verfügung gestellt wird, die anschließend von einer Anwendung in MQI-Aufrufen verwendet wird. Die Verbindung wird entweder durch den MQCONN-Aufruf oder automatisch durch den MQOPEN-Aufruf hergestellt.

Control Language (CL). In MQSeries for AS/400 eine Sprache, mit deren Hilfe Befehle entweder über die Befehlszeile oder über die Erstellung eines CL-Programms ausgegeben werden können.

CPF. Command Prefix (siehe *Befehlspräfix*).

D

DAE. Siehe *Dump Analysis and Elimination*.

Datagramm. Die einfachste Nachricht, die von MQSeries unterstützt wird. Für diese Nachrichtenart wird keine Antwort erwartet.

Datei (Page Set). Eine VSAM-Datei, die in MQSeries for OS/390 bei der Versetzung von Daten aus Puffern im Hauptspeicher (z. B. Warteschlangen und Nachrichten) in einen permanenten Sicherungsspeicher (DASD) verwendet wird.

Datenträger-Image. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 eine Protokollsatzfolge mit dem Image eines Objekts. Das Objekt kann auf der Basis dieses Images erneut erstellt werden.

Datenumsetzungsschnittstelle (DCI). Die MQSeries-Schnittstelle, mit der vom Benutzer bzw. Lieferanten geschriebene Programme zur Umsetzung von Anwendungsdaten zwischen verschiedenen Maschinenverschlüsselungen und CCSIDs kompatibel sein müssen. Bestandteil von MQSeries Framework.

DCE. Siehe *Distributed Computing Environment*.

DCI. Data Conversion Interface (siehe *Datenumsetzungsschnittstelle*).

Dienstprogramm. In MQSeries eine Reihe von Programmen, die dem Systembediener bzw. dem Systemadministrator Funktionen zur Verfügung stellen, die jene der MQSeries-Befehle ergänzen. Mit einigen dieser Dienstprogramme werden mehrere Funktionen aufgerufen.

Distributed Computing Environment (DCE). Eine Middleware, die eine Reihe grundlegender Services zur

Verfügung stellt, die die Entwicklung verteilter Anwendungen erleichtert. DCE wird von der OSF (Open Software Foundation) definiert.

DLQ. Siehe *Warteschlange für nicht zustellbare Nachrichten* (Dead-Letter Queue).

Doppelmodus. Siehe *Doppelte Protokollierung*.

Doppelte Protokollierung. Ein Verfahren zur Protokollierung der Aktivität in MQSeries for OS/390, bei dem jede Änderung in zwei Dateien aufgezeichnet wird; ist eine dieser Dateien bei einem erforderlichen Wiederanlauf nicht lesbar, kann auf die anderen Datei zurückgegriffen werden. Vgl. *Einfache Protokollierung*.

DQM. Siehe *Verteilte Warteschlangenverwaltung* (Distributed Queue Management).

Dump Analysis and Elimination (DAE, Speicherauszugsanalyse und -entfernung). Ein OS/390-Service, mit dessen Hilfe eine Installation unnötige SVC- und ABEND SYSUDUMP-Speicherauszüge verhindern kann, wenn diese mit bereits ausgegebenen Speicherauszügen identisch sind.

Dynamische Warteschlange. Eine lokale Warteschlange, die beim Öffnen eines Modell-WS-Objekts durch ein Programm erstellt wird. Siehe auch *Permanente dynamische Warteschlange* und *Temporäre dynamische Warteschlange*.

E

Ein-/Ausgabeparameter. Ein Parameter eines MQI-Aufrufs, in dem beim Absetzen des Aufrufs Informationen angegeben werden können, die vom WS-Manager je nach Ergebnis des Aufrufs (erfolgreich bzw. nicht erfolgreich) geändert werden.

Einfache Protokollierung. Ein Verfahren bei der Aufzeichnung der Aktivität in MQSeries for OS/390, bei dem jede Änderung nur in eine Datei geschrieben wird. Vgl. *Doppelte Protokollierung*.

Eingabeparameter. Ein Parameter eines MQI-Aufrufs, in dem beim Absetzen des Aufrufs Informationen angegeben werden können.

Einphasige Festschreibung. Ein Verfahren, bei dem ein Programm Aktualisierungen in einer Warteschlange festschreiben kann, ohne dass diese Aktualisierungen mit den Änderungen synchronisiert werden, die vom Programm an Ressourcen vorgenommen wurden, die von einem anderen Ressourcenmanager gesteuert werden. Vgl. *Zweiphasige Festschreibung*.

Einphasige Zurücksetzung. Ein Verfahren, bei dem ein aktiver Vorgang nicht beendet werden darf und bei dem alle Änderungen, die im Verlauf des Vorgangs vorgenommen wurden, zurückgesetzt werden.

Empfängerkanal. Beim Message-Queuing ein Kanal, der auf einen Senderkanal reagiert sowie Nachrichten aus einer Kommunikationsverbindung abruft und in eine lokale Warteschlange einreiht.

Empfangsprogramm. Beim verteilten Queuing in MQSeries ein Programm zum Überwachen eingehender Netzverbindungen.

Ereignis. Siehe *Kanalereignis*, *Instrumentierungsereignis*, *Leistungsereignis* und *WS-Managerereignis*.

Ereignisanzeige. Ein Tool in Windows NT, mit dem Protokolldateien überprüft und verwaltet werden können.

Ereignisdaten. Der Teil der Nachrichtendaten in einer Ereignisnachricht, der Informationen über das Ereignis enthält, z. B. den Namen des WS-Managers sowie der Anwendung, durch die das Ereignis ausgelöst wurde. Siehe auch *Ereignis-Header*.

Ereignis-Header. Der Teil der Nachrichtendaten in einer Ereignisnachricht, der Aufschluss über die Ereignisart des Ursachencodes für das betreffende Ereignis gibt.

Ereignisnachricht. Enthält Informationen (z. B. die Ereigniskategorie, den Namen der Anwendung, von dem das Ereignis ausgelöst wurde, und Statistikdaten für den WS-Manager), die Aufschluss über den Ursprung eines Instrumentierungsereignisses in einem Netz aus MQSeries-Systemen geben.

Ereignisprotokoll. Siehe *Anwendungsprotokoll*.

Ereigniswarteschlange. Die Warteschlange, in die der WS-Manager bei der Entdeckung eines Ereignisses eine entsprechende Ereignisnachricht stellt. Für jede Ereigniskategorie (WS-Manager-, Leistungs- oder Kanalereignis) ist eine eigene Ereigniswarteschlange vorhanden.

Erzwungener Abschluss. Der Abschluss eines CICS-Adapters, bei dem die Verbindung zwischen Adapter und MQSeries for OS/390 sofort und ohne Rücksicht auf den Status der zu diesem Zeitpunkt aktiven Tasks abgebrochen wird. Vgl. *Gesteuerter Abschluss*.

Erzwungener Abschluss. In MQSeries der Systemabschluss eines WS-Managers, ohne eine Verbindungsunterbrechung von Anwendungen bzw. den Abschluss von MQI-Aufrufen abzuwarten. Vgl. *Sofortiger Abschluss* und *Gesteuerter Abschluss*.

ESM. Externe Berechtigungsprüffunktion (siehe *External Security Manager*).

ESTAE. Extended Specify Task Abnormal Exit.

Extended Specify Task Abnormal Exit (ESTAE). Ein OS/390-Makro, das die Fehlerbehebung unterstützt und die Steuerung an die angegebene Exit-Routine zur

Verarbeitung, zur Diagnose einer abnormalen Beendigung oder zur Angabe einer Wiederholungsadresse übergibt.

External Security Manager (ESM). Ein externes Sicherheitsprogramm, das von der OS/390 System Authorization Facility aufgerufen wird. Ein Beispiel für eine ESM ist RACF.

F

Ferner WS-Manager. In bezug auf ein Programm handelt es sich hier um einen WS-Manager, mit dem das Programm nicht in Verbindung steht.

Fernes Warteschlangenobjekt. Siehe *Lokale Definition einer fernen Warteschlange*.

Ferne Warteschlange. Eine Warteschlange, die einem fernen WS-Manager zugeordnet ist. Programme können Nachrichten in ferne Warteschlangen einreihen, jedoch nicht aus ihnen abrufen. Vgl. *Lokale Warteschlange*.

Festschreiben. Ein Vorgang, bei dem alle Änderungen übernommen werden, die während der aktuellen Arbeitseinheit mit Wiederherstellung bzw. der aktuellen Arbeitseinheit vorgenommen wurden. Nach Abschluss dieses Vorgangs beginnt eine neue Arbeitseinheit mit Wiederherstellung bzw. eine neue Arbeitseinheit. Vgl. *Zurücksetzen*.

FFST. First Failure Support Technology.

FIFO. Siehe *First In/First Out*.

First Failure Support Technology (FFST). Dieses Verfahren wird von MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp, MQSeries for Windows NT and Windows 2000 und MQSeries for AS/400 zur Ermittlung und Meldung von Softwareproblemen eingesetzt.

First In/First Out (FIFO). Ein Abrufverfahren im Zusammenhang mit Warteschlangen, bei dem immer zunächst der Eintrag abgerufen wird, der am längsten in der Warteschlange vorhanden ist. (A)

Framework. In MQSeries eine Reihe von Programmierschnittstellen, die Kunden bzw. Lieferanten die Erstellung von Programmen ermöglicht, die bestimmte Funktionen in MQSeries-Produkten erweitern oder ersetzen. Dabei handelt es sich um folgende Schnittstellen:

- MQSeries-Datenumsetzungsschnittstelle (DCI)
- MQSeries-Nachrichtenkanalschnittstelle (MCI)
- MQSeries-Namensserviceschnittstelle (NSI)
- MQSeries-Schnittstelle für Sicherheitsaktivierung (SEI)
- MQSeries-Auslösemonitorschnittstelle (TMI)

FRR. Functional Recovery Routine (siehe *Funktionswiederherstellungsroutine*).

Funktionswiederherstellungsroutine (FRR). Eine OS/390-Verwaltungseinrichtung zur Wiederherstellung/Beendigung, die bei einer Programmunterbrechung für die Übergabe der Steuerung an eine Wiederherstellungsroutine sorgt.

G

GCPC. Siehe *Generalized Command Preprocessor* (allgemeiner Befehlsvorprozessor).

Generalized Command Preprocessor (GCPC). Eine MQSeries for OS/390-Komponente zur Verarbeitung und Ausführung von MQSeries-Befehlen.

Generalized Trace Facility (GTF). Ein OS/390-Serviceprogramm, das für die Fehlerbestimmung wichtige Systemereignisse wie beispielsweise Supervisoraufrufe und E/A-Startvorgänge aufzeichnet.

Gespeicherte Nachricht. Eine Nachricht, die in einen Zusatz(platten)speicher geschrieben wird, damit die Nachricht im Falle eines Systemausfalls nicht verloren geht. Siehe auch *Permanente Nachricht*.

Gesteuerter Abschluss. In MQSeries der Systemabschluss eines WS-Managers, bei dem die Verbindungen aller aktiven Anwendungen ordnungsgemäß beendet werden. Vgl. *Sofortiger Abschluss* und *Erzwungener Abschluss*. Der Abschluss eines CICS-Adapters, bei dem die Verbindung zwischen dem Adapter und MQSeries erst nach Beendigung aller aktiven Tasks unterbrochen wird. Vgl. *Erzwungener Abschluss*.

Gesteuerter Systemabschluss. Siehe *Gesteuerter Abschluss*.

get. Beim Message-Queuing der Abruf einer Nachricht aus einer Warteschlange mit einem MQGET-Aufruf. Siehe auch *Anzeigen (browse)*.

Globaler Trace. Eine Trace-Option in MQSeries for OS/390, bei der Trace-Daten aus dem gesamten MQSeries for OS/390-Subsystem protokolliert werden.

Gruppe von Registrierungsschlüsseln. In Windows NT die Struktur der in der Registrierung gespeicherten Daten.

GTF. Siehe *Generalized Trace Facility* (allgemeine Trace-Einrichtung).

I

ID des codierten Zeichensatzes (CCSID). Der Name eines codierten Zeichensatzes sowie die entsprechenden Codepunktzuordnungen.

Initialisierungseingabedatensätze. Dateien, die beim Start von MQSeries for OS/390 verwendet werden.

Initialisierungswarteschlange. Eine lokale Warteschlange, in die der WS-Manager Auslösenachrichten stellt.

Installierbare Services. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 zusätzliche Funktionen, die als voneinander unabhängige Komponenten zur Verfügung gestellt werden. Die Installation dieser Komponenten ist wahlfrei; statt dessen können auch eigene Komponenten oder Komponenten anderer Hersteller verwendet werden. Siehe auch *Berechtigungsservice*, *Namensservice* und *Benutzer-ID-Service*.

Instrumentierungsereignis. Eine Einrichtung, die die Überwachung des WS-Manager-Betriebs in einem Netz aus MQSeries-Systemen ermöglicht. MQSeries stellt Instrumentierungsereignisse zur Überwachung von Ressourcendefinitionen von WS-Managern, des Leistungsstatus und des Kanalstatus zur Verfügung. Die Instrumentierungsereignisse können von benutzerdefinierten Meldeverfahren in Verwaltungsanwendungen verwendet werden, die dem Systembediener aufgetretene Ereignisse anzeigen. Darüber hinaus ermöglichen sie es, dass Anwendungen von anderen Verwaltungsnetzen als Agenten für die Überwachung von Meldungen und die Erstellung der entsprechenden Alerts eingesetzt werden können.

Interactive Problem Control System (IPCS, interaktives Problemsteuersystem). Eine OS/390-Komponente, die die Online-Fehlerverwaltung, interaktive Fehlerdiagnose, Online-Fehlerbehebung für plattenresidente Speicherauszüge nach einem Absturz, Fehlerverfolgung und Fehlermeldung ermöglicht.

Interactive System Productivity Facility (ISPF, interaktive Einrichtung für Systemleistung). Ein IBM Lizenzprogramm, das als Gesamtanzeigeeditor und Dialogmanager eingesetzt wird. Dieses Programm wird für die Erstellung von Anwendungsprogrammen verwendet und ermöglicht die Generierung standardmäßiger Bildschirmanzeigen und interaktiver Dialoge zwischen Anwendungsprogrammierer und Terminalbenutzer.

IPCS. Siehe *Interactive Problem Control System*.

ISPF. Siehe *Interactive System Productivity Facility*.

K

Kanal. Siehe *Nachrichtenkanal*.

Kanaldefinitionsdatei (CDF). In MQSeries eine Datei mit Übertragungskanaldefinitionen, über die den Übertragungswarteschlangen Kommunikationsverbindungen zugeordnet werden.

Kanalereignis. Ein Ereignis, das anzeigt, dass ein Kanalexemplar jetzt zur Verfügung steht bzw. nicht mehr verfügbar ist. Kanalereignisse werden in den WS-Managern an beiden Kanalenden generiert.

Kanal für Client-Verbindungen. Eine MQI-Kanaldefinition, die einem MQSeries-Client zugeordnet ist. Siehe auch *Kanal für Serververbindungen*.

Kanal für Serververbindungen. Eine MQI-Kanaldefinition, die dem Server zugeordnet ist, auf dem ein WS-Manager läuft. Siehe auch *Kanal für Client-Verbindungen*.

Kanalsteuerfunktion (Channel Control Function, CCF). In MQSeries ein Programm zum Übertragen von Nachrichten aus einer Übertragungswarteschlange an eine Kommunikationsverbindung und von dieser in eine lokale Warteschlange; dazu gehört eine Steuerkonsolschnittstelle für den Aufbau und die Steuerung von Kanälen.

Kennung. Siehe *Verbindungskennung* und *Objektkennung*.

Konfigurationsdatei. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 eine Datei, die Konfigurationsdaten zum Beispiel zu Protokollen, zur Kommunikation oder zu installierbaren Services enthält. Synonym für *.ini-Datei*. Siehe auch *Zeilengruppe*.

Kontext. Informationen über den Ursprung einer Nachricht.

Kontextsicherheit. In MQSeries ein sicherungstechnisches Verfahren, bei dem Nachrichten im Nachrichtendeskriptor Angaben zu ihrem Ursprung enthalten müssen.

L

Ländereinstellung. Auf UNIX-Systemen eine Untergruppe der Benutzerumgebung, die die Konventionen für einen bestimmten Kulturkreis definiert (z. B. das Zeit- und Währungsformat, die Zahlendarstellung sowie Zeichenklassifikation, -folge und -umsetzung). Die CCSID des WS-Managers wird von den länderspezifischen Angaben der Benutzer-ID abgeleitet, unter der der WS-Manager erstellt wurde.

Leistungsereignis. Eine Ereignisart, die bei Auftreten einer Grenzbedingung ausgegeben wird.

Leistungs-Trace. Eine Trace-Option in MQSeries, bei der Trace-Daten für die Leistungsanalyse und -optimierung herangezogen werden.

Lineare Protokollierung. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 das Speichern von Daten für den Wiederanlauf in einer Dateifolge. Je nach Bedarf werden neue Dateien erstellt. Der Speicherbereich, in den die Daten geschrieben werden, wird erst

nach einem Wiederanlauf des WS-Managers freigegeben. Vgl. *Zyklische Protokollierung*.

Logische Arbeitseinheit (LUW). Siehe *Arbeitseinheit*.

Lokale Definition. Ein MQSeries-Objekt, das einem lokalen WS-Manager zugeordnet ist.

Lokale Definition einer fernen Warteschlange. Ein MQSeries-Objekt, das einem lokalen WS-Manager zugeordnet ist. Dabei definiert dieses Objekt die Attribute einer Warteschlange, die einem anderen WS-Manager zugeordnet ist. Darüber hinaus wird dieses Objekt zur Aliasnamensumsetzung für WS-Manager und für Warteschlangen für Antwortnachrichten verwendet.

Lokaler WS-Manager. Der Warteschlangenmanager, zu dem eine Verbindung mit einem Programm besteht und der diesem Programm Services für das Message-Queuing zur Verfügung stellt. Warteschlangenmanager, zu denen für ein Programm keine Verbindungen bestehen, werden als *ferne WS-Manager* bezeichnet, auch wenn sie auf demselben System wie das Programm aktiv sind.

Lokale Warteschlange. Eine Warteschlange, die zu einem lokalen WS-Manager gehört. Eine lokale Warteschlange kann eine Liste der Nachrichten enthalten, die zur Verarbeitung anstehen. Vgl. *ferne Warteschlange*.

M

MCA. Siehe *Nachrichtenkanalagent* (Message Channel Agent).

MCI. Message Channel Interface (siehe *Nachrichtenkanalschnittstelle*).

Message-Queuing. Eine Programmiertechnik, bei der jedes Programm innerhalb einer Anwendung mit anderen Programmen kommuniziert, indem Nachrichten in Warteschlangen eingereiht werden.

Modellwarteschlangenobjekt. Eine Gruppe von Warteschlangenattributen, die bei Erstellung einer dynamischen Warteschlange durch das Programm als Schablone dienen.

MQAI. MQSeries Administration Interface (siehe *MQSeries-Verwaltungsschnittstelle*).

MQI. Siehe *Schnittstelle für Nachrichtenwarteschlangen*.

MQI-Kanal. Stellt die Verbindung zwischen einem MQSeries-Client und einem WS-Manager auf einem Serversystem her; über diesen Kanal werden nur MQI-Aufrufe und Antworten in beide Richtungen übertragen. Vgl. *Nachrichtenkanal*.

MQSC. MQSeries-Befehle (MQSeries Commands).

MQSeries. Eine Produktfamilie lizenzierter IBM Programme, die Message-Queuing-Services zur Verfügung stellen.

MQSeries-Befehle (MQSC). Benutzerlesbare Befehle, mit deren Hilfe die Bearbeitung von MQSeries-Objekten möglich ist. Vgl. *Programmable Command Format (PCF)*.

MQSeries-Client. Eine Komponente des MQSeries-Produkts, die auf einem System installiert werden kann, ohne dass ein vollständiger WS-Manager installiert sein muss. Der MQSeries-Client empfängt MQI-Aufrufe von Anwendungen und kommuniziert mit einem WS-Manager auf einem Serversystem.

MQSeries-Verwaltungsschnittstelle (MQSeries Administration Interface, MQAI). Eine Programmierschnittstelle in MQSeries.

N

Nachricht. Bei Anwendungen für das Message-Queuing Informationen, die zwischen Programmen ausgetauscht werden. Siehe auch *Permanente Nachricht* und *Nicht permanente Nachricht*. Bei der Systemprogrammierung handelt es sich hier um Informationen für den Terminalbediener bzw. den Systemadministrator.

Nachrichtendeskriptor. Steuerinformationen, die Nachrichtenformat und -darstellung beschreiben und als Teil der MQSeries-Nachricht übertragen werden. Das Format des Nachrichtendeskriptors hängt von der MQMD-Struktur ab.

Nachrichtenfolgenummerierung. Ein Programmierverfahren, bei dem Nachrichten während ihrer Übertragung über eine Kommunikationsverbindung eindeutige Nummern zugeordnet werden. Der empfangende Prozess kann anhand dieser Nummern feststellen, ob alle Nachrichten eingegangen sind, diese in ihrer ursprünglichen Reihenfolge in eine Warteschlange stellen und doppelt vorhandene Nachrichten löschen.

Nachrichtenkanal. Beim verteilten Message-Queuing ein Verfahren zur Weiterleitung von Nachrichten zwischen den verschiedenen WS-Managern. Ein Nachrichtenkanal besteht aus zwei Nachrichtenkanalagenten (ein Sender an einem, ein Empfänger am anderen Ende) sowie einer Kommunikationsverbindung. Vgl. *MQI-Kanal*.

Nachrichtenkanalagent (MCA, Message Channel Agent). Ein Programm, das vorbereitete Nachrichten aus einer Übertragungswarteschlange an eine Kommunikationsverbindung bzw. von einer Kommunikationsverbindung an eine Bestimmungswarteschlange überträgt. Siehe auch *Schnittstelle für Nachrichtenwarteschlangen (MQI)*.

Nachrichtenkanalschnittstelle (Message Channel Interface, MCI). Die MQSeries-Schnittstelle, mit der benutzerdefinierte bzw. kommerzielle Programme kompatibel sein müssen, die Nachrichten zwischen dem MQSeries-WS-Manager und anderen Nachrichtenübertragungssystemen übertragen. Bestandteil von MQSeries Framework.

Nachrichtenpriorität. In MQSeries ein Nachrichtenattribut, das die Abrufreihenfolge aus einer Warteschlange festlegt und das angibt, ob ein Auslöseereignis generiert wird.

Nachrichtenübertragung. Siehe *Synchrone Nachrichtenübertragung* und *Asynchrone Nachrichtenübertragung*.

Nachrichtenwarteschlange. Synonym für *Warteschlange*.

Namensliste. Ein MQSeries-Objekt, das eine Liste mit Namen (z. B. Warteschlangennamen) enthält.

Namensservice. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 eine Einrichtung, mit der festgelegt wird, welcher WS-Manager Eigner einer bestimmten Warteschlange ist.

Namensserviceschnittstelle (NSI). Eine MQSeries-Schnittstelle, mit der benutzerdefinierte oder kommerzielle Programme, die das Eigentumsrecht an Warteschlangennamen auflösen, kompatibel sein müssen. Bestandteil von MQSeries Framework.

Namensumsetzung. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 ein interner Prozess, der den Namen eines WS-Managers in einen eindeutigen und in dem verwendeten System zulässigen Namen umsetzt. Extern bleibt der Name des WS-Managers unverändert.

New Technology File System (NTFS). Ein wiederherstellbares Dateisystem in Windows NT, das Dateisicherheit bietet.

Nicht permanente Nachricht. Eine Nachricht, die nach dem Wiederanlauf eines WS-Managers nicht mehr vorhanden ist. Vgl. *Permanente Nachricht*.

NSI. Name Service Interface (siehe *Namensserviceschnittstelle*).

NTFS. Siehe *New Technology File System*.

Nullzeichen. Das Zeichen, das mit X'00' dargestellt wird.

O

OAM. Siehe *Object Authority Manager*.

Object Authority Manager (OAM, Objektberechtigungsmanager). In MQSeries auf UNIX-Systemen, MQSeries for AS/400 und MQSeries for Windows NT and Windows 2000 der standardmäßige Berechtigungsservice für die Befehls- und Objektverwaltung. OAM kann durch benutzereigene Sicherheitsservices ersetzt oder in Kombination mit diesen eingesetzt werden.

Objekt. In MQSeries handelt es sich bei einem Objekt um einen WS-Manager, eine Warteschlange, eine Prozessdefinition, einen Kanal, eine Namensliste oder um eine Speicherklasse (nur in OS/390).

Objektdeskriptor. Eine Datenstruktur zur Beschreibung eines bestimmten MQSeries-Objekts. Der Deskriptor enthält Objektname und -art.

Objektkennung. Die Kennung bzw. der Token, über die bzw. den ein Programm Zugriff auf das MQSeries-Objekt erhält, mit dem es arbeitet.

P

PCF. Programmable Command Format.

PCF-Befehl. Siehe *Programmable Command Format*.

Permanente dynamische Warteschlange. Eine dynamische Warteschlange, die nach dem Schließen nur gelöscht wird, wenn dies explizit gefordert wird. Permanente dynamische Warteschlangen werden im Anschluss an einen WS-Managerausfall wiederhergestellt, so dass sie auch permanente Nachrichten enthalten können. Vgl. *Temporäre dynamische Warteschlange*.

Permanente Nachricht. Eine Nachricht, die auch nach dem Wiederanlauf eines WS-Managers noch vorhanden ist. Vgl. *Nicht permanente Nachricht*.

Ping. Bei der verteilten Steuerung von Warteschlangen eine Diagnosehilfe, die über eine Testnachricht feststellt, ob ein Nachrichtenkanal bzw. eine TCP/IP-Verbindung einwandfrei arbeitet.

Plattform. In MQSeries das Betriebssystem, unter dem ein WS-Manager ausgeführt wird.

Principal. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 eine Benutzer-ID. Der Principal wird vom Objektberechtigungsmanager bei der Überprüfung der Zugriffsberechtigung auf Systemressourcen verwendet.

Programmable Command Format (PCF, programmierbares Befehlsformat). Eine MQSeries-Nachrichtenart, die von folgenden Komponenten verwendet wird:

- von Anwendungen für die benutzerspezifische Verwaltung, um PCF-Befehle in die systemspezifische Befehlseingabewarteschlange eines bestimmten WS-Managers einzureihen;

- von Anwendungen für die benutzerspezifische Verwaltung, um das Ergebnis eines PCF-Befehls aus einem bestimmten WS-Manager abzurufen;
- von einem WS-Manager, um auf das Auftreten eines Ereignisses hinzuweisen.

Vgl. *MQSC*.

Protokoll. Eine Datei in MQSeries, in der die Aktivitäten der WS-Manager beim Empfangen, Übertragen und Zustellen von Nachrichten aufgezeichnet wird. Dies ermöglicht bei Auftreten eines Fehlers die Wiederherstellung.

Protokolldatei. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 eine Datei, in der alle wichtigen Änderungen an den von einem WS-Manager gesteuerten Daten aufgezeichnet werden. Wenn die primären Protokolldateien voll sind, werden von MQSeries sekundäre Protokolldateien angelegt.

Protokollsteuerdatei. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 die Datei mit den Informationen, die zur Überwachung der Verwendung von Protokolldateien herangezogen werden (z. B. Größe, Verzeichnispfad und Namen der nächsten verfügbaren Datei).

Prozessdefinitionsobjekt. Ein MQSeries-Objekt, das die Definition einer MQSeries-Anwendung enthält. Diese Definition wird beispielsweise von einem WS-Manager bei der Verwendung von Auslösenachrichten verwendet.

Prüfpunkt. Der Zeitpunkt, zu dem wichtige Informationen in das Protokoll geschrieben werden. Vgl. *Synchronisationspunkt*. In MQSeries auf UNIX-Systemen der Zeitpunkt, zu dem ein Datensatz im Protokoll mit dem Datensatz in der Warteschlange identisch ist. Prüfpunkte werden automatisch generiert und werden beim Systemneustart verwendet.

PTF. Program Temporary Fix (siehe *Vorläufige Programmkorrektur*).

Puffer-Pool. Ein Bereich im Hauptspeicher, der für Warteschlangen, Nachrichten und Objektdefinitionen von MQSeries for OS/390 verwendet wird. Siehe auch *Datei (Page Set)*.

Q

Qualifikationsmerkmal der oberen Ebene für Zielbibliothek (thlqual). Ein Qualifikationsmerkmal der oberen Ebene für OS/390-Zieldateinamen.

R

RBA. Relative Byte Address (siehe *Relative Byteadresse*).

Recovery Termination Manager (RTM). Ein Programm, das die normale und abnormale Beendigung von Tasks bearbeitet, indem es die Steuerung an eine Wiederherstellungsroutine übergibt, die der Beendigungsfunktion zugeordnet ist.

Regeltabelle. Eine Steuerdatei mit einer oder mehreren Regeln, die von der Steuerroutine für Warteschlangen für nicht zustellbare Nachrichten auf die Nachrichten in dieser Warteschlange angewandt werden.

Registrierungsdatenbank. In Windows NT eine sichere Datenbank, die eine zentrale Quelle für die Konfigurationsdaten des Systems und von Anwendungen darstellt.

Registrierungseditor. In Windows NT die Programmkomponente, die dem Benutzer die Bearbeitung der Registrierung ermöglicht.

Relative Byteadresse (RBA). Der Abstand in Byte eines gespeicherten Datensatzes bzw. Steuerintervalls vom Beginn des Speicherbereichs, der der zugehörigen Datei zugeordnet ist.

Requester-Kanal. Beim Message-Queuing ein Kanal, der von einem Senderkanal fern gestartet werden kann. Der Requester-Kanal empfängt über eine Kommunikationsverbindung Nachrichten von dem Senderkanal und reiht diese in die lokale Warteschlange ein, die in den Nachrichten angegeben ist. Siehe auch *Serverkanal*.

RESLEVEL. Eine Option in MQSeries for OS/390, über die die Anzahl an CICS-Benutzer-IDs festgelegt wird, die hinsichtlich der Sicherheit der API-Ressourcen in MQSeries for OS/390 geprüft werden.

Resource Recovery Services (RRS). Eine OS/390-Einrichtung, die einen zweiphasigen Synchronisationspunkt über alle beteiligten Ressourcenmanager hinweg unterstützt.

Responder. Bei der verteilten Steuerung von Warteschlangen ein Programm, das auf Netzverbindungsanforderungen anderer Systeme antwortet.

Ressource. Die vom Rechnersystem bzw. dem Betriebssystem zur Verfügung gestellten Funktionen, die für die Ausführung eines Jobs bzw. einer Task erforderlich sind. In MQSeries for OS/390 handelt es sich bei Ressourcen z. B. um Pufferpools, Dateien, Protokolldateien, Warteschlangen oder Nachrichten.

Ressourcenmanager. Eine Anwendung, ein Programm oder eine Transaktion, die den Zugriff auf gemeinsam benutzte Ressourcen (z. B. Speicherpuffer oder Dateien) steuert. MQSeries, CICS und IMS sind Ressourcenmanager.

Resynchronisation. In MQSeries eine Option, mit der ein Kanal gestartet und angewiesen werden kann, alle

unbestätigten Statusnachrichten ohne einen Neustart der Nachrichtenübertragung aufzulösen.

ROLLBACK-Operation. Synonym für *Zurücksetzen*.

RRS. Siehe *Resource Recovery Services*.

RTM. Siehe *Recovery Termination Manager*.

Rückkehrcodes. Sammelbegriff für Beendigungscode und Ursachencode.

Rückruf. In MQSeries leitet ein Requester-Nachrichtenkanal Übertragungen von einem Senderkanal dadurch ein, dass er zunächst den Sender aufruft. Anschließend wird der Requester-Nachrichtenkanal beendet und wartet auf einen Rückruf.

S

SAF. Siehe *System Authorization Facility*.

Schnittstelle für Nachrichtenwarteschlangen (MQI, Message Queue Interface). Die von MQSeries-WS-Managern zur Verfügung gestellte Programmierschnittstelle. Mit dieser Programmierschnittstelle erhalten Anwendungsprogramme Zugriff auf die Message-Queuing-Services.

Schnittstelle für Sicherheitsaktivierung (Security Enabling Interface, SEI). Eine Schnittstelle in MQSeries, mit der benutzerdefinierte oder kommerzielle Programme, die Berechtigungen prüfen, Benutzer-IDs zur Verfügung stellen oder Authentifizierungsvorgänge durchführen, kompatibel sein müssen. Bestandteil von MQSeries Framework.

SDWA. System Diagnostic Work Area (siehe *Arbeitsbereich für Systemdiagnose*).

SEI. Security Enabling Interface (siehe *Schnittstelle für Sicherheitsaktivierung*).

Senderkanal. Beim Message-Queuing ein Kanal, der Übertragungen einleitet sowie Nachrichten aus Übertragungswarteschlangen entfernt und über eine Kommunikationsverbindung in einen Empfänger- oder Requester-Kanal stellt.

Sequenzielle Zustellung. In MQSeries ein Verfahren, bei dem Nachrichten mit einer Folgenummer übertragen werden; dies ermöglicht es dem Empfängerkanal, die Nachrichten in der richtigen Reihenfolge zu speichern. Dies ist erforderlich, wenn Nachrichten nur einmal und in der richtigen Reihenfolge übertragen werden können.

Server. (1) In MQSeries ein WS-Manager, der einer Client-Anwendung, die auf einer fernen Workstation ausgeführt wird, Warteschlangenservices zur Verfügung stellt. (2) Ein Programm, das in einer Client/Server-Umgebung (ein Zweiprogrammmodell, bei dem zwischen beiden Programmen Informationen ausgetauscht werden) auf Informationsanforderungen antwortet. Siehe auch *Client*.

Serverkanal. Beim Message-Queuing ein Kanal, der auf einen Requester-Kanal reagiert sowie Nachrichten aus Übertragungswarteschlangen entfernt und über eine Kommunikationsverbindung in einem Requester-Kanal stellt.

Serviceintervall. Ein Zeitintervall, mit dem die abgelaufene Zeit zwischen einem PUT- bzw. GET-Vorgang und einem anschließenden GET-Vorgang vom WS-Manager verglichen wird. Anhand dieses Wertes entscheidet der WS-Manager ob ein Serviceintervallereignis ausgegeben wird oder nicht. Das Serviceintervall für eine Warteschlange wird über ein Warteschlangenattribut angegeben.

Serviceintervallereignis. Ein Ereignis, das in Zusammenhang mit dem Serviceintervall ausgegeben wird.

Signalübertragung. In MQSeries for OS/390 und MQSeries for Windows 2.1 eine Funktion, die es dem Betriebssystem ermöglicht, einem Programm die Ankunft einer erwarteten Nachricht in einer Warteschlange zu melden.

SIT. System Initialization Table (siehe *Systeminitialisierungstabelle*).

Sitzungs-ID. In MQSeries for OS/390 die in CICS eindeutige ID für die Kommunikationsverbindung, die von einem Nachrichtenkanalagenten bei der Übertragung von Nachrichten aus einer Übertragungswarteschlange in eine Verbindung verwendet werden soll.

Sofortiger Abschluss. In MQSeries der Systemabschluss eines WS-Managers, ohne eine Verbindungsunterbrechung der Anwendungen abzuwarten. Aktuelle MQI-Aufrufe können abgeschlossen werden, neue MQI-Aufrufe hingegen können bei Anforderung eines sofortigen Abschlusses nicht ausgeführt werden. Vgl. *Gesteuerter Abschluss* und *Erzwungener Abschluss*.

Speicherklasse. In MQSeries for OS/390 definiert eine Speicherklasse die Datei (Page Set), in der die Nachrichten für eine bestimmte Warteschlange gespeichert werden. Die Speicherklasse wird bei der Definition der Warteschlange angegeben.

Speichern und weiterleiten. Die temporäre Speicherung von Datenpaketen, Nachrichten oder Blöcken in einem Datennetz, bevor sie an die Zieladresse weitergeleitet werden.

Standardobjekt. Die Definition eines Objekts (z. B. einer Warteschlange), in der alle Attribute angegeben sind. Bei der Definition eines Objekts durch den Benutzer, bei der nur Angaben zu bestimmten Attributen erfolgen, übernimmt der WS-Manager für alle nicht angegebenen Attribute die Standardwerte.

Steuerbefehl. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 ein Befehl, der interaktiv in einer Befehlszeile des Betriebssystems eingegeben werden kann. Einzige Voraussetzung für diese Befehle ist die Installation des MQSeries-Produkts; darüber hinaus sind keine besonderen Dienstprogramme zur Ausführung dieser Befehle erforderlich.

Steuerintervall (Control Interval, CI). Ein Bereich im Direktzugriffsspeicher mit fester Länge, in dem von VSAM Datensätze gespeichert werden und freier Speicherbereich erstellt wird. Beim Steuerintervall handelt es sich um eine Informationseinheit, die von VSAM an den bzw. aus dem Direktzugriffsspeicher übertragen wird.

Steuerroutine der Warteschlange für nicht zustellbare Nachrichten. Ein von MQSeries zur Verfügung gestelltes Dienstprogramm zur Überwachung von Warteschlangen für nicht zustellbare Nachrichten und zur Verarbeitung von Nachrichten in der Warteschlange anhand einer benutzerdefinierten Regeltabelle.

Steuerung von fernen Warteschlangen. Beim Message-Queuing Services, die es Anwendungen ermöglichen, Nachrichten in Warteschlangen einzureihen, die anderen WS-Managern zugeordnet sind.

Steuerung von Warteschlangen (Queuing). Siehe *Message-Queuing*.

Stilllegung. In MQSeries der Status eines WS-Managers, bevor er gestoppt wird. In diesem Status können alle Programme ordnungsgemäß beendet werden, es können jedoch keine neuen Programme gestartet werden.

Subsystem. In OS/390 eine Gruppe von Modulen, die eine Funktion zur Verfügung stellt, die von OS/390 unterstützt wird. Bei MQSeries for OS/390 beispielsweise handelt es sich um ein OS/390-Subsystem.

Supervisor-Aufruf (SVC). Eine OS/390-Anweisung, die ein laufendes Programm unterbricht und die Steuerung an den Supervisor übergibt, damit dieser den in der Anweisung angegebenen Service ausführen kann (siehe *SVC*).

SVC. Supervisor Call (siehe *Supervisor-Aufruf*).

Switch-Profil. In MQSeries for OS/390 ein RACF-Profil, das beim Start von MQSeries oder bei Eingabe eines Befehls zur Sicherheitsaktualisierung verwendet wird. Jedes von MQSeries ermittelte Switch-Profil inaktiviert die Überprüfung der angegebenen Ressource.

Symptomzeichenfolge. Diagnoseinformationen, die in einem strukturierten Format angezeigt werden, das für die Suche in der Datenbank der IBM Softwareunterstützung konzipiert wurde.

Synchrone Nachrichtenübertragung. Ein Kommunikationsverfahren zwischen verschiedenen Programmen, bei dem von den einzelnen Programmen Nachrichten in Nachrichtenwarteschlangen eingereiht werden. Bei der synchronen Nachrichtenübertragung können die sendenden Programme mit der Verarbeitung momentan anstehender Vorgänge erst fortfahren, nachdem eine Anwort auf die Nachricht eingegangen ist. Vgl. *Asynchrone Nachrichtenübertragung*.

Synchronisationspunkt. Ein Punkt im Verlauf bzw. am Ende der Verarbeitung einer Transaktion, an dem sich die geschützten Ressourcen der Transaktion in einem konsistenten Zustand befinden. An einem Synchronisationspunkt können Änderungen an den Ressourcen ohne Bedenken festgeschrieben oder auf den vorherigen Synchronisationspunkt zurückgesetzt werden.

SYS1.LOGREC. Eine Servicehilfe, die Informationen über Programm- und Hardwarefehler enthält.

Systemabschluss. Siehe *Sofortiger Abschluss*, *Präventiver Abschluss* und *Gesteuerter Abschluss*.

System Authorization Facility (SAF). Eine OS/390-Einrichtung, über die MQSeries for OS/390 mit einer externen Berechtigungsprüffunktion (ESM) wie beispielsweise RACF kommuniziert.

SYSTEM.COMMAND.INPUT. Eine lokale Warteschlange, in die Anwendungsprogramme MQSeries-Befehle stellen können. Die Befehle werden vom Befehlsserver aus der Warteschlange abgerufen, überprüft und anschließend zu ihrer Ausführung an den Befehlsprozessor übergeben.

Systeminitialisierungstabelle (SIT). Eine Tabelle mit Parametern, die von CICS beim Start verwendet werden.

Systemsteuerbefehle. Befehle für die Manipulation plattformspezifischer Einheiten wie beispielsweise Pufferpools, Speicherklassen oder Dateien.

T

TACL. Tandem Advanced Command Language.

Task-Steuerblock (TCB). Ein OS/390-Steuerblock, der der Meldung von Informationen über Tasks innerhalb eines Adressraums dient, die mit einem OS/390-Subsystem wie beispielsweise MQSeries for OS/390 oder CICS verbunden sind.

Task-Switch. Die Überlappung von E/A-Vorgängen und die task-übergreifende Verarbeitung. In MQSeries for OS/390 wird mit der Task-Switch-Funktion die Leistung dadurch optimiert, dass die Ausführung bestimmter MQI-Aufrufe nicht im Rahmen des übergeordneten CICS-Task-Steuerblocks, sondern im Rahmen von Sub-Tasks erfolgt.

TCB. Task Control Block (siehe *Task-Steuerblock*).

Temporäre dynamische Warteschlange. Eine dynamische Warteschlange, die beim Schließen gelöscht wird. Temporäre dynamische Warteschlangen werden im Anschluss an einen WS-Managerausfall nicht wiederhergestellt, so dass sie nur nicht permanente Nachrichten enthalten können. Vgl. *Permanente dynamische Warteschlange*.

thlqual. Target Library High-Level Qualifier (Qualifikationsmerkmal der oberen Ebene für Zielbibliothek).

Thread. In MQSeries die niedrigste Ebene, auf der in einem Betriebssystem eine parallele Verarbeitung möglich ist.

TMI. Trigger Monitor Interface (siehe *Auslösemonitorschnittstelle*).

Trace. In MQSeries eine Funktion zur Aufzeichnung der MQSeries-Aktivität. Als Zieladresse für Trace-Einträge kommen unter anderem GTF und die Systemverwaltungsfunktion (SMF) in Frage. Siehe auc *Globaler Trace* und *Leistungs-Trace*.

tranid. Siehe *Transaktions-ID*.

Transaktions-ID. In CICS der Name, der bei der Definition einer Transaktion festgelegt und über den diese aufgerufen wird.

U

Übertragungsprogramm. Siehe *Nachrichtenkanalagent*.

Übertragungswarteschlange. Eine lokale Warteschlange, in der vorbereitete Nachrichten an einen fernen WS-Manager temporär gespeichert werden.

UIS. User Identifier Service (siehe *Benutzer-ID-Service*).

Umgebung. Siehe *Anwendungsumgebung*.

Unbestätigte Arbeitseinheit mit Wiederherstellung. In MQSeries der Status, in dem sich eine Arbeitseinheit mit Wiederherstellung befindet, für die ein Synchronisationspunkt angefordert, aber noch nicht bestätigt wurde.

Undo/Redo-Datensatz. Ein Protokolldatensatz, der bei Wiederherstellungsvorgängen verwendet wird. Der Redo-Teil (Widerruf zurücknehmen) gibt Aufschluss über die Änderung, die an einem MQSeries-Objekt vorgenommen werden soll. Der Undo-Teil (Widerrufen) gibt Aufschluss darüber, wie diese Änderung zurückgesetzt werden soll, wenn die Arbeitseinheit nicht festgeschrieben wird.

Unterbrechung durch Maschinenfehler. Eine Unterbrechung tritt auf Grund einer Gerätestörung oder eines Fehlers auf. Bei einer Maschinenfehlerunterbrechung kann es sich um einen behebbaren Hard- oder Softwarefehler oder um einen nicht behebbaren Fehler handeln.

Unveränderte Weiterleitung. Bei der Fehlerbehebung die Übergabe der Steuerung von einer Fehlerbehebungsroutine an eine Fehlerbehebungsroutine der nächsthöheren Ebene nach einem vorgegebenen Schema.

Ursachencode. Ein Rückkehrcode, der Aufschluss über die Ursache eines fehlgeschlagenen bzw. teilweise fehlgeschlagenen MQI-Aufrufs gibt.

Ursachencode für abnormale Beendigung. Ein hexadezimaler Code mit einer Länge von 4 Byte, mit dem Fehlerursachen in MQSeries for OS/390 eindeutig ermittelt werden. Eine vollständige Liste mit allen Ursachencodes für abnormale Beendigung in MQSeries for OS/390 und deren Beschreibung finden Sie im Handbuch *MQSeries for OS/390 Messages and Codes*.

V

Verbindung. Ein OS/390-Adressraum, der mit MQSeries for OS/390 verbunden ist.

Verbindungskennung. Die Kennung bzw. der Token, über die bzw. den ein Programm Zugriff auf den WS-Manager erhält, mit dem es verbunden ist.

Verbundener Adressraum. Siehe *Verbindung*.

Verteilte Anwendung. Beim Message-Queuing eine Gruppe von Anwendungsprogrammen, die einzeln jeweils mit verschiedenen WS-Managern verbunden sein können, zusammen jedoch eine einzige Anwendung darstellen.

Verteilte Warteschlangenverwaltung (DQM). Beim Message-Queuing die Konfiguration und Steuerung von Nachrichtenkanälen zu WS-Managern auf anderen Systemen.

Verzögerte Verbindung. Ein anstehendes Ereignis, das beim Versuch eines CICS-Subsystems, vor dem Start von MQSeries for OS/390 eine Verbindung zu MQSeries for OS/390 herzustellen, aktiviert wird.

Vorläufige Programmkorrektur (PTF). Die temporäre Lösung bzw. Umgehung von Problemen, die von IBM Ingenieuren im Außendienst gefunden und durch Fehler in einem aktuellen, unveränderten Programm-Release verursacht wurden.

W

Warteschlange. Ein MQSeries-Objekt. Anwendungen für Message-Queuing können Nachrichten in eine Warteschlangen einreihen bzw. aus ihr abrufen. Eine Warteschlange ist einem WS-Manager zugeordnet, von dem sie verwaltet wird. Lokale Warteschlangen können

Listen mit Nachrichten enthalten, die zur Verarbeitung anstehen. Andere Warteschlangenarten können keine Nachrichten enthalten; sie verweisen lediglich auf andere Warteschlangen oder können als Modell für dynamische Warteschlangen verwendet werden.

Warteschlange für nicht zugestellte Nachrichten. Siehe *Warteschlange für nicht zustellbare Nachrichten*.

Warteschlange für nicht zustellbare Nachrichten. Eine Warteschlange, in die von WS-Managern bzw. Anwendungen Nachrichten eingereiht werden, die nicht an die eigentliche Zieladresse übertragen werden können.

Warteschlangen für Antwortnachrichten. Gibt die Warteschlange an, an die auf Anforderung des Programms, das einen MQPUT-Aufruf abgesetzt hat, eine Antwort- bzw. Berichtsnachricht gesendet werden soll.

Wert für Folgenummernserie. In MQSeries ein Verfahren, mit dem gewährleistet wird, dass die Nachrichtenfolgenummern an beiden Enden der Kommunikationsverbindung gleichzeitig zurückgesetzt werden. Durch die Übertragung von Nachrichten mit Folgenummern wird sichergestellt, dass der Empfangskanal die Nachrichten in der ursprünglichen Reihenfolge speichern kann.

Wiederherstellungsprotokoll. In MQSeries for OS/390 Datensätze, die Informationen für die Wiederherstellung von Nachrichten, Warteschlangen und des MQSeries-Subsystems enthalten. MQSeries for OS/390 schreibt alle Datensätze in eine Datei, das sogenannte *aktive Protokoll*. Sobald dieses Protokoll voll ist, wird der Inhalt in einen permanenten Sicherungsspeicher (DASD) oder eine Banddatei, das sogenannte *Archivierungsprotokoll*, verschoben. Synonym für *Protokoll*.

Wiederherstellungspunkt. In MQSeries for OS/390 bezeichnet der Begriff die Sicherungskopien von MQSeries for OS/390-Dateien (Page Sets) und den entsprechenden Protokolldatensätzen, die für die Wiederherstellung dieser Dateien (Page Sets) erforderlich sind. Diese Sicherungskopien stellen den potenziellen Wiederanlaufspunkt für den Fall von Dateiverlusten dar (z. B. Ein-/Ausgabefehler im Zusammenhang mit Dateien (Page Sets)).

WS-Manager. Ein Systemprogramm, das Anwendungen Services für die Steuerung von Warteschlangen zur Verfügung stellt. Der WS-Manager stellt eine Anwendungsprogrammierschnittstelle zur Verfügung, die anderen Programmen den Zugriff auf Nachrichten in den Warteschlangen ermöglicht, die dem WS-Manager zugeordnet sind. Siehe auch *Lokaler WS-Manager* und *Ferner WS-Manager*. Ein MQSeries-Objekt, das die Attribute eines bestimmten WS-Managers definiert.

WS-Manager-Ereignis. Ein Ereignis, das auf Folgendes hinweist:

- das Auftreten einer Fehlerbedingung in Zusammenhang mit den Ressourcen, auf die von einem WS-Manager zugegriffen wird. Beispiel: Eine Warteschlange ist nicht mehr verfügbar.
- Im WS-Manager ist eine erhebliche Änderung aufgetreten. Beispiel: Der WS-Manager wurde gestoppt bzw. gestartet.

Z

Zeilengruppe. Eine Zeilengruppe in einer Konfigurationsdatei, mit der ein Wert für einen Parameter festgelegt wird, der das Verhalten eines WS-Managers, Clients oder Kanals bestimmt. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 kann eine Konfigurationsdatei (.ini) mehrere Zeilengruppen beinhalten.

Zeitunabhängige Nachrichtenübertragung. Siehe *Asynchrone Nachrichtenübertragung*.

Zurücksetzen. Ein Vorgang, bei dem alle Änderungen wieder rückgängig gemacht werden, die während der aktuellen Arbeitseinheit mit Wiederherstellung bzw. der aktuellen Arbeitseinheit vorgenommen wurden. Nach Abschluss dieses Vorgangs beginnt eine neue Arbeitseinheit mit Wiederherstellung bzw. eine neue Arbeitseinheit. Vgl. *Festschreiben*.

Zweiphasige Festschreibung. Ein Protokoll für die Koordinierung von Änderungen an wiederherstellbaren Ressourcen, wenn von einer einzelnen Transaktion mehrere Ressourcenmanager verwendet werden. Vgl. *Einphasige Festschreibung*.

Zyklische Protokollierung. In MQSeries auf UNIX-Systemen, MQSeries for OS/2 Warp und MQSeries for Windows NT and Windows 2000 ein Verfahren, bei dem alle Daten für den Wiederanlauf in einem Protokolldateiring geführt werden. Bei der Protokollierung wird zunächst in die erste Datei im Ring geschrieben, dann in die zweite, usw. Sobald alle Dateien voll sind, beginnt die Datenaufzeichnung wieder mit der ersten Datei im Ring, vorausgesetzt, der Bereich wurde freigegeben bzw. wird nicht länger benötigt. Die zyklische Protokollierung wird beim Wiederanlauf verwendet; dabei werden anhand des Protokolls Transaktionen zurückgesetzt, die zum Zeitpunkt des Systemabbruchs aktiv waren. Vgl. *Lineare Protokollierung*.

Index

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Kommentare an IBM senden

Sie können uns Anmerkungen zu dem vorliegenden Handbuch über die nachfolgend aufgeführten Wege zukommen lassen.

Bitte lassen Sie es uns wissen, wenn Informationen Ihrer Meinung nach fehlerhaft sind oder ganz fehlen, oder wenn Sie Anmerkungen zur Richtigkeit, zum Aufbau, Inhalt oder Vollständigkeit des Handbuchs haben.

Bitte senden Sie uns Kommentare nur im Zusammenhang mit dem vorliegenden Handbuch und nur über die hier aufgeführten Übermittlungskanäle zu.

Wenn Sie Anmerkungen zu Funktionen von IBM Produkten oder IBM Systemen haben, wenden Sie sich bitte an Ihren IBM Ansprechpartner bzw. den zuständigen IBM Vertriebspartner.

Bei IBM eingehende Kommentare können von IBM beliebig verwendet werden, ohne dass hieraus eine Verpflichtung gegenüber dem Absender entsteht.

Ihre Kommentare können Sie IBM auf folgenden Wegen zukommen lassen:

- Per Post an folgende Adresse:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  Großbritannien

- Per Fax:
  - Benutzer außerhalb von Großbritannien müssen im Anschluss an die jeweilige internationale Durchwahl (in Deutschland z. B. 00) folgende Nummer wählen: 44–1962–842327
  - Benutzer in Großbritannien müssen folgende Nummer wählen: 01962–842327
- Per E-Mail, unter Angabe der entsprechenden Netz-ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Unabhängig von der Übertragungsart sind auf jeden Fall folgende Angaben erforderlich:

- Die Bestellnummer sowie der Titel der Veröffentlichung
- Der Abschnitt, auf den Sie sich beziehen
- Ihre Adresse: Name, Adresse, Telefonnummer, Faxnummer, Netz-ID

Antwort

MQSeries for Compaq OpenVMS Alpha
Systemverwaltung
Version 5 Release 1

IBM Form SC12-2998-00

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

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

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

Kommentare:

Danke für Ihre Bemühungen.

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

Name	Adresse
Firma oder Organisation	
Rufnummer	E-Mail-Adresse

Antwort
SC12-2998-00

IBM Deutschland GmbH
SW TSC Germany

70548 Stuttgart

Printed in Denmark

SC12-2998-00