8.0

IBM MQ 관리

참고

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 337 페이지의 『주의사항』에 있는 정보를 확인하십시오.

이 개정판은 새 개정판에 별도로 명시하지 않는 한, IBM® MQ 의 버전 8릴리스 0및 모든 후속 릴리스와 수정에 적용됩니다.

목차

IBM MQ 관리

큐 관리자 및 연관된 자원 관리에는 해당 자원을 활성화하고 관리하기 위해 자주 수행하는 태스크가 포함됩니다. 큐 관리자 및 연관된 자원을 관리할 때 선호하는 메소드를 선택하십시오.

IBM MQ 오브젝트를 로컬 또는 원격으로 관리할 수 있습니다. 7 페이지의 『로컬 및 원격 관리』의 내용을 참조하십시오.

IBM MQ에서 큐 관리자 및 해당 관련 자원을 작성 및 관리하기 위해 사용할 수 있는 여러 가지 메소드가 있습니다. 이러한 메소드에는 명령행 인터페이스, 그래픽 사용자 인터페이스 및 관리 API가 포함됩니다. 이러한 각 인터페이스에 대한 자세한 정보는 이 토픽의 절 및 링크를 참조하십시오.

플랫폼에 따라 IBM MQ를 관리하는 데 사용할 수 있는 여러 명령 세트가 있습니다.

- 5 페이지의 『IBM MQ 제어 명령』
- 5 페이지의 『MQSC(IBM MQ Script) 명령』
- 6 페이지의 『프로그래밍 가능 명령 형식(PCF)』
- IBM i 6 페이지의 『IBM i 제어 언어(CL)』

또한 IBM MQ 오브젝트를 작성 및 관리하기 위한 다음과 같은 기타 옵션도 있습니다.

- 6 페이지의 『MQ Explorer』
- 7 페이지의 『Windows 기본 구성 애플리케이션』
- 7 페이지의 『MSCS(Microsoft Cluster Service)』

z/OS IBM MQ for z/OS®의 관리 인터페이스 및 옵션에 대한 정보는 232 페이지의 『IBM MQ for z/OS 관리』의 내용을 참조하십시오.

PCF 명령을 사용하여 로컬 및 리모트 큐 관리자 모두에 대한 일부 관리 및 모니터링 태스크를 자동화할 수 있습니다. 이러한 명령은 또한 일부 플랫폼에서 MQAI(IBM MQ Administration Interface)를 사용하여 단순화될 수 있습니다. 관리 태스크 자동화에 대한 자세한 정보는 8 페이지의 『관리 태스크 자동화』의 내용을 참조하십시오.

IBM MQ 제어 명령

제어 명령을 사용하면 큐 관리자 자체에서 관리 태스크를 수행할 수 있습니다.

IBM MQ for Windows, UNIX 및 Linux® 시스템은 시스템 명령행에서 실행하는 제어 명령 을 제공합니다.

제어 명령은 분배 플랫폼에서 큐 관리자 작성 및 관리에서 설명됩니다. 제어 명령을 위한 명령 참조의 경우, IBM MQ 제어 명령을 참조하십시오.

MQSC(IBM MQ Script) 명령

큐 관리자 자체, 큐, 프로세스 정의, 이름 리스트, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 인증 정보 오브젝트를 포함하여, 큐 관리자 오브젝트를 관리하기 위해 MQSC 명령을 사용하십시오.

`runmqsc` 명령을 사용하여 큐 관리자에 MQSC 명령을 실행할 수 있습니다. 이를 키보드에서 명령을 실행하여 대화식으로 수행하거나, ASCII 텍스트 파일에서 명령 순서를 실행하도록 표준 입력 디바이스(stdin)를 경로 재지정할 수 있습니다. 두 경우 모두, 명령의 형식이 동일합니다.

명령에 설정된 플래그에 따라서 `runmqsc` 명령을 세 가지 모드에서 실행할 수 있습니다.

- 확인 모드 - 여기서 MQSC 명령은 로컬 큐 관리자에서 확인되지만 실행되지는 않습니다.
- 직접 모드 - 여기서 MQSC 명령이 로컬 큐 관리자에서 실행됩니다.
- 간접 모드 - 여기서 MQSC 명령이 리모트 큐 관리자에서 실행됩니다.

대소문자가 구분되지 않더라도 MQSC 명령에 지정되는 오브젝트 속성이 이 절에서 대문자로 표시됩니다(예: RQMNAME). MQSC 명령 속성 이름은 8자로 제한됩니다.

MQSC 명령은 모든 플랫폼IBM i z/OS에서 사용 가능합니다(IBM i 및 z/OS 포함). MQSC 명령은 명령 세트 비교에 요약됩니다.

Windows, UNIX 또는 Linux에서 MQSC를 시스템 명령행에서 실행되는 단일 명령으로 사용할 수 있습니다. 보다 복잡한 명령 또는 여러 명령을 실행하기 위해 MQSC를 Windows, UNIX 또는 Linux 시스템 명령행에서 실행하는 파일로 빌드할 수 있습니다. MQSC는 리모트 큐 관리자에 송신할 수 있습니다. 상세한 내용은 명령 스크립트 빌드의 내용을 참조하십시오.

IBM i IBM i에서 IBM i 서버에서 명령을 실행하려면 스크립트 파일에 명령 목록을 작성한 다음 STRMQMMQSC 명령을 사용하여 파일을 실행하십시오.

참고사항: IBM i

1. QTEMP 라이브러리 사용이 제한되므로 STRMQMMQSC의 입력 라이브러리로 QTEMP 라이브러리를 사용하지 마십시오. 명령의 입력 파일로 다른 라이브러리를 사용해야 합니다.
2. IBM i에서 스크립트 파일에서 실행된 명령에 대한 MQSC 응답은 스풀 파일로 리턴됩니다.

69 페이지의 『스크립트(MQSC) 명령』에는 각 MQSC 명령 및 해당 구문의 설명이 들어 있습니다.

로컬 관리에서 MQSC 명령 사용에 대한 자세한 정보는 68 페이지의 『MQSC 명령을 사용하여 로컬 관리 태스크 수행』의 내용을 참조하십시오.

프로그래밍 가능 명령 형식(PCF)

프로그래밍 가능 명령 형식(PCF)는 네트워크의 큐 관리자(PCF 지원) 및 프로그램 사이에서 교환할 수 있는 명령 및 응답 메시지를 정의합니다. IBM MQ 오브젝트(인증 정보 오브젝트, 채널, 채널 리스너, 이름 목록, 프로세스 정의, 큐 관리자, 큐, 서비스 및 스토리지 클래스)의 관리를 위해 시스템 관리 애플리케이션 프로그램에서 PCF 명령을 사용할 수 있습니다. 애플리케이션은 네트워크의 단일 지점에서 작동하여 로컬 큐 관리자를 사용하여 로컬 또는 원격으로 모든 큐 관리자와 명령 및 응답 정보를 통신할 수 있습니다.

PCF에 대한 자세한 정보는 9 페이지의 『프로그래밍 가능 명령 형식 소개』의 내용을 참조하십시오.

PCF 정의 및 명령 및 응답의 구조는 프로그래밍 가능 명령 형식(PCF) 참조서를 참조하십시오.

IBM i 제어 언어(CL)

IBM i

이 언어는 관리 명령을 IBM MQ for IBM i에 실행할 때 사용할 수 있습니다. 명령은 명령행에서 또는 CL 프로그램을 작성하여 실행할 수 있습니다. 이러한 명령은 PCF 명령과 유사한 기능을 수행하지만 형식은 다릅니다. CL 명령은 서버용으로 독점적으로 설계되어 있고 CL 응답은 사람이 판독할 수 있도록 설계되어 있는 반면, PCF 명령은 플랫폼 독립적이며 명령 및 응답 형식 둘 모두 프로그램용입니다.

IBM i 제어 언어(CL)에 대한 자세한 내용의 경우, IBM MQ for IBM i CL 명령의 내용을 참조하십시오.

MQ Explorer

MQ Explorer를 사용하여 다음 조치를 수행할 수 있습니다.

- 큐 관리자, 큐, 프로세스 정의, 이름 목록, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 클러스터를 포함하여 다양한 자원을 정의 및 제어하십시오.
- 로컬 큐 관리자 및 해당 연관된 프로세스를 시작하거나 중지하십시오.
- 워크스테이션 또는 기타 워크스테이션에서 큐 관리자 및 연관된 오브젝트를 보십시오.
- 큐 관리자, 클러스터 및 채널의 상태를 검사하십시오.
- 큐 상태에서 어떤 애플리케이션, 사용자 또는 채널에 특정 큐가 열려 있는지를 보려면 검사하십시오.

Windows 및 Linux 시스템에서 시스템 메뉴, `MQExplorer` 실행 파일 또는 **`strmqcfg`** 명령을 사용하여 MQ Explorer 를 시작할 수 있습니다.

Linux에서 MQ Explorer를 성공적으로 시작하려면 홈 디렉토리가 존재하고 이 홈 디렉토리에 파일을 작성할 수 있어야 합니다.

자세한 정보는 56 페이지의 『MQ Explorer를 사용하여 관리』의 내용을 참조하십시오.

MQ Explorer를 사용하여 다른 플랫폼(예:z/OS)에 리모트 큐 관리자를 관리할 수 있습니다. 자세한 내용을 보고 SupportPac MS0T를 다운로드하려면 https://www.ibm.com/support/docview.wss?uid=swg24021041의 내용을 참조하십시오.

Windows 기본 구성 애플리케이션

Windows 기본 구성 프로그램을 사용하여 IBM MQ 오브젝트의 시작 (또는 기본) 세트를 작성할 수 있습니다. 작성된 기본 오브젝트의 요약은 표 1: Windows 기본 구성 애플리케이션에서 작성한 오브젝트에 나열됩니다.

MSCS(Microsoft Cluster Service)

MSCS(Microsoft Cluster Service)는 데이터 및 애플리케이션의 더 높은 가용성을 제공하고 시스템을 관리하는 것을 보다 용이하게 하면서 클러스터에 서버를 연결시킬 수 있도록 합니다. MSCS는 서버 또는 애플리케이션 실패를 자동으로 감지하고 복구할 수 있습니다.

MSCS 센스의 클러스터를 IBM MQ 클러스터와 혼동하지 않는 것이 중요합니다. 차이점은 다음과 같습니다.

IBM MQ 클러스터
하나 이상의 컴퓨터에 있는 두 개 이상의 큐 관리자 그룹으로, 자동 상호연결을 제공하고 로드 밸런싱 및 중복을 위해 큐가 서로 공유될 수 있도록 합니다.

MSCS 클러스터
상호 연결되어 있으며 한 대의 컴퓨터에서 장애가 발생하면 MSCS가 장애 복구를 수행하여 애플리케이션의 상태 데이터를 장애가 발생한 컴퓨터로부터 클러스터의 다른 컴퓨터로 전송하고 해당 조작을 그 위치에서 다시 시작하는 방식으로 구성된 컴퓨터의 그룹입니다.

MSCS(Supporting the Microsoft Cluster Service)에서는 MSCS를 사용하기 위해 IBM MQ for Windows 시스템을 구성하는 방법에 대한 자세한 정보를 제공합니다.

관련 개념

64 페이지의 『로컬 IBM MQ 오브젝트 관리』
이 섹션은 메시지 큐 인터페이스(MQI)를 사용하는 애플리케이션 프로그램을 지원하기 위해 로컬 IBM MQ 오브젝트를 관리하는 방법에 대해 알려줍니다. 이 컨텍스트에서 로컬 관리는 IBM MQ 오브젝트 작성, 표시, 변경, 복사 및 삭제를 의미합니다.

118 페이지의 『원격 IBM MQ 오브젝트 관리』

165 페이지의 『IBM i 관리』
IBM i에서 IBM MQ을 관리할 수 있도록 하는 메소드를 소개합니다.

232 페이지의 『IBM MQ for z/OS 관리』
큐 관리자 및 연관된 자원 관리에는 해당 자원을 활성화하고 관리하기 위해 자주 수행하는 태스크가 포함됩니다. 큐 관리자 및 연관된 자원을 관리할 때 선호하는 메소드를 선택하십시오.

관련 정보

IBM MQ 기술 개요

계획 중

z/OS z/OS에서 IBM MQ 환경 계획

구성

z/OS z/OS 구성

트랜잭션 지원 시나리오

XA 자원 관리자에 대한 연결이 끊길 경우 고려사항

로컬 및 원격 관리

IBM MQ 오브젝트를 로컬 또는 원격으로 관리할 수 있습니다.

로컬 관리는 로컬 시스템에서 정의한 큐 관리자에서 관리 태스크를 수행한다는 것을 의미합니다. 기타 시스템(예: TCP/IP 터미널 에뮬레이션 프로그램 **telnet**을 통해)에 액세스할 수 있고 거기에서 관리를 수행할 수 있습니

다. IBM MQ에서 포함되는 채널이 없기 때문에 로컬 관리로서 이를 고려할 수 있습니다. 즉, 통신이 운영 체제에 의해 관리됩니다.

IBM MQ는 원격 관리로 알려진 것을 통해 한 개의 연결 지점에서 관리를 지원합니다. 이를 사용하면 다른 시스템에서 처리되는 로컬 시스템에서 명령을 실행할 수 있고 IBM MQ 탐색기에도 적용됩니다. 예를 들어, 리모트 큐 관리자에서 큐 정의를 변경하기 위한 리모트 명령을 실행할 수 있습니다. 적절한 채널이 정의되도록 해야 하지만 해당 시스템에 로그온할 필요는 없습니다. 대상 시스템의 큐 관리자 및 명령 서버가 실행 중이어야 합니다.

일부 명령은 특히, 큐 관리자를 작성하거나 시작하고 명령 서버를 시작하는 방법으로 실행될 수 없습니다. 이 태스크 유형을 수행하려면 원격 시스템에 로그온하고 거기에서 명령을 실행하거나 사용자를 위해 명령을 실행할 수 있는 프로세스를 작성해야 합니다. 이 제한은 IBM MQ 탐색기에도 적용됩니다.

118 페이지의 『원격 IBM MQ 오브젝트 관리』 에서는 더 큰 세부사항에서 원격 관리의 주제에 대해 설명합니다.

IBM MQ 제어 명령 사용 방법

이 섹션은 IBM MQ 제어 명령을 사용하는 방법을 설명합니다.

제어 명령을 실행하려면 사용자 ID가 대부분의 제어 명령에 대한 mqm 그룹의 멤버여야 합니다. 이에 대한 자세한 정보는 Authority to administer IBM MQ on UNIX, Linux and Windows systems의 내용을 참조하십시오. 또한, 다음 환경별 정보를 참고하십시오.

IBM MQ for Windows
모든 제어 명령은 명령행에서 실행될 수 있습니다. 명령어 및 해당 플래그는 대소문자를 구분하지 않습니다. 대문자, 소문자 또는 대문자와 소문자의 결합으로 입력할 수 있습니다. 그러나 큐 이름과 같은 명령을 제어하기 위한 인수는 대소문자가 구분됩니다.

구문 설명에서 하이픈(-)은 플래그 표시기로 사용됩니다. 하이픈 대신에 포워드 슬래시(/)를 사용할 수 있습니다.

IBM MQ for UNIX 및 Linux 시스템
모든 IBM MQ 제어 명령은 쉘에서 실행될 수 있습니다. 모든 명령은 대소문자를 구분합니다.

제어 명령의 서브세트는 IBM MQ 탐색기를 사용하여 실행될 수 있습니다.

자세한 정보는 IBM MQ 제어 명령을 참조하십시오.

관리 태스크 자동화

일부 관리 및 모니터링 태스크를 자동화하면 설치에 유익할 것이라고 결정할 수 있습니다. 프로그래밍 가능 명령 형식(PCF) 명령을 사용하여 로컬 및 리모트 큐 관리자 모두의 관리 태스크를 자동화할 수 있습니다. 이 절에서는 사용자가 IBM MQ 오브젝트를 관리해 본 적이 있다고 가정합니다.

PCF 명령

IBM MQ 프로그래밍 가능 명령 형식(PCF) 명령이 관리 프로그램에서 프로그램 관리 태스크에 사용될 수 있습니다. 이 방식으로 프로그램으로부터 큐 관리자 오브젝트(큐, 프로세스 정의, 이름 리스트, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 인증 정보 오브젝트)를 조작하고, 심지어 큐 관리자 자체를 조작할 수도 있습니다.

PCF 명령은 MQSC 명령에서 제공하는 동일한 범위의 기능을 포함합니다. 단일 노드에서 네트워크의 임의의 큐 관리자에 PCF 명령을 발행하는 프로그램을 작성할 수 있습니다. 이러한 방법으로 관리 태스크를 집중시키고 자동화할 수 있습니다.

각 PCF 명령은 IBM MQ 메시지의 애플리케이션 데이터 부분에 임베드되는 데이터 구조입니다. 각 명령은 다른 메시지와 동일한 방식으로 MQI 기능인 MQPUT을 사용하여 대상 큐 관리자로 송신됩니다. 명령 서버가 메시지를 수신하는 큐 관리자에서 실행 중인 경우, 명령 서버는 메시지를 명령 메시지로 해석하고 명령을 실행합니다. 응답을 가져오기 위해 애플리케이션은 MQGET 호출을 실행하고 응답 데이터는 다른 데이터 구조로 리턴됩니다. 그런 다음, 애플리케이션은 응답을 처리하고 그에 따라 수행합니다.

참고: MQSC 명령과는 다르게 PCF 명령 및 그에 대한 응답은 사용자가 읽을 수 있는 텍스트 형식이 아닙니다.

다음은 PCF 명령 메시지를 작성하는 데 필요한 항목입니다.

메시지 디스크립터
이는 표준 IBM MQ 메시지 디스크립터입니다.

- 메시지 유형(*MsqType*)은 MQMT_REQUEST입니다.
- 메시지 형식(*Format*)은 MQFMT_ADMIN입니다.

애플리케이션 데이터
PCF 헤더를 포함하는 PCF 메시지가 들어 있습니다.

- PCF 메시지 유형(*Type*)은 MQCFT_COMMAND를 지정합니다.
- 명령 ID는 명령(예: *Change Queue*(MQCMD_CHANGE_Q)을 지정합니다.

PCF 데이터 구조와 구현 방법에 대한 자세한 설명은 9 페이지의 『프로그래밍 가능 명령 형식 소개』의 내용을 참조하십시오.

PCF 오브젝트 속성

PCF의 오브젝트 속성은 MQSC 명령에 대한 문자 수가 8자로 제한되지 않습니다. 이는 이 안내서에서 이탤릭체로 표시됩니다. 예를 들어, RQMNAME의 PCF 등가물은 *RemoteQMgrName*입니다.

PCF 나가기

PCF 나가기는 메시지 텍스트 내에 MQSC 명령을 포함하는 PCF 명령입니다. PCF를 사용하여 리모트 큐 관리자로 명령을 송신할 수 있습니다. PCF 나가기에 대한 자세한 정보는 나가기를 참조하십시오.

프로그래밍 가능 명령 형식 소개

프로그래밍 가능 명령 형식(PCF)는 네트워크의 큐 관리자(PCF 지원) 및 프로그램 사이에서 교환할 수 있는 명령 및 응답 메시지를 정의합니다. PCF를 사용하면 큐 관리자 관리와 다른 네트워크 관리를 간단하게 수행할 수 있습니다. 특히 네트워크가 규모면에서 커진 것처럼 분산 네트워크 및 복잡도의 복합적 관리의 문제를 해결할 때 사용할 수 있습니다.

이 제품 문서에 설명된 PCF(Programmable Command Format)가 다음에서 지원됩니다.

- IBM MQ for AIX®
- IBM MQ for HP-UX
- IBM i IBM MQ for IBM i
- Linux용 IBM MQ
- IBM MQ for Solaris
- IBM MQ for Windows
- z/OS IBM MQ for z/OS
- IBM WebSphere® MQ for HP Integrity NonStop Server

문제점 PCF 명령 해결

분배된 네트워크의 관리가 복잡해질 수 있습니다. 관리의 문제점은 네트워크의 크기 및 복잡도가 증가하기 때문에 계속 증가합니다.

메시징 및 큐잉에 특정한 관리 예에는 다음이 포함됩니다.

- 자원 관리.

  예: 큐 작성 및 삭제.
- 성능 모니터링.

  예를 들어, 최대 큐 용량 또는 메시지 비율.
- 제어

예를 들어, 최대 큐 용량, 최대 메시지 길이와 같은 큐 매개변수 성능 조정 및 큐 사용 및 사용 안함.

- 메시지 라우팅.

  대체의 정의가 네트워크를 통해 라우팅됩니다.

IBM MQ PCF 명령은 큐 관리자 관리 및 기타 네트워크 관리를 단순화하기 위해 사용될 수 있습니다. PCF 명령을 사용하면 네트워크 내의 단일 큐 관리자에서 네트워크 관리를 수행하기 위해 단일 애플리케이션을 사용할 수 있습니다.

PCF의 개념

PCF는 네트워크에서 PCF를 지원하는 큐 관리자와 프로그램 사이에 교환될 수 있는 명령 및 응답 메시지를 정의합니다. IBM MQ 오브젝트(인증 정보 오브젝트, 채널, 채널 리스너, 이름 목록, 프로세스 정의, 큐 관리자, 큐, 서비스 및 스토리지 클래스)의 관리를 위해 시스템 관리 애플리케이션 프로그램에서 PCF 명령을 사용할 수 있습니다. 애플리케이션은 네트워크의 단일 지점에서 작동하여 로컬 큐 관리자를 사용하여 로컬 또는 원격으로 모든 큐 관리자와 명령 및 응답 정보를 통신할 수 있습니다.

각 큐 관리자에는 표준 큐 이름을 가지는 관리 큐가 있고 애플리케이션은 해당 큐로 PCF 명령 메시지를 보낼 수 있습니다. 각 큐 관리자에는 관리 큐에서 명령 메시지를 제공할 명령 서버도 있습니다. 따라서, PCF 명령 메시지가 네트워크에서 큐 관리자에 의해 처리될 수 있고 응답 데이터는 지정된 응답 큐를 사용하여 애플리케이션으로 리턴될 수 있습니다. PCF 명령 및 응답 메시지가 정상 메시지 큐 인터페이스(MQI)를 사용하여 보내지고 수신됩니다.

해당 매개변수를 포함하여 사용 가능한 PCF 명령 목록의 경우 PCF(Programmable Command Format)의 정의를 참조하십시오.

PCF(Programmable Command Format) 사용

IBM MQ 원격 관리를 위해 시스템 관리 프로그램에서 pcf를 사용할 수 있습니다.

이 섹션에는 다음이 포함됩니다.

- 10 페이지의 『PCF 명령 메시지』
- 13 페이지의 『응답』
- z/OS 14 페이지의 『확장된 응답』
- IBM MQ 오브젝트 이름 지정 규칙
- 16 페이지의 『PCF 명령에 대한 권한 검사』

PCF 명령 메시지

PCF 명령 메시지는 해당 헤더 및 사용자 정의 메시지 데이터에서 식별되는 PCF 헤더, 매개변수로 구성됩니다. 메시지는 메시지 큐 인터페이스 호출을 사용하여 실행됩니다.

각 명령 및 해당 매개변수는 수많은 매개변수 구조가 뒤에 오는 PCF 헤더를 포함하는 분리 명령 메시지로서 보내집니다. PCF 헤더의 세부사항의 경우, MQCFH - PCF 헤더를 참조하고 매개변수 구조의 예는 MQCFST - PCF 문자열 매개변수를 참조하십시오. PCF 헤더는 동일한 메시지에서 수행하는 매개변수 구조의 수와 명령을 식별합니다. 각 매개변수 구조에서는 명령에 매개변수를 제공합니다.

명령 서버에서 생성되는 명령에 대한 응답에 유사한 구조가 있습니다. 매개변수 구조의 수가 뒤에 오는 PCF 헤더가 있습니다. 응답은 둘 이상의 메시지로 구성되지만 명령은 항상 하나의 메시지로만 구성됩니다.

z/OS 이외의 플랫폼에서 PCF 명령이 보내지는 큐는 항상 SYSTEM.ADMIN.COMMAND.QUEUE라고 합니다.

z/OS z/OS에서는 명령이 SYSTEM.COMMAND.INPUT으로 송신되지만, SYSTEM.ADMIN.COMMAND.QUEUE가 이에 대한 알리어스일 수 있습니다. 이 큐를 서비스하는 명령 서버는 명령 메시지의 메시지 디스크립터에 있는 *ReplyToQ* 및 *ReplyToQMgr* 필드에 의해 정의된 큐에 응답을 전송합니다.

PCF 명령 메시지 발행 방법

정상 메시지 큐 인터페이스(MQI) 호출, MQPUT, MQGET 등을 사용하여 해당 큐로 PCF 명령 및 응답 메시지를 배치시키고 검색하십시오.

참고:

명령 서버가 해당 큐 관리자에서 처리할 PCF 명령에 대한 대상 큐 관리자에서 실행 중인지 확인하십시오.

제공된 헤더 파일 목록의 경우, IBM MQ COPY, 헤더, 포함 및 모듈 파일을 참조하십시오.

PCF 명령에 대한 메시지 디스크립터

IBM MQ 메시지 디스크립터는 MQMD - 메시지 디스크립터에서 완전하게 문서화됩니다.

PCF 명령 메시지는 메시지 디스크립터에 다음 필드를 포함합니다.

Report
필요한 경우, 임의의 올바른 값.

MsgType
이 필드는 응답을 요청하는 메시지를 표시하기 위한 MQMT_REQUEST여야 합니다.

Expiry
필요한 경우, 임의의 올바른 값.

Feedback
MQFB_NONE으로 설정하십시오.

Encoding
IBM i IBM i, Windows, UNIX 또는 Linux 시스템으로 보내는 경우, 이 필드를 메시지 데이터에 사용되는 인코딩으로 설정하십시오. 필요한 경우 변환이 수행됩니다.

CodedCharSetId
만약

IBM i IBM i,

Windows, UNIX 또는 Linux 시스템으로 보내는 경우, 이 필드를 메시지 데이터에 사용되는 코드화 문자 세트 ID로 설정하십시오. 필요한 경우 변환이 수행됩니다.

Format
MQFMT_ADMIN으로 설정하십시오.

Priority
필요한 경우, 임의의 올바른 값.

Persistence
필요한 경우, 임의의 올바른 값.

MsgId
전송 애플리케이션은 값을 지정할 수 있거나 MQMI_NONE은 고유한 메시지 ID를 생성하기 위한 큐 관리자를 요청하기 위해 지정될 수 있습니다.

CorrelId
전송 애플리케이션은 값을 지정할 수 있거나 MQCI_NONE은 상관 ID를 표시하지 않도록 지정될 수 있습니다.

ReplyToQ
응답을 수신할 큐의 이름.

ReplyToQMgr
응답에 대한 큐 관리자의 이름(또는 공백).

메시지 컨텍스트 필드
필요한 경우, 이러한 필드는 임의의 올바른 값으로 설정될 수 있습니다. 일반적으로 Put 메시지 옵션 MQPMO_DEFAULT_CONTEXT는 기본값에 메시지 컨텍스트 필드를 설정하기 위해 사용됩니다.

버전-2 MQMD 구조를 사용 중인 경우, 다음 추가 필드를 설정해야 합니다.

GroupId
MQGI_NONE으로 설정

MsgSeqNumber
1로 설정

Offset
0으로 설정

MsgFlags
MQMF_NONE으로 설정

OriginalLength
MQOL_UNDEFINED으로 설정

사용자 데이터 송신

PCF 구조는 사용자 정의되는 메시지 데이터를 보내기 위해 사용될 수도 있습니다. 이 경우, 메시지 디스크립터 *Format* 필드는 MQFMT_PCF로 설정되어야 합니다.

지정된 큐에서 *PCF* 메시지 송신 및 수신

PCF 메시지를 지정된 큐에 보내기

지정된 큐로 메시지를 송신하기 위해 mqPutBag 호출은 지정된 백의 내용을 PCF 메시지로 변환하며 메시지를 지정된 큐로 송신합니다. 호출 후에는 백의 컨텐츠가 변경되지 않습니다.

이 호출에 대한 입력으로 다음을 제공해야 합니다.

- MQI 연결 핸들.
- 메시지가 놓일 큐에 대한 오브젝트 핸들.
- 메시지 디스크립터. 메시지 디스크립터에 대한 자세한 정보는 MQMD - 메시지 디스크립터의 내용을 참조하십시오.
- MQPMO 구조를 사용하여 메시지 넣기 옵션. MQPMO 구조에 대한 자세한 정보는 MQPMO - 메시지 넣기 옵션의 내용을 참조하십시오.
- 메시지로 변환될 백의 핸들.

  참고: 백에 관리 메시지가 있으며 mqAddInquiry 호출이 값을 백에 삽입하는 데 사용된 경우, MQIASY_COMMAND 데이터 항목의 값은 MQAI가 인식하는 INQUIRE 명령이어야 합니다.

mqPutBag 호출에 대한 전체 설명은 mqPutBag을 참조하십시오.

지정된 큐로부터 PCF 메시지를 수신

지정된 큐로부터 메시지를 수신하기 위해 mqGetBag 호출은 지정된 큐로부터 PCF 메시지를 가져와서 메시지 데이터를 데이터 백으로 변환합니다.

이 호출에 대한 입력으로 다음을 제공해야 합니다.

- MQI 연결 핸들.
- 메시지가 놓일 큐에 대한 오브젝트 핸들.
- 메시지 디스크립터. MQMD 구조 내에 `Format` 매개변수는 MQFMT_ADMIN, MQFMT_EVENT 또는 MQFMT_PCF여야 합니다.

  참고: 메시지가 작업 단위 내에서 수신되고(즉, MQGMO_SYNCPOINT 옵션과 함께) 메시지에 지원되지 않은 형식이 있는 경우, 작업 단위가 백아웃될 수 있습니다. 그런 다음, 메시지는 큐에서 복원되고 mpGetBag 호출 대신 MQGET 호출을 사용하여 검색할 수 있습니다. 메시지 디스크립터에 대한 자세한 정보는 MQGMO - 메시지 가져오기 옵션의 내용을 참조하십시오.
- MQGMO 구조를 사용하여 메시지 가져오기 옵션. MQGMO 구조에 대한 자세한 정보는 MQMD - 메시지 디스크립터의 내용을 참조하십시오.
- 변환된 메시지가 포함될 백의 핸들.

mqGetBag 호출에 대한 전체 설명은 mqGetBag을 참조하십시오.

응답

각 명령에 따라, 명령 서버는 하나 이상의 응답 메시지를 생성합니다. 응답 메시지에는 명령 메시지와 유사한 형식을 가지고 있습니다.

PCF 헤더에는 응답인 명령과 동일한 명령 ID 값이 있습니다(자세한 내용은 MQCFH - PCF 헤더 참조). 메시지 ID 및 상관 ID는 요청의 보고서 옵션에 따라 설정합니다.

명령 메시지의 PCF 헤더 유형이 MQCFT_COMMAND인 경우, 표준 응답만 생성됩니다. 그러한 명령은 z/OS를 제외하고 모든 플랫폼에서 지원됩니다. 이 애플리케이션은 z/OS에서 PCF를 지원하지 않습니다. IBM MQ Windows 탐색기도 PCF를 지원하지 않는 애플리케이션입니다(그러나 버전 6.0 또는 IBM MQ 탐색기 이상은 z/OS의 PCF를 지원함).

명령 메시지의 PCF 헤더 유형이 MQCFT_COMMAND_XR인 경우, 확장된 응답 또는 표준 응답이 생성됩니다. 그러한 명령은 z/OS 및 일부 기타 플랫폼에서 지원됩니다. z/OS에서 발행된 명령은 확장된 응답만 생성합니다. 기타 플랫폼에서 응답의 유형이 생성될 수도 있습니다.

단일 명령이 일반 오브젝트 이름을 지정하면, 자체 메시지에 일치하는 각 오브젝트에 대한 분리 응답이 리턴됩니다. 응답 발생의 경우, 일반 이름을 가진 단일 명령은 다중 개인 명령으로 처리됩니다(제어 필드 MQCFC_LAST 또는 MQCFC_NOT_LAST 제외). 그렇지 않으면, 하나의 명령 메시지는 하나의 응답 메시지를 생성합니다.

특정 PCF 응답은 요청되지 않은 경우에도 구조를 리턴할 수 있습니다. 이 구조는 항상 리턴됨으로서 응답(PCF(Programmable Command Format)의 정의)의 정의에 표시됩니다. 그 이유는 이러한 응답의 경우에 데이터가 적용되는 오브젝트를 식별하려면 응답에 오브젝트의 이름을 지정해야 하기 때문입니다.

응답을 위한 메시지 디스크립터

응답 메시지는 메시지 디스크립터에 다음 필드를 가집니다.

MsgType
이 필드는 MQMT_REPLY입니다.

MsgId
이 필드는 큐 관리자에 의해 생성됩니다.

CorrelId
이 필드는 명령 메시지의 보고서 옵션에 따라 생성됩니다.

Format
이 필드는 MQFMT_ADMIN입니다.

Encoding
MQENC_NATIVE로 설정하십시오.

CodedCharSetId
MQCCSI_Q_MGR로 설정하십시오.

Persistence
명령 메시지의 경우와 동일합니다.

Priority
명령 메시지의 경우와 동일합니다.

응답은 MQPMO_PASS_IDENTITY_CONTEXT로 생성됩니다.

표준 응답

MQCFT_COMMAND, 표준 응답의 헤더 유형을 가진 명령 메시지가 생성됩니다. 그러한 명령은 z/OS를 제외하고 모든 플랫폼에서 지원됩니다.

세 가지 종류의 표준 응답이 있습니다.

- 확인 응답
- 오류 응답
- 데이터 응답

확인 응답

이 응답은 MQCC_OK 또는 MQCC_WARNING의 *CompCode* 필드로, 명령어 형식 헤더로 시작하는 메시지로 구성됩니다.

MQCC_OK의 경우, *Reason*은 MQRC_NONE입니다.

MQCC_WARNING의 경우, *Reason*은 경고의 성질을 식별합니다. 이 경우에 명령 형식 헤더는 이 이유 코드에 적절한 하나 이상의 경고 매개변수 구조가 이어질 수도 있습니다.

이 경우, 조회 명령의 경우 추가 매개변수 구조가 다음 섹션에서 설명되는 대로 뒤에 옵니다.

오류 응답

명령에 오류가 있으면 하나 이상의 오류 응답 메시지가 송신됩니다. 일반적으로 하나의 응답 메시지만 있는 명령에 대해서도 둘 이상의 응답 메시지가 송신될 수 있습니다. 이러한 오류 응답 메시지의 경우, MQCFC_LAST 또는 MQCFC_NOT_LAST를 적절히 설정해야 합니다.

그러한 각 메시지는 특정 오류를 식별하는 MQCC_FAILED 및 *Reason* 필드의 *CompCode* 값으로 응답 형식 헤더로 시작합니다. 일반적으로, 각 메시지는 다른 오류를 설명합니다. 또한 각 메시지에는 헤더 뒤에 0 또는 하나(그 이상은 불가능)의 오류 매개변수 구조가 있습니다. 이 매개변수 구조는(하나가 있는 경우) 다음 중 하나를 포함하는 *Parameter* 필드를 가지는 MQCFIN 구조입니다.

- MQIACF_PARAMETER_ID

  구조의 *Value* 필드는 오류가 있는 매개변수의 매개변수 ID입니다(예: MQCA_Q_NAME).

- MQIACF_ERROR_ID

  이 값은 MQRC_UNEXPECTED_ERROR의 *Reason* 값(명령 형식 헤더와 함께) 사용됩니다. MQCFIN 구조의 *Value* 필드는 명령 서버에서 수신한 예상치 못한 이유 코드입니다.

- MQIACF_SELECTOR

  명령과 함께 송신된 목록 구조(MQCFIL)에 복제 선택자 또는 올바르지 않은 것이 포함되는 경우 이 값이 발생합니다. 명령 형식 헤더에서 *Reason* 필드는 오류를 식별하고 MQCFIN 구조에서 *Value* 필드가 오류가 있는 명령의 MQCFIL 구조의 매개변수 값입니다.

- MQIACF_ERROR_OFFSET

  이 값은 Ping 채널 명령에 데이터 비교 오류가 있을 때 발생합니다. 구조의 *Value* 필드는 Ping 채널 비교 오류의 오프셋입니다.

- MQIA_CODED_CHAR_SET_ID

  이 값은 수신 PCF 명령 메시지의 메시지 디스크립터에 있는 코드화 문자 세트 ID가 대상 큐 관리자의 코드화 문자 세트 ID와 일치하지 않을 때 발생합니다. 구조의 *Value* 필드는 큐 관리자의 코드화 문자 세트 ID입니다.

마지막(또는 유일한) 오류 응답 메시지는 MQCC_FAILED의 *CompCode* 필드 및 MQRCCF_COMMAND_FAILED의 *Reason* 필드를 가지는 요약 응답입니다. 이 메시지에는 헤더를 따르는 매개변수 구조가 없습니다.

데이터 응답

이 응답은 조회 명령에 대한(이전에 설명된 것처럼) 확인 응답으로 구성됩니다. 확인 응답은 PCF(Programmable Command Format)의 정의에 설명된 대로 요청된 데이터를 포함하는 추가 구조가 이어집니다.

애플리케이션은 특정 순서로 리턴되는 이러한 추가 매개변수 구조에 따라 다르지 않아야 합니다.

확장된 응답

z/OS에서 발행된 명령은 확장된 응답을 생성합니다.

세 가지 유형의 확장된 응답이 있습니다.

- 유형 MQCFT_XR_MSG를 가지는 메시지 응답
- 유형 MQCFT_XR_ITEM을 가지는 항목 응답
- 유형 MQCFT_XR_SUMMARY를 가지는 요약 응답

각 명령은 하나 이상의 응답 세트를 생성할 수 있습니다. 각각의 응답세트는 PCF 헤더의 *MsgSeqNumber* 필드에 1부터 순차적으로 번호를 매긴 하나 이상의 메시지로 구성됩니다. 각 세트의 마지막(또는 유일한) 응답의 *Control* 필드에 MQCFC_LAST 값이 있습니다. 세트에서 기타 모든 응답의 경우, 이 값은 MQCFC_NOT_LAST 입니다.

응답에는 *Parameter* 필드가 MQBACF_RESPONSE_SET로 설정된 하나 이상의 선택적 MQCFBS 구조가 포함될 수 있습니다. 이 값은 응답 세트 ID입니다. ID는 고유하고 응답을 포함하는 응답 세트를 식별합니다. 모든 응답 세트의 경우, 이를 식별하는 MQCFBS 구조가 있습니다.

확장된 응답에 최소 두 개의 매개변수 구조가 있습니다.

- *Parameter* 필드가 MQBACF_RESPONSE_ID로 설정된 MQCFBS 구조. 이 필드의 값은 응답이 속하는 응답 세트의 ID입니다. 첫 세트의 ID는 임의적입니다. 후속 세트에서 ID는 이전에 MQBACF_RESPONSE_SET 구조에서 공지화된 것입니다.
- 응답 세트가 나온 큐 관리자의 이름이 되는 값인 MQCACF_RESPONSE_Q_MGR_NAME으로 설정된 *Parameter* 필드를 가지는 MQCFST 구조.

많은 응답에는 추가 매개변수 구조가 있고 이러한 구조가 다음 섹션에서 설명됩니다.

MQCFC_LAST를 가지는 하나를 발견할 때까지 응답을 가져오는 것 이외에 세트에 있는 응답 수를 미리 판별할 수 없습니다. 어느 쪽 세트로서 있는 응답의 얼마나 많은 세트가 추가적 세트가 생성되는 것을 나타내려면 MQBACF_RESPONSE_SET 구조를 포함할 수도 있는지 미리 판별할 수 있습니다.

Inquire 명령에 대한 확장된 응답

Inquire 명령은 지정된 검색 기준에 일치하는 각 항목이 발견되었기 때문에 항목 응답(MQCFT_XR_ITEM 유형)을 일반적으로 생성합니다. 항목 응답에는 MQCC_OK의 값을 가지는 *CompCode* 필드가 있고 MQRC_NONE의 값을 가지는 *Reason*필드가 있습니다. PCF(Programmable Command Format)의 정의에서 설명된 대로 항목 및 해당 요청 속성을 설명하는 기타 매개변수 구조도 포함됩니다.

항목에 오류가 있는 경우, 헤더의 *CompCode* 필드에 MQCC_FAILED의 값이 있고 *Reason* 필드는 특정 오류를 식별합니다. 추가 매개변수 구조가 항목을 식별하기 위해 포함됩니다.

특정 Inquire 명령은 항목 응답뿐만 아니라 일반(특정 이름 아님)메시지 응답을 리턴할 수 있습니다. 이러한 응답은 MQCFT_XR_MSG의 정보 또는 오류 응답입니다.

Inquire 명령이 성공하면, 선택적으로 MQCC_OK의 *CompCode* 값 및 MQRC_NONE의*Reason* 필드 값으로 요약 응답(MQCFT_XR_SUMMARY 유형)일 수 있습니다.

Inquire 명령이 실패하면, 항목 응답이 리턴되고 선택적으로 MQCC_FAILED의 *CompCode* 값 및 MQRCCF_COMMAND_FAILED의 *Reason* 필드 값으로 요약 응답(MQCFT_XR_SUMMARY 유형)일 수 있습니다.

Inquire 이외의 명령에 대한 확장된 응답

성공적인 명령은 헤더의 *CompCode* 필드에 MQCC_OK의 값이 있고 *Reason* 필드에 MQRC_NONE의 값이 있는 메시지 응답을 생성합니다. 최소 하나의 메시지가 있습니다. 이는 정보(MQCFT_XR_MSG) 또는 요약(MQCFT_XR_SUMMARY)일 수 있습니다. 선택적으로 추가 정보(MQCFT_XR_MSG 유형) 메시지일 수 있습니다. 각 정보 메시지에는 명령에 대한 정보를 가지는 많은 추가 매개변수 구조가 포함될 수 있습니다. 발생할 수 있는 구조에 대한 개별 명령 설명을 참조하십시오.

실패한 명령은 오류 메시지 응답(MQCFT_XR_MSG 유형)을 생성합니다. 여기에서 헤더의 *CompCode* 필드에 MQCC_FAILED 값이 있고 *Reason* 필드는 특정 오류를 식별합니다. 각 메시지에는 오류에 대한 정보를 가지는 많은 추가 매개변수 구조가 포함될 수 있습니다. 발생할 수 있는 구조에 대한 개별 오류 설명을 참조하십시오. 정보 메시지 응답이 생성될 수 있습니다. 선택적으로 MQCC_FAILED의 *CompCode* 값 및 MQRCCF_COMMAND_FAILED의*Reason* 필드 값으로 요약 응답(MQCFT_XR_SUMMARY)일 수 있습니다.

CommandScope을 사용하여 명령에 대한 확장된 응답

명령은 *CommandScope* 매개변수를 사용하거나 *CommandScope* 매개변수를 사용하는 명령이 생성되도록 하는 경우, 명령이 수신되는 큐 관리자로부터 설정된 초기 응답이 있습니다. 그런 다음, 응답의 분리 세트가 명령이 전달되는 각 큐 관리자에 대해 생성됩니다(여러 개별 명령이 실행된 것처럼). 마지막으로 전체 요약 응답

(MQCFT_XR_SUMMARY 유형)을 포함하는 수신 큐 관리자로부터 설정된 응답이 있습니다. MQCACF_RESPONSE_Q_MGR_NAME 매개변수 구조는 각 세트를 생성하는 큐 관리자를 식별합니다.

초기 응답 세트에는 다음 추가 매개변수 구조가 있습니다.

- MQIACF_COMMAND_INFO (MQCFIN). 이 구조의 가능한 값은 MQCMDI_CMDSCOPE_ACCEPTED 또는 MQCMDI_CMDSCOPE_GENERATED입니다.
- MQIACF_CMDSCOPE_Q_MGR_COUNT (MQCFIN). 이 구조는 명령이 보내지는 큐 관리자의 수를 표시합니다.

PCF 명령에 대한 권한 검사

PCF 명령이 처리될 때, 명령 메시지에 있는 메시지 디스크립터의 *UserIdentifier*가 필수 IBM MQ 오브젝트 권한 검사에 사용됩니다. 권한 검사는 이 주제에서 설명된 대로 각 플랫폼에 다르게 구현됩니다.

검사는 명령이 처리되는 시스템에서 수행되므로 이 사용자 ID는 반드시 대상 시스템에 존재해야 하며 명령 처리에 필요한 권한을 갖고 있어야 합니다. 메시지가 원격 시스템에서 발생한 경우, 대상 시스템에 기존 ID를 달성하는 한 방법은 로컬 시스템과 원격 시스템 모두에 대해 일치하는 사용자 ID를 가지고 있는 것입니다.

IBM MQ for IBM i

IBM i

PCF 명령을 처리하려면, 사용자 ID는 대상 시스템에 IBM MQ 오브젝트를 위한 *dsp* 권한이 있어야 합니다.

또한 IBM MQ 오브젝트 권한 검사는 17 페이지의 표 1에 나타난 바와 같이, 특정 PCF 명령에 대하여 실행됩니다.

대부분의 경우에 이러한 검사는 로컬 시스템에 실행된 동등한 IBM MQ CL 명령에 의해 수행된 해당 검사와 같은 검사입니다. IBM MQ 권한에서 IBM i 시스템 권한으로의 맵핑 및 IBM MQ CL 명령에 대한 권한 요구사항에 대한 자세한 정보는 IBM i 에서 보안 설정 을 참조하십시오. 종료와 관계가 있는 보안에 대한 세부사항은 보안 엑시트를 사용하는 링크 레벨 보안 문서에 제공됩니다.

다음 명령 중 하나를 처리하려면 사용자 ID가 그룹 프로파일 QMQMADM의 구성원이어야 합니다.

- 채널 ping
- 채널 변경
- 채널 복사
- 채널 작성
- 채널 삭제
- 채널 재설정
- 채널 분석
- 채널 시작
- 채널 중지
- 채널 시작기 시작
- 채널 리스너 시작

IBM MQ for Windows, 유닉스 및 Linux 시스템

Windows Linux UNIX

PCF 명령을 처리하기 위해 사용자 ID에는 대상 시스템에 큐 관리자 오브젝트에 대한 *dsp* 권한이 있어야 합니다. 또한 IBM MQ 오브젝트 권한 검사는 17 페이지의 표 1에 나타난 바와 같이, 특정 PCF 명령에 대하여 실행됩니다.

다음 명령 중 하나를 처리하려면 사용자 ID가 그룹 *mqm*에 속해야 합니다.

참고: Windows **전용**의 경우, 사용자 ID는 관리자 또는 그룹 *mqm*에 속할 수 있습니다.

- 채널 변경
- 채널 복사

- 채널 작성
- 채널 삭제
- 채널 ping
- 채널 재설정
- 채널 시작
- 채널 중지
- 채널 시작기 시작
- 채널 리스너 시작
- 채널 분석
- 클러스터 재설정
- 클러스터 새로 고치기
- 큐 관리자 일시중단
- 큐 관리자 재개

IBM WebSphere MQ for HP Integrity NonStop Server

PCF 명령을 처리하기 위해 사용자 ID에는 대상 시스템에 큐 관리자 오브젝트에 대한 *dsp* 권한이 있어야 합니다. 또한 IBM MQ 오브젝트 권한 검사는 17 페이지의 표 1에 나타난 바와 같이, 특정 PCF 명령에 대하여 실행됩니다.

다음 명령 중 하나를 처리하려면 사용자 ID가 그룹 *mqm*에 속해야 합니다.

- 채널 변경
- 채널 복사
- 채널 작성
- 채널 삭제
- 채널 ping
- 채널 재설정
- 채널 시작
- 채널 중지
- 채널 시작기 시작
- 채널 리스너 시작
- 채널 분석
- 클러스터 재설정
- 클러스터 새로 고치기
- 큐 관리자 일시중단
- 큐 관리자 재개

IBM MQ 오브젝트 권한

표 *1. Windows, HP Integrity NonStop Server,* IBM i *IBM i,* 유닉스 및 *Linux* 시스템 - 오브젝트 권한

명령	**IBM MQ 오브젝트 권한**	**클래스 권한(오브젝트 유형의 경우)**
인증 정보 변경	dsp 및 chg	해당사항 없음
채널 변경	dsp 및 chg	해당사항 없음
채널 리스너 변경	dsp 및 chg	해당사항 없음

표 *1. Windows, HP Integrity NonStop Server,* IBM i *IBM i,* 유닉스 및 *Linux* 시스템 - 오브젝트 권한 *(*계속*)*

명령	**IBM MQ 오브젝트 권한**	**클래스 권한(오브젝트 유형의 경우)**
클라이언트 연결 채널 변경	dsp 및 chg	해당사항 없음
이름 목록 변경	dsp 및 chg	해당사항 없음
프로세스 변경	dsp 및 chg	해당사항 없음
큐 변경	dsp 및 chg	해당사항 없음
큐 관리자 변경	chg 참고 *3* 및 참고 *5* 참조	해당사항 없음
서비스 변경	dsp 및 chg	해당사항 없음
큐 지우기	clr	해당사항 없음
인증 정보 복사	dsp	crt
인증 정보 복사(바꾸기) 참고 *1* 참조	dsp에서 chg까지	crt
채널 복사	dsp	crt
채널 복사(바꾸기) 참고 *1* 참조	dsp에서 chg까지	crt
채널 리스너 복사	dsp	crt
채널 리스너 복사(바꾸기) 참고 *1* 참조	dsp에서 chg까지	crt
클라이언트 연결 채널 복사	dsp	crt
클라이언트 연결 채널 복사(바꾸기) 참고 *1* 참조	dsp에서 chg까지	crt
이름 목록 복사	dsp	crt
이름 목록 복사(바꾸기) 참고 *1* 참조	dsp에서 dsp 및 chg까지	crt
프로세스 복사	dsp	crt
프로세스 복사(바꾸기) 참고 *1* 참조	dsp에서 chg까지	crt
큐 복사	dsp	crt
큐 복사(바꾸기) 참고 *1* 참조	dsp에서 dsp 및 chg까지	crt
인증 정보 작성	*(*시스템 기본 인증 정보*)* dsp	crt
인증 정보 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 인증 정보*)* dsp에서 chg까지	crt
채널 작성	*(*시스템 기본 채널*)* dsp	crt
채널 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 채널*)* dsp에서 chg까지	crt
채널 리스너 작성	*(*시스템 기본 리스너*)* dsp	crt
채널 리스너 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 리스너*)* dsp에서 chg까지	crt
클라이언트 연결 채널 작성	*(*시스템 기본 채널*)* dsp	crt
클라이언트 연결 채널 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 채널*)* dsp에서 chg까지	crt
이름 목록 작성	*(*시스템 기본 이름 목록*)* dsp	crt

표 *1. Windows, HP Integrity NonStop Server,* IBM i *IBM i,* 유닉스 및 *Linux* 시스템 - 오브젝트 권한 *(계속)*

명령	**IBM MQ 오브젝트 권한**	**클래스 권한(오브젝트 유형의 경우)**
이름 목록 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 이름 목록*)* dsp에서 dsp 및 chg까지	crt
프로세스 작성	*(*시스템 기본 프로세스*)* dsp	crt
프로세스 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 프로세스*)* dsp에서 chg까지	crt
큐 작성	*(*시스템 기본 큐*)* dsp	crt
큐 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 큐*)* dsp에서 dsp 및 chg까지	crt
서비스 작성	*(*시스템 기본 큐*)* dsp	crt
서비스 작성(바꾸기) 참고 *1* 참조	*(*시스템 기본 큐*)* dsp에서 chg까지	crt
인증 정보 삭제	dsp 및 dlt	해당사항 없음
권한 레코드 삭제	*(*큐 관리자 오브젝트*)* chg 참고 *4* 참조	참고 *4* 참조
채널 삭제	dsp 및 dlt	해당사항 없음
채널 리스너 삭제	dsp 및 dlt	해당사항 없음
클라이언트 연결 채널 삭제	dsp 및 dlt	해당사항 없음
이름 목록 삭제	dsp 및 dlt	해당사항 없음
프로세스 삭제	dsp 및 dlt	해당사항 없음
큐 삭제	dsp 및 dlt	해당사항 없음
서비스 삭제	dsp 및 dlt	해당사항 없음
인증 정보 조회	dsp	해당사항 없음
권한 레코드 조회	참고 *4* 참조	참고 *4* 참조
채널 조회	dsp	해당사항 없음
채널 리스너 조회	dsp	해당사항 없음
채널 상태 조회(**ChannelType** MQCHT_CLSSDR의 경우)	inq	해당사항 없음
클라이언트 연결 채널 조회	dsp	해당사항 없음
이름 목록 조회	dsp	해당사항 없음
프로세스 조회	dsp	해당사항 없음
큐 조회	dsp	해당사항 없음
큐 관리자 조회	참고 *3* 참조	해당사항 없음
큐 상태 조회	dsp	해당사항 없음
서비스 조회	dsp	해당사항 없음
채널 ping	ctrl	해당사항 없음
큐 관리자 ping	참고 *3* 참조	해당사항 없음
큐 관리자 새로 고치기	(큐 관리자 오브젝트) chg	해당사항 없음

표 *1. Windows, HP Integrity NonStop Server,* IBM i *IBM i,* 유닉스 및 *Linux* 시스템 - 오브젝트 권한 *(계속)*		
명령	**IBM MQ 오브젝트 권한**	**클래스 권한(오브젝트 유형의 경우)**
보안 새로 고치기(**SecurityType** MQSECTYPE_SSL의 경우)	(큐 관리자 오브젝트) chg	해당사항 없음
채널 재설정	ctrlx	해당사항 없음
큐 관리자 재설정	(큐 관리자 오브젝트) chg	해당사항 없음
큐 통계 재설정	dsp 및 chg	해당사항 없음
채널 분석	ctrlx	해당사항 없음
권한 레코드 설정	*(*큐 관리자 오브젝트*)* chg 참고 *4* 참조	참고 *4* 참조
채널 시작	ctrl	해당사항 없음
채널 중지	ctrl	해당사항 없음
연결 중지	(큐 관리자 오브젝트) chg	해당사항 없음
리스너 시작	ctrl	해당사항 없음
리스너 중지	ctrl	해당사항 없음
서비스 시작	ctrl	해당사항 없음
서비스 중지	ctrl	해당사항 없음
나가기	참고 *2* 참조	참고 *2* 참조

참고사항:

1. 대체할 오브젝트가 존재하면 이 명령이 적용됩니다. 그렇지 않으면 권한 검사는 대체없는 작성 또는 복사에 대한 것입니다.
2. 필수 권한은 이스케이프 텍스트에 의해 정의되는 MQSC 명령에 의해 판별되고 이는 이전 명령 중 하나와 동등합니다.
3. PCF 명령을 처리하기 위해 사용자 ID에는 대상 시스템에 큐 관리자 오브젝트에 대한 dsp 권한이 있어야 합니다.
4. 이 PCF 명령은 명령 서버가 -a 매개변수로 시작되지 않는 경우 권한이 부여됩니다. 기본적으로 큐 관리자가 -a 매개변수 없이 시작될 때 명령 서버가 시작됩니다. 추가 정보는 시스템 관리 안내서를 참조하십시오.
5. 큐 관리자에 사용자 ID *chg* 권한을 부여하면 모든 그룹 및 사용자에 대한 권한 레코드를 설정하기 위한 기능을 제공합니다. 일반 사용자 또는 애플리케이션에 이 권한을 부여하지 마십시오.

IBM MQ에서는 보안 검사에 사용자 엑시트 프로그램을 제공할 수 있도록 일부 채널 보안 엑시트 지점을 제공하기도 합니다. 세부사항은 채널 표시 매뉴얼에 제공됩니다.

IBM MQ for z/OS

z/OS

z/OS의 권한 점검에 대한 정보는 태스크 1: z/OS 시스템 매개변수 식별 을 참조하십시오.

MQAI를 사용하여 PCF의 사용 단순화

MQAI는 AIX, HP-UX, IBM i, Linux, Solaris및 윈도우에서 사용 가능한 IBM MQ 에 대한 관리 인터페이스입니다.

MQAI는 데이터 백을 사용하여 큐 관리자에서 관리 태스크를 수행합니다. 데이터 백을 사용하면 PCF를 사용하는 것 보다 더 용이한 방법으로 오브젝트의 특성(또는 매개변수)을 처리할 수 있습니다.

MQAI를 사용하는 장점은 다음과 같습니다.

PCF 메시지의 사용을 단순화합니다.
MQAI는 IBM MQ를 관리하기 위한 더 쉬운 방법입니다. MQAI를 사용하면, 사용자의 PCF 메시지를 작성할 필요가 없습니다. 이는 복잡한 데이터 구조와 연관되는 문제점을 방지합니다.

MQI 호출을 사용하여 작성된 프로그램에서 매개변수를 전달하기 위해, PCF 메시지는 명령 및 문자열 또는 정수 데이터에 대한 세부사항을 포함해야 합니다. 수동으로 이 구성을 작성하려면, 여러 명령문을 모든 구성을 위한 사용자의 프로그램에 추가하여야 하고 메모리 영역을 할당해야 합니다. 이는 길고 힘든 작업일 수 있습니다.

MQAI를 사용하여 작성된 프로그램은 매개변수를 적절한 데이터 백으로 전달하고 각 구조에 대해 하나의 명령문만 필요합니다. MQAI 데이터 백을 사용하면 배열을 처리하고 스토리지를 할당하기 위한 필요성을 제거하고 PCF의 세부사항으로부터 어느 정도의 격리를 제공합니다.

오류 조건을 보다 쉽게 핸들링합니다.
PCF 명령에서 리턴 코드를 다시 가져오는 것은 어렵습니다. MQAI는 오류 조건을 핸들링하기 위해 프로그램을 더 용이하게 합니다.

데이터 백을 작성하고 채운 후, `mqExecute` 호출을 사용하여 관리 명령 메시지를 큐 관리자의 명령 서버에 보낼 수 있습니다. 이 호출은 응답 메시지도 기다립니다. `mqExecute` 호출은 명령 서버를 가진 교환을 핸들링하고 응답 백에서 응답을 리턴합니다.

MQAI에 대한 자세한 정보는 21 페이지의 『MQAI(IBM MQ Administration Interface)에 대한 소개』의 내용을 참조하십시오.

관련 정보
IBM MQ 관리 인터페이스 참조

MQAI(IBM MQ Administration Interface)에 대한 소개

MQAI(IBM MQ Administration Interface)는 IBM MQ에 대한 프로그래밍 인터페이스입니다. 이는 프로그래밍 가능 명령 형식(PCF)을 사용하는 것보다 더 용이한 방법으로 오브젝트의 특성(또는 매개변수)을 핸들링하기 위해 데이터 백을 사용하여 IBM MQ 큐 관리자에서 관리 태스크를 수행합니다.

MQAI 개념 및 용어

MQAI는 IBM MQ에 대한 프로그래밍 인터페이스이며, C 언어 및 Visual Basic for Windows를 사용합니다. 이는 z/OS 이외의 플랫폼에서 사용 가능합니다.

이는 데이터 백을 사용하여 IBM MQ 큐 관리자에서 관리 태스크를 수행합니다. 데이터 백을 사용하면 기타 관리 인터페이스 프로그래밍 가능 명령 형식(PCF)을 사용하는 것 보다 더 용이한 방법으로 오브젝트의 특성(또는 매개변수)을 핸들링할 수 있습니다. MQAI를 사용하면 MQGET 및 MQPUT 호출을 사용하는 것보다 쉽게 PCF를 조작할 수 있습니다.

데이터 백에 대한 자세한 정보는 47 페이지의 『데이터 백』의 내용을 참조하십시오. PCF에 대한 자세한 정보는 9 페이지의 『프로그래밍 가능 명령 형식 소개』의 내용을 참조하십시오.

MQAI의 사용

MQAI를 사용하여 다음을 수행할 수 있습니다.

- PCF 메시지의 사용을 단순화합니다. MQAI는 IBM MQ를 관리하기 위한 쉬운 방법입니다. PCF 메시지를 작성할 필요는 없습니다. 따라서, 복잡한 데이터 구조와 연관된 문제점을 피하십시오.
- 오류 조건을 더 쉽게 핸들링합니다. MQSC(IBM MQ script) 명령에서 리턴 코드를 가져오는 것은 어렵지만, MQAI는 오류 조건을 핸들링하기 위해 프로그램을 더 용이하게 합니다.
- 애플리케이션 사이에서 데이터를 교환합니다. 애플리케이션 데이터가 PCF 형식으로 보내지고 MQAI에 의해 패키지로 묶이거나 풀릴 수 있습니다. 메시지 데이터가 정수 및 문자열로 구성되는 경우, MQAI를 사용하여 PCF 데이터에 대한 IBM MQ 내장 데이터 변환을 사용할 수 있습니다. 이는 데이터 변환 엑시트를 작성할 필요가 없습니다. MQAI를 사용하여 IBM MQ를 관리하고 애플리케이션 사이에 데이터를 교환하는 것에 대한 자세한 정보는 20 페이지의 『MQAI를 사용하여 PCF의 사용 단순화』의 내용을 참조하십시오.

MQAI 사용의 예

표시되는 목록은 MQAI의 사용을 증명하는 일부 예 프로그램을 제공합니다. 샘플은 다음 태스크를 수행합니다.

1. 로컬 큐를 작성한다. 22 페이지의 『로컬 큐를 작성하기 위한 샘플 C 프로그램(amqsaicq.c)』
2. 단순 이벤트 모니터를 사용하여 화면에 이벤트를 표시합니다.26 페이지의 『이벤트 모니터를 사용하여 이벤트를 표시하는 샘플 C 프로그램(amqsaiem.c)』
3. 모든 로컬 큐 및 현재 용량 목록을 인쇄합니다.38 페이지의 『큐 및 인쇄 정보를 조회하기 위한 샘플 C 프로그램(amqsailq.c)』
4. 모든 채널 및 유형 목록을 인쇄합니다.33 페이지의 『채널 오브젝트를 조회하기 위한 샘플 C 프로그램(amqsaicl.c)』

MQAI 애플리케이션 빌드

MQAI를 사용하여 애플리케이션을 빌드하려면 IBM MQ에 도움이 된 것과 동일한 라이브러리에 링크하십시오. IBM MQ 애플리케이션 빌드 방법에 대한 정보는 절차적 애플리케이션 빌드를 참조하십시오.

MQAI를 사용하여 IBM MQ를 구성하기 위한 힌트 및 팁

MQAI는 명령 서버 자체에서 직접 처리하지 않고 PCF 메시지를 사용하여 관리 명령을 명령 서버로 송신합니다. MQAI를 사용하여 IBM MQ를 구성하기 위한 팁을 42 페이지의 『IBM MQ 구성을 위한 힌트 및 팁』에서 찾을 수 있습니다.

MQAI(IBM MQ Administration Interface)

IBM MQ for Windows, AIX, IBM i IBM i, Linux, HP-UX 및 Solaris에서는 MQAI(IBM MQ Administration Interface)를 지원합니다. MQAI는 PCF를 송신하고 수신하기 위해 MQI에 대한 대안을 주는 IBM MQ에 대한 프로그래밍 인터페이스입니다.

MQAI는 데이터 백을 사용합니다. 이를 사용하면 MQAI의 방법으로 직접 PCF를 사용하는 것 보다 더 쉽게 오브젝트의 특성(또는 매개변수)를 핸들링할 수 있습니다.

MQAI에서는 하나의 명령문만 각 구조에 필요하도록 데이터 백에서 매개변수를 전달하여 PCF 메시지에 대한 보다 용이한 프로그래밍 액세스를 제공합니다. 이 액세스는 배열을 핸들링하고 스토리지를 할당하기 위해 매개변수에 대한 필요성을 제거하고 PCF의 세부사항으로부터 일부 격리를 제공합니다.

MQAI는 명령 서버에 PCF 메시지를 보내고 응답을 대기함으로써 IBM MQ를 관리합니다.

MQAI는 이 매뉴얼의 두 번째 절에서 설명됩니다. MQAI에 대한 컴포넌트 오브젝트 모델 인터페이스의 설명은 Java™ 사용 문서를 참조하십시오.

로컬 큐를 작성하기 위한 샘플 C 프로그램(amqsaicq.c)

샘플 C 프로그램 `amqsaicq.c`는 MQAI를 사용하여 로컬 큐를 작성합니다.

```
/*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICQ.C                                                  */
/*                                                                           */
/* Description:  Sample C program to create a local queue using the          */
/*               IBM MQ Administration Interface (MQAI).                     */
/*                                                                           */
/* Statement:    Licensed Materials - Property of IBM                        */
/*                                                                           */
/*               84H2000, 5765-B73                                           */
/*               84H2001, 5639-B42                                           */
/*               84H2002, 5765-B74                                           */
/*               84H2003, 5765-B75                                           */
/*               84H2004, 5639-B43                                           */
/*                                                                           */
/*               (C) Copyright IBM Corp. 1999, 2023.                         */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Function:                                                                 */
```

```
/*    AMQSAICQ is a sample C program that creates a local queue and is an     */
/*    example of the use of the mqExecute call.                               */
/*                                                                            */
/*     - The name of the queue to be created is a parameter to the program.   */
/*                                                                            */
/*     - A PCF command is built by placing items into an MQAI bag.            */
/*       These are:-                                                          */
/*            - The name of the queue                                         */
/*            - The type of queue required, which, in this case, is local.    */
/*                                                                            */
/*     - The mqExecute call is executed with the command MQCMD_CREATE_Q.      */
/*       The call generates the correct PCF structure.                        */
/*       The call receives the reply from the command server and formats into */
/*       the response bag.                                                    */
/*                                                                            */
/*     - The completion code from the mqExecute call is checked and if there  */
/*       is a failure from the command server then the code returned by the   */
/*       command server is retrieved from the system bag that is              */
/*       embedded in the response bag to the mqExecute call.                  */
/*                                                                            */
/* Note: The command server must be running.                                  */
/*                                                                            */
/*                                                                            */

/****************************************************************************/
/*                                                                          */
/* AMQSAICQ has 2 parameters - the name of the local queue to be created    */
/*                           - the queue manager name (optional)            */
/*                                                                          */
/****************************************************************************/
/****************************************************************************/
/* Includes                                                                 */
/****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                      */
#include <cmqcfc.h>                     /* PCF                      */
#include <cmqbc.h>                      /* MQAI                     */

void CheckCallResult(MQCHAR *, MQLONG , MQLONG );
void CreateLocalQueue(MQHCONN, MQCHAR *);

int main(int argc, char *argv[])
{
   MQHCONN hConn;                         /* handle to IBM MQ connection   */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name          */
   MQLONG connReason;                     /* MQCONN reason code            */
   MQLONG compCode;                       /* completion code               */
   MQLONG reason;                         /* reason code                   */

   /*************************************************************************/
   /* First check the required parameters                                   */
   /*************************************************************************/
   printf("Sample Program to Create a Local Queue\n");
   if (argc < 2)
   {
     printf("Required parameter missing - local queue name\n");
     exit(99);
   }

   /*************************************************************************/
   /* Connect to the queue manager                                          */
   /*************************************************************************/
   if (argc > 2)
      strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
      MQCONN(QMName, &hConn, &compCode, &connReason);

/****************************************************************************/
/* Report reason and stop if connection failed                              */
/****************************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("MQCONN", compCode, connReason);
      exit( (int)connReason);
   }


/****************************************************************************/
/* Call the routine to create a local queue, passing the handle to the      */
```

```
/* queue manager and also passing the name of the queue to be created.       */
/*****************************************************************************/
   CreateLocalQueue(hConn, argv[1]);

   /*****************************************************************************/
   /* Disconnect from the queue manager if not already connected                */
   /*****************************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
   }
   return 0;

}
/*****************************************************************************/
/*                                                                           */
/* Function:    CreateLocalQueue                                             */
/* Description: Create a local queue by sending a PCF command to the command */
/*              server.                                                      */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Handle to the queue manager                            */
/*                    Name of the queue to be created                        */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.    */
/*        The call generates the correct PCF structure.                      */
/*        The default options to the call are used so that the command is sent*/
/*        to the SYSTEM.ADMIN.COMMAND.QUEUE.                                 */
/*        The reply from the command server is placed on a temporary dynamic */
/*        queue.                                                             */
/*        The reply is read from the temporary queue and formatted into the  */
/*        response bag.                                                      */
/*                                                                           */
/*        The completion code from the mqExecute call is checked and if there */
/*        is a failure from the command server then the code returned by the */
/*        command server is retrieved from the system bag that is            */
/*        embedded in the response bag to the mqExecute call.                */
/*                                                                           */
/*****************************************************************************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG reason;                       /* reason code                     */
   MQLONG compCode;                     /* completion code                 */
   MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute    */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute   */
   MQHBAG resultBag;                    /* result bag from mqExecute       */
   MQLONG mqExecuteCC;                  /* mqExecute completion code       */
   MQLONG mqExecuteRC;                  /* mqExecute reason code           */

   printf("\nCreating Local Queue %s\n\n", qName);


   /*****************************************************************************/
   /* Create a command Bag for the mqExecute call. Exit the function if the   */
   /* create fails.                                                             */
   /*****************************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
   CheckCallResult("Create the command bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /*****************************************************************************/
   /* Create a response Bag for the mqExecute call, exit the function if the  */
   /* create fails.                                                             */
   /*****************************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
   CheckCallResult("Create the response bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /*****************************************************************************/
   /* Put the name of the queue to be created into the command bag. This will */
   /* be used by the mqExecute call.                                            */
   /*****************************************************************************/
   mqAddString(commandBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, qName, &compCode,
               &reason);
   CheckCallResult("Add q name to command bag", compCode, reason);
```

```
/*************************************************************************/
/* Put queue type of local into the command bag. This will be used by the  */
/* mqExecute call.                                                       */
/*************************************************************************/
mqAddInteger(commandBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type to command bag", compCode, reason);

/*************************************************************************/
/* Send the command to create the required local queue.                  */
/* The mqExecute call will create the PCF structure required, send it to   */
/* the command server and receive the reply from the command server into   */
/* the response bag.                                                     */
/*************************************************************************/
mqExecute(hConn,                    /* IBM MQ connection handle            */
          MQCMD_CREATE_Q,           /* Command to be executed              */
          MQHB_NONE,                /* No options bag                      */
          commandBag,               /* Handle to bag containing commands    */
          responseBag,              /* Handle to bag to receive the response*/
          MQHO_NONE,                /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,                /* Create a dynamic q for the response  */
          &compCode,           /* Completion code from the mqExecute   */
          &reason);            /* Reason code from mqExecute call      */

if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName>\n")
   MQDISC(&hConn, &compCode, &reason);
   CheckCallResult("MQDISC", compCode, reason);
   exit(98);
}


/*************************************************************************/
/* Check the result from mqExecute call and find the error if it failed.   */
/*************************************************************************/
if ( compCode == MQCC_OK )
   printf("Local queue %s successfully created\n", qName);
else
{
   printf("Creation of local queue %s failed: Completion Code = %d
            qName, compCode, reason);
   if (reason == MQRCCF_COMMAND_FAILED)
   {
      /*********************************************************************/
      /* Get the system bag handle out of the mqExecute response bag.      */
      /* This bag contains the reason from the command server why the      */
      /* command failed.                                                   */
      /*********************************************************************/
      mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
                   &reason);
      CheckCallResult("Get the result bag handle", compCode, reason);

      /*********************************************************************/
      /* Get the completion code and reason code, returned by the command  */
      /* server, from the embedded error bag.                              */
      /*********************************************************************/
      mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                       &compCode, &reason);
      CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
      mqInquireInteger(resultBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                       &compCode, &reason);
      CheckCallResult("Get the reason code from the result bag", compCode,
                       reason);
      printf("Error returned by the command server: Completion code = %d :
              Reason = %d\n", mqExecuteCC, mqExecuteRC);
   }
}
/*************************************************************************/
/* Delete the command bag if successfully created.                       */
/*************************************************************************/
if (commandBag != MQHB_UNUSABLE_HBAG)
{
   mqDeleteBag(&commandBag, &compCode, &reason);
   CheckCallResult("Delete the command bag", compCode, reason);
}

/*************************************************************************/
/* Delete the response bag if successfully created.                      */
/*************************************************************************/
if (responseBag != MQHB_UNUSABLE_HBAG)
{
```

```
      mqDeleteBag(&responseBag, &compCode, &reason);
      CheckCallResult("Delete the response bag", compCode, reason);
   }
} /* end of CreateLocalQueue */


/******************************************************************************/
/*                                                                            */
/* Function: CheckCallResult                                                  */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* Input Parameters:  Description of call                                     */
/*                    Completion code                                         */
/*                    Reason code                                             */
/*                                                                            */
/* Output Parameters: None                                                    */
/*                                                                            */
/* Logic: Display the description of the call, the completion code and the    */
/*        reason code if the completion code is not successful                */
/*                                                                            */
/******************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %d :
                 Reason = %d\n", callText, cc, rc);

}
```

이벤트 모니터를 사용하여 이벤트를 표시하는 샘플 C 프로그램(amqsaiem.c)

샘플 C 프로그램 amqsaiem.c는 MQAI를 사용하여 기본 이벤트 모니터를 보여줍니다.

```


******************************************************************************/
/*                                                                            */
/* Program name: AMQSAIEM.C                                                   */
/*                                                                            */
/* Description:  Sample C program to demonstrate a basic event monitor        */
/*               using the IBM MQ Admin Interface (MQAI).                     */
/* Licensed Materials - Property of IBM                                       */
/*                                                                            */
/* 63H9336                                                                    */
/* (c) Copyright IBM Corp. 1999, 2023. All Rights Reserved.                   */
/*                                                                            */
/* US Government Users Restricted Rights - Use, duplication or                */
/* disclosure restricted by GSA ADP Schedule Contract with                    */
/* IBM Corp.                                                                  */
/******************************************************************************/
/*                                                                            */
/* Function:                                                                  */
/*    AMQSAIEM is a sample C program that demonstrates how to write a simple  */
/*    event monitor using the mqGetBag call and other MQAI calls.             */
/*                                                                            */
/*    The name of the event queue to be monitored is passed as a parameter    */
/*    to the program. This would usually be one of the system event queues:-  */
/*            SYSTEM.ADMIN.QMGR.EVENT        Queue Manager events             */
/*            SYSTEM.ADMIN.PERFM.EVENT       Performance events               */
/*            SYSTEM.ADMIN.CHANNEL.EVENT     Channel events                   */
/*            SYSTEM.ADMIN.LOGGER.EVENT      Logger events                    */
/*                                                                            */
/*    To monitor the queue manager event queue or the performance event queue,*/
/*    the attributes of the queue manager need to be changed to enable        */
/*    these events. For more information about this, see Part 1 of the        */
/*    Programmable System Management book. The queue manager attributes can   */
/*    be changed using either MQSC commands or the MQAI interface.            */
/*    Channel events are enabled by default.                                  */
/*                                                                            */
/* Program logic                                                              */
/*    Connect to the Queue Manager.                                           */
/*    Open the requested event queue with a wait interval of 30 seconds.      */
/*    Wait for a message, and when it arrives get the message from the queue  */
/*    and format it into an MQAI bag using the mqGetBag call.                 */
/*    There are many types of event messages and it is beyond the scope of    */
/*    this sample to program for all event messages. Instead the program      */
/*    prints out the contents of the formatted bag.                           */
/*    Loop around to wait for another message until either there is an error  */
/*    or the wait interval of 30 seconds is reached.                          */
```

```
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored    */
/*                           - the queue manager name (optional)    */
/*                                                                  */
/********************************************************************

/********************************************************************/
/* Includes                                                         */
/********************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                      */
#include <cmqcfc.h>                     /* PCF                      */
#include <cmqbc.h>                      /* MQAI                     */

/********************************************************************/
/* Macros                                                           */
/********************************************************************/
#if MQAT_DEFAULT == MQAT_WINDOWS_NT
  #define Int64 "I64"
#elif defined(MQ_64_BIT)
  #define Int64 "l"
#else
  #define Int64 "ll"
#endif

/********************************************************************/
/* Function prototypes                                              */
/********************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);
void GetQEvents(MQHCONN, MQCHAR *);
int PrintBag(MQHBAG);
int PrintBagContents(MQHBAG, int);

/********************************************************************/
/* Function: main                                                   */
/********************************************************************/
int main(int argc, char *argv[])
{
   MQHCONN hConn;                       /* handle to connection     */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QM name     */
   MQLONG reason;                       /* reason code              */
   MQLONG connReason;                   /* MQCONN reason code       */
   MQLONG compCode;                     /* completion code          */

   /*****************************************************************/
   /* First check the required parameters                           */
   /*****************************************************************/
   printf("Sample Event Monitor (times out after 30 secs)\n");
   if (argc < 2)
   {
     printf("Required parameter missing - event queue to be monitored\n");
     exit(99);
   }

   /*****************************************************************/
   /* Connect to the queue manager                                  */
   /*****************************************************************/
   if (argc > 2)
     strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(QMName, &hConn, &compCode, &connReason);
   /*****************************************************************/
   /* Report the reason and stop if the connection failed           */
   /*****************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("MQCONN", compCode, connReason);
      exit( (int)connReason);
   }

   /*****************************************************************/
   /* Call the routine to open the event queue and format any event messages  */
   /* read from the queue.                                          */
   /*****************************************************************/
   GetQEvents(hConn, argv[1]);

   /*****************************************************************/
```

```
   /* Disconnect from the queue manager if not already connected             */
   /*************************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
   }

   return 0;

}

/*************************************************************************/
/*                                                                       */
/* Function: CheckCallResult                                             */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/* Input Parameters:  Description of call                                */
/*                    Completion code                                    */
/*                    Reason code                                        */
/*                                                                       */
/* Output Parameters: None                                               */
/*                                                                       */
/* Logic: Display the description of the call, the completion code and the    */
/*        reason code if the completion code is not successful           */
/*                                                                       */
/*************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %d : Reason = %d\n",
                callText, cc, rc);

}

/*************************************************************************/
/*                                                                       */
/* Function: GetQEvents                                                  */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/* Input Parameters:  Handle to the queue manager                        */
/*                    Name of the event queue to be monitored            */
/*                                                                       */
/* Output Parameters: None                                               */
/*
/* Logic:   Open the event queue.                                        */
/*          Get a message off the event queue and format the message into     */
/*          a bag.                                                       */
/*          A real event monitor would need to be programmed to deal with     */
/*          each type of event that it receives from the queue. This is       */
/*          outside the scope of this sample, so instead, the contents of     */
/*          the bag are printed.                                         */
/*          The program waits for 30 seconds for an event message and then    */
/*          terminates if no more messages are available.                */
/*                                                                       */
/*************************************************************************/
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG openReason;                     /* MQOPEN reason code             */
   MQLONG reason;                         /* reason code                    */
   MQLONG compCode;                       /* completion code                */
   MQHOBJ eventQueue;                     /* handle to event queue          */

   MQHBAG eventBag = MQHB_UNUSABLE_HBAG;    /* event bag to receive event msg */
   MQOD   od = {MQOD_DEFAULT};              /* Object Descriptor              */
   MQMD   md = {MQMD_DEFAULT};              /* Message Descriptor             */
   MQGMO  gmo = {MQGMO_DEFAULT};            /* get message options            */
   MQLONG bQueueOK = 1;                     /* keep reading msgs while true   */

   /*************************************************************************/
   /* Create an Event Bag in which to receive the event.                    */
   /* Exit the function if the create fails.                                */
   /*************************************************************************/
   mqCreateBag(MQCBO_USER_BAG, &eventBag, &compCode, &reason);
   CheckCallResult("Create event bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /*************************************************************************/
   /* Open the event queue chosen by the user                               */
```

```
/******************************************************************************/
strncpy(od.ObjectName, qName, (size_t)MQ_Q_NAME_LENGTH);
MQOPEN(hConn, &od, MQOO_INPUT_AS_Q_DEF+MQOO_FAIL_IF_QUIESCING, &eventQueue,
       &compCode, &openReason);
CheckCallResult("Open event queue", compCode, openReason);

/******************************************************************************/
/* Set the GMO options to control the action of the get message from the   */
/* queue.                                                                     */
/******************************************************************************/
gmo.WaitInterval = 30000;               /* 30 second wait for message      */
gmo.Options = MQGMO_WAIT + MQGMO_FAIL_IF_QUIESCING + MQGMO_CONVERT;
gmo.Version = MQGMO_VERSION_2;          /* Avoid need to reset Message ID  */
gmo.MatchOptions = MQMO_NONE;           /* and Correlation ID after every  */
                                 /* mqGetBag
/******************************************************************************/
/* If open fails, we cannot access the queue and must stop the monitor.    */
/******************************************************************************/
if (compCode != MQCC_OK)
   bQueueOK = 0;

/******************************************************************************/
/* Main loop to get an event message when it arrives                       */
/******************************************************************************/
while (bQueueOK)
{
  printf("\nWaiting for an event\n");

  /******************************************************************************/
  /* Get the message from the event queue and convert it into the event    */
  /* bag.                                                                       */
  /******************************************************************************/
  mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

  /******************************************************************************/
  /* If get fails, we cannot access the queue and must stop the monitor.   */
  /******************************************************************************/
  if (compCode != MQCC_OK)
  {
     bQueueOK = 0;

     /******************************************************************************/
     /* If get fails because no message available then we have timed out, */
     /* so report this, otherwise report an error.                        */
     /******************************************************************************/
     if (reason == MQRC_NO_MSG_AVAILABLE)
     {
        printf("No more messages\n");
     }
     else
     {
        CheckCallResult("Get bag", compCode, reason);
     }
  }

  /******************************************************************************/
  /* Event message read - Print the contents of the event bag              */
  /******************************************************************************/
  else
  {
    if ( PrintBag(eventBag) )
        printf("\nError found while printing bag contents\n");

  }  /* end of msg found */
} /* end of main loop */
/******************************************************************************/
/* Close the event queue if successfully opened                            */
/******************************************************************************/
if (openReason == MQRC_NONE)
{
   MQCLOSE(hConn, &eventQueue, MQCO_NONE, &compCode, &reason);
   CheckCallResult("Close event queue", compCode, reason);
}

/******************************************************************************/
/* Delete the event bag if successfully created.                           */
/******************************************************************************/
if (eventBag != MQHB_UNUSABLE_HBAG)
{
   mqDeleteBag(&eventBag, &compCode, &reason);
   CheckCallResult("Delete the event bag", compCode, reason);
}
```

```
} /* end of GetQEvents */

/********************************************************************************/
/*                                                                              */
/* Function: PrintBag                                                           */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Input Parameters:  Bag Handle                                                */
/*                                                                              */
/* Output Parameters: None                                                      */
/*                                                                              */
/* Returns:           Number of errors found                                    */
/*                                                                              */
/* Logic: Calls PrintBagContents to display the contents of the bag.            */
/*                                                                              */
/********************************************************************************

int PrintBag(MQHBAG dataBag)
{
    int errors;

    printf("\n");
    errors = PrintBagContents(dataBag, 0);
    printf("\n");

    return errors;
}

/********************************************************************************/
/*                                                                              */
/* Function: PrintBagContents                                                   */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Input Parameters:  Bag Handle                                                */
/*                    Indentation level of bag                                  */
/*                                                                              */
/* Output Parameters: None                                                      */
/*                                                                              */
/* Returns:           Number of errors found                                    */
/*                                                                              */
/* Logic: Count the number of items in the bag                                  */
/*        Obtain selector and item type for each item in the bag.               */
/*        Obtain the value of the item depending on item type and display the   */
/*        index of the item, the selector and the value.                        */
/*        If the item is an embedded bag handle then call this function again   */
/*        to print the contents of the embedded bag increasing the              */
/*        indentation level.                                                    */
/*                                                                              */
/********************************************************************************/
int PrintBagContents(MQHBAG dataBag, int indent)
{
   /*****************************************************************************/
   /* Definitions                                                               */
   /*****************************************************************************/
   #define LENGTH 500                     /* Max length of string to be read*/
   #define INDENT 4                       /* Number of spaces to indent     */
                                          /* embedded bag display           */


   /*****************************************************************************/
   /* Variables                                                                 */
   /*****************************************************************************/
   MQLONG  itemCount;                     /* Number of items in the bag     */
   MQLONG  itemType;                      /* Type of the item               */
   int     i;                             /* Index of item in the bag       */
   MQCHAR  stringVal[LENGTH+1];           /* Value if item is a string      */
   MQBYTE  byteStringVal[LENGTH];         /* Value if item is a byte string */
   MQLONG  stringLength;                  /* Length of string value         */
   MQLONG  ccsid;                         /* CCSID of string value          */
   MQINT32 iValue;                        /* Value if item is an integer    */
   MQINT64 i64Value;                      /* Value if item is a 64-bit      */
                                          /* integer                        */
   MQLONG  selector;                      /* Selector of item               */
   MQHBAG  bagHandle;                     /* Value if item is a bag handle  */
   MQLONG  reason;                        /* reason code                    */
   MQLONG  compCode;                      /* completion code                */
   MQLONG  trimLength;                    /* Length of string to be trimmed */
   int     errors = 0;                    /* Count of errors found          */
   char    blanks[] = "                     "; /* Blank string used to      */
```

```
                                            /* indent display           */

/*****************************************************************/
/* Count the number of items in the bag                          */
/*****************************************************************/
mqCountItems(dataBag, MQSEL_ALL_SELECTORS, &itemCount, &compCode, &reason);

if (compCode != MQCC_OK)
   errors++;
else
{
   printf("
   printf("
   printf("
}

/*****************************************************************/
/* If no errors found, display each item in the bag              */
/*****************************************************************/
if (!errors)
{
   for (i = 0; i < itemCount; i++)
   {

       /*************************************************************/
       /* First inquire the type of the item for each item in the bag */
       /*************************************************************/
       mqInquireItemInfo(dataBag,             /* Bag handle          */
                         MQSEL_ANY_SELECTOR,  /* Item can have any selector*/
                         i,                   /* Index position in the bag */
                         &selector,           /* Actual value of selector  */
                                              /* returned by call    */
                         &itemType,           /* Actual type of item */
                                              /* returned by call    */
                         &compCode,           /* Completion code     */
                         &reason);            /* Reason Code         */

       if (compCode != MQCC_OK)
          errors++;

       switch(itemType)
       {
       case MQITEM_INTEGER:
            /*********************************************************/
            /* Item is an integer. Find its value and display its index,   */
            /* selector and value.                                   */
            /*********************************************************/
            mqInquireInteger(dataBag,         /* Bag handle          */
                             MQSEL_ANY_SELECTOR, /* Allow any selector     */
                             i,               /* Index position in the bag */
                             &iValue,         /* Returned integer value
                             &compCode,       /* Completion code     */
                             &reason);        /* Reason Code         */

            if (compCode != MQCC_OK)
               errors++;
            else
               printf("%.*s  %-2d     %-4d     (%d)\n",
                       indent, blanks, i, selector, iValue);
            break

       case MQITEM_INTEGER64:
            /*********************************************************/
            /* Item is a 64-bit integer. Find its value and display its    */
            /* index, selector and value.                            */
            /*********************************************************/
            mqInquireInteger64(dataBag,       /* Bag handle          */
                               MQSEL_ANY_SELECTOR, /* Allow any selector   */
                               i,             /* Index position in the bag */
                               &i64Value,     /* Returned integer value    */
                               &compCode,     /* Completion code     */
                               &reason);      /* Reason Code         */

            if (compCode != MQCC_OK)
               errors++;
            else
               printf("%.*s  %-2d     %-4d     (%"Int64"d)\n",
                       indent, blanks, i, selector, i64Value);
            break;


      case MQITEM_STRING:
```

```
      /*****************************************************************/
      /* Item is a string. Obtain the string in a buffer, prepare    */
      /* the string for displaying and display the index, selector,  */
      /* string and Character Set ID.                                  */
      /*****************************************************************/
      mqInquireString(dataBag,          /* Bag handle                  */
                      MQSEL_ANY_SELECTOR, /* Allow any selector      */
                      i,                /* Index position in the bag */
                      LENGTH,           /* Maximum length of buffer  */
                      stringVal,        /* Buffer to receive string  */
                      &stringLength,    /* Actual length of string   */
                      &ccsid,           /* Coded character set id    */
                      &compCode,        /* Completion code           */
                      &reason);         /* Reason Code               */

      /*****************************************************************/
      /* The call can return a warning if the string is too long for */
      /* the output buffer and has been truncated, so only check     */
      /* explicitly for call failure.                                  */
      /*****************************************************************/
      if (compCode == MQCC_FAILED)
          errors++;
      else
      {
         /**************************************************************/
         /* Remove trailing blanks from the string and terminate with*/
         /* a null. First check that the string should not have been */
         /* longer than the maximum buffer size allowed.              */
         /**************************************************************/
         if (stringLength > LENGTH)
            trimLength = LENGTH;
         else
            trimLength = stringLength;
         mqTrim(trimLength, stringVal, stringVal, &compCode, &reason);
         printf("%.*s  %-2d     %-4d     '%s' %d\n",
                 indent, blanks, i, selector, stringVal, ccsid);
      }
      break;

   case MQITEM_BYTE_STRING:
      /*****************************************************************/
      /* Item is a byte string. Obtain the byte string in a buffer,  */
      /* prepare the byte string for displaying and display the      */
      /* index, selector and string.                                   */
      /*****************************************************************/
      mqInquireByteString(dataBag,      /* Bag handle                  */
                          MQSEL_ANY_SELECTOR, /* Allow any selector  */
                          i,            /* Index position in the bag */
                          LENGTH,       /* Maximum length of buffer  */
                          byteStringVal, /* Buffer to receive string */
                          &stringLength, /* Actual length of string  */
                          &compCode,    /* Completion code           */
                          &reason);     /* Reason Code

      /*****************************************************************/
      /* The call can return a warning if the string is too long for */
      /* the output buffer and has been truncated, so only check     */
      /* explicitly for call failure.                                  */
      /*****************************************************************/
      if (compCode == MQCC_FAILED)
          errors++;
      else
      {
         printf("%.*s  %-2d     %-4d     X'",
                indent, blanks, i, selector);

         for (i = 0 ; i < stringLength ; i++)
            printf("

         printf("'\n");
      }
      break;

   case MQITEM_BAG:
      /*****************************************************************/
      /* Item is an embedded bag handle, so call the PrintBagContents*/
      /* function again to display the contents.                       */
      /*****************************************************************/
      mqInquireBag(dataBag,             /* Bag handle                  */
                   MQSEL_ANY_SELECTOR,  /* Allow any selector        */
                   i,                   /* Index position in the bag */
                   &bagHandle,          /* Returned embedded bag hdle*/
```

```
                            &compCode,           /* Completion code           */
                            &reason);            /* Reason Code               */

               if (compCode != MQCC_OK)
                  errors++;
               else
               {
                  printf("%.*s  %-2d     %-4d     (%d)\n", indent, blanks, i,
                          selector, bagHandle);
                  if (selector == MQHA_BAG_HANDLE)
                     printf("
                  else
                     printf("
                  PrintBagContents(bagHandle, indent+INDENT);
               }
               break;

          default:
               printf("
          }
      }
   }
   return errors;
}
```

채널 오브젝트를 조회하기 위한 샘플 C 프로그램(amqsaicl.c)

샘플 C 프로그램 amqsaicl.c는 MQAI를 사용하여 채널 오브젝트를 조회합니다.

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSAICL.C                                                     */
/*                                                                              */
/* Description:  Sample C program to inquire channel objects                    */
/*               using the IBM MQ Administration Interface (MQAI)               */
/*                                                                              */
/* <N_OCO_COPYRIGHT>                                                            */
/* Licensed Materials - Property of IBM                                         */
/*                                                                              */
/* 63H9336                                                                      */
/* (c) Copyright IBM Corp. 2008, 2023. All Rights Reserved.                     */
/*                                                                              */
/* US Government Users Restricted Rights - Use, duplication or                  */
/* disclosure restricted by GSA ADP Schedule Contract with                      */
/* IBM Corp.                                                                    */
/* <NOC_COPYRIGHT>                                                              */
/********************************************************************************/
/*                                                                              */
/* Function:                                                                    */
/*    AMQSAICL is a sample C program that demonstrates how to inquire           */
/*    attributes of the local queue manager using the MQAI interface. In        */
/*    particular, it inquires all channels and their types.                     */
/*                                                                              */
/*     - A PCF command is built from items placed into an MQAI administration   */
/*       bag.                                                                   */
/*       These are:-                                                            */
/*            - The generic channel name "*"                                    */
/*            - The attributes to be inquired. In this sample we just want      */
/*              name and type attributes                                        */
/*                                                                              */
/*     - The mqExecute MQCMD_INQUIRE_CHANNEL call is executed.                  */
/*       The call generates the correct PCF structure.                          */
/*       The default options to the call are used so that the command is sent   */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                     */
/*       The reply from the command server is placed on a temporary dynamic     */
/*       queue.                                                                 */
/*       The reply from the MQCMD_INQUIRE_CHANNEL is read from the              */
/*       temporary queue and formatted into the response bag.                   */
/*                                                                              */
/*     - The completion code from the mqExecute call is checked and if there    */
/*       is a failure from the command server, then the code returned by the    */
/*       command server is retrieved from the system bag that has been          */
/*       embedded in the response bag to the mqExecute call.                    */
/*                                                                              */
/* Note: The command server must be running.                                    */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
```

```
/* AMQSAICL has 2 parameter - the queue manager name (optional)               */
/*                          - output file (optional) default varies           */
/*****************************************************************************/

/*****************************************************************************/
/* Includes                                                                  */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#if (MQAT_DEFAULT == MQAT_OS400)
#include <recio.h>
#endif

#include <cmqc.h>                       /* MQI                               */
#include <cmqcfc.h>                     /* PCF                               */
#include <cmqbc.h>                      /* MQAI                              */
#include <cmqxc.h>                      /* MQCD                              */

/*****************************************************************************/
/* Function prototypes                                                       */
/*****************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/*****************************************************************************/
/* DataTypes                                                                 */
/*****************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
typedef _RFILE OUTFILEHDL;
#else
typedef FILE OUTFILEHDL;
#endif

/*****************************************************************************/
/* Constants                                                                 */
/*****************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "*SDR     ",    /* MQCHT_SENDER    */
  "*SVR     ",    /* MQCHT_SERVER    */
  "*RCVR    ",    /* MQCHT_RECEIVER  */
  "*RQSTR   ",    /* MQCHT_REQUESTER */
  "*ALL     ",    /* MQCHT_ALL       */
  "*CLTCN   ",    /* MQCHT_CLNTCONN  */
  "*SVRCONN ",    /* MQCHT_SVRCONN   */
  "*CLUSRCVR",    /* MQCHT_CLUSRCVR  */
  "*CLUSSDR "     /* MQCHT_CLUSSDR   */
};
#else
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "sdr      ",    /* MQCHT_SENDER    */
  "svr      ",    /* MQCHT_SERVER    */
  "rcvr     ",    /* MQCHT_RECEIVER  */
  "rqstr    ",    /* MQCHT_REQUESTER */
  "all      ",    /* MQCHT_ALL       */
  "cltconn  ",    /* MQCHT_CLNTCONN  */
  "svrcn    ",    /* MQCHT_SVRCONN   */
  "clusrcvr ",    /* MQCHT_CLUSRCVR  */
  "clussdr  "     /* MQCHT_CLUSSDR   */
};
#endif

/*****************************************************************************/
/* Macros                                                                    */
/*****************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
  #define OUTFILE "QTEMP/AMQSAICL(AMQSAICL)"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = _Ropen((fname),"wr, rtncode=Y");
  #define CLOSEOUTFILE(hdl) \
    _Rclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    _Rwrite((hdl),(buf),(buflen));
```

```
#elif (MQAT_DEFAULT == MQAT_UNIX)
  #define OUTFILE "/tmp/amqsaicl.txt"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#else
  #define OUTFILE "amqsaicl.txt"
  #define OPENOUTFILE(fname) \
    fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#endif

#define ChlType2String(t) ChlTypeMap[(t)-1].name

/********************************************************************************/
/* Function: main                                                               */
/********************************************************************************/
int main(int argc, char *argv[])
{
   /*****************************************************************************/
   /* MQAI variables                                                            */
   /*****************************************************************************/
   MQHCONN hConn;                          /* handle to MQ connection         */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name             */
   MQLONG reason;                          /* reason code                     */
   MQLONG connReason;                      /* MQCONN reason code              */
   MQLONG compCode;                        /* completion code                 */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute         */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
   MQHBAG cAttrsBag;                       /* bag containing chl attributes   */
   MQHBAG errorBag;                        /* bag containing cmd server error */
   MQLONG mqExecuteCC;                     /* mqExecute completion code       */
   MQLONG mqExecuteRC;                     /* mqExecute reason code           */
   MQLONG chlNameLength;                   /* Actual length of chl name       */
   MQLONG chlType;                         /* Channel type                    */
   MQLONG i;                               /* loop counter                    */
   MQLONG numberOfBags;                    /* number of bags in response bag  */
   MQCHAR chlName[MQ_OBJECT_NAME_LENGTH+1];/* name of chl extracted from bag  */
   MQCHAR OutputBuffer[100];               /* output data buffer              */
   OUTFILEHDL *outfp = NULL;               /* output file handle              */

   /*****************************************************************************/
   /* Connect to the queue manager                                              */
   /*****************************************************************************/
   if (argc &gt; 1)
      strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(qmName, &hConn;, &compCode;, &connReason;);

   /*****************************************************************************/
   /* Report the reason and stop if the connection failed.                      */
   /*****************************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("Queue Manager connection", compCode, connReason);
      exit( (int)connReason);
   }

   /*****************************************************************************/
   /* Open the output file                                                      */
   /*****************************************************************************/
   if (argc &gt; 2)
   {
     OPENOUTFILE(outfp, argv[2]);
   }
   else
   {
     OPENOUTFILE(outfp, OUTFILE);
   }

   if(outfp == NULL)
   {
     printf("Could not open output file.\n");
     goto MOD_EXIT;
```

```
}
/******************************************************************************/
/* Create an admin bag for the mqExecute call                                 */
/******************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag;, &compCode;, &reason;);
CheckCallResult("Create admin bag", compCode, reason);

/******************************************************************************/
/* Create a response bag for the mqExecute call                               */
/******************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag;, &compCode;, &reason;);
CheckCallResult("Create response bag", compCode, reason);

/******************************************************************************/
/* Put the generic channel name into the admin bag                            */
/******************************************************************************/
mqAddString(adminBag, MQCACH_CHANNEL_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode;, &reason;);
CheckCallResult("Add channel name", compCode, reason);

/******************************************************************************/
/* Put the channel type into the admin bag                                    */
/******************************************************************************/
mqAddInteger(adminBag, MQIACH_CHANNEL_TYPE, MQCHT_ALL, &compCode;, &reason;);
CheckCallResult("Add channel type", compCode, reason);

/******************************************************************************/
/* Add an inquiry for various attributes                                      */
/******************************************************************************/
mqAddInquiry(adminBag, MQIACH_CHANNEL_TYPE, &compCode;, &reason;);
CheckCallResult("Add inquiry", compCode, reason);

/******************************************************************************/
/* Send the command to find all the channel names and channel types.          */
/* The mqExecute call creates the PCF structure required, sends it to         */
/* the command server, and receives the reply from the command server into */
/* the response bag. The attributes are contained in system bags that are  */
/* embedded in the response bag, one set of attributes per bag.               */
/******************************************************************************/
mqExecute(hConn,                    /* MQ connection handle                   */
          MQCMD_INQUIRE_CHANNEL,    /* Command to be executed                 */
          MQHB_NONE,                /* No options bag                         */
          adminBag,                 /* Handle to bag containing commands      */
          responseBag,              /* Handle to bag to receive the response*/
          MQHO_NONE,                /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,                /* Create a dynamic q for the response  */
          &compCode;,                /* Completion code from the mqexecute   */
          &reason;);                 /* Reason code from mqexecute call      */

/******************************************************************************/
/* Check the command server is started. If not exit.                          */
/******************************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName="">\n");
   goto MOD_EXIT;
}

/******************************************************************************/
/* Check the result from mqExecute call. If successful find the channel    */
/* types for all the channels. If failed find the error.                      */
/******************************************************************************/
if ( compCode == MQCC_OK )                      /* Successful mqExecute    */
{
  /****************************************************************************/
  /* Count the number of system bags embedded in the response bag from the */
  /* mqExecute call. The attributes for each channel are in separate bags. */
  /****************************************************************************/
  mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags;,
               &compCode;, &reason;);
  CheckCallResult("Count number of bag handles", compCode, reason);

  for ( i=0; i<numberOfbags; i++)
  {
    /**************************************************************************/
    /* Get the next system bag handle out of the mqExecute response bag.   */
    /* This bag contains the channel attributes                               */
    /**************************************************************************/
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &cAttrsbag,
                 &compCode, &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);
```

```
      /*****************************************************************/
      /* Get the channel name out of the channel attributes bag        */
      /*****************************************************************/
      mqInquireString(cAttrsBag, MQCACH_CHANNEL_NAME, 0, MQ_OBJECT_NAME_LENGTH,
                      chlName, &chlNameLength, NULL, &compCode, &reason);
      CheckCallResult("Get channel name", compCode, reason);

      /*****************************************************************/
      /* Get the channel type out of the channel attributes bag        */
      /*****************************************************************/

    mqInquireInteger(cAttrsBag, MQIACH_CHANNEL_TYPE, MQIND_NONE, &chlType,
                       &compCode, &reason);
      CheckCallResult("Get type", compCode, reason);

      /*****************************************************************/
      /* Use mqTrim to prepare the channel name for printing.          */
      /* Print the result.                                             */
      /*****************************************************************/
      mqTrim(MQ_CHANNEL_NAME_LENGTH, chlName, chlName, &compCode, &reason);
      sprintf(OutputBuffer, "%-20s%-9s", chlName, ChlType2String(chlType));
      WRITEOUTFILE(outfp,OutputBuffer,29)
    }
  }

  else                                              /* Failed mqExecute     */
  {
    printf("Call to get channel attributes failed: Cc = %ld : Rc = %ld\n",
              compCode, reason);
    /*****************************************************************/
    /* If the command fails get the system bag handle out of the mqexecute   */
    /* response bag.This bag contains the reason from the command server     */
    /* why the command failed.                                       */
    /*****************************************************************/
    if (reason == MQRCCF_COMMAND_FAILED)
    {
      mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag,
                   &compCode, &reason);
      CheckCallResult("Get the result bag handle", compCode, reason);

      /*****************************************************************/
      /* Get the completion code and reason code, returned by the command    */
      /* server, from the embedded error bag.                          */
      /*****************************************************************/
      mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                       &compCode, &reason );
      CheckCallResult("Get the completion code from the result bag",
                      compCode, reason);
      mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                        &compCode, &reason);
      CheckCallResult("Get the reason code from the result bag",
                      compCode, reason);
      printf("Error returned by the command server: Cc = %ld : Rc = %ld\n",
               mqExecuteCC, mqExecuteRC);
    }
  }

MOD_EXIT:
  /*******************************************************************/
  /* Delete the admin bag if successfully created.                   */
  /*******************************************************************/
  if (adminBag != MQHB_UNUSABLE_HBAG)
  {
     mqDeleteBag(&adminBag, &compCode, &reason);
     CheckCallResult("Delete the admin bag", compCode, reason);
  }

  /*******************************************************************/
  /* Delete the response bag if successfully created.                */
  /*******************************************************************/
  if (responseBag != MQHB_UNUSABLE_HBAG)
  {
     mqDeleteBag(&responseBag, &compCode, &reason);
     CheckCallResult("Delete the response bag", compCode, reason);
  }

  /*******************************************************************/
  /* Disconnect from the queue manager if not already connected      */
  /*******************************************************************/
  if (connReason != MQRC_ALREADY_CONNECTED)
  {
     MQDISC(&hConn, &compCode, &reason);
```

```
      CheckCallResult("Disconnect from Queue Manager", compCode, reason);
   }

   /******************************************************************************/
   /* Close the output file if open                                              */
   /******************************************************************************/
   if(outfp != NULL)
      CLOSEOUTFILE(outfp);

   return 0;
}


/******************************************************************************/
/*                                                                            */
/* Function: CheckCallResult                                                  */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* Input Parameters:  Description of call                                     */
/*                    Completion code                                         */
/*                    Reason code                                             */
/*                                                                            */
/* Output Parameters: None                                                    */
/*                                                                            */
/* Logic: Display the description of the call, the completion code and the    */
/*        reason code if the completion code is not successful                */
/*                                                                            */
/******************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %ld : Reason = %ld\n", callText,
                cc, rc);
}
```

큐 및 인쇄 정보를 조회하기 위한 샘플 C 프로그램(amqsailq.c)

샘플 C 프로그램 `amqsailq.c`는 MQAI를 사용하여 로컬 큐의 현재 용량을 조회합니다.

```
/******************************************************************************/
/*                                                                            */
/* Program name: AMQSAILQ.C                                                   */
/*                                                                            */
/* Description:  Sample C program to inquire the current depth of the local   */
/*               queues using the IBM MQ Administration Interface (MQAI)      */
/*                                                                            */
/* Statement:    Licensed Materials - Property of IBM                         */
/*                                                                            */
/*               84H2000, 5765-B73                                            */
/*               84H2001, 5639-B42                                            */
/*               84H2002, 5765-B74                                            */
/*               84H2003, 5765-B75                                            */
/*               84H2004, 5639-B43                                            */
/*                                                                            */
/*               (C) Copyright IBM Corp. 1999, 2023                           */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* Function:                                                                  */
/*    AMQSAILQ is a sample C program that demonstrates how to inquire         */
/*    attributes of the local queue manager using the MQAI interface. In      */
/*    particular, it inquires the current depths of all the local queues.     */
/*                                                                            */
/*     - A PCF command is built by placing items into an MQAI administration  */
/*       bag.                                                                 */
/*       These are:-                                                          */
/*            - The generic queue name "*"                                    */
/*            - The type of queue required. In this sample we want to         */
/*              inquire local queues.                                         */
/*            - The attribute to be inquired. In this sample we want the      */
/*              current depths.                                               */
/*                                                                            */
/*     - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.     */
/*       The call generates the correct PCF structure.                        */
/*       The default options to the call are used so that the command is sent */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                   */
/*       The reply from the command server is placed on a temporary dynamic   */
```

```
/*       queue.                                                             */
/*       The reply from the MQCMD_INQUIRE_Q command is read from the        */
/*       temporary queue and formatted into the response bag.               */
/*                                                                          */
/*     - The completion code from the mqExecute call is checked and if there */
/*       is a failure from the command server, then the code returned by    */
/*       command server is retrieved from the system bag that has been      */
/*       embedded in the response bag to the mqExecute call.                */
/*                                                                          */
/*     - If the call is successful, the depth of each local queue is placed */
/*       in system bags embedded in the response bag of the mqExecute call. */
/*       The name and depth of each queue is obtained from each of the bags */
/*       and the result displayed on the screen.                            */
/*                                                                          */
/* Note: The command server must be running.                                */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/* AMQSAILQ has 1 parameter - the queue manager name (optional)             */
/*                                                                          */
/****************************************************************************/

/****************************************************************************/
/* Includes                                                                 */
/****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                              */
#include <cmqcfc.h>                     /* PCF                              */
#include <cmqbc.h>                      /* MQAI                             */

/****************************************************************************/
/* Function prototypes                                                      */
/****************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/****************************************************************************/
/* Function: main                                                           */
/****************************************************************************/
int main(int argc, char *argv[])
{
   /*************************************************************************/
   /* MQAI variables                                                        */
   /*************************************************************************/
   MQHCONN hConn;                       /* handle to IBM MQ connection     */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name          */
   MQLONG reason;                       /* reason code                     */
   MQLONG connReason;                   /* MQCONN reason code              */
   MQLONG compCode;                     /* completion code                 */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute      */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute    */
   MQHBAG qAttrsBag;                    /* bag containing q attributes     */
   MQHBAG errorBag;                     /* bag containing cmd server error */
   MQLONG mqExecuteCC;                  /* mqExecute completion code       */
   MQLONG mqExecuteRC;                  /* mqExecute reason code           */
   MQLONG qNameLength;                  /* Actual length of q name         */
   MQLONG qDepth;                       /* depth of queue                  */
   MQLONG i;                            /* loop counter                    */
   MQLONG numberOfBags;                 /* number of bags in response bag  */
   MQCHAR qName[MQ_Q_NAME_LENGTH+1];    /* name of queue extracted from bag*/


   printf("Display current depths of local queues\n\n");

   /*************************************************************************/
   /* Connect to the queue manager                                          */
   /*************************************************************************/
   if (argc > 1)
      strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(qmName, &hConn, &compCode, &connReason);

   /*************************************************************************/
   /* Report the reason and stop if the connection failed.                  */
   /*************************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("Queue Manager connection", compCode, connReason);
      exit( (int)connReason);
   }
```

```
/*****************************************************************************/
/* Create an admin bag for the mqExecute call                                */
/*****************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag, &compCode, &reason);
CheckCallResult("Create admin bag", compCode, reason);
/*****************************************************************************/
/* Create a response bag for the mqExecute call                              */
/*****************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
CheckCallResult("Create response bag", compCode, reason);

/*****************************************************************************/
/* Put the generic queue name into the admin bag                             */
/*****************************************************************************/
mqAddString(adminBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode, &reason);
CheckCallResult("Add q name", compCode, reason);

/*****************************************************************************/
/* Put the local queue type into the admin bag                               */
/*****************************************************************************/
mqAddInteger(adminBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type", compCode, reason);

/*****************************************************************************/
/* Add an inquiry for current queue depths                                   */
/*****************************************************************************/
mqAddInquiry(adminBag, MQIA_CURRENT_Q_DEPTH, &compCode, &reason);
CheckCallResult("Add inquiry", compCode, reason);

/*****************************************************************************/
/* Send the command to find all the local queue names and queue depths.      */
/* The mqExecute call creates the PCF structure required, sends it to        */
/* the command server, and receives the reply from the command server into   */
/* the response bag. The attributes are contained in system bags that are    */
/* embedded in the response bag, one set of attributes per bag.              */
/*****************************************************************************/
mqExecute(hConn,                    /* IBM MQ connection handle              */
          MQCMD_INQUIRE_Q,          /* Command to be executed                */
          MQHB_NONE,                /* No options bag                        */
          adminBag,                 /* Handle to bag containing commands     */
          responseBag,              /* Handle to bag to receive the response*/
          MQHO_NONE,                /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,                /* Create a dynamic q for the response   */
          &compCode,                /* Completion code from the mqExecute    */
          &reason);                 /* Reason code from mqExecute call       */


/*****************************************************************************/
/* Check the command server is started. If not exit.                         */
/*****************************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName>\n");
   MQDISC(&hConn, &compCode, &reason);
   CheckCallResult("Disconnect from Queue Manager", compCode, reason);
   exit(98);
}

/*****************************************************************************/
/* Check the result from mqExecute call. If successful find the current     */
/* depths of all the local queues. If failed find the error.                 */
/*****************************************************************************/
if ( compCode == MQCC_OK )                      /* Successful mqExecute     */
{
  /***************************************************************************/
  /* Count the number of system bags embedded in the response bag from the */
  /* mqExecute call. The attributes for each queue are in a separate bag.  */
  /***************************************************************************/
  mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags, &compCode,
               &reason);
  CheckCallResult("Count number of bag handles", compCode, reason);

  for ( i=0; i<numberOfBags; i++)
  {
    /*************************************************************************/
    /* Get the next system bag handle out of the mqExecute response bag.   */
    /* This bag contains the queue attributes                                */
    /*************************************************************************/
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &qAttrsBag, &compCode,
                 &reason);
```

```
      CheckCallResult("Get the result bag handle", compCode, reason);

      /*****************************************************************/
      /* Get the queue name out of the queue attributes bag            */
      /*****************************************************************/
      mqInquireString(qAttrsBag, MQCA_Q_NAME, 0, MQ_Q_NAME_LENGTH, qName,
                      &qNameLength, NULL, &compCode, &reason);
      CheckCallResult("Get queue name", compCode, reason);

      /*****************************************************************/
      /* Get the depth out of the queue attributes bag                 */
      /*****************************************************************/
      mqInquireInteger(qAttrsBag, MQIA_CURRENT_Q_DEPTH, MQIND_NONE, &qDepth,
                       &compCode, &reason);
      CheckCallResult("Get depth", compCode, reason);

      /*****************************************************************/
      /* Use mqTrim to prepare the queue name for printing.            */
      /* Print the result.                                             */
      /*****************************************************************/
      mqTrim(MQ_Q_NAME_LENGTH, qName, qName, &compCode, &reason)
      printf("%4d  %-48s\n", qDepth, qName);
    }
  }

  else                                          /* Failed mqExecute     */
  {
    printf("Call to get queue attributes failed: Completion Code = %d :
            Reason = %d\n", compCode, reason);

    /*****************************************************************/
    /* If the command fails get the system bag handle out of the mqExecute   */
    /* response bag. This bag contains the reason from the command server    */
    /* why the command failed.                                       */
    /*****************************************************************/
    if (reason == MQRCCF_COMMAND_FAILED)
    {
      mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
                   &reason);
      CheckCallResult("Get the result bag handle", compCode, reason);

     /*****************************************************************/
     /* Get the completion code and reason code, returned by the command     */
     /* server, from the embedded error bag.                          */
     /*****************************************************************/
     mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                      &compCode, &reason );
     CheckCallResult("Get the completion code from the result bag",
                      compCode, reason);
     mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                       &compCode, &reason);
     CheckCallResult("Get the reason code from the result bag",
                      compCode, reason);
     printf("Error returned by the command server: Completion Code = %d :
             Reason = %d\n", mqExecuteCC, mqExecuteRC);
   }
}

/*****************************************************************/
/* Delete the admin bag if successfully created.                 */
/*****************************************************************/
if (adminBag != MQHB_UNUSABLE_HBAG)
{
   mqDeleteBag(&adminBag, &compCode, &reason);
   CheckCallResult("Delete the admin bag", compCode, reason);
}

/*****************************************************************/
/* Delete the response bag if successfully created.              */
/*****************************************************************/
if (responseBag != MQHB_UNUSABLE_HBAG)
{
   mqDeleteBag(&responseBag, &compCode, &reason);
   CheckCallResult("Delete the response bag", compCode, reason);
}

/*****************************************************************/
/* Disconnect from the queue manager if not already connected    */
/*****************************************************************/
if (connReason != MQRC_ALREADY_CONNECTED)
{
   MQDISC(&hConn, &compCode, &reason);
```

```
      CheckCallResult("Disconnect from queue manager", compCode, reason);
  }
  return 0;
}

/******************************************************************************/
*                                                                             */
* Function: CheckCallResult                                                   */
*                                                                             */
/******************************************************************************/
*                                                                             */
* Input Parameters:  Description of call                                      */
*                    Completion code                                          */
*                    Reason code                                              */
*                                                                             */
* Output Parameters: None                                                     */
*                                                                             */
* Logic: Display the description of the call, the completion code and the     */
*        reason code if the completion code is not successful                 */
*                                                                             */
/******************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
  if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
                callText, cc, rc);
}
```

IBM MQ 구성을 위한 힌트 및 팁

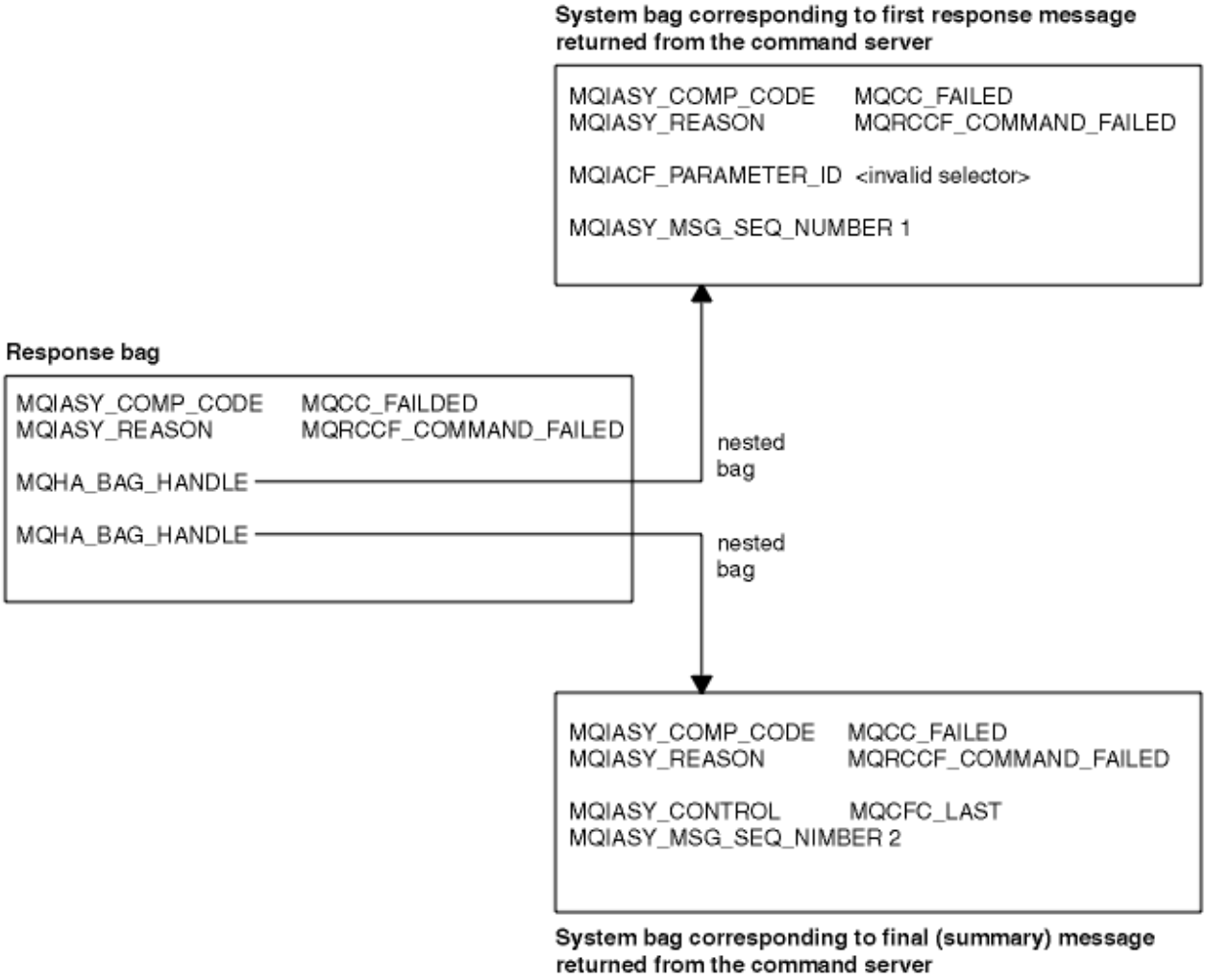
MQAI를 사용할 때 프로그래밍 힌트 및 팁입니다.

MQAI는 명령 서버 자체에서 직접 처리하지 않고 PCF 메시지를 사용하여 관리 명령을 명령 서버로 송신합니다. MQAI를 사용하여 IBM MQ을 구성하기 위한 일부 팁이 여기 있습니다.

- IBM MQ의 문자열은 고정된 길이로 채워진 공백입니다. C를 사용할 때, 널(Null) 종료 문자열은 일반적으로 IBM MQ 프로그래밍 인터페이스에 입력 매개변수로서 공급될 수 있습니다.
- 문자열 속성의 값을 지우려면, 비어 있는 문자열이 아닌 하나의 공백으로 설정하십시오.
- 해당 속성에서 변경하고 조회하려는 속성을 미리 고려하십시오.
- 특정 속성은 변경될 수 없습니다(예: 큐 이름 또는 채널 유형). 수정될 수 있는 해당 속성만 변경하려고 시도하는지 확인하십시오. 특정 PCF 변경 오브젝트를 위한 필수 및 선택적 매개변수 목록을 참조하십시오. PCF(Programmable Command Format)의 정의를 참조하십시오.
- MQAI 호출에 실패하면 실패에 대한 일부 세부사항은 응답 백으로 리턴됩니다. 추가적인 세부사항은 선택자 MQHA_BAG_HANDLE이 액세스할 수 있는 중첩 백에서 찾을 수 있습니다. 예를 들어, mqExecute 호출이 MQRCCF_COMMAND_FAILED의 이유 코드와 함께 실패하는 경우, 이 정보는 응답 백에서 리턴됩니다. 이 이유 코드에 대한 가능한 이유는 지정된 선택자가 명령 메시지의 유형에 효과적이지 않았다는 것이고 정보에 대한 이 세부사항이 백 핸들에 의해 액세스될 수 있는 중첩 백에서 발견됩니다.

  MQExecute에 대한 자세한 정보는 55 페이지의 『mqExecute 호출을 사용하여 명령 서버에 관리 명령 보내기』의 내용을 참조하십시오.

  다음 다이어그램에 이 시나리오가 표시됩니다.



고급 토픽

색인 작성, 데이터 변환 및 메시지 디스크립터의 사용에 대한 정보입니다.

- 색인 작성

  삽입 순서를 보존하기 위해 백으로부터 기존 데이터 항목을 대체하거나 제거할 때 색인이 사용됩니다. 색인화에 대한 전체 세부사항은 43 페이지의 『MQAI의 색인 작성』에 있습니다.

- 데이터 변환

  MQAI 데이터 백에 포함된 문자열은 다양한 부호화 문자 집합에 있을 수 있고 이것들이 mqSetInteger 호출을 사용하여 변환할 수 있습니다. 데이터 변환에 대한 전체 세부사항은 44 페이지의 『MQAI에서 데이터 변환』에 있습니다.

- 메시지 디스크립터 사용

  MQAI는 데이터 백이 작성될 때 초기값으로 설정되는 메시지 디스크립터를 작성합니다. 메시지 디스크립터 사용에 대한 전체 세부사항은 45 페이지의 『MQAI에서 메시지 디스크립터 사용』에 있습니다.

*MQAI*의 색인 작성

색인은 백에서 기존 데이터 항목을 바꾸거나 제거할 때 사용됩니다. 세 가지 색인 작성 유형이 있으며, 색인 작성을 사용하면 데이터 항목을 쉽게 검색할 수 있습니다.

백의 데이터 항목 내에 있는 각 선택자 및 값에 세 개의 연관된 색인 번호가 있습니다.

- 동일한 선택자가 있는 다른 항목에 대한 색인
- 항목이 속하는 선택자의 카테고리(사용자 또는 시스템)에 대한 색인

- 백에 있는 모든 데이터 항목에 대한 색인(사용자 및 시스템)

이를 사용하면 44 페이지의 그림 1에 표시된 대로 사용자 선택자, 시스템 선택자 또는 둘 다에 의한 색인 작성이 가능합니다.

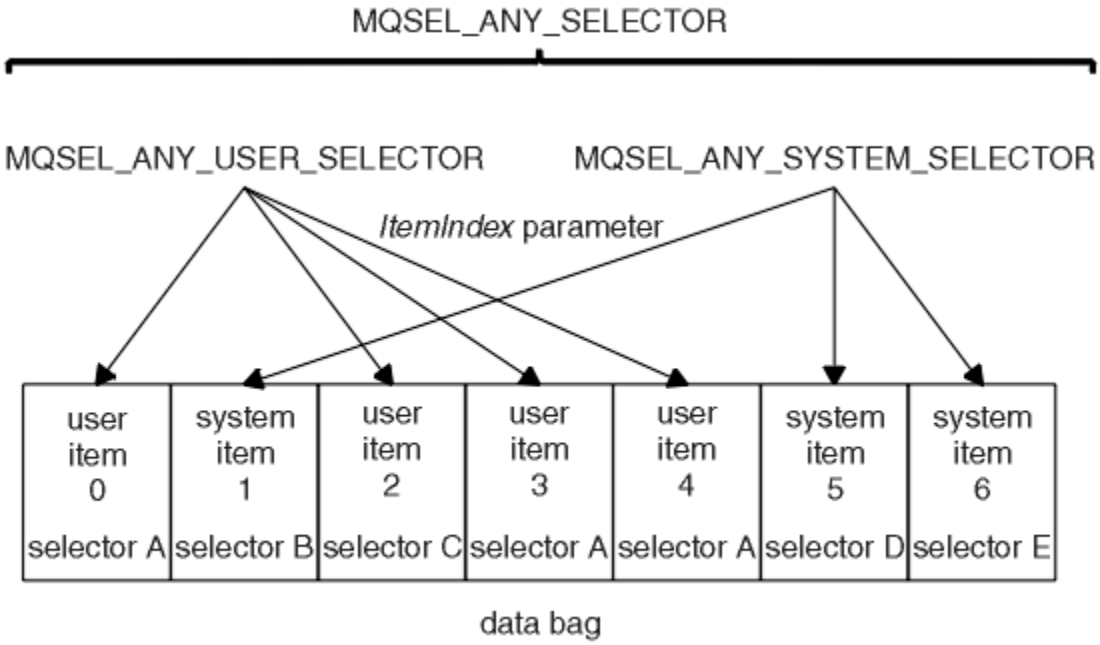


그림 *1.* 색인 작성

그림 44 페이지의 그림 1에서, 다음 색인 쌍으로 사용자 항목 3(선택자 A)을 참조할 수 있습니다.

Selector	***ItemIndex***
선택자 A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	3

색인은 C의 배열과 같이 0을 기반으로 합니다. 'n'개가 발생하는 경우 색인의 범위는 0에서 'n-1'까지이며 갭은 없습니다.

색인은 백에서 기존 데이터 항목을 바꾸거나 제거할 때 사용됩니다. 이러한 방식으로 사용될 때 삽입 순서는 그대로 유지되지만 다른 데이터 항목의 색인에 영향을 줄 수 있습니다. 이 예에 대해서는 백 내에서 정보 변경 및 데이터 항목 삭제를 참조하십시오.

세 유형의 색인 작성을 사용하면 데이터 항목을 쉽게 검색할 수 있습니다. 예를 들어, 백에 특정 선택자에 대한 세 인스턴스가 있는 경우, mqCountItems 호출로 해당 선택자의 인스턴스 수를 셀 수 있고 mqInquire* 호출로 선택자와 색인을 모두 지정하여 해당 값만 조회할 수 있습니다. 이는 값 목록이 있는 속성에서 유용합니다(예: 채널에 있는 엑시트 중 일부).

*MQAI*에서 데이터 변환

MQAI 데이터 백에 포함된 문자열은 다양한 코드화 문자 세트에 있을 수 있습니다. mqSetInteger 호출을 사용하여 이러한 문자열을 변환할 수 있습니다.

PCF 메시지와 같이, MQAI 데이터 백에 포함된 문자열은 다양한 코드화 문자 세트에 있습니다. 대개 PCF 메시지의 모든 문자열은 동일한 코드화 문자 세트, 즉 큐 관리자와 동일한 세트에 있습니다.

데이터 백의 각 문자열 항목에는 두 개의 값(문자열과 CCSID)이 있습니다. 백에 추가되는 문자열은 mqAddString 또는 mqSetString 호출의 *Buffer* 매개변수에서 가져옵니다. CCSID는 MQIASY_CODED_CHAR_SET_ID의 선택자를 포함한 시스템 항목에서 얻을 수 있습니다. 이는 백 *CCSID*로 알려져 있으며 mqSetInteger 호출을 사용하여 변경할 수 있습니다.

데이터 백에 포함된 문자열 값을 조회하는 경우, CCSID는 호출의 출력 매개변수입니다.

45 페이지의 표 2에서는 데이터 백을 메시지로 변환하거나 그 반대인 경우에 적용되는 규칙을 보여줍니다.

표 *2. CCSID* 처리

MQAI 호출	**CCSID**	**호출할 입력**	**호출할 출력**
mqBagToBuffer	Bag CCSID(1)	무시함	변경하지 않음
mqBagToBuffer	백에 있는 문자열 CCSID	사용함	변경하지 않음
mqBagToBuffer	버퍼에 있는 문자열 CCSID	적용할 수 없음	백에 있는 문자열 CCSID에서 복사함
mqBufferToBag	Bag CCSID(1)	무시함	변경하지 않음
mqBufferToBag	버퍼에 있는 문자열 CCSID	사용함	변경하지 않음
mqBufferToBag	백에 있는 문자열 CCSID	적용할 수 없음	버퍼에 있는 문자열 CCSID에서 복사함
mqPutBag	MQMD CCSID	사용함	변경하지 않음(2)
mqPutBag	Bag CCSID(1)	무시함	변경하지 않음
mqPutBag	백에 있는 문자열 CCSID	사용함	변경하지 않음
mqPutBag	송신한 메시지에 있는 문자열 CCSID	적용할 수 없음	백에 있는 문자열 CCSID에서 복사함
mqGetBag	MQMD CCSID	메시지의 데이터 변환에 사용함	리턴된 데이터의 CCSID로 설정됨(3)
mqGetBag	Bag CCSID(1)	무시함	변경하지 않음
mqGetBag	메시지에 있는 문자열 CCSID	사용함	변경하지 않음
mqGetBag	백에 있는 문자열 CCSID	적용할 수 없음	메시지에 있는 문자열 CCSID에서 복사함
mqExecute	요청-백 CCSID	요청 메시지의 MQMD에 사용됨(4)	변경하지 않음
mqExecute	응답-백 CCSID	응답 메시지의 데이터 변환에 사용함(4)	리턴된 데이터의 CCSID로 설정됨(3)
mqExecute	요청 백에 있는 문자열 CCSID	요청 메시지에 사용함	변경하지 않음
mqExecute	응답 백에 있는 문자열 CCSID	적용할 수 없음	응답 메시지에 있는 문자열 CCSID에서 복사함

참고사항:

1. 백 CCSID는 선택자 MQIASY_CODED_CHAR_SET_ID가 있는 시스템 항목입니다.
2. MQCCSI_Q_MGR은 실제 큐 관리자 CCSID로 변경됩니다.
3. 데이터 변환을 요청하는 경우, 리턴된 데이터의 CCSID가 출력 값과 동일합니다. 데이터 변환을 요청하지 않는 경우, 리턴된 데이터의 CCSID는 메시지 값과 동일합니다. 데이터 변환을 요청했지만 실패하는 경우 메시지가 리턴되지 않습니다.
4. CCSID가 MQCCSI_DEFAULT인 경우 큐 관리자의 CCSID가 사용됩니다.

*MQAI*에서 메시지 디스크립터 사용

MQAI가 생성하는 메시지 디스크립터는 데이터 백이 작성될 때 초기값으로 설정됩니다.

PCF 명령 유형은 선택자가 MQIASY_TYPE인 시스템 항목에서 얻을 수 있습니다. 데이터 백을 작성하는 경우, 이 항목의 초기값은 작성한 백의 유형에 따라 설정됩니다.

표 *3. PCF* 명령 유형

백 유형	**MQIASY_TYPE 항목의 초기값**
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_*	MQCFT_USER

MQAI가 메시지 디스크립터를 생성할 때 *Format* 및 *MsgType* 매개변수에 사용된 값은 46 페이지의 표 3에 표시된 대로 선택자가 MQIASY_TYPE인 시스템 항목 값에 따라 다릅니다.

표 *4. MQMD*의 형식 및 *MsgType* 매개변수

PCF 명령 유형	**형식**	**MsgType**
MQCFT_COMMAND	MQFMT_ADMIN	MQMT_REQUEST
MQCFT_REPORT	MQFMT_ADMIN	MQMT_REPORT
MQCFT_RESPONSE	MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	MQFMT_ADMIN	MQMT_DATAGRAM
MQCFT_EVENT	MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_*	MQFMT_PCF	MQMT_DATAGRAM

46 페이지의 표 4에서는 관리 백 또는 명령 백을 작성하는 경우, 메시지 디스크립터의 *Format*이 MQFMT_ADMIN이고 *MsgType*이 MQMT_REQUEST임을 보여줍니다. 이는 응답이 되돌아올 것으로 예상되는 경우 명령 서버에 송신된 PCF 요청 메시지에 적당합니다.

메시지 디스크립터의 다른 매개변수는 46 페이지의 표 5에 표시된 값을 사용합니다.

표 *5.* 메시지 디스크립터 값

매개변수	**가치**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	46 페이지의 표 4 참조
Expiry	30초(47 페이지의 『1』 참고)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	CCSID 백에 따라 다름(47 페이지의 『2』 참고)
Format	46 페이지의 표 4 참조
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0

표 5. 메시지 디스크립터 값 (계속)	
매개변수	**가치**
ReplyToQ	47 페이지의 『3』 참조
ReplyToQMgr	공백

참고사항:

1. 이 값은 mqExecute 호출에서 *OptionsBag* 매개변수를 사용하여 대체할 수 있습니다. 이에 대한 정보는 mqExecute의 내용을 참조하십시오.
2. 44 페이지의 『MQAI에서 데이터 변환』 의 내용을 참조하십시오.
3. MQMT_REQUEST 유형의 메시지에 대한 MQAI 생성 임시 동적 큐 또는 사용자 지정 응답 큐의 이름입니다. 그렇지 않으면 공백입니다.

데이터 백

데이터 백은 MQAI를 사용하는 오브젝트의 매개변수 또는 특성 처리 수단입니다.

데이터 백

- 데이터 백에는 0개 이상의 데이터 항목이 포함됩니다. 이러한 데이터 항목은 백에 있을 때 백 내에 순서화됩니다. 이는 삽입 순서라고 합니다. 각 데이터 항목에는 정수, 64비트 정수, 정수 필터, 문자열, 문자열 필터, 바이트 문자열, 바이트 문자열 필터 또는 다른 백의 처리가 될 수 있는 해당 데이터 항목의 값 및 데이터 항목을 식별하는 선택자가 포함됩니다. 데이터 항목은 49 페이지의 『데이터 항목』 에서 상세하게 설명됩니다

  두 가지 유형의 선택자(사용자 선택자 및 시스템 선택자)가 있습니다. 이는 MQAI 선택자에서 설명됩니다. 선택자는 일반적으로 고유하지만 동일한 선택자에 대해 여러 값을 가질 수 있습니다. 이 경우, 색인은 필수적인 선택자의 특정 발생을 식별합니다. 색인은 43 페이지의 『MQAI의 색인 작성』 에 설명되어 있습니다.

  이러한 개념의 계층이 그림 1에 표시되어 있습니다.

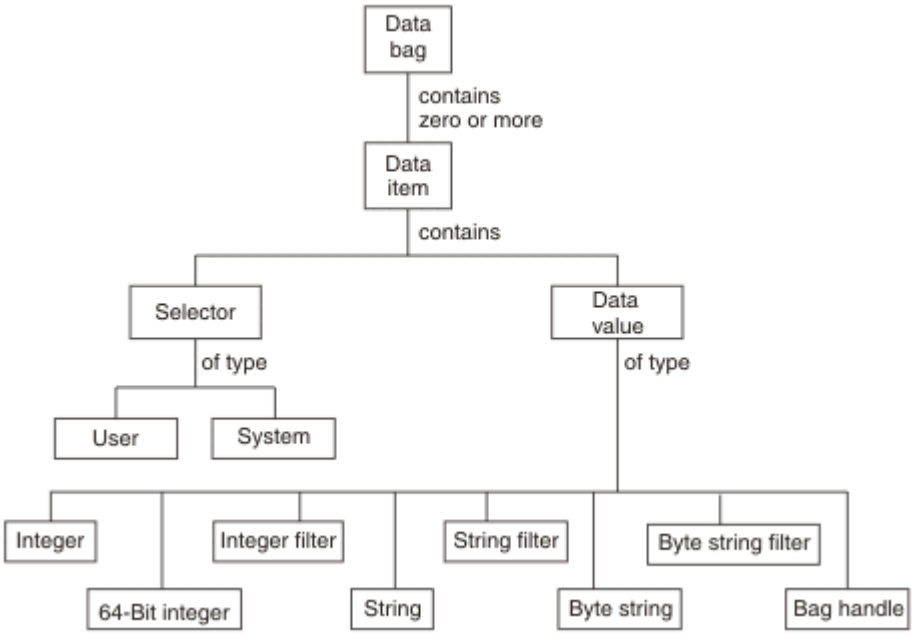


그림 *2. MQAI* 개념의 계층

계층은 이전 단락에서 설명되었습니다.

데이터 백의 유형

수행하려는 태스크에 따라 작성하려는 데이터 백의 유형을 선택할 수 있습니다.

사용자 백(user bag)
사용자 데이터에 사용되는 단순 백.

관리 백(administration bag)
관리 메시지를 명령 서버에 보냄으로써 IBM MQ 오브젝트를 관리하기 위해 사용되는 데이터를 위해 작성되는 백. 관리 백은 48 페이지의 『데이터 백 작성 및 삭제』에 설명된 대로 특정 옵션을 자동으로 구현합니다.

명령 백(command bag)
IBM MQ 오브젝트 관리를 위한 명령에도 작성된 백. 그러나, 관리 백과 달리, 명령 백은 이러한 옵션이 사용 가능할지라도 자동으로 특정 옵션을 의미하지는 않습니다. 옵션에 대한 자세한 정보는 48 페이지의 『데이터 백 작성 및 삭제』의 내용을 참조하십시오.

그룹 백
그룹화된 데이터 항목 세트에 사용되는 백. 그룹 백은 IBM MQ 오브젝트를 관리하기 위해 사용될 수 없습니다.

또한, **시스템 백**이 응답 메시지가 명령 서버에서 리턴되고 사용자의 출력 백에 배치될 때 MQAI에서 작성됩니다. 시스템 백은 사용자에 의해 수정될 수 없습니다.

데이터 백 사용: 데이터 백을 사용하는 여러가지 방법이 이 토픽에 나열됩니다.

데이터 백 사용

데이터 백을 사용하는 여러가지 방법이 다음 목록에 표시됩니다.

- 데이터 백을 작성하고 삭제할 수 있습니다. 48 페이지의 『데이터 백 작성 및 삭제』
- 데이터 백을 사용하여 애플리케이션 사이에 데이터를 송신할 수 있습니다. 49 페이지의 『데이터 백 넣기 및 수신』
- 데이터 백에 데이터 항목을 추가할 수 있습니다. 50 페이지의 『백에 데이터 항목 추가』
- 데이터 백 내에서 조회 명령을 추가할 수 있습니다. 51 페이지의 『백에 조회 명령 추가』
- 데이터 백 내에서 조회할 수 있습니다. 51 페이지의 『데이터 백 내에서 조회』
- 데이터 백 내에서 데이터 항목을 셀 수 있습니다. 54 페이지의 『데이터 항목 계산』
- 데이터 백 내에서 정보를 변경할 수 있습니다. 52 페이지의 『백 내의 정보 변경』
- 데이터 백을 지울 수 있습니다. 53 페이지의 『mqClearBag 호출을 사용하여 백 지우기』
- 데이터 백을 자를 수 있습니다. 53 페이지의 『mqTruncateBag 호출을 사용하여 백 자르기』
- 백 및 버퍼를 변환할 수 있습니다. 53 페이지의 『백 및 버퍼 변환』

데이터 백 작성 및 삭제

데이터 백 작성

MQAI를 사용하려면 먼저 mqCreateBag 호출을 사용하여 데이터 백을 작성하십시오. 이 호출을 입력할 때에는 백의 작성을 제어하기 위해 하나 이상의 옵션을 제공합니다.

MQCreateBag 호출의 *Options* 매개변수는 사용자 백, 명령 백, 그룹 백 또는 관리 백을 작성할지를 선택하게 합니다.

사용자 백, 명령 백 또는 그룹 백을 작성하려면, 다음을 수행하기 위해 하나 이상의 추가 옵션을 선택할 수 있습니다.

- 백에 동일한 선택자가 두 번 이상 인접하여 나타나는 경우 목록 형식을 사용하십시오.
- 매개변수가 올바른 순서로 되어 있도록 하기 위해 PCF 메시지에 데이터 항목을 추가할 때 데이터 항목을 재정렬하십시오. 데이터 항목에 대한 자세한 정보는 49 페이지의 『데이터 항목』의 내용을 참조하십시오.
- 백에 추가할 항목에 대해 사용자 선택자 값을 검사하십시오.

관리 백은 자동으로 이러한 옵션을 내포합니다.

데이터 백은 해당 핸들로 식별됩니다. 백 핸들은 mqCreateBag으로부터 리턴되며 데이터 백을 사용하는 다른 모든 호출에 제공되어야 합니다.

mqCreateBag 호출에 대한 전체 설명은 mqCreateBag을 참조하십시오.

데이터 백 삭제

사용자가 작성한 데이터 백은 mpDeleteBag 호출을 사용하여 삭제되어야 합니다. 예를 들어, 백이 사용자 코드로 작성된 경우, 사용자 코드로 삭제되어야 합니다.

시스템 백은 MQAI가 자동으로 작성하고 삭제합니다. 이에 대한 자세한 정보는 55 페이지의 『mqExecute 호출을 사용하여 명령 서버에 관리 명령 보내기』의 내용을 참조하십시오. 사용자 코드는 시스템 백을 삭제할 수 없습니다.

mqDeleteBag 호출에 대한 전체 설명은 mqDeleteBag을 참조하십시오.

데이터 백 넣기 및 수신

mqPugBag 및 mqGetBag 호출을 사용하여 데이터 백을 넣고 가져오는 것으로 애플리케이션 사이에 데이터를 송신할 수도 있습니다. 이는 MQAI가 애플리케이션 보다 버퍼를 핸들링하도록 합니다. mqPutBag 호출은 지정된 백의 컨텐츠를 PCF 메시지로 변환하고 메시지를 지정된 큐로 보내고 mqGetBag 호출은 지정된 큐로부터 메시지를 제거하고 이를 데이터 백으로 변환합니다. 따라서, mqPutBag 호출은 MQPUT가 뒤에 오는 mqBagToBuffer 호출과 같고 mqGetBag은 mqBufferToBag이 뒤에 오는 MQGET 호출과 같습니다.

특정 큐에서 PCF 메시지를 보내고 수신하는 것에 대한 정보는 12 페이지의 『지정된 큐에서 PCF 메시지 송신 및 수신』의 내용을 참조하십시오.

참고: mqGetBag 호출을 사용하기 위해 선택하는 경우, 메시지 내의 PCF 세부사항은 정확해야 합니다. 그렇지 않으면, 적절한 오류 결과 및 PCF 메시지가 리턴되지 않습니다.

데이터 항목

데이터 항목은 작성될 때 데이터 백을 채우기 위해 사용됩니다. 이러한 데이터 항목은 사용자 또는 시스템 항목일 수 있습니다.

이러한 사용자 항목은 관리되고 있는 오브젝트의 속성과 같은 사용자 데이터를 포함합니다. 시스템 항목은 생성된 메시지를 통해 더 많은 제어를 위해 사용되어야 합니다(예 : 메시지 헤더의 생성). 시스템 항목에 대한 자세한 정보는 50 페이지의 『시스템 항목』의 내용을 참조하십시오.

데이터 항목의 유형

데이터 백을 작성할 때, 정수 또는 문자열 항목으로 채울 수 있습니다. 세 종류의 모든 항목에 대해 조회할 수 있습니다.

데이터 항목은 정수 또는 문자열 항목일 수 있습니다. MQAI에서 사용 가능한 데이터 항목의 유형은 다음과 같습니다.

- 정수
- 64비트 정수
- 정수 필터
- 문자열
- 문자열 필터
- 바이트 문자열
- 바이트 문자열 필터
- 백 핸들

데이터 항목 사용

데이터 항목을 사용하는 방법은 다음과 같습니다.

- 54 페이지의 『데이터 항목 계산』.
- 54 페이지의 『데이터 항목 삭제』.
- 50 페이지의 『백에 데이터 항목 추가』.
- 51 페이지의 『데이터 항목 필터링 및 조회』.

시스템 항목

시스템 항목이 다음에 사용될 수 있습니다.

- PCF 헤더의 생성. 시스템 항목은 PCF 명령 ID, 제어 옵션, 메시지 순서 매기기 및 명령 유형을 제어할 수 있습니다.
- 데이터 변환. 시스템 항목은 백에서 문자열 항목을 위한 문자 세트 ID를 핸들링합니다.

다른 모든 데이터 항목처럼, 시스템 항목은 선택자와 값으로 구성됩니다. 이러한 선택자 및 이의 용도에 대한 정보는 MQAI 선택자를 참조하십시오.

시스템 항목은 고유합니다. 하나 이상의 시스템 항목을 시스템 선택자로 식별될 수 있습니다. 각 시스템 선택자가 하나만 발생합니다.

대부분의 시스템 항목은 수정될 수 있지만(52 페이지의 『백 내의 정보 변경』의 내용 참조), 백 작성 옵션이 사용자에 의해 변경될 수 없습니다. 시스템 항목은 삭제할 수 없습니다 (54 페이지의 『데이터 항목 삭제』의 내용 참조.)

백에 데이터 항목 추가

데이터 백이 작성될 때, 데이터 항목으로 채울 수 있습니다. 이러한 데이터 항목은 사용자 또는 시스템 항목일 수 있습니다. 데이터 항목에 대한 자세한 정보는 49 페이지의 『데이터 항목』의 내용을 참조하십시오.

MQAI는 정수 항목과 64비트 정수 항목, 정수 필터 항목, 문자열 항목, 문자열 필터, 바이트 문자열 항목과 바이트 문자열 필터 항목을 백에 추가하게 하며 이것은 50 페이지의 그림 3에 표시되어 있습니다. 항목은 선택자로 식별됩니다. 일반적으로 한 선택자는 한 항목만을 식별하지만 항상 그런 것은 아닙니다. 지정된 선택자가 있는 데이터 항목이 이미 백에 있는 경우, 선택자의 추가 인스턴스가 백 끝에 추가됩니다.

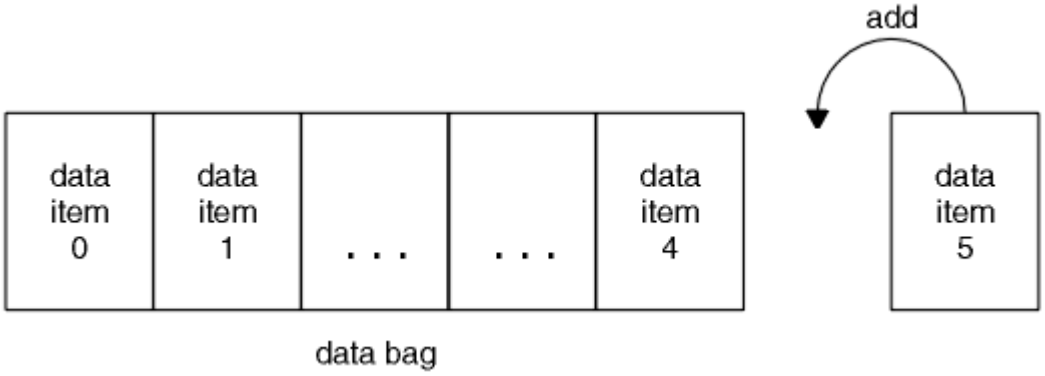


그림 *3.* 데이터 항목 추가

mqAdd* 호출을 사용하여 데이터 항목을 백에 추가하십시오.

- 정수 항목을 추가하려면 mqAddInteger에 설명된 대로 mqAddInteger 호출을 사용하십시오.
- 64비트 정수 항목을 추가하려면 mqAddInteger64에 설명된 대로 mqAddInteger64 호출을 사용하십시오.
- 정수 필터 항목을 추가하려면 mqAddIntegerFilter에 설명된 대로 mqAddIntegerFilter 호출을 사용하십시오.
- 문자열 항목을 추가하려면 mqAddString에 설명된 대로 mqAddString 호출을 사용하십시오.
- 문자열 필터 항목을 추가하려면 mqAddStringFilter에 설명된 대로 mqAddStringFilter 호출을 사용하십시오.
- 바이트 문자열 항목을 추가하려면 mqAddByteString에 설명된 대로 mqAddByteString 호출을 사용하십시오.

- 바이트 문자열 필터 항목을 추가하려면 mqAddByteStringFilter에 설명된 대로 mqAddByteStringFilter 호출을 사용하십시오.

데이터 항목을 백에 추가하는 것에 대한 자세한 정보는 50 페이지의 『시스템 항목』의 내용을 참조하십시오.

백에 조회 명령 추가

mqAddInquiry 호출은 백에 조회 명령을 추가하는 데 사용됩니다. 호출은 특히 관리 목적이므로 관리 백과 함께 사용될 수 있습니다. 이는 IBM MQ에서 조회하려는 속성의 선택자를 지정하도록 합니다.

mqAddInquiry 호출에 대한 전체 설명은 mqAddInquiry를 참조하십시오.

데이터 항목 필터링 및 조회

IBM MQ 오브젝트의 속성에 대해 조회하기 위해 MQAI를 사용할 때, 두 가지 방법으로 프로그램으로 리턴되는 데이터를 제어할 수 있습니다.

- mqAddInteger 및 mqAddString 호출을 사용하여 리턴되는 데이터를 **필터링** 할 수 있습니다. 이 접근방법은 *Selector* 및 *ItemValue* 쌍을 지정하게 합니다. 예:

  ```
  mqAddInteger(inputbag, MQIA_Q_TYPE, MQQT_LOCAL)
  ```

  이 예는 큐 유형(*Selector*)이 로컬이어야 하고(*ItemValue*) 조회 중인 것에 대한 오브젝트의 속성(이 경우, 큐)과 이 스펙이 일치해야 한다는 것을 지정합니다.

  필터링될 수 있는 기타 속성은 9 페이지의 『프로그래밍 가능 명령 형식 소개』에서 찾을 수 있는 PCF Inquire* 명령에 해당됩니다. 예를 들어, 채널의 속성에 대해 조회하려면 이 제품 문서에서 채널 조회 명령을 참조하십시오. 채널 조회 명령의 "필수 매개변수" 및 "선택적 매개변수"는 필터링에 사용할 수 있는 선택자를 식별합니다.
- mqAddInquiry 호출을 사용하는 오브젝트의 특정 속성을 **조회** 할 수 있습니다. 이는 관심있는 선택자를 지정합니다. 선택자를 지정하지 않으면, 오브젝트의 모든 속성이 리턴됩니다.

다음은 큐 속성의 필터링 및 조회의 예입니다.

```
/* Request information about all queues */
mqAddString(adminbag, MQCA_Q_NAME, "*")

/* Filter attributes so that local queues only are returned */
mqAddInteger(adminbag, MQIA_Q_TYPE, MQQT_LOCAL)

/* Query the names and current depths of the local queues */
mqAddInquiry(adminbag, MQCA_Q_NAME)
mqAddInquiry(adminbag, MQIA_CURRENT_Q_DEPTH)

/* Send inquiry to the command server and wait for reply */
mqExecute(MQCMD_INQUIRE_Q, ...)
```

데이터 항목의 필터링과 조회의 추가 예는 22 페이지의 『MQAI 사용의 예』의 내용을 참조하십시오.

데이터 백 내에서 조회

다음에 대해 조회할 수 있습니다.

- mqInquireInteger 호출을 사용하여 정수 항목 값. mqInquireInteger의 내용을 참조하십시오.
- mqInquireInteger64 호출을 사용하는 64비트 정수 항목의 값. mqInquireInteger64의 내용을 참조하십시오.
- mqInquireIntegerFilter 호출을 사용하는 정수 필터 항목의 값. mqInquireIntegerFilter의 내용을 참조하십시오.
- mqInquireString 호출을 사용하여 문자열 항목 값. mqInquireString의 내용을 참조하십시오.
- mqInquireStringFilter 호출을 사용하는 문자열 필터 항목의 값. mqInquireStringFilter의 내용을 참조하십시오.
- mqInquireByteString 호출을 사용하는 바이트 문자열 항목의 값. mqInquireByteString의 내용을 참조하십시오.

- mqInquireByteStringFilter 호출을 사용하는 바이트 문자열 필터 항목의 값. mqInquireByteStringFilter의 내용을 참조하십시오.
- mqInquireBag 호출을 사용하는 백 핸들 값. mqInquireBag의 내용을 참조하십시오.

또한 mqInquireItemInfo 호출을 사용하여 특정 항목의 유형(정수, 64비트 정수, 정수 필터, 문자열, 문자열 필터, 바이트 문자열, 바이트 문자열 필터 또는 백 핸들)에 대해 조사할 수 있습니다. mqInquireItemInfo의 내용을 참조하십시오.

백 내의 정보 변경

MQAI는 mqSet* 호출을 사용하여 백 내의 정보를 변경하도록 합니다. 해당 도움말을 보려면 다음을 수행하십시오.

1. 백 내의 데이터 항목을 수정합니다. 색인은 수정될 항목의 발생을 식별하여 매개변수의 개별 인스턴스가 대체될 수 있게 합니다(52 페이지의 그림 4의 내용 참조).

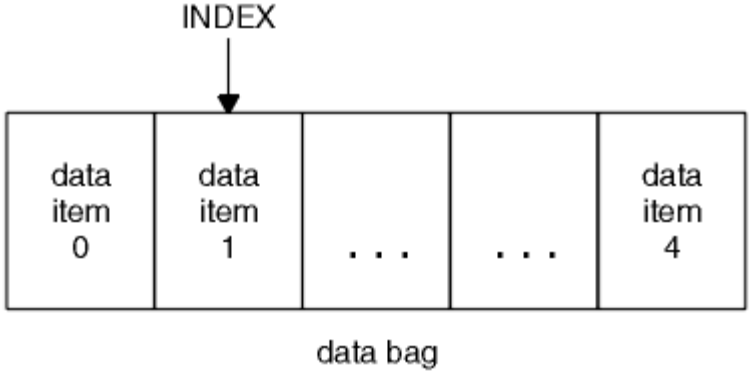


그림 *4.* 단일 데이터 항목 수정

2. 지정된 선택자의 기존 모든 발생을 삭제하고 백의 끝에 새로운 발생을 추가합니다. (52 페이지의 그림 5의 내용 참조.) 특수 색인 값을 사용하면 매개변수의 **모든** 인스턴스가 대체될 수 있습니다.

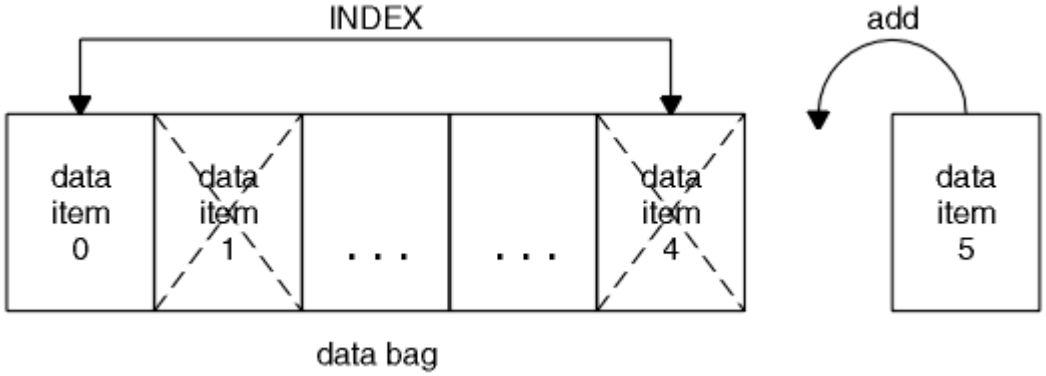


그림 *5.* 모든 데이터 항목 수정

참고: 색인은 백 내의 삽입 순서를 보존하지만, 기타 데이터 항목의 색인에 영향을 미칠 수 있습니다.

mqSetInteger 호출을 사용하여 백 내의 정수 항목을 수정하십시오. mqSetInteger64 호출을 사용하여 64비트 정수 항목을 수정하십시오. mqSetIntegerFilter 호출을 사용하여 정수 필터 항목을 수정하십시오. mqSetString 호출을 사용하여 문자열 항목을 수정하십시오. mqSetStringFilter 호출을 사용하여 문자열 필터 항목을 수정하십시오. mqSetByteString 호출을 사용하여 바이트 문자열 항목을 수정하십시오. mqSetByteStringFilter 호출을 사용하여 바이트 문자열 필터 항목을 수정하십시오. 또는 이러한 호출을 사용하여 지정된 선택자의 기존 모든 발생을 삭제하고 백의 끝에 새 발생을 추가할 수 있습니다. 데이터 항목은 사용자 항목 또는 시스템 항목입니다.

이러한 호출에 대한 전체 설명은 다음을 참조하십시오.

- mqSetInteger
- mqSetInteger64
- mqSetIntegerFilter

- mqSetString
- mqSetStringFilter
- mqSetByteString
- mqSetByteStringFilter

mqClearBag 호출을 사용하여 백 지우기

mqClearBag 호출은 사용자 백에서 모든 사용자 항목을 제거하고 초기값으로 시스템 항목을 재설정합니다. 백에 포함된 시스템 백도 삭제됩니다.

mqClearBag 호출에 대한 전체 설명은 mqClearBag을 참조하십시오.

mqTruncateBag 호출을 사용하여 백 자르기

mqTruncateBag 호출은 가장 최근 추가된 항목부터 백의 끝에서 항목을 삭제하여 사용자 백에 있는 사용자 항목의 수를 줄입니다. 예를 들어, 둘 이상의 메시지를 생성하기 위해 동일한 헤더 정보를 사용할 때 사용될 수 있습니다.

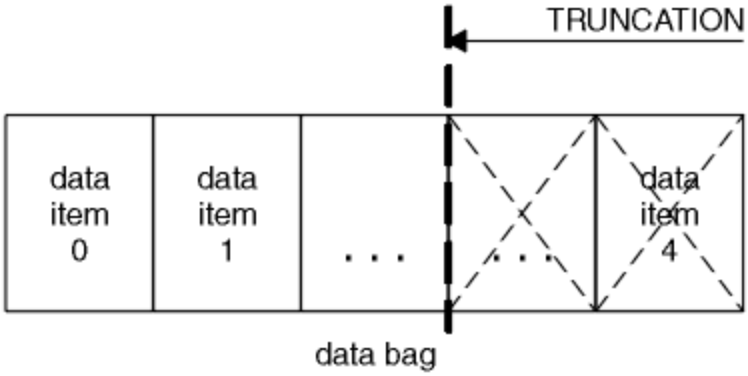


그림 *6.* 백 자르기

mqTruncateBag 호출에 대한 전체 설명은 mqTruncateBag을 참조하십시오.

백 및 버퍼 변환

애플리케이션 사이에 데이터를 송신하려면, 먼저 메시지 데이터를 백에 놓습니다. 그런 다음, mqBagToBuffer 호출을 사용하여 백에 있는 데이터를 PCF 메시지로 변환합니다. MQPUT 호출을 사용하여 PCF 메시지를 필수 큐로 송신합니다. 이 내용은 53 페이지의 그림 7에 표시되어 있습니다. mqBagToBuffer 호출에 대한 전체 설명은 mqBagToBuffer를 참조하십시오.

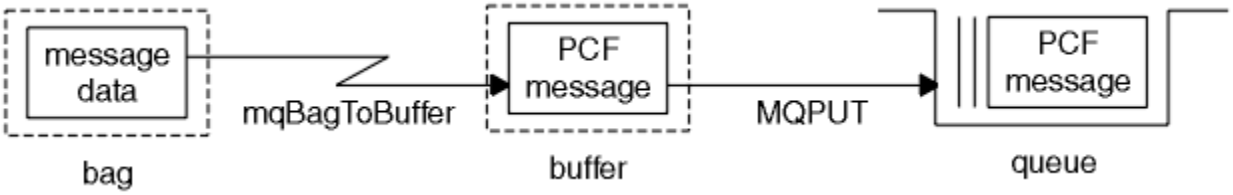


그림 *7. PCF* 메시지로 백 변환

데이터를 수신하려면, 먼저 메시지가 MQGET 호출을 사용하여 버퍼로 수신됩니다. 그런 다음, 버퍼에 올바른 PCF 메시지가 있다는 가정 하에 mqBufferToBag 호출을 사용하여 버퍼에 있는 데이터를 백으로 변환합니다. 이 내용은 54 페이지의 그림 8에 표시되어 있습니다. mqBufferToBag 호출에 대한 전체 설명은 mqBufferToBag을 참조하십시오.

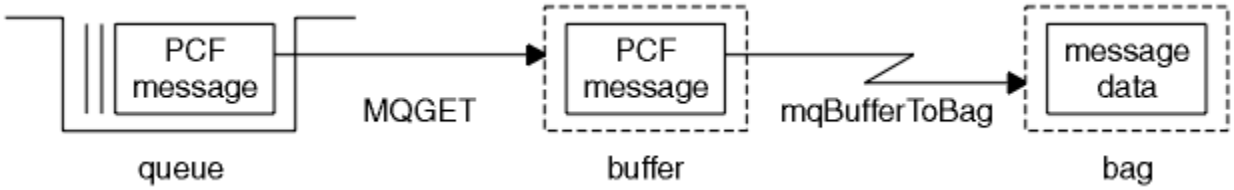


그림 *8.* 백 양식으로 *PCF* 메시지 변환

데이터 항목 계산

mqCountItems 호출로 데이터 백에 저장된 사용자 항목, 시스템 항목 또는 모두의 수를 계산하여 이 숫자를 리턴할 수 있습니다. 예를 들어, `mqCountItems( Bag, 7, ...)`는 7인선택기가 있는 백에 있는 항목 수를 리턴합니다. 개별 선택자, 사용자 선택자, 시스템 선택자 또는 모든 선택자별로 항목을 계수할 수 있습니다.

참고: 이 호출은 백에 있는 고유 선택자 수가 아닌 데이터 항목 수를 계수합니다. 선택자는 여러 번 발생할 수 있으므로 데이터 항목 수보다 백에 있는 고유 선택자 수가 더 적을 수 있습니다.

mqCountItems 호출에 대한 전체 설명은 mqCountItems를 참조하십시오.

데이터 항목 삭제

여러 방식으로 백에서 항목을 삭제할 수 있습니다. 해당 도움말을 보려면 다음을 수행하십시오.

- 백에서 하나 이상의 사용자 항목을 제거합니다. 자세한 정보는 54 페이지의 『mqDeleteItem 호출을 사용하여 백에서 데이터 항목 삭제』 를 참조하십시오.
- 백에서 **모든** 사용자 항목을 삭제하십시오(즉, 백 지우기). 자세한 정보는 53 페이지의 『mqClearBag 호출을 사용하여 백 지우기』의 내용을 참조하십시오.
- 백의 끝에서 사용자 항목을 삭제하십시오(즉, 백 자르기). 자세한 정보는 53 페이지의 『mqTruncateBag 호출을 사용하여 백 자르기』 를 참조하십시오.

mqDeleteItem 호출을 사용하여 백에서 데이터 항목 삭제

mqDeleteItem 호출은 백에서 하나 이상의 사용자 항목을 제거합니다. 색인을 사용하여 다음 중 하나를 삭제합니다.

1. 지정된 선택자의 단일 발생. (54 페이지의 그림 9의 내용 참조.)

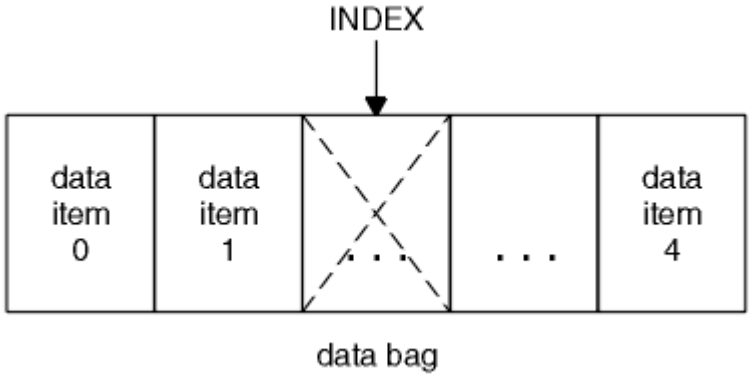


그림 *9.* 단일 데이터 항목 삭제

또는

2. 지정된 선택자의 모든 발생. (55 페이지의 그림 10의 내용 참조.)

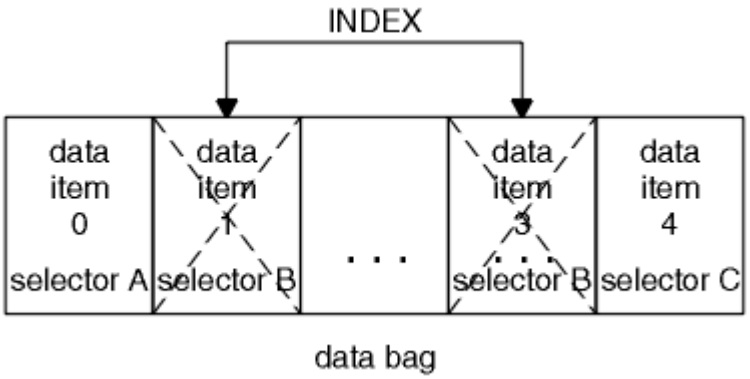


그림 *10.* 모든 데이터 항목 삭제

참고: 색인은 백 내의 삽입 순서를 보존하지만, 기타 데이터 항목의 색인에 영향을 미칠 수 있습니다. 예를 들어, mqDeleteItem 호출은 삭제된 항목 다음에 나오는 데이터 항목의 색인 값을 보존하지 않습니다. 삭제된 항목으로 인해 남은 공간을 채우기 위해 색인을 재구성하기 때문입니다.

mqDeleteItem 호출에 대한 전체 설명은 mqDeleteItem을 참조하십시오.

mqExecute 호출을 사용하여 명령 서버에 관리 명령 보내기

데이터 백이 작성되고 채워지면, 관리 명령 메시지가 mqExecute 호출을 사용하여 큐 관리자의 명령 서버에 보내질 수 있습니다. 이는 명령 서버와의 교환을 처리하고 백에 응답을 리턴합니다.

데이터 백을 작성하고 채운 후, 관리 명령 메시지를 큐 관리자의 명령 서버에 보낼 수 있습니다. 이를 수행하는 가장 쉬운 방법은 mqExecute 호출을 사용하는 것입니다. mqExecute 호출은 관리 명령 메시지를 비지속 메시지로 보내고 응답을 기다립니다. 응답은 응답 백에 리턴됩니다. 이는 여러 IBM MQ 오브젝트 또는 일련의 PCF 오류 응답 메시지와 관련되는 속성에 대한 정보를 포함할 수 있습니다. 따라서, 응답 백은 리턴 코드만 포함할 수도 있고 중첩 백을 포함할 수도 있습니다.

응답 메시지는 시스템이 작성하는 시스템 백에 배치됩니다. 예를 들어, 오브젝트 이름에 대해 조회하기 위해 시스템 백이 해당 오브젝트 이름을 보유하도록 작성되고 해당 백이 사용자 백에 삽입됩니다. 그런 다음 이러한 백에 대한 핸들이 응답 백에 삽입되고 MQHA_BAG_HANDLE 선택자가 중첩 백에 액세스할 수 있습니다. 시스템 백이 삭제되지 않으면 응답 백이 삭제될 때까지 스토리지에 계속 있습니다.

중첩의 개념이 55 페이지의 그림 11에 표시됩니다.

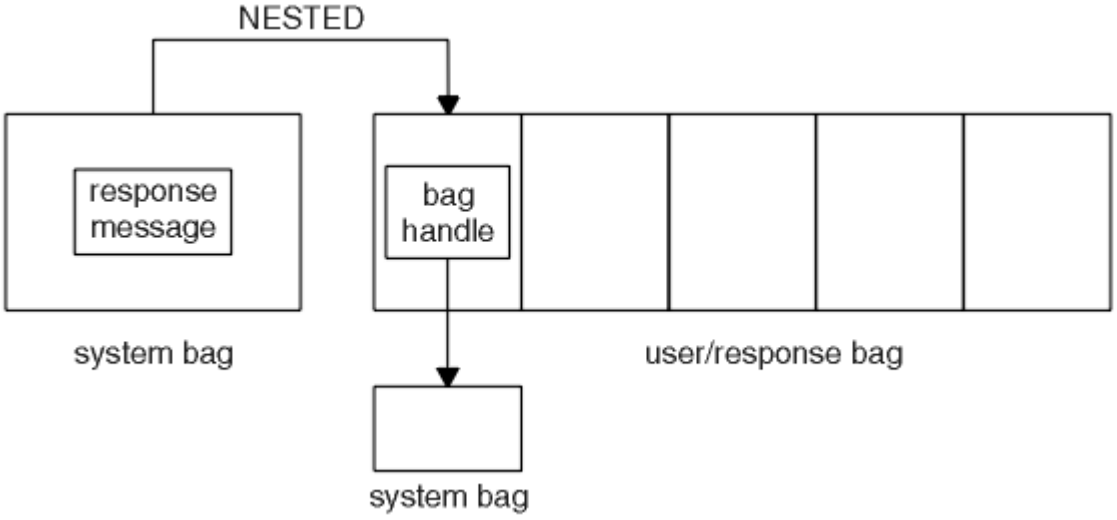


그림 *11.* 중첩

mqExecute 호출에 대한 입력으로서 다음을 제공해야 합니다.

- MQI 연결 핸들.
- 실행할 명령. MQCMD_* 값 중 하나여야 합니다.

참고: 이 값을 MQAI에서 인식하지 못하는 경우, 값은 여전히 허용됩니다. 그러나, 백에 값을 삽입하기 위해 mqAddInquiry 호출이 사용되는 경우, 이 매개변수는 MQAI에서 인식하는 INQUIRE 명령이어야 합니다. 즉, 매개변수는 MQCMD_INQUIRE_* 양식 중 하나여야 합니다.

- 선택적으로, 호출 처리를 제어하는 옵션을 포함하는 백의 핸들. 이는 또한 MQAI가 각 응답 메시지를 기다려야 하는 최대 시간(밀리초)을 지정할 수 있는 위치입니다.
- 실행할 관리 명령의 세부사항을 포함하는 관리 백의 핸들.
- 응답 메시지를 수신하는 응답 백의 핸들.

다음은 선택적입니다.

- 관리 명령이 대체되는 큐의 오브젝트 핸들.

  지정되는 오브젝트 핸들이 없는 경우, 관리 명령은 현재 연결된 큐 관리자에 속하는 SYSTEM.ADMIN.COMMAND.QUEUE에 배치됩니다. 기본값입니다.
- 응답 메시지가 대체되는 큐의 오브젝트 핸들.

  MQAI에서 자동으로 작성하는 동적 큐에 응답 메시지를 배치하기 위해 선택할 수 있습니다. 작성된 큐는 호출 지속 기간에만 존재하며 mqExecute 호출 종료 시 MQAI에 의해 삭제됩니다.

mqExecute 호출의 사용 예제는 예제 코드를 참조하십시오.

MQ Explorer를 사용하여 관리

MQ Explorer를 사용하면 Windows 또는 Linux x86-64에서만 실행 중인 컴퓨터에서 네트워크의 로컬 또는 원격 관리를 수행할 수 있습니다.

IBM MQ for Windows 및 Linux x86-64 용 IBM MQ에서는 제어 또는 MQSC 명령 사용에 대한 대안으로서 관리 태스크를 수행하기 위해 MQ Explorer라고 하는 관리 인터페이스를 제공합니다. 명령 세트 비교에서는 MQ Explorer를 사용하여 수행할 수 있는 것을 표시합니다.

흥미 있는 큐 관리자 및 클러스터에서 MQ Explorer를 가리켜서 MQ Explorer를 사용하면 Windows 또는 Linux x86-64를 실행 중인 컴퓨터에서 네트워크의 로컬 또는 원격 관리를 수행할 수 있습니다. MQ Explorer 를 사용하여 관리할 수 있는 IBM MQ 의 플랫폼 및 레벨은 57 페이지의 『리모트 큐 관리자』 에 설명되어 있습니다.

MQ Explorer 가 이를 관리할 수 있도록 원격 IBM MQ 큐 관리자를 구성하려면 58 페이지의 『필수 소프트웨어 및 정의』 을 참조하십시오.

이를 사용하면 Windows 또는 Linux x86-64 시스템 도메인 내에서 로컬 또는 원격으로 IBM MQ에 대한 작업 환경을 설정하고 미세 조정하는 것과 일반적으로 연관되는 태스크를 수행할 수 있습니다.

Linux에서 MQ Explorer는 Eclipse가 둘 이상 설치된 경우 시작되지 못할 수 있습니다. 이것이 발생하면 기타 Eclipse 설치에 사용하는 것에 다른 사용자 ID를 사용하여 MQ Explorer를 시작하십시오.

Linux에서 MQ Explorer를 성공적으로 시작하려면 홈 디렉토리가 존재하고 이 홈 디렉토리에 파일을 작성할 수 있어야 합니다.

IBM MQ Explorer로 할 수 있는 작업

이는 IBM MQ Explorer를 사용하여 수행할 수 있는 태스크의 목록입니다.

IBM MQ 탐색기로 다음을 수행할 수 있습니다.

- 큐 관리자를 작성하고 삭제하십시오(로컬 시스템 전용).
- 큐 관리자를 시작하고 중지하십시오(로컬 시스템 전용).
- 큐 및 채널과 같은 IBM MQ 오브젝트를 정의하고 해당 정의를 표시 및 대체하십시오.
- 큐에서 메시지를 찾으십시오.
- 채널을 시작하고 중지하십시오.
- 채널, 리스너, 큐 또는 서비스 오브젝트에 대한 상태 정보를 보십시오.
- 클러스터에서 큐 관리자를 보십시오.
- 애플리케이션, 사용자 또는 채널이 특정 큐를 여는지 확인하기 위해 검사하십시오.

- 새 클러스터 작성 마법사를 사용하여 새 큐 관리자 클러스터를 작성하십시오.
- 클러스터에 큐 관리자 추가 마법사를 사용하여 클러스터에 큐 관리자를 추가하십시오.
- SSL(Secure Sockets Layer) 채널 보안과 함께 사용되는 인증 정보 오브젝트를 관리하십시오.
- 채널 시작기, 트리거 모니터 및 리스너를 작성하고 삭제하십시오.
- 명령 서버, 채널 시작기, 트리거 모니터 및 리스너를 시작하거나 중지하십시오.
- 큐 관리자가 시작될 때 자동으로 시작되도록 특정 서비스를 설정하십시오.
- 큐 관리자의 특성을 수정하십시오.
- 로컬 기본 큐 관리자를 변경하십시오.
- **strmqikm** (ikeyMan) GUI를 호출하여 SSL (Secure Sockets Layer) 인증서를 관리하고, 인증서를 큐 관리자와 연관시키고, 인증서 저장소를 구성하고 설정하십시오 (로컬 시스템에만 해당).
- IBM MQ 오브젝트에서 JMS 오브젝트를 작성하고 JMS 오브젝트에서 IBM MQ 오브젝트를 작성하십시오.
- 현재 지원되는 유형에 대한 JMS 연결 팩토리를 작성하십시오.
- 서비스에 대한 매개변수(예: 리스너의 경우 TCP 포트 번호) 또는 채널 시작기 큐 이름을 수정하십시오.
- 서비스 추적을 시작 또는 중지하십시오.

일련의 컨텐츠 보기 및 특성 대화 상자를 사용하여 관리 태스크를 수행합니다.

컨텐츠 보기
컨텐츠 보기는 다음을 표시할 수 있는 패널입니다.

- IBM MQ 자체에 관한 속성 및 관리 옵션.
- 하나 이상의 관련된 오브젝트에 관한 속성 및 관리 옵션.
- 클러스터에 대한 속성 및 관리 옵션.

특성 대화 상자
특성 대화 상자는 오브젝트에 관한 속성을 일련의 필드에 표시하는 패널로, 필드 중 일부를 편집할 수 있습니다.

Navigator 보기를 사용하여 IBM MQ 탐색기를 탐색합니다. 네비게이터를 사용하면 요청하는 컨텐츠 보기를 선택할 수 있습니다.

리모트 큐 관리자

연결할 수 있는 지원되는 큐 관리자에 두 가지 예외가 있습니다.

Windows 또는 Linux(x86및 x86-64 플랫폼) 시스템으로부터 IBM MQ Explorer는 다음 예외를 가지는 지원되는 모든 큐 관리자에 연결할 수 있습니다.

- IBM MQ for z/OS 큐 관리자가 버전 6.0보다 이전입니다.
- 현재 지원되는 MQSeries® V2 큐 관리자.

IBM MQ Explorer는 다른 명령 레벨과 플랫폼 사이의 기능에 있는 차이점을 처리합니다. 그러나, 인식하지 못하는 속성이 발생하는 경우, 해당 속성은 보이지 않습니다.

IBM MQ V5.3 컴퓨터에서 IBM MQ 탐색기를 사용하여 Windows 에서 V6.0 이상의 큐 관리자를 원격으로 관리하려는 경우, IBM MQ for Windows V5.3 컴퓨터에 수정팩 9 (CSD9) 이상을 설치해야 합니다.

If you intend to remotely administer a V5.3 queue manager on iSeries using the IBM MQ Explorer on an IBM MQ V6.0 or later computer, you must install Fix Pack 11 (CSD11) or later on your IBM MQ for iSeries V5.3 computer. 이 수정팩은 IBM MQ Explorer와 iSeries 큐 관리자 사이의 연결 문제점을 수정합니다.

IBM MQ Explorer 사용 여부 결정

설치 시 IBM MQ Explorer를 사용할지 여부를 결정할 때 이 토픽에 나열된 정보를 고려하십시오.

다음과 같은 점을 알고 있어야 합니다.

오브젝트 이름
MQSC 명령을 사용하여 오브젝트에 대해 작업할 때 IBM MQ Explorer와 함께 큐 관리자 및 기타 오브젝트에 대한 소문자 이름을 사용하는 경우, 오브젝트 이름을 작은따옴표로 묶어야 합니다. 그렇지 않으면 IBM MQ가 이를 인식하지 않습니다.

큰 큐 관리자
IBM MQ Explorer는 작은 큐 관리자로 가장 잘 작동됩니다. 단일 큐 관리자에 많은 수의 오브젝트를 가지는 경우, IBM MQ Explorer가 보기에 표시되도록 필수 정보를 추출하는 동안 지연을 경험할 수 있습니다.

클러스터
IBM MQ 클러스터는 잠재적으로 수백 또는 수천 개의 큐 관리자를 포함할 수 있습니다. IBM MQ Explorer는 트리 구조를 사용하여 클러스터에서 큐 관리자를 제공합니다. 클러스터의 실제 크기는 선택할 때까지 IBM MQ Explorer가 클러스터에서 큐 관리자에 연결되지 않으므로 IBM MQ Explorer의 속도에 자동으로 영향을 주지 않습니다.

IBM MQ Explorer 설정

이 섹션은 IBM MQ Explorer를 설정하기 위해 사용해야 하는 단계를 파악합니다.

- 58 페이지의 『필수 소프트웨어 및 정의』
- 58 페이지의 『보안』
- 62 페이지의 『큐 관리자와 클러스터 표시 및 숨기기』
- 62 페이지의 『클러스터 멤버십』
- 63 페이지의 『데이터 변환』

필수 소프트웨어 및 정의

MQ Explorer 사용을 시도하기 전에 다음 요구사항을 충족하는지 확인하십시오.

MQ Explorer는 TCP/IP 통신 프로토콜을 사용하여 리모트 큐 관리자에 연결될 수 있습니다.

다음을 확인하십시오.

1. 명령 서버는 원격으로 관리되는 모든 큐 관리자에서 실행되고 있습니다.
2. 적절한 TCP/IP 리스너 오브젝트는 모든 리모트 큐 관리자에서 실행되어야 합니다. 이 오브젝트는 IBM MQ 리스너 또는 유닉스 및 Linux 시스템에 inetd 디먼일 수 있습니다.
3. 기본 이름 지정된 SYSTEM.ADMIN.SVRCONN에 의해 서버 연결 채널은 모든 리모트 큐 관리자에 존재합니다.

   다음 MQSC 명령을 사용하여 채널을 작성할 수 있습니다.

   ```
   DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   ```

   이 명령은 기본 채널 정의를 작성합니다. 더 복잡한 정의(예: 보안을 설정하기 위해)를 원하는 경우,추가 매개변수가 필요합니다. 자세한 정보는 채널 정의를 참조하십시오.
4. 시스템 큐(SYSTEM.MQEXPLORER.REPLY.MODEL)가 존재해야 합니다.

보안

특정 오브젝트에 대한 사용자 액세스 제어가 중요한 환경에서 IBM MQ을 사용 중인 경우 IBM MQ Explorer를 사용하는 보안 측면을 고려해야 합니다.

*IBM MQ Explorer*를 사용하기 위한 권한 부여

사용자는 IBM MQ 탐색기를 사용할 수 있지만, 큐 관리자에 연결 및 액세스하고 이를 관리하기 위해서는 특정 권한이 필요합니다.

IBM MQ 탐색기를 사용하여 로컬 관리 태스크를 수행하려면, 사용자는 관리 태스크를 수행하기 위한 필수 권한을 가지고 있어야 합니다. 사용자가 mqm 그룹의 구성원인 경우, 사용자에게 모든 로컬 관리 태스크를 수행하기 위한 권한이 있습니다.

IBM MQ 탐색기를 사용하여 리모트 큐 관리자에 연결하고 원격 관리 태스크를 수행하기 위해 IBM MQ 탐색기를 실행하는 사용자는 다음 권한을 가지고 있어야 합니다.

- 대상 큐 관리자 오브젝트의 CONNECT 권한
- 대상 큐 관리자 오브젝트의 INQUIRE 권한
- 대상 큐 관리자 오브젝트의 DISPLAY 권한
- 큐 SYSTEM.MQEXPLORER.REPLY.MODEL에 대한 INQUIRE 권한
- 큐 SYSTEM.MQEXPLORER.REPLY.MODEL에 대한 DISPLAY 권한
- 큐 SYSTEM.MQEXPLORER.REPLY.MODEL에 대한 INPUT(get) 권한
- 큐 SYSTEM.ADMIN.COMMAND.QUEUE에 대한 OUTPUT(put) 권한
- 큐 SYSTEM.ADMIN.COMMAND.QUEUE의 INQUIRE 권한
- 선택된 조치를 수행하기 위한 권한

참고: INPUT 권한은 큐로부터 사용자로의 입력(get 조작)과 관련됩니다. OUTPUT 권한은 사용자에서 큐로의 출력에 관련됩니다(Put 조작).

IBM MQ for z/OS에서 리모트 큐 관리자에 연결하고 IBM MQ 탐색기를 사용하여 원격 관리 태스크를 수행하려면 다음이 제공되어야 합니다.

- 시스템 큐 SYSTEM.MQEXPLORER.REPLY.MODEL에 대한 RACF® 프로파일
- 큐 AMQ.MQEXPLORER.*에 대한 RACF 프로파일

또한, IBM MQ 탐색기를 실행하는 사용자는 다음 권한을 가지고 있어야 합니다.

- 시스템 큐 SYSTEM.MQEXPLORER.REPLY.MODEL에 대한 RACF UPDATE 권한
- 큐 AMQ.MQEXPLORER.*에 대한 RACF UPDATE 권한
- 대상 큐 관리자 오브젝트의 CONNECT 권한
- 선택된 조치를 수행하기 위한 권한
- MQCMDS 클래스에서 모든 hlq.DISPLAY.object 프로파일에 대한 READ 권한

IBM MQ 오브젝트에 권한을 부여하는 방법에 대한 정보는 UNIX 또는 Linux 시스템에서 IBM MQ 오브젝트에 대한 액세스 부여 및 Windows 의 내용을 참조하십시오.

사용자가 수행하기 위해 권한을 부여 받지 않는 조작을 수행하려고 시도하면, 대상 큐 관리자는 인증 실패 프로시저를 호출하고 작동에 실패합니다.

IBM MQ 탐색기의 기본 필터는 모든 IBM MQ 오브젝트를 표시하는 것입니다. 사용자에게 DISPLAY 권한이 없는 IBM MQ 오브젝트가 있는 경우, 권한 실패가 생성됩니다. 권한 이벤트가 기록되는 경우,사용자에게 DISPLAY 권한이 있는 해당 오브젝트에 표시되는 오브젝트 범위를 제한하십시오.

리모트 큐 관리자에 연결하기 위한 보안

IBM MQ Explorer와 각 리모트 큐 관리자 사이의 채널을 보안화해야 합니다.

IBM MQ Explorer는 MQI 클라이언트 애플리케이션으로서 리모트 큐 관리자에 연결합니다. 이는 각 리모트 큐 관리자에 서버 연결 채널의 정의 및 적절한 TCP/IP 리스너가 있어야 한다는 것을 의미합니다. 서버 연결 채널을 보안 설정하지 않으면 악의적인 애플리케이션이 동일한 서버 연결 채널에 연결하고 무제한 권한으로 큐 관리자 오브젝트에 액세스할 수 있습니다. 서버 연결 채널을 보안 설정하려면 채널의 MCAUSER 속성에 공백이 아닌 값을 지정하거나 채널 인증 레코드를 사용하거나 보안 엑시트를 사용하십시오.

MCAUSER 속성의 기본값은 로컬 사용자 ID입니다. 서버 연결 채널의 MCAUSER 속성으로 비공백 사용자 이름을 지정하는 경우, 이 채널을 사용하여 큐 관리자에 연결되는 모든 프로그램은 이름 지정된 사용자의 ID로 실행되고 동일한 권한 레벨을 가집니다. 이는 채널 인증 레코드를 사용하는 경우에는 발생하지 않습니다.

*IBM MQ Explorer*로 보안 엑시트 사용

IBM MQ Explorer를 사용하여 기본 보안 엑시트 및 큐 관리자 특정 보안 엑시트를 지정할 수 있습니다.

기본 보안 엑시트를 정의할 수 있고, 이는 IBM MQ Explorer에서 모든 새 클라이언트 연결에 사용될 수 있습니다. 연결이 작성될 때 이 기본 엑시트는 대체될 수 있습니다. 또한 단일 큐 관리자 또는 일련의 큐 관리자에 대한 보안

엑시트를 정의할 수 있으며, 연결이 작성될 때 적용됩니다. IBM MQ Explorer를 사용하여 엑시트를 지정합니다. 자세한 정보는 IBM MQ Help Center를 참조하십시오.

*MQ Explorer*를 사용하여 *SSL* 사용 *MQI* 채널을 사용하는 리모트 큐 관리자에 연결

MQ Explorer는 MQI 채널을 사용하여 리모트 큐 관리자에 연결됩니다. SSL 보안을 사용하여 MQI 채널을 보안화하려면, 클라이언트 채널 정의 테이블을 사용하여 채널을 설정해야 합니다.

클라이언트 채널 정의 테이블을 사용하여 MQI 채널을 설정하는 방법에 대한 정보는 IBM MQ MQI clients의 개요를 참조하십시오.

When you have established the channel using a client channel definition table, you can use the MQ Explorer to connect to a remote queue manager using SSL-enabled MQI channel, as described in 60 페이지의 『리모트 큐 관리자를 호스트하는 시스템의 태스크』 and 60 페이지의 『MQ Explorer를 호스트하는 시스템의 태스크』.

리모트 큐 관리자를 호스트하는 시스템의 태스크

리모트 큐 관리자를 호스트하는 시스템에서 다음 태스크를 수행하십시오.

1. 채널의 서버 연결 및 클라이언트 연결 쌍을 정의하고 두 채널의 서버 연결에서 *SSLCIPH* 속성에 적합한 값을 지정하십시오. *SSLCIPH* 속성에 대한 자세한 정보는 SSL을 사용하여 채널 보호 를 참조하십시오.
2. 큐 관리자의 `@ipcc` 디렉토리에 있는 채널 정의 테이블 `AMQCLCHL.TAB`을(를) MQ Explorer을(를) 호스팅하는 시스템으로 전송하십시오.
3. 지정된 포트에서 TCP/IP 리스너를 시작하십시오.
4. 큐 관리자의 SSL 디렉토리에 CA 및 개인 SSL 인증서 모두를 배치하십시오.
   - `/var/mqm/qmgrs/+QMNAME+/SSL` for 유닉스 및 Linux systems
   - `C: \Program Files\IBM\WebSphere MQ\qmgrs\+QMNAME+\SSL` for Windows systems

     여기서 `+QMNAME+`은 큐 관리자의 이름을 표시하는 토큰입니다.
5. `key.kdb`라는 CMS 유형의 키 데이터베이스 파일을 작성하십시오. **strmqikm**(iKeyman) GUI에서 해당 옵션을 선택하거나 **runmqckm** 명령과 함께 `-stash` 옵션을 사용하여 비밀번호를 파일에 숨기십시오.
6. 이전 단계에서 작성된 키 데이터베이스에 CA 인증서를 추가하십시오.
7. 큐 관리자에 대한 개인 인증서를 키 데이터베이스로 가져오십시오.

Windows 시스템에서 SSL(Secure Sockets Layer)에 대해 작업하는 것에 대한 자세한 정보는 UNIX, Linux 및 Windows 시스템에서 SSL 또는 TLS에 대한 작업을 참조하십시오.

MQ Explorer를 호스트하는 시스템의 태스크

MQ Explorer를 호스트하는 시스템에서 다음 태스크를 수행하십시오.

1. `key.jks`라는 JKS 유형의 키 데이터베이스 파일을 작성하십시오. 이 키 데이터베이스 파일에 대한 비밀번호를 설정하십시오.

   MQ Explorer는 SSL 보안을 위한 Java 키 저장소 파일(JKS)을 사용하고 MQ Explorer에 대한 SSL을 구성하기 위해 작성되는 키 저장소 파일이 이와 일치해야 합니다.
2. 이전 단계에서 작성된 키 데이터베이스에 CA 인증서를 추가하십시오.
3. 큐 관리자에 대한 개인 인증서를 키 데이터베이스로 가져오십시오.
4. Windows 및 Linux 시스템에서 시스템 메뉴, `MQExplorer` 실행 가능 파일 또는 **strmqcfg** 명령을 사용하여 MQ Explorer를 시작하십시오.
5. MQ Explorer 도구 모음에서 **창 -> 환경 설정**을 클릭한 다음, **IBM MQ 탐색기**를 펼치고 **SSL 클라이언트 인증서 저장소**를 클릭하십시오. 신뢰성있는 인증서 저장소 및 개인 인증서 저장소 모두에 60 페이지의 『MQ Explorer를 호스트하는 시스템의 태스크』의 1단계에서 작성된 JKS 파일의 이름 및 비밀번호를 입력한 다음, **확인**을 클릭하십시오.
6. **환경 설정** 창을 닫고 **큐 관리자**를 마우스 오른쪽 단추로 클릭하십시오. **큐 관리자 표시/숨기기**를 클릭한 다음 **큐 관리자 표시/숨기기** 화면에서 **추가**를 클릭하십시오.

7. 큐 관리자의 이름을 입력하고 **직접 연결** 옵션을 선택하십시오. 다음을 클릭하십시오.
8. **클라이언트 채널 정의 테이블(CCDT) 사용**을 선택하고 리모트 큐 관리자를 호스트하는 시스템에서 60 페이지의 『리모트 큐 관리자를 호스트하는 시스템의 태스크』의 2단계에서 리모트 큐 관리자로부터 전송한 채널 테이블 파일의 위치를 지정하십시오.
9. **마침**을 클릭하십시오. MQ Explorer에서 리모트 큐 관리자에 액세스할 수 있습니다.

다른 큐 관리자를 통한 연결

IBM MQ Explorer를 사용하면 IBM MQ Explorer가 이미 연결되어 있는 큐 관리자에 중간 큐 관리자를 통해 연결할 수 있습니다.

이 경우, IBM MQ Explorer는 다음을 지정하는 중간 큐 관리자에 PCF 명령 메시지를 둡니다.

- 대상 큐 관리자의 이름으로서 오브젝트 디스크립터(MQOD)의 *ObjectQMgrName* 매개변수. 큐 이름 해석에 대한 자세한 정보는 이름 해석을 참조하십시오.
- 로컬 사용자 ID로서 메시지 디스크립터(MQMD)의 *UserIdentifier* 매개변수.

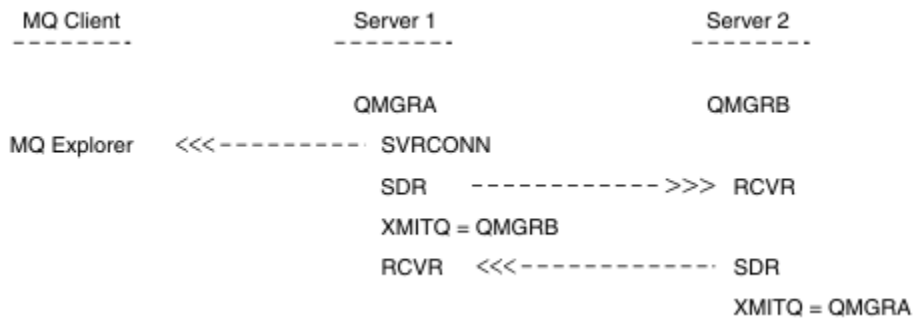
연결이 중간 큐 관리자를 통해 대상 큐 관리자에 연결하기 위해 사용되는 경우, 사용자 ID가 메시지 디스크립터(MQMD)의 *UserIdentifier* 매개변수에 다시 플로우됩니다. 대상 큐 관리자의 MCA 리스너가 이 메시지를 승인하기 위해서는 MCAUSER 속성이 설정되어야 하거나 put 권한이 있는 사용자 ID가 이미 존재해야 합니다.

대상 큐 관리자의 명령 서버는 메시지 디스크립터(MQMD)에서 *UserIdentifier* 매개변수에 사용자 ID를 지정하여 메시지를 송신 큐에 넣습니다. 이러한 넣기가 성공하기 위해서는 대상 큐 관리자에 넣기 권한이 있는 사용자 ID가 이미 존재해야 합니다.

다음 예는 중간 큐 관리자를 통해 큐 관리자를 IBM MQ Explorer에 연결하는 방법을 표시합니다.

큐 관리자에 대한 원격 관리 연결을 설정하십시오. 다음을 확인하십시오.

- 서버의 큐 관리자가 활성이고 서버 연결 채널(SVRCONN)이 정의되어 있습니다.
- 리스너가 활성입니다.
- 명령 서버가 활성입니다.
- SYSTEM.MQ EXPLORER.REPLY.MODEL 큐가 작성되었고 사용자에게 충분한 권한이 있습니다.
- 큐 관리자 리스너, 명령 서버 및 송신자 채널이 시작됩니다.



이 예제에서

- IBM MQ Explorer는 클라이언트 연결을 사용하여 큐 관리자QMGRA(Server1에서 실행)에 연결됩니다.
- 이제 Server2 의 큐 관리자 QMGRB 이 (가) 중간 큐 관리자 (QMGRA) 를 통해 IBM MQ 탐색기에 연결될 수 있습니다.
- IBM MQ 탐색기를 사용하여 QMGRB 에 연결하는 경우 중간 큐 관리자로 QMGRA 를 선택하십시오.

이 상황에서는 IBM MQ 탐색기에서 QMGRB 에 대한 직접 연결이 없습니다. QMGRB 에 대한 연결은 QMGRA를 통해 수행됩니다.

서버2의 큐 관리자 QMGRB가 송신자-수신자 채널을 사용하여 서버1의 QMGRA에 연결됩니다. QMGRA와 QMGRB 사이의 채널은 원격 관리가 사용 가능한 그러한 방식으로 설정되어야 합니다. 121 페이지의 『원격 관리를 위한 채널 및 전송 큐 준비』의 내용을 참조하십시오.

큐 관리자와 클러스터 표시 및 숨기기

MQ Explorer는 한 번에 둘 이상의 큐 관리자를 표시할 수 있습니다. 큐 관리자 표시/숨기기 패널(큐 관리자 트리 노드에 대한 메뉴에서 선택 가능)에서 다른(원격) 시스템에 대한 정보를 표시할지 여부를 선택할 수 있습니다. 로컬 큐 관리자는 자동으로 감지됩니다.

리모트 큐 관리자를 표시하려면 다음을 수행하십시오.

1. **큐 관리자** 트리 노드를 마우스의 오른쪽 단추로 클릭한 후 **큐 관리자 표시/숨기기**를 선택하십시오.
2. **추가**를 클릭하십시오. 큐 관리자 표시/숨기기 패널이 표시됩니다.
3. 제공된 필드에서 리모트 큐 관리자와 호스트 이름 또는 IP 주소의 이름을 입력하십시오.

   호스트 이름 또는 IP 주소는 기본 서버 연결 채널(SYSTEM.ADMIN.SVRCONN) 또는 사용자 정의된 서버 연결 채널을 사용하여 리모트 큐 관리자에 클라이언트 연결을 설정하는 데 사용됩니다.
4. **완료**를 클릭하십시오.

큐 관리자 표시/숨기기 패널은 모든 가시적 큐 관리자의 목록을 표시합니다. 이 패널을 사용하여 탐색 보기에서 큐 관리자를 숨길 수 있습니다.

MQ Explorer가 클러스터의 멤버인 큐 관리자를 표시하는 경우, 클러스터가 감지되고 자동으로 표시됩니다.

이 패널에서 리모트 큐 관리자의 목록을 내보내려면 다음을 수행하십시오.

1. 큐 관리자 표시/숨기기 패널을 닫으십시오.
2. MQ Explorer의 도움말 탐색창에서 맨 위 **IBM MQ** 트리 노드를 마우스의 오른쪽 단추로 클릭한 후 **MQ Explorer 설정 내보내기**를 선택하십시오.
3. **MQ Explorer > MQ Explorer 설정**을 클릭하십시오.
4. **연결 정보 > 리모트 큐 관리자**를 선택하십시오.
5. 내보낸 설정을 저장하기 위한 파일을 선택하십시오.
6. 마지막으로, **마침**을 클릭하여 리모트 큐 관리자 연결 정보를 지정된 파일로 내보내십시오.

리모트 큐 관리자의 목록을 가져오려면 다음을 수행하십시오.

1. MQ Explorer의 도움말 탐색창에서 맨 위 **IBM MQ** 트리 노드를 마우스의 오른쪽 단추로 클릭한 후 **MQ Explorer 설정 가져오기**를 선택하십시오.
2. **MQ Explorer > MQ Explorer 설정**을 클릭하십시오.
3. **찾아보기**를 클릭하고 리모트 큐 관리자 연결 정보를 포함하는 파일의 경로로 이동하십시오.
4. **열기**를 클릭하십시오. 리모트 큐 관리자의 목록이 파일에 포함되는 경우 **연결 정보 > 리모트 큐 관리자** 상태가 선택됩니다.
5. 마지막으로, **완료**를 클릭하여 리모트 큐 관리자 연결 정보를 IBM MQ Explorer로 내보내십시오.

클러스터 멤버십

IBM MQ Explorer에는 클러스터의 멤버인 큐 관리자에 대한 정보가 필요합니다.

큐 관리자가 클러스터의 멤버인 경우, 클러스터 트리 노드가 자동적으로 채워집니다.

IBM MQ Explorer가 실행 중인 동안 큐 관리자가 클러스터의 멤버인 경우, 효과적으로 통신할 수 있고 요청될 때 올바른 클러스터 정보를 표시할 수 있도록 클러스터에 대한 최신 관리 데이터로 IBM MQ Explorer를 유지보수해야 합니다. 이를 수행하려면 IBM MQ Explorer에 다음 정보가 필요합니다.

- 저장소 큐 관리자의 이름
- 리모트 큐 관리자에 있는 경우 저장소 큐 관리자의 연결 이름

이 정보를 사용하여 IBM MQ Explorer는 다음을 수행할 수 있습니다.

- 저장소 큐 관리자를 사용하여 클러스터에 있는 큐 관리자 목록을 확보합니다.
- 지원되는 플랫폼 및 명령 레벨에 있는 클러스터의 멤버인 큐 관리자를 관리합니다.

다음의 경우 관리가 가능하지 않습니다.

- 선택된 저장소가 사용 불가능하게 됩니다. IBM MQ Explorer는 대체 저장소로 자동으로 전환되지 않습니다.
- 선택된 저장소가 TCP/IP를 통해 접속될 수 없습니다.
- 선택된 저장소는 IBM MQ Explorer에 의해서 지원되지 않는 플랫폼과 명령 레벨에서 실행하고 있는 큐 관리자에서 실행 중입니다.

관리될 수 있는 클러스터 멤버는 로컬일 수 있으며 TCP/IP를 사용하여 접속할 수 있는 경우에는 원격일 수 있습니다. IBM MQ Explorer는 클라이언트 연결을 사용하지 않고 직접적으로 클러스터의 멤버인 로컬 큐 관리자에 연결됩니다.

데이터 변환

IBM MQ Explorer는 CCSID 1208(UTF-8)에서 작동합니다. 이는 IBM MQ Explorer가 리모트 큐 관리자로부터 데이터를 올바르게 표시할 수 있도록 합니다. 큐 관리자에 직접적으로 연결하거나 중간 큐 관리자를 사용할지 여부에 따라, IBM MQ Explorer는 모든 수신 메시지가 CCSID 1208(UTF-8)로 변환되도록 요청합니다.

IBM MQ Explorer가 인식하지 않는 CCSID를 가지는 큐 관리자와 IBM MQ Explorer 사이에 연결을 설정하려고 노력하는 경우 오류 메시지가 실행됩니다.

지원되는 변환이 코드 페이지 변환에서 설명됩니다.

MQ Explorer 확장

IBM MQ for Windows 및 Linux x86-64 용 IBM MQ에서는 제어 또는 MQSC 명령 사용에 대한 대안으로서 관리 태스크를 수행하기 위해 MQ Explorer라고 하는 관리 인터페이스를 제공합니다.

This information applies to IBM MQ for Windows, and IBM MQ for Linux x86-64 platforms only.

MQ Explorer는 Eclipse가 지원하는 Eclipse 프레임워크와 기타 플러그인 애플리케이션의 플랫폼과 일치하는 스타일로 정보를 제공합니다.

Through extending the MQ Explorer, system administrators have the ability to customize the MQ Explorer to improve the way they administer IBM MQ.

자세한 정보는 MQ Explorer 제품 문서에서 *MQ Explorer* 확장 을 참조하십시오.

IBM MQ 작업 표시줄 애플리케이션(Windows 전용) 사용

IBM MQ 작업 표시줄 애플리케이션은 서버의Windows 시스템 트레이에 아이콘을 표시합니다. 아이콘은 일부 단순한 조치를 수행할 수 있는 메뉴 및 IBM MQ의 현재 상태를 제공합니다.

Windows에서 IBM MQ 아이콘은 서버의 시스템 트레이에 있고 색상 코드화 상태 기호로 오버레이됩니다. 이는 다음 의미 중 하나를 가질 수 있습니다.

초록색
올바르게 작업 중이며 현재 경보가 작동하지 않음

파랑색
불확정; IBM MQ이 시작 중이거나 시스템 종료 중임

노랑색
경보; 하나 이상의 서비스가 실패 중이거나 이미 실패했음

메뉴를 표시하려면 IBM MQ 아이콘을 마우스 오른쪽 단추로 클릭하십시오. 메뉴에서 다음 조치를 수행할 수 있습니다.

- **열기**를 클릭하여 IBM MQ 경보 모니터를 여십시오.
- **종료**를 클릭하여 IBM MQ 작업 표시줄 애플리케이션을 종료하십시오.
- **IBM MQ 탐색기**를 클릭하여 IBM MQ 탐색기를 시작하십시오.
- **IBM MQ 중지**를 클릭하여 IBM MQ를 중지하십시오.
- **IBM MQ 정보**를 클릭하여 IBM MQ 경보 모니터에 대한 정보를 표시하십시오.

IBM MQ 경보 모니터 애플리케이션(Windows 전용)

IBM MQ 경보 모니터는 로컬 시스템에서 IBM MQ로 문제를 식별하고 기록하는 오류 검출 도구입니다.

경보 모니터는 IBM MQ 서버의 로컬 설치의 현재 상태에 대한 정보를 표시합니다. 이는 Windows ACPI(Advanced Configuration and Power Interface)를 모니터링하고 ACPI 설정이 실시되는지 확인합니다.

IBM MQ 경보 모니터에서 다음을 수행할 수 있습니다.

- IBM MQ 탐색기에 대한 직접적인 액세스
- 모든 미해결 경로와 관련하는 정보 보기
- 로컬 시스템에서 IBM MQ 서비스 시스템 종료
- 구성 가능한 사용자 계정 또는 Windows Workstation 또는 서버로 네트워크를 통한 경보 메시지 라우팅

로컬 IBM MQ 오브젝트 관리

이 섹션은 메시지 큐 인터페이스(MQI)를 사용하는 애플리케이션 프로그램을 지원하기 위해 로컬 IBM MQ 오브젝트를 관리하는 방법에 대해 알려줍니다. 이 컨텍스트에서 로컬 관리는 IBM MQ 오브젝트 작성, 표시, 변경, 복사 및 삭제를 의미합니다.

이 섹션에서 자세히 설명된 접근 방법 뿐만 아니라 IBM MQ 탐색기를 사용하여 로컬 IBM MQ 오브젝트를 관리할 수 있습니다. 56 페이지의 『MQ Explorer를 사용하여 관리』의 내용을 참조하십시오.

이 섹션에는 다음 정보가 포함됩니다.

큐 관리자 시작 및 중지

이 주제는 큐 관리자 중지 및 시작에 대한 소개로 사용합니다.

큐 관리자 시작

큐 관리자를 시작하려면 다음과 같은 `strmqm` 명령을 사용하십시오.

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

IBM MQ for Windows 및 IBM MQ for Linux(x86 및 x86-64 플랫폼) 시스템의 경우 다음과 같이 큐 관리자를 시작할 수 있습니다.

1. IBM MQ 탐색기를 여십시오.
2. 네비게이터 보기에서 큐 관리자를 선택하십시오.
3. `Start` 을 클릭하십시오. 큐 관리자가 시작됩니다.

큐 관리자 시작을 위해 몇 초 이상이 소요되는 경우 IBM MQ는 시작 진행 상황을 나타내는 정보 메시지를 간헐적으로 실행합니다.

`strmqm` 명령은 큐 관리자가 시작되어 연결 요청을 승인할 수 있을 때까지 제어를 리턴하지 않습니다.

큐 관리자 자동 시작

IBM MQ for Windows에서 시스템은 IBM MQ 탐색기를 사용하여 시작될 때 큐 관리자를 자동으로 시작할 수 있습니다. 자세한 정보는 56 페이지의 『MQ Explorer를 사용하여 관리』의 내용을 참조하십시오.

큐 관리자 중지

endmqm 명령을 사용하여 큐 관리자를 중지하십시오.

참고: 작업 중인 큐 관리자와 연관된 설치에서 **endmqm** 명령을 사용해야 합니다. `dspmq -o installation` 명령을 사용하여 큐 관리자가 연관된 설치를 찾을 수 있습니다.

예를 들어, QMB라는 큐 관리자를 중지하려면 다음 명령을 입력하십시오.

```
endmqm QMB
```

IBM MQ for Windows 및 IBM MQ for Linux(x86 및 x86-64 플랫폼) 시스템의 경우 다음과 같이 큐 관리자를 중지할 수 있습니다.

1. IBM MQ 탐색기를 여십시오.
2. 네비게이터 보기에서 큐 관리자를 선택하십시오.
3. `Stop...` 을 클릭하십시오. 큐 관리자 종료 패널이 표시됩니다.
4. 제어됨 또는 즉시를 선택하십시오.
5. OK 을 클릭하십시오. 큐 관리자가 중지됩니다.

정상 종료(Quiesced shutdown)

기본적으로 **endmqm** 명령은 지정된 큐 관리자의 정상 종료(Quiesced shutdown)를 수행합니다. 이 명령을 완료하는 데는 약간의 시간이 걸릴 수 있습니다. 정상 종료(Quiesced shutdown)는 연결된 모든 애플리케이션이 끊어질 때까지 대기합니다.

애플리케이션이 중지하도록 알리려면 이 종료 유형을 사용하십시오. 다음을 실행하는 경우,

```
endmqm -c QMB
```

모든 애플리케이션이 중지되었을 때 이를 통보받지 않습니다. (`endmqm -c QMB` 명령은 `endmqm QMB` 명령과 동일합니다.)

그러나, 다음을 실행하는 경우,

```
endmqm -w QMB
```

명령은 모든 애플리케이션이 중지되고 큐 관리자가 종료될 때까지 대기합니다.

즉시 종료

즉시 종료의 경우, 현재 MQI 호출을 모두 완료할 수 있지만 새 호출은 실패합니다. 이 종료 유형은 애플리케이션이 큐 관리자로부터 연결이 끊어질 때까지 대기하지 않습니다.

즉시 종료의 경우, 다음을 입력하십시오.

```
endmqm -i QMB
```

강제 종료(preemptive shutdown)

참고: endmqm 명령을 사용하여 큐 관리자를 중지하려는 다른 시도가 모두 실패할 때까지 이 방법을 사용하지 마십시오. 이 방법은 연결된 애플리케이션에 대해 예측할 수 없는 결과를 초래할 수 있습니다.

즉시 종료가 작동하지 않을 경우, -p 플래그를 지정하여 강제 종료(Preemptive shutdown)를 사용해야 합니다. 예를 들면, 다음과 같습니다.

```
endmqm -p QMB
```

큐 관리자가 즉시 중지됩니다. 이 메소드가 작동하지 않는 경우, 대체 솔루션의 경우 66 페이지의 『수동으로 큐 관리자 중지』의 내용을 참조하십시오.

endmqm 명령 및 해당 옵션에 대한 세부 설명은 endmqm을 참조하십시오.

큐 관리자를 종료하는 데 문제점이 있는 경우

큐 관리자 종료 문제점은 애플리케이션에 그 원인이 있는 경우가 많습니다. 예를 들어, 애플리케이션이 다음과 같을 경우,

- MQI 리턴 코드를 올바르게 검사하지 않는 경우
- 일시정지 알림을 요청하지 않는 경우
- 큐 관리자 연결을 끊지 않고 종료하는 경우(MQDISC 호출 발행)

큐 관리자를 중지할 때 문제점이 발생하면 Ctrl-C를 사용하여 **endmqm** 명령을 중단할 수 있습니다. 그런 다음 다른 **endmqm** 명령을 실행할 수 있지만, 이번에는 필요한 종료의 유형을 지정하는 플래그로 실행할 수 있습니다.

수동으로 큐 관리자 중지

큐 관리자를 중지하기 위한 표준 메소드에 실패하는 경우, 여기에 설명된 방법을 시도하십시오.

큐 관리자를 중지하는 표준 방법은 endmqm 명령을 사용하는 것입니다. 수동으로 큐 관리자를 중지하려면 이 섹션에 설명된 프로시저 중 하나를 사용하십시오. 제어 명령을 사용하여 큐 관리자에서 조작을 수행하는 방법에 대한 자세한 내용은 분배된 플랫폼에서 큐 관리자 작성 및 관리를 참조하십시오.

*IBM MQ for Windows*에서 큐 관리자 중지

프로세스 및 IBM MQ 서비스 종료 및 IBM MQ for Windows에서 큐 관리자 중지 방법입니다.

IBM MQ for Windows에서 실행 중인 큐 관리자를 중지하려면 다음을 수행하십시오.

1. Windows 태스크 관리자를 사용하여 실행 중인 프로세스의 이름(ID)을 나열하십시오.
2. End the processes by using Windows Task Manager, or the **taskkill** command, in the following order (if they are running):

AMQZMUC0	중요한 프로세스 관리자
AMQZXMA0	실행 제어기
AMQZFUMA	OAM 프로세스
AMQZLAA0	LQM 에이전트
AMQZLSA0	LQM 에이전트
AMQZMUF0	유틸리티 관리자
AMQZMGR0	프로세스 제어기
AMQZMUR0	재시작 가능한 프로세스 관리자
AMQFQPUB	발행/구독 프로세스
AMQFCXBA	브로커 작업 프로그램 프로세스
AMQRMPPA	프로세스 풀링 프로세스
AMQCRSTA	스레드되지 않은 응답자 작업 프로세스
AMQCRS6B	LU62 수신자 채널 및 클라이언트 연결

AMQRRMFA	저장소 프로세스(클러스터의 경우)
AMQPCSEA	명령 서버
RUNMQTRM	서버의 트리거 모니터 호출
RUNMQDLQ	데드-레터 큐 핸들러 호출
RUNMQCHI	채널 시작기 프로세스
RUNMQLSR	채널 리스너 프로세스
AMQXSSVN	공유 메모리 서버

3. Windows 제어판에서 **관리 도구** > **서비스**로부터 IBM MQ 서비스를 중지하십시오.
4. 모든 방법을 시도했지만 큐 관리자가 중지되지 않으면 시스템을 다시 시작하십시오.

Windows 태스크 관리자 및 **tasklist** 명령은 태스크에 관한 제한된 정보를 제공합니다. 자세한 정보가 어느 프로세스가 특별한 큐 관리자와 관계가 있는지 판별하기 위해 돕고 https://www.microsoft.com에 있는 Microsoft 웹 사이트로부터 다운로드에 사용 가능한 프로세스 익스플로러(procexp.exe)와 같은 도구를 사용하는 것을 고려합니다.

IBM MQ for UNIX 및 *Linux* 시스템에서 큐 관리자 중지

프로세스 및 IBM MQ 서비스를 종료하고 IBM MQ for UNIX 및 Linux에서 큐 관리자를 중지하는 방법입니다. 큐 관리자를 중지하고 제거하기 위한 표준 메소드가 실패하는 경우 여기에 설명된 메소드를 시도할 수 있습니다.

IBM MQ for UNIX 및 Linux 시스템에서 실행 중인 큐 관리자를 중지하려면 다음을 수행하십시오.

1. ps 명령을 사용하여 여전히 실행 중인 큐 관리자 프로그램의 프로세스 ID를 찾으십시오. 예를 들어, 큐 관리자가 QMNAME인 경우, 다음 명령을 사용하십시오.

   ```
   ps -ef | grep QMNAME
   ```

2. 여전히 실행 중인 큐 관리자 프로세스를 종료하십시오. Use the `kill` command, specifying the process IDs discovered by using the ps command.

   프로세스를 종료하려면 **kill -KILL <pid>** 또는 동등한 **kill -9 <pid>** 명령을 사용하십시오.

   여러분은 죽이고 싶은 PID로 작업해야 하며, 하나씩, 매번 그 명령을 실행해야 한다.

   중요사항: 9(SIGKILL) 이외의 다른 신호를 사용하는 경우 프로세스가 중지되지 않고 예측할 수 없는 결과가 발생합니다.

   다음 순서로 프로세스를 종료하십시오.

amqzmuc0	중요한 프로세스 관리자
amqzxma0	실행 제어기
amqzfuma	OAM 프로세스
amqzlaa0	LQM 에이전트
amqzlsa0	LQM 에이전트
amqzmuf0	유틸리티 관리자
amqzmur0	재시작 가능한 프로세스 관리자
amqzmgr0	프로세스 제어기
amqfqpub	발행/구독 프로세스
amqfcxba	브로커 작업 프로그램 프로세스
amqrmppa	프로세스 풀링 프로세스
amqcrsta	스레드되지 않은 응답자 작업 프로세스

amqcrs6b	LU62 수신자 채널 및 클라이언트 연결
amqrrmfa	저장소 프로세스(클러스터의 경우)
amqpcsea	명령 서버
runmqtrm	서버의 트리거 모니터 호출
runmqdlq	데드-레터 큐 핸들러 호출
runmqchi	채널 시작기 프로세스
runmqlsr	채널 리스너 프로세스

참고: **kill -9** 명령을 사용하여 중지하는 데 실패한 프로세스를 종료할 수 있습니다.

큐 관리자를 수동으로 중지하면 FFST 가 사용되고 FDC 파일은 /var/mqm/errors. 에 있습니다. 이를 큐 관리자의 결함으로 간주하지 마십시오.

큐 관리자는 이 메소드를 사용하여 이를 중지한 이후에도 일반적으로 재시작됩니다.

MQI 채널 중지

서버 연결 채널에 대한 STOP CHANNEL 명령을 실행할 때, 클라이언트 연결 채널을 중지하기 위해 사용할 메소드를 선택할 수 있습니다. 이는 MQGET 대기 호출을 실행하는 클라이언트 채널을 제어할 수 있고 채널 중지 방법 및 시기를 결정할 수 있음을 의미합니다.

STOP CHANNEL 명령은 채널이 중지되는 방식을 표시하는 세 가지 모드로 실행될 수 있습니다.

일시정지
모든 현재 메시지가 처리된 후에 채널을 중지하십시오.

대화를 공유하는 것이 사용으로 설정되면, IBM MQ MQI client는 적절한 시간에 중지 요청을 알게 됩니다. 이번은 네트워크의 속도에 의존합니다. 클라이언트 애플리케이션은 IBM MQ에 대한 후속 호출을 발행한 결과로서 중지 요청을 인식하게 됩니다.

강제 실행
채널을 즉시 중지하십시오.

종료
채널을 즉시 중지하십시오. 채널이 프로세스로 실행 중이면 채널의 프로세스를 종료할 수 있습니다. 또는 채널이 스레드로 실행 중이면 해당 스레드를 종료할 수 있습니다.

이는 다단계 프로세스입니다. 모드 종료를 사용하면 처음에는 일시정지 모드로 그 다음에는 강제 실행 모드로 그리고 필요한 경우에 종료 모드로 서버 연결 채널을 중지하려고 합니다. 클라이언트는 서로 다른 종료 스테이지에서 서로 다른 리턴 코드를 수신할 수 있습니다. 프로세스 또는 스레드가 종료되면 클라이언트는 통신 오류를 수신합니다.

애플리케이션에 리턴되는 리턴 코드는 실행된 MQI 호출 및 실행된 STOP CHANNEL 명령에 따라 다릅니다. 클라이언트가 MQRC_CONNECTION_QUIESCING 또는 MQRC_CONNECTION_BROKEN 리턴 코드를 수신합니다. 클라이언트가 MQRC_CONNECTION_QUIESCING을 감지하면 현재 트랜잭션을 완료하고 종료하려고 시도해야 합니다. 이는 MQRC_CONNECTION_BROKEN으로는 가능하지 않습니다. 클라이언트가 트랜잭션을 완료하지 않고 빨리 종결하는 경우 몇 초 후에 CONNECTION_BROKEN을 수신하게 됩니다. MODE(FORCE) 또는 MODE(TERMINATE)를 사용한 STOP CHANNEL 명령은 MODE(QUIESCE)를 사용하는 경우보다 CONNECTION_BROKEN이 발생할 가능성이 높습니다.

관련 정보
MQI 채널

MQSC 명령을 사용하여 로컬 관리 태스크 수행

이 절에서는 MQSC 명령을 소개하며 일부 공용 태스크에 사용하는 방법을 설명합니다.

IBM MQ for Windows 또는 IBM MQ for Linux(x86 및 x86-64 플랫폼)를 사용하는 경우, IBM MQ 탐색기를 사용하여 이 절에서 설명되는 조작을 수행할 수도 있습니다. 자세한 정보는 56 페이지의 『MQ Explorer를 사용하여 관리』 의 내용을 참조하십시오.

큐 관리자 자체, 큐, 프로세스 정의, 채널, 클라이언트 연결 채널, 리스너, 서비스, 이름 목록, 클러스터 및 인증 정보 오브젝트를 포함하여 MQSC 명령을 사용하여 큐 관리자 오브젝트를 관리할 수 있습니다. 이 절은 큐 관리자, 큐 및 프로세스 정의를 다룹니다. 채널의 개요, 클라이언트 연결 채널 및 리스너 오브젝트의 개요의 경우, 오브젝트를 참조하십시오. 큐 관리자 오브젝트를 관리하기 위한 모든 MQSC 명령에 대한 정보는 69 페이지의 『스크립트(MQSC) 명령』의 내용을 참조하십시오.

`runmqsc` 명령을 사용하여 큐 관리자에 MQSC 명령을 실행합니다. (이 명령에 대한 자세한 내용은 runmqsc를 참조하십시오.) 키보드에서 명령을 실행하여 이를 대화식으로 수행할 수 있거나 ASCII 텍스트 파일에서 명령 순서를 실행시키기 위해 표준 입력 디바이스(`stdin`)의 경로를 재지정할 수 있습니다. 두 경우 모두, 명령의 형식이 동일합니다. (텍스트 파일에서 명령을 실행하는 것에 대한 정보는 72 페이지의 『텍스트 파일에서 MQSC 명령 실행』의 내용을 참조하십시오.)

명령에 설정되는 플래그에 따라 세 가지 방법으로 `runmqsc` 명령을 실행할 수 있습니다.

- 이를 실행하지 않고 명령을 확인하십시오. 여기서 MQSC 명령은 로컬 큐 관리자에서 확인되지만 실행되지는 않습니다.
- 로컬 큐 관리자에서 명령을 실행하십시오. 여기서 MQSC 명령은 로컬 큐 관리자에서 실행됩니다.
- 리모트 큐 관리자에서 명령을 실행하십시오. 여기서 MQSC 명령은 리모트 큐 관리자에서 실행됩니다.

구문을 표시하기 위해 물음표가 뒤에 오는 명령을 실행할 수도 있습니다.

대소문자가 구분되지 않더라도 MQSC 명령에 지정되는 오브젝트 속성이 이 절에서 대문자로 표시됩니다(예: RQMNAME). MQSC 명령 속성 이름은 8자로 제한됩니다. MQSC 명령은 IBM i 및z/OS를 포함하여 기타 플랫폼에서 사용 가능합니다.

IBM MQ 8.0부터는 MQPROMPT 환경 변수를 사용하여 사용자가 원하는 프롬프트를 설정할 수 있습니다. In addition to plain text, the MQPROMPT variable also allows environment variables to be inserted, by using +VARNAME+ `notation`, in the same manner as IBM MQ service object definitions (see 107 페이지의 『서비스 오브젝트 정의』). 예를 들면, 다음과 같습니다.

```
sh> export MQPROMPT="+USER+ @ +QMNAME+ @ +MQ_HOST_NAME+> "
sh> runmqsc MY.QMGR
5724-H72 (C) Copyright IBM Corp. 1994, 2023.
Starting MQSC for queue manager MY.QMGR.
username @ MY.QMGR @ aix1> DISPLAY QMSTATUS
```

MQSC 명령이 MQSC commands 절에서 상세히 설명됩니다.

관련 정보

runmqsc(MQSC 명령 실행)

스크립트(MQSC) 명령

MQSC 명령에서는 IBM MQ 플랫폼에서 사람이 읽을 수 있는 명령을 실행하는 균등 분포 메소드를 제공합니다. PCF(*Programmable Command Format*) 명령에 대한 정보는 9 페이지의 『프로그래밍 가능 명령 형식 소개』의 내용을 참조하십시오.

명령의 일반 형식이 MQSC 명령에 표시됩니다.

MQSC 명령을 사용할 때 다음 규칙을 준수해야 합니다.

- 각 명령은 1차 매개변수(동사)로 시작하고, 이 뒤에 2차 매개변수(명사)가 옵니다. 그런 다음 대부분의 명령에서와 같이 오브젝트가 하나 있는 경우 오브젝트의 이름 또는 일반 이름이(괄호 안에) 옵니다. 그 뒤에 매개변수가 임의의 순서로 발생할 수 있습니다. 매개변수에 해당 값이 있는 경우에는 관련된 매개변수 다음에 바로 값이 발생해야 합니다.

  참고: z/OS z/OS에서 보조 매개변수가 2차가 될 필요는 없습니다.
- 키워드, 괄호 및 값을 공백 및 쉼표 수로 분리할 수 있습니다. 구문 다이어그램에 표시된 쉼표는 항상 하나 이상의 공백으로 대체될 수 있습니다. z/OS z/OS를 제외하고 각 매개변수의 바로 앞에 하나 이상의 공백이 있어야 합니다(기본 매개변수 뒤).
- 많은 공백이 명령의 시작 또는 끝에서 발생할 수 있고 매개변수, 구두점 및 값 사이에 발생할 수 있습니다. 예를 들어, 다음 명령이 유효합니다.

```
ALTER QLOCAL ('Account' )      TRIGDPTH ( 1)
```

한 쌍의 구두점 내에 있는 공백은 중요합니다.

- 공백이 허용되는 어디에서나 추가 쉼표가 나타날 수 있으며 공백인 것처럼 취급됩니다(물론 따옴표로 묶은 문자열 안에 있지 않은 경우).
- 반복되는 매개변수는 허용되지 않습니다. REPLACE NOREPLACE에서와 같이, 해당 "NO" 버전을 가진 매개변수를 반복하도록 허용되지 않습니다.
- 다음을 제외한 특수 문자, 공백 및 소문자가 들어 있는 문자열:
  - 마침표(.)
  - 정방향 슬래시(/)
  - 밑줄(_)
  - 퍼센트 부호(%)

  아래의 경우를 제외하고는 작은 따옴표로 묶어야 합니다.
  - z/OS IBM MQ for z/OS 운영 및 제어판에서 발행됩니다.
  - 별표로 끝나는 일반 값 IBM i (IBM i에서 이는 작은따옴표로 묶여야 함)
  - 단일 별표(예: TRACE(*)) IBM i (IBM i에서는 이들을 작은따옴표로 묶어야 함)
  - 콜론을 포함하는 범위 스펙(예: CLASS(01:03))

  문자열 자체가 작은따옴표를 포함하는 경우 작은따옴표는 두 개의 작은따옴표로 표시됩니다.따옴표로 묶지 않은 소문자는 대문자로 변환됩니다.
- z/OS이외의 플랫폼에서 문자를 포함하지 않는 문자열 (즉, 사이에 공백이 없는 두 개의 작은따옴표) 은 작은따옴표로 묶인 공백으로 해석됩니다. 즉, (' ') 와 동일한 방식으로 해석됩니다. 이에 대한 예외는 사용 중인 속성이 다음 중 하나인 경우입니다.
  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR

  그런 다음, 공백이 없는 두 개의 작은따옴표가 0의 문자열 길이로 해석됩니다.

  z/OS z/OS에서, 공백을 작은따옴표로 묶으려는 경우 (' ')와 같이 입력해야 합니다. ('') 문자가 포함되지 않은 문자열은 ()을 입력하는 것과 같습니다.
- v7.0에서 SELECTOR, 하위 사용자 데이터 같이 MQCHARV 유형을 기초로 하는 문자열 속성에 있는 모든 후미 공백은 의미를 갖는 것으로 취급되므로 'abc '가 'abc'와 같지 않습니다.
- 사이에 의미있는 정보를 갖지 않고 뒤에 오른쪽 괄호가 오는 왼쪽 괄호(예:

```
NAME ( )
```

)는 특별히 지정되는 경우가 아니면 올바르지 않습니다.
- 키워드는 대소문자를 구분하지 않습니다. AltER, 대체 및 ALTER 모두 허용 가능합니다.인용부호 내에 포함되지 않은 것은 대문자로 변환됩니다.
- 동의어는 일부 매개변수에 대해 정의됩니다.예를 들어, DEF는 항상 DEFINE의 동의어이므로, DEF QLOCAL은 유효합니다. 그러나, 동의어는 최소 문자열이 아닙니다. DEFI는 DEFINE에 대한 올바른 동의어가 아닙니다.

  참고: DELETE 매개변수에 대한 동의어가 없습니다. DEFINE에 대한 동의어, DEF를 사용할 때 오브젝트가 삭제되는 읽을 방지하기 위한 것입니다.

IBM MQ 관리를 위해 MQSC 명령을 사용하는 것에 대한 개요는 68 페이지의 『MQSC 명령을 사용하여 로컬 관리 태스크 수행』 의 내용을 참조하십시오.

MQSC 명령은 특정 의미를 나타내도록 특정 특수 문자를 사용합니다. 이러한 특수 문자 및 사용 방법에 대한 자세한 정보는 특수 의미를 갖는 문자를 참조하십시오.

MQSC 명령을 사용하여 스크립트를 빌드할 수 있는 방법을 찾으려면 명령 스크립트 빌드를 참조하십시오.

z/OS 열의 기호에 대한 설명은 z/OS에서 명령 사용의 내용을 참조하십시오.

MQSC 명령의 전체 목록은 MQSC 명령을 참조하십시오.

관련 정보
명령 스크립트 빌드

IBM MQ 오브젝트 이름

MQSC 명령에서 오브젝트 이름을 사용하는 방법입니다.

예에서, 오브젝트에 일부 긴 이름을 사용합니다. 이는 처리 중인 오브젝트의 유형을 식별하는 데 도움이 됩니다.

MQSC 명령을 실행할 때, 큐의 로컬 이름만 지정해야 합니다. 다음 예에서 다음과 같은 큐 이름을 사용합니다.

```
ORANGE.LOCAL.QUEUE
```

이름의 LOCAL.QUEUE 파트는 이 큐가 로컬 큐라는 것을 설명하는 것입니다. 이는 일반적으로 로컬 큐의 이름에 필요하지 **않습니다** .

큐 관리자 이름으로 saturn.queue.manager 이름을 사용합니다. 이름의 queue.manager 파트는 이 오브젝트가 큐 관리자인지 설명하는 것입니다. 이는 일반적으로 큐 관리자의 이름에 필요하지 않습니다.

MQSC 명령에서 대소문자 구분

속성을 포함하는 MQSC 명령이 대문자 또는 소문자로 작성될 수 있습니다. MQSC 명령의 오브젝트 이름은 이름이 작은따옴표로 묶이지 않으면 대문자로 묶입니다(즉, QUEUE 및 큐가 구별되지 않음). 따옴표 표시를 사용하지 않으면, 오브젝트가 대문자로 이름을 처리합니다. 자세한 정보는 특수 의미를 가진 문자를 참고하십시오.

runmqsc 명령 호출은 모든 IBM MQ 제어 명령과 마찬가지로 일부 IBM MQ 환경에서 대소문자가 구분됩니다. 자세한 정보는 제어 명령 사용을 참조하십시오.

표준 입력 및 출력

표준 입력 디바이스(stdin으로 참조됨)는 시스템에 대한 입력이 수행되는 디바이스입니다. 일반적으로 이는 키보드이지만, 입력이 직렬 포트 또는 디스크 파일에서 발생하는 것을 지정할 수 있습니다. 예: 표준 출력 디바이스(stdout으로도 참조됨)는 시스템의 출력이 송신되는 디바이스입니다. 일반적으로 이는 표시장치이지만, 직렬 포트 또는 파일로 출력의 경로를 재지정할 수 있습니다.

운영 체제 명령 및 IBM MQ 제어 명령에서는 연산자가 입력을 재지정합니다. 이 연산자가 파일 이름 뒤에 오는 경우, 파일에서 입력이 수행됩니다. 마찬가지로, > 연산자는 출력의 경로를 재지정합니다. 이 연산자가 파일 이름 뒤에 오는 경우, 출력이 해당 파일로 보내집니다.

대화식으로 MQSC 명령 사용

명령창 또는 쉘을 사용하여 대화식으로 MQSC 명령을 사용할 수 있습니다.

대화식으로 MQSC 명령을 사용하려면, 명령 창 또는 쉘을 열고 다음을 입력하십시오.

```
runmqsc
```

이 명령에서 큐 관리자 이름이 지정되지 않으므로 기본 큐 관리자가 MQSC 명령을 처리합니다. 다른 큐 관리자를 사용하려면, **runmqsc** 명령에 큐 관리자 이름을 지정하십시오. 예를 들어, 큐 관리자 jupiter.queue.manager에서 MQSC 명령을 실행하려면 명령을 사용하십시오.

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

이후 입력하는 모든 MQSC 명령은 동일한 노드에 있으며 이미 실행 중인 것으로 가정하여 이 큐 관리자에서 처리합니다.

필요한 경우 MQSC 명령에 입력할 수 있습니다. 예를 들어, 다음을 시도하십시오.

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

매개변수가 너무 많아 한 행에 맞출 수 없는 명령의 경우, 연속 문자를 사용하여 명령이 다음 행에서 계속된다는 것을 표시하십시오.

- 빼기 부호(-)는 다음 행의 시작에서 명령이 계속된다는 것을 표시합니다.
- 더하기 부호 (+)는 다음 행의 공백이 없는 첫 번째 문자에서 명령이 계속된다는 것을 표시합니다.

명령 입력은 연속 문자가 아닌 공백이 없는 최종 문자로 종료됩니다. 세미콜론(;)을 입력하여 명령 입력을 명확하게 종료할 수도 있습니다. (이는 특히 명령 입력의 최종 행의 끝에서 연속 문자를 우연히 입력하는 경우 특히 유용합니다.)

MQSC 명령으로부터의 피드백

MQSC 명령을 실행할 때, 큐 관리자는 조치를 확인하거나 사용자가 만든 오류에 대해 알려주는 운영자 메시지를 리턴합니다. 예를 들면, 다음과 같습니다.

```
AMQ8006: IBM MQ queue created.
```

이 메시지는 큐가 작성되었다는 것을 확인합니다.

```
AMQ8405: Syntax error detected at or near end of command segment below:-

AMQ8426: Valid MQSC commands are:


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

이 메시지는 구문 오류를 작성했다는 것을 표시합니다.

이러한 메시지는 표준 출력 디바이스에 보내집니다. 명령을 올바르게 입력하지 않은 경우, 올바른 구문에 대해 MQSC 명령을 참조하십시오.

MQSC 명령의 대화식 입력 종료

MQSC 명령에 대한 작업을 중지하려면 END 명령을 입력하십시오.

그렇지 않으면, 운영 체제에 EOF 문자를 사용할 수 있습니다.

텍스트 파일에서 MQSC 명령 실행

MQSC 명령을 대화식으로 실행하는 방법은 빠른 테스트에 적합합니다. 그러나 매우 긴 명령을 사용하거나 특정한 명령 순서를 반복해서 사용하는 경우 텍스트 파일에서 `stdin`을 경로 재지정하는 방법을 고려하십시오.

71 페이지의 『표준 입력 및 출력』 에는 `stdin` 및 `stdout`에 대한 정보가 포함됩니다. 텍스트 파일에서 `stdin`의 경로를 재지정하려면 일반 텍스트 편집기를 사용하여 MQSC 명령을 포함하는 텍스트 파일을 먼저 작성하십시

오. runmqsc 명령을 사용할 때, 경로 재지정 연산자를 사용하십시오. 예를 들어, 다음 명령은 텍스트 파일 myprog.in에 포함된 명령 순서를 실행합니다.

```
runmqsc < myprog.in
```

마찬가지로, 출력을 파일로 경로를 재지정할 수도 있습니다. 입력을 위한 MQSC 명령을 포함하는 파일은 MQSC 명령 파일이라고 합니다. 큐 관리자의 응답을 포함하는 출력 파일을 출력 파일이라고 합니다.

runmqsc 명령에서 stdin 및 stdout 양식 모두의 경로를 재지정하려면 이 양식의 명령을 사용하십시오.

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

이 명령은 MQSC 명령 파일 myprog.in. 에 포함된 MQSC 명령을 호출합니다. 큐 관리자 이름을 지정하지 않았기 때문에 MQSC 명령은 디폴트 큐 관리자에 대해 실행됩니다. 출력이 텍스트 파일 myprog.out으로 전송됩니다. 73 페이지의 그림 12에서는 MQSC 명령 파일 myprog.in에 추출을 표시하고 74 페이지의 그림 13에서는 myprog.out에 출력의 해당 추출을 표시합니다.

runmqsc 명령에서 stdin 및 stdout의 경로를 재지정하려면, 기본값이 아닌 큐 관리자 (saturn.queue.manager)의 경우 이 명령의 양식을 사용하십시오.

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

MQSC 명령 파일

MQSC 명령은 ASCII 텍스트에서 즉 사람이 읽을 수 있는 양식으로 작성됩니다. 73 페이지의 그림 12은 해당 속성으로 MQSC 명령(DEFINE QLOCAL)을 표시하는 MQSC 명령 파일로부터의 추출입니다. MQSC 명령에는 각 MQSC 명령 및 해당 구문에 대한 설명이 포함됩니다.

```
.
.
.
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;
.
.
.
```

그림 *12. MQSC* 명령 파일에서 추출

IBM MQ 환경 사이의 이식성의 경우, MQSC 명령 파일의 행 길이를 72자로 제한하십시오. 더하기 부호는 명령이 다음 행에서 계속된다는 것을 표시합니다.

MQSC 명령 보고서

runmqsc 명령은 보고서를 리턴하고, 이는 stdout로 전송됩니다. 보고서에는 다음이 포함됩니다.

- 보고서의 소스로서 MQSC 명령을 식별하는 헤더:

```
Starting MQSC for queue manager jupiter.queue.manager.
```

여기서 jupiter.queue.manager는 큐 관리자의 이름입니다.

- 실행된 MQSC 명령의 번호가 매겨진 선택적 목록. 기본적으로 입력 텍스트는 출력에 표시됩니다. 이 출력 내에서 각 명령 앞에는 74 페이지의 그림 13에 표시된 대로 순서 번호가 옵니다. 그러나 출력을 억제하기 위해 runmqsc 명령에 -e 플래그를 사용할 수 있습니다.
- 오류가 있는 명령의 구문 오류 메시지.
- 각 명령의 실행 결과를 표시하는 운영자 메시지. 예를 들어, DEFINE QLOCAL 명령이 성공적으로 완료했다는 운영자 메시지는 다음과 같습니다.

```
AMQ8006: IBM MQ queue created.
```

- 스크립트 파일 실행시 일반 오류의 결과로 발생한 기타 메시지.
- 명령의 수가 읽는 것을 나타내는 보고서와 문자열 오류를 가진 명령의 수와 처리될 수 없는 명령의 수을 간결한 통계적 요약서.

참고: 큐 관리자는 구문 오류가 없는 명령만 처리하도록 시도합니다.

```
Starting MQSC for queue manager jupiter.queue.manager.
.
.
12:   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;
AMQ8006: IBM MQ queue created.
:
.
.
```

그림 *13. MQSC* 명령 보고서 파일에서 추출

제공된 MQSC 명령 파일 실행

다음 MQSC 명령 파일에는 IBM MQ가 공급됩니다.

amqscos0.tst
샘플 프로그램에서 사용한 오브젝트의 정의.

amqscic0.tst
CICS® 트랜잭션에 대한 큐의 정의.

IBM MQ for Windows에서 이러한 파일이 *MQ_INSTALLATION_PATH*\tools\mqsc\samples 디렉토리에 위치합니다. *MQ_INSTALLATION_PATH*는 IBM MQ가 설치된 상위 레벨 디렉토리를 나타냅니다.

유닉스 및 Linux 시스템에서 이러한 파일은 *MQ_INSTALLATION_PATH*/samp 디렉토리에 위치합니다. *MQ_INSTALLATION_PATH*는 IBM MQ가 설치된 상위 레벨 디렉토리를 나타냅니다.

실행하는 명령은 다음과 같습니다.

```
runmqsc < amqscos0.tst >test.out
```

runmqsc를 사용하여 명령 확인

runmqsc 명령을 사용하여 실제로 실행하지 않고 로컬 큐 관리자에서 MQSC 명령을 확인할 수 있습니다. 이를 수행하려면 runmqsc 명령에서 -v 플래그를 설정하십시오. 예:

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

MQSC 명령 파일에 대응하여 runmqsc를 호출할 때, 큐 관리자는 각 명령을 확인하고 MQSC 명령을 실제로 실행하지 않고 보고서를 리턴합니다. 이를 사용하면 명령 파일에서 명령의 구문을 검사할 수 있습니다. 다음의 경우 이는 특히 중요합니다.

- 명령 파일로부터 수많은 명령 실행.
- 여러 번 MQSC 명령 파일 사용.

리턴된 보고서는 74 페이지의 그림 13에 표시된 것과 유사합니다.

원격 조작으로 MQSC 명령을 확인하기 위해 이 방법을 사용할 수 없습니다. 예를 들어, 이 명령을 시도하면:

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

큐 관리자가 리모트인 것을 나타내려면 사용하는 -w 플래그는 무시당하고 명령이 검증 모드에서 로컬에서 실행됩니다. 30은 IBM MQ이 리모트 큐 관리자로부터 응답을 기다리는 초 수입니다.

배치 파일에서 MQSC 명령 실행

매우 긴 명령을 사용하거나 특정한 명령 순서를 반복해서 사용하는 경우 배치 파일에서 stdin을 경로 재지정하는 방법을 고려하십시오.

배치 파일에서 stdin의 경로를 재지정하려면 일반 텍스트 편집기를 사용하여 MQSC 명령을 포함하는 배치 파일을 먼저 작성하십시오. runmqsc 명령을 사용할 때, 경로 재지정 연산자를 사용하십시오. 다음 예제:

1. 테스트 큐 관리자, TESTQM을 작성합니다.
2. TCP/IP 포트 1600을 사용하기 위해 일치하는 CLNTCONN 및 리스너 세트를 작성합니다.
3. 테스트 큐, TESTQ를 작성합니다.
4. amqsputc 샘플 프로그램을 사용하여, 큐에 메시지를 넣습니다.

```
export MYTEMPQM=TESTQM
export MYPORT=1600
export MQCHLLIB=/var/mqm/qmgrs/$MQTEMPQM/@ipcc

crtmqm $MYTEMPQM
strmqm $MYTEMPQM
runmqlsr -m $MYTEMPQM -t TCP -p $MYPORT &

runmqsc $MYTEMPQM << EOF
DEFINE CHANNEL(NTLM) CHLTYPE(SVRCONN) TRPTYPE(TCP)
DEFINE CHANNEL(NTLM) CHLTYPE(CLNTCONN) QMNAME('$MYTEMPQM') CONNAME('hostname($MYPORT)')
ALTER CHANNEL(NTLM) CHLTYPE(CLNTCONN)
DEFINE QLOCAL(TESTQ)
EOF

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

그림 *14.* 배치 파일로부터 *MQSC* 명령을 실행시키기 위한 예제 스크립트

MQSC 명령으로 문제점 해결

실행할 MQSC 명령을 가져올 수 없는 경우, 이러한 공통 문제점이 사용자에게 적용되는지 여부를 보기 위해 이 주제에 있는 정보를 사용하십시오. 명령에서 생성되는 오류를 읽을 때 문제점이 무엇인지 항상 명확하게 알 수 있는 것은 아닙니다.

runmqsc 명령을 사용할 때, 다음을 기억하십시오.

- **<** 연산자를 사용하여 파일에서 입력을 재지정합니다. 이 연산자를 생략하는 경우, 큐 관리자는 큐 관리자 이름과 같은 파일 이름을 해석하고 다음 오류 메시지를 발행합니다.

```
AMQ8118: IBM MQ queue manager does not exist.
```

- 파일로 출력을 경로 재지정하는 경우, **>** 경로 재지정 연산자를 사용하십시오. 기본적으로 runmqsc가 호출될 때 파일은 현재 작업 디렉토리에 배치됩니다. 사용자의 출력을 특정 파일과 디렉토리에 보내기 위해 충분한 자격이 있는 파일 이름을 지정하십시오.
- 모든 큐 관리자를 표시하기 위해 다음 명령을 사용하여 명령을 실행하려는 큐 관리자를 작성했는지 검사하십시오.

```
dspmq
```

- 큐 관리자가 실행 중이어야 합니다. 그렇지 않은 경우, 이를 시작하십시오(큐 관리자 시작 참조). 이미 실행 중인 큐 관리자를 시작하려고 시도하는 경우 오류 메시지를 가져옵니다.
- 기본 큐 관리자를 정의하지 않거나 이 오류를 가져오는 경우 runmqsc 명령에서 큐 관리자 이름을 지정하십시오.

```
AMQ8146: IBM MQ queue manager not available.
```

- runmqsc 명령의 매개변수로서 MQSC 명령을 지정할 수 없습니다. 예를 들어, 이는 올바르지 않습니다.

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

- runmqsc 명령을 실행하기 전에 MQSC 명령을 입력할 수 없습니다.
- runmqsc 에서 제어 명령을 실행할 수 없습니다. 예를 들어, MQSC 명령을 대화식으로 실행 중인 동안 큐 관리자를 시작하기 위한 strmqm 명령을 실행할 수 없습니다. 이를 수행하려면, 다음과 유사한 오류 메시지를 수신합니다.

```
runmqsc
.
.
Starting MQSC for queue manager jupiter.queue.manager.

1 : strmqm saturn.queue.manager
AMQ8405: Syntax error detected at or near end of cmd segment below:-s

AMQ8426: Valid MQSC commands are:
ALTER
CLEAR
DEFINE
DELETE
DISPLAY
END
PING
REFRESH
RESET
RESOLVE
RESUME
START
STOP
SUSPEND
2 : end
```

큐 관리자에 대한 작업

큐 관리자 속성을 표시하거나 대체하기 위해 사용할 수 있는 MQSC 명령의 예입니다.

큐 관리자 속성 표시

runmqsc 명령에 지정된 큐 관리자의 속성을 표시하려면 다음 MQSC 명령을 사용하십시오.

```
DISPLAY QMGR
```

이 명령에서 일반 설치 출력이 77 페이지의 그림 15에 표시됩니다

```
DISPLAY QMGR
     1 : DISPLAY QMGR
AMQ8408: Display Queue Manager details.
   QMNAME(QM1)                             ACCTCONO(DISABLED)
   ACCTINT(1800)                           ACCTMQI(OFF)
   ACCTQ(OFF)                              ACTIVREC(MSG)
   ACTVCONO (DISABLED)                     ACTVTRC (OFF)
   ALTDATE(2012-05-27)                     ALTTIME(16.14.01)
   AUTHOREV(DISABLED)                      CCSID(850)
   CHAD(DISABLED)                          CHADEV(DISABLED)
   CHADEXIT( )                             CHLEV(DISABLED)
   CLWLDATA( )                             CLWLEXIT( )
   CLWLLEN(100)                            CLWLMRUC(999999999)
   CLWLUSEQ(LOCAL)                         CMDEV(DISABLED)
   CMDLEVEL(800)                           COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
   CONFIGEV(DISABLED)                      CRDATE(2011-05-27)
   CRTIME(16.14.01)                        DEADQ()
   DEFXMITQ( )                             DESCR( )
   DISTL(YES)                              INHIBTEV(DISABLED)
   IPADDRV(IPV4)                           LOCALEV(DISABLED)
   LOGGEREV(DISABLED)                      MARKINT(5000)
   MAXHANDS(256)                           MAXMSGL(4194304)
   MAXPROPL(NOLIMIT)                       MAXPRTY(9)
   MAXUMSGS(10000)                         MONACLS(QMGR)
   MONCHL(OFF)                             MONQ(OFF)
   PARENT( )                               PERFMEV(DISABLED)
   PLATFORM(WINDOWSNT)                        PSRTYCNT(5)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSSYNCPT(IFPER)                         QMID(QM1_2011-05-27_16.14.01)
   PSMODE(ENABLED)                         REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   ROUTEREC(MSG)                           SCHINIT(QMGR)
   SCMDSERV(QMGR)                          SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             SSLKEYR(C:\Program Files\IBM\WebSphere
MQ\Data\qmgrs\QM1\ssl\key)
   SSLRKEYC(0)                             STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SYNCPT
   TREELIFE(1800)                          TRIGINT(999999999)
```

그림 *15. DISPLAY QMGR* 명령에서 일반 설치 출력

참고: SYNCPT는 읽기 전용 큐 관리자 속성입니다.

ALL 매개변수는 DISPLAY QMGR 명령에 대한 기본값입니다. 이는 모든 큐 관리자 속성을 표시합니다. 특히, 출력은 기본 큐 관리자 이름, 데드-레터 큐 이름 및 명령 큐 이름을 알려줍니다.

이러한 큐가 명령을 입력하여 존재하는지 확인할 수 있습니다.

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

스템 SYSTEM.*과 일치하는 큐의 목록이 표시됩니다. 괄호가 필요합니다.

큐 관리자 속성 대체

runmqsc 명령에서 지정된 큐 관리자의 속성을 대체하려면 변경하려는 속성 및 값을 지정하는 MQSC 명령 ALTER QMGR을 사용하십시오. 예를 들어, 다음 명령을 사용하여 jupiter.queue.manager의 속성을 대체하십시오.

```
runmqsc jupiter.queue.manager
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

ALTER QMGR 명령은 사용된 데드-레터 큐를 변경하고 이벤트가 억제되도록 합니다.

관련 정보
큐 관리자의 속성

로컬 큐에 대한 작업

이 절에는 로컬, 모델 및 알리어스 큐를 관리하기 위해 사용할 수 있는 MQSC 명령 예가 포함됩니다.

이러한 명령에 대한 자세한 정보는 MQSC 명령을 참조하십시오.

로컬 큐 정의

애플리케이션의 경우, 로컬 큐 관리자는 애플리케이션이 연결되는 큐 관리자입니다. 로컬 큐 관리자에 의해 관리되는 큐는 해당 큐 관리자에 대해 로컬이라고 합니다.

MQSC 명령 DEFINE QLOCAL을 사용하여 로컬 큐를 작성하십시오. 또한 기본 로컬 큐 정의에 정의된 기본값을 사용하거나 기본 로컬 큐 정의에서 큐 특성을 수정할 수 있습니다.

참고: 기본 로컬 큐는 SYSTEM.DEFAULT.LOCAL.QUEUE로 이름 지정되고 시스템 설치에서 작성됩니다.

예를 들어, 뒤따르는 DEFINE QLOCAL 명령은 ORANGE.LOCAL.QUEUE라는 큐를 특성으로 정의합니다.

- 가져오기(GET)에 사용되고, 넣기(PUT)에 사용되고 우선순위 순서 기본에서 실행됩니다.
- 이는 정상 큐입니다. 이는 이니시에이션 큐 또는 전송 큐가 아니고 트리거 메시지를 생성하지 않습니다.
- 최대 큐 용량은 5000개의 메시지입니다. 최대 메시지 길이는 4194304 바이트입니다.

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
DESCR('Queue for messages from other systems') +
PUT (ENABLED) +
GET (ENABLED) +
NOTRIGGER +
MSGDLVSQ (PRIORITY) +
MAXDEPTH (5000) +
MAXMSGL (4194304) +
USAGE (NORMAL);
```

참고:

1. 설명의 값을 제외하고는, 표시된 모든 속성 값은 기본값입니다. 여기에는 단지 예시 목적으로 표시하였습니다. 기본값이 원하는 것이거나 변경되지 않는 것이라고 확인하는 경우 생략할 수 있습니다. 78 페이지의 『기본 오브젝트 속성 표시』의 내용을 참조하십시오.
2. USAGE(NORMAL)는 이 큐가 전송 큐가 아니라는 것을 표시합니다.
3. 동일한 큐 관리자에 ORANGE.LOCAL.QUEUE라는 이름의 로컬 큐가 이미 있는 경우에는 이 명령이 실패합니다. 기존 큐 정의를 겹쳐쓰려면 REPLACE 속성을 사용하십시오. 79 페이지의 『로컬 큐 속성 변경』의 내용을 참조하십시오.

기본 오브젝트 속성 표시

DISPLAY QUEUE 명령을 사용하여 IBM MQ 오브젝트가 정의될 때 기본 오브젝트에서 가져온 속성을 표시할 수 있습니다.

IBM MQ 오브젝트를 정의할 때, 기본 오브젝트에서 지정하지 않는 속성을 사용합니다. 예를 들어, 로컬 큐를 정의할 때, 큐는 정의에서 생략하는 속성을 기본 로컬 큐에서 상속하고, 이를 SYSTEM.DEFAULT.LOCAL.QUEUE라고 합니다. 이러한 속성의 개념을 정확하게 보려면 다음 명령을 참조하십시오.

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

이 명령 구문은 해당 DEFINE 명령의 구문과 다릅니다. DISPLAY 명령에서 큐 이름만 제공할 수 있습니다. 반면, DEFINE 명령에서 큐의 유형을 지정해야 합니다(즉, QLOCAL, QALIAS, QMODEL 또는 QREMOTE).

개별적으로 지정하여 선택적으로 속성을 표시할 수 있습니다. 예를 들면, 다음과 같습니다.

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

이 명령은 다음과 같이 세 개의 지정된 속성을 표시합니다.

```
AMQ8409: Display Queue details.
QUEUE(ORANGE.LOCAL.QUEUE)        TYPE(QLOCAL)
CURDEPTH(0)                MAXDEPTH(5000)
MAXMSGL(4194304)
```

CURDEPTH는 현재 큐 용량입니다(즉, 큐의 메시지 수). 이는 큐 용량을 모니터링하여 큐가 채워지지 않는다는 것을 확인할 수 있으므로 표시할 유용한 속성입니다.

로컬 큐 정의 복사

DEFINE 명령에 LIKE 속성을 사용하여 큐 정의를 복사할 수 있습니다.

예를 들면, 다음과 같습니다.

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

이 명령은 시스템 기본 로컬 큐의 속성보다 오히려 최초 큐 ORANGE.LOCAL.QUEUE와 동일한 속성을 작성합니다. 큐를 작성했을 때 입력된 것처럼 **정확히** 복사될 큐의 이름을 입력하십시오. 이름이 소문자를 포함하면, 이름을 작은따옴표로 묶으십시오.

DEFINE 명령의 양식을 사용하여 큐 정의를 복사할 수 있지만, 원래 속성에 대한 하나 이상의 변경사항을 대체할 수 있습니다. 예를 들면, 다음과 같습니다.

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

이 명령은 큐 ORANGE.LOCAL.QUEUE의 속성을 큐 THIRD.QUEUE에 복사하지만, 새 큐에 대한 최대 메시지 길이가 4194304보다 1024바이트가 될 수 있도록 지정합니다.

참고:

1. DEFINE 명령에서 LIKE 속성을 사용할 때, 큐 속성만 복사합니다. 큐에서 메시지를 복사하지 않습니다.
2. LIKE를 지정하지 않고 로컬 큐를 정의하는 경우, DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE)와 동일합니다.

로컬 큐 속성 변경

REPLACE 속성으로 ALTER QLOCAL 명령 또는 DEFINE QLOCAL 명령을 사용하는 두 가지 방법으로 큐 속성을 변경할 수 있습니다.

78 페이지의 『로컬 큐 정의』에서 ORANGE.LOCAL.QUEUE라는 큐가 정의됩니다. 예를 들어, 이 큐의 최대 메시지 길이를 10,000바이트로 줄이려고 한다고 가정하십시오.

- ALTER 명령 사용:

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

  이 명령은 단일 속성, 즉 최대 메시지 길이의 속성을 변경합니다. 기타 모든 속성이 동일하게 유지됩니다.

- REPLACE 옵션을 DEFINE 명열 사용. 예:

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

  이 명령은 최대 메시지 길이를 변경할 뿐만 아니라, 기본값으로 제공된 기타 모든 속성들도 변경합니다. 이전에는 큐를 넣을(put) 수 없었지만, 이제는 넣을(put) 수 있습니다. SYSTEM.DEFAULT.LOCAL.QUEUE 큐에서 지정하는 대로 넣기 가능이 기본값입니다.

  기존 큐의 최대 메시지 길이를 **줄일 경우** 기존 메시지는 영향을 받지 않습니다. 그러나, 새 메시지는 새 기준을 충족해야 합니다.

로컬 큐 지우기

CLEAR 명령을 사용하여 로컬 큐를 지울 수 있습니다.

MAGENTA.QUEUE라는 로컬 큐에서 모든 메시지를 삭제하려면 다음 명령을 사용하십시오.

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

참고: 이러한 선택을 변경할 수 있게 하는 어떤 프롬프트도 표시되지 않습니다. Enter 키를 누르면 메시지가 유실됩니다.

다음의 경우 큐를 지울 수 없습니다.

- 동기점 아래에 큐에 걸린 미확약된 메시지가 있습니다.
- 애플리케이션이 현재 큐를 열어 두었습니다.

로컬 큐 삭제

MQSC 명령 DELETE QLOCAL을 사용하여 로컬 큐를 삭제할 수 있습니다.

큐에 미확약된 메시지가 있는 경우 큐는 삭제될 수 없습니다. 그러나, 큐에 하나 이상의 확약된 메시지가 있고 미확약된 메시지가 없는 경우, PURGE 옵션을 지정하는 경우에만 삭제될 수 있습니다. 예를 들면, 다음과 같습니다.

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

PURGE 대신에 NOPURGE를 지정하면 확약된 메시지가 포함된 경우 큐가 삭제되지 않도록 합니다.

큐 찾아보기

IBM MQ에서는 큐에서 메시지의 컨텐츠를 찾아보기 위해 사용할 수 있는 샘플 큐 브라우저를 제공합니다. 브라우저가 소스 및 실행 가능 형식 모두로 제공됩니다.

*MQ_INSTALLATION_PATH*는 IBM MQ가 설치된 상위 레벨 디렉토리를 나타냅니다.

IBM MQ for Windows에서 샘플 큐 브라우저에 대한 파일 이름 및 경로는 다음과 같습니다.

소스
MQ_INSTALLATION_PATH\tools\c\samples\

실행 파일
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

IBM MQ for UNIX 및 Linux에서 파일 이름 및 경로는 다음과 같습니다.

소스
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

실행 파일
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

샘플에는 두 개의 입력 매개변수, 큐 이름 및 큐 관리자 이름이 필요합니다. 예를 들면, 다음과 같습니다.

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

이 명령으로부터의 일반 설치 결과가 81 페이지의 그림 16에 표시됩니다.

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 850
  Format : 'MQEVENT '
  Priority : 0  Persistence : 0
  MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                    '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '104'

****  Message   ****

length - 104 bytes

00000000: 0700 0000 2400 0000 0100 0000 2C00 0000 '....→.......,...'
00000010: 0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020: 0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030: 0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040: 7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager    '
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060: 2020 2020 2020 2020                     '                '

No more messages
MQCLOSE
MQDISC
```

그림 *16.* 큐 브라우저로부터의 일반 설치 결과

관련 정보
브라우저 샘플 프로그램

큰 큐를 사용으로 설정

IBM MQ에서는 2GB보다 큰 큐를 지원합니다.

Windows 시스템에서 추가 실행 가능성 없이 큰 파일에 대한 지원을 사용할 수 있습니다. AIX, HP-UX, Linux및 Solaris 시스템에서 2GB 보다 더 큰 큐 파일을 작성할 수 있으려면 명확하게 큰 파일 지원을 사용 가능하게 설정해야 합니다. 이를 수행하는 방법에 대한 정보는 운영 체제 문서를 참조하십시오.

일부 유틸리티(예: tar)는 2GB 보다 큰 파일을 대체할 수 없습니다. 큰 파일을 지원할 수 있으려면 사용하는 유틸리티의 제한사항에 대한 정보에 대해 운영 체제 문서를 확인하십시오.

큐에 필요한 스토리지의 양을 계획하는 데 대한 정보는 IBM MQ 웹 사이트에서 플랫폼별 성능 보고서를 참조하십시오. `https://www.ibm.com/support/docview.wss?rs=171&uid=swg27007150&loc=en_US&cs=utf-8&lang=en#1`

알리어스 큐에 대한 작업

다른 큐 또는 토픽을 간접적으로 참조하기 위한 알리어스 큐를 정의할 수 있습니다.

V 8.0.0.6

주의: 분배 목록에서는 토픽 오브젝트를 가리키는 알리어스 큐 사용을 지원하지 않습니다. 버전 8.0.0, 수정팩 6부터는 알리어스 큐가 분배 목록의 토픽 오브젝트를 가리키는 경우 IBM MQ는 MQRC_ALIAS_BASE_Q_TYPE_ERROR를 리턴합니다.

알리어스 큐가 참조하는 큐는 다음 중 하나일 수 있습니다.

- 로컬 큐(78 페이지의 『로컬 큐 정의』 참조).
- 리모트 큐의 로컬 정의(125 페이지의 『리모트 큐의 로컬 정의 작성』 참조).
- 토픽.

알리어스 큐는 실제 큐는 아니지만, 런타임 시 실제(또는 대상) 큐로 해석되는 정의입니다. 알리어스 큐 정의는 대상 큐를 지정합니다. 애플리케이션이 알리어스 큐에 대해 MQOPEN 호출을 할 때, 큐 관리자는 대상 큐 이름에 대한 알리어스를 해석합니다.

알리어스 큐는 로컬로 정의된 알리어스 큐로 해석될 수 없습니다. 그러나, 알리어스 큐는 로컬 큐 관리자가 멤버인 클러스터의 어딘가에서 정의되는 알리어스 큐로 해석될 수 있습니다. 추가 정보는 이름 해석을 참조하십시오.

알리어스 큐는 다음에 유용합니다.

- 다른 애플리케이션에 대상 큐에 대한 다른 레벨의 액세스 권한 제공.
- 다른 애플리케이션에서 다른 방법으로 동일한 큐에 대해 작업할 수 있도록 허용. (아마도 다른 기본 특성 또는 다른 기본 지속 값을 지정하려고 합니다.)
- 유지보수, 마이그레이션 및 워크로드 밸런싱 단순화. (아마도 애플리케이션을 변경하지 않고 대상 큐 이름을 변경하려고 합니다. 이는 알리어스를 계속 사용합니다.)

예를 들어, 애플리케이션이 MY.ALIAS.QUEUE라는 큐에 메시지를 배치하기 위해 개발된다고 가정하십시오. MQOPEN 요청을 작성할 때와 간접적으로 이 큐에 메시지를 배치하는 경우 이 큐의 이름을 지정합니다. 애플리케이션은 큐가 알리어스 큐라는 것을 인지하지 않습니다. 이 알리어스를 사용하는 각 MQI 호출의 경우, 큐 관리자는 실제 큐 이름을 해석합니다. 이는 이 큐 관리자에 정의되는 로컬 큐 또는 리모트 큐가 될 수 있습니다.

TARGQ 속성의 값을 변경하여, 다른 큐 관리자에서 사용 가능한 다른 큐로 MQI 호출의 경로를 재지정할 수 있습니다. 이는 유지보수, 마이그레이션 및 로드 밸런싱에 유용합니다.

알리어스 큐 정의

다음 명령은 알리어스 큐를 작성합니다.

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

이 명령은 MY.ALIAS.QUEUE를 큐 YELLOW.QUEUE로 지정하는 MQI 호출의 경로를 재지정합니다. 명령은 대상 큐를 작성하지 않습니다. MQI 호출은 YELLOW.QUEUE가 런타임 시 존재하지 않는 경우 실패합니다.

알리어스 정의를 변경하는 경우, 다른 큐로 MQI 호출의 경로를 재지정할 수 있습니다. 예를 들면, 다음과 같습니다.

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

이 명령은 MQI 호출의 경로를 다른 큐 MAGENTA.QUEUE로 재지정합니다.

알리어스 큐를 사용하여 단일 큐(대상 큐)가 다른 애플리케이션에 대해 다른 속성을 갖는 것처럼 보이도록 할 수 있습니다. 각 애플리케이션에 하나씩 두 개의 알리어스를 정의하여 이를 수행합니다. 두 개의 애플리케이션이 있다고 가정하십시오.

- 애플리케이션 ALPHA는 YELLOW.QUEUE에 메시지를 배치할 수 있지만, 여기에서 메시지를 가져올 수 없습니다.
- 애플리케이션 BETA는 YELLOW.QUEUE에서 메시지를 가져올 수 있지만, 여기에 메시지를 배치할 수 없습니다.

다음 명령은 애플리케이션 ALPHA에 사용 불가능한 get 및 사용 가능한 put인 알리어스를 정의합니다.

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

다음 명령은 애플리케이션 BETA에 사용 가능한 get 및 사용 불가능한 put인 알리어스를 정의합니다.

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

ALPHA는 MQI 호출에서 큐 이름 ALPHAS.ALIAS.QUEUE를 사용합니다. BETA는 큐 이름 BETAS.ALIAS.QUEUE를 사용합니다. 이는 모두 동일한 큐에 액세스하지만, 다른 방법으로 액세스합니다.

로컬 큐와 함께 해당 속성을 사용하는 동일한 방법으로 큐 알리어스를 정의할 때 LIKE 및 REPLACE 속성을 사용할 수 있습니다.

알리어스 큐로 기타 명령 사용

적절한 MQSC 명령을 사용하여 알리어스 큐 속성을 표시 또는 대체하거나 알리어스 큐 오브젝트를 삭제할 수 있습니다. 예를 들면, 다음과 같습니다.

다음 명령을 사용하여 알리어스 큐의 속성을 표시하십시오.

```
DISPLAY QUEUE (ALPHAS.ALIAS.QUEUE)
```

다음 명령을 사용하여 알리어스가 해석하는 기본 큐 이름을 대체하십시오. 여기서, `force` 옵션은 큐가 열릴지라도 변경을 강제 실행합니다.

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

다음 명령을 사용하여 이 큐 알리어스를 삭제하십시오.

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

애플리케이션에서 현재 큐가 열려 있는 경우 알리어스 큐를 삭제할 수 없습니다. 이 알리어스 큐 명령 및 다른 것에 대한 자세한 정보는 MQSC 명령을 참조하십시오.

데드-레터 큐에 대한 작업

올바른 대상으로 전달될 수 없는 해당 메시지가 이후 검색을 위해 저장될 수 있도록 각 큐 관리자에는 일반적으로 데드-레터 큐로서 사용할 로컬 큐가 있습니다. 데드-레터 큐에 대한 큐 관리자에 대해 알려주고 데드-레터 큐에서 발견된 메시지가 처리되는 방법을 지정합니다. 데드-레터 큐를 사용하는 것은 메시지가 전달되는 순서에 영향을 줄 수 있습니다. 따라서 이를 사용하지 않도록 선택할 수 있습니다.

데드-레터 큐에 대한 큐 관리자를 알려주려면 `crtmqm` 명령(예: `crtmqm -u DEAD.LETTER.QUEUE`)에서 또는 나중에 하나를 지정하기 위해 ALTER QMGR 명령에서 DEADQ 속성을 사용하여 데드-레터 큐 이름을 지정하십시오. 이를 사용하기 전에 데드-레터 큐를 정의해야 합니다.

SYSTEM.DEAD.LETTER.QUEUE라는 샘플 데드-레터 큐는 제품과 함께 사용할 수 있습니다. 이 큐는 큐 관리자를 작성할 때 자동으로 작성됩니다. 필요한 경우 이 정의를 수정하고 이름을 바꿀 수 있습니다.

데드-레터 큐에는 다음을 제외하고 특별한 요구사항이 없습니다.

- 이는 로컬 큐여야 합니다
- MAXMSGL(최대 메시지 길이) 속성은 큐가 데드-레터 헤더(MQDLH)의 크기 **뿐만 아니라** 큐 관리자가 핸들링해야 하는 가장 큰 메시지를 수용할 수 있도록 할 수 있어야 합니다

데드-레터 큐를 사용하는 것은 메시지가 전달되는 순서에 영향을 줄 수 있습니다. 따라서 이를 사용하지 않도록 선택할 수 있습니다. 메시지가 전달될 수 없을 때 데드-레터 큐가 사용되는지 여부를 판별하기 위해 USEDLQ 채널 속성을 설정합니다. 이 속성은 기타 기능이 수행되지 않는 동안 큐 관리자의 일부 기능이 데드-레터 큐를 사용할 수 있도록 구성될 수 있습니다. 다른 MQSC 명령에서USEDLQ 채널 속성 사용에 대한 자세한 정보는 DEFINE CHANNEL, DISPLAY CHANNEL, ALTER CHANNEL 및 DISPLAY CLUSQMGR을 참조하십시오.

IBM MQ에서는 데드-레터 큐에서 발견되는 메시지가 처리되거나 제거되는 방법을 지정할 수 있도록 데드-레터 큐 핸들러를 제공합니다. 84 페이지의 『데드-레터 큐에서 메시지 처리』의 내용을 참조하십시오.

관련 정보

데드-레터 큐

미배달 메시지 문제점 해결

데드-레터 큐에서 메시지 처리

데드-레터 큐(DLQ)에 대한 메시지를 처리하려면, MQ는 기본 DLQ 핸들러를 공급합니다. 핸들러는 정의하는 규칙 테이블의 입력 항목에 대응하는 DLQ에 대한 메시지와 일치합니다.

큐 관리자, 메시지 채널 에이전트(MCA) 및 애플리케이션에 의해 DLQ에 메시지가 기록될 수 있습니다. DLQ에 대한 모든 메시지는 데드 레터 헤더 구조, MQDLH를 접두부로 사용합니다. 큐 관리자 또는 메시지 채널 에이전트에 의해 DLQ에 넣어지는 메시지에는 항상 다음 헤더가 있습니다. DLQ에 메시지를 넣는 애플리케이션은 이 헤더를 제공해야 합니다. MQDLH 구조의 *Reason* 필드에는 메시지가 DLQ에 있는 이유를 식별하는 이유 코드가 포함됩니다.

모든 IBM MQ 환경에는 정기적으로 DLQ에서 메시지를 처리하기 위한 루틴이 필요합니다. IBM MQ 는 `runmqdlq` 명령을 사용하여 호출하는 데드-레터 큐 핸들러 (DLQ 핸들러) 라는 기본 루틴을 제공합니다.

DLQ에 있는 메시지를 처리하기 위한 지시사항은 사용자 작성 규칙 테이블 방식으로 DLQ 핸들러에 제공됩니다. 즉, DLQ 핸들러는 규칙 테이블의 입력 항목에 대응하여 DLQ에 있는 메시지와 일치합니다. DLQ 메시지가 규칙 테이블에 있는 입력 항목과 일치하면 DLQ 핸들러가 해당 입력 항목과 연관된 조치를 수행합니다.

관련 정보

데드-레터 큐

미배달 메시지 문제점 해결

IBM MQ for IBM i 데드-레터 큐 핸들러

이 정보를 사용하여 데드-레터 큐 핸들러 및 이를 호출하는 방법에 대해 학습하십시오.

데드-레터 큐(DLQ)(때때로 미전달 메시지 큐로 불림)는 목적지 큐로 전달되지 못한 메시지에 대한 큐를 보유합니다. 네트워크 내의 모든 큐 관리자는 연관된 DLQ가 있어야 합니다.

참고: 대부분 메시지를 DLQ에 넣는 것을 피하는 것을 선호합니다. DLQ의 사용과 회피에 대한 정보는 84 페이지의 『데드-레터 큐에 대한 작업』의 내용을 참조하십시오.

큐 관리자, 메시지 채널 에이전트 및 애플리케이션은 DLQ에 메시지를 넣을 수 있습니다. DLQ에 대한 모든 메시지는 데드 레터 헤더 구조, MQDLH를 접두부로 사용합니다. 메시지는 항상 MQDLH를 가지는 메시지 채널 채널 에이전트 또는 큐 관리자에 의해 DLQ에 넣어집니다. DLQ에 메시지를 넣는 애플리케이션에 항상 MQDLH를 제공하십시오. MQDLH 구조의 *Reason* 필드에는 메시지가 DLQ에 있는 이유를 식별하는 이유 코드가 포함됩니다.

모든 IBM MQ 환경에서, DLQ에 대한 메시지를 처리하기 위해 정기적으로 실행하는 루틴이 있어야 합니다. IBM MQ에서는 STRMQMDLQ 명령을 사용하여 호출하는 데드-레터 큐 핸들러(DLQ 핸들러)라는 기본 루틴을 제공합니다. 사용자 작성 규칙 테이블은 DLQ의 메시지를 처리하도록 DLQ 핸들러에 지시사항을 제공합니다. 즉, DLQ 핸들러는 DLQ의 메시지가 규칙 테이블의 입력 항목과 일치하는지 확인합니다. DLQ 메시지가 규칙 테이블의 입력 항목과 일치하면 DLQ 핸들러가 해당 입력 항목과 연관된 조치를 수행합니다.

DLQ 핸들러 호출

STRMQMDLQ 명령을 사용하여 DLQ 핸들러를 호출하십시오. 처리하려는 DLQ 및 사용하려는 큐 관리자의 이름을 두 가지 방식으로 지정할 수 있습니다.

- 명령 프롬프트에서 STRMQMDLQ에 대한 매개변수로 지정. 예를 들면, 다음과 같습니다.

```
STRMQMDLQ UDLMSGQ(ABC1.DEAD.LETTER.QUEUE) SRCMBR(QRULE) SRCFILE(library/QTXTSRC)
MQMNAME(MY.QUEUE.MANAGER)
```

- 규칙 테이블에서. 예를 들면, 다음과 같습니다.

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE)
```

참고: 규칙 테이블은 임의의 이름을 사용할 수 있는 소스 실제 파일 내의 멤버입니다.

이 예는 기본 큐 관리자가 소유한 ABC1.DEAD.LETTER.QUEUE라는 DLQ에 적용됩니다.

표시된 대로 DLQ 또는 큐 관리자를 지정하지 않으면, 설치를 위한 기본 큐 관리자가 해당 큐 관리자에 속하는 DLQ와 함께 사용됩니다.

STRMQMDLQ 명령은 규칙 테이블에서 해당 입력을 가져옵니다.

DLQ 핸들러를 실행하기 위해서는 DLQ 자체 및 DLQ에 있는 메시지가 전달되는 모든 메시지 큐에 대해 권한이 있어야 합니다. 또한 메시지 컨텍스트에서 사용자 ID의 권한을 사용하여 큐에 메시지를 넣기 위해 DLQ에 대한 기타 사용자의 ID를 가정하기 위해서도 권한이 있어야 합니다.

관련 정보

데드-레터 큐

미배달 메시지 문제점 해결

DLQ 핸들러 규칙 테이블

DLQ 핸들러 규칙 테이블은 DLQ 핸들러가 DLQ에 도착하는 메시지를 처리하는 방법을 정의합니다

DLQ 핸들러 규칙 테이블은 DLQ 핸들러가 DLQ에 도착한 메시지를 처리하는 방법을 정의합니다. 규칙 테이블에는 두 가지 유형의 입력 항목이 있습니다.

- 표의 첫 번째 항목(선택적)에는 제어 데이터가 포함됩니다.
- 테이블의 기타 모든 입력 항목은 DLQ 핸들러가 따라야 할 규칙입니다. 각 규칙은 메시지가 일치되는 패턴(메시지 특성 세트)과, DLQ의 메시지가 지정된 패턴과 일치할 때 수행되는 조치로 구성됩니다. 규칙 테이블에는 최소 하나의 규칙이 있어야 합니다.

규칙 테이블의 각 입력 항목은 하나 이상의 키워드로 구성됩니다.

제어 데이터

이 절에서는 DLQ 핸들러 규칙 테이블의 제어 데이터 입력 항목에 포함시킬 수 있는 키워드에 대해 설명합니다. 다음에 유의하십시오.

- 키워드가 있는 경우 기본값은 밑줄 그어져 있습니다.
- 세로 선(|)은 대안을 분리합니다. 다음 중 하나만 지정할 수 있습니다.

• 모든 키워드는 선택사항입니다.

INPUTQ (*QueueName* |' ')
처리하려는 DLQ의 이름입니다.

1. 매개변수로 **STRMQMDLQ** 명령에 지정하는 UDLMSGQ 값(또는 *DFT)은 규칙 테이블에서 INPUTQ 값을 대체합니다.
2. 매개변수로 **STRMQMDLQ** 명령에 공백 UDLMSGQ 값을 지정하는 경우, 규칙 테이블의 INPUTQ 값이 사용됩니다.
3. 매개변수로 **STRMQMDLQ** 명령에 공백 UDLMSGQ 값을 지정하고 규칙 테이블에 공백 INPUTQ 값을 지정하는 경우, 시스템 기본 데드-레터 큐가 사용됩니다.

INPUTQM (*QueueManagerName* |' ')
INPUTQ 키워드에 이름 지정된 DLQ을 소유하는 큐 관리자의 이름입니다.

큐 관리자를 지정하지 않거나 INPUTQM(' ')을 규칙 테이블에 지정하는 경우, 시스템이 설치를 위한 기본 큐 관리자를 사용합니다.

RETRYINT (*Interval* | 60)
DLQ 핸들러가 첫 번째 시도에서 처리될 수 없는 메시지를 DLQ에서 처리하려고 시도하고 반복되는 시도가 요청되는 간격(초)입니다. 기본적으로, 재시도 간격은 60초입니다.

WAIT (YES |NO| *nnn*)
DLQ 핸들러가 처리할 수 있는 추가 메시지가 없다는 것을 감지할 때 DLQ에 도착할 추가 메시지를 기다려야 하는지 여부입니다.

YES
DLQ 핸들러가 무기한으로 대기하도록 합니다.

아니오
DLQ가 비어 있거나 처리할 수 있는 메시지를 포함하지 않는다는 것을 감지할 때 DLQ 핸들러가 종료하도록 합니다.

nnn
DLQ 핸들러가 큐가 비어 있거나 처리할 수 있는 메시지가 없음을 감지한 후 새 작업이 종료하기 전에 도착하도록 *nnn*초 동안 대기하도록 합니다.

사용이 많은 DLQ의 경우 WAIT(YES)를 지정하고, 활동 레벨이 낮은 DLQ의 경우 WAIT(NO) 또는 WAIT(*nnn*)를 지정하십시오. DLQ 핸들러를 종료할 수 있는 경우, 트리거를 사용하여 이를 다시 호출하십시오.

규칙에서 제어 데이터를 포함하여 목록화하기 위한 대안으로서, **STRMQMDLQ** 명령에 입력 매개변수로서 DLQ의 이름을 제공할 수 있습니다. 임의의 값이 규칙 테이블 및 **STRMQMDLQ**에 대한 입력 모두에 지정되는 경우, **STRMQMDLQ** 명령에 지정된 값이 우선입니다.

참고: 제어 데이터 입력 항목이 규칙 테이블에 포함되는 경우 테이블의 첫 번째 입력 항목이어야 합니다.

규칙(패턴 및 조치)
이 정보를 사용하여 DLQ 규칙을 이해하십시오.

다음은 DLQ 핸들러 규칙 테이블의 규칙 예입니다.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
ACTION (RETRY) RETRY (3)
```

이 규칙에 따라 DLQ 핸들러는 MQPUT 및 MQPUT1이 금지되기 때문에 DLQ에 있는 지속적인 메시지를 목적지 큐에 이동하도록 세 번 시도합니다.

이 섹션은 규칙에 포함할 수 있는 키워드를 설명합니다. 다음에 유의하십시오.

• 키워드가 있는 경우 기본값은 밑줄 그어져 있습니다. 대부분의 키워드의 경우 기본값이 *(별표)이며 어느 값과도 일치합니다.
• 세로 선(|)은 대안을 분리합니다. 다음 중 하나만 지정할 수 있습니다.
• 모든 키워드 예외 ACTION은 선택사항입니다.

이 섹션은 DLQ의 메시지가 일치되는 패턴 일치 키워드의 설명으로 시작됩니다. 그런 다음, 조치 키워드에 대해 설명합니다(DLQ 핸들러가 일치 메시지를 처리하는 방법을 판별하는 것).

패턴 일치 키워드
패턴 일치 키워드는 다음 예제에서 설명됩니다. DLQ에서 메시지가 일치되는지에 대한 값을 지정하려면 사용하십시오. 모든 패턴 일치 키워드는 선택사항입니다.

APPLIDAT (*ApplIdentityData* | *)
메시지 디스크립터 MQMD에 지정된 DLQ에 있는 메시지의 *ApplIdentityData* 값입니다.

APPLNAME (*PutApplName* | *)
DLQ에 메시지의 메시지 디스크립터 MQMD의 *PutApplName* 필드에 지정된 것처럼, MQPUT 또는 MQPUT1 호출을 실행한 애플리케이션의 이름입니다.

APPLTYPE (*PutApplType* | *)
DLQ에 있는 메시지의 메시지 디스크립터 MQMD에 지정된 *PutApplType* 값입니다.

DESTQ (*QueueName* | *)
메시지의 목적지로 지정된 메시지 큐의 이름입니다.

DESTQM (*QueueManagerName* | *)
메시지의 목적지로 지정된 메시지 큐의 큐 관리자 이름입니다.

FEEDBACK (*Feedback* | *)
MsgType 값이 MQMT_REPORT일 때, *Feedback*에 보고서의 성질에 대해 설명합니다.

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQFB_COA를 사용하여 목적지 큐에서 도착을 확인해야 하는 DLQ의 메시지를 식별할 수 있습니다.

FORMAT (*Format* | *)
메시지 송신자가 메시지 데이터 형식을 설명하는 데 사용하는 이름입니다.

MSGTYPE (*MsgType* | *)
DLQ에 있는 메시지의 메시지 유형입니다.

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQMT_REQUEST를 사용하여 응답이 필요한 DLQ의 해당 메시지를 식별할 수 있습니다.

PERSIST (*Persistence* | *)
메시지의 지속성 값입니다. (메시지의 지속성은 큐 관리자 재시작 후에도 지속되는지 여부를 판별합니다.)

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQPER_PERSISTENT를 사용하여 DLQ에 있는 지속적인 메시지를 식별할 수 있습니다.

REASON (*ReasonCode* | *)
메시지가 DLQ에 넣어진 이유에 대해 설명하는 이유 코드입니다.

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQRC_Q_FULL을 사용하여 목적지 큐가 가득 찼기 때문에 DLQ에 배치된 메시지를 식별할 수 있습니다.

REPLYQ (*QueueName* | *)
DLQ에 있는 메시지의 메시지 디스크립터 MQMD에 지정된 응답 대상 큐 이름입니다.

REPLYQM (*QueueManagerName* | *)
REPLYQ 키워드에서 지정된 응답 대상 큐의 큐 관리자 이름입니다.

USERID (*UserIdentifier* | *)
메시지 디스크립터 MQMD에 지정된 대로 DLQ에서 메시지를 생성한 사용자의 사용자 ID입니다.

조치 키워드
조치 키워드가 설명됩니다. 이를 사용하여 일치하는 메시지가 처리되는 방법에 대해 판별하십시오.

ACTION (DISCARD|IGNORE|RETRY|FWD)
이 규칙에서 정의된 패턴과 일치하는 DLQ의 메시지에 대해 수행된 조치입니다.

DISCARD
DLQ에서 메시지를 삭제합니다.

IGNORE
메시지가 DLQ에 보관되도록 합니다.

RETRY
DLQ 핸들러가 목적지 큐에 메시지를 다시 넣으려고 합니다.

FWD
DLQ 핸들러가 FWDQ 키워드에서 이름이 지정된 큐로 메시지를 전달합니다.

ACTION 키워드를 지정해야 합니다. 조치를 구현하기 위해 시도한 횟수 RETRY 키워드에 의해 제어됩니다. 제어 데이터의 RETRYINT 키워드는 시도 사이의 간격을 제어합니다.

FWDQ (*QueueName* | 앤드 DESTQ |, REPLYQ)
ACTION 키워드를 선택할 때 메시지가 전달되는 메시지 큐의 이름입니다.

QueueName
메시지 큐의 이름입니다. FWDQ(' ')가 올바르지 않습니다.

&DESTQ
MQDLH 구조의 *DestQName* 필드에서 큐 이름을 가져옵니다.

&REPLYQ
메시지 디스크립터(MQMD)의 *ReplyToQ* 필드에서 큐 이름을 가져옵니다.

메시지 패턴에서 REPLYQ (? *) 를 지정하여 오류 메시지를 방지할 수 있습니다. FWDQ를 지정하는 규칙이 공백 *ReplyToQ* 필드가 있는 메시지와 일치하는 경우입니다.

FWDQM (*QueueManager*이름 | 앤드 DESTQM | & REPLYQM | ' ')
메시지가 전달되는 큐의 큐 관리자입니다.

QueueManagerName
ACTION(FWD) 키워드를 선택할 때 메시지가 전달되는 큐에 대한 큐 관리자 이름입니다.

&DESTQM
MQDLH 구조의 *DestQMgrName* 필드에서 큐 관리자 이름을 가져옵니다.

&REPLYQM
메시지 디스크립터(MQMD)의 *ReplyToQMgr* 필드에서 큐 관리자 이름을 가져옵니다.

' '
기본값인 FWDQM(' ')은 로컬 큐 관리자를 식별합니다.

HEADER (YES |NO)
MQDLH가 ACTION(FWD)이 요청된 메시지에 남아 있어야 하는지 여부를 지정합니다. 기본적으로 MQDLH는 메시지에 남아 있습니다. HEADER 키워드는 FWD 이외의 다른 조치에는 유효하지 않습니다.

PUTAUT (DEF |CTX)
DLQ 핸들러가 메시지를 넣을 때 필요한 권한입니다.

DEF
DLQ 핸들러 자체의 권한으로 메시지를 넣습니다.

CTX
메시지 컨텍스트에서 사용자 ID의 권한을 사용하여 메시지를 넣습니다. PUTAUT(CTX)를 지정한 경우 다른 사용자의 ID를 가정할 수 있는 권한이 있어야 합니다.

RETRY (*RetryCount* | 1)
조치를 시도하기 위한 횟수(1 - 999,999,999 범위)(제어 데이터의 RETRYINT 키워드에 지정되는 간격)

참고: DLQ 핸들러가 특정 규칙을 구현하기 위해 시도한 횟수는 DLQ 핸들러의 현재 인스턴스에 따라 다릅니다. 재시작하는 동안에는 계수가 지속되지 않습니다. DLQ 핸들러를 재시작하면 규칙을 적용하기 위해 시도한 횟수가 0으로 재설정됩니다.

규칙 테이블 규약
규칙 테이블은 해당 구문, 구조 및 컨텐츠에 관하여 다음 규약을 고수해야 합니다.

- 규칙 테이블에는 최소 하나의 규칙이 있어야 합니다.
- 키워드는 임의의 순서로 발생할 수 있습니다.
- 키워드는 모든 규칙에 한 번만 포함될 수 있습니다.
- 키워드는 대소문자를 구분하지 않습니다.
- 키워드 및 매개변수 값은 최소 하나의 공백이나 쉼표로 다른 키워드와 구분해야 합니다.
- 규칙의 시작 또는 끝 부분과 키워드, 구두점 및 값 사이에 여러 개의 공백이 있을 수 있습니다.
- 각 규칙은 새 행에서 시작해야 합니다.
- 이식성의 경우, 행의 중요한 길이는 72자를 넘어서는 안됩니다.
- 행에서 마지막 공백이 아닌 문자로서 더하기 부호(+)를 사용하여 다음 행에서 첫 번째 공백이 아닌 문자에서 규칙이 계속되는지 표시하십시오. 행에서 마지막 공백이 아닌 문자로서 빼기 부호(-)를 사용하여 다음 행의 시작에서 규칙이 계속되는지 표시하십시오. 키워드 및 매개변수 내에서 연속 문자가 발생할 수 있습니다.

  예를 들면, 다음과 같습니다.

```
APPLNAME('ABC+
D')
```

  결과: 'ABCD'.

```
APPLNAME('ABC-
D')
```

  결과: 'ABC D'.
- 별표(*)로 시작되는 주석 행은 규칙 테이블에서 어디에나 발생할 수 있습니다.
- 빈 줄은 무시됩니다.
- DLQ 핸들러 규칙 테이블의 각 입력 항목은 하나 이상의 키워드 및 연관된 매개변수로 구성됩니다. 매개변수는 다음 구문 규칙을 따라야 합니다.
  - 각 매개변수 값은 최소 하나의 유효 문자를 포함해야 합니다. 작은따옴표로 묶인 값을 분리하는 따옴표는 중요한 것으로 여겨지지 않습니다. 예를 들어, 올바른 매개변수는 다음과 같습니다.

`FORMAT('ABC')`	3개의 유효 문자
`FORMAT(ABC)`	3개의 유효 문자
`FORMAT('A')`	1개의 유효 문자
`FORMAT(A)`	1개의 유효 문자
`FORMAT(' ')`	1개의 유효 문자

    이러한 매개변수는 중요한 문자를 포함하지 않으므로 올바르지 않습니다.

    ```
    FORMAT('')
    FORMAT( )
    FORMAT()
    FORMAT
    ```

  - 와일드카드 문자가 지원됩니다. 후미 문자 공백을 제외하고 단일 문자 대신 물음표(?)를 사용할 수 있습니다. 0개 이상의 인접 문자 대신 별표(*)를 사용할 수 있습니다. 별표(*) 및 물음표(?)는 매개변수 값에서 **항상** 와일드카드 문자로 해석됩니다.
  - ACTION, HEADER, RETRY, FWDQ, FWDQM 및 PUTAUT 키워드의 매개변수에는 와일드카드 문자를 포함할 수 없습니다.
  - 매개변수 값과 DLQ 메시지의 해당 필드에 있는 후미 공백은 와일드카드 일치를 수행할 때 유의미하지 않습니다. 그러나 따옴표로 묶인 문자열 내에 있는 선두 공백과 임베드된 공백은 와일드카드 일치에서 유의미합니다.

- 숫자 매개변수는 물음표(?) 와일드카드 문자를 포함할 수 없습니다. 전체 숫자 매개변수 대신 별표(*)를 사용할 수 있지만, 별표를 숫자 매개변수의 일부로서 사용할 수 없습니다. 예를 들어, 다음은 올바른 숫자 매개변수입니다.

`MSGTYPE(2)`	응답 메시지만 적합합니다.
`MSGTYPE(*)`	모든 메시지 유형이 적합합니다.
`MSGTYPE('*')`	모든 메시지 유형이 적합합니다.

  그러나 `MSGTYPE('2*')`는 숫자 매개변수의 일부로 별표(*)가 포함되어 있으므로 올바르지 않습니다.
- 숫자 매개변수는 범위 0-999 999 999에 있어야 합니다. 매개변수 값이 이 범위에 있으면 키워드와 관련된 필드에서 현재 그 값이 올바르지 않더라도 허용됩니다. 숫자 매개변수에 기호 이름을 사용할 수 있습니다.
- 문자열 값이 키워드와 관련된 MQDLH 또는 MQMD에 있는 필드보다 짧으면, 필드 길이에 맞게 값이 공백으로 채워집니다. 별표(*)를 제외하고 값이 필드보다 길면 오류가 진단됩니다. 예를 들어, 다음은 모두 8개 문자 필드에서 올바른 문자열 값입니다.

`'ABCDEFGH'`	8자
`'A*C*E*G*I'`	별표를 제외한 5문자
`'*A*C*E*G*I*K*M*O*'`	별표를 제외한 8문자

- 공백, 소문자 또는 마침표(.), 슬래시(/), 밑줄(_) 및 퍼센트 부호(%) 이외의 특수 문자를 포함하는 문자열은 작은따옴표로 묶여야 합니다. 따옴표로 묶지 않은 소문자는 대문자로 변경됩니다. 문자열에 따옴표가 포함되는 경우, 두 개의 작은따옴표는 따옴표의 시작과 끝을 표시하는 데 사용되어야 합니다. 문자열의 길이를 계산할 때, 큰따옴표는 각각 하나의 문자로 계수됩니다.

규칙 테이블 처리
DLQ 핸들러는 규칙 테이블에서 DLQ에 있는 메시지와 일치하는 패턴이 있는 규칙을 검색합니다.

검색은 테이블의 첫 번째 규칙부터 시작되며 테이블 전체에서 순차적으로 계속됩니다. 일치하는 패턴이 있는 규칙이 발견되면, 규칙 테이블은 해당 규칙에서 조치를 시도합니다. DLQ 핸들러가 해당 규칙을 적용하려고 시도할 때마다 규칙에 대한 재시도 수가 1씩 증가합니다. 첫 번째 시도가 실패하면 시도 횟수가 RETRY 키워드에서 지정된 수와 일치할 때까지 시도가 반복됩니다. 모든 시도가 실패하면, DLQ 핸들러는 테이블에서 그 다음으로 일치하는 규칙을 검색합니다.

이 프로세스는 조치가 성공할 때까지 일치하는 후속 규칙에 대해 반복됩니다. 일치하는 각 규칙이 RETRY 키워드에 지정된 횟수만큼 시도되고 모든 시도가 실패하면, ACTION(IGNORE)이 가정됩니다. 일치하는 규칙을 찾지 못한 경우에도 ACTION(IGNORE)이 가정됩니다.

참고:

1. DLQ에서 MQDLH로 시작하는 메시지에 대해서만 일치하는 규칙 패턴을 찾습니다. MQDLH로 시작하지 않는 메시지는 정기적으로 오류가 있는 것으로 보고되며, 무기한 DLQ에 남게 됩니다.
2. 모든 패턴 키워드는 기본값일 수 있으므로 규칙은 조치로만 구성될 수 있습니다. 그러나 조치만으로 구성된 규칙은 MQDLH가 있고 테이블의 다른 규칙에 따라 아직 처리되지 않은 큐의 모든 메시지에 적용됩니다.
3. 규칙 테이블은 DLQ 핸들러가 시작될 때 유효성 검증되며, 그 시간에 오류 플래그가 지정됩니다. (DLQ 핸들러가 발행한 오류 메시지가 메시지 및 이유 코드에서 설명됩니다.) 언제든지 규칙 테이블로 변경할 수 있지만, DLQ 핸들러가 다시 시작될 때까지 해당 변경사항이 적용되지 않습니다.
4. DLQ 핸들러는 MQDLH 또는 메시지 디스크립터의 메시지 컨텐츠를 대체하지 않습니다. DLQ 핸들러는 메시지 옵션 MQPMO_PASS_ALL_CONTEXT를 사용하여 항상 다른 큐에 메시지를 넣습니다.
5. 규칙 테이블의 유효성 검사가 반복 오류의 생성을 제거하기 때문에, 규칙 테이블의 연속적 구문 오류는 인지되지 않을 수도 있습니다.
6. DLQ 핸들러는 MQOO_INPUT_AS_Q_DEF 옵션을 사용하여 DLQ를 엽니다.
7. DLQ 핸들러의 다중 인스턴스는 동일한 규칙 테이블을 사용하여 동일한 큐에 대해 동시에 실행될 수 있습니다. 그러나, 일반적으로 DLQ와 DLQ 핸들러 간에는 1:1 관계가 성립됩니다.

모든 *DLQ* 메시지가 처리되었는지 확인
DLQ 핸들러는 발견되었지만 제거되지 않은 모든 메시지의 레코드를 DLQ에 보관합니다.

DLQ 핸들러를 필터로 사용하여 DLQ에서 메시지의 작은 서브세트를 추출하는 경우, DLQ 핸들러는 처리하지 않은 해당 메시지의 레코드를 DLQ에 계속 보관합니다. 또한 DLQ 핸들러는 DLQ가 선입선출(FIFO)로 정의된 경우에도 DLQ에 도착하는 새 메시지가 표시되는 것을 보장할 수 없습니다. 큐가 비어있지 않으면, DLQ는 모든 메시지를 검사하기 위해 주기적으로 재검색됩니다.

이런 이유로 DLQ에 가능한 적은 수의 메시지가 포함되는지 확인하십시오. 어떤 이유로든 제거하거나 다른 큐로 전달할 수 없는 메시지가 큐에 누적되도록 허용하면 DLQ 핸들러의 워크로드가 증가하고 DLQ 자체가 가득 찰 위험이 있습니다.

DLQ 핸들러가 DLQ를 비울 수 있도록 특정 조치를 수행할 수 있습니다. 예를 들어, DLQ에 메시지를 남기는 ACTION(IGNORE)을 사용하지 않도록 하십시오. (ACTION(IGNORE)은 테이블의 다른 규칙에 의해 명시적으로 주소 지정되지 않은 메시지에 대해 가정되는 점을 기억하십시오.) 대신, 무시할 메시지에 대해서는 메시지를 다른 큐로 이동하는 조치를 사용하십시오. 예를 들면, 다음과 같습니다.

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

마찬가지로, 테이블의 마지막 규칙은 테이블의 이전 규칙이 설명하지 못한 메시지를 처리해야 합니다. 예를 들어, 테이블의 최종 규칙은 다음과 같을 수 있습니다.

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

이 조치는 테이블의 최종 규칙을 통해 메시지가 큐 REALLY.DEAD.QUEUE로 전달되도록 하여 여기서 수동으로 처리될 수 있도록 합니다. 이러한 규칙이 없으면 메시지가 DLQ에 무기한 남아 있을 가능성이 높습니다.

DLQ 핸들러 규칙 테이블 예

다음은 단일 제어 데이터 입력 항목 및 여러 개의 규칙을 포함하는 규칙 테이블 예제입니다.

```
**************************************************************************
*    An example rules table for the STRMQMDLQ command        *
**************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* STRMQMDLQ, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to STRMQMDLQ,
* 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 is 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 does 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 must 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 do not 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 핸들러 호출

runmqdlq 명령을 사용하여 DLQ 핸들러를 호출하십시오. 처리하려는 DLQ 및 두 가지 방법으로 사용하려는 큐 관리자의 이름을 지정할 수 있습니다.

두 가지 방법은 다음과 같습니다.

- 명령 프롬프트에서 runmqdlq에 대한 매개변수로. 예를 들면, 다음과 같습니다.

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrule.rul
```

- 규칙 테이블에서. 예를 들면, 다음과 같습니다.

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

예가 큐 관리자 ABC1.QUEUE.MANAGER가 소유하는 ABC1.DEAD.LETTER.QUEUE라는 DLQ에 적용됩니다.

표시된 대로 DLQ 또는 큐 관리자를 지정하지 않으면, 설치를 위한 기본 큐 관리자가 해당 큐 관리자에 속하는 DLQ와 함께 사용됩니다.

runmqdlq 명령은 stdin에서 해당 입력을 사용합니다. 규칙 테이블에서 stdin의 경로를 재지정하여 runmqdlq와 규칙 테이블을 연관시킵니다.

DLQ 핸들러를 실행하려면 DLQ의 메시지가 전달되는 임의의 메시지 큐 및 DLQ 자체 모두에 액세스하기 위한 권한이 부여되어야 합니다. 메시지 컨텍스트에서 사용자 ID의 권한을 가지는 큐에 메시지를 넣기 위한 DLQ 핸들러의 경우, 기타 사용자의 ID를 가정하기 위한 권한이 부여되어야 합니다.

runmqdlq 명령에 대한 자세한 정보는 runmqdlq를 참조하십시오.

관련 정보
데드-레터 큐
미배달 메시지 문제점 해결

샘플 *DLQ* 핸들러, *amqsdlq*
runmqdlq 명령을 사용하여 호출되는 DLQ 핸들러 뿐만 아니라, IBM MQ에서는 runmqdlq 에서 제공하는 것과 유사한 기능을 가지는 샘플 DLQ 핸들러(amqsdlq)의 소스를 제공합니다.

요구사항을 충족하는 DLQ 핸들러를 제공하기 위해 amqsdlq를 사용자 정의할 수 있습니다. 예를 들어, 데드-레터 헤더 없이 메시지를 처리할 수 있는 DLQ 핸들러를 원한다고 결정할 수 있습니다. (기본 DLQ 핸들러 및 샘플(amqsdlq) 모두 데드-레터 헤더 MQDLH로 시작하는 DLQ의 해당 메시지만 처리합니다. MQDLH로 시작하지 않는 메시지는 정기적으로 오류가 있는 것으로 보고되며, 무기한 DLQ에 남게 됩니다.)

*MQ_INSTALLATION_PATH*는 IBM MQ가 설치된 상위 레벨 디렉토리를 나타냅니다.

IBM MQ for Windows에서는 amqsdlq의 소스가 디렉토리에 제공됩니다.

```
MQ_INSTALLATION_PATH\tools\c\samples\dlq
```

그리고 컴파일된 버전이 디렉토리에 제공됩니다.

```
MQ_INSTALLATION_PATH\tools\c\samples\bin
```

IBM MQ for UNIX 및 Linux 시스템에서 amqsdlq의 소스가 디렉토리에 제공됩니다.

MQ_INSTALLATION_PATH/samp/dlq

그리고 컴파일된 버전이 디렉토리에 제공됩니다.

MQ_INSTALLATION_PATH/samp/bin

클라이언트 모드에서 **amqsdlq**를 컴파일할 수도 있습니다. 자세한 내용은 클라이언트 절차 애플리케이션 작성, IBM MQ MQI clients에 대한 애플리케이션 빌드 및 IBM MQ MQI client 환경에서 애플리케이션 실행을 참조하십시오.

DLQ 핸들러 규칙 테이블
DLQ 핸들러 규칙 테이블은 DLQ 핸들러가 DLQ에 도착한 메시지를 처리하는 방법을 정의합니다.

규칙 테이블에는 두 가지 유형의 입력 항목이 있습니다.

- 표의 첫 번째 항목(선택적)에는 제어 데이터가 포함됩니다.
- 테이블의 기타 모든 입력 항목은 DLQ 핸들러가 따라야 할 규칙입니다. 각 규칙은 메시지가 일치되는 패턴(메시지 특성 세트)과, DLQ의 메시지가 지정된 패턴과 일치할 때 수행되는 조치로 구성됩니다. 규칙 테이블에는 최소 하나의 규칙이 있어야 합니다.

규칙 테이블의 각 입력 항목은 하나 이상의 키워드로 구성됩니다.

관련 정보
데드-레터 큐
미배달 메시지 문제점 해결

제어 데이터
이 절에서는 DLQ 핸들러 규칙 테이블의 제어 데이터 입력 항목에 포함시킬 수 있는 키워드에 대해 설명합니다.

참고:

- 수직 행(|)은 대안을 분리하며 이들 중 하나만 지정할 수 있습니다.
- 모든 키워드는 선택사항입니다.

INPUTQ (*QueueName* |' ')
처리하려는 DLQ의 이름입니다.

1. runmqdlq 명령에 매개변수로서 제공하는 INPUTQ 값이 규칙 테이블에서 INPUTQ 값을 대체합니다.
2. runmqdlq 명령에 매개변수로서 INPUTQ 값을 지정하지 않지만 규칙 테이블에서 값을 **지정하는** 경우, 규칙 테이블의 INPUTQ 값이 사용됩니다.
3. DLQ가 지정되지 않거나 규칙 테이블에서 INPUTQ(' ')를 지정하는 경우, runmqdlq 명령에 매개변수로서 제공되는 이름을 가지는 큐 관리자에 속하는 DLQ의 이름이 사용됩니다.
4. runmqdlq 명령에 매개변수로서 INPUTQ 값을 지정하지 않거나 규칙 테이블에서 값으로 INPUTQ 값을 지정하지 않는 경우, 규칙 테이블의 INPUTQ 키워드에서 이름 지정된 큐 관리자에 속하는 DLQ가 사용됩니다.

INPUTQM (*QueueManagerName* |' ')
INPUTQ 키워드에서 이름이 지정된 DLQ를 소유하는 큐 관리자의 이름입니다.

1. runmqdlq 명령에 매개변수로서 제공하는 INPUTQM 값이 규칙 테이블에서 INPUTQM 값을 대체합니다.
2. runmqdlq 명령에 매개변수로서 INPUTQM 값을 지정하지 않는 경우, 규칙 테이블에서 INPUTQM 값이 사용됩니다.
3. 큐 관리자가 지정되지 않거나 규칙 테이블에 INPUTQM(' ')을 지정하는 경우, 설치를 위한 기본 큐 관리자가 사용됩니다.

RETRYINT (*Interval* | 60)
DLQ 핸들러가 첫 번째 시도에서 처리될 수 없는 DLQ의 메시지를 처리해야 하고 반복되는 시도가 요청되는 간격(초)입니다. 기본적으로, 재시도 간격은 60초입니다.

WAIT (YES |NO| *nnn*)
DLQ 핸들러가 처리할 수 있는 추가 메시지가 없다는 것을 감지할 때 DLQ에 도착할 추가 메시지를 기다려야 하는지 여부입니다.

YES
DLQ 핸들러가 무기한 대기합니다.

아니오
DLQ 핸들러는 DLQ가 비어 있거나 처리할 수 있는 메시지가 포함되어 있다는 것을 감지할 때 종료됩니다.

nnn
DLQ 핸들러는 큐가 비어 있거나 처리할 수 있는 메시지를 포함하지 않는 다는 것을 감지한 이후 종료하기 전에 도착할 새 작업에 대해 *nnn* 초 동안 대기합니다.

활동의 하위 레벨을 가지고 있는 DLQ에 대해 WAIT (NO) 또는 WAIT (*nnn*) 및 사용 중인 DLQ에 대해 WAIT (YES)를 지정하십시오. DLQ 핸들러가 종료하도록 허용되는 경우, 트리거를 사용하여 다시 호출하십시오. 트리거에 대한 자세한 정보는 트리거를 사용하여 IBM MQ 애플리케이션 시작을 참조하십시오.

규칙 테이블에 제어 데이터를 포함하는 것에 대한 대체는 runmqdlq 명령에 입력 매개변수로서 해당 큐 관리자 및 DLQ의 이름을 제공하는 것입니다. 규칙 테이블에 값을 지정하고 runmqdlq 명령에 입력으로 값을 지정하는 것이 모두 가능한 경우, runmqdlq 명령에 지정되는 값이 우위에 섭니다.

규칙 테이블에 제어-데이터 입력 항목을 포함하는 경우, 테이블에 **첫 번째** 입력 항목일 수 있습니다.

규칙*(패턴 및 조치)*
DLQ의 메시지가 일치되는 패턴 일치 키워드 및 DLQ 핸들러가 일치 메시지를 처리하는 방법을 판별하는 조치 키워드에 대한 설명입니다. 예시 규칙도 제공됩니다.

패턴 일치 키워드

DLQ의 메시지가 일치되는 값을 지정하려면 사용하는 패턴 일치 키워드는 다음과 같습니다. (모든 패턴 일치 키워드는 선택사항입니다):

APPLIDAT (*ApplIdentityData* | *)
DLQ의 메시지에 대한 메시지 디스크립터(MQMD)에 지정된 *ApplIdentityData* 값입니다.

APPLNAME (*PutApplName* | *)
DLQ에 있는 메시지의 메시지 디스크립터, MQMD의 *PutAppl*이름 필드에 지정된 대로 MQPUT 또는 MQPUT1 호출을 실행한 애플리케이션의 이름입니다.

APPLTYPE (*PutApplType* | *)
DLQ에 메시지의 메시지 디스크립터, MQMD에 지정되는 *PutApplType* 값입니다.

DESTQ (*QueueName* | *)
메시지의 목적지로 지정된 메시지 큐의 이름입니다.

DESTQM (*QueueManagerName* | *)
메시지가 향하는 메시지 큐의 큐 관리자의 이름입니다.

FEEDBACK (*Feedback* | *)
MsgType 값이 MQFB_REPORT일 때, *Feedback*은 보고서의 성질을 설명합니다.

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQFB_COA를 사용하여 목적지 큐에서 도착 확인이 필요한 DLQ에 대한 해당 메시지를 식별할 수 있습니다.

FORMAT (*Format* | *)
메시지 송신자가 메시지 데이터 형식을 설명하는 데 사용하는 이름입니다.

MSGTYPE (*MsgType* | *)
DLQ에 있는 메시지의 메시지 유형입니다.

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQMT_REQUEST를 사용하여 응답이 필요한 DLQ에서 해당 메시지를 식별할 수 있습니다.

PERSIST (*Persistence* | *)
메시지의 지속성 값입니다. (메시지의 지속성은 큐 관리자 재시작 후에도 지속되는지 여부를 판별합니다.)

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQPER_PERSISTENT를 사용하여 지속적인 DLQ에서 메시지를 식별할 수 있습니다.

REASON (*ReasonCode* | *)
메시지가 DLQ에 넣어진 이유에 대해 설명하는 이유 코드입니다.

기호 이름을 사용할 수 있습니다. 예를 들어, 기호 이름 MQRC_Q_FULL을 사용하여 목적지 큐가 가득 찼기 때문에 DLQ에 배치된 메시지를 식별할 수 있습니다.

REPLYQ (*QueueName* | *)
DLQ에서 메시지의 메시지 디스크립터(MQMD)에 지정된 응답 대상 큐의 이름입니다.

REPLYQM (*QueueManagerName* | *)
DLQ에서 메시지의 메시지 디스크립터(MQMD)에 지정된 것으로 응답 대상 큐의 큐 관리자 이름입니다.

USERID (*UserIdentifier* | *)
DLQ에서 메시지의 메시지 디스크립터(MQMD)에 지정된 것으로 DLQ에서 메시지를 생성한 사용자의 사용자 ID입니다.

조치 키워드

일치 메시지가 처리되는 방법에 대해 설명하기 위해 사용되는 조치 키워드는 다음과 같습니다.

ACTION (DISCARD|IGNORE|RETRY|FWD)
이 규칙에 정의되는 패턴과 일치하는 DLQ에서 메시지에 대해 취하는 조치입니다.

DISCARD
DLQ에서 메시지를 삭제하십시오.

IGNORE
DLQ에 메시지를 남기십시오.

RETRY
해당 목적지 큐에 메시지를 넣으려는 첫 번째 시도가 실패하면 다시 시도하십시오. RETRY 키워드는 조치를 구현하기 위한 시도 횟수를 설정합니다. 제어 데이터의 RETRYINT 키워드는 시도 사이의 간격을 제어합니다.

FWD
FWDQ 키워드에 이름이 지정된 큐로 메시지를 전달합니다.

ACTION 키워드를 지정해야 합니다.

FWDQ (*QueueName* | 앤드 DESTQ |, REPLYQ)
ACTION(FWD)이 요청될 때 메시지가 전달되는 메시지 큐의 이름입니다.

QueueName
메시지 큐의 이름입니다. FWDQ(' ')가 올바르지 않습니다.

&DESTQ
MQDLH 구조의 *DestQName* 필드에서 큐 이름을 가져옵니다.

&REPLYQ
메시지 디스크립터(MQMD)의 *ReplyToQ* 필드에서 큐 이름을 가져옵니다.

FWDQ (S-REPLYQ) 를 지정하는 규칙이 공백 *ReplyToQ* 필드가 있는 메시지와 일치하는 경우 오류 메시지를 방지하려면 메시지 패턴에 REPLYQ (? *) 를 지정하십시오.

FWDQM (*QueueManager*이름 | 앤드 DESTQM | & REPLYQM | ' ')
메시지를 전달할 큐의 큐 관리자입니다.

QueueManagerName
ACTION(FWD)이 요청될 때 메시지가 전달되는 큐의 큐 관리자의 이름입니다.

&DESTQM
MQDLH 구조의 *DestQMgrName* 필드에서 큐 관리자 이름을 가져옵니다.

&REPLYQM
메시지 디스크립터(MQMD)의 *ReplyToQMgr* 필드에서 큐 관리자 이름을 가져옵니다.

' '
기본값인 FWDQM(' ')은 로컬 큐 관리자를 식별합니다.

HEADER (YES |NO)
MQDLH가 ACTION(FWD)이 요청된 메시지에 남아 있어야 하는지 여부를 지정합니다. 기본적으로 MQDLH는 메시지에 남아 있습니다. HEADER 키워드는 FWD 이외의 다른 조치에는 유효하지 않습니다.

PUTAUT (DEF |CTX)
DLQ 핸들러가 메시지를 넣을 때 필요한 권한입니다.

DEF
DLQ 핸들러 자체 권한을 가진 메시지를 넣으십시오.

CTX
메시지 컨텍스트에서 사용자 ID의 권한을 가진 메시지를 넣으십시오. PUTAUT(CTX)를 지정하면, 기타 사용자의 ID를 추측할 수 있는 권한이 지정되어야 합니다.

RETRY (*RetryCount* | 1)
제어 데이터의 RETRYINT 키워드에 지정된 간격에서 조치를 시도하기 위한 횟수(1 - 999,999,999 범위)입니다. DLQ 핸들러가 특정 규칙을 구현하기 위해 시도한 횟수는 DLQ 핸들러의 현재 인스턴스에 따라 다릅니다. 재시작하는 동안에는 계수가 지속되지 않습니다. DLQ 핸들러가 재시작되면, 규칙에 적용되는 시도 횟수는 0으로 재설정됩니다.

예 규칙

다음은 DLQ 핸들러 규칙 테이블의 규칙 예입니다.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
ACTION (RETRY) RETRY (3)
```

이 규칙에 따라 DLQ 핸들러는 MQPUT 및 MQPUT1이 금지되어 DLQ에 있는 지속적인 메시지를 목적지 큐에 이동하도록 세 번 시도합니다.

규칙에 사용할 수 있는 모든 키워드에 대해서는 이 절 나머지 부분에서 설명됩니다. 다음에 유의하십시오.

- 키워드가 있는 경우 기본값은 밑줄 그어져 있습니다. 대부분의 키워드의 경우 기본값이 *(별표)이며 어느 값과도 일치합니다.
- 수직 행(|)은 대안을 분리하며 이들 중 하나만 지정할 수 있습니다.
- 모든 키워드 예외 ACTION은 선택사항입니다.

규칙 테이블 규약

DLQ 핸들러 규칙 테이블의 컨텐츠, 구조 및 구문은 이러한 규약을 고수해야 합니다.

규칙 테이블은 다음 규약을 고수해야 합니다.

- 규칙 테이블에는 최소 하나의 규칙이 있어야 합니다.
- 키워드는 임의의 순서로 발생할 수 있습니다.
- 키워드는 규칙에 한 번만 포함될 수 있습니다.
- 키워드는 대소문자를 구분하지 않습니다.
- 키워드 및 매개변수 값은 최소 하나의 공백이나 쉼표로 다른 키워드와 구분해야 합니다.
- 공백은 규칙 처음이나 끝에 넣을 수 있으며, 키워드, 구두점, 값 사이에 공백을 넣을 수 있습니다.
- 각 규칙은 새 행에서 시작해야 합니다.
- Windows 시스템의 표에서 마지막 규칙은 캐리지 리턴/라인 공급 문자로 끝나야만 합니다. 테이블의 마지막 행이 빈 줄이도록 규칙의 끝에서 Enter 키를 입력하는지 확인하여 이를 얻을 수 있습니다.
- 이식성의 이유로 행의 중요한 길이는 72자를 넘어서는 안됩니다.
- 다음 행의 첫 번째 비공백 문자부터 규칙이 계속되는 것을 표시하려면 행의 마지막 비공백 문자로 더하기 부호(+)를 사용하십시오. 다음 행의 시작부터 규칙이 계속되는 것을 표시하려면 행의 마지막 비공백 문자로 빼기 부호(-)를 사용하십시오. 키워드 및 매개변수 내에서 연속 문자가 발생할 수 있습니다.

  예를 들면, 다음과 같습니다.

```
APPLNAME('ABC+
D')
```

  결과적으로 'ABCD'가 되고

```
APPLNAME('ABC-
D')
```

  결과: 'ABC D'.
- 별표(*)로 시작되는 주석 행은 규칙 테이블에서 어디에나 발생할 수 있습니다.
- 빈 줄은 무시됩니다.
- DLQ 핸들러 규칙 테이블의 각 입력 항목은 하나 이상의 키워드 및 연관된 매개변수로 구성됩니다. 매개변수는 다음 구문 규칙을 따라야 합니다.
  - 각 매개변수 값은 최소 하나의 유효 문자를 포함해야 합니다. 따옴표로 묶이는 값을 분리하는 작은따옴표는 중요한 것으로 여겨지지 않습니다. 예를 들어, 올바른 매개변수는 다음과 같습니다.

`FORMAT('ABC')`	3개의 유효 문자
`FORMAT(ABC)`	3개의 유효 문자
`FORMAT('A')`	1개의 유효 문자
`FORMAT(A)`	1개의 유효 문자
`FORMAT(' ')`	1개의 유효 문자

    이러한 매개변수는 중요한 문자를 포함하지 않으므로 올바르지 않습니다.

    ```
    FORMAT('')
    FORMAT( )
    FORMAT()
    FORMAT
    ```

- 와일드카드 문자가 지원됩니다. 후미 공백을 제외하고는 단일 문자 대신 물음표(?)를 사용할 수 있습니다. 0개 이상의 인접 문자 대신 별표(*)를 사용할 수 있습니다. 별표(*) 및 물음표(?)는 매개변수 값에서 **항상** 와일드카드 문자로 해석됩니다.
- 와일드카드 문자는 키워드(ACTION, HEADER, RETRY, FWDQ, FWDQM 및 PUTAUT)의 매개변수에 포함될 수 없습니다.
- 매개변수 값과 DLQ 메시지의 해당 필드에 있는 후미 공백은 와일드카드 일치를 수행할 때 유의미하지 않습니다. 그러나, 작은따옴표로 묶인 문자열 내의 선두 문자 및 임베드된 공백은 와일드카드 일치에 중요합니다.
- 숫자 매개변수는 물음표(?) 와일드카드 문자를 포함할 수 없습니다. 전체 숫자 매개변수 대신 별표(*)를 사용할 수 있지만, 숫자 매개변수의 일부로서 사용할 수 없습니다. 예를 들어, 다음은 올바른 숫자 매개변수입니다.

`MSGTYPE(2)`	응답 메시지만 적합합니다.
`MSGTYPE(*)`	모든 메시지 유형이 적합합니다.
`MSGTYPE('*')`	모든 메시지 유형이 적합합니다.

  그러나 `MSGTYPE('2*')`는 숫자 매개변수의 일부로 별표(*)가 포함되어 있으므로 올바르지 않습니다.
- 숫자 매개변수는 범위 0-999 999 999에 있어야 합니다. 매개변수 값이 이 범위에 있으면 키워드와 관련된 필드에서 현재 그 값이 올바르지 않더라도 허용됩니다. 숫자 매개변수에 기호 이름을 사용할 수 있습니다.
- 문자열 값이 키워드와 관련된 MQDLH 또는 MQMD에 있는 필드보다 짧으면, 필드 길이에 맞게 값이 공백으로 채워집니다. 별표(*)를 제외하고 값이 필드보다 길면 오류가 진단됩니다. 예를 들어, 다음은 모두 8개 문자 필드에서 올바른 문자열 값입니다.

`'ABCDEFGH'`	8자
`'A*C*E*G*I'`	별표를 제외한 5문자
`'*A*C*E*G*I*K*M*O*'`	별표를 제외한 8문자

- 공백, 소문자 또는 마침표(.), 정방향 슬래시(/), 밑줄(_) 및 퍼센트 부호(%) 이외의 특수 문자가 포함된 문자열을 작은따옴표로 묶으십시오. 작은따옴표로 묶이지 않은 소문자는 대문자로 접힙니다. 문자열에 따옴표가 포함되면, 두 개의 작은따옴표를 사용하여 따옴표 시작과 끝을 모두 나타내야 합니다. 문자열의 길이를 계산할 때, 큰따옴표는 각각 하나의 문자로 계수됩니다.

규칙 테이블이 처리되는 방법

DLQ 핸들러는 패턴이 DLQ의 메시지와 일치하는 규칙에 대해 규칙 테이블을 검색합니다.

검색은 테이블의 첫 번째 규칙부터 시작되며 테이블 전체에서 순차적으로 계속됩니다. DLQ 핸들러가 일치하는 패턴을 가진 규칙을 찾을 때, 해당 규칙으로부터 조치를 취합니다. DLQ 핸들러는 해당 규칙에 적용될 때마다 규칙에 대한 재시도 수가 1씩 증가합니다. 첫 번째 시도에 실패한 경우, DLQ 핸들러는 RETRY 키워드에 지정되는 수와 시도 수가 일치할 때까지 다시 시도합니다. 모든 시도가 실패하면, DLQ 핸들러는 테이블에서 그 다음으로 일치하는 규칙을 검색합니다.

이 프로세스는 조치가 성공할 때까지 일치하는 후속 규칙에 대해 반복됩니다. 일치하는 각 규칙이 RETRY 키워드에 지정된 횟수만큼 시도되고 모든 시도가 실패하면, ACTION(IGNORE)이 가정됩니다. 일치하는 규칙을 찾지 못한 경우에도 ACTION(IGNORE)이 가정됩니다.

참고:

1. DLQ에서 MQDLH로 시작하는 메시지에 대해서만 일치하는 규칙 패턴을 찾습니다. MQDLH로 시작하지 않는 메시지는 정기적으로 오류가 있는 것으로 보고되며, 무기한 DLQ에 남게 됩니다.
2. 규칙이 조치로만 구성될 수 있도록, 모든 패턴 키워드는 기본값에 허용될 수 있습니다. 그러나 조치만으로 구성된 규칙은 MQDLH가 있고 테이블의 다른 규칙에 따라 아직 처리되지 않은 큐의 모든 메시지에 적용됩니다.
3. DLQ 핸들러가 시작될 때 규칙 테이블의 유효성이 검증되고 그 때 오류가 플래그됩니다. 언제든지 규칙 테이블로 변경할 수 있지만, DLQ 핸들러가 다시 시작할 때까지 해당 변경사항이 발효되지 않습니다.
4. DLQ 핸들러는 메시지, MQDLH 또는 메시지 디스크립터의 컨텐츠를 대체하지 않습니다. DLQ 핸들러는 메시지 옵션 MQPMO_PASS_ALL_CONTEXT를 사용하여 항상 다른 큐에 메시지를 넣습니다.

5. 규칙 테이블의 연속 구문 오류는 규칙 테이블이 유효성 검증 동안 반복적인 오류의 생성을 제거하도록 설계되기 때문에 인식되지 않을 수도 있습니다.
6. DLQ 핸들러는 MQOO_INPUT_AS_Q_DEF 옵션을 사용하여 DLQ를 엽니다.
7. DLQ 핸들러의 다중 인스턴스는 동일한 규칙 테이블을 사용하여 동일한 큐에 대해 동시에 실행될 수 있습니다. 그러나, 일반적으로 DLQ와 DLQ 핸들러 간에는 1:1 관계가 성립됩니다.

관련 정보

데드-레터 큐

미배달 메시지 문제점 해결

모든 *DLQ* 메시지가 처리되었는지 확인

DLQ 핸들러는 발견되었지만 제거되지 않은 모든 메시지의 레코드를 DLQ에 보관합니다.

DLQ 핸들러를 필터로 사용하여 DLQ에서 소규모 메시지 서브세트를 추출하는 경우, DLQ 핸들러는 여전히 DLQ에 아직 처리하지 않은 메시지에 대한 기록을 보존해야 합니다. 또한 DLQ 핸들러는 DLQ가 FIFO(First-In-First-Out)로 정의된 경우에도 DLQ에 도착하는 새 메시지를 볼 수 있다고 보장할 수 없습니다. 큐가 비어있지 않으면, DLQ는 모든 메시지를 검사하기 위해 주기적으로 재검색됩니다.

이러한 이유로, DLQ에 가능한 적은 메시지가 포함되는지 확인하십시오. 기타 큐로 전달되거나 삭제될 수 없는(어떤 이유로든) 메시지가 큐에 축적되도록 허용되지 않는 경우, DLQ 핸들러의 워크로드가 증가하고 DLQ 자체가 가득 차게 됩니다.

DLQ 핸들러가 DLQ를 비울 수 있도록 특정 조치를 수행할 수 있습니다. 예를 들어, DLQ에 메시지를 남기는 ACTION(IGNORE)을 사용하지 않도록 하십시오. (ACTION(IGNORE)은 테이블의 다른 규칙에 의해 명시적으로 주소 지정되지 않은 메시지에 대해 가정되는 점을 기억하십시오.) 대신, 무시할 메시지에 대해서는 메시지를 다른 큐로 이동하는 조치를 사용하십시오. 예를 들면, 다음과 같습니다.

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

마찬가지로, 테이블의 마지막 규칙은 테이블의 이전 규칙이 설명하지 못한 메시지를 처리해야 합니다. 예를 들어, 테이블의 최종 규칙은 다음과 같을 수 있습니다.

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

이는 테이블의 최종 규칙으로 보내진 메시지를 큐 REALLY.DEAD.QUEUE로 전달합니다. 여기서 메시지는 수동으로 처리될 수 있습니다. 이러한 규칙이 없으면 메시지가 DLQ에 무기한 남아 있을 가능성이 높습니다.

DLQ 핸들러 규칙 테이블 예

단일 제어 데이터 입력 항목 및 여러 규칙을 포함하는 runmqdlq 명령에 대한 예 규칙 테이블입니다.

```
*********************************************************************
*    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)
```

관련 정보

데드-레터 큐

미배달 메시지 문제점 해결

runmqdlq(데드-레터 큐 핸들러 실행)

모델 큐에 대한 작업

큐 관리자는 모델 큐로서 정의되는 큐 이름을 지정하여 애플리케이션에서 MQI 호출을 수신하는 경우 동적 큐를 작성합니다. 새 동적 큐의 이름은 큐가 작성될 때 큐 관리자에서 생성됩니다. 모델 큐는 작성되는 동적 큐의 속성을 지정하는 템플리트입니다. 모델 큐는 애플리케이션에게 필요한 대로 큐를 작성할 수 있는 편리한 방법을 제공합니다.

모델 큐 정의

로컬 큐를 정의하는 동일한 방법으로 속성 세트와 함께 모델 큐를 정의합니다. 모델 큐 및 로컬 큐에는 작성되는 동적 큐가 임시 또는 영구적인지 여부를 지정할 수 있는 모델 큐의 속성을 제외하고 동일한 속성 세트가 있습니

다. (영구적 큐는 큐 관리자 재시작을 통해 유지보수되고 임시 큐는 그렇지 않습니다). 예를 들면, 다음과 같습니다.

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
DESCR('Queue for messages from application X') +
PUT (DISABLED) +
GET (ENABLED) +
NOTRIGGER +
MSGDLVSQ (FIFO) +
MAXDEPTH (1000) +
MAXMSGL (2000) +
USAGE (NORMAL) +
DEFTYPE (PERMDYN)
```

이 명령은 모델 큐 정의를 작성합니다. DFNTYPE 속성에서 이 템플리트에서 작성된 실제 큐가 영구적 동적 큐라는 것을 볼 수 있습니다. 지정되지 않은 속성은 SYSYTEM.DEFAULT.MODEL.QUEUE 기본 큐에서 자동적으로 복사됩니다.

로컬 큐로 사용하는 동일한 방법으로 모델 큐를 정의할 때 REPLACE *YES 속성을 사용할 수 있습니다.

모델 큐로 기타 명령 사용

적절한 MQSC 명령을 사용하여 모델 큐의 속성을 표시하거나 대체할 수 있거나 모델 큐 오브젝트를 삭제할 수 있습니다. 예를 들면, 다음과 같습니다.

다음 명령을 사용하여 모델 큐의 속성을 표시하십시오.

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

다음 명령을 사용하여 이 모델에서 작성되는 동적 큐에 배치할 수 있도록 모델을 대체하십시오.

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

다음 명령을 사용하여 이 모델 큐를 삭제하십시오.

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

관리 토픽에 대한 작업

MQSC 명령을 사용하여 관리 토픽을 관리하십시오.

이러한 명령에 대한 자세한 정보는 MQSC 명령을 참조하십시오.

관련 개념

102 페이지의 『관리 토픽 정의』
MQSC 명령 **DEFINE TOPIC**을 사용하여 관리 토픽을 작성하십시오. 관리 토픽을 정의할 때,각 토픽 속성을 선택적으로 설정할 수 있습니다.

102 페이지의 『관리 토픽 오브젝트 속성 표시』
MQSC 명령 **DISPLAY TOPIC**을 사용하여 관리 토픽 오브젝트를 표시하십시오.

103 페이지의 『관리 토픽 속성 변경』
ALTER TOPIC 명령 또는 **DEFINE TOPIC** 명령을 **REPLACE** 속성과 함께 사용하여 두 가지 방법으로 토픽 속성을 변경할 수 있습니다.

103 페이지의 『관리 토픽 정의 복사』
DEFINE 명령에서 LIKE 속성을 사용하여 토픽 정의를 복사할 수 있습니다.

104 페이지의 『관리 토픽 정의 삭제』
MQSC 명령 **DELETE TOPIC**을 사용하여 관리 토픽을 삭제할 수 있습니다.

관련 정보

관리 토픽 오브젝트

관리 토픽 정의

MQSC 명령 **DEFINE TOPIC**을 사용하여 관리 토픽을 작성하십시오. 관리 토픽을 정의할 때,각 토픽 속성을 선택적으로 설정할 수 있습니다.

명백하게 설정되지 않는 토픽의 속성도 시스템 설치가 설치되었을 때 작성된 기본 관리 토픽, SYSTEM.DEFAULT.TOPIC으로부터 상속됩니다.

예를 들어, 뒤따르는 **DEFINE TOPIC** 명령은 **ORANGE.TOPIC**이라는 토픽을 이러한 특성으로 정의합니다.

- 토픽 문자열 ORANGE로 해결됩니다. 토픽 문자열을 사용하는 방법에 대한 정보는 토픽 문자열 결합을 참조하십시오.
- ASPARENT로 설정되는 속성은 이 토픽의 상위 토픽에 의해 정의된 대로 속성을 사용합니다. 이 조치는 루트 토픽(SYSTEM.BASE.TOPIC)이 발견되는 한 토픽 트리가 반복됩니다. 자세한 정보는 토픽 트리를 참조하십시오.

```
DEFINE TOPIC (ORANGE.TOPIC) +
TOPICSTR (ORANGE) +
DEFPRTY(ASPARENT) +
NPMSGDLV(ASPARENT)
```

참고:

- 토픽 문자열의 값을 제외하고, 표시되는 모든 속성값은 기본값입니다. 이는 설명으로만 여기에 표시됩니다. 기본값이 원하는 것이거나 변경되지 않는 것이라고 확인하는 경우 생략할 수 있습니다. 또한 102 페이지의 『관리 토픽 오브젝트 속성 표시』 도 참조하십시오.
- 이름 ORANGE.TOPIC으로 동일한 큐 관리자에 이미 관리 토픽이 있는 경우, 이 명령은 실패합니다. 토픽의 기존 정의를 덮어쓰려는 경우 REPLACE 속성을 사용하되, 103 페이지의 『관리 토픽 속성 변경』의 내용도 참조하십시오.

관리 토픽 오브젝트 속성 표시

MQSC 명령 **DISPLAY TOPIC**을 사용하여 관리 토픽 오브젝트를 표시하십시오.

모든 토픽을 표시하려면 다음을 사용하십시오.

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

개별적으로 지정하여 선택적으로 속성을 표시할 수 있습니다. 예를 들면, 다음과 같습니다.

```
DISPLAY TOPIC(ORANGE.TOPIC) +
TOPICSTR +
DEFPRTY +
NPMSGDLV
```

이 명령은 다음과 같이 세 개의 지정된 속성을 표시합니다.

```
AMQ8633: Display topic details.
     TOPIC(ORANGE.TOPIC)                                     TYPE(LOCAL)
     TOPICSTR(ORANGE)                                           DEFPRTY(ASPARENT)
     NPMSGDLV(ASPARENT)
```

런타임 시 사용되는 것처럼 토픽 ASPARENT 값을 표시하려면 DISPLAY TPSTATUS를 사용하십시오. 예를 들어, 다음을 사용하십시오.

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

명령은 다음 세부사항을 표시합니다.

```
AMQ8754: Display topic status details.
```

```
TOPICSTR(ORANGE)            DEFPRTY(0)
NPMSGDLV(ALLAVAIL)
```

관리 토픽을 정의할 때, 기본 관리 토픽에서 명시적으로 지정하지 않는 속성을 사용합니다. 이는 SYSTEM.DEFAULT.TOPIC이라고 합니다. 이러한 기본 속성이 무엇인지 보려면 다음 명령을 사용하십시오.

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

관리 토픽 속성 변경

ALTER TOPIC 명령 또는 **DEFINE TOPIC** 명령을 **REPLACE** 속성과 함께 사용하여 두 가지 방법으로 토픽 속성을 변경할 수 있습니다.

예를 들어, ORANGE.TOPIC이라는 토픽으로 전달된 메시지의 기본 우선순위를 5로 변경하려는 경우, 다음 명령 중 하나를 사용하십시오.

- **ALTER** 명령 사용:

  ```
  ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
  ```

  이 명령은 하나의 속성, 즉 이 토픽에 전달된 메시지의 우선순위를 5로 변경합니다. 다른 모든 속성들은 그대로 남습니다.
- **DEFINE** 명령 사용:

  ```
  DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
  ```

  이 명령은 이 토픽으로 전달된 메시지의 기본 우선순위를 변경합니다. 기타 모든 속성에는 기본값이 제공됩니다.

이 토픽에 보내진 메시지의 우선순위를 변경하면, 기존 메시지는 영향을 받지 않습니다. 그러나, 발행 애플리케이션에서 제공하지 않으면 새 메시지는 지정된 우선순위를 사용합니다.

관리 토픽 정의 복사

DEFINE 명령에서 LIKE 속성을 사용하여 토픽 정의를 복사할 수 있습니다.

예를 들면, 다음과 같습니다.

```
DEFINE TOPIC (MAGENTA.TOPIC) +
LIKE (ORANGE.TOPIC)
```

이 명령은 시스템 기본 관리 토픽의 속성보다 오히려, 최초 토픽(ORANGE.TOPIC)과 동일한 속성으로 토픽(MAGENTA.TOPIC)을 작성합니다. 토픽을 작성했을 때 입력된 것처럼 정확하게 복사될 토픽의 이름을 입력하십시오. 이름이 소문자를 포함하면, 이름을 작은따옴표로 묶으십시오.

DEFINE 명령의 이 양식을 사용하여 토픽 정의를 복제할 수도 있지만, 원래 속성으로 변경하십시오. 예를 들면, 다음과 같습니다.

```
DEFINE TOPIC(BLUE.TOPIC) +
TOPICSTR(BLUE) +
LIKE(ORANGE.TOPIC)
```

또한 토픽 BLUE.TOPIC에서 토픽 GREEN.TOPIC의 속성을 복사하고 발행물이 해당 정확한 구독자 큐에 전달될 수 없을 때 데드-레터 큐 위에 배치되지 않는 것을 지정할 수 있습니다. 예를 들면, 다음과 같습니다.

```
DEFINE TOPIC(GREEN.TOPIC) +
TOPICSTR(GREEN) +
LIKE(BLUE.TOPIC) +
USEDLQ(NO)
```

관리 토픽 정의 삭제

MQSC 명령 **DELETE TOPIC**을 사용하여 관리 토픽을 삭제할 수 있습니다.

```
DELETE TOPIC(ORANGE.TOPIC)
```

애플리케이션이 발행을 위한 토픽을 더 이상 열 수 없거나 오브젝트 이름(ORANGE.TOPIC)을 사용하여 새 구독을 더 이상 작성하지 않습니다. 토픽을 열어 둔 발행 애플리케이션은 해석된 토픽 문자열을 계속 발행할 수 있습니다. 이미 이 토픽에 작성된 구독은 토픽이 삭제된 이후에도 계속 발행물을 수신합니다.

이 토픽 오브젝트를 참조하지 않지만 이 토픽 오브젝트가, 이 예에서'ORANGE'로, 표시하는 해석된 토픽 문자열을 사용 중인 애플리케이션은 계속 작동합니다. 이 경우, 토픽 트리에 있는 더 높은 상위 토픽 오브젝트로부터 특성을 상속합니다. 자세한 정보는 토픽 트리를 참조하십시오.

구독에 대한 작업

MQSC 명령을 사용하여 구독을 관리하십시오.

구독은 **SUBTYPE** 속성에서 정의되는 세 가지 유형 중 하나가 될 수 있습니다.

ADMIN
사용자에 의해 관리적으로 정의됩니다.

PROXY
큐 관리자 사이에 발행물을 라우팅하기 위해 내부적으로 작성된 구독.

API
예를 들어, MQI MQSUB 호출을 사용하여 프로그래밍 방식으로 작성됩니다.

이러한 명령에 대한 자세한 정보는 MQSC 명령을 참조하십시오.

관련 개념

104 페이지의 『관리 구독 정의』
MQSC 명령 **DEFINE SUB**를 사용하여 관리 구독을 작성하십시오. 기본 로컬 구독 정의에 정의되는 기본값을 사용할 수도 있습니다. 또는, 시스템이 설치될 때 작성되는 기본 로컬 구독(SYSTEM.DEFAULT.SUB)의 특성에서 구독 특성을 수정할 수 있습니다.

105 페이지의 『구독의 속성 표시』
DISPLAY SUB 명령을 사용하여 큐 관리자에 알려진 구독의 구성된 속성을 표시할 수 있습니다.

106 페이지의 『로컬 구독 속성 변경』
ALTER SUB 명령 또는 **DEFINE SUB** 명령을 **REPLACE** 속성과 함께 사용하여 두 가지 방법으로 등록 속성을 변경할 수 있습니다.

106 페이지의 『로컬 구독 정의 복사』
DEFINE 명령에서 **LIKE** 속성을 사용하여 구독 정의를 복사할 수 있습니다.

106 페이지의 『구독 삭제』
MQSC 명령 **DELETE SUB**을 사용하여 로컬 구독을 삭제할 수 있습니다.

관리 구독 정의

MQSC 명령 **DEFINE SUB**를 사용하여 관리 구독을 작성하십시오. 기본 로컬 구독 정의에 정의되는 기본값을 사용할 수도 있습니다. 또는, 시스템이 설치될 때 작성되는 기본 로컬 구독(SYSTEM.DEFAULT.SUB)의 특성에서 구독 특성을 수정할 수 있습니다.

예를 들어, 뒤따르는 **DEFINE SUB** 명령은 ORANGE라는 구독을 다음 특성으로 정의합니다.

- 무제한 만기로 큐 관리자 재시작에 대해 지속적임을 의미하는 지속 가능한 구독입니다.
- 발행 애플리케이션에 의해 설정된 메시지 우선순위를 갖는 ORANGE 토픽 문자열에 만들어진 발행물을 수신합니다.
- 이 구독에 전달된 발행물은 로컬 큐 SUBQ에 전송됩니다. 이 큐는 구독의 정의 전에 정의되어야 합니다.

```
DEFINE SUB (ORANGE) +
TOPICSTR (ORANGE) +
DESTCLAS (PROVIDED) +
DEST (SUBQ) +
EXPIRY (UNLIMITED) +
PUBPRTY (ASPUB)
```

참고:

- 구독과 토픽 문자열 이름은 일치할 필요가 없습니다.
- 목적지 및 토픽 문자열의 값을 제외하고, 표시되는 모든 속성 값은 기본값입니다. 이는 설명으로만 여기에 표시됩니다. 기본값이 원하는 것이거나 변경되지 않는 것이라고 확인하는 경우 생략할 수 있습니다. 105 페이지의 『구독의 속성 표시』의 내용도 참조하십시오.
- 이름 ORANGE를 가지는 동일한 큐 관리자에 로컬 구독이 이미 있는 경우, 이 명령은 실패합니다. 큐의 기존 정의를 겹쳐쓰지만 106 페이지의 『로컬 구독 속성 변경』도 참조하려면 **REPLACE** 속성을 사용하십시오.
- 큐 SUBQ가 존재하지 않으면 이 명령은 실패합니다.

구독의 속성 표시

DISPLAY SUB 명령을 사용하여 큐 관리자에 알려진 구독의 구성된 속성을 표시할 수 있습니다.

예를 들어, 다음을 사용하십시오.

```
DISPLAY SUB (ORANGE)
```

개별적으로 지정하여 선택적으로 속성을 표시할 수 있습니다. 예를 들면, 다음과 같습니다.

```
DISPLAY SUB (ORANGE) +
SUBID +
TOPICSTR +
DURABLE
```

이 명령은 다음과 같이 세 개의 지정된 속성을 표시합니다.

```
AMQ8096: IBM MQ subscription inquired.
     SUBID(414D5120414141202020202020202020EE921E4E20002A03)
     SUB(ORANGE)                                                    TOPICSTR(ORANGE)
     DURABLE(YES)
```

TOPICSTR은 이 구독자가 운영되는 해석된 토픽 문자열입니다. 구독이 토픽 오브젝트를 사용하기 위해 정의될 때, 해당 오브젝트의 토픽 문자열은 구독을 작성할 때 제공되는 토픽 문자열에 대한 접두부로 사용됩니다. SUBID는 구독이 작성될 때 큐 관리자에 의해 지정되는 고유한 ID입니다. 이는 일부 구독 이름이 길거나 비실용적일 수 있는 다른 문자 세트에 있으므로 표시하기에 유용한 속성입니다.

구독을 표시하기 위한 대체 방법은 SUBID를 사용하는 것입니다.

```
DISPLAY SUB +
SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
TOPICSTR +
DURABLE
```

이 명령은 이전 것과 동일한 다음과 같은 출력을 제공합니다.

```
AMQ8096: IBM MQ subscription inquired.
SUBID(414D5120414141202020202020202020EE921E4E20002A03)
     SUB(ORANGE)                                                    TOPICSTR(ORANGE)
     DURABLE(YES)
```

큐 관리자의 프록시 구독은 기본적으로 표시되지 않습니다. 이를 표시하려면 PROXY 또는 ALL의 **SUBTYPE**을 지정하십시오.

DISPLAY SBSTATUS 명령을 사용하여 런타임 속성을 표시할 수 있습니다. 예를 들어, 다음 명령을 사용하십시오.

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

다음 출력이 표시됩니다.

```
AMQ8099: IBM MQ subscription status inquired.
SUB(ORANGE)
SUBID(414D5120414141202020202020202020EE921E4E20002A03)
NUMMSGS(0)
```

관리 구독을 정의할 때, 기본 구독에서 명시적으로 지정하지 않는 속성을 사용하고 이는 SYSTEM.DEFAULT.SUB 이라고 합니다. 이러한 기본 속성이 무엇인지 보려면 다음 명령을 사용하십시오.

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

로컬 구독 속성 변경

ALTER SUB 명령 또는 **DEFINE SUB** 명령을 **REPLACE** 속성과 함께 사용하여 두 가지 방법으로 등록 속성을 변경할 수 있습니다.

예를 들어, ORANGE라고 하는 구독으로 전달된 메시지의 우선순위가 5가 되도록 하려는 경우, 다음 명령 중 하나를 사용하십시오.

- ALTER 명령 사용:

  ```
  ALTER SUB(ORANGE) PUBPRTY(5)
  ```

  이 명령은 하나의 속성, 즉 이 구독에 전달된 메시지의 우선순위를 5로 변경합니다. 다른 모든 속성들은 그대로 남습니다.
- DEFINE 명령 사용:

  ```
  DEFINE SUB (ORANGE) PUBPRTY(5) REPLACE
  ```

  이 명령은 이 구독으로 전달된 메시지의 우선순위가 아니라, 기본값이 제공되는 기타 모든 속성을 변경합니다.

이 구독에 보내진 메시지의 우선순위를 변경하면, 기존 메시지는 영향을 받지 않습니다. 그러나, 새 메시지도 지정된 우선권을 가집니다.

로컬 구독 정의 복사

DEFINE 명령에서 **LIKE** 속성을 사용하여 구독 정의를 복사할 수 있습니다.

예를 들면, 다음과 같습니다.

```
DEFINE SUB (BLUE) +
LIKE (ORANGE)
```

서브 REAL의 속성을 서브 THIRD.SUB로 복사하고 전달된 발행물의 correlID가 발행자 correlID라기 보다는 THIRD라고 지정할 수 있습니다. 예를 들면, 다음과 같습니다.

```
DEFINE SUB(THIRD.SUB) +
LIKE(BLUE) +
DESTCORL(ORANGE)
```

구독 삭제

MQSC 명령 **DELETE SUB**을 사용하여 로컬 구독을 삭제할 수 있습니다.

```
DELETE SUB(ORANGE)
```

SUBID를 사용하여 구독을 삭제할 수도 있습니다.

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

구독에서 메시지 검사

이 태스크 정보

구독이 정의될 때 큐와 연관됩니다. 이 구독과 일치하는 발행된 메시지가 이 큐에 놓입니다.

다음 **runmqsc** 명령이 메시지를 받은 해당 구독만 보여준다는 점을 참고하십시오.

구독을 위해 최근에 큐되는 메시지를 검사하려면 다음 단계를 수행하십시오.

프로시저

1. 구독 유형 **DISPLAY SBSTATUS(<sub_name>) NUMMSGS**에 대해 큐되는 메시지를 검사하려면 105 페이지의 『구독의 속성 표시』의 내용을 참조하십시오.
2. **NUMMSGS** 값이 0보다 더 크면 **DISPLAY SUB(<sub_name>)DEST**를 입력하여 구독과 연관된 큐를 식별하십시오.
3. 리턴된 큐의 이름을 사용하면서 80 페이지의 『큐 찾아보기』에서 설명된 기술을 수행하여 메시지를 볼 수 있습니다.

서비스에 대한 작업

서비스 오브젝트는 추가 프로세스가 큐 관리자의 일부로서 관리될 수 있는 수단입니다. 서비스를 사용하면 큐 관리자가 시작되고 종료될 때 시작되고 중지되는 프로그램을 정의할 수 있습니다. IBM MQ 서비스는 큐 관리자를 시작한 사용자의 사용자 ID에서 항상 시작됩니다.

새 IBM MQ 서비스 정의를 정의하려면 MQSC 명령 DEFINE SERVICE를 사용하십시오.

서비스 오브젝트는 다음 유형 중 하나가 될 수 있습니다.

SERVER
서버는 SERVER로 지정된 매개변수 SERVTYPE을 가지는 서비스 오브젝트입니다. 서버 서비스 오브젝트는 지정된 큐 관리자가 시작될 때 실행되는 프로그램의 정의입니다. 서버 서비스 오브젝트는 일반적으로 오랜 기간 동안 실행되는 프로그램을 정의합니다. 예를 들어, 서버 서비스 오브젝트는 트리거 모니터 프로세스를 실행하기 위해 사용될 수 있습니다(예:`runmqtrm`).

서버 서비스 오브젝트의 하나의 인스턴스만 동시에 실행될 수 있습니다. 실행 중인 서버 서비스 오브젝트의 상태는 MQSC 명령(DISPLAY SVSTATUS)을 사용하여 모니터될 수 있습니다.

명령
명령은 COMMAND로 지정된 매개변수 SERVTYPE을 가지는 서비스 오브젝트입니다. 명령 서비스 오브젝트는 서버 서비스 오브젝트와 유사하지만, 명령 서비스 오브젝트의 다중 인스턴스는 동시에 실행될 수 있고, 해당 상태가 MQSC 명령 DISPLAY SVSTATUS를 사용하여 모니터될 수 없습니다.

MQSC 명령 STOP SERVICE를 실행하면 MQSC 명령 START SERVICE로 시작된 프로그램이 중지 프로그램을 실행하기 전에 활성 상태인지 확인할 수 없습니다.

서비스 오브젝트 정의

다양한 속성으로 서비스 오브젝트를 정의합니다.

속성은 다음과 같습니다.

SERVTYPE
서비스 오브젝트의 유형을 정의합니다. 가능한 값은 다음과 같습니다.

SERVER
서버 서비스 오브젝트.

한 번에 하나의 서버 서비스 오브젝트 인스턴스를 실행할 수 있습니다. 서버 서비스 오브젝트의 상태는 MQSC 명령(DISPLAY SVSTATUS)을 사용하여 모니터링될 수 있습니다.

명령
명령 서비스 오브젝트.

명령 서비스 오브젝트의 다중 인스턴스가 동시에 실행될 수 있습니다. 명령 서비스 오브젝트의 상태는 모니터링될 수 없습니다.

STARTCMD
서비스를 시작하기 위해 실행되는 프로그램. 프로그램으로에 대한 완전한 경로가 지정되어야 합니다.

STARTARG
시작 프로그램으로 전달되는 인수.

STDERR
서비스 프로그램의 표준 오류(stderr)의 경로가 재지정되어야 하는 파일에 경로를 지정합니다.

STDOUT
서비스 프로그램의 표준 출력(stdout)의 경로가 재지정되어야 하는 파일에 경로를 지정합니다.

STOPCMD
서비스를 중지하기 위해 실행되는 프로그램. 프로그램으로에 대한 완전한 경로가 지정되어야 합니다.

STOPARG
중지 프로그램으로 전달되는 인수.

CONTROL
서비스가 시작되고 정지되는 방식을 지정합니다.

MANUAL
서비스가 자동으로 시작되거나 자동으로 중지되지 않습니다. START SERVICE 및 STOP SERVICE 명령을 사용하여 제어됩니다. 이 값은 기본값입니다.

큐 관리자
큐 관리자가 시작되고 중지되는 것과 동시에 정의한 서비스가 시작되고 중지됩니다.

STARTONLY
큐 관리자가 시작되는 것과 동시에 서비스가 시작되지만 큐 관리자가 중지될 때 중지가 요청되지 않습니다.

관련 개념
108 페이지의 『서비스 관리』
CONTROL 매개변수를 사용하여 서비스 오브젝트의 인스턴스는 큐 관리자에 의해 자동으로 시작되고 중지될 수 있거나 MQSC 명령 START SERVICE 및 STOP SERVICE를 사용하여 시작되고 중지될 수 있습니다.

서비스 관리

CONTROL 매개변수를 사용하여 서비스 오브젝트의 인스턴스는 큐 관리자에 의해 자동으로 시작되고 중지될 수 있거나 MQSC 명령 START SERVICE 및 STOP SERVICE를 사용하여 시작되고 중지될 수 있습니다.

서비스 오브젝트의 인스턴스가 시작될 때, 서비스 오브젝트의 이름 및 시작된 프로세스의 프로세스 ID를 포함하는 큐 관리자 오류 로그로 메시지가 작성됩니다. 서버 서비스 오브젝트 시작의 예 로그 항목이 그 뒤에 표시됩니다.

```
02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

EXPLANATION:
The Server process has started.
ACTION:
None.
```

명령 서비스 오브젝트 시작의 예 로그 항목이 그 뒤에 표시됩니다.

```
02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

EXPLANATION:
The Command has started.
ACTION:
None.
```

인스턴스 서버 서비스가 중지될 때, 서비스의 이름 및 종료 프로세스의 프로세스 ID를 포함하는 큐 관리자 오류 로그에 메시지가 작성됩니다. 서버 서비스 오브젝트 중지에 대한 예 로그 항목이 그 뒤에 표시됩니다.

```
02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

EXPLANATION:
The Server process has ended.
ACTION:
None.
```

관련 참조

109 페이지의 『추가 환경 변수』

서비스가 시작될 때, 서비스 프로세스가 시작되는 환경은 큐 관리자의 환경으로부터 상속됩니다. service.env 환경 대체 파일 중 하나에 정의하려는 변수를 추가하여 서비스 프로세스의 환경에 설정될 추가 환경 변수를 정의할 수 있습니다.

추가 환경 변수

서비스가 시작될 때, 서비스 프로세스가 시작되는 환경은 큐 관리자의 환경으로부터 상속됩니다. service.env 환경 대체 파일 중 하나에 정의하려는 변수를 추가하여 서비스 프로세스의 환경에 설정될 추가 환경 변수를 정의할 수 있습니다.

참고:

환경 변수를 추가할 수 있는 두 개의 가능한 파일이 있습니다.

- 유닉스 및 Linux 시스템의 /var/mqm 또는 Windows 시스템의 설치 동안 선택되는 데이터 디렉토리에 있는 시스템 범위 service.env 파일.
- 큐 관리자 데이터 디렉토리에 있는 큐 관리자 범위 service.env 파일. 예를 들어, QMNAME이라는 큐 관리자에 대한 환경 대체 파일의 위치는 다음과 같습니다.
  - 유닉스 및 Linux 시스템의 경우, /var/mqm/qmgrs/QMNAME/service.env
  - Windows 시스템의 경우, C:\ProgramData\IBM\MQ\qmgrs\QMNAME\service.env

가능한 경우, 시스템 범위 파일에서 해당 정의보다 우선순위에 서는 큐 관리자 범위 파일의 정의로 양쪽 모든 파일이 처리됩니다.

환경 변수가 service.env에 지정될 수 있습니다. 예를 들어, IBM MQ 서비스가 수많은 명령을 실행시키는 경우, service.env 파일에서 PATH 사용자 변수를 설정하는 것은 유용합니다. 변수를 설정하는 값은 환경 변수가 될 수 없습니다. 예를 들어, CLASSPATH= *%CLASSPATH%*는 올바르지 않습니다. 마찬가지로, Linux PATH= *$PATH* :/opt/mqm/bin는 예상치 못한 결과를 발생시킵니다.

CLASSPATH는 대문자로 표시되어야 하고 클래스 경로 명령문은 리터럴만을 포함할 수 있습니다. 일부 서비스(예: 텔레메트리)는 자체 클래스 경로를 설정합니다. service.env에 정의된 CLASSPATH는 해당 파일에 추가됩니다.

파일 service.env에 정의된 변수의 형식은 이름 및 값 변수 쌍의 목록입니다. 각 변수는 새 행에 정의되어야 하고 공백 문자를 포함하여 명백하게 정의된 것처럼 각 변수가 사용됩니다. 파일의 예 service.env는 다음을 따릅니다.

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2022.                        *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                  *#
#*                                                                  *#
#********************************************************************#
#********************************************************************#
#* Module Name: service.env                                         *#
#* Type       : IBM MQ service environment file                     *#
#* Function   : Define additional environment variables to be set   *#
#*              for SERVICE programs.                               *#
#* Usage      : <VARIABLE>=<VALUE>                                  *#
#*                                                                  *#
#********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

관련 참조

110 페이지의 『서비스 정의에서 대체 가능한 삽입』
서비스 오브젝트의 정의에서 토큰을 대체할 수 있습니다. 대체되는 토큰은 서비스 프로그램이 실행될 때 확장된 텍스트로 자동적으로 대체됩니다. 대체 토큰은 공통 토큰의 다음 목록 또는 service.env 파일에 정의되는 변수에서 가져올 수 있습니다.

서비스 정의에서 대체 가능한 삽입

서비스 오브젝트의 정의에서 토큰을 대체할 수 있습니다. 대체되는 토큰은 서비스 프로그램이 실행될 때 확장된 텍스트로 자동적으로 대체됩니다. 대체 토큰은 공통 토큰의 다음 목록 또는 service.env 파일에 정의되는 변수에서 가져올 수 있습니다.

다음은 서비스 오브젝트의 정의에서 토큰을 대체하기 위해 사용할 수 있는 공통 토큰입니다.

MQ_INSTALL_PATH
IBM MQ이 설치된 위치입니다.

MQ_DATA_PATH
IBM MQ 데이터 디렉토리의 위치입니다.

- 유닉스 및 Linux 시스템에서 IBM MQ 데이터 디렉토리 위치는 /var/mqm/입니다.
- Windows 시스템에서 IBM MQ 데이터 디렉토리의 위치는 IBM MQ의 설치 중에 선택된 데이터 디렉토리입니다.

QMNAME
현재 큐 관리자 이름입니다.

MQ_SERVICE_NAME
서비스 이름입니다.

MQ_SERVER_PID
이 토큰은 STOPARG 및 STOPCMD 인수에 의해서만 사용될 수 있습니다.

서버 서비스 오브젝트의 경우, 이 토큰은 STARTCMD 및 STARTARG 인수에서 시작하는 프로세스의 프로세스 ID로 대체됩니다. 그렇지 않으면, 이 토큰은 0으로 대체됩니다.

MQ_Q_MGR_DATA_PATH
큐 관리자 데이터 디렉토리의 위치입니다.

MQ_Q_MGR_DATA_NAME
큐 관리자의 변환된 이름입니다. 이름 변환에 대한 자세한 정보는 IBM MQ 파일 이름 이해를 참조하십시오.

대체가능한 삽입을 사용하려면, 임의의 STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT 또는 STDERR 문자열로 + 문자 내에 토큰을 삽입하십시오. 이것의 예는 111 페이지의 『서비스 오브젝트 사용 예』의 내용을 참조하십시오.

서비스 오브젝트 사용 예

이 섹션의 서비스는 달리 정해진 경우를 제외하고 UNIX 스타일 경로 구문 기호 문자로 작성됩니다.

서버 서비스 오브젝트 사용

이 예에는 트리거 모니터를 시작하기 위해 서버 서비스 오브젝트를 정의, 사용 및 변경하는 방법이 표시됩니다.

1. 다음 MQSC 명령을 사용하여, 서버 서비스 오브젝트가 정의됩니다.

```
DEFINE SERVICE(S1) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

   여기서:

   +MQ_INSTALL_PATH+는 설치 디렉토리를 나타내는 토큰입니다.

   +QMNAME+은 큐 관리자의 이름을 표시하는 토큰입니다.

   ACCOUNTS.INITIATION.QUEUE는 이니시에이션 큐입니다.

   amqsstop은 큐 관리자가 프로세스 ID에 대한 모든 연결을 중단하도록 요청하는 IBM MQ와 함께 제공되는 샘플 프로그램입니다. amqsstop은 PCF 명령을 생성합니다. 따라서 명령 서버가 실행되어야 합니다.

   +MQ_SERVER_PID+는 중지 프로그램에 전달된 프로세스 ID를 나타내는 토큰입니다.

   공통 토큰의 목록의 경우 110 페이지의 『서비스 정의에서 대체 가능한 삽입』의 내용을 참조하십시오.

2. 큐 관리자가 다음에 시작될 때 서버 서비스 오브젝트의 인스턴스가 실행됩니다. 그러나, 다음 MQSC 명령으로 즉각적으로 서버 서비스 오브젝트의 인스턴스를 시작합니다.

```
START SERVICE(S1)
```

3. 다음 MQSC 명령을 사용하여 서버 서비스 오브젝트의 상태가 표시됩니다.

```
DISPLAY SVSTATUS(S1)
```

4. 이 예는 서버 서비스 프로세스를 수동으로 재시작하여 업데이트를 선택하고 서버 서비스 오브젝트를 변경하는 방법에 대해 표시합니다. 이니시에이션 큐가 JUPITER.INITIATION.QUEUE로 지정되도록 서버 서비스 오브젝트가 대체됩니다. 다음 MQSC 명령이 사용됩니다.

```
ALTER SERVICE(S1) +
STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

   참고: 실행 중인 서비스는 재시작될 때까지 서비스 정의에 대한 업데이트를 선택하지 않습니다.

5. 대체가 다음 MQSC 명령을 사용하여 선택되도록 서버 서비스 프로세스가 재시작됩니다.

```
STOP SERVICE(S1)
```

   그 후에 다음 명령을 입력하십시오.

```
START SERVICE(S1)
```

   서버 서비스 프로세스가 재시작되고 111 페이지의 『4』에서 작성된 변경을 선택합니다.

참고: MQSC 명령(STOP SERVICE)은 STOPCMD 인수가 서비스 정의에 지정되는 경우에만 사용될 수 있습니다.

명령 서비스 오브젝트 사용

이 예는 큐 관리자가 시작되거나 중지될 때 운영 체제의 시스템 로그에 입력 항목을 작성하는 프로그램을 시작하기 위한 명령 서비스 오브젝트를 정의하는 방법에 대해 표시합니다.

1. 명령 서비스 오브젝트는 다음 MQSC 명령을 사용하여 정의됩니다.

```
DEFINE SERVICE(S2) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STARTCMD('/usr/bin/logger') +
STARTARG('Queue manager +QMNAME+ starting') +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

여기서:

logger는 시스템 로그에 작성하기 위한 유닉스 및 Linux 시스템 공급 명령입니다.
+QMNAME+은 큐 관리자의 이름을 표시하는 토큰입니다.

큐 관리자가 종료될 때만 명령 서비스 오브젝트 사용

이 예는 큐 관리자가 중지될 때에만 운영 체제의 시스템 로그에 입력 항목을 작성하는 프로그램을 시작하기 위해 명령 서비스 오브젝트를 정의하는 방법을 표시합니다.

1. 명령 서비스 오브젝트는 다음 MQSC 명령을 사용하여 정의됩니다.

```
DEFINE SERVICE(S3) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

여기서:

logger 는 운영 체제의 시스템 로그에 항목을 기록할 수 있는 IBM MQ 와 함께 제공되는 샘플 프로그램입니다.
+QMNAME+은 큐 관리자의 이름을 표시하는 토큰입니다.

인수 전달에 대한 자세한 내용

이 예제는 큐 관리자가 시작될 때 runserv라는 프로그램을 시작하도록 서버 서비스 오브젝트를 정의하는 방법에 대해 설명합니다.

이 예제는 Windows 스타일 경로 구분 기호 문자로 작성됩니다.

시작 프로그램에 전달되는 인수 중 하나는 공백을 포함하는 문자열입니다. 이 인수는 단일 문자열로서 전달되어야 합니다. 이를 달성하려면 큰따옴표가 명령 서비스 오브젝트를 정의하기 위해 다음 명령에 표시된 대로 사용됩니다.

1. 다음 MQSC 명령을 사용하여, 서버 서비스 오브젝트가 정의됩니다.

```
DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
STARTCMD('C:\Program Files\Tools\runserv.exe') +
STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')


DEFINE SERVICE(S4) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('C:\Program Files\Tools\runserv.exe') +
STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
```

여기서:

+QMNAME+은 큐 관리자의 이름을 표시하는 토큰입니다.

"C:\Program Files\Tools\"는 공백을 포함하는 문자열이며, 단일 문자열로서 전달됩니다.

서비스 자동 시작

이 예는 큐 관리자가 시작될 때 자동으로 트리거 모니터를 시작하기 위해 사용할 수 있는 서버 서비스 오브젝트를 정의하는 방법을 표시합니다.

1. 다음 MQSC 명령을 사용하여, 서버 서비스 오브젝트가 정의됩니다.

```
DEFINE SERVICE(TRIG_MON_START) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('runmqtrm') +
STARTARG('-m +QMNAME+ -q +IQNAME+')
```

여기서:

+QMNAME+은 큐 관리자의 이름을 표시하는 토큰입니다.

+IQNAME+은 이니시에이션 큐의 이름을 표시하는 service.env 파일 중 하나에서 사용자에 의해 정의된 환경 변수입니다.

트리거에 대한 오브젝트 관리

IBM MQ를 사용하면 큐의 특정 조건이 충족될 때 애플리케이션을 자동으로 시작할 수 있습니다. 예를 들어, 큐의 메시지 수가 지정된 수에 도달할 때 애플리케이션을 시작하려고 할 수 있습니다. 이 기능은 트리거라고 합니다. 트리거를 지원하는 오브젝트를 정의해야 합니다.

트리거는 트리거를 사용하여 IBM MQ 애플리케이션 시작에서 자세하게 설명됩니다.

트리거에 대한 애플리케이션 큐 정의

애플리케이션 큐는 MQI를 통해 메시징을 위해 애플리케이션에서 사용하는 로컬 큐입니다. 트리거는 애플리케이션 큐에 정의되는 수많은 큐 속성이 필요합니다.

트리거 자체는 *Trigger* 속성 (MQSC 명령에서 TRIGGER) 으로 사용 가능합니다. 이 예제에서, 트리거 이벤트는 다음과 같이 로컬 큐 MOTOR.INSURANCE.QUEUE에 우선순위가 5 이상인 메시지가 100개가 있을 때 생성됩니다.

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
MAXMSGL (2000) +
DEFPSIST (YES) +
INITQ (MOTOR.INS.INIT.QUEUE) +
TRIGGER +
TRIGTYPE (DEPTH) +
TRIGDPTH (100)+
TRIGMPRI (5)
```

설명:

QLOCAL (MOTOR.INSURANCE.QUEUE)
정의되는 애플리케이션 큐의 이름입니다.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
트리거 모니터 프로그램에서 시작하는 애플리케이션을 정의하는 프로세스 정의의 이름입니다.

MAXMSGL (2000)
큐의 최대 길이의 메시지입니다.

DEFPSIST (YES)
이 큐의 메시지가 기본적으로 지속되는지 지정합니다.

INITQ (MOTOR.INS.INIT.QUEUE)
큐 관리자가 트리거 메시지를 배치할 이니시에이션 큐의 이름입니다.

TRIGGER
트리거 속성 값입니다.

TRIGTYPE (DEPTH)
필수 우선순위(TRIGMPRI)의 메시지 수가 TRIGDPTH에 지정된 수에 도달할 때 트리거 이벤트가 생성된다는 것을 지정합니다.

TRIGDPTH (100)
트리거 이벤트를 생성하기 위해 필요한 메시지 수입니다.

TRIGMPRI (5)
트리거 이벤트를 생성할지 여부를 결정하는 데 있어서 큐 관리자가 계수해야 하는 메시지의 우선순위입니다. 우선순위 5 이상인 메시지 수만 계수됩니다.

이니시에이션 큐 정의

트리거 이벤트가 발생할 때, 큐 관리자는 애플리케이션 큐 정의에 지정되는 이니시에이션 큐에 트리거 메시지를 배치합니다. 이니시에이션 큐에는 특수 설정이 없지만, 자세한 내용을 위해 로컬 큐 MOTOR.INS.INIT.QUEUE의 다음 정의를 사용할 수 있습니다.

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
GET (ENABLED) +
NOSHARE +
NOTRIGGER +
MAXMSGL (2000) +
MAXDEPTH (1000)
```

프로세스 정의

DEFINE PROCESS 명령을 사용하여 프로세스 정의를 작성하십시오. 프로세스 정의는 애플리케이션 큐에서 메시지를 처리하기 위해 사용되는 애플리케이션을 정의합니다. 애플리케이션 큐 정의는 사용될 프로세스의 이름을 지정합니다. 그렇게 함으로써 메시지를 처리하기 위해 사용될 애플리케이션과 애플리케이션 큐를 연관합니다. 이는 애플리케이션 큐 MOTOR.INSURANCE.QUEUE에서 PROCESS 속성을 통해 수행됩니다. 다음 MQSC 명령은 이 예에 식별된 필수 프로세스 MOTOR.INSURANCE.QUOTE.PROCESS를 정의합니다.

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
DESCR ('Insurance request message processing') +
APPLTYPE (UNIX) +
APPLICID ('/u/admin/test/IRMP01') +
USERDATA ('open, close, 235')
```

여기서:

MOTOR.INSURANCE.QUOTE.PROCESS
프로세스 정의의 이름입니다.

DESCR ('Insurance request message processing')
이 정의가 관련되는 애플리케이션 프로그램을 설명합니다. 이 텍스트는 DISPLAY PROCESS 명령을 사용할 때 표시됩니다. 이는 프로세스가 수행하는 것을 식별하는 데 도움이 될 수 있습니다. 문자열에서 공백을 사용하는 경우, 작은따옴표로 문자열을 묶어야 합니다.

APPLTYPE (UNIX)
시작할 애플리케이션의 유형입니다.

APPLICID ('/u/admin/test/IRMP01')
완전한 파일 이름으로 지정되는 애플리케이션 실행 가능 파일의 이름입니다. Windows 시스템에서 일반 APPLICID 값은 `c:\appl\test\irmp01.exe`입니다.

USERDATA ('open, close, 235')
애플리케이션에서 사용할 수 있는 사용자 정의 데이터입니다.

프로세스 정의 속성 표시

DISPLAY PROCESS 명령을 사용하여 사용자 정의 결과를 조사하십시오. 예를 들면, 다음과 같습니다.

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
DESCR ('Insurance request message processing')
APPLICID ('/u/admin/test/IRMP01')
USERDATA (open, close, 235)
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
APPLTYPE (UNIX)
```

MQSC 명령 ALTER PROCESS를 사용하여 기존 프로세스 정의를 대체할 수 있고, DELETE PROCESS 명령을 사용하여 프로세스 정의를 삭제할 수도 있습니다.

두 시스템 사이에 dmpmqmsg 유틸리티 사용

dmpmqmsg 유틸리티(이전 **qload**)는 버전 8.0 제품에 통합되었습니다. 이전에는 **qload** 유틸리티를 SupportPac MO03으로 사용할 수 있었습니다.

개요

dmpmqmsg 유틸리티를 사용하여 큐 또는 해당 메시지의 컨텐츠를 파일에 복사하거나 이동할 수 있습니다. 이 파일은 필요에 따라 다른 곳에 저장할 수 있으며 나중에 메시지를 큐에 다시 로드하기 위해 사용할 수 있습니다.

중요사항: 파일은 유틸리티가 이해할 수 있는 특정 형식을 갖습니다. 그러나 이 파일은 사람이 읽을 수 있으므로 다시 로드하기 전에 편집기로 업데이트할 수 있습니다. 파일을 편집하는 경우 해당 형식을 변경해서는 안 됩니다.

가능한 용도는 다음과 같습니다.

- 큐에 있는 메시지를 파일에 저장(아카이브 용도로 또는 나중에 큐에 다시 로드하기 위해)
- 이전에 파일에 저장한 메시지로 큐 다시 로드
- 큐에서 이전 메시지 제거
- 저장된 위치에서 테스트 메시지 '재실행' - 필요한 경우 메시지 간에 올바른 시간 유지

주의: SupportPac MO03은 로컬 또는 클라이언트 바인딩을 지정하기 위해 **-l** 매개변수를 사용했습니다. **-l**는 **-c** 매개변수로 대체되었습니다.

이제 코드 페이지 정보에는 **-c** 대신 **-P**가 사용됩니다.

명령 및 사용 가능 매개변수에 대한 추가 정보는 dmpmqmsg를 참조하십시오.

Linux에서 Windows 시스템을 사용하여 dmpmqmsg 유틸리티 사용 예

동일한 큐 관리자에서 다른 큐(*Q2*)로 이동하려는 큐(*Q1*)에 메시지가 있는 큐 관리자가 Linux 시스템에 있습니다. Windows 시스템에서 **dmpmqmsg** 유틸리티를 시작하려고 합니다.

큐(*Q1*)에는 샘플 **amqsput**(로컬 큐 관리자) 또는 **amqsputc**(리모트 큐 관리자) 애플리케이션을 사용하여 추가된 네 개의 메시지가 있습니다.

Linux 시스템의 경우 다음과 같이 표시됩니다.

```
display ql(Q1) CURDEPTH
       2 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(4)
```

MQSERVER 환경 변수가 Linux의 큐 관리자를 가리키도록 설정하십시오. 예를 들면, 다음과 같습니다.

```
set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/veracruz.x.com(1414)
```

여기서 *veracruz*는 시스템의 이름입니다.

큐, *Q1*에서 읽을 **dmpmqmsg** 유틸리티를 실행하고 `c:\temp\mqqload.txt`에 출력을 저장하십시오.

MQSERVER로 설정되어 Linux 호스트와 포트에서 실행 중인 큐 관리자, *QM_VER*에 원격 클라이언트로 연결하십시오. 원격 클라이언트로서 연결하려면 `-c` 속성을 사용해야 합니다.

```
dmpmqmsg -m QM_VER -i Q1 -f c:\temp\mqqload.txt -c
Read      - Files:    0   Messages:       4   Bytes:           22
Written - Files:    1   Messages:       4   Bytes:           22
```

출력 파일 `c:\temp\mqqload.txt`에는 **dmpmqmsg** 유틸리티가 이해하는 형식을 사용하는 텍스트가 포함됩니다.

On the Windows machine, issue the **dmpmqmsg** command (using the `[]` option instead of the `-i` option) to load queue (*Q2*) on the Linux machine from a file on the Windows machine:

```
dmpmqmsg -m QM_VER -o Q2 -f c:\temp\mqqload.txt -c
Read      - Files:    1   Messages:       4   Bytes:           22
Written - Files:    0   Messages:       4   Bytes:           22
```

Linux 시스템의 경우, 이제 파일에서 복원된 네 개의 메시지가 큐에 있음에 유의하십시오.

```
display ql(Q2) CURDEPTH
        6 : display ql(Q2) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q2)
TYPE(QLOCAL)
     CURDEPTH(4)
```

Linux 시스템의 경우,

원래 큐에서 메시지를 삭제하십시오.

```
clear qlocal(Q1)
        4 : clear qlocal(Q1)
AMQ8022: IBM MQ queue cleared.
```

원래 큐에 더 이상 메시지가 없는지 확인하십시오.

```
display ql(Q1) CURDEPTH
        5 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(0)
```

명령 및 해당 매개변수에 대한 설명은 dmpmqmsg를 참조하십시오.

관련 개념

116 페이지의 『dmpmqmsg 유틸리티 사용 예』

dmpmqmsg 유틸리티(이전 **qload**)를 사용할 수 있는 간단한 방법입니다. 이 유틸리티는 버전 8.0 제품에 통합되었습니다.

dmpmqmsg 유틸리티 사용 예

dmpmqmsg 유틸리티(이전 **qload**)를 사용할 수 있는 간단한 방법입니다. 이 유틸리티는 버전 8.0 제품에 통합되었습니다.

이전에는 **qload** 유틸리티를 SupportPac MO03으로 사용할 수 있었습니다.

파일에 큐 언로드

명령행에서 다음 옵션을 사용하면 큐에 있는 메시지가 파일에 저장됩니다.

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile
```

이 명령은 큐에서 메시지 사본을 가져와 지정된 필드에 저장합니다.

일련의 파일에 큐 언로드

파일 이름에서 `insert` 문자를 사용하여 일련의 파일에 큐를 언로드할 수 있습니다. 이 모드에서는 각 메시지가 새 파일에 기록됩니다.

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile%n
```

이 명령은 큐를 `myfile1`, `myfile2`, `myfile3` 등의 파일에 언로드합니다.

파일에서 큐 로드

116 페이지의 『파일에 큐 언로드』에 저장한 메시지로 큐를 다시 로드하려면 명령행에서 다음 옵션을 사용하십시오.

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

이 명령은 큐를 `myfile1`, `myfile2`, `myfile3` 등의 파일에 언로드합니다.

일련의 파일에서 큐 로드

파일 이름에서 `insert` 문자를 사용하여 일련의 파일에서 큐를 로드할 수 있습니다. 이 모드에서는 각 메시지가 새 파일에 기록됩니다.

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

이 명령은 큐를 `myfile1`, `myfile2`, `myfile3` 등의 파일에 로드합니다.

특정 큐의 메시지를 다른 큐에 복사

116 페이지의 『파일에 큐 언로드』의 파일 매개변수를 다른 큐 이름으로 바꾸고 다음 옵션을 사용하십시오.

```
dmpmqmsg -m QM1 -i Q1 -o Q2
```

이 명령을 사용하면 특정 큐의 메시지를 다른 큐에 복사할 수 있습니다.

특정 큐의 첫 번째 100개 메시지를 다른 큐에 복사

이전 예제에서 명령을 사용하고 `-r#100` 옵션을 추가하십시오.

```
dmpmqmsg -m QM1 -i Q1 -o Q2 -r#100
```

특정 큐의 메시지를 다른 큐로 이동

117 페이지의 『파일에서 큐 로드』의 변형입니다. 큐를 찾는 경우에만 **-i** (소문자) 를 사용하고 큐에서 소멸되는 **-I** (대문자) 를 사용하는 것 사이의 차이에 유의하십시오.

```
dmpmqmsg -m QM1 -I Q1 -o Q2
```

특정 큐에서 1일이 지난 메시지를 다른 큐로 이동

이 예는 기간 선택 사용 방법을 보여줍니다. 기간보다 오래되거나 오래되지 않았거나 기간 범위 내에 있는 메시지를 선택할 수 있습니다.

```
dmpmqmsg -m QM1 -I Q1 -o Q2 -T1440
```

현재 큐에 있는 메시지의 기간 표시

명령행에서 다음 옵션을 사용하십시오.

```
dmpmqmsg -m QM1 -i Q1 -f stdout -dT
```

메시지 파일에 대한 작업

큐에서 메시지를 언로드하면(116 페이지의 『파일에 큐 언로드』 참조) 파일을 편집하려고 할 수 있습니다.

또한 큐 언로드 시점에 지정하지 않은 표시 옵션 중 하나를 사용하도록 파일 형식을 변경할 수 있습니다.

dmpmqmsg 유틸리티를 사용하면 큐를 언로드한 후에도 파일을 원하는 형식으로 재처리할 수 있습니다. 명령행에서 다음 옵션을 사용하십시오.

```
dmpmqmsg -f c:\oldfile -f c:\newfile -dA
```

명령 및 해당 매개변수에 대한 설명은 dmpmqmsg를 참조하십시오.

원격 IBM MQ 오브젝트 관리

이 섹션은 사용자에게 MQSC 명령을 사용하여 리모트 큐 관리자에 IBM MQ 오브젝트를 관리하는 방법 및 메시지와 응답 메시지의 목적지를 제어하기 위해 멀리 있는 큐 오브젝트를 사용하는 방법을 말합니다.

이 절에서는 다음을 설명합니다.

- 118 페이지의 『채널, 클러스터 및 리모트 큐잉』
- 119 페이지의 『로컬 큐 관리자에서 원격 관리』
- 125 페이지의 『리모트 큐의 로컬 정의 작성』
- 129 페이지의 『알리어스로서 리모트 큐 정의 사용』
- 129 페이지의 『데이터 변환』

채널, 클러스터 및 리모트 큐잉

큐 관리자는 필요한 경우 메시지를 보내고 응답을 다시 수신하여 다른 큐 관리자와 통신합니다. 수신 큐 관리자는 다음과 같을 수 있습니다.

- 동일한 시스템에서
- 동일한 위치의 다른 시스템에서(또는 심지어 세계 반대편에서)
- 로컬 큐 관리자와 동일한 플랫폼에서 실행
- IBM MQ에서 지원하는 다른 플랫폼에서 실행

이러한 메시지는 다음에서 발생할 수 있습니다.

- 하나의 노드에서 다른 노드로 데이터를 전송하는 사용자 작성 애플리케이션 프로그램
- PCF 명령 또는 MQAI를 사용하는 사용자 작성 관리 애플리케이션
- IBM MQ 탐색기.
- 큐 관리자 보내기:

- 다른 큐 관리자에 대한 도구 이벤트 메시지
- 간접 모드로 `runmqsc` 명령에서 발행된 MQSC 명령(여기서 명령은 다른 큐 관리자에서 실행됨)

메시지를 리모트 큐 관리자로 보낼 수 있으려면, 로컬 큐 관리자에 메시지의 도착을 감지하고 구성하고 있는 메시지를 전송하기 위해 메커니즘이 필요합니다.

- 최소 하나의 채널
- 전송 큐
- 채널 시작기

메시지를 수신하기 위한 리모트 큐 관리자의 경우, 리스너가 필수입니다.

채널은 두 큐 관리자간 단방향 통신 링크이며, 리모트 큐 관리자의 다수의 큐로 향하는 메시지를 전달할 수 있습니다.

각 채널 끝에는 별도 정의가 있습니다. 예를 들어, 한쪽 끝은 송신자나 서버이고, 다른쪽 끝은 수신자나 요청자입니다. 단순한 채널은 리모트 큐 관리자 끝에 수신자 채널 정의 및 로컬 큐 관리자 끝에 송신자 채널 정의로 구성됩니다. 두 정의는 같은 이름을 가지고 함께 하나의 메시지 채널을 구성해야 합니다.

리모트 큐 관리자가 로컬 큐 관리자로 보내진 메시지에 대해 응답하도록 하려면, 로컬 큐 관리자로 응답을 다시 보내도록 두 번째 채널을 설정하십시오.

MQSC 명령 DEFINE CHANNEL을 사용하여 채널을 정의하십시오. 이 절에서 채널에 관한 예가 지정되지 않는 경우 기본 채널 속성을 사용합니다.

메시지의 송신 및 수신을 제어하는 채널의 끝에 메시지 채널 에이전트(MCA)가 있습니다. MCA는 전송 큐에서 메시지를 가져와서 큐 관리자 사이의 통신 링크에 메시지를 배치합니다.

전송 큐는 MCA가 선택하고 리모트 큐 관리자로 이를 전송하기 전에 임시로 메시지를 보유하는 특별한 로컬 큐입니다. 리모트 큐 정의에 전송 큐의 이름을 지정합니다.

MCA가 다중 스레드를 사용하여 메시지를 전송하도록 할 수 있습니다. 이 프로세스는 파이프라이닝이라고 합니다. 파이프라이닝을 사용하면 MCA가 메시지를 좀 더 효율적으로 전송할 수 있으며 채널 성능도 향상됩니다. 파이프라이닝을 사용하기 위해 채널을 구성하는 방법에 대한 세부사항은 채널의 속성을 참조하십시오.

121 페이지의 『원격 관리를 위한 채널 및 전송 큐 준비』 에는 이 정의를 사용하여 원격 관리를 설정하는 방법이 표시되어 있습니다.

분산 큐잉 설정에 대한 자세한 정보는 분산 큐잉 컴포넌트를 참조하십시오.

클러스터를 사용하여 원격 관리

분산 큐잉을 사용한 IBM MQ 네트워크에서 모든 큐 관리자는 독립적입니다. 한 큐 관리자가 다른 큐 관리자에게 메시지를 보내야 하는 경우 전송 큐, 리모트 큐 관리자에 대한 채널 및 메시지를 보내려는 모든 큐에 대한 리모트 큐 정의를 정의해야 합니다.

클러스터는 큐 관리자가 복합적 송신 큐와 채널과 큐 정의 없이 단일 네트워크 위에서 서로 서로 직접적으로 통신할 수 있는 이와 같은 방식으로 설치된 한 그룹의 큐 관리자입니다. 클러스터는 쉽게 설정할 수 있으며, 일반적으로 클러스터에는 논리적으로 관련되어 데이터나 응용프로그램을 공유해야 하는 큐 관리자가 포함되어 있습니다. 심지어 가장 작은 클리스터는 시스템 관리 비용을 줄입니다.

클러스터에서 큐 관리자 네트워크를 설정하면 일반적인 분산 큐잉 환경을 설정하는 것보다 정의가 적습니다. 정의를 적게 작성하면서도 네트워크를 보다 빠르고 쉽게 설정하거나 변경할 수 있으므로, 정의를 작성할 때 오류가 발생할 위험도 줄어듭니다.

클러스터를 설정하려면, 일반적으로 각 큐 관리자마다 하나의 클러스터 송신자(CLUSSDR)와 하나의 클러스터 수신자(CLUSRCVR) 정의가 필요합니다. 트랜스미션 큐 정의나 리모트 큐 정의는 필요 없습니다. 원격 관리 기본 원칙은 클러스터 내에서 사용할 때와 동일하지만, 정의 자체가 일반적으로 상당히 간소합니다.

로컬 큐 관리자에서 원격 관리

이 섹션은 MQSC 및 PCF 명령을 사용하여 로컬 큐 관리자에서 리모트 큐 관리자를 관리하는 방법에 대해 설명합니다.

큐 및 채널 준비는 MQSC 및 PCF 명령 모두에 대해 필수적으로 동일합니다. 이 섹션에서 이해하기 쉽기 때문에 예제는 MQSC 명령을 표시합니다. PCF 명령을 사용하여 관리 프로그램을 작성하는 것에 대한 자세한 정보는 10 페이지의 『PCF(Programmable Command Format) 사용』의 내용을 참조하십시오.

리모트 큐 관리자로 MQSC 명령을 송신(대화식으로 또는 명령이 들어 있는 텍스트 파일에서)합니다. 리모트 큐 관리자는 동일한 시스템에 있을 수도 있지만 일반적으로 다른 시스템에 있는 경우가 많습니다. 유닉스 및 Linux 시스템, Windows 시스템, IBM i및 z/OS를 포함하여 다른 IBM MQ 환경에서 큐 관리자를 원격으로 관리할 수 있습니다.

원격 관리를 구현하려면 특정 오브젝트를 작성해야 합니다. 요구사항을 전문화하지 않는 경우, 기본값(예: 최대 메시지 길이의 경우)이 충분합니다.

원격 관리를 위한 큐 관리자 준비

원격 관리를 위한 큐 관리자를 준비하기 위해 MQSC 명령을 사용하는 방법입니다.

120 페이지의 그림 17에는 **runmqsc** 명령을 사용하여 원격 관리에 필요한 큐 관리자 및 채널의 구성이 표시됩니다. 오브젝트 `source.queue.manager`는 MQSC 명령을 실행할 수 있고 이러한 명령의 결과(운영자 메시지)가 리턴되는 소스 큐 관리자입니다. 오브젝트 `target.queue.manager`는 명령을 처리하고 운영자 메시지를 생성하는 대상 큐 관리자의 이름입니다.

참고: runmqsc를 -w 옵션과 함께 사용하는 경우, `source.queue.manager`는 기본 큐 관리자가 **되어야 합니다** . 큐 관리자 작성에 대한 추가 정보는 crtmqm을 참조하십시오.

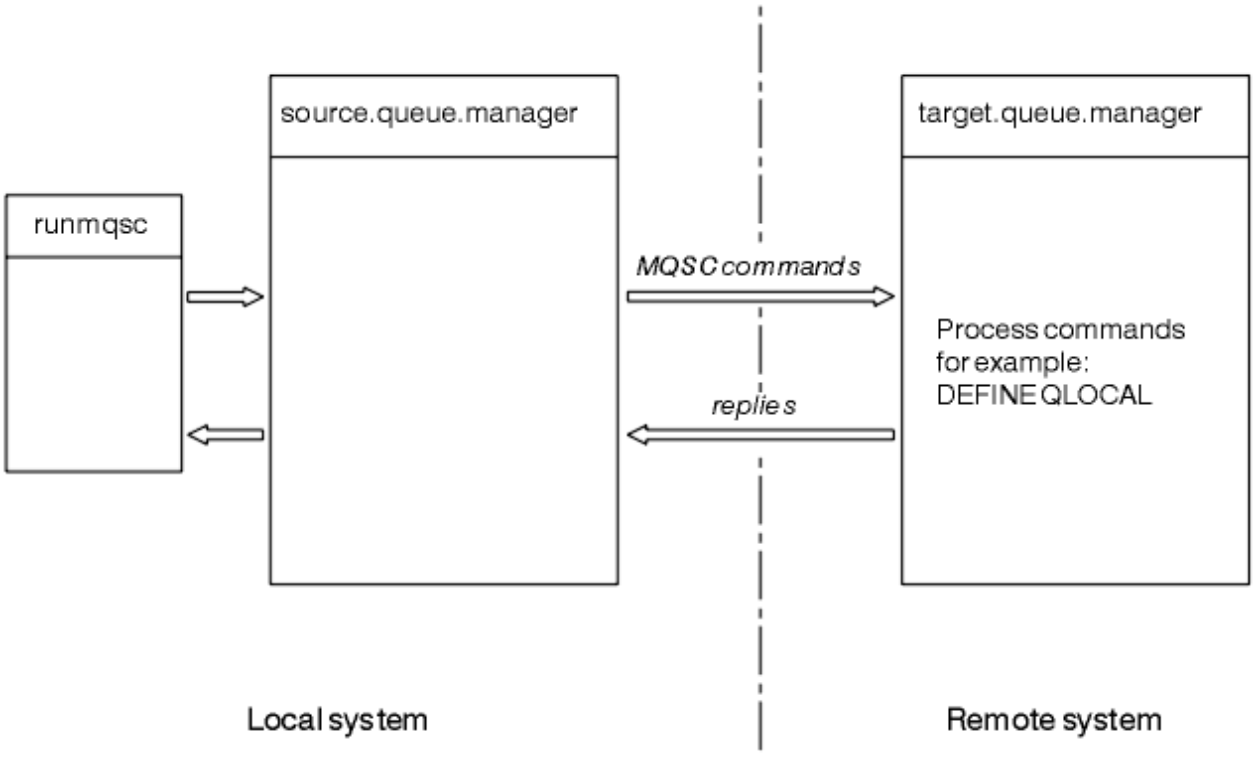


그림 *17. MQSC* 명령을 사용하는 원격 관리

양쪽 시스템 모두에서 다음을 아직 수행하지 않은 경우:

- `crtmqm` 명령을 사용하여 큐 관리자 및 기본 오브젝트를 작성하십시오.
- `strmqm` 명령을 사용하여 큐 관리자를 시작하십시오.

대상 큐 관리자에서:

- 명령 큐(SYSTEM.ADMIN.COMMAND.QUEUE)가 존재해야 합니다. 이 큐는 큐 관리자가 작성될 때 기본적으로 작성됩니다.

이러한 명령을 로컬로 또는 Telnet와 같은 네트워크 기능을 통해 실행해야 합니다.

원격 관리를 위한 채널 및 전송 큐 준비

원격 관리를 위한 채널 및 전송 큐를 준비하기 위해 MQSC 명령을 사용하는 방법입니다.

MQSC 명령을 원격으로 실행하려면, 각 방향당 하나씩 2개의 채널과 연관된 송신 큐를 설정하십시오. 이 예에서는 전송 유형으로 TCP/IP를 사용 중이고 포함된 TCP/IP 주소를 알고 있다고 가정합니다.

채널 `source.to.target`은 소스 큐 관리자에서 대상 큐 관리자로 MQSC 명령을 보내기 위한 것입니다. 해당 송신자는 `source.queue.manager`에 있고 해당 수신자는 `target.queue.manager`에 있습니다. 채널 `target.to.source`는 생성되는 운영자 메시지 및 명령의 출력을 소스 큐 관리자로 리턴하기 위한 것입니다. 각 채널에 대한 전송 큐를 정의해야 합니다. 이 큐는 수신 큐 관리자의 이름이 제공되는 로컬 큐입니다. XMITQ 이름은 큐 관리자 알리어스를 사용 중이지 않는 경우 원격 관리가 작업하기 위해 리모트 큐 관리자 이름과 일치해야 합니다. 121 페이지의 그림 18은 이 구성을 요약합니다.

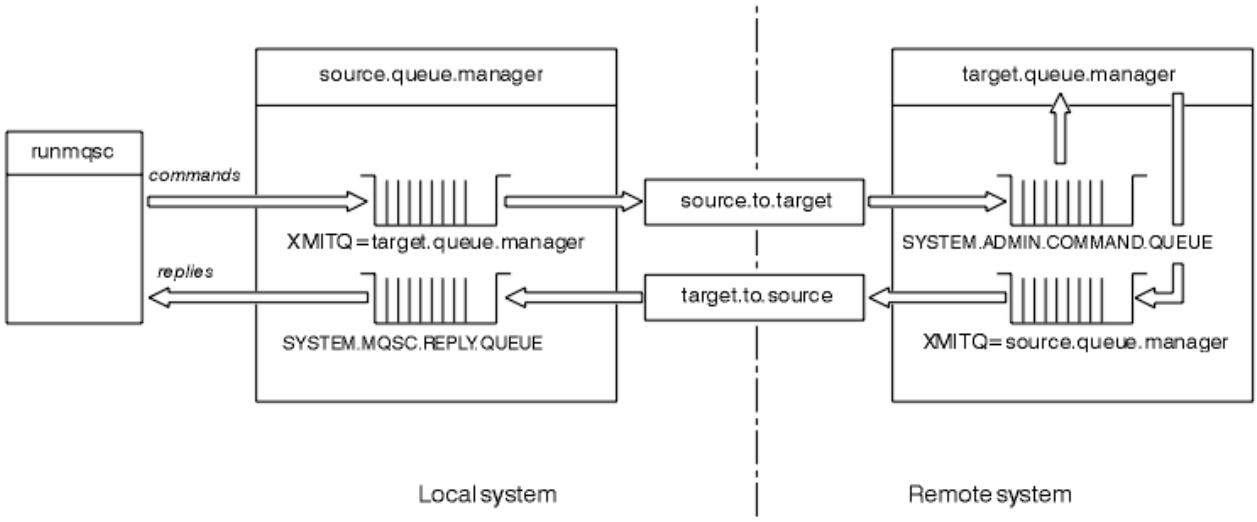


그림 *18.* 원격 관리를 위한 채널 및 큐 설정

채널 설정에 대한 자세한 정보는 분산 큐잉 구성을 참조하십시오.

채널, 리스너 및 전송 큐 정의

소스 큐 관리자(`source.queue.manager`)에서 다음 MQSC 명령을 실행하여 채널, 리스너 및 전송 큐를 정의하십시오.

1. 소스 큐 관리자에서 송신자 채널을 정의하십시오.

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

2. 소스 큐 관리자에서 수신자 채널을 정의하십시오.

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

3. 소스 큐 관리자에서 리스너를 정의하십시오.

```
DEFINE LISTENER ('source.queue.manager') +
TRPTYPE (TCP)
```

4. 소스 큐 관리자에서 전송 큐를 정의하십시오.

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

대상 큐 관리자(target.queue.manager)에서 다음 명령을 실행하여 채널, 리스너 및 전송 큐를 작성하십시오.

1. 대상 큐 관리자에서 송신자 채널을 정의하십시오.

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

2. 대상 큐 관리자에서 수신자 채널을 정의하십시오.

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

3. 대상 큐 관리자에서 리스너를 정의하십시오.

```
DEFINE LISTENER ('target.queue.manager') +
TRPTYPE (TCP)
```

4. 대상 큐 관리자에서 전송 큐를 정의하십시오.

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

참고: 송신자 채널 정의에서 CONNAME 속성에 지정되는 TCP/IP 연결 이름은 설명 전용입니다. 이는 연결의 다른 끝에 있는 시스템의 네트워크 이름입니다. 네트워크에 적절한 값을 사용하십시오.

리스너 및 채널 시작

리스너와 채널을 시작하기 위해 MQSC 명령을 사용하는 방법입니다.

다음 MQSC 명령을 사용하여 리스너 모두를 시작하십시오.

1. 다음 MQSC 명령을 실행하여 소스 큐 관리자(source.queue.manager)에서 리스너를 시작하십시오.

```
START LISTENER ('source.queue.manager')
```

2. 다음 MQSC 명령을 실행하여 대상 큐 관리자(target.queue.manager)에서 리스너를 시작하십시오.

```
START LISTENER ('target.queue.manager')
```

다음 MQSC 명령을 사용하여 송신자 채널 모두를 시작하십시오.

1. 다음 MQSC 명령을 실행하여 소스 큐 관리자(source.queue.manager)에서 송신자 채널을 시작하십시오.

```
START CHANNEL ('source.to.target')
```

2. 다음 MQSC 명령을 실행하여 대상 큐 관리자(target.queue.manager)에서 송신자 채널을 시작하십시오.

```
START CHANNEL ('target.to.source')
```

채널의 자동 정의

MQSC 명령(ALTER QMGR(또는 PCF 명령 변경 큐 관리자))을 사용하여 큐 관리자 오브젝트를 업데이트하는 것으로 수신자 및 서버 연결 정의의 자동 정의를 사용할 수 있습니다.

IBM MQ이 인바운드 첨부 요청을 수신하고 적절한 수신기 또는 서버 연결 채널을 발견할 수 없는 경우, 자동으로 채널을 작성합니다. 자동 정의는 IBM MQ: SYSTEM.AUTO.RECEIVER 및 SYSTEM.AUTO.SVRCONN을 참조하십시오.

채널 정의를 자동으로 작성하는 것에 대한 자세한 정보는 채널 준비를 참조하십시오. 클러스터에 대한 채널을 자동으로 정의하는 것에 대한 정보는 자동 정의된 채널에 대한 작업을 참조하십시오.

원격 관리를 위한 명령 서버 관리

명령 서버의 상태를 시작, 중지 및 표시하는 방법. 명령 서버는 PCF 명령, MQAI를 포함하는 모든 관리 및 원격 관리에도 필수적입니다.

각 큐 관리자에는 이와 연관되는 명령 서버가 있을 수 있습니다. 명령 서버는 리모트 큐 관리자에서 수신되는 명령 또는 애플리케이션에서 PCF 명령을 처리합니다. 이는 처리를 위해 큐 관리자에 명령을 나타내고 원래 명령에 따라 완료 코드 또는 운영자 메시지를 리턴합니다.

참고: 원격 관리의 경우, 대상 큐 관리자가 실행 중인지 확인하십시오. 그렇지 않으면, 명령을 포함하는 메시지는 발행되는 큐 관리자를 남길 수 없습니다. 그 대신, 이러한 메시지는 리모트 큐 관리자를 제공하는 로컬 전송 큐에서 큐잉됩니다. 이 상황을 피하십시오.

명령 서버를 시작하고 중지하기 위한 분리 제어 명령이 있습니다. Providing the command server is running, users of IBM MQ for Windows or IBM MQ for Linux (x86 and x86-64 platforms) can perform the operations described in the following sections using the IBM MQ Explorer. 자세한 정보는 56 페이지의 『MQ Explorer를 사용하여 관리』의 내용을 참조하십시오.

명령 서버 시작

큐 관리자 속성의 값(*SCMDSERV*)에 따라, 명령 서버는 큐 관리자가 시작될 때 자동으로 시작되거나 수동으로 시작되어야 합니다. 큐 관리자 속성의 값은 매개변수 SCMDSERV를 지정하는 MQSC 명령 ALTER QMGR을 사용하여 대체될 수 있습니다. 기본적으로, 명령 서버가 자동으로 시작됩니다.

*SCMDSERV*가 MANUAL로 설정되는 경우, 명령을 사용하여 명령 서버를 시작하십시오.

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

여기서 saturn.queue.manager는 명령 서버가 시작되는 큐 관리자입니다.

명령 서버의 상태 표시

원격 관리의 경우, 대상 큐 관리자의 명령 서버가 실행 중인지 확인하십시오. 실행 중이지 않은 경우, 원격 명령을 처리할 수 없습니다. 명령을 포함하는 메시지도 대상 큐 관리자의 명령 큐에서 큐잉됩니다.

큐 관리자를 위한 명령 서버의 상태를 표시하려면, 다음 MQSC 명령을 실행하십시오.

```
DISPLAY QMSTATUS CMDSERV
```

명령 서버 중지

이전 예가 시작하는 명령 서버를 종료하기 위해 다음 명령을 사용하십시오.

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

두 가지 방법으로 명령 서버를 중지할 수 있습니다.

- 제어된 중지의 경우, -c 플래그로 endmqcsv 명령을 사용하십시오. 이는 기본값입니다.
- 즉각적인 중지의 경우, -i 플래그로 endmqcsv 명령을 사용하십시오.

참고: 큐 관리자를 중지하면 이와 연관되는 명령 서버도 종료합니다.

리모트 큐 관리자에서 MQSC 명령 실행

runmqsc 명령의 특정 양식을 사용하여 리모트 큐 관리자에서 MQSC 명령을 실행할 수 있습니다.

MQSC 명령을 원격으로 처리하려는 경우, 명령 서버는 대상 큐 관리자에서 실행 **되어야** 합니다. (이는 소스 큐 관리자에서 필수는 아닙니다.) 큐 관리자에서 명령 서버를 시작하는 방법에 대한 정보는 123 페이지의 『원격 관리를 위한 명령 서버 관리』의 내용을 참조하십시오.

소스 큐 관리자에서 입력하여 간접 모드에서 대화식으로 MQSC 명령을 실행할 수 있습니다.

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

-w 플래그로 runmqsc 명령의 이 양식은 명령이 명령 서버 입력큐에(변형된 형상에서) 놓여지고 순서대로 실행되는 간접 모드에서 MQSC 명령을 실행시킵니다.

MQSC 명령에 입력할 때, 이 경우, 리모트 큐 관리자(target.queue.manager)로 경로가 재지정됩니다. 제한시간은 30초로 설정됩니다. 응답이 30초 내에 수신되지 않으면, 다음 메시지가 로컬(소스) 큐 관리자에서 생성됩니다.

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

MQSC 명령을 실행하는 것을 중지할 때, 로컬 큐 관리자는 도착한 제한시간 초과 응답을 표시하고 추가 응답을 제거합니다.

소스 큐 관리자는 기본 로컬 큐 관리자에 대한 기본값으로 지정합니다. **runmqsc** 명령에서 -m *LocalQmgrName* 옵션을 지정하는 경우, 로컬 큐 관리자의 방법으로 실행될 수 있도록 명령을 전달할 수 있습니다.

간접 모드에서 리모트 큐 관리자에 MQSC 명령 파일을 실행할 수도 있습니다. 예를 들면, 다음과 같습니다.

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

여기서 mycomds.in은 MQSC 명령을 포함하는 파일이고 report.out은 보고서 파일입니다.

원격으로 명령을 실행하기 위한 제안된 메소드

리모트 큐 관리자에서 명령을 실행할 때, 다음 접근 방식을 사용할 것을 고려하십시오.

1. 명령 파일의 원격 시스템에서 실행하도록 MQSC 명령을 배치하십시오.
2. runmqsc 명령에서 -v 플래그를 지정하여 로컬로 MQSC 명령을 확인하십시오.

   다른 큐 관리자에서 runmqsc를 사용하여 MQSC 명령을 확인할 수 없습니다.
3. 명령 파일이 오류 없이 로컬로 실행되는지 검사하십시오.
4. 원격 시스템에서 명령 파일을 실행하십시오.

MQSC 명령을 원격으로 사용하는 문제가 있는 경우

MQSC 명령을 원격으로 실행하는 것에 어려움이 있는 경우, 다음과 같은지 확인하십시오.

- 대상 큐 관리자에서 명령 서버가 시작됨.
- 올바른 전송 큐가 정의됨.
- 양쪽 모두에 대한 메시지 채널의 두 개의 종류가 정의됨.
  - 명령이 보내지는 채널.
  - 응답이 리턴되는 채널.
- 채널 정의에 올바른 연결 이름(CONNAME)이 지정됨.
- 메시지 채널을 시작된 이후 리스너가 시작됨.
- 예를 들어, 채널이 시작되지만 일정 기간이 지난 후 시스템이 종료되는 경우 연결 끊긴 간격이 만기되지 않는지 검사됩니다. 이는 채널을 수동으로 시작하는 경우 특히 중요합니다.

- 소스 큐 관리자에서 대상 큐 관리자가 인지하지 못하는 요청(예를 들어 리모트 큐 관리자에서 지원되지 않는 매개변수를 포함하는 요청)을 전송했음.

76 페이지의 『MQSC 명령으로 문제점 해결』의 내용을 참조하십시오.

*z/OS*에서 큐 관리자에 대한 작업

이 안내서에 설명된 플랫폼의 큐 관리자에서 z/OS 큐 관리자로 MQSC 명령을 실행할 수 있습니다. 그러나, 이렇게 하려면 `runmqsc` 명령 및 송신자의 채널 정의를 수정해야 합니다.

특히, 대상 큐 관리자가 z/OS:에서 실행되고 있다는 것을 지정하기 위해 소스 노드의 `runmqsc` 명령에 `-x` 플래그를 추가하십시오.

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

리모트 큐의 로컬 정의 작성

리모트 큐의 로컬 정의는 리모트 큐 관리자에서 큐를 참조하는 로컬 큐 관리자의 정의입니다.

로컬 위치에서 리모트 큐를 정의할 필요는 없지만, 로컬 위치에서 리모트 큐를 정의하면 애플리케이션이 리모트 큐가 위치한 큐 관리자의 ID로 규정되는 이름을 지정해야 하는 대신 로컬로 정의된 이름으로 리모트 큐를 참조할 수 있다는 장점이 있습니다.

리모트 큐의 로컬 정의 작업 방법 이해

애플리케이션이 로컬 큐 관리자에 연결되고 MQOPEN 호출을 실행합니다. 열린 호출에서 지정된 큐 이름은 로컬 큐 관리자에서 리모트 큐 정의의 이름입니다. 리모트 큐 정의에서는 대상 큐, 대상 큐 관리자 및 선택적으로 전송 큐의 이름을 제공합니다. To put a message on the remote queue, the application issues an MQPUT call, specifying the handle returned from the MQOPEN call. 큐 관리자는 메시지의 초기에 전송 헤더에서 리모트 큐 이름 및 리모트 큐 관리자 이름을 사용합니다. 이 정보는 네트워크에서 메시지를 올바른 정의로 라우팅하기 위해 사용됩니다.

관리자로서 리모트 큐 정의를 대체하여 메시지의 정의를 제어할 수 있습니다.

다음 예에는 애플리케이션이 리모트 큐 관리자가 소유하는 큐에 메시지를 배치하는 방법이 표시됩니다. 애플리케이션이 큐 관리자에 연결됩니다(예: `saturn.queue.manager`). 대상 큐는 다른 큐 관리자에 의해 소유됩니다.

MQOPEN 호출에서 애플리케이션은 이러한 필드를 지정합니다.

필드 값	설명
ObjectName CYAN.REMOTE.QUEUE	리모트 큐 오브젝트의 로컬 이름을 지정합니다. 이는 대상 큐 및 대상 큐 관리자를 정의합니다.
ObjectType(큐)	이 오브젝트를 큐로 식별합니다.
ObjectQmgrName Blank 또는 `saturn.queue.manager`	이 필드는 선택적입니다. 공백이면, 로컬 큐 관리자 이름은 추측됩니다. (이는 리모트 큐 정의가 있는 큐 관리자입니다.)

이후, 애플리케이션은 이 큐에 메시지를 배치하기 위해 MQPUT 호출을 실행합니다.

로컬 큐 관리자에, 다음 MQSC 명령을 사용하여 리모트 큐의 로컬 정의를 작성할 수 있습니다.

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
DESCR ('Queue for auto insurance requests from the branches') +
RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
RQMNAME (jupiter.queue.manager) +
XMITQ (INQUOTE.XMIT.QUEUE)
```

설명:

QREMOTE (CYAN.REMOTE.QUEUE)
리모트 큐 오브젝트의 로컬 이름을 지정합니다. 이는 이 큐 관리자에 연결된 애플리케이션이 리모트 큐 관리자 jupiter.queue.manager에서 큐 AUTOMOBILE.INSURANCE.QUOTE.QUEUE를 열기 위해 MQOPEN 호출에 지정해야 하는 이름입니다.

DESCR ('Queue for auto insurance requests from the branches')
큐의 사용을 설명하는 추가 텍스트를 제공합니다.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
리모트 큐 관리자에서 대상 큐의 이름을 지정합니다. 이는 큐 이름CYAN.REMOTE.QUEUE를 지정하는 애플리케이션에서 보내지는 메시지에 대한 실제 대상 큐입니다. 큐 AUTOMOBILE.INSURANCE.QUOTE.QUEUE는 리모트 큐 관리자에서 로컬 큐로 정의되어야 합니다.

RQMNAME (jupiter.queue.manager)
대상 큐 AUTOMOBILE.INSURANCE.QUOTE.QUEUE를 소유하는 리모트 큐 관리자의 이름을 지정합니다.

XMITQ (INQUOTE.XMIT.QUEUE)
전송 큐의 이름을 지정합니다. 이는 선택적입니다. 전송 큐의 이름이 지정되지 않으면, 리모트 큐 관리자와 동일한 이름을 가지는 큐가 사용됩니다.

두 가지 경우 모두 적절한 전송 큐가 전송 큐임을 지정하는 *Usage* 속성이 있는 로컬 큐로 정의되어야 합니다 (MQSC 명령에서 USAGE(XMITQ)).

리모트 큐에서 메시지를 배치하는 대체 방법

리모트 큐의 로컬 정의를 사용하는 것이 리모트 큐에 메시지를 배치하는 유일한 방법은 아닙니다. 애플리케이션은 MQOPEN 호출의 일부로서 리모트 큐 관리자 이름을 포함하여 전체 큐 이름을 지정할 수 있습니다. 이 경우, 리모트 큐의 로컬 정의는 필요하지 않습니다. 그러나, 이는 애플리케이션이 런타임 시 리모트 큐 관리자의 이름에 대해 알거나 이에 대한 액세스 권한을 가지고 있어야 한다는 것을 의미합니다.

리모트 큐로 다른 명령 사용

MQSC 명령을 사용하여 리모트 큐 오브젝트의 속성을 표시하거나 대체할 수 있고, 또는 리모트 큐 오브젝트를 삭제할 수 있습니다. 예를 들면, 다음과 같습니다.

- 리모트 큐의 속성을 표시하려면 다음을 수행하십시오.

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- 배치를 사용 가능하게 하기 위해 리모트 큐를 변경하려면 다음을 수행하십시오. 이는 대상 큐에 영향을 주는 것이 아니라, 이 리모트 큐를 지정하는 애플리케이션에만 영향을 줍니다.

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

- 이 리모트 큐를 삭제하려면 다음을 수행하십시오. 이는 대상 큐에 영향을 주지 않고, 해당 로컬 정의에만 영향을 줍니다.

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

참고: 리모트 큐를 삭제할 때, 리모트 큐의 로컬 표현만 삭제합니다. 리모트 큐 자체 또는 메시지를 삭제하지 않습니다.

전송 큐 정의

전송 큐는 큐 관리자가 메시지 채널을 통해 리모트 큐 관리자로 메시지를 전달할 때 사용되는 로컬 큐입니다.

채널은 리모트 큐 관리자에 단방향 링크를 제공합니다. 메시지는 채널이 메시지를 승인할 때까지 전송 큐에 큐잉됩니다. 채널을 정의할 때, 메시지 채널의 송신 측에 전송 큐 이름을 지정해야 합니다.

MQSC 명령 속성 USAGE는 큐가 전송 큐인지 또는 정상 큐인지 여부를 정의합니다.

기본 전송 큐

큐 관리자가 리모트 큐 관리자로 메시지를 보낼 때 다음 순서를 사용하여 전송 큐를 식별합니다.

1. 리모트 큐의 로컬 정의의 XMITQ 속성에 이름 지정된 전송 큐.
2. 대상 큐 관리자와 동일한 이름을 가지는 전송 큐. (이 값은 리모트 큐의 로컬 정의의 XMITQ에 있는 기본값임.)
3. 로컬 큐 관리자의 DEFXMITQ 속성에 이름 지정된 전송 큐.

예를 들어, 다음 MQSC 명령은 `target.queue.manager`로 이동하는 메시지에 대해 `source.queue.manager`에서 기본 전송 큐를 작성합니다.

```
DEFINE QLOCAL ('target.queue.manager') +
DESCR ('Default transmission queue for target qm') +
USAGE (XMITQ)
```

애플리케이션은 전송 큐에 직접 또는 리모트 큐 정의를 통해 간접적으로 메시지를 넣습니다. 125 페이지의 『리모트 큐의 로컬 정의 작성』의 내용도 참조하십시오.

분산 네트워크에 대한 비동기 명령이 완료되었는지 확인

분배된 네트워크에서 사용될 때 많은 명령은 비동기입니다. 명령 및 명령이 실행될 때 네트워크 상태에 따라 완료하기 위해 상당한 시간이 필요할 수 있습니다. 큐 관리자가 완료 시 메시지를 발행하지 않으므로 명령이 완료되는지 확인하는 다른 방법이 필요합니다.

이 태스크 정보

클러스터에 작성하는 거의 모든 구성 변경은 비동기적으로 완료될 수 있습니다. 이는 클러스터 내에서 작동하는 순환의 업데이트 및 내부 관리 때문입니다. 발행/구독 계층의 경우, 구독에 영향을 미치는 구성 변경도 비동기적으로 완료할 수 있습니다. 이는 명령의 이름으로부터 항상 명백하지 않습니다.

다음 MQSC 명령은 모두 비동기적으로 완료할 것입니다. 각각의 이러한 명령은 동등한 PCF를 가지고 있고 대부분이 MQ Explorer로부터 사용 가능합니다. 워크로드가 없는 작은 네트워크에서 실행될 때, 이러한 명령은 일반적으로 몇 초 내에 완료됩니다. 그러나, 이는 더 크고 더 많이 사용되는 네트워크에 대한 경우는 아닙니다. 특히 여러 큐 관리자에서 동시에 실행될 때 **REFRESH CLUSTER** 명령은 더 오래 걸릴 수도 있습니다.

이러한 명령이 완료되었음을 신뢰하려면 리모트 큐 관리자에 예상된 오브젝트가 존재하는지 확인하십시오.

프로시저

- ALTER QMGR

  ALTER QMGR PARENT 명령의 경우, `DISPLAY PUBSUB TYPE(PARENT) ALL`을 사용하여 요청된 상위 관계의 상태를 추적하십시오.

  ALTER QMGR REPOS 및 ALTER QMGR REPOSNL 명령의 경우, `DISPLAY CLUSQMGR QMTYPE`을 사용하여 완료를 확인하십시오.

- DEFINE CHANNEL, ALTER CHANNEL 및 DELETE CHANNEL

  테이블 ALTER CHANNEL 매개변수에 나열된 모든 매개변수의 경우, `DISPLAY CLUSQMGR` 명령을 사용하여 변경사항이 클러스터에 전파될 때 모니터하십시오.

- DEFINE NAMELIST, ALTER NAMELIST 및 DELETE NAMELIST.

  QMgr 오브젝트의 **CLUSNL** 속성에서 **NAMELIST**를 사용하는 경우, 큐 또는 클러스터 채널이 해당 오브젝트에 영향을 줄 수 있습니다. 영향을 받는 오브젝트에 대해 적당한 것으로 모니터하십시오.

  `SYSTEM.QPUBSUB.QUEUE.NAMELIST`에 대한 변경사항은 발행/구독 계층에서 프록시 구독의 작성 또는 취소에 영향을 줄 수 있습니다. `DISPLAY SUB SUBTYPE(PROXY)` 명령을 사용하여 이를 모니터하십시오.

- DEFINE queues, ALTER queues 및 DELETE queues.

  테이블 DISPLAY QUEUE 명령에 의해 리턴될 수 있는 매개변수에 나열되는 모든 매개변수의 경우, `DISPLAY QCLUSTER` 명령을 사용하여 변경사항이 클러스터로 전파될 때 모니터하십시오.

- DEFINE SUB 및 DELETE SUB

  토픽 문자열에 대하여 첫 번째 구독을 정의할 때, 발행/구독 계층 또는 발행/구독 클러스터에서 프록시 구독을 작성할 수도 있습니다. 마찬가지로, 토픽 문자열에 마지막 구독을 삭제할 때, 발행/구독 계층 또는 발행/구독 클러스터에서 프록시 구독을 취소시킬 것입니다.

  구독을 정의하거나 삭제하는 명령이 끝났는지 확인하려면, 예상된 프록시 구독이 분배된 네트워크의 기타 큐 관리자에 있는지 여부를 확인하십시오. 클러스터에서 직접 라우팅을 사용 중인 경우, 예상된 프록시 구독이 클러스터에서 다른 부분 저장소에 있는지 확인하십시오. 클러스터에서 토픽 호스트 라우팅을 사용 중인 경우, 예상된 프록시 구독이 일치하는 토픽 호스트에 존재하는지 확인하십시오. 다음 MQSC 명령을 사용하십시오.

  ```
  DISPLAY SUB(*) SUBTYPE(PROXY)
  ```

  클러스터 또는 계층에서 실행될 때 다음 동일한 구독 및 비구독 MQI 호출에 동일한 검사를 사용하십시오.

  - MQSUB를 사용하여 구독하십시오.
  - MQCO_REMOVE_SUB와 함께 MQCLOSE를 사용하여 구독 해제하십시오.
- DEFINE TOPIC, ALTER TOPIC 및DELETE TOPIC

  클러스터된 토픽을 정의, 변경 또는 삭제하는 명령이 완료되는지 확인하려면 클러스터(직접 라우팅을 사용 중인 경우) 또는 다른 토픽 호스트(토픽 호스트 라우팅)의 다른 부분 저장소에서 토픽을 표시하십시오.

  테이블 DISPLAY TOPIC 명령에 의해 리턴될 수 있는 매개변수에 나열된 모든 매개변수의 경우, DISPLAY TCLUSTER 명령을 사용하여 변경사항이 클러스터에 전파될 때 모니터하십시오.

  참고:

  - **CLUSTER** 매개변수는 발행/구독 클러스터에서 프록시 구독의 작성 또는 취소에 영향을 줄 수 있습니다.
  - **PROXYSUB** 및 **SUBSCOPE** 매개변수는 발행/구독 계층 또는 발행/구독 클러스터에서 프록시 구독의 작성 또는 취소에 영향을 줄 수 있습니다.
  - DISPLAY SUB SUBTYPE(PROXYSUB) 명령을 사용하여 이를 모니터하십시오.
- REFRESH CLUSTER

  REFRESH CLUSTER 명령을 실행 중인 경우, 클러스터 명령 큐 용량을 폴링하십시오. 0에 도달하는 것을 기다리고, 오브젝트를 찾기 전에 0에 남아 있으십시오.

  1. 다음 MQSC 명령을 사용하여 클러스터 명령 큐 용량이 0이라는 것을 확인하십시오.

     ```
     DISPLAY QL(SYSTEM.CLUSTER.COMMAND.QUEUE) CURDEPTH
     ```

  2. 큐 용량이 0에 도달할 때까지 검사를 반복하고 후속 검사에서 0이 됩니다.

  REFRESH CLUSTER 명령은 오브젝트를 제거하고 재작성하고 큰 구성에서는 완료에 많은 시간이 걸릴 수 있습니다. 발행/구독 클러스터에 대한 REFRESH CLUSTER 고려사항을 참조하십시오.
- REFRESH QMGR TYPE(PROXYSUB)

  REFRESH QMGR TYPE(PROXYSUB) 명령이 완료되는지 확인하려면 프록시 구독이 분배된 네트워크의 기타 큐 관리자에서 수정되는지 확인하십시오. 클러스터에서 직접 라우팅을 사용 중인 경우, 프록시 구독이 클러스터의 다른 부분 저장소에서 수정되는지 확인하십시오. 토픽 호스트 라우팅을 클러스터에서 사용 중인 경우, 예상된 프록시 구독이 일치하는 토픽 호스트에서 수정되는지 확인하십시오. 다음 MQSC 명령을 사용하십시오.

  ```
  DISPLAY SUB(*) SUBTYPE(PROXYSUB)
  ```
- 클러스터 재설정

  RESET CLUSTER 명령이 완료되는지 확인하려면 DISPLAY CLUSQMGR을 사용하십시오.
- RESET QMGR TYPE(PUBSUB)

RESET QMGR 명령이 완료되는지 확인하려면 DISPLAY PUBSUB TYPE(PARENT|CHILD)을 사용하십시오.

참고: RESET QMGR 명령은 발행/구독 계층 또는 발행/구독 클러스터에서 프록시 구독의 취소를 야기할 수 있습니다. DISPLAY SUB SUBTYPE(PROXYSUB) 명령을 사용하여 이를 모니터하십시오.

- 명령이 완료될 때 0의 큐 용량으로 향하는 다른 시스템 큐를 모니터할 수도 있습니다.

  예를 들어, SYSTEM.INTER.QMGR.CONTROL 큐 및 SYSTEM.INTER.QMGR.FANREQ 큐를 모니터하려고 할 수 있습니다. 클러스터에서 프록시 구독 트래픽 모니터 및 발행/구독 네트워크에서 작성자 및 이용자 밸런싱을 참조하십시오.

다음에 수행할 작업

이는 비동기 명령이 완료된다는 것을 확인하지 못하는 경우, 오류가 발생할 수도 있습니다. 조사하기 위해 먼저 명령이 실행된 큐 관리자에 대한 로그를 확인하고, 그런 다음(클러스터의 경우) 클러스터 전체 저장소 로그를 검사하십시오.

관련 정보

z/OS z/OS에서 CLUSTER 명령의 비동기 작동

알리어스로서 리모트 큐 정의 사용

다른 큐 관리자에서 큐를 찾을 뿐만 아니라, 큐 관리자 알리어스 및 응답 대상 큐에 대한 리모트 큐의 로컬 정의를 사용할 수도 있습니다. 두 유형의 알리어스 모두 리모트 큐의 로컬 정의를 통해 해결됩니다. 메시지가 목적지에 도착하기에 적절한 채널을 설정해야 합니다.

큐 관리자 알리어스

알리어스는 메시지에 지정된 것처럼 대상 큐 관리자의 이름이 메시지 라우트의 큐 관리자에 의해 수정된다는 프로세스입니다.큐 관리자의 네트워크 내에서 메시지의 목적지를 제어하기 위해 사용할 수 있으므로 큐 관리자 알리어스는 중요합니다.

제어 시점에서 큐 관리자의 리모트 큐 정의를 대체하여 이를 수행하십시오. 송신 애플리케이션은 지정된 큐 관리자 이름이 알리어스라는 것을 알지 못합니다.

큐 관리자 알리어스에 대한 자세한 정보는 알리어스의 개념을 참조하십시오.

응답 대상 큐 알리어스

선택적으로, 애플리케이션은 큐에 요청 메시지를 배치할 때 응답 대상 큐의 이름을 지정할 수 있습니다.

메시지를 처리하는 애플리케이션이 응답 대상 큐의 이름을 추출하는 경우, 필요하면 응답 메시지를 송신할 위치를 압니다.

응답 대상 큐 알리어스는 요청 메시지에 지정된 것처럼 응답 대상 큐가 메시지 라우트에서 큐 관리자에 의해 대체되는 프로세스입니다. 송신 애플리케이션은 지정된 응답 대상 큐 이름이 알리어스임을 알지 못합니다.

응답 대상 큐 알리어스를 사용하면 응답 대상 큐의 이름과 선택적으로 큐 관리자를 대체할 수 있습니다. 이것은 응답 메시지에 사용되는 라우트를 제어할 수 있습니다.

요청 메시지, 응답 메시지 및 응답 대상 큐에 대한 자세한 정보는 메시지의 유형 및 응답 대상 큐 및 큐 관리자를 참조하십시오.

응답 대상 큐 알리어스에 대한 자세한 정보는 응답 대상 큐 알리어스 및 클러스터를 참조하십시오.

데이터 변환

IBM MQ 정의된 형식 (내장 형식 이라고도 함) 의 메시지 데이터 두 문자 세트 모두 단일 언어 또는 유사한 언어 그룹과 관련되어 있는 경우, 하나의 코드화된 문자 세트에서 다른 코드화 문자 세트로 큐 관리자에 의해 변환될 수 있습니다.

예를 들면, 코드화 문자 세트 ID(CCSID)가 850과 500인 코드화 문자 세트 간 변환이 지원되는데, 두 문자 세트 모두 서유럽 언어에 해당하기 때문입니다.

ASCII에 대한 EBCDIC 줄 바꾸기(NL) 문자 변환은 모든 큐 관리자를 참조하십시오.

지원되는 변환은 데이터 변환에 정의되어 있습니다.

큐 관리자가 내장 형식의 메시지를 변환할 수 없는 경우

CCSID가 다른 자국어(NL) 그룹을 나타내는 경우 큐 관리자는 내장 형식에서 자동으로 메시지를 변환할 수 없습니다. 예를 들어, CCSID 850과 CCSID 1025 사이의 변환(키릴 문자 스크립트를 사용하는 언어에 설정된 EBCDIC 코드화 문자임)은 지원되지 않습니다. 하나의 코드화 문자 세트의 많은 문자가 다른 코드화 문자 세트로 표시될 수 없기 때문입니다. 다른 자국어(NL)에서 작동하는 큐 관리자의 네트워크 및 일부 코드화 문자 세트 사이의 데이터 변환이 지원되지 않는 경우, 기본 변환을 사용할 수 있습니다. 기본 데이터 변환은 130 페이지의 『기본 데이터 변환』 에 설명되어 있습니다.

파일 ccsid.tbl

파일 ccsid.tbl은 다음 목적으로 사용됩니다.

- IBM MQ for Windows에서 이는 지원되는 모든 코드 세트를 기록합니다.
- AIX 및 HP-UX 플랫폼에서 지원되는 코드 세트가 운영 체제에서 내부적으로 보유됩니다.
- 기타 모든 유닉스 및 Linux 플랫폼의 경우, 지원된 코드 세트가 IBM MQ에서 제공하는 변환 테이블에 있습니다.
- 이는 추가 코드 세트를 지정합니다. 추가 코드 세트를 지정하려면 ccsid.tbl을 편집해야 합니다(이를 수행하는 방법에 대한 안내가 파일에 제공됨).
- 이는 기본 데이터 변환을 지정합니다.

ccsid.tbl에 기록되는 정보를 업데이트할 수 있습니다. 예를 들어, 운영 체제의 향후 릴리스가 추가 코드화 문자 세트를 지원하는 경우 이를 수행할 수도 있습니다.

IBM MQ for Windows에서, ccsid.tbl은 기본적으로 디렉토리 `C: \Program Files\IBM\WebSphere MQ\conv\table`에 있습니다.

IBM MQ for UNIX 및 Linux 시스템에서 ccsid.tbl은 `/var/mqm/conv/table` 디렉토리에 있습니다.

기본 데이터 변환

데이터 변환이 일반적으로 지원되지 않는 두 개의 시스템 사이의 채널을 설정하는 경우, 작업할 채널에 대한 기본 데이터 변환을 사용해야 합니다.

기본 데이터 변환을 사용하려면 기본 EBCDIC CCSID 및 기본 ASCII CCSID를 지정하기 위해 ccsid.tbl 파일을 편집하십시오. 수행 방법에 대한 지시사항은 파일에 포함되어 있습니다. 채널을 사용하여 연결될 모든 시스템에서 이를 수행해야 합니다. 변경사항을 적용하려면 큐 관리자를 재시작하십시오.

기본 데이터 변환 프로세스는 다음과 같습니다.

- 소스 및 대상 CCSID 사이의 변환이 지원되지 않지만, 소스 및 대상 환경의 CCSID가 양쪽 EBCDIC 또는 양쪽 ASCII인 경우, 문자 데이터는 변환되지 않고 대상 애플리케이션으로 전달됩니다.
- 하나의 CCSID가 ASCII 코드화 문자 세트를 나타내고 다른 CCSID가 EBCDIC 코드화 문자 세트를 나타내는 경우, IBM MQ는 ccsid.tbl에 정의된 기본 데이터 변환 CCSDI를 사용하여 데이터를 변환합니다.

참고: 메시지에 지정된 코드화 문자 세트 및 기본 코드화 문자 세트에 동일한 코드 값이 있는 문자로 변환되는 문자를 제한하십시오. IBM MQ 오브젝트 이름에 유효한 문자 세트만 사용하는 경우(IBM MQ 오브젝트 이름 지정에 정의됨) 일반적으로 이 요구사항을 충족하게 됩니다. 일본에서 사용되는 EBCDIC CCSID 290, 930, 1279 및 5026의 경우 예외가 발생하는데, 여기서는 소문자가 다른 EBCDIC CCSID에 사용된 코드와 다른 코드를 갖기 때문입니다.

사용자 정의 형식의 메시지 변환

큐 관리자는 하나의 코드화 문자 세트에서 다른 문자 세트로 사용자 정의 형식의 메시지를 변환할 수 없습니다. 사용자 정의 형식의 데이터를 변환해야 하는 경우, 그와 같은 각각의 형식에 데이터 변환 엑시트를 제공해야 합니다. 사용자 정의 형식의 문자 세트를 변환하기 위해 기본 CCSID를 사용하지 마십시오. 사용자 정의 형식의 데이

터를 변환하고 데이터 변환 엑시트를 작성하는 것에 대한 자세한 정보는 데이터 변환 엑시트 작성을 참조하십시오.

큐 관리자 CCSID 변경

큐 관리자의 CCSID를 변경하기 위해 ALTER OMGR 명령의 CCSID 속성을 사용할 때, 명령 서버와 채널 프로그램을 포함하여 실행 중인 모든 애플리케이션이 중지되고 재시작되도록 큐 관리자를 중지하고 재시작하십시오.

이는 큐 관리자 CCSID가 변경될 때 실행 중인 모든 애플리케이션은 기존 CCSID를 계속 사용하므로 필수적입니다.

IBM MQ Telemetry 관리

IBM MQ Telemetry는 MQ Explorer를 사용하거나 명령행에서 관리됩니다. 탐색기를 사용하여 텔레메트리 채널을 구성하고 텔레메트리 서비스를 제어하며 IBM MQ에 연결되는 MQTT 클라이언트를 모니터하십시오. JAAS, SSL및 IBM MQ 오브젝트 권한 관리자를 사용하여 IBM MQ Telemetry 의 보안을 구성하십시오.

MQ Explorer를 사용하여 관리

탐색기를 사용하여 텔레메트리 채널을 구성하고 텔레메트리 서비스를 제어하며 IBM MQ에 연결되는 MQTT 클라이언트를 모니터하십시오. JAAS, SSL및 IBM MQ 오브젝트 권한 관리자를 사용하여 IBM MQ Telemetry 의 보안을 구성하십시오.

명령행을 사용하여 관리

IBM MQ Telemetry는 IBM MQ MQSC 명령을 사용하여 명령행에서 완전하게 관리될 수 있습니다.

IBM MQ Telemetry 문서에는 IBM MQ Telemetry Transport v3 클라이언트 애플리케이션의 기본 사용법을 증명하는 샘플 스크립트가 있습니다.

사용하기 전에 IBM MQ Telemetry 에 대한 애플리케이션 개발 섹션의 IBM MQ Telemetry Transport 샘플 프로그램 에서 샘플을 읽고 이해하십시오.

관련 정보

IBM MQ Telemetry

MQXR 특성

Linux 및 AIX에서 텔레메트리에 대한 큐 관리자 구성

IBM MQ Telemetry를 실행하도록 큐 관리자를 구성하려면 다음 수동 단계를 수행하십시오. MQ Explorer에 대한 IBM MQ Telemetry 지원을 사용하여 보다 간단한 구성을 설정할 수 있는 자동화된 프로시저를 실행할 수 있습니다.

시작하기 전에

1. IBM MQ 및 IBM MQ Telemetry 기능을 설치하는 방법에 대한 정보는 IBM MQ Telemetry설치 를 참조하십시오.
2. 큐 관리자를 작성하고 시작하십시오. 이 태스크에서는 큐 관리자를 *qMgr*이라고 합니다.
3. 이 태스크의 일부로 텔레메트리(MQXR) 서비스를 구성합니다. MQXR 특성 설정은 플랫폼 특정 특성 파일(`mqxr_unix.properties`)에 저장됩니다. 거의 모든 설정이 MQSC 관리 명령 또는 MQ Explorer를 통해 구성할 수 있으므로 MQXR 특성 파일을 직접 편집할 필요는 없습니다. 파일을 직접적으로 편집하기로 결심한 경우, 변경하기 전에 큐 관리자를 중지하십시오. MQXR 특성을 참조하십시오.

이 태스크 정보

MQ Explorer에 대한 IBM MQ Telemetry 지원에는 마법사 및 샘플 명령 프로시저 `sampleMQM`이 포함됩니다. 게스트 사용자 ID를 사용하여 초기 구성을 설정합니다. MQ Explorer 을 사용하여 IBM MQ Telemetry 설치 확인 및 IBM MQ Telemetry Transport 샘플 프로그램을 참조하십시오.

다른 권한 부여 설계를 사용하여 IBM MQ Telemetry를 수동으로 구성하려면 이 태스크의 단계를 수행하십시오.

프로시저

1. 텔레메트리 샘플 디렉토리에서 명령 창을 여십시오.

   텔레메트리 샘플 디렉토리는 /opt/mqm/mqxr/samples입니다.
2. 텔레메트리 전송 큐를 작성하십시오.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc
   qMgr
   ```

   텔레메트리(MQXR) 서비스를 처음 시작하면 SYSTEM.MQTT.TRANSMIT.QUEUE가 작성됩니다.

   이 태스크에서 SYSTEM.MQTT.TRANSMIT.QUEUE는 수동으로 작성합니다. SYSTEM.MQTT.TRANSMIT.QUEUE에 대한 액세스 권한을 부여하려면 텔레메트리(MQXR) 서비스가 시작되기 전에 해당 큐가 존재해야 하기 때문입니다.
3. 기본 전송 큐 설정하기

   텔레메트리(MQXR) 서비스를 처음 시작하면 큐 관리자를 대체하여 SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 만들지 않습니다.

   SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 만들려면 기본 전송 큐 특성을 대체하십시오. MQ Explorer 또는 다음 예제의 명령을 사용하여 특성을 변경하십시오.

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   기본 전송 큐를 대체하면 기존 구성에 지장을 줄 수 있습니다. 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 대체하는 이유는 MQTT 클라이언트로 직접 메시지를 보내는 작업을 더욱 쉽게 만들기 위해서입니다. 기본 전송 큐를 대체하지 않고 MQ Explorer 메시지를 수신하는 모든 클라이언트에 대한 리모트 큐 정의를 추가해야 합니다. 136 페이지의 『클라이언트에 직접 메시지 송신』의 내용을 참조하십시오.
4. 하나 이상의 사용자 ID를 작성하려면 138 페이지의 『IBM MQ 오브젝트에 액세스하기 위해 MQTT 클라이언트 권한 부여』에서 프로시저를 수행하십시오. 사용자 ID에 발행물을 MQTT 클라이언트에 발행, 구독 및 전송하기 위한 권한이 있습니다.
5. 텔레메트리(MQXR) 서비스를 설치하십시오.

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   133 페이지의 그림 19의 예제 코드도 참조하십시오.
6. 서비스를 시작하십시오.

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   텔레메트리(MQXR) 서비스는 큐 관리자가 시작될 때 자동으로 시작됩니다.

   큐 관리자가 이미 실행 중이므로 이 태스크에서 수동으로 시작됩니다.
7. MQ Explorer를 사용하여 MQTT 클라이언트에서 연결을 승인하기 위해 텔레메트리 채널을 구성하십시오.

   텔레메트리 채널 ID가 4단계에 정의된 사용자 ID 중 하나가 되도록 텔레메트리 채널을 구성해야 합니다.

   DEFINE CHANNEL(MQTT)도 참조하십시오.
8. 샘플 클라이언트를 실행하여 구성을 확인하십시오.

   샘플 클라이언트가 텔레메트리 채널에서 작동하게 하려면 해당 채널이 클라이언트에게 발행, 구독 및 발행을 수신할 권한을 부여해야 합니다. 샘플 클라이언트는 기본적으로 포트 1883에서 텔레메트리 채널에 연결합니다. IBM MQ Telemetry Transport 샘플 프로그램도 참조하십시오.

예

133 페이지의 그림 19 는 Linux에서 SYSTEM.MQXR.SERVICE 을 수동으로 작성하기 위한 **runmqsc** 명령을 표시합니다.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+"') +
STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

그림 *19. installMQXRService_unix.mqsc*

Windows에 텔레메트리용 큐 관리자 구성

IBM MQ Telemetry를 실행하도록 큐 관리자를 구성하려면 다음 수동 단계를 수행하십시오. MQ Explorer에 대한 IBM MQ Telemetry 지원을 사용하여 보다 간단한 구성을 설정할 수 있는 자동화된 프로시저를 실행할 수 있습니다.

시작하기 전에

1. IBM MQ설치 방법 및 IBM MQ Telemetry 기능에 대한 정보는 IBM MQ Telemetry설치 를 참조하십시오.
2. 큐 관리자를 작성하고 시작하십시오. 이 태스크에서는 큐 관리자를 *qMgr*이라고 합니다.
3. 이 태스크의 일부로 텔레메트리(MQXR) 서비스를 구성합니다. MQXR 특성 설정은 플랫폼 특정 특성 파일(mqxr_win.properties)에 저장됩니다. 거의 모든 설정이 MQSC 관리 명령 또는 MQ Explorer를 통해 구성될 수 있으므로 일반적으로 MQXR 특성 파일을 직접적으로 편집할 필요가 없습니다. 파일을 직접적으로 편집하기로 결심한 경우, 변경하기 전에 큐 관리자를 중지하십시오. MQXR 특성을 참조하십시오.

이 태스크 정보

MQ Explorer에 대한 IBM MQ Telemetry 지원에는 마법사 및 샘플 명령 프로시저 sampleMQM이 포함됩니다. 게스트 사용자 ID를 사용하여 초기 구성을 설정합니다. MQ Explorer 을 사용하여 IBM MQ Telemetry 설치 확인 및 IBM MQ Telemetry Transport 샘플 프로그램을 참조하십시오.

다른 권한 부여 설계를 사용하여 IBM MQ Telemetry를 수동으로 구성하려면 이 태스크의 단계를 수행하십시오.

프로시저

1. 텔레메트리 샘플 디렉토리에서 명령 창을 여십시오.

   텔레메트리 샘플 디렉토리는 *WMQ program installation directory*\mqxr\samples입니다.
2. 텔레메트리 전송 큐를 작성하십시오.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   텔레메트리(MQXR) 서비스를 처음 시작하면 SYSTEM.MQTT.TRANSMIT.QUEUE가 작성됩니다.

   이 태스크에서 SYSTEM.MQTT.TRANSMIT.QUEUE는 수동으로 작성합니다.
   SYSTEM.MQTT.TRANSMIT.QUEUE에 대한 액세스 권한을 부여하려면 텔레메트리(MQXR) 서비스가 시작되기 전에 해당 큐가 존재해야 하기 때문입니다.
3. *qMgr*에 대한 기본 전송 큐를 설정하십시오.

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   그림 *20.* 기본 전송 큐 설정

텔레메트리(MQXR) 서비스를 처음 시작하면 큐 관리자를 대체하여 SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 만들지 않습니다.

SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 만들려면 기본 전송 큐 특성을 대체하십시오. 133 페이지의 그림 20에서 MQ Explorer 또는 명령을 통해 특성을 변경하십시오.

기본 전송 큐를 대체하면 기존 구성에 지장을 줄 수 있습니다. 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 대체하는 이유는 MQTT 클라이언트로 직접 메시지를 보내는 작업을 더욱 쉽게 만들기 위해서입니다. 기본 전송 큐를 대체하지 않고 IBM MQ 메시지를 수신하는 모든 클라이언트에 대한 리모트 큐 정의를 추가해야 합니다. 136 페이지의 『클라이언트에 직접 메시지 송신』의 내용을 참조하십시오.

4. 하나 이상의 사용자 ID를 작성하려면 138 페이지의 『IBM MQ 오브젝트에 액세스하기 위해 MQTT 클라이언트 권한 부여』에서 프로시저를 수행하십시오. 사용자 ID에 발행물을 MQTT 클라이언트에 발행, 구독 및 전송하기 위한 권한이 있습니다.
5. 텔레메트리(MQXR) 서비스를 설치하십시오.

```
type
installMQXRService_win.mqsc | runmqsc qMgr
```

6. 서비스를 시작하십시오.

```
echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
```

텔레메트리(MQXR) 서비스는 큐 관리자가 시작될 때 자동으로 시작됩니다.

큐 관리자가 이미 실행 중이므로 이 태스크에서 수동으로 시작됩니다.

7. MQ Explorer를 사용하여 MQTT 클라이언트에서 연결을 승인하기 위해 텔레메트리 채널을 구성하십시오.

텔레메트리 채널 ID가 4단계에 정의된 사용자 ID 중 하나가 되도록 텔레메트리 채널을 구성해야 합니다.

DEFINE CHANNEL(MQTT)도 참조하십시오.

8. 샘플 클라이언트를 실행하여 구성을 확인하십시오.

샘플 클라이언트가 텔레메트리 채널에서 작동하게 하려면 해당 채널이 클라이언트에게 발행, 구독 및 발행을 수신할 권한을 부여해야 합니다. 샘플 클라이언트는 기본적으로 포트 1883에서 텔레메트리 채널에 연결합니다. IBM MQ Telemetry Transport 샘플 프로그램도 참조하십시오.

수동으로 SYSTEM.MQXR.SERVICE 작성

134 페이지의 그림 21 는 Windows에서 SYSTEM.MQXR.SERVICE 을 수동으로 작성하기 위한 **runmqsc** 명령을 표시합니다.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\."') +
STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

그림 *21. installMQXRService_win.mqsc*

MQTT 클라이언트에 메시지를 보내기 위한 분산 큐잉 구성

IBM MQ 애플리케이션은 클라이언트가 작성한 구독에 발행하거나 메시지를 직접적으로 보냄으로써 MQTT v3 클라이언트 메시지를 보낼 수 있습니다. 둘 중 어떤 방식을 사용하든 메시지는 SYSTEM.MQTT.TRANSMIT.QUEUE에 배치되고 텔레메트리(MQXR) 서비스를 통해 클라이언트에 보내집니다. SYSTEM.MQTT.TRANSMIT.QUEUE에 메시지를 배치하는 방법에는 여러 가지가 있습니다.

MQTT 클라이언트 구독에 대한 메시지 발행

텔레메트리(MQXR) 서비스는 MQTT 클라이언트 대신 구독을 작성합니다. 클라이언트는 클라이언트가 보낸 구독과 일치하는 발행물에 대한 목적지입니다. 텔레메트리 서비스는 일치하는 발행을 클라이언트로 다시 전달합니다.

MQTT 클라이언트는 해당 `ClientIdentifier`로 설정된 큐 관리자 이름을 가지는 큐 관리자로서 IBM MQ에 연결됩니다. 클라이언트로 보낼 발행의 목적지는 `SYSTEM.MQTT.TRANSMIT.QUEUE` 전송 큐입니다. 텔레메트리 서비스는 대상 큐 관리자 이름을 특정 클라이언트의 키로 사용하여 `SYSTEM.MQTT.TRANSMIT.QUEUE` 의 메시지를 MQTT 클라이언트로 전달합니다.

텔레메트리(MQXR) 서비스는 `ClientIdentifier`를 큐 관리자 이름으로서 사용하여 전송 큐를 엽니다. 텔레메트리(MQXR) 서비스는 클라이언트 구독과 일치하는 발행물을 전달하기 위해 큐의 오브젝트 핸들을 `MQSUB` 호출로 전달합니다. 오브젝트 이름 확인에서는 `ClientIdentifier`가 리모트 큐 관리자 이름으로 작성되므로 전송 큐를 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 해석해야 합니다. 표준 IBM MQ 오브젝트 이름 분석을 사용하여 *`ClientIdentifier`* 는 다음과 같이 분석됩니다. 135 페이지의 표 6을 참조하십시오.

1. *`ClientIdentifier`*가 어떤 것과도 일치하지 않습니다.

   *`ClientIdentifier`*는 리모트 큐 관리자 이름입니다. 로컬 큐 관리자 이름, 큐 관리자 알리어스 또는 전송 큐 이름과 일치하지 않습니다.
   큐 이름이 정의되지 않습니다. 현재 텔레메트리(MQXR) 서비스에서는 `SYSTEM.MQTT.PUBLICATION.QUEUE`가 큐 이름으로 설정됩니다. MQTT v3 클라이언트는 큐를 지원하지 않습니다. 따라서 해석된 큐 이름이 클라이언트에 의해 무시됩니다.
   발행이 `SYSTEM.MQTT.TRANSMIT.QUEUE`에 넣어져 클라이언트로 보내질 수 있도록 로컬 큐 관리자 특성 `Default transmission queue`의 이름은 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 설정되어야 합니다.
2. *`ClientIdentifier`*는 이름이 *`ClientIdentifier`*인 큐 관리자 알리어스와 일치합니다.

   *`ClientIdentifier`*는 리모트 큐 관리자 이름입니다. 이는 큐 관리자 알리어스의 이름과 일치합니다.
   큐 관리자 알리어스는 *`ClientIdentifier`*와 함께 리모트 큐 관리자 이름으로 정의해야 합니다.
   큐 관리자 알리어스 정의에 전송 큐 이름을 설정하면 기본 전송을 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 설정할 필요가 없습니다.

표 *6. MQTT* 큐 관리자 알리어스의 이름 해석

	입력		Output		
`ClientIdentifier`	**큐 관리자 이름**	**큐 이름**	**큐 관리자 이름**	**큐 이름**	**전송 큐**
일치 항목 없음	*`ClientIdentifier`*	*undefined*	*`ClientIdentifier`*	*undefined*	기본 전송 큐. `SYSTEM.MQTT.TRANSMIT.QUEUE`
*`ClientIdentifier`*라는 큐 관리자 알리어스와 일치합니다.	*`ClientIdentifier`*	*undefined*	*`ClientIdentifier`*	*undefined*	`SYSTEM.MQTT.TRANSMIT.QUEUE`

이름 해석에 대한 추가 정보는 이름 해석을 참조하십시오.

IBM MQ 프로그램은 동일한 토픽으로 발행될 수 있습니다. 실행은 토픽에 구독을 하는 MQTT v3 클라이언트를 포함하여 해당 구독자에게 보내집니다.

예를 들어, 관리 토픽이 `CLUSTER(`*`clusterName`*`)` 속성이 있는 클러스터에서 작성되면 클러스터의 모든 애플리케이션이 클라이언트에 발행할 수 있습니다.

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

그림 *22. Windows*에서 클러스터 토픽 정의

참고: SYSTEM.MQTT.TRANSMIT.QUEUE에 클러스터 속성을 부여하지 마십시오.

MQTT 클라이언트 구독자 및 발행자는 다른 큐 관리자에 연결될 수 있습니다. 구독자 및 발행자는 일부 동일한 클러스터에 포함되거나 발행/구독 계층을 통해 연결될 수 있습니다. 발행은 IBM MQ을 사용하여 발행자에서 구독자로 전달됩니다.

클라이언트에 직접 메시지 송신

구독을 작성하고 구독 토픽과 일치하는 실행을 수신하는 클라이언트에 대한 대안은 직접적으로 메시지를 MQTT v3 클라이언트에 보냅니다. MQTT V3 클라이언트 애플리케이션은 직접적으로 메시지를 보낼 수 없지만, IBM MQ 애플리케이션과 같은 다른 애플리케이션은 보낼 수 있습니다.

IBM MQ 애플리케이션은 MQTT v3 클라이언트의 ClientIdentifier를 알아야 합니다. MQTT v3 클라이언트에는 큐가 없으므로 대상 큐 이름이 MQTT v3 애플리케이션 클라이언트 messageArrived 메소드에 토픽 이름으로 전달됩니다. 예를 들어, MQI 프로그램에서 클라이언트가 포함된 오브젝트 디스크립터를 ObjectQmgrName으로 작성하십시오.

```
MQOD.ObjectQmgrName = ClientIdentifier ;
MQOD.ObjectName = name ;
```

그림 *23.* 메시지를 *MQTT v3* 클라이언트 목적지에 보내기 위한 *MQI* 오브젝트 디스크립터

애플리케이션이 JMS을 사용하여 작성되면, 포인트-투-포인트 목적지를 작성하십시오. 예:

```
javax.jms.Destination jmsDestination =
(javax.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

그림 *24.* 메시지를 *MQTT v3* 클라이언트에 보내기 위한 *JMS* 목적지

비청구 메시지를 MQTT 클라이언트에 보내려면 리모트 큐 정의를 사용하십시오. 리모트 큐 관리자 이름은 클라이언트의 ClientIdentifier로 해석되어야 합니다. 전송 큐는 SYSTEM.MQTT.TRANSMIT.QUEUE로 해석되어야 합니다. 자세한 정보는 136 페이지의 표 7의 내용을 참조하십시오. 리모트 큐 이름은 어떤 것이든 가능합니다. 클라이언트는 이 이름을 토픽 문자열로 수신합니다.

표 *7. MQTT* 클라이언트 리모트 큐 정의의 이름 해석

입력		Output		
큐 이름	**큐 관리자 이름**	**큐 이름**	**큐 관리자 이름**	**전송 큐**
리모트 큐 정의의 이름	공백 또는 로컬 큐 관리자 이름	토픽 문자열로 사용되는 리모트 큐 이름	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

클라이언트가 연결된 경우 메시지가 MQTT 클라이언트에 직접 전송되며, 이는 messageArrived 메소드를 호출합니다. messageArrived 메소드를 참조하십시오.

클라이언트가 지속적 세션과 연결이 끊어진 경우 메시지는 SYSTEM.MQTT.TRANSMIT.QUEUE 에 저장됩니다. MQTT Stateless및 Stateful 세션을 참조하십시오. 클라이언트가 세션에 다시 연결할 때 클라이언트로 전달됩니다.

비지속 메시지를 보내는 경우, "최소 한 번" 서비스 품질(QoS=0)을 가진 클라이언트로 보내집니다. 지속 메시지를 클라이언트에 직접 보내는 경우, 기본적으로 "정확히 한 번" 서비스 품질(QoS=2)과 함께 보내집니다. 클라이언트에 지속 메커니즘이 없을 수 있기 때문에, 클라이언트는 직접적으로 보내진 메시지를 위해 승인하는 서비스 품질(QoS)을 줄일 수 있습니다. 클라이언트로 직접 보내지는 메시지를 위해 QoS(Quality of Service)를 줄이려면 토픽 DEFAULT.QoS에 대한 구독을 작성하십시오. 클라이언트가 지원할 수 있는 최대 서비스 품질(QoS)을 지정하십시오.

MQTT 클라이언트 식별, 권한 부여, 인증

텔레메트리(MQXR) 서비스는 MQTT 채널을 사용하여 MQTT 클라이언트 대신 IBM MQ 토픽을 발행하고 구독합니다. IBM MQ 관리자는 IBM MQ 권한 부여에 사용되는 MQTT 채널 ID를 구성합니다. 관리자는 채널에 대한 공통 ID를 정의하거나 채널에 연결된 클라이언트의 Username 또는 ClientIdentifier를 사용할 수 있습니다.

텔레메트리(MQXR) 서비스는 클라이언트 인증서를 사용하거나 클라이언트가 제공하는 Username을 사용하여 클라이언트를 인증할 수 있습니다. Username은 클라이언트에서 제공하는 비밀번호를 사용하여 인증됩니다.

요약: 클라이언트 ID는 클라이언트 ID의 선택입니다. 컨텍스트에 따라, 클라이언트는 ClientIdentifier, Username, 관리자에서 작성하는 공통 클라이언트 ID 또는 클라이언트 인증서에 의해 식별됩니다. 인증 검사에 사용되는 클라이언트 ID는 권한 부여에 사용되는 ID와 동일할 필요는 없습니다.

MQTT 클라이언트 프로그램은 MQTT 채널을 사용하여 서버로 보내지는 Username 및 Password를 설정합니다. 이는 연결을 암호화하고 인증하기 위해 필요한 SSL 특성을 설정할 수도 있습니다. 관리자는 MQTT 채널을 인증하는지 여부와 채널을 인증하는 방법을 결정합니다.

IBM MQ 오브젝트에 액세스하도록 MQTT 클라이언트에 권한을 부여하려면 ClientIdentifier에 권한을 부여하거나 클라이언트의 Username에 권한을 부여하거나 공통 클라이언트 ID에 권한을 부여하십시오. 클라이언트가 IBM MQ에 연결되도록 허용하려면 Username에 권한을 부여하거나 클라이언트 인증서를 사용하십시오. JAAS를 구성하여 Username을 인증하고 SSL을 구성하여 클라이언트 인증서를 인증하십시오.

클라이언트에서 Password를 설정하면, VPN을 사용하여 연결을 암호화하거나 SSL을 사용할 MQTT 채널을 구성하여 비밀번호를 개인용으로 유지하십시오.

클라이언트 인증서는 관리하기 어렵습니다. 이러한 이유로, 비밀번호 인증과 연관되는 위험을 감수할 수 있는 경우, 비밀번호 인증이 종종 클라이언트를 인증하기 위해 사용됩니다.

클라이언트 인증서를 관리하고 저장하기 위한 안전한 방법이 있는 경우, 인증서 인증에 의존할 수 있습니다. 그러나, 텔레메트리가 사용되는 환경의 유형에서 인증서가 안전하게 관리될 수 있는 경우는 드뭅니다. 대신, 클라이언트 인증서를 사용하는 디바이스의 인증이 서버에서 클라이언트 비밀번호를 인증하여 보완됩니다. 클라이언트 인증서 사용은 더 복잡하므로 매우 민감한 애플리케이션에서만 사용됩니다. 두 가지 형식의 인증 사용을 두 요소 인증이라고 합니다. 한 가지 요소(예: 비밀번호)를 알고 있고 다른 요소(예: 인증서)가 있어야 합니다.

매우 민감한 애플리케이션(예: 칩 앤 핀 디바이스)에서 디바이스는 제조할 때 내부 하드웨어와 소프트웨어가 도용되지 않도록 잠겨집니다. 신뢰할 수 있는 시간 제한된 클라이언트 인증서가 디바이스로 복사됩니다. 디바이스는 사용할 수 있는 위치에 배치됩니다. 디바이스를 사용할 때마다 비밀번호 또는 스마트 카드의 다른 인증서가 사용되어 추가적인 인증이 수행됩니다.

MQTT 클라이언트 ID 및 권한

권한 부여에 대한 공통 클라이언트 ID 및 클라이언트 ID(Username)를 사용하여IBM MQ 오브젝트에 액세스하십시오.

IBM MQ 관리자에게는 MQTT 채널의 ID를 선택하기 위한 세 가지 선택사항이 있습니다. 관리자는 클라이언트가 사용하는 MQTT 채널을 정의하거나 수정할 때 선택합니다. ID는 IBM MQ 토픽에 액세스 권한을 부여하기 위해 사용됩니다. 다음 순서로 선택됩니다.

1. 클라이언트 ID(USECLNTID 참조).
2. 관리자가 채널에 제공하는 ID(채널의 MCAUSER. MCAUSER 참조).

3. 이전 선택사항 중 어떤 것도 적용되지 않는 경우, MQTT 클라이언트에서 전달되는 사용자 이름(사용자 이름은 `MqttConnectOptions` 클래스의 속성입니다. 클라이언트가 서비스에 연결되기 전에 설정되어야 합니다. 해당 기본값은 NULL입니다).

문제점 예방: DISPLAY CHSTATUS(MQTT)에 의한 예가 클라이언트의 MCAUSER로서 명령을 내리기 때문에 이 프로세스에 의해 선택된 ID는 그 후에 언급됩니다. 이 ID는 선택 (2)에서 참조되는 채널의 MCAUSER와 반드시 동일한 ID는 아닙니다.

IBM MQ **setmqaut** 명령을 사용하여 MQTT 채널과 연관되는 ID에서 사용하도록 권한 부여되는 오브젝트 및 조치를 선택하십시오. 예를 들어, 다음 코드는 큐 관리자 `QM1`의 관리자가 제공하는 채널 ID `MQTTClient`에 권한을 부여합니다.

```
 setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

IBM MQ 오브젝트에 액세스하기 위해 *MQTT* 클라이언트 권한 부여

IBM MQ오브젝트에 발행하고 구독하기 위해 MQTT 클라이언트에 권한을 부여하려면 단계를 수행하십시오. 단계는 네 개의 대체 액세스 제어 패턴을 따릅니다.

시작하기 전에

MQTT 클라이언트는 텔레메트리 채널에 연결할 때 ID를 지정하여 IBM MQ에서 오브젝트에 액세스하기 위한 권한이 부여됩니다. IBM MQ 관리자는 IBM MQ 탐색기를 사용하여 세 가지 유형의 ID 중 하나의 클라이언트에 제공하는 텔레메트리 채널을 구성합니다.

1. `ClientIdentifier`
2. `Username`
3. 관리자가 채널에 지정하는 이름

사용되는 유형이 어떤 것이든 ID는 설치된 권한 서비스에 의해 프린시펄로서 IBM MQ에 정의되어야 합니다. Windows또는 Linux에 대한 기본 권한 서비스는 QAM(Object Authority Manager)이라고 합니다. OAM을 사용 중인 경우, ID는 사용자 ID로 정의되어야 합니다.

이 ID를 사용하여 IBM MQ에 정의된 토픽으로 발행하거나 구독하기 위한 권한을 클라이언트 또는 클라이언트 콜렉션에 제공하십시오. MQTT 클라이언트가 토픽을 구독하는 경우, ID를 사용하여 결과 발행물을 수신하기 위한 권한을 제공하십시오.

각각이 개별 액세스 권한을 요청하는 수만의 MQTT 클라이언트로 시스템을 관리하는 것은 어렵습니다. 하나의 솔루션은 공용 ID를 정의하는 것이고 공용 ID 중 하나와 개별 MQTT 클라이언트를 연관시키는 것입니다. 권한의 다른 결합을 정의하기 위해 필요한 수만큼의 공용 ID를 정의하십시오. 다른 솔루션은 수천명의 사용자가 운영 체제 보다 더 용이하게 처리할 수 있는 자체 권한 서비스를 작성하는 것입니다.

OAM을 사용하여, 두 가지 방법으로 MQTT 클라이언트를 공용 ID에 결합시킬 수 있습니다.

1. 관리자가 IBM MQ 탐색기를 사용하여 할당하는 다른 사용자 ID를 가진 각각의 다중 텔레메트리 채널을 정의하십시오. 다른 TCP/IP 포트 번호를 사용하여 연결하는 클라이언트는 다른 텔레메트리 채널과 연관되며 다른 ID가 지정됩니다.
2. 단일 텔레메트리 채널을 정의하지만, 각 클라이언트가 작은 사용자 ID 세트에서 `Username`을 선택하도록 하십시오. 관리자는 해당 ID로서 클라이언트 `Username`을 선택하도록 텔레메트리 채널을 구성합니다.

이 태스크에서 Telemetry 채널의 ID는 설정 방식에 관계없이 *`mqttUser`*라고 합니다. 클라이언트 콜렉션이 서로 다른 ID를 사용할 경우 클라이언트 콜렉션별로 하나씩 여러 *`mqttUsers`*를 사용하십시오. 태스크에서 OAM이 사용되는 경우 각 *`mqttUser`*이 사용자 ID여야 합니다.

이 태스크 정보

이 태스크에서 특정 요구사항에 따라 네 가지 액세스 제어 패턴 중 하나를 선택할 수 있습니다. 패턴은 액세스 제어의 단위에 따라 다릅니다.

- 139 페이지의 『액세스 제어 없음』

- 139 페이지의 『거친 액세스 제어』
- 139 페이지의 『중간 단위 액세스 제어』
- 139 페이지의 『세밀한 액세스 제어』

모델의 결과는 IBM MQ에 발행하고 구독하기 위해 권한의 *mqttUsers* 세트를 지정하는 것이고, IBM MQ로부터 발행물을 수신합니다.

액세스 제어 없음
MQTT 클라이언트는 IBM MQ 관리 권한을 받고 오브젝트에서 조치를 수행할 수 있습니다.

프로시저

1. 모든 MQTT 클라이언트의 ID로서 수행하기 위한 사용자 ID *mqttUser*를 작성하십시오.
2. *mqttUser* 를 mqm 그룹에 추가하십시오. Windows 의 그룹에 사용자 추가 또는 Linux 의 그룹에 사용자 추가 를 참조하십시오.

거친 액세스 제어
MQTT 클라이언트에 메시지를 발행 및 구독하고 이를 MQTT 클라이언트에 보내기 위한 권한이 있습니다. 기타 조치를 수행하거나 기타 오브젝트에 액세스하기 위한 권한이 없습니다.

프로시저

1. 모든 MQTT 클라이언트의 ID로서 수행하기 위한 사용자 ID *mqttUser*를 작성하십시오.
2. 모든 토픽을 발행 및 구독하고 발행물을 MQTT 클라이언트에 보내기 위한 *mqttUser*에 권한을 부여하십시오.

```
setmqaut -m qMgr -t topic -n SYSTEM.BASE.TOPIC -p mqttUser -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqttUser -all +put
```

중간 단위 액세스 제어
MQTT 클라이언트는 토픽의 다른 세트에 발행 및 구독하고 메시지를 MQTT 클라이언트에 보내기 위해 다양한 그룹으로 나뉘어집니다.

프로시저

1. 발행/구독 토픽 트리에서 다중 사용자 ID(*mqttUsers*)와 다중 관리 토픽을 작성하십시오.
2. 다양한 *mqttUsers*에게 서로 다른 토픽에 대한 권한을 부여하십시오.

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. 그룹 *mqtt*를 작성한 후 해당 그룹에 모든 *mqttUsers*를 추가하십시오.
4. Authorize *mqtt* to send topics to MQTT clients.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

세밀한 액세스 제어
MQTT 클라이언트는 오브젝트에서 조치를 수행하기 위한 그룹에 권한을 부여하는 액세스 제어의 기존 시스템으로 통합됩니다.

이 태스크 정보

사용자 ID가 필요한 권한 부여에 따라 하나 이상의 운영 체제 그룹으로 지정됩니다. IBM MQ 애플리케이션이 MQTT 클라이언트와 동일한 토픽 공간을 발행하고 구독하는 경우, 이 모델을 사용하십시오. 그룹은 `Publish` *X*, `Subscribe` *Y*, `mqtt`로 참조됩니다.

Publish *X*
Publish *X* 그룹의 멤버가 *topicX*에 발행될 수 있습니다.

Subscribe *Y*
Subscribe *Y* 그룹의 멤버는 *topicY*를 구독할 수 있습니다.

mqtt
mqtt 그룹의 구성원은 MQTT 클라이언트에 서적을 전송할 수 있습니다.

프로시저

1. 실행/구독 토픽 트리의 다중 관리 토픽에 할당되는 다중 그룹, Publish *X* 및 Subscribe *Y*를 작성하십시오.
2. 그룹 mqtt를 작성하십시오.
3. 부여되는 권한에 따라 여러 사용자 ID *mqttUsers*를 작성하고 해당 사용자를 임의의 그룹에 추가하십시오.
4. 서로 다른 Publish *X* 및 Subscribe *X* 그룹에 다른 토픽을 부여하고 *mqtt* 그룹이 MQTT 클라이언트에 메시지를 전송할 수 있는 권한을 부여합니다.

```
setmqaut -m qMgr -t topic -n topic1 -p Publish X -all +pub
setmqaut -m qMgr -t topic -n topic1 -p Subscribe X -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

비밀번호를 사용하여 MQTT 클라이언트 인증

클라이언트 비밀번호를 사용하여 Username을 인증하십시오. 클라이언트에 권한을 부여하기 위해 사용되는 다른 ID를 사용하여 토픽에 발행하고 구독하는 클라이언트를 인증할 수 있습니다.

텔레메트리(MQXR) 서비스는 JAAS를 사용하여 클라이언트 Username을 인증합니다. JAAS는 MQTT 클라이언트에서 제공하는 비밀번호를 사용합니다.

IBM MQ 관리자는 클라이언트가 연결하는 MQTT 채널을 구성하여 Username을 인증하는지 여부를 결정합니다. 클라이언트는 다른 채널에 지정될 수 있고 각 채널은 다른 방법으로 해당 클라이언트를 인증하기 위해 구성될 수 있습니다. JAAS를 사용하여 클라이언트를 인증해야 하는 메소드 및 클라이언트를 선택적으로 인증할 수 있는 메소드를 구성할 수 있습니다.

인증을 위한 ID 선택은 권한 부여 ID 선택에 영향을 미치지 않습니다. 관리상의 편의를 위해 권한 부여를 위한 공통 ID를 설정할 수 있지만 해당 ID를 사용하도록 사용자별로 인증하십시오. 다음 프로시저는 개별 사용자가 공용 ID를 사용하도록 인증하는 단계를 파악합니다.

1. IBM MQ 관리자는 IBM MQ 탐색기를 사용하여 MQTT 채널 ID를 임의의 이름 (예: MQTTClientUser) 으로 설정합니다.
2. IBM MQ 관리자는 토픽에 발행하고 구독하도록 MQTTClient에 권한을 부여합니다.

```
 setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

3. MQTT 클라이언트 애플리케이션 개발자는 서버에 연결하기 전에 MqttConnectOptions 오브젝트를 작성하고 사용자 이름 및 비밀번호를 설정합니다.
4. 보안 개발자는 비밀번호를 가지는 사용자 이름에 권한을 부여하기 위한 JAAS LoginModule을 작성하고 JAAS 구성 파일에 이를 포함시킵니다.
5. IBM MQ 관리자는 JAAS를 사용하여 클라이언트의 사용자 이름에 권한을 부여하기 위한 MQTT 채널을 구성합니다.

SSL을 사용하는 MQTT 클라이언트 인증

MQTT 클라이언트와 큐 관리자 사이의 연결은 항상 MQTT 클라이언트가 시작합니다. MQTT 클라이언트는 항상 SSL 클라이언트입니다. 서버의 클라이언트 인증 및 MQTT 클라이언트의 서버 인증은 모두 선택사항입니다.

개인용 서명 디지털 인증서를 사용하여 클라이언트를 제공하면 WebSphere MQ에 대해 MQTT 클라이언트를 인증할 수 있습니다. WebSphere MQ 관리자는 MQTT 클라이언트가 SSL을 사용하여 큐 관리자에 대해 인증하도록 강제로 실행할 수 있습니다. 클라이언트를 상호 인증의 부분으로만 요청할 수 있습니다.

SSL 사용의 대안으로서 IPsec 등과 같은 일종의 가성 사설망(VPN)은 TCP/IP 연결의 엔드 포인트를 인증합니다. VPN은 네트워크를 통해 플로우하는 각 IP 패킷을 암호화합니다. 이러한 VPN 연결이 설정되면 신뢰할 수 있는 네트워크가 설정된 것입니다. VPN 네트워크를 통해 TCP/IP를 사용하여 MQTT 클라이언트를 임시 채널에 연결할 수 있습니다.

SSL을 사용하는 클라이언트 인증 방식은 비밀 키를 가진 클라이언트에 따라 달라집니다. 본인확인정보는 자체 서명 인증서의 경우에 클라이언트의 개인 키이거나 인증 기관이 제공하는 키입니다. 키는 클라이언트의 디지털 인증서에 서명하는 데 사용됩니다. 해당하는 공개 키를 가진 사용자는 누구나 디지털 인증서를 확인할 수 있습니다. 인증서는 신뢰할 수 있거나, 체인 연결된 경우 인증서 체인을 통해 신뢰되는 루트 인증서로 추적될 수 있습니다. 클라이언트 확인은 클라이언트가 제공한 인증서 체인에 있는 모든 인증서를 서버로 보냅니다. 서버는 신뢰하는 인증서를 찾을 때까지 인증서 체인을 검사합니다. 신뢰되는 인증서는 자체 서명 인증서에서 생성된 공용 인증서이거나 일반적으로 인증 기관이 발행한 루트 인증서입니다. 마지막으로, 신뢰되는 인증서를 "라이브" 인증서 폐기 목록과 비교할 수 있는 선택적 단계가 있습니다.

신뢰되는 인증서는 인증 기관이 발행하거나 JRE 인증서 저장소에 이미 포함되었을 수 있습니다. 이는 자체 서명 인증서이거나 신뢰되는 인증서로서 텔레메트리 채널 키 저장소에 추가된 인증서일 수도 있습니다.

참고: 텔레메트리 채널에는 하나 이상의 텔레메트리 채널 및 클라이언트를 인증하는 데 필요한 모든 공용 인증서 둘 모두를 보유하는 결합된 키 저장소/신뢰 저장소가 있습니다. SSL 채널에는 키 저장소가 있어야 하고 채널 신뢰 저장소와 동일한 파일이므로 JRE 인증서 저장소는 절대 참조되지 않습니다. 클라이언트의 인증에 CA 루트 인증서가 필요한 경우 CA 루트 인증서가 이미 JRE 인증서 저장소에 있더라도 채널의 키 저장소에 루트 인증서를 배치해야 한다는 의미입니다. JRE 인증서 저장소는 절대 참조되지 않습니다.

클라이언트 인증이 카운터하려는 위협과 클라이언트 및 서버가 위협 카운터링 시에 수행하는 역할에 대해 생각하십시오. 클라이언트 인증서만을 인증하는 것은 시스템에 비인가 액세스를 방지하기에는 충분하지 않습니다. 다른 누군가가 클라이언트 디바이스를 보유한 경우 클라이언트 디바이스는 인증서 홀더의 권한을 사용하여 수행할 필요가 없습니다. 원치 않는 공격에 대응하는 방법으로 한 가지 방어책에만 의존하지 마십시오. 적어도 두 요소 인증 방법을 사용하고 개인용 정보에 대한 지식이 있는 인증서를 보조로 소유하십시오. 예를 들어, JAAS를 사용하여 서버가 발행한 비밀번호를 사용하여 클라이언트를 인증하십시오.

클라이언트 인증서에 대한 1차 위협은 이것이 올바르지 않은 사용자의 수중에 들어간다는 점입니다. 인증서는 클라이언트에서 비밀번호로 보호된 키 저장소에 보관됩니다. 이를 키 저장소에 배치하는 방법은 무엇입니까? MQTT 클라이언트가 키 저장소에 비밀번호를 가져오는 방법은 무엇입니까? 비밀번호 보호는 얼마나 안전합니까? 텔레메트리 디바이스는 종종 제거하기 쉬우므로 개인적으로 해킹될 수 있습니다. 디바이스 하드웨어가 변경할 수 없도록 설정되어야 합니까? 클라이언트측 인증서를 분배하고 보호하는 일은 인식하기가 어렵습니다. 이는 키 관리 문제점이라고 불립니다.

2차 위협은 디바이스가 의도하지 않은 방법으로 서버에 액세스하기 위해 잘못 사용되는 것입니다. 예를 들어, MQTT 애플리케이션이 변조되는 경우, 인증된 클라이언트 ID를 사용하여 서버 구성에서 약점을 이용할 수 있습니다.

SSL을 사용하여 MQTT 클라이언트를 인증하려면 텔레메트리 채널을 구성한 다음 클라이언트를 구성하십시오.

관련 개념

141 페이지의 『SSL을 사용하여 MQTT 클라이언트 인증을 위한 텔레메트리 채널 구성』
IBM MQ 관리자는 서버에서 텔레메트리 채널을 구성합니다. 각 채널은 서로 다른 포트 번호에서 TCP/IP 연결을 승인하도록 구성됩니다. SSL 채널은 키 파일에 대해 비밀번호 문구로 보호된 액세스로 구성됩니다. SSL 채널이 비밀번호 문구나 키 파일이 없는 상태로 정의되어 있을 경우 해당 채널은 SSL 연결을 승인하지 않습니다.

관련 정보

SSL을 사용하는 채널 인증을 위한 MQTT 클라이언트 구성

*SSL*을 사용하여 *MQTT* 클라이언트 인증을 위한 텔레메트리 채널 구성

IBM MQ 관리자는 서버에서 텔레메트리 채널을 구성합니다. 각 채널은 서로 다른 포트 번호에서 TCP/IP 연결을 승인하도록 구성됩니다. SSL 채널은 키 파일에 대해 비밀번호 문구로 보호된 액세스로 구성됩니다. SSL 채널이 비밀번호 문구나 키 파일이 없는 상태로 정의되어 있을 경우 해당 채널은 SSL 연결을 승인하지 않습니다.

SSL Telemetry 채널의 `com.ibm.mq.MQTT.ClientAuth` 특성을 `REQUIRED`로 설정하여 해당 채널에 연결되어 있는 모든 클라이언트가 디지털 인증서를 확인했는지에 대해 강제로 입증하도록 합니다. 클라이언트 인증서

는 인증 기관의 인증서를 사용하여 인증되며 신뢰되는 루트 인증서로 안내됩니다. 클라이언트 인증서가 자체 서명되었거나 인증 기관에서 발급한 인증서에 의해 서명된 경우 공개적으로 서명된 클라이언트 인증서나 인증 기관이 서버에 안전하게 보관되어야 합니다.

텔레메트리 채널 키 저장소에 공용으로 서명된 클라이언트 인증서 또는 인증 기관의 인증서를 배치하십시오. 서버에서 공개적으로 서명된 인증서는 별도의 신뢰 저장소가 아니라 개인적으로 서명된 인증서로 동일한 키 파일에 저장됩니다.

서버는 모든 공용 인증서와 서버에서 보유한 암호 스위트를 사용하여 전송되는 모든 클라이언트 인증서의 서명을 확인합니다. 서버는 키 체인을 확인합니다. 큐 관리자는 인증서 폐기 목록(CRL)에 대응하는 인증서를 테스트하도록 구성될 수 있습니다. 큐 관리자 폐기 이름 목록 특성은 SSLCRLNL입니다.

클라이언트가 보낸 모든 인증서가 서버 키 저장소에 있는 인증서에 의해 확인되면 클라이언트가 인증됩니다.

IBM MQ 관리자는 JAAS를 사용하여 클라이언트 `Password`로 클라이언트의 `UserName` 또는 `ClientIdentifier`를 확인하기 위해 동일한 텔레메트리 채널을 구성할 수 있습니다.

다중 텔레메트리 채널에 동일한 키 저장소를 사용할 수 있습니다.

디바이스에서 비밀번호로 보호되는 클라이언트 키 저장소에 있는 최소 하나의 디지털 인증서를 확인하면 클라이언트가 서버에 대해 인증됩니다. 디지털 인증서는 IBM MQ에 의한 인증에만 사용됩니다. 클라이언트의 TCP/IP 주소를 검증하거나, 권한 부여 또는 회계를 위한 클라이언트 ID를 설정하는 데에는 사용되지 않습니다. 서버에 의해 채택된 클라이언트의 ID는 `Username` 또는 클라이언트의 `ClientIdentifier` 또는 IBM MQ 관리자에 의해 작성된 ID입니다.

클라이언트 인증에 SSL 암호 스위트를 사용할 수도 있습니다. SHA-2 암호 스위트를 사용하려는 경우 143 페이지의 『MQTT 채널과 함께 SHA-2 암호 스위트를 사용하기 위한 시스템 요구사항』의 내용을 참조하십시오.

관련 개념

143 페이지의 『SSL을 사용하여 채널 인증에 대한 텔레메트리 채널 구성』
IBM MQ 관리자는 서버에서 텔레메트리 채널을 구성합니다. 각 채널은 서로 다른 포트 번호에서 TCP/IP 연결을 승인하도록 구성됩니다. SSL 채널은 키 파일에 대해 비밀번호 문구로 보호된 액세스로 구성됩니다. SSL 채널이 비밀번호 문구나 키 파일이 없는 상태로 정의되어 있을 경우 해당 채널은 SSL 연결을 승인하지 않습니다.

관련 정보

DEFINE CHANNEL(MQTT)
ALTER CHANNEL(MQTT)
CipherSpecs 및 CipherSuites

SSL을 사용하여 텔레메트리 채널 인증

MQTT 클라이언트와 큐 관리자 사이의 연결은 항상 MQTT 클라이언트가 시작합니다. MQTT 클라이언트는 항상 SSL 클라이언트입니다. 서버의 클라이언트 인증 및 MQTT 클라이언트의 서버 인증은 모두 선택사항입니다.

클라이언트가 익명 연결을 지원하는 CipherSpec을 사용하도록 구성되지 않는 한 클라이언트는 항상 서버를 인증하려고 시도합니다. 인증에 실패하면 연결이 설정되지 않습니다.

SSL 사용의 대안으로서 IPsec 등과 같은 일종의 가성 사설망(VPN)은 TCP/IP 연결의 엔드 포인트를 인증합니다. VPN은 네트워크를 통해 플로우하는 각 IP 패킷을 암호화합니다. 이러한 VPN 연결이 설정되면 신뢰할 수 있는 네트워크가 설정된 것입니다. VPN 네트워크를 통해 TCP/IP를 사용하여 MQTT 클라이언트를 임시 채널에 연결할 수 있습니다.

SSL을 사용한 서버 인증을 통해 기밀 정보를 보낼 서버를 인증합니다. 클라이언트는 서버에서 보낸 인증서와 일치하는 검사, 신뢰 저장소에 있는 인증서 또는 JRE `cacerts` 저장소에 있는 인증서를 수행합니다.

JRE 인증 저장소는 JKS 파일 `cacerts`입니다. 이 파일은 `JRE InstallPath\lib\security\`에 있습니다. 이 파일은 `changeit`라는 기본 비밀번호로 설치됩니다. JRE 인증서 저장소에서 신뢰할 수 있는 인증서나 클라이언트 신뢰 저장소에서 신뢰할 수 있는 인증서 중 하나를 저장할 수 있습니다. 두 저장소를 모두 사용할 수는 없습니다. 클라이언트가 신뢰하는 공개 인증서를 다른 Java 애플리케이션에서 사용하는 인증서와 별도로 보관하려면 클라이언트 신뢰 저장소를 사용하십시오. 클라이언트에서 실행 중인 모든 Java 애플리케이션에 대한 공통 인증서 저장소를 사용하려면 JRE 인증서 저장소를 사용하십시오. JRE 인증서 저장소를 사용하기로 결정하면 해당 저장소에 속한 인증서를 검토하여 이러한 인증서를 신뢰할 수 있는지 확인합니다.

다른 신뢰 제공자를 통해 JSSE 구성을 수정할 수 있습니다. 신뢰 제공자를 사용자 정의하여 인증서에 대해 다른 검사를 수행할 수도 있습니다. MQTT 클라이언트를 사용한 일부 OGSi 환경에서 환경은 다른 신뢰 제공자를 제공합니다.

SSL을 통해 텔레메트리 채널을 인증하려면 서버와 클라이언트를 구성하십시오.

SSL을 사용하여 채널 인증에 대한 텔레메트리 채널 구성

IBM MQ 관리자는 서버에서 텔레메트리 채널을 구성합니다. 각 채널은 서로 다른 포트 번호에서 TCP/IP 연결을 승인하도록 구성됩니다. SSL 채널은 키 파일에 대해 비밀번호 문구로 보호된 액세스로 구성됩니다. SSL 채널이 비밀번호 문구나 키 파일이 없는 상태로 정의되어 있을 경우 해당 채널은 SSL 연결을 승인하지 않습니다.

텔레메트리 채널이 서버에 사용할 키 저장소에서 해당 개인 키로 서명되는 서버의 디지털 인증서를 저장하십시오. 키 체인을 클라이언트로 전송하려면 키 스토어의 키 체인에 인증서를 저장하십시오. SSL을 사용하기 위해 IBM MQ 탐색기를 사용하여 텔레메트리 채널을 구성하십시오. 이 채널은 키 저장소에 대한 경로 및 키 저장소에 액세스하는 데 필요한 비밀번호 문구와 함께 제공됩니다. 채널의 TCP/IP 포트 번호를 설정하지 않는 경우, SSL 텔레메트리 채널 포트 번호의 기본값은 8883입니다.

채널 인증에 SSL 암호 스위트를 사용할 수도 있습니다. SHA-2 암호 스위트를 사용하려는 경우 143 페이지의 『MQTT 채널과 함께 SHA-2 암호 스위트를 사용하기 위한 시스템 요구사항』의 내용을 참조하십시오.

관련 개념

141 페이지의 『SSL을 사용하여 MQTT 클라이언트 인증을 위한 텔레메트리 채널 구성』
IBM MQ 관리자는 서버에서 텔레메트리 채널을 구성합니다. 각 채널은 서로 다른 포트 번호에서 TCP/IP 연결을 승인하도록 구성됩니다. SSL 채널은 키 파일에 대해 비밀번호 문구로 보호된 액세스로 구성됩니다. SSL 채널이 비밀번호 문구나 키 파일이 없는 상태로 정의되어 있을 경우 해당 채널은 SSL 연결을 승인하지 않습니다.

관련 정보

DEFINE CHANNEL(MQTT)
ALTER CHANNEL(MQTT)
CipherSpecs 및 CipherSuites

MQTT 채널과 함께 SHA-2 암호 스위트를 사용하기 위한 시스템 요구사항

SHA-2 암호 스위트를 지원하는 Java의 버전을 사용할 경우, 이러한 스위트를 사용하여 MQTT(텔레메트리) 채널 및 클라이언트 앱을 보안할 수 있습니다.

텔레메트리 (MQXR) 서비스를 포함하는 IBM MQ 8.0 의 경우 최소 Java 버전은 IBM , SR6의 Java 7입니다. SHA-2 암호 스위트는 IBM, SR4부터 기본적으로 Java 7에서 지원됩니다. 따라서 텔레메트리(MQXR) 서비스와 함께 SHA-2 암호 스위트를 사용하여 MQTT(텔레메트리) 채널을 보안할 수 있습니다.

다른 JRE와 함께 MQTT 클라이언트를 실행 중인 경우, SHA-2 암호 스위트 또한 지원하는지 확인해야 합니다.

관련 개념

143 페이지의 『SSL을 사용하여 채널 인증에 대한 텔레메트리 채널 구성』
IBM MQ 관리자는 서버에서 텔레메트리 채널을 구성합니다. 각 채널은 서로 다른 포트 번호에서 TCP/IP 연결을 승인하도록 구성됩니다. SSL 채널은 키 파일에 대해 비밀번호 문구로 보호된 액세스로 구성됩니다. SSL 채널이 비밀번호 문구나 키 파일이 없는 상태로 정의되어 있을 경우 해당 채널은 SSL 연결을 승인하지 않습니다.

관련 정보

텔레메트리(MQXR) 서비스
DEFINE CHANNEL(MQTT)
ALTER CHANNEL(MQTT)

텔레메트리 채널의 발행물 개인정보 보호

리모트 측정 채널 전체에 걸쳐 각각 방향으로 보내진 MQTT 공개의 프라이버시는 연결 위에서 전송을 암호화하기 위해 SSL을 사용하여 보안화됩니다.

텔레메트리 채널에 연결되는 MQTT 클라이언트는 SSL을 사용하여 대칭 키 암호화를 사용하는 채널에서 전송된 발행물의 개인정보 보호를 보호합니다. 엔드 포인트가 인증되지 않으므로 채널 암호화만 신뢰할 수는 없습니다. 개인정보 보호와 서버 또는 상호 인증을 결합하십시오.

SSL 사용의 대안으로서 IPsec 등과 같은 일종의 가성 사설망(VPN)은 TCP/IP 연결의 엔드 포인트를 인증합니다. VPN은 네트워크를 통해 플로우하는 각 IP 패킷을 암호화합니다. 이러한 VPN 연결이 설정되면 신뢰할 수 있는 네트워크가 설정된 것입니다. VPN 네트워크를 통해 TCP/IP를 사용하여 MQTT 클라이언트를 임시 채널에 연결할 수 있습니다.

채널을 암호화하고 서버를 인증하는 일반 구성의 경우, 142 페이지의 『SSL을 사용하여 텔레메트리 채널 인증』의 내용을 참조하십시오.

서버를 인증하지 않고 SSL 연결을 암호화하면 연결이 중간자 공격(man-in-the-middle attack)에 노출됩니다. 교환하는 정보가 도청에 대해 보호되는 경우에도 누구와 교환하는지 알지 못합니다. 네트워크를 제어하지 않는 경우, IP 전송을 인터셉트하고 엔드 포인트로 위장하는 사람에게 노출됩니다.

서버를 인증하지 않고 익명 SSL을 지원하는 Diffie-Hellman 키 교환 CipherSpec을 사용하여 암호화된 SSL로 연결할 수 있습니다. 클라이언트와 서버 사이에 공유되고 SSL 전송을 암호화하는 데 사용되는 마스터 보안은 개인적으로 사인된 서버 인증서를 교환하지 않고 설정됩니다.

익명 연결은 안전하지 않으므로 대부분의 SSL 구현은 기본적으로 익명 CipherSpec을 사용하지 않습니다. 텔레메트리 채널에서 SSL 연결에 대한 클라이언트 요청이 승인되는 경우, 채널에는 비밀번호 문구로 보호되는 키 저장소가 있어야 합니다. 기본적으로 SSL 구현은 익명 CipherSpec을 사용하지 않으므로 키 저장소에는 클라이언트가 인증할 수 있는 개인적으로 사인된 인증서가 포함되어 있어야 합니다.

익명 CipherSpec을 사용하는 경우 서버 키 저장소가 있어야 하지만 개인적으로 서명된 인증서를 포함할 필요는 없습니다.

암호화된 연결을 설정하는 다른 방법은 클라이언트에서 신뢰 제공자를 사용자 고유의 구현으로 바꾸는 것입니다. 사용자의 신뢰 제공자는 서버 인증서를 인증하지 않지만 연결이 암호화됩니다.

주의: MQTT 에서 TLS를 사용하는 경우 대형 메시지를 사용할 수 있지만 이를 수행할 때 성능에 영향을 미칠 수 있습니다. MQTT 는 작은 메시지를 처리하기 위해 최적화됩니다 (일반적으로 크기는 1KB - 1MB).

MQTT Java 클라이언트 및 텔레메트리 채널의 SSL 구성

MQTT Java 클라이언트와 텔레메트리 채널을 인증하기 위해 SSL을 구성하고 그 사이의 메시지 전송을 암호화하십시오.MQTT Java clients use Java Secure Socket Extension (JSSE) to connect telemetry channels using SSL. SSL 사용의 대안으로서 IPsec 등과 같은 일종의 가성 사설망(VPN)은 TCP/IP 연결의 엔드 포인트를 인증합니다. VPN은 네트워크를 통해 플로우하는 각 IP 패킷을 암호화합니다. 이러한 VPN 연결이 설정되면 신뢰할 수 있는 네트워크가 설정된 것입니다. VPN 네트워크를 통해 TCP/IP를 사용하여 MQTT 클라이언트를 임시 채널에 연결할 수 있습니다.

TCP/IP를 통해 SSL 프로토콜을 사용하도록 Java MQTT 클라이언트와 텔레메트리 채널 사이의 연결을 구성할 수 있습니다. 보안 개념은 사용자가 JSSE를 사용하도록 SSL을 구성하는 방법에 따라 다릅니다. 가장 안전한 구성에서 시작하여 세 가지 다른 보안 레벨을 구성할 수 있습니다.

1. 신뢰받는 MQTT 클라이언트만 연결되도록 하십시오. 신뢰받는 텔레메트리 채널에만 MQTT 클라이언트를 연결시키십시오. 클라이언트와 큐 관리자 사이의 메시지를 암호화하십시오. 140 페이지의 『SSL을 사용하는 MQTT 클라이언트 인증』의 내용을 참조하십시오
2. 신뢰받는 텔레메트리 채널에만 MQTT 클라이언트를 연결시키십시오. 클라이언트와 큐 관리자 사이에 메시지를 암호화하십시오. 142 페이지의 『SSL을 사용하여 텔레메트리 채널 인증』의 내용을 참조하십시오.
3. 클라이언트와 큐 관리자 사이에 메시지를 암호화하십시오. 143 페이지의 『텔레메트리 채널의 발행물 개인정보 보호』의 내용을 참조하십시오.

JSSE 구성 매개변수

SSL 연결 구성 방법을 대체하려면 JSSE 매개변수를 수정하십시오. JSSE 구성 매개변수는 세 가지 세트로 배열됩니다.

1. IBM MQ Telemetry 채널
2. MQTT Java 클라이언트
3. JRE

IBM MQ 탐색기를 사용하여 텔레메트리 채널 매개변수를 구성하십시오.
MqttConnectionOptions.SSLProperties 속성에서 MQTT Java 클라이언트 매개변수를 설정하십시오.
클라이언트 및 서버 모두의 JRE 보안 디렉토리에서 파일을 편집하여 JRE 보안 매개변수를 수정하십시오.

IBM MQ Telemetry 채널

IBM MQ 탐색기를 사용하여 모든 텔레메트리 채널 SSL 매개변수를 설정하십시오.

ChannelName

ChannelName은 모든 채널의 필수 매개변수입니다.

채널 이름은 특정 포트 번호와 연관된 채널을 식별합니다. 사용자를 돕기 위한 이름 채널은 MQTT 클라이언트 세트를 관리합니다.

PortNumber

PortNumber는 모든 채널에서 선택적 매개변수입니다. 기본값은 TCP 채널의 경우 1883으로 설정되고 SSL 채널의 경우 8883으로 설정됩니다.

이 채널과 연관된 TCP/IP 포트 번호입니다. MQTT 클라이언트는 채널에 대해 정의된 포트를 지정하여 채널에 연결됩니다. 채널에 SSL 특성이 있는 경우, 클라이언트는 SSL 프로토콜을 사용하여 연결해야 합니다. 예:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

KeyFileName은 SSL 채널의 필수 매개변수입니다. TCP 채널의 경우에는 생략되어야 합니다.

KeyFileName은 제공하는 디지털 인증서를 포함하는 Java 키 저장소에 대한 경로입니다. 서버에서 키 저장소의 유형으로 JKS, JCEKS 또는 PKCS12를 사용하십시오.

다음 파일 확장자 중 하나를 사용하여 키 저장소 유형을 식별하십시오.

```
.jks
.jceks
.p12
.pkcs12
```

다른 파일 확장자를 사용하는 키 저장소를 JKS 키 저장소라고 가정합니다.

서버의 키 저장소 유형과 클라이언트의 다른 키 저장소 유형을 결합할 수 있습니다.

키 저장소에 서버의 개인용 인증서를 저장하십시오. 이 인증서를 서버 인증서라고 합니다. 인증서는 자체 서명되거나 서명 기관에서 서명한 인증서 체인의 일부가 될 수 있습니다.

인증서 체인을 사용하는 경우에는 연관된 인증서를 서버 키 저장소에 저장하십시오.

서버 인증서 및 해당 인증서 체인의 모든 인증서는 클라이언트로 보내져 서버 ID를 인증합니다.

ClientAuth 를 Required로 설정한 경우 키 저장소는 클라이언트를 인증하는 데 필요한 인증서를 포함해야 합니다. 클라이언트가 자체 서명 인증서 또는 인증서 체인을 송신하면 이를 키 저장소의 인증서와 대조 확인한 후 클라이언트가 인증됩니다. 인증서 체인을 사용하면 클라이언트가 다양한 클라이언트 인증서를 사용하여 실행되더라도 하나의 인증서로 많은 클라이언트를 확인할 수 있습니다.

PassPhrase

PassPhrase는 SSL 채널에 대한 필수 매개변수입니다. TCP 채널의 경우에는 생략되어야 합니다.

비밀번호 문구는 키 저장소를 보호하기 위해 사용됩니다.

ClientAuth

ClientAuth는 선택적 SSL 매개변수입니다. 클라이언트 인증으로 기본값이 설정됩니다. TCP 채널의 경우에는 생략되어야 합니다.

텔레메트리(MQXR) 서비스에서 클라이언트를 인증하도록 하려면 ClientAuth를 설정한 후 클라이언트가 텔레메트리 채널에 연결하도록 허용하십시오.

ClientAuth를 설정하면 클라이언트는 SSL을 사용하여 서버에 연결하고 서버를 인증해야 합니다. ClientAuth 설정에 대한 응답으로 클라이언트는 디지털 인증서 및 키 저장소에 있는 다른 모든 인증서를 서버로 송신합니다. 이 디지털 인증서는 클라이언트 인증서라고 합니다. 이러한 인증서는 채널 키 저장소 및 JRE cacerts 저장소에 있는 인증서에 대응하여 인증됩니다.

CipherSuite

CipherSuite는 선택적 SSL 매개변수입니다. 기본값은 사용 가능한 모든 CipherSpec을 시도하는 것입니다. TCP 채널의 경우에는 생략되어야 합니다.

특정 CipherSpec을 사용하려는 경우에는 CipherSuite를 SSL 연결을 설정하는 데 사용해야 하는 CipherSpec 이름으로 설정하십시오.

텔리메트리 서비스와 MQTT 클라이언트는 각 마지막에 사용으로 설정되는 모든 CipherSpec으로부터 공통 CipherSpec을 협상합니다. 연결의 한 끝 또는 양 끝에 특정 CipherSpec 지정된 경우에는 반대쪽 끝에 있는 CipherSpec과 일치시켜야 합니다.

JSSE에 추가 제공자를 추가하여 추가 암호를 설치하십시오.

FIPS(Federal Information Processing Standard)

FIPS는 선택적 설정입니다. 기본적으로 설정되지 않습니다.

큐 관리자의 특성 패널 또는 **runmqsc**를 사용하여 SSLFIPS를 설정하십시오. SSLFIPS는 FIPS 인증 알고리즘만 사용되는지 여부를 지정합니다.

폐기 이름 목록

폐기 이름 목록은 선택적 설정입니다. 기본적으로 설정되지 않습니다.

큐 관리자의 특성 패널 또는 **runmqsc**를 사용하여 SSLCRLNL을 설정하십시오. SSLCRLNL은 인증서 폐기 위치를 제공하는 데 사용되는 인증 정보 오브젝트의 이름 목록을 지정합니다.

SSL 특성을 설정하는 다른 큐 관리자 매개변수는 사용되지 않습니다.

MQTT Java 클라이언트

MqttConnectionOptions.SSLProperties에서 Java 클라이언트를 위한 SSL 특성을 설정하십시오. 예:

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

특정 특성의 이름 및 값은 MqttConnectOptions 클래스에서 설명됩니다.MQTT 클라이언트 라이브러리에 대한 클라이언트 API 문서에 대한 링크는 MQTT 클라이언트 프로그래밍 참조를 참조하십시오.

프로토콜

Protocol은 선택사항입니다.

프로토콜은 텔레메트리 서버와 협상할 때 선택됩니다. 특정 프로토콜이 필요한 경우에는 선택할 수 있습니다. 텔레메트리 서버가 해당 프로토콜을 지원하지 않는 경우, 연결이 실패합니다.

ContextProvider

ContextProvider는 선택사항입니다.

KeyStore

KeyStore는 선택사항입니다. 서버에 ClientAuth가 설정되어 클라이언트 인증을 강제 실행하는 경우 이를 구성하십시오.

개인 키를 사용하여 서명된 클라이언트의 디지털 인증서를 키 저장소에 저장하십시오. 키 저장소 경로와 비밀번호를 지정하십시오. 유형 및 제공자는 선택사항입니다. JKS는 기본 유형이며 IBMJCE는 기본 제공자입니다.

새 키 저장소 제공자를 추가하는 클래스를 참조하려면 다른 키 저장소 제공자를 지정하십시오. 키 관리자 이름을 설정하여 KeyManagerFactory를 인스턴스화하려면 키 저장소 제공자가 사용하는 알고리즘 이름을 전달하십시오.

TrustStore

TrustStore는 선택사항입니다. JRE cacerts 저장소에서 신뢰하는 모든 인증서를 배치할 수 있습니다.

클라이언트에 다른 신뢰 저장소를 사용하려면 신뢰 저장소를 구성하십시오. cacerts에 루트 인증서가 이미 저장되어 있는 잘 알려진 CA에서 실행하는 인증서를 서버에서 사용하고 있으면 신뢰 저장소를 구성할 수 없습니다.

공용으로 서명된 서버 인증서 또는 루트 인증서를 신뢰 저장소에 추가하고 신뢰 저장소의 경로와 비밀번호를 지정하십시오. JKS는 기본 유형이며 IBMJCE는 기본 제공자입니다.

새 신뢰 저장소 제공자를 추가하는 클래스를 참조하려면 다른 신뢰 저장소 제공자를 지정하십시오. 신뢰 관리자 이름을 설정하여 TrustManagerFactory를 인스턴스화하려면 키 저장소 제공자가 사용하는 알고리즘 이름을 전달하십시오.

JRE

클라이언트 및 서버의 SSL 동작에 영향을 주는 다른 측면의 Java 보안이 JRE에 구성되어 있습니다. Windows의 구성 파일은 *Java Installation Directory*\jre\lib\security에 있습니다. 다음 표에서는 IBM MQ와 함께 제공되는 JRE를 사용하는 경우의 경로를 보여줍니다.

표 *8. JRE SSL* 구성 파일의 플랫폼별 파일 경로

플랫폼	**파일 경로**
Windows	*WMQ Installation Directory*\java\jre\lib\security
기타 유닉스 및 Linux 플랫폼	*WMQ Installation Directory*/java/jre64/jre/lib/security

잘 알려진 인증 기관

cacerts 파일에는 잘 알려진 인증 기관의 루트 인증서가 포함되어 있습니다. 신뢰 저장소를 지정하지 않은 경우 기본적으로 cacerts가 사용됩니다. cacerts 저장소를 사용하거나 신뢰 저장소를 제공하지 않은 경우 cacerts의 서명자 목록을 검토하고 편집하여 보안 요구사항을 충족해야 합니다.

IBM 키 관리 유틸리티를 실행하는 IBM MQ 명령 strmqikm을 사용하여 cacerts를 열 수 있습니다. 비밀번호 changeit를 사용하여 cacerts를 JKS 파일로 여십시오. 비밀번호를 수정하여 파일을 보호하십시오.

보안 클래스 구성

java.security 파일을 사용하여 추가 보안 제공자 및 기타 기본 보안 특성을 등록하십시오.

권한

java.policy 파일을 사용하여 자원에 부여된 권한을 수정하십시오. javaws.policy는 javaws.jar에 권한을 부여합니다.

암호화 강도

일부 JRE는 암호화 강도가 약해진 상태로 배송됩니다. 키를 키 저장소로 가져올 수 없는 경우, 약해진 암호화 강도가 원인일 수 있습니다. 또한, **strmqikm** 명령을 사용하여 **ikeyman**을 시작하려고 시도하거나, IBM 개발자 킷, 보안 정보로부터 강하지만 제한된 관할 파일을 다운로드하십시오.

중요사항: 비밀번호화 소프트웨어의 원산 국가는 다른 나라로 가져오기, 소유권, 사용 또는 다시 내보내기를 수행하는 데 제한사항이 있습니다. 제한 없는 정책 파일을 다운로드하거나 사용하려면 사용자 국가

의 법을 확인하십시오. 암호화 소프트웨어의 수입, 소유, 사용 및 재수출과 관련된 규정 및 정책을 조사하여 허용되는지 판별하십시오.

클라이언트가 서버에 연결할 수 있도록 신뢰 제공자 수정

예는 신뢰 제공자를 추가하고 MQTT 클라이언트 코드로부터 그것에 참조 사항을 다는 방법을 설명합니다. 예는 클라이언트 또는 서버의 인증을 수행하지 않습니다. 따라서 SSL 연결은 인증되지 않고 암호화됩니다.

148 페이지의 그림 25 의 코드 스니펫은 MQTT 클라이언트에 대한 `AcceptAllProviders` 신뢰 제공자 및 신뢰 관리자를 설정합니다.

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

그림 *25. MQTT* 클라이언트 코드 스니펫

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
private static final long serialVersionUID = 1L;
public AcceptAllProvider() {
super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
put("TrustManagerFactory.TrustAllCertificates",
AcceptAllTrustManagerFactory.class.getName());
}
```

그림 *26. AcceptAllProvider.java*

```
protected static class AcceptAllTrustManagerFactory extends
javax.net.ssl.TrustManagerFactorySpi {
public AcceptAllTrustManagerFactory() {}
protected void engineInit(java.security.KeyStore keystore) {}
protected void engineInit(
javax.net.ssl.ManagerFactoryParameters parameters) {}
protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
}
```

그림 *27. AcceptAllTrustManagerFactory.java*

```
protected static class AcceptAllX509TrustManager implements
javax.net.ssl.X509TrustManager {
public void checkClientTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Client authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public void checkServerTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Server authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public java.security.cert.X509Certificate[] getAcceptedIssuers() {
return new java.security.cert.X509Certificate[0];
}
private static void report(String string) {
System.out.println(string);
}
}
```

그림 *28. AcceptAllX509TrustManager.java*

텔레메트리 채널 JAAS 구성

클라이언트에서 보내는 Username을 인증하기 위해 JAAS를 구성하십시오.

IBM MQ 관리자는 JAAS를 사용하여 클라이언트 인증을 요청하는 MQTT 채널을 구성합니다. JAAS 인증을 수행할 각 채널에 대한 JAAS 구성 이름을 지정하십시오. 채널은 동일한 JAAS 구성을 모두 사용할 수 있거나 다른 JAAS 구성을 사용할 수 있습니다. 구성은 *WMQData directory*\qmgrs*qMgrName*\mqxr\jaas.config 에 정의됩니다.

jaas.config 파일은 JAAS 구성 이름에 의해 조직됩니다. 로그인 구성의 목록은 각 구성 이름 아래에 있습니다. 150 페이지의 그림 29의 내용을 참조하십시오.

JAAS에서는 네 개의 표준 로그인 모듈을 제공합니다. 표준 NT 및 UNIX 로그인 모듈은 제한된 값입니다.

JndiLoginModule
JNDI(Java Naming and Directory Interface)에 구성된 디렉토리 서비스에 대응하여 인증합니다.

Krb5LoginModule
Kerberos 프로토콜을 사용하여 인증합니다.

NTLoginModule
현재 사용자에 대한 NT 보안 정보를 사용하여 인증합니다.

UnixLoginModule
현재 사용자에 대한 UNIX 보안 정보를 사용하여 인증합니다.

NTLoginModule 또는 UnixLoginModule 를 사용하는 문제점은 텔레메트리 (MQXR) 서비스가 MQTT 채널의 ID가 아니라 mqm ID로 실행되는 것입니다. mqm은 클라이언트의 ID가 아닌 인증을 위해 NTLoginModule 또는 UnixLoginModule로 전달되는 ID입니다.

이 문제점을 해결하려면 자체 로그인 모듈을 작성하거나 기타 표준 로그인 모듈을 사용하십시오. 샘플 JAASLoginModule.java 에 IBM MQ Telemetry가 제공됩니다. 이는 javax.security.auth.spi.LoginModule 인터페이스의 구현입니다. 이를 사용하여 자체 인증 메소드를 개발하십시오.

사용자가 제공하는 새 LoginModule 클래스는 텔레메트리(MQXR) 서비스의 클래스 경로에 있어야 합니다. 클래스 경로에 있는 IBM MQ 디렉토리에 클래스를 배치하지 마십시오. 자체 디렉토리를 작성하고 텔레메트리 (MQXR) 서비스에 대한 전체 클래스 경로를 정의하십시오.

service.env 파일에서 클래스 경로를 설정하여 텔레메트리(MQXR) 서비스가 사용하는 클래스를 보강할 수 있습니다. CLASSPATH는 대문자여야 하고 클래스 경로 명령문에는 리터럴만 포함될 수 있습니다. 변수를

CLASSPATH에서 사용할 수 없습니다. 예를 들어, CLASSPATH=%CLASSPATH%는 올바르지 않습니다. 텔레메트리(MQXR) 서비스는 자체 CLASSPATH를 설정합니다. service.env에 정의된 CLASSPATH는 해당 파일에 추가됩니다.

텔레메트리(MQXR) 서비스가 MQTT 채널에 연결된 클라이언트에 대해 Username과 Password을 리턴하는 두 개의 콜백을 제공합니다. 사용자 이름 및 비밀번호 는 MqttConnectOptions 오브젝트에 설정됩니다. Username 및 Password에 액세스하는 방법에 대한 예제는 150 페이지의 그림 30의 내용을 참조하십시오.

예:

이름 지정된 구성 MQXRConfig가 있는 JAAS 구성 파일의 예입니다.

```
MQXRConfig {
samples.JAASLoginModule required debug=true;
//com.ibm.security.auth.module.NTLoginModule required;
//com.ibm.security.auth.module.Krb5LoginModule required
//         principal=principal@your_realm
//         useDefaultCcache=TRUE
//         renewTGT=true;
//com.sun.security.auth.module.NTLoginModule required;
//com.sun.security.auth.module.UnixLoginModule required;
//com.sun.security.auth.module.Krb5LoginModule required
//         useTicketCache="true"
//         ticketCache="${user.home}${/}tickets";
};
```

그림 *29.* 샘플 *jaas.config* 파일

JAAS 로그인 모듈의 예는 MQTT 클라이언트에서 제공하는 Username 및 Password를 수신하기 위해 코드화됩니다.

```
public boolean login()
throws javax.security.auth.login.LoginException {
javax.security.auth.callback.Callback[] callbacks =
new javax.security.auth.callback.Callback[2];
callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
callbacks[1] = new javax.security.auth.callback.PasswordCallback(
"PasswordCallback", false);
try {
callbackHandler.handle(callbacks);
String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
.getName();
char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
.getPassword();
 // Accept everything.
if (true) {
loggedIn = true;
} else
throw new javax.security.auth.login.FailedLoginException("Login failed");

principal= new JAASPrincipal(username);

} catch (java.io.IOException exception) {
throw new javax.security.auth.login.LoginException(exception.toString());
} catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
throw new javax.security.auth.login.LoginException(exception.toString());
}

return loggedIn;
}
```

그림 *30.* 샘플 *JAASLoginModule.Login()* 메소드

관련 정보

AuthCallback MQXR 클래스

문제점 해결: 텔레메트리 서비스가 JAAS 로그인 모듈을 호출하지 않음

V 8.0.0.4 IBM MQ Light 관리

MQ Explorer 를 사용하거나 명령행에서 MQ Light 를 관리할 수 있습니다. 탐색기를 사용하여 채널을 구성하고 IBM MQ에 연결된 MQ Light 클라이언트를 모니터하십시오. TLS 및 JAAS를 사용하여 MQ Light 보안을 구성할 수 있습니다.

시작하기 전에

플랫폼에 AMQP 설치에 대한 정보는 설치 항목 선택을 참조하십시오. V8.0.0.4 수정팩이 아닌 IBM MQ V8.0.0.4 MR(Manufacturing Refresh)을 사용하여 AMQP 서비스 컴포넌트를 설치하십시오. V8.0.0.4 이전 큐 관리자 버전에는 AMQP 컴포넌트를 설치할 수 없습니다.

MQ Explorer를 사용하여 관리

탐색기를 사용하여 AMQP 채널을 구성하고 IBM MQ에 연결된 MQ Light 클라이언트를 모니터하십시오. TLS 및 JAAS를 사용하여 MQ Light 보안을 구성할 수 있습니다.

명령행을 사용하여 관리

MQ Light는 명령행에서 IBM MQ MQSC 명령을 사용하여 관리할 수 있습니다.

V 8.0.0.4 MQ Light 클라이언트에서 사용 중인 IBM MQ 오브젝트 보기

MQ Light 클라이언트에서 사용할 다른 IBM MQ 자원 (예: 연결 및 등록) 을 볼 수 있습니다.

연결

AMQP 서비스가 시작되면 새 Hconn이 작성되어 큐 관리자에 연결됩니다. 이 Hconn 풀은 MQ Light 클라이언트가 메시지를 발행할 때 사용됩니다. **DISPLAY CONN** 명령을 사용하여 Hconns를 볼 수 있습니다. 예를 들면, 다음과 같습니다.

```
DISPLAY CONN(*) TYPE(CONN) WHERE (APPLDESC LK 'WebSphere MQ Advanced Message Queuing Protocol*')
```

이 명령은 또한 클라이언트별 Hconn을 표시합니다. 공백 클라이언트 ID 속성을 갖는 Hconn은 풀에서 사용되는 Hconn입니다.

MQ Light 클라이언트가 AMQP 채널에 연결되면 새 Hconn이 큐 관리자에 연결됩니다. 이 Hconn은 MQ Light 클라이언트가 작성한 구독에 대한 메시지를 비동기로 이용하는 데 사용됩니다. **DISPLAY CONN** 명령을 사용하여 특정 MQ Light 클라이언트가 사용하는 Hconn을 볼 수 있습니다. 예를 들면, 다음과 같습니다.

```
DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_abcd1234')
```

클라이언트가 작성한 구독

MQ Light 클라이언트가 토픽을 구독하면 새 IBM MQ 구독이 작성됩니다. 구독 이름에는 다음 정보가 포함됩니다.

- 클라이언트의 이름. 클라이언트가 공유 구독을 결합한 경우에는 공유 이름이 사용됩니다.
- 클라이언트가 구독한 토픽 패턴
- 접두부. 접두부는 `private`(클라이언트가 비공유 구독을 작성한 경우) 또는 `share`(클라이언트가 공유 구독을 결합한 경우)입니다.

특정 MQ Light 클라이언트가 사용 중인 구독을 보려면 **DISPLAY SUB** 명령을 실행하고 `private` 접두부를 필터링하십시오.

```
DISPLAY SUB(':private:*')
```

여러 클라이언트가 사용 중인 공유 구독을 보려면 **DISPLAY SUB** 명령을 실행하고 share 접두부를 필터링하십시오.

```
DISPLAY SUB(':share:*')
```

공유 구독은 여러 MQ Light 클라이언트가 사용할 수 있으므로 현재 공유 구독의 메시지를 이용하는 클라이언트를 볼 수 있습니다. 이 작업을 수행하려면 현재 구독 큐에서 핸들이 열려 있는 Hconn을 나열하면 됩니다. 현재 공유를 사용하는 클라이언트를 보려면 다음 단계를 완료하십시오.

1. 공유 구독이 목적지로 사용하는 큐 이름을 찾으십시오. 예를 들면, 다음과 같습니다.

```
DISPLAY SUB(':private:recv_e298452:public') DEST
   5 : DISPLAY SUB(':private:recv_e298452:public') DEST
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120514D312020202020202020707E0A565C2D0020)
   SUB(:private:recv_e298452:public)
   DEST(SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
```

2. **DISPLAY CONN** 명령을 실행하여 해당 큐에서 열려 있는 핸들을 찾으십시오.

```
DISPLAY CONN(*) TYPE(HANDLE) WHERE  (OBJNAME
EQ SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
   21 : DISPLAY CONN(*) TYPE(HANDLE) WHERE(OBJNAME EQ
SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)

AMQ8276: Display Connection details.
   CONN(707E0A56642B0020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(HANDLE)

   OBJNAME(SYSTEM.BASE.TOPIC)      OBJTYPE(TOPIC)

   OBJNAME(SYSTEM.MANAGED.DURABLE.560A7E7020002961)
   OBJTYPE(QUEUE)
```

3. 각 핸들마다 핸들이 열려 있는 MQ Light 클라이언트 ID를 확인하십시오.

```
DISPLAY CONN(707E0A56642B0020) CLIENTID
   23 : DISPLAY CONN(707E0A56642B0020) CLIENTID

AMQ8276: Display Connection details.
   CONN(707E0A56642B0020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(CONN)
   CLIENTID(recv_8f02c9d)
DISPLAY CONN(707E0A565F290020) CLIENTID
   24 : DISPLAY CONN(707E0A565F290020) CLIENTID
AMQ8276: Display Connection details.
   CONN(707E0A565F290020)
   EXTCONN(414D5143514D31202020202020202020)
   TYPE(CONN)
   CLIENTID(recv_86d8888)
```

V 8.0.0.4 MQ Light 클라이언트 식별, 권한 부여, 인증

다른 IBM MQ 클라이언트 애플리케이션과 마찬가지로 AMQP 연결도 여러 가지 방법으로 보안을 설정할 수 있습니다.

다음 보안 기능을 사용하여 IBM MQ에 대한 AMQP 연결에 보안을 설정할 수 있습니다.

- 채널 인증 레코드
- 연결 인증
- 채널 MCA 사용자 구성
- IBM MQ 권한 정의
- TLS 연결성

보안 관점에서 연결 설정은 다음 두 단계로 구성됩니다.

- 연결을 계속해야 하는지 여부 결정
- 애플리케이션이 나중에 권한 검사를 수행하기 위해 가정하는 IBM MQ ID 결정

다음 정보는 AMQP 클라이언트가 연결 작성을 시도할 때 수행하는 단계와 다양한 IBM MQ 구성을 개략적으로 설명합니다. IBM MQ 구성에 따라 설명된 일부 단계를 사용하지 않을 수 있습니다. 예를 들어, 일부 구성은 회사 방화벽 내부 연결을 위해 TLS를 사용하지 않고 일부 구성은 TLS를 사용하지만 인증을 위해 클라이언트 인증서를 사용하지 않습니다. 사용자 정의 JAAS 모듈을 사용하지 않는 환경도 많습니다.

연결 설정

다음 단계는 AMQP 클라이언트가 연결을 설정할 때 그 결과를 설명합니다. 이 단계는 연결이 계속되는지 여부와 애플리케이션이 권한 검사를 위해 가정하는 IBM MQ ID를 결정합니다.

1. 클라이언트가 IBM MQ에 대한 TLS 연결을 열고 인증서를 제공하는 경우, 큐 관리자는 클라이언트 인증서의 유효성을 검증하려고 합니다.
2. 클라이언트가 사용자 이름 및 비밀번호 신임 정보를 제공하면 큐 관리자가 AMQP SASL 프레임을 수신하고 MQ CONNAUTH 구성을 확인합니다.
3. MQ 채널 인증 규칙을 검사합니다(예를 들어, IP 주소와 TLS 인증 DN이 올바른지 여부).
4. 채널 인증 규칙이 다르게 판별하지 않는 한 채널 MCAUSER를 확인합니다.
5. JAAS 모듈이 구성된 경우 모듈이 호출됩니다.
6. 결과 MQ 사용자 ID에 MQ CONNECT 권한 검사가 적용되지 않습니다.
7. 가정된 IBM MQ ID로 연결이 설정됩니다.

메시지 발행

다음 단계는 AMQP 클라이언트가 메시지를 발행할 때 그 결과를 설명합니다. 이 단계는 연결이 계속되는지 여부와 애플리케이션이 권한 검사를 위해 가정하는 IBM MQ ID를 결정합니다.

1. AMQP 링크 첨부 프레임이 큐 관리자에 도착합니다. 연결 중에 설정된 MQ 사용자 ID에 대해 지정된 토픽 문자열에 대한 IBM MQ 발행 권한을 검사합니다.
2. 지정된 토픽 문자열에 메시지가 발행됩니다.

토픽 패턴 구독

다음 단계는 AMQP 클라이언트가 토픽 패턴을 구도할 때 그 결과를 설명합니다. 이 단계는 연결이 계속되는지 여부와 애플리케이션이 권한 검사를 위해 가정하는 IBM MQ ID를 결정합니다.

1. AMQP 링크 첨부 프레임이 큐 관리자에 도착합니다. 연결 중에 설정된 MQ 사용자 ID에 대해 지정된 토픽 패턴에 대한 IBM MQ 구독 권한을 검사합니다.
2. 구독이 작성됩니다.

V 8.0.0.4 MQ Light 클라이언트 ID 및 권한

MQ Light 클라이언트 ID, MQ Light 사용자 이름, 또는 채널이나 채널 인증 규칙에 정의된 공용 클라이언트 ID를 사용하여 IBM MQ 오브젝트 액세스 권한을 부여할 수 있습니다.

관리자는 AMQP 채널을 정의하거나 수정할 때 큐 관리자 CONNAUTH 설정을 구성하거나 채널 인증 규칙을 정의하여 선택합니다. ID는 IBM MQ 토픽에 액세스 권한을 부여하기 위해 사용됩니다. 선택 시 다음 항목을 기반으로 합니다.

1. 채널 USECLNTID 속성
2. 큐 관리자 CONNAUTH 규칙의 ADOPTCTX 속성
3. 채널에 정의된 MCAUSER 속성
4. 일치 채널 인증 규칙의 USERSRC 속성

문제점 예방: 이 프로세스에서 선택한 ID는 예를 들어, DISPLAY CHSTATUS (AMQP) 명령에 의해 클라이언트의 MCAUSER로 참조됩니다. 이 ID는 선택 (2)에서 참조되는 채널의 MCAUSER와 반드시 동일한 ID는 아닙니다.

IBM MQ **setmqaut** 명령을 사용하여 AMQP 채널과 연관된 ID가 사용할 수 있도록 권한이 부여된 오브젝트와 조치를 선택할 수 있습니다. 예를 들어, 다음 명령은 큐 관리자 QM1의 관리자가 제공하는 채널 ID AMQPClient에 권한을 부여합니다.

```
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPClient -all +pub +sub
```

및

```
setmqaut -m QM1 -t qmgr -p AMQPClient -all +connect
```

V 8.0.0.4 비밀번호를 사용하여 MQ Light 클라이언트 인증

클라이언트 비밀번호를 사용하여 MQ Light 사용자 이름을 인증합니다. 클라이언트가 토픽을 발행하고 구독하도록 권한 부여된 ID와 다른 ID를 사용하여 클라이언트를 인증할 수 있습니다.

AMQP 서비스는 MQ CONNAUTH 또는 JAAS를 사용하여 클라이언트 사용자 이름을 인증할 수 있습니다. 이들 중 하나가 구성된 경우, 클라이언트가 제공하는 비밀번호를 MQ CONNAUTH 구성 또는 JAAS 모듈로 확인합니다.

다음 프로시저는 로컬 OS 사용자 및 비밀번호에 대해 개별 사용자를 인증하고 인증에 성공하는 경우 공용 ID AMQPUser를 채택하는 단계 예를 개략적으로 보여줍니다.

1. IBM MQ 관리자는 IBM MQ Explorer를 사용하여 AMQP 채널 MCAUSER ID를 임의 이름(예: AMQPUser)으로 설정합니다.
2. IBM MQ 관리자는 모든 토픽을 발행 및 구독할 수 있는 권한을 AMQPUser에 부여합니다.

   ```
   setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPUser -all +pub +sub +connect
   ```

3. IBM MQ 관리자는 클라이언트가 제공하는 사용자 이름과 비밀번호를 확인하는 IDPWOS CONNAUTH 규칙을 구성합니다. CONNAUTH 규칙은 CHCKCLNT(REQUIRED) 및 ADOPTCTX(NO)를 설정해야 합니다.

참고: 큐 관리자에 대한 연결을 보다 강력하게 제어하려면 채널 인증 규칙을 사용하고 MCAUSER 채널 속성을 권한이 없는 사용자로 설정하는 것이 좋습니다.

V 8.0.0.4 채널에서 발행물 개인정보 보호

AMQP 채널 전체에 걸쳐 각각 방향으로 보내진 AMQP 발행물의 개인정보 보호정책은 연결을 통해 전송을 암호화하기 위해 TLS을 사용하여 보호됩니다.

AMQP 채널에 연결되는 AMQP 클라이언트는 대칭 키 암호화를 사용하여 채널로 전송된 발행물의 개인정보 보호를 위해 TLS를 사용합니다. 엔드 포인트가 인증되지 않으므로 채널 암호화만 신뢰할 수는 없습니다. 개인정보 보호와 서버 또는 상호 인증을 결합하십시오.

TLS를 사용하는 대신 IPsec와 같은 일부 유형의 가상 사설 네트워크(VPN)는 TCP/IP 연결의 엔드 포인트를 인증합니다. VPN은 네트워크를 통해 플로우하는 각 IP 패킷을 암호화합니다. 이러한 VPN 연결이 설정되면 신뢰할 수 있는 네트워크가 설정된 것입니다. VPN 네트워크에서 TCP/IP를 사용하여 AMQP 클라이언트를 AMQP 채널에 연결할 수 있습니다.

서버를 인증하지 않고 TLS 연결을 암호화하면 연결이 중간자 공격(man-in-the-middle attack)에 노출됩니다. 교환하는 정보가 도청에 대해 보호되는 경우에도 누구와 교환하는지 알지 못합니다. 네트워크를 제어하지 않는 경우, IP 전송을 인터셉트하고 엔드 포인트로 위장하는 사람에게 노출됩니다.

서버를 인증하지 않고 익명 TLS을 지원하는 Diffie-Hellman 키 교환 CipherSpec을 사용하여 암호화된 TLS로 연결할 수 있습니다. 클라이언트와 서버 사이에 공유되고 TLS 전송을 암호화하는 데 사용되는 마스터 보안은 개인적으로 서명된 서버 인증서를 교환하지 않고 설정됩니다.

익명 연결은 안전하지 않으므로 대부분의 TLS 구현은 기본적으로 익명 CipherSpec을 사용하지 않습니다. AMQP 채널에서 TLS 연결에 대한 요청이 승인되는 경우, 채널에는 암호구로 보호되는 키 저장소가 있어야 합니

다. 기본적으로 TLS 구현은 익명 CipherSpec을 사용하지 않으므로 키 저장소에는 클라이언트가 인증할 수 있는 개인적으로 서명된 인증서가 포함되어 있어야 합니다.

익명 CipherSpec을 사용하는 경우 서버 키 저장소가 있어야 하지만 개인적으로 서명된 인증서를 포함할 필요는 없습니다.

암호화된 연결을 설정하는 다른 방법은 클라이언트에서 신뢰 제공자를 사용자 고유의 구현으로 바꾸는 것입니다. 사용자의 신뢰 제공자는 서버 인증서를 인증하지 않지만 연결이 암호화됩니다.

V 8.0.0.4 TLS로 MQ Light 클라이언트 구성

TLS를 사용하여 네트워크에서 이동하는 데이터를 보호하고 클라이언트가 연결되는 큐 관리자의 ID를 인증하도록 MQ Light 클라이언트를 구성할 수 있습니다.

MQ Light 클라이언트에서 AMQP 채널로의 연결에 TLS를 사용하려면 TLS에 맞게 큐 관리자가 구성되었는지 확인해야 합니다. 큐 관리자에서 SSL 구성은 큐 관리자가 TLS 인증서를 읽는 키 저장소를 구성하는 방법을 설명합니다.

큐 관리자가 키 저장소와 함께 구성된 경우에는 클라이언트가 연결될 AMQP 채널에서 TLS 속성을 구성해야 합니다. AMQP 채널은 TLS 구성과 관련된 다음 네 가지 속성을 갖습니다.

SSLCAUTH
SSLCAUTH 속성은 큐 관리자에서 MQ Light 클라이언트가 ID를 확인하기 위해 클라이언트 인증서를 제시해야 하는지 여부를 지정하는 데 사용됩니다.

SSLCIPH
SSLCIPH 속성은 채널이 TLS 플로우에서 데이터를 인코드하기 위해 사용해야 하는 암호를 지정합니다.

SSLPEER
SSLPEER 속성은 연결이 허용되려면 클라이언트 인증서와 일치해야 하는 식별 이름(DN)을 지정하는 데 사용됩니다.

CERTLABL
CERTLABL은 큐 관리자가 클라이언트에게 제공해야 하는 인증서를 지정합니다. 큐 관리자의 키 저장소에는 여러 인증서가 포함될 수 있습니다. 이 속성을 사용하면 이 채널에 대한 연결에 사용할 인증서를 지정할 수 있습니다. CERTLABL이 지정되지 않은 경우, 큐 관리자 CERTLABL 속성에 해당하는 레이블이 있는 큐 관리자 키 저장소의 인증서가 사용됩니다.

TLS 속성으로 AMQP 채널을 구성한 경우 다음 명령을 사용하여 AMQP 서비스를 재시작해야 합니다.

```
STOP SERVICE(SYSTEM.AMQP.SERVICE) START SERVICE(SYSTEM.AMQP.SERVICE)
```

MQ Light 클라이언트가 TLS로 보호되는 AMQP 채널에 연결되면 큐 관리자가 제공한 인증서의 ID를 클라이언트가 확인합니다. 이를 수행하려면 큐 관리자의 인증서가 있는 신뢰 저장소로 MQ Light 클라이언트를 구성해야 합니다. 이 단계는 사용하는 MQ Light 클라이언트에 따라 다릅니다.

- Node JS API용 MQ Light 클라이언트 문서는 https://www.npmjs.com/package/mqlight의 내용을 참조하십시오.
- Java API용 MQ Light 클라이언트 문서는 https://mqlight.github.io/java-mqlight/의 내용을 참조하십시오.
- Ruby용 MQ Light 클라이언트 문서는 https://www.rubydoc.info/github/mqlight/ruby-mqlight/의 내용을 참조하십시오.
- Python용 MQ Light 클라이언트 문서는 https://python-mqlight.readthedocs.org/en/latest/의 내용을 참조하십시오.

V 8.0.0.4 큐 관리자에서 MQ Light 클라이언트 연결 끊기

큐 관리자에서 MQ Light 연결을 끊으려면 PURGE CHANNEL 명령을 실행하거나 MQ Light 클라이언트에 대한 연결을 중지하십시오.

- **PURGE CHANNEL** 명령을 실행하십시오. 예를 들면, 다음과 같습니다.

```
PURGE CHANNEL(MYAMQP) CLIENTID('recv_28dbb7e')
```

- 또는 다음 단계를 완료하여 MQ Light 클라이언트가 클라이언트 연결을 끊기 위해 사용하는 연결을 중지하십시오.

  1. **DISPLAY CONN** 명령을 실행하여 클라이언트가 사용 중인 연결을 찾으십시오. 예를 들면, 다음과 같습니다.

     ```
     DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_28dbb7e')
     ```

     명령 출력은 다음과 같습니다.

     ```
     DISPLAY CONN(*)  TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
       40 : DISPLAY CONN(*) TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
     AMQ8276: Display  Connection details.
       CONN(707E0A565F2D0020)
       EXTCONN(414D5143514D31202020202020202020)
       TYPE(CONN)
       CLIENTID(recv_28dbb7e)
     ```

  2. 연결을 중지하십시오. 예를 들면, 다음과 같습니다.

     ```
     STOP CONN(707E0A565F2D0020)
     ```

멀티캐스트 관리

이 정보를 사용하여 멀티캐스트 메시지의 크기를 줄이고 데이터 변환을 사용 가능하도록 설정하는 것과 같은 IBM MQ 멀티캐스트 관리 태스크에 대해 학습하십시오.

멀티캐스트 시작하기

이 정보를 사용하여 IBM MQ 멀티캐스트 토픽 및 통신 정보 오브젝트를 시작하십시오.

이 태스크 정보

IBM MQ 멀티캐스트 메시징은 토픽을 그룹 주소로 맵핑하여 메시지를 전달하도록 네트워크를 사용합니다. 다음 태스크를 수행하면 멀티캐스트 메시징을 위한 필수 IP 주소 및 포트가 올바르게 구성되는지의 여부를 빠르게 테스트할 수 있습니다.

멀티캐스트에 대한 COMMINFO 오브젝트 작성

통신 정보(COMMINFO) 오브젝트에는 멀티캐스트 전송과 연관되는 속성이 포함됩니다. COMMINFO 오브젝트 매개변수에 대한 자세한 정보는 DEFINE COMMINFO를 참조하십시오.

다음 명령행 예를 사용하여 멀티캐스트에 대한 COMMINFO 오브젝트를 정의하십시오.

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

여기서 *MC1*은 COMMINFO 오브젝트의 이름이고, *group address*는 그룹 멀티캐스트 IP 주소 또는 DNS 이름이며, *port number*는 전송할 포트(기본값은 1414임)입니다.

*MC1*이라는 새 COMMINFO 오브젝트가 작성됩니다. 이 이름은 다음 예에서 TOPIC 오브젝트를 정의할 때 지정해야 하는 이름입니다.

멀티캐스트에 대한 TOPIC 오브젝트 작성

토픽은 발행/구독 메시지로 발행된 정보의 주제이며, 이러한 토픽은 TOPIC 오브젝트를 작성하여 정의합니다. TOPIC 오브젝트에는 멀티캐스트와 함께 사용될 수 있을지 여부를 정의하는 두 개의 매개변수가 있습니다. 이러한 매개변수는 **COMMINFO** 및 **MCAST**입니다.

- **COMMINFO** 이 매개변수는 멀티캐스트 통신 정보 오브젝트의 이름을 지정합니다. COMMINFO 오브젝트 매개변수에 대한 자세한 정보는 DEFINE COMMINFO를 참조하십시오.
- **MCAST** 이 매개변수는 토픽 트리의 이 지점에서 멀티캐스트를 허용할 수 있는지의 여부를 지정합니다.

다음 명령행 예를 사용하여 멀티캐스트에 대해 TOPIC 오브젝트를 정의하십시오.

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

*ALLSPORTS*라는 새 TOPIC 오브젝트가 작성됩니다. 이 오브젝트에는 토픽 문자열 *Sports*가 있고, 관련된 통신 정보 오브젝트는 *MC1*이라고 하며(이전 예에서 COMMINFO 오브젝트를 정의할 때 지정된 이름임) 멀티캐스트가 사용으로 설정됩니다.

멀티캐스트 발행/구독 테스트

TOPIC 및 COMMINFO 오브젝트가 작성되면 amqspubc 샘플 및 amqssubc 샘플을 사용하여 이들 오브젝트를 테스트할 수 있습니다. 이러한 샘플에 대한 자세한 정보는 발행/구독 샘플 프로그램을 참조하십시오.

1. 두 개의 명령행 창을 여십시오. 첫 번째 명령행은 amqspubc 발행 샘플에 대한 것이고 두 번째 명령행은 amqssubc 구독 샘플에 대한 명령행입니다.
2. 명령행 1에서 다음 명령을 입력하십시오.

   ```
   amqspubc Sports QM1
   ```

   여기서 *Sports*는 이전 예에서 정의된 TOPIC 오브젝트의 토픽 문자열이고 *QM1*은 큐 관리자의 이름입니다.
3. 명령행 2에 다음 명령을 입력하십시오.

   ```
   amqssubc Sports QM1
   ```

   여기서 *Sports* 및 *QM1*은 157 페이지의 『2』 단계에서 사용되는 것과 동일합니다.
4. 명령행 1에서 Hello world 를 입력하십시오. COMMINFO 오브젝트에 지정된 포트 및 IP 주소가 올바르게 구성된 경우, 지정된 주소의 발행물에 대해 포트를 인식하는 amqssubc 샘플은 명령행 2에서 Hello world 를 출력합니다.

IBM MQ 멀티캐스트 토픽 토폴로지

이 예를 사용하여 IBM MQ 멀티캐스트 토픽 토폴로지를 이해하십시오.

IBM MQ 멀티캐스트 지원에서는 전체 계층 내의 각 서브트리에 자체 멀티캐스트 그룹 및 데이터 스트림이 있어야 합니다.

클래스풀 네트워크 IP 주소 지정 설계에서는 멀티캐스트 주소를 위한 주소 공간을 지정합니다. IP 주소의 전체 멀티캐스트 범위는 224.0.0.0 - 239.255.255.255지만, 이러한 주소 중 일부는 예약된 주소입니다. 예약된 주소 목록은 시스템 관리자에게 문의하거나 https://www.iana.org/assignments/multicast-addresses에서 자세한 정보를 참조하십시오. 239.0.0.0 - 239.255.255.255범위에서 로컬로 범위가 지정된 멀티캐스트 주소를 사용하는 것이 좋습니다.

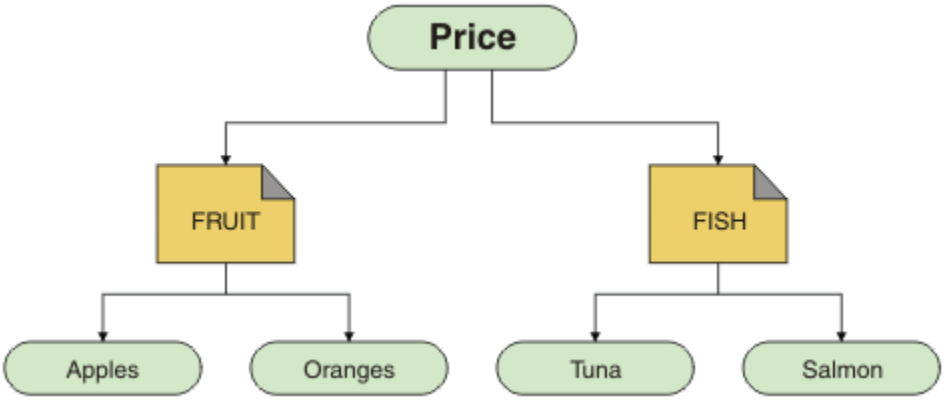
다음 다이어그램에는 사용 가능한 두 개의 멀티캐스트 데이터 스트림이 있습니다.

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

여기서 *239.XXX.XXX.XXX* 및 *239.YYY.YYY.YYY*는 올바른 멀티캐스트 주소입니다.

이러한 토픽 정의를 사용하여 다음 다이어그램에 표시된 대로 토픽 트리를 작성할 수 있습니다.

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)

DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



각 멀티캐스트 통신 정보(COMMINFO) 오브젝트는 해당 그룹 주소가 다르므로 다른 데이터 스트림을 표시합니다. 이 예에서 FRUIT 토픽은 COMMINFO 오브젝트 MC1을 사용하기 위해 정의되고, FISH 토픽은 COMMINFO 오브젝트 MC2를 사용하기 위해 정의되며, Price 노드에는 멀티캐스트 정의가 없습니다.

IBM MQ 멀티캐스트는 토픽 문자열에 대해 255자 제한을 가집니다. 이 제한사항은 트리 내에 있는 노드 및 리프 노드의 이름에 주의해야 함을 의미합니다. 노드 및 리프 노드의 이름이 지나치게 길면 토픽 문자열이 255자를 초과하여 2425(0979)(RC2425): MQRC_TOPIC_STRING_ERROR 이유 코드를 리턴하기 때문입니다. 토픽 문자열이 길면 성능에 좋지 않은 영향을 주므로 가능한 한 토픽 문자열을 짧게 작성하는 것이 좋습니다.

멀티캐스트 메시지의 크기 조절

IBM MQ 메시지 유형에 대해 학습하기 위해 이 정보를 사용하고 IBM MQ 메시지의 크기를 줄이십시오.

IBM MQ 메시지에는 메시지 디스크립터에 포함되는 것과 연관되는 수많은 속성을 가지고 있습니다. 작은 메시지의 경우, 이러한 속성은 대부분의 데이터 트래픽을 나타내고, 전송 속도에 대한 중요한 악영향을 가질 수 있습니다. IBM MQ 멀티캐스트는 사용자가 있다해도 이러한 속성 중에 어느 것이 메시지와 함께 전송되는지 구성할 수 있게 합니다.

토픽 문자열 이외의 메시지 속성 존재는 속성이 보내져야 하는지 여부에 대해 COMMINFO 오브젝트가 언급하는지 여부에 따라 다릅니다. 속성이 전송되지 않으면, 수신 애플리케이션은 기본값을 적용합니다. 기본 MQMD값은 반드시 MQMD_DEFAULT값과 동일하지 않으며 159 페이지의 표 9에 설명되어 있습니다.

COMMINFO 오브젝트에는 메시지로 플로우하는 MQMD 필드 및 사용자 특성 수를 제어하는 MCPROP 속성이 포함됩니다. 적절한 레벨에 이 속성의 값을 설정하여, IBM MQ 멀티캐스트 메시지의 크기를 제어할 수 있습니다.

MCPROP
멀티캐스트 특성은 메시지와 함께 플로우되는 MQMD 특성 및 사용자 특성 수를 제어합니다.

모두
모든 사용자 특성 및 MQMD의 모든 필드는 전송됩니다.

REPLY
메시지에 대한 응답을 처리하는 MQMD 필드 및 사용자 특성만 전송됩니다. 이러한 특성은 다음과 같습니다.

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

user
사용자 특성만 전송됩니다.

NONE
사용자 특성 또는 MQMD 필드는 전송되지 않습니다.

COMPAT
이 값은 메시지의 전송이 RMM에 호환 모드에서 행해지게 하며, 현재 XMS 애플리케이션과 IBM Integration Bus RMM 애플리케이션으로 작동 사이 일부를 허용합니다.

멀티캐스트 메시지 속성

메시지 속성은 MQMD, MQRFH2의 필드 및 메시지 속성과 같은 다양한 장소에서 발생할 수 있습니다.

다음 테이블은 MCPROP 값에 메시지를 보내고 속성이 전송되지 않을 때 사용되는 기본값을 표시합니다.

표 *9.* 메시징 속성 및 멀티캐스트와의 관련 방법

속성	**멀티캐스트를 사용할 때 조치**	**전송되지 않는 경우 기본값**
TopicString	항상 포함됨	적용할 수 없음
MQMQ StrucId	전송되지 않음	적용할 수 없음
MQMD 버전	전송되지 않음	적용할 수 없음
보고서	기본값이 아닌 경우 포함됨	0
MsgType	기본값이 아닌 경우 포함됨	MQMT_DATAGRAM
만기	기본값이 아닌 경우 포함됨	0
Feedback	기본값이 아닌 경우 포함됨	0
Encoding	기본값이 아닌 경우 포함됨	MQENC_NORMAL(equiv)
CodedCharSetId	기본값이 아닌 경우 포함됨	1208
형식	기본값이 아닌 경우 포함됨	MQRFH2
Priority	기본값이 아닌 경우 포함됨	4
지속	기본값이 아닌 경우 포함됨	MQPER_NOT_PERSISTENT
MsgId	기본값이 아닌 경우 포함됨	Null
CorrelId	기본값이 아닌 경우 포함됨	Null
BackoutCount	기본값이 아닌 경우 포함됨	0
ReplyToQ	기본값이 아닌 경우 포함됨	Blank
큐 관리자에 응답	기본값이 아닌 경우 포함됨	Blank
UserIdentifier	기본값이 아닌 경우 포함됨	Blank
AccountingToken	기본값이 아닌 경우 포함됨	Null
PutAppIType	기본값이 아닌 경우 포함됨	MQAT_JAVA
PutAppIName	기본값이 아닌 경우 포함됨	Blank
PutDate	기본값이 아닌 경우 포함됨	Blank
PutTime	기본값이 아닌 경우 포함됨	Blank
ApplOriginData	기본값이 아닌 경우 포함됨	Blank
GroupID	제외됨	적용할 수 없음
MsgSeqNumber	제외됨	적용할 수 없음
오프셋	제외됨	적용할 수 없음
MsgFlags	제외됨	적용할 수 없음
OriginalLength	제외됨	적용할 수 없음

표 9. 메시징 속성 및 멀티캐스트와의 관련 방법 (계속)		
속성	**멀티캐스트를 사용할 때 조치**	**전송되지 않는 경우 기본값**
UserProperties	포함됨	적용할 수 없음

관련 참조
ALTER COMMINFO
DEFINE COMMINFO

멀티캐스트 메시징에 대한 데이터 변환 사용 가능

이 정보를 사용하여 IBM MQ Multicast 메시징에서 데이터 변환이 작동하는 방식을 이해하십시오.

IBM MQ Multicast는 연결되지 않은 공유 프로토콜이므로, 각 클라이언트가 특정하게 데이터 변환을 요청할 수 없습니다. 동일한 멀티캐스트 스트림을 구독하는 모든 클라이언트가 동일한 2진 데이터를 수신합니다. 따라서 IBM MQ 데이터 변환이 필요한 경우 각 클라이언트에서 로컬로 변환을 수행합니다.

혼합 플랫폼 설치에서 전송 애플리케이션의 고유 형식이 아닌 형식에서 대부분의 클라이언트에 데이터가 필요할 수도 있습니다. 이 상황에서 멀티캐스트 COMMINFO 오브젝트의 **CCSID** 및 **ENCODING** 값은 효율성을 위해 메시지 전송의 인코딩을 정의하기 위해 사용될 수 있습니다.

IBM MQ 멀티캐스트는 다음 형식으로 빌드된 것에 대한 메시지 페이로드의 데이터 변환을 지원합니다.

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

이러한 형식뿐만 아니라 자체 형식을 정의하고 MQDXP - 데이터-변환 엑시트 매개변수 데이터 변환 엑시트를 사용할 수도 있습니다.

데이터 변환 프로그래밍에 대한 정보는 멀티캐스트 메시징을 위한 MQI의 데이터 변환을 참조하십시오.

데이터 변환에 대한 자세한 정보는 데이터 변환을 참조하십시오.

데이터 변환 엑시트와 ClientExitPath에 대한 자세한 정보는 클라이언트 구성 파일의 ClientExitPath 스탠자를 참조하십시오.

멀티캐스트 애플리케이션 모니터링

이 정보를 사용하여 IBM MQ 멀티캐스트 관리 및 모니터링에 대해 학습하십시오.

멀티캐스트 트래픽에 대한 현재 발행자 및 구독자의 상태(예: 송신되고 수신되는 메시지 수 또는 유실된 메시지 수)는 클라이언트에서 서버로 주기적으로 전송됩니다. 상태가 수신될 때, COMMINFO 오브젝트의 COMMEV 속성은 큐 메시지가 SYSTEM.ADMIN.PUBSUB.EVENT에 이벤트 메시지를 배치할지 여부를 지정합니다. 이벤트 메시지에는 수신된 상태 정보가 포함됩니다. 이 정보는 문제점의 소스를 찾는 데 있어서 매우 귀중한 진단 보조 프로그램입니다.

MQSC 명령 **DISPLAY CONN**을 사용하면 큐 관리자에 연결된 애플리케이션에 대한 연결 정보를 표시할 수 있습니다. **DISPLAY CONN** 명령에 대한 자세한 정보는 DISPLAY CONN을 참조하십시오.

MQSC 명령 **DISPLAY TPSTATUS**를 사용하면 발행자 및 구독자의 상태를 표시할 수 있습니다. **DISPLAY TPSTATUS** 명령에 대한 자세한 정보는 DISPLAY TPSTATUS를 참조하십시오.

COMMEV 및 멀티캐스트 메시지 신뢰도 표시기

COMMINFO 오브젝트의 **COMMEV** 속성과 함께 사용되는 신뢰도 표시기는 IBM MQ 멀티캐스트 발행자 및 구독자의 모니터링에 핵심 요소입니다. 신뢰도 표시기(발행 또는 구독 상태 명령에서 리턴되는 **MSGREL**필드)는 Sometimes 메시지가 전송 오류로 인해 재전송되어야 하는 오류가 없는 전송 백분율을 설명하는 IBM MQ 표시

기입니다. 이는 **MSGREL**의 값에 반영됩니다. 전송 오류의 잠재적 원인에는 느린 구독자, 사용량이 많은 네트워크 및 네트워크 가동 중단이 포함됩니다. **COMMEV**는 COMMINFO 오브젝트를 사용하여 작성되고 세 개의 가능한 값 중 하나로 설정되는 멀티캐스트 핸들을 위해 이벤트 메시지가 생성되는지 여부를 제어합니다.

DISABLED
이벤트 메시지가 작성되지 않습니다.

ENABLED
이벤트 메시지는 COMMINFO **MONINT** 매개변수로 정의된 빈도로 항상 작성됩니다.

EXCEPTION
메시지 신뢰성이 신뢰성 임계값보다 낮을 경우 이벤트 메시지가 기록됩니다. 90% 이하의 메시지 신뢰도 레벨은 네트워크 구성을 가진 문제점일 될 수도 있고 또는 하나 이상의 발행/구독 애플리케이션이 너무 느리게 실행된다는 것을 표시합니다.

- **MSGREL(100,100)**의 값은 단기간 또는 장기간 시간 범위에서 실행되지 않는지 표시합니다.
- **MSGREL(80,60)**의 값은 메시지의 20%가 현재 문제점을 가지고 있지만, 60의 장기간 값에 개선이 있다는 것을 표시합니다.

클라이언트는 큐 관리자에 대한 유니캐스트 연결이 중단될 때 멀티캐스트 트래픽을 계속 전송하고 수신할 수 있습니다. 따라서, 데이터는 시대에 뒤떨어질 수 있습니다.

멀티캐스트 메시지 신뢰성

이 정보를 사용하여 IBM MQ 멀티캐스트 구독 및 메시지 실행 기록을 설정하는 방법에 대해 학습하십시오.

멀티캐스트를 가지는 전송 실패 극복의 핵심 요소는 IBM MQ에 의해 전송된 데이터의 버퍼링(링크의 전송 종료 시 유지될 메시지의 실행 기록)입니다. 이 프로세스는 IBM MQ에서 신뢰성을 제공하므로 넣기 애플리케이션 프로세스에 필요한 메시지의 버퍼링이 아니라는 것을 의미합니다. 다음 정보에 설명된 대로 통신 정보 (COMMINFO) 오브젝트를 통해 이 실행 기록의 크기가 구성됩니다. 더 큰 전송 버퍼는 필요한 경우 재전송될 추가 전송 실행 기록이 있지만, 멀티캐스트의 성질로 인해 100% 보장 전달이 지원될 수 없다는 것을 의미합니다.

IBM MQ 멀티캐스트 메시지 히스토리는 **MSGHIST** 속성에 의해 통신 정보(COMMINFO) 오브젝트에서 제어됩니다.

MSGHIST
이 값은 NACK(부정적 수신확인)의 경우에 재전송을 핸들링하기 위해 시스템에서 보관하는 메시지 실행 기록의 양(KB)입니다.

값이 0이면 신뢰도 레벨이 가장 낮습니다. 기본값은 100KB입니다.

IBM MQ 멀티캐스트 새 등록 히스토리는 **NSUBHIST** 속성에 의해 통신 정보 (COMMINFO) 오브젝트에서 제어됩니다.

NSUBHIST
새 구독자 실행 기록은 발행 스트림을 조인하는 구독자가 현재 사용 가능한 만큼의 데이터를 수신하는지 또는 구독 시 작성된 발행물만 수신하는지를 제어합니다.

NONE
NONE 값을 사용하면 전송자가 구독 시 작성된 발행물만 전송할 수 있습니다. NONE은 기본값입니다.

모두
ALL 값을 지정하면 전송자가 알려진 모든 토픽의 실행 기록을 재전송합니다. 어떤 경우, 이 상환은 보유된 발행물에 유사한 작동을 제공할 수 있습니다.

참고: ALL 값을 사용하면 모든 토픽 실행 기록이 재전송되기 때문에 대용량 토픽 실행 기록이 있는 경우 성능에 부정적인 영향을 미칠 수 있습니다.

관련 정보
DEFINE COMMINFO
ALTER COMMINFO

고급 멀티캐스트 태스크

이 정보를 사용하여 IBM MQ LLM과의 상호 운용성 및 `.ini` 파일 구성과 같은 고급 IBM MQ 멀티캐스트 관리 태스크에 대해 학습하십시오.

멀티캐스트 설치의 보안에 대한 고려사항의 경우, 멀티캐스트 보안을 참조하십시오.

멀티캐스트 및 비멀티캐스트 발행/구독 도메인 사이의 브릿징

이 정보를 사용하여 비멀티캐스트 발행자가 IBM MQ 멀티캐스트 사용 토픽에 발행할 때 발생하는 것에 대해 이해하십시오.

비멀티캐스트 발행자가 **MCAST** 사용 및 **BRIDGE** 사용으로 정의되는 토픽에 발행하는 경우, 큐 관리자는 멀티캐스트를 통해 대기할 수 있는 구독자에게 직접적으로 메시지를 전송합니다. 멀티캐스트 발행자는 멀티캐스트가 사용되지 않는 토픽에 발행할 수 없습니다.

기존 토픽은 토픽 오브젝트의 **MCAST** 및 **COMMINFO** 매개변수를 설정하여 사용 가능하게 되는 멀티캐스트가 될 수 있습니다. 이러한 매개변수에 대한 자세한 정보는 초기 멀티캐스트 개념을 참조하십시오.

COMMINFO 오브젝트 **BRIDGE** 속성은 멀티캐스트를 사용하고 있지 않는 애플리케이션으로부터 발행을 제어합니다. **BRIDGE**이 ENABLED로 설정되고 토픽의 **MCAST** 매개변수도 ENABLED로 설정되면, 멀티캐스트를 사용하고 있지 않는 애플리케이션으로부터의 발행물은 애플리케이션과 브릿지됩니다. **BRIDGE** 매개변수에 대한 자세한 정보는 DEFINE COMMINFO의 내용을 참조하십시오.

멀티캐스트용 .ini 파일 구성

이 정보를 사용하여 `.ini` 파일에서 IBM MQ 멀티캐스트 필드를 이해하십시오.

추가 IBM MQ 멀티캐스트 구성은 `ini` 파일로 작성될 수 있습니다. 사용해야 하는 특정 `ini` 파일은 애플리케이션의 유형에 따라 다릅니다.

- 클라이언트: *MQ_DATA_PATH* `/mqclient.ini` 파일을 구성합니다.
- 큐 관리자: *MQ_DATA_PATH* `/qmgrs/`*QMNAME*`/qm.ini` 파일을 구성합니다.

여기서 *MQ_DATA_PATH*는 IBM MQ 데이터 디렉토리(`/var/mqm/mqclient.ini`)의 위치이고 *QMNAME*은 `.ini` 파일이 적용되는 큐 관리자의 이름입니다.

`.ini` 파일에는 IBM MQ 멀티캐스트의 작동을 미세 조정하는 데 사용되는 필드가 포함되어 있습니다.

```
Multicast:
Protocol       = IP | UDP
IPVersion       = IPv4 | IPv6 | ANY | BOTH
LimitTransRate    = DISABLED | STATIC | DYNAMIC
TransRateLimit    = 100000
SocketTTL       = 1
Batch         = NO
Loop         = 1
Interface       = <IPaddress>
FeedbackMode     = ACK | NACK | WAIT1
HeartbeatTimeout   = 20000
HeartbeatInterval   = 2000
```

프로토콜

UDP
이 모드에서, 패킷은 UDP 프로토콜을 사용하여 보내집니다. 그러나, IP 모드에서 수행한 것처럼 네트워크 요소는 멀티캐스트 분산에서 보조를 제공할 수 없습니다. 패킷 형식은 PGM으로 호환된 채로 남아 있습니다. 이 값은 기본값입니다.

IP
이 모드에서, 전송자는 원시 IP 패킷을 보냅니다. PGM 지원을 가지는 네트워크 요소는 신뢰할 수 있는 멀티캐스트 패킷 분배를 지원합니다. 이 모드는 PGM 표준과 완전히 호환 가능합니다.

IPVersion

IPv4
IPv4 프로토콜만 사용하여 통신하십시오. 이 값은 기본값입니다.

IPv6
IPv6 프로토콜만 사용하여 통신하십시오.

ANY
사용 가능한 프로토콜에 따라 IPv4, IPv6 또는 둘 모두를 사용하여 통신하십시오.

BOTH
IPv4 및IPv6 모두를 사용하는 통신을 지원합니다.

LimitTransRate

DISABLED
전송율 제어가 없습니다. 이 값은 기본값입니다.

정적
정적 전송율 제어를 구현합니다. 전송자는 TransRateLimit 매개변수에서 지정하는 비율을 초과하는 비율에서 전송하지 않습니다.

DYNAMIC
전송자는 수신자로부터 얻는 피드백에 따라 해당 전송 속도를 채택합니다. 이 경우에 전송 속도 한계는 TransRateLimit 매개변수에 의해 지정된 값 보다 더 많을 수 없습니다. 전송자는 최적 전송 속도에 도달하려고 합니다.

TransRateLimit
전송 비율 한계(Kbp).

SocketTTL
SocketTTL의 값은 멀티캐스트 트래픽이 라우터 또는 통과할 수 있는 라우터 수를 통해 전달될 수 있을지 여부를 판별합니다.

배치
메시지가 배치되거나 즉시 보내지는지 여부를 제어합니다. 가능한 값은 두 개입니다.

- *NO* 메시지가 배치되지 않고 바로 보내집니다.
- *YES* 메시지가 배치됩니다.

Loop
멀티캐스트 루트를 사용으로 설정하기 위해 값을 1로 설정하십시오. 멀티캐스트 루프는 보내진 데이터가 호스트로 다시 루프되는지 여부를 정의합니다.

인터페이스
멀티캐스트 트래픽이 플로우하는 인터페이스의 IP 주소. 자세한 정보 및 문제점 해결은 멀티캐스트가 아닌 네트워크에서 멀티캐스트 애플리케이션 테스트 및 멀티캐스트 트래픽에 대해 적절한 네트워크 설정을 참조하십시오.

FeedbackMode

NACK
부정적인 수신확인에 의한 피드백. 이 값은 기본값입니다.

ACK
긍정적인 수신확인에 의한 피드백.

WAIT1
전송자가 임의의 수신자로부터 1 ACK만 대기하는 긍정적인 수신확인에 의한 피드백.

HeartbeatTimeout
하트비트 제한시간(밀리초). 0의 값은 하트비트 제한시간 이벤트가 수신자 또는 토픽의 수신자에 의해 제기되지 않는지 확인합니다. 기본값은 20000입니다.

HeartbeatInterval
하트비트 간격(밀리초). 0의 값은 보내진 하트비트가 없다는 것을 표시합니다. 하트비트 간격은 False 하트비트 제한시간 이벤트를 피하기 위해 **HeartbeatTimeout** 값 보다 훨씬 작을 수 있습니다. 기본값은 2000입니다.

IBM MQ LLM(Low Latency Messaging)을 가지는 멀티캐스트 상호 운용성

이 정보를 사용하여IBM MQ 멀티캐스트 및 IBM MQ LLM(Low Latency Messaging) 사이의 상호 운용성에 대해 이해하십시오.

기본 페이로드 전송은 다른 애플리케이션이 양쪽 지시사항으로 메시지를 교환하기 위해 멀티캐스트를 사용하면서, LLM을 사용하는 애플리케이션을 위해 가능합니다. 멀티캐스트가 LLM 기술을 사용하지만, LLM 제품 자체는 임베드되지 않습니다. 따라서 LLM과 IBM MQ 멀티캐스트 모두 설치할 수 있고, 개별적으로 두 개의 제품을 작동시키고 서비스를 제공할 수 있습니다.

멀티캐스트와 통신하는 LLM 애플리케이션은 메시지 특성을 보내고 수신해야 할 수도 있습니다. IBM MQ 메시지 특성과 MQMD 필드는 다음 표에 나타난 바와 같이 특정 LLM 메시지 특성 코드를 가진 LLM 메시지 특성으로 전송됩니다.

표 *10. IBM MQ LLM* 특성 맵핑에 대한 *IBM MQ* 메시지 특성

IBM MQ 특성	IBM MQ LLM 특성 유형	LLM 특성 종류	LLM 특성 코드
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1007
MQMD.Priority	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1008
MQMD.Persistence	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1009
MQMD.MsgId	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_ByteArray	-1010
MQMD.BackoutCount	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1012
MQMD.ReplyToQ	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

LLM에 대한 자세한 정보는 LLM 제품 문서(IBM MQ LLM(Low Latency Messaging))를 참조하십시오.

HP Integrity NonStop Server 관리

이 정보를 사용하여 HP Integrity NonStop Server의 IBM MQ 클라이언트에 대한 관리 태스크에 대해 학습할 수 있습니다.

사용 가능한 관리 태스크는 다음 두 가지입니다.

1. Pathway에서 수동으로 TMF/게이트웨이를 시작합니다.
2. Pathway에서 TMF/게이트웨이를 중지합니다.

Pathway에서 수동으로 TMF/게이트웨이 시작

Pathway가 첫 번째 등록 요청에서 자동으로 TMF/게이트웨이를 시작하도록 허용하거나 Pathway에서 수동으로 TMF/게이트웨이를 시작할 수 있습니다.

프로시저

Pathway에서 TMF/게이트웨이를 수동으로 시작하려면 다음 PATHCOM 명령을 입력하십시오.

```
START SERVER <server_class_name>
```

TMF/게이트웨이가 인다우트(in-doubt) 트랜잭션의 복구를 완료하기 전에 클라이언트 애플리케이션이 등록 요청을 작성하면 요청이 최대 1초 동안 보류됩니다. 해당 시간 내에 복구가 완료되지 않으면 등록이 거부됩니다. 이렇게 되면 클라이언트는 트랜잭션 MQI 사용에서 MQRC_UOW_ENLISTMENT_ERROR 오류를 수신합니다.

Pathway에서 TMF/게이트웨이 중지

이 태스크는 Pathway에서 TMF/게이트웨이를 중지하는 방법과 TMF/게이트웨이를 중지한 후 다시 시작하는 방법을 설명합니다.

프로시저

1. TMF/게이트웨이에 새 등록 요청을 하는 것을 방지하려면 다음 명령을 입력하십시오.

   ```
   FREEZE SERVER <server_class_name>
   ```

2. TMF/게이트웨이가 모든 인플라이트 조작을 완료하고 종료하도록 트리거하려면 다음 명령을 입력하십시오.

   ```
   STOP SERVER <server_class_name>
   ```

3. TMF/게이트웨이가 첫 번째 등록에서 자동으로 또는 수동으로 다시 시작되도록 하려면 1단계와 2단계를 수행한 후 다음 명령을 입력하십시오.

   ```
   THAW SERVER <server_class_name>
   ```

   애플리케이션이 새 등록을 요청할 수 없고 **THAW** 명령을 실행할 때까지 **START** 명령을 실행할 수 없습니다.

IBM i 관리

IBM i에서 IBM MQ을 관리할 수 있도록 하는 메소드를 소개합니다.

관리 태스크에는 클러스터, 프로세스 및 IBM MQ 오브젝트(큐 관리자, 큐, 이름 목록, 프로세스 정의, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 인증 정보 오브젝트)의 작성, 시작, 변경, 보기, 중지 및 삭제가 포함됩니다.

IBM MQ for IBM i를 관리하기 위한 방법에 대한 세부사항은 다음 링크를 관리하십시오.

- 166 페이지의 『CL 명령을 사용하여 IBM MQ for IBM i 관리』
- 179 페이지의 『IBM MQ for IBM i 관리의 대체 방법』
- 183 페이지의 『작업 관리』

관련 개념

190 페이지의 『가용성, 백업, 복구 및 재시작』
이 정보를 사용하여 IBM MQ for IBM i가 해당 백업 및 복원 전략을 돕기 위한 IBM i 저널링 지원을 사용하는 방법에 대해 이해하십시오.

관련 참조

229 페이지의 『IBM MQ for IBM i 일시정지』
이 절에서는 IBM MQ for IBM i를 일시정지하는 방법(우아한 종료)을 설명합니다.

관련 정보
IBM i에서 구성 정보 변경
IBM MQ for IBM i 큐 관리자 라이브러리 이름 이해
IBM i에서 보안 설정
IBM i의 데드 레터 큐 핸들러
IBM MQ for IBM i 애플리케이션으로 문제점 판별
IBM i의 설치 가능 서비스 및 컴포넌트
IBM i의 시스템 및 기본 오브젝트

CL 명령을 사용하여 IBM MQ for IBM i 관리

이 정보를 사용하여 IBM MQ IBM i 명령을 이해하십시오.

큐 관리자, 큐, 토픽, 채널, 이름 목록, 프로세스 정의 및 인증 정보 오브젝트와 연관되는 것을 포함하는 대부분의 IBM MQ 명령 그룹이 관련 **WRK*** 명령을 사용하여 액세스될 수 있습니다.

세트의 프린시펄 명령은 **WRKMQM**입니다. 예를 들어, 이 명령을 사용하면 상태 정보와 함께 시스템에서 모든 큐 관리자 목록을 표시할 수 있습니다. 그렇지 않으면, 각 입력 항목에 대한 다양한 옵션을 사용하여 모든 큐 관리자 특정 명령을 처리할 수 있습니다.

WRKMQM 명령에서 각 큐 관리자의 특정 영역(예: 채널, 토픽 또는 큐에 대한 작업)을 선택할 수 있고 거기에서 개별 오브젝트를 선택하십시오.

IBM MQ 애플리케이션 정의 레코딩

IBM MQ 애플리케이션을 작성하거나 사용자 정의할 때, 이는 IBM MQ 정의를 작성되도록 하는 데 유용합니다. 이 레코드는 다음에 사용될 수 있습니다.

- 복구 목적
- 유지보수
- IBM MQ 애플리케이션 롤 아웃

두 가지 방법 중 하나로 IBM MQ 애플리케이션 정의를 레코드할 수 있습니다.

1. 서버에 대한 IBM MQ 정의를 생성하기 위한 CL 프로그램 작성.
2. 크로스 플랫폼 IBM MQ 명령 언어를 사용하여 IBM MQ 정의를 생성하기 위해 SRC 멤버로 MQSC 텍스트 파일 작성.

큐 오브젝트를 정의하는 것에 대한 추가 세부사항은 69 페이지의 『스크립트(MQSC) 명령』 및 10 페이지의 『PCF(Programmable Command Format) 사용』의 내용을 참조하십시오.

시작하기 전에 CL 명령을 사용하여 IBM MQ for IBM i 사용

이 정보를 사용하여 IBM MQ 서브시스템을 시작하고 로컬 큐 관리자를 작성하십시오.

시작하기 전에

IBM MQ 서브시스템이 실행 중이고(명령 STRSBS QMQM/QMQM 사용) 서브시스템과 연관되는 작업 큐가 유지되지 않는 다는 것을 확인하십시오. 기본적으로, IBM MQ 서브시스템 및 작업 큐는 모두 라이브러리 QMQM에서 QMQM으로 이름 지정됩니다.

이 태스크 정보

IBM i 명령행을 사용하여 큐 관리자 시작

프로시저

1. IBM i 명령행에서 CRTMQM 명령을 실행하여 로컬 큐 관리자를 작성하십시오.

큐 관리자를 작성할 때, 해당 큐 관리자가 기본 큐 관리자를 작성하는 옵션을 가지고 있습니다. 기본 큐 관리자(유일하게 하나일 수 있음)는 큐 관리자 이름 매개변수(MQMNAME)이 생략되는 경우 CL 명령이 적용되는 큐 관리자입니다.

2. IBM i 명령행에서 STRMQM 명령을 실행하여 로컬 큐 관리자를 시작하십시오.

   큐 관리자 시동에 몇 초 이상 걸리면, IBM MQ는 시동 프로세스를 자세히 설명하여 간헐적으로 상태 메시지를 표시합니다. 이러한 메시지에 대한 자세한 정보는 메시지 및 이유 코드를 참조하십시오.

다음에 수행할 작업

IBM i 명령행에서 ENDMQM 명령을 실행하여 큐 관리자를 중지하고 IBM i 명령행에서 다른 IBM MQ 명령을 실행하여 큐 관리자를 제어할 수 있습니다.

리모트 큐 관리자는 원격으로 시작될 수 없지만 로컬 연산자로서 해당 시스템에서 작성되고 시작되어야 합니다. 이에 대한 예외는 오퍼레이팅 기능(IBM MQ for IBM i 외부)이 그런 조작을 사용 가능하게 하도록 존재하는 위치입니다.

로컬 큐 관리자는 리모트 큐 관리자를 중지할 수 없습니다.

참고: IBM MQ 시스템을 정지하는 것의 일부로서 활성 큐 관리자를 일시정지해야 합니다. 이 프로그램은 229 페이지의 『IBM MQ for IBM i 일시정지』 에서 설명합니다.

IBM MQ for IBM i 오브젝트 작성

이 정보를 사용하여 IBM i에 대한 IBM MQ 오브젝트를 작성하기 위한 방법을 이해하십시오.

시작하기 전에

다음 태스크는 명령행에서 IBM MQ for IBM i를 사용할 수 있는 다양한 방법을 제안합니다.

이 태스크 정보

IBM MQ 오브젝트를 작성하기 위한 두 가지 온라인 방법이 있으며, 다음과 같습니다.

프로시저

1. 예를 들어, 작성 명령을 사용하여 **Create MQM Queue** 명령을 실행하십시오. **CRTMQMQ**
2. MQM 오브젝트에 대한 작업 명령 다음에 F6을 사용하십시오 (예: **Work with MQM Queues** 명령: **WRKMQMQ**).

다음에 수행할 작업

모든 명령의 목록의 경우, IBM MQ for IBM i CL 명령의 내용을 참조하십시오.

참고: 모든 MQM 명령은 메시지 큐 관리자 명령 메뉴에서 제출될 수 있습니다. 이 메뉴를 표시하려면 명령행에 GO CMDMQM을 입력하고 Enter 키를 누르십시오.

명령을 이 메뉴에서 선택할 때 시스템은 자동으로 신속한 패널을 표시합니다. 직접적으로 명령행에 입력한 명령을 위한 신속한 패널을 표시하려면, Enter 키를 누르기 전에 F4를 누르십시오.

CRTMQMQ 명령을 사용하여 로컬 큐 작성

프로시저

1. 명령행에서 CHGMQM을 입력하고 F4 키를 누르십시오.
2. **MQM큐 패널 작성**에서 Queue name 필드에 작성할 큐의 이름을 입력하십시오. 혼합된 사건명을 지정하려면, 이름을 아포스트로피로 묶으십시오.
3. Queue type 필드에 *LCL 을 입력하십시오.

4. 기본 큐 관리자를 사용하지 않으면 큐 관리자 이름을 지정하고 Enter 키를 누르십시오. 값을 새 값으로 겹쳐 쓸 수 있습니다. 추가 필드를 보려면 앞으로 이동하십시오. 클러스터에 사용되는 옵션은 옵션 목록의 끝에 있습니다.
5. 값을 변경했을 때, Enter 키를 눌러 큐를 작성하십시오.

WRKMQMQ 명령을 사용하여 로컬 큐 작성

프로시저

1. 명령행에서 WRKMQMQ를 입력하십시오.
2. 큐 관리자의 이름을 입력하십시오.
3. 프롬프트 패널을 표시하려는 경우, F4를 누르십시오. 프롬프트 패널은 일반 큐 이름 또는 큐 유형을 지정하여 표시되는 큐의 수를 감소시키는 데 유용합니다.
4. Enter 를 누르고 **MQM큐에 대한 작업 패널** 이 표시됩니다. 새 값으로 값을 겹쳐 쓸 수 있습니다. 추가 필드를 보려면 앞으로 이동하십시오. 클러스터에 사용되는 옵션은 옵션 목록의 끝에 있습니다.
5. F6을 눌러 새 큐를 작성하십시오. 이는 **CRTMQMQ** 패널로 안내합니다. 큐 작성 방법에 대한 지시사항은 167 페이지의 『CRTMQMQ 명령을 사용하여 로컬 큐 작성』의 내용을 참조하십시오. 큐를 작성했을 때, **MQM 큐에 대한 작업 패널**이 다시 표시됩니다. F5=Refresh를 누를 때 새 큐가 목록에 추가됩니다.

큐 관리자 속성 대체

이 태스크 정보

CHGMQM 명령에서 지정되는 큐 관리자의 속성을 대체하려면 변경하려는 속성 및 값을 지정하십시오. 예를 들어, 다음 옵션을 사용하여 jupiter.queue.manager의 속성을 변경하십시오.

프로시저

명령행에서 **CHGMQM**을 입력하고 F4 키를 누르십시오.

결과

명령은 사용되는 데드-레터 큐를 변경하고 이벤트 억제를 사용합니다.

로컬 큐에 대한 작업

이 섹션에는 로컬 큐를 관리하기 위해 사용할 수 있는 일부 명령에 대한 예가 포함됩니다. 표시되는 모든 명령은 **WRKMQMQ 명령 패널**에서 옵션을 사용하여 사용 가능합니다.

로컬 큐 정의

애플리케이션의 경우, 로컬 큐 관리자는 애플리케이션이 연결되는 큐 관리자입니다. 로컬 큐 관리자에 의해 관리되는 큐는 해당 큐 관리자에 대해 로컬이라고 합니다.

CRTMQMQ QTYPE *LCL 명령을 사용하여 로컬 큐의 정의를 작성하고 큐라고 하는 데이터 구조도 작성하십시오. 기본 로컬 큐의 특성에서 큐 특성을 수정할 수도 있습니다.

이 예에서 정의하는 큐(orange.local.queue)가 해당 특성으로 지정됩니다.

- 가져오기(GET)에 사용되고, 넣기(PUT)에 사용 불가능하고 FIFO(First-In-First-Out) 기본에서 실행됩니다.
- 이는 원래 큐입니다. 즉, 이는 이니시에이션 큐 또는 전송 큐가 아니고 트리거 메시지를 생성하지 않습니다.
- 최대 큐 용량은 1000개의 메시지입니다. 최대 메시지 길이는 2000바이트입니다.

다음 명령은 기본 큐 관리자에서 이를 수행합니다.

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL)
TEXT('Queue for messages from other systems')
PUTENBL(*NO)
GETENBL(*YES)
```

```
TRGENBL(*NO)
MSGDLYSEQ(*FIFO)
MAXDEPTH(1000)
MAXMSGLEN(2000)
USAGE(*NORMAL)
```

참고:

1. USAGE *NORMAL에서는 이 큐가 전송 큐가 아니라는 것을 표시합니다.
2. 동일한 큐 관리자에서 orange.local.queue 이름을 가지는 로컬 큐가 이미 있는 경우, 이 명령에 실패합니다. 큐의 기존 정의를 덮어쓰려는 경우 REPLACE *YES 속성을 사용하고 170 페이지의 『로컬 큐 속성 변경』의 내용도 참조하십시오.

데드-레터 큐 정의

각 큐 관리자는 올바른 목적지로 전달될 수 없는 메시지가 나중에 검색을 위해 저장될 수 있도록 로컬 큐가 데드-레터 큐로서 사용되도록 해야 합니다. 데드-레터 큐에 대한 큐 관리자를 명확하게 구별해야 합니다. **CRTMQM** 명령에서 데드-레터 큐를 지정하여 이를 수행할 수 있거나 **CHGMQM** 명령을 사용하여 나중에 하나를 지정할 수 있습니다. 사용될 수 있으려면 데드-레터 큐도 정의해야 합니다.

SYSTEM.DEAD.LETTER.QUEUE라는 샘플 데드-레터 큐가 제품과 함께 제공됩니다. 이 큐는 큐 관리자를 작성할 때 자동으로 작성됩니다. 필요한 경우 이 정의를 수정할 수 있습니다. 사용자가 좋아하는 경우 사용할 수 있을지라도 이름을 바꿀 필요는 없습니다.

데드-레터 큐에는 다음을 제외하고 특별한 요구사항이 없습니다.

- 이는 로컬 큐여야 합니다.
- MAXMSGL(최대 메시지 길이) 속성은 큐가 데드-레터 헤더(MQDLH)의 크기와 **함께** 큐 관리자가 핸들링해야 하는 가장 큰 메시지를 수용할 수 있도록 설정해야 합니다.

IBM MQ에서는 데드-레터 큐에서 발견되는 메시지가 처리되거나 제거되는 방법을 지정할 수 있도록 데드-레터 큐 핸들러를 제공합니다. 추가 정보의 경우, IBM MQ for IBM i 데드-레터 큐 핸들러를 참조하십시오.

기본 오브젝트 속성 표시

IBM MQ 오브젝트를 정의할 때, 기본 오브젝트에서 지정하지 않는 속성을 사용합니다. 예를 들어, 로컬 큐를 정의할 때, 큐는 SYSTEM.DEFAULT.LOCAL.QUEUE라는 기본 로컬 큐의 정의에서 생략하는 속성을 상속합니다. 이러한 속성의 개념을 정확하게 보려면 다음 명령을 참조하십시오.

```
DSPMQMQ QNAME(SYSTEM.DEFAULT.LOCAL.QUEUE) MQMNAME(MYQUEUEMANAGER)
```

로컬 큐 정의 복사

CPYMQMQ 명령을 사용하여 큐 정의를 복사할 수 있습니다. 예를 들면, 다음과 같습니다.

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

이 명령은 시스템 기본 로컬 큐의 속성 보다 원래 큐 orange.local.queue와 동일한 속성을 가진 큐를 작성합니다.

CPYMQMQ 명령을 사용하여 큐 정의를 복사할 수도 있지만, 원래의 속성에 대한 하나 이상의 변경사항 대체를 사용할 수도 있습니다. 예를 들면, 다음과 같습니다.

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('third.queue') MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(1024)
```

이 명령은 큐 orange.local.queue의 속성을 큐 third.queue로 복사하지만, 새 큐의 최대 메시지 길이가 2000바이트라기 보다는 1024바이트가 되도록 지정합니다.

참고: CPYMQMQ 명령을 사용할 때, 큐의 메시지가 아니라 큐 속성만 복사합니다.

로컬 큐 속성 변경

REPLACE *YES 속성으로 **CHGMQMQ** 명령 또는 **CPYMQMQ** 명령을 사용하여, 두 가지 방법으로 큐 속성을 변경할 수 있습니다. 168 페이지의 『로컬 큐 정의』 에서 큐 orange.local.queue를 정의합니다. 예를 들면, 10,000 바이트에 이 큐의 최대 메시지 길이를 증가시킬 필요가 있는 경우.

- **CHGMQMQ** 명령 사용:

```
CHGMQMQ QNAME('orange.local.queue') MQMNAME(MYQUEUEMANAGER) MAXMSGLEN(10000)
```

  이 명령은 단일 속성, 즉 최대 메시지 길이의 속성을 변경합니다. 기타 모든 속성이 동일하게 유지됩니다.
- REPLACE *YES 옵션으로 **CRTMQMQ** 명령 사용. 예:

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL) MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(10000) REPLACE(*YES)
```

  이 명령은 최대 메시지 길이가 아닌 기타 모든 속성을 변경시키고, 기본값이 제공됩니다. 이전에는 큐를 넣을(put) 수 없었지만, 이제는 넣을(put) 수 있습니다. 변경하지 않으면 큐 SYSTEM.DEFAULT.LOCAL.QUEUE에 의해 지정된 대로 가능한 넣기가 기본값이 됩니다.

  기존 큐의 최대 메시지 길이를 **줄일 경우** 기존 메시지는 영향을 받지 않습니다. 그러나, 새 메시지는 새 기준을 충족해야 합니다.

로컬 큐 지우기

magenta.queue라는 로컬 큐에서 모든 메시지를 삭제하려면 다음 명령을 사용하십시오.

```
CLRMQMQ QNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

다음의 경우 큐를 지울 수 없습니다.

- 동기점 아래에 큐에 걸린 미확약된 메시지가 있습니다.
- 애플리케이션이 현재 큐를 열어 두었습니다.

로컬 큐 삭제

DLTMQMQ 명령을 사용하여 로컬 큐를 삭제하십시오.

큐에 미확약된 메시지가 있는 경우 또는 사용 중인 경우 큐는 삭제될 수 없습니다.

큰 큐를 사용으로 설정

IBM MQ에서는 2GB보다 큰 큐를 지원합니다. IBM i가 큰 파일을 지원할 수 있게 하는 방법에 대한 정보는 운영 체제 문서를 참조하십시오.

IBM i 제품 문서는 다음에서 찾아보십시오. https://publib.boulder.ibm.com/iseries/

일부 유틸리티는 2GB 보다 큰 파일을 대체할 수 없습니다. 큰 파일 지원을 사용으로 설정하기 전에, 그와 같은 지원에 대한 제한에 대한 정보는 사용자의 운영 체제 문서를 확인하십시오.

알리어스 큐에 대한 작업

이 절에는 알리어스 큐를 관리하기 위해 사용할 수 있는 일부 명령의 예가 포함됩니다. 표시되는 모든 명령은 **WRKMQMQ 명령 패널**에서 옵션을 사용하여 사용 가능합니다.

알리어스 큐(가끔 큐 알리어스로 알려짐)에서는 MQI 호출의 경로를 재지정하는 메소드를 제공합니다. 알리어스 큐는 실제 큐가 아니라 실제 큐로 해석되는 정의입니다. 알리어스 큐 정의에는 대상 큐 이름이 포함됩니다. 이는 TGTQNAME 속성에 의해 지정됩니다.

애플리케이션이 MQI 호출에서 알리어스 큐를 지정할 때, 큐 관리자는 런타임 시 실제 큐 이름을 해석합니다.

예를 들어, 애플리케이션은 my.alias.queue라고 하는 큐에 메시지를 배치하기 위해 개발됩니다. **MQOPEN** 요청을 작성할 때와 간접적으로 이 큐에 메시지를 배치하는 경우 이 큐의 이름을 지정합니다. 애플리케이션은 큐가 알리어스 큐라는 것을 인지하지 않습니다. 이 알리어스를 사용하는 각 MQI 호출의 경우, 큐 관리자는 실제 큐 이름을 해석합니다. 이는 이 큐 관리자에 정의되는 로컬 큐 또는 리모트 큐가 될 수 있습니다.

TGTQNAME 속성의 값을 변경하여, 다른 큐 관리자에서 사용 가능한 다른 큐로 MQI 호출의 경로를 재지정할 수 있습니다. 이는 유지보수, 마이그레이션 및 로드 밸런싱에 유용합니다.

알리어스 큐 정의

다음 명령은 알리어스 큐를 작성합니다.

```
CRTMQMQ QNAME('my.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
MQMNAME(MYQUEUEMANAGER)
```

이 명령은 my.alias.queue를 큐 yellow.queue로 지정하는 MQI 호출의 경로를 재지정합니다. 명령은 대상 큐를 작성하지 않습니다. MQI 호출은 yellow.queue가 런타임 시 존재하지 않는 경우 실패합니다.

알리어스 정의를 변경하는 경우, 다른 큐로 MQI 호출의 경로를 재지정할 수 있습니다. 예를 들면, 다음과 같습니다.

```
CHGMQMQ QNAME('my.alias.queue') TGTQNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

이 명령은 MQI 호출의 경로를 다른 큐 magenta.queue로 재지정합니다.

알리어스 큐를 사용하여 단일 큐(대상 큐)가 다른 애플리케이션에 대해 다른 속성을 갖는 것처럼 보이도록 할 수 있습니다. 각 애플리케이션에 하나씩 두 개의 알리어스를 정의하여 이를 수행합니다. 두 개의 애플리케이션이 있다고 가정하십시오.

- 애플리케이션 ALPHA는 yellow.queue에 메시지를 배치할 수 있지만, 여기에서 메시지를 가져올 수 없습니다.
- 애플리케이션 BETA는 yellow.queue에서 메시지를 가져올 수 있지만, 여기에 메시지를 배치할 수 없습니다.

다음 명령을 사용하여 이를 수행할 수 있습니다.

```
/* This alias is put enabled and get disabled for application ALPHA */

CRTMQMQ QNAME('alphas.alias.queue') QTYPE(*ALS)     TGTQNAME('yellow.queue')
PUTENBL(*YES) GETENBL(*NO) MQMNAME(MYQUEUEMANAGER)

/* This alias is put disabled and get enabled for application BETA */

CRTMQMQ QNAME('betas.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
PUTENBL(*NO) GETENBL(*YES) MQMNAME(MYQUEUEMANAGER)
```

ALPHA는 MQI 호출에서 큐 이름 alphas.alias.queue를 사용합니다. BETA는 큐 이름 betas.alias.queue를 사용합니다. 이는 모두 동일한 큐에 액세스하지만, 다른 방법으로 액세스합니다.

로컬 큐로 사용하는 동일한 방법으로 알리어스 큐를 정의할 때 REPLACE *YES 속성을 사용할 수 있습니다.

알리어스 큐로 기타 명령 사용

적절한 명령을 사용하여 알리어스 큐 속성을 표시하거나 변경할 수 있습니다. 예를 들면, 다음과 같습니다.

```
* Display the alias queue's attributes */

DSPMQMQ QNAME('alphas.alias.queue') MQMNAME(MYQUEUEMANAGER)

/* ALTER the base queue name, to which the alias resolves. */
/* FORCE = Force the change even if the queue is open. */

CHQMQMQ QNAME('alphas.alias.queue') TGTQNAME('orange.local.queue') FORCE(*YES)
MQMNAME(MYQUEUEMANAGER)
```

모델 큐에 대한 작업

이 섹션에는 모델 큐를 관리하기 위해 사용할 수 있는 일부 명령의 예가 포함됩니다. 표시되는 모든 명령은 **WRKMQMQ 명령 패널**에서 옵션을 사용하여 사용 가능합니다.

큐 관리자는 모델 큐로서 정의되는 큐 이름을 지정하여 애플리케이션에서 MQI를 수신하는 경우 동적 큐를 작성합니다. 새 동적 큐의 이름은 큐가 작성될 때 큐 관리자에서 생성됩니다. 모델 큐는 작성되는 동적 큐의 속성을 지정하는 템플리트입니다.

모델 큐에서는 요청될 때 큐를 작성하기 위한 애플리케이션에 편리한 메소드를 제공합니다.

모델 큐 정의

로컬 큐를 정의하는 동일한 방법으로 속성 세트와 함께 모델 큐를 정의합니다. 모델 큐 및 로컬 큐에는 작성되는 동적 큐가 임시 또는 영구적인지 여부를 지정할 수 있는 모델 큐의 속성을 제외하고 동일한 속성 세트가 있습니다. (영구적 큐는 큐 관리자 재시작을 통해 유지보수되고 임시 큐는 그렇지 않습니다). 예를 들면, 다음과 같습니다.

```
CRTMQMQ QNAME('green.model.queue') QTYPE(*MDL) DFNTYPE(*PERMDYN)
```

이 명령은 모델 큐 정의를 작성합니다. DFNTYPE 속성에서 이 템플리트에서 작성된 실제 큐는 영구적 동적 큐입니다. 지정되지 않은 속성은 SYSYTEM.DEFAULT.MODEL.QUEUE 기본 큐에서 자동적으로 복사됩니다.

로컬 큐로 사용하는 동일한 방법으로 모델 큐를 정의할 때 REPLACE *YES 속성을 사용할 수 있습니다.

모델 큐로 기타 명령 사용

적절한 명령을 사용하여 모델 큐의 속성을 표시하거나 대체할 수 있습니다. 예를 들면, 다음과 같습니다.

```
/* Display the model queue's attributes */

DSPMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('green.model.queue')

/* ALTER the model queue to enable puts on any */
/* dynamic queue created from this model. */

CHGMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('blue.model.queue') PUTENBL(*YES)
```

트리거에 대한 작업

이 정보를 사용하여 트리거에 대해 학습하고 정의를 처리하십시오.

IBM MQ에서는 큐에 대한 특정 조건이 충족될 때 자동으로 애플리케이션을 시작하기 위한 기능을 제공합니다. 조건의 한 예는 큐에 대한 메시지 수가 지정된 수에 도달할 때입니다. 이 기능은 트리거라고 하고 트리거 채널에서 상세하게 설명됩니다.

트리거 개념

큐 관리자는 트리거 이벤트를 구성할 때 특정 조건을 정의합니다. 트리거가 큐에 사용되도록 설정되고 트리거 이벤트가 발생하는 경우, 큐 관리자는 트리거 메시지를 이니시에이션 큐라고 하는 큐에 보냅니다. 이니시에이션 큐에 있는 트리거 메시지는 트리거 이벤트가 발생했음을 표시합니다.

큐 관리자에서 생성된 트리거 메시지는 지속적이지 않습니다. 이는 로깅을 줄이고(성능 개선) 재시작 동안 복제를 최소화하는 효과가 있으므로 재시작 시간을 향상시키는 효과가 있습니다.

트리거 모니터의 개념

이니시에이션 큐를 처리하는 프로그램은 트리거 모니터 애플리케이션이라고 하고 해당 기능은 트리거 메시지에 포함된 정보를 기반으로 트리거 메시지를 읽고 적절한 조치를 수행하는 것입니다. 일반적으로 이 조치는 트리거 메시지가 생성되도록 하는 큐를 처리하기 위해 일부 기타 애플리케이션을 시작하는 것입니다. 큐 관리자의 관점

에서 트리거 모니터 애플리케이션에 대해 특별한 것이 없고, 큐(이니시에이션 큐)에서 메시지를 읽는 것은 다른 애플리케이션입니다.

트리거 모니터의 작업 제출 속성 대체

STRMQMTRM 명령으로 제공된 트리거 모니터는 시스템 기본 작업 설명(QDFTJOBD)을 사용하여 각 트리거 메시지에 대한 작업을 제출합니다. 이는 제출된 작업을 항상 QDFTJOBD라고 하고 라이브러리 목록(*SYSVAL)을 포함하여 기본 작업 설명의 속성을 가지고 있다는 점에서 제한사항이 있습니다. IBM MQ에서는 이러한 속성을 대체하기 위한 메소드를 제공합니다. 예를 들어, 다음과 같이 더 의미있는 작업 이름을 가지고 있도록 제출된 작업을 사용자 정의할 수 있습니다.

1. 작업 설명에서 사용자가 원하는 설명을 지정하십시오(예: 로깅 값).
2. 트리거 프로세스에서 사용되는 프로세스 정의의 환경 데이터를 지정하십시오.

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOBD)')
```

   지정된 설명을 사용하여 트리거 모니터가 SBMJOB을 수행합니다.

프로세스 정의의 환경 데이터에서 적절한 키워드 및 값을 지정하여 SBMJOB의 기타 속성을 대체할 수 있습니다. 이 속성이 트리거 모니터에 의해 채워지기 때문에 이에 대한 유일한 예외는 CMD 키워드입니다. 작업 이름과 설명 모두 변경되는 프로세스 정의의 환경 데이터를 지정하기 위한 명령의 예는 다음과 같습니다.

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOB)
JOB(TRIGGER)')
```

트리거에 대한 애플리케이션 큐 정의

애플리케이션 큐는 MQI를 통해 메시징을 위해 애플리케이션에서 사용하는 로컬 큐입니다. 트리거는 애플리케이션 큐에 정의되는 수많은 큐 속성이 필요합니다. 트리거 자체가 TRGENBL 속성에 의해 사용됩니다.

이 예에서 트리거 이벤트는 다음과 같이 로컬 큐(motor.insurance.queue)에 우선순위가 5 이상인 메시지가 100가지가 있을 때 생성되는 것입니다.

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.insurance.queue') QTYPE(*LCL)
PRCNAME('motor.insurance.quote.process') MAXMSGLEN(2000)
DFTMSGPST(*YES) INITQNAME('motor.ins.init.queue')
TRGENBL(*YES) TRGTYPE(*DEPTH) TRGDEPTH(100) TRGMSGPTY(5)
```

여기서 매개변수는 다음과 같습니다.

MQMNAME(MYQUEUEMANAGER)
대기열 관리자의 이름입니다.

QNAME('motor.insurance.queue')
정의되는 애플리케이션 큐의 이름.

PRCNAME('motor.insurance.quote.process')
트리거 모니터 프로그램에 의해 시작되는 애플리케이션의 이름.

MAXMSGLEN(2000)
큐의 최대 메시지 길이.

DFTMSGPST(*YES)
기본적으로 이 큐의 메시지가 지속됩니다.

INITQNAME('motor.ins.init.queue')
큐 관리자가 트리거 메시지를 사용하는 이니시에이션 큐의 이름.

TRGENBL(*YES)
트리거 속성 값.

TRGTYPE(*DEPTH)
트리거 이벤트는 필수 우선순위(**TRGMSGPTY**)의 메시지 수가 **TRGDEPTH**에 지정되는 수에 도달할 때 생성됩니다.

TRGDEPTH(100)
트리거 이벤트를 생성하기 위해 필요한 메시지 수.

TRGMSGPTY(5)
트리거 이벤트를 생성할지 여부를 결정하는 큐 관리자에 의해 계산되는 메시지의 우선순위. 우선순위 5 이상인 메시지 수만 계수됩니다.

이니시에이션 큐 정의

트리거 이벤트가 발생할 때, 큐 관리자는 애플리케이션 큐 정의에 지정되는 이니시에이션 큐에 트리거 메시지를 배치합니다. 이니시에이션 큐에 특별한 설정이 없지만, 자세한 내용을 위해 로컬 큐motor.ins.init.queue의 다음 정의를 사용할 수 있습니다.

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.ins.init.queue') QTYPE(*LCL)
GETENBL(*YES) SHARE(*NO) TRGTYPE(*NONE)
MAXMSGL(2000)
MAXDEPTH(1000)
```

프로세스 정의 작성

CRTMQMPRC 명령을 사용하여 프로세스 정의를 작성하십시오. 프로세스 정의는 큐로부터 메시지를 처리하는 애플리케이션과 애플리케이션 큐를 연관시킵니다. 이는 애플리케이션 큐 motor.insurance.queue에서 PRCNAME 속성을 통해 수행됩니다. 다음 명령은 이 예에서 식별되는 필수 프로세스 (motor.insurance.quote.process)를 작성합니다.

```
CRTMQMPRC MQMNAME(MYQUEUEMANAGER) PRCNAME('motor.insurance.quote.process')
TEXT('Insurance request message processing')
APPTYPE(*OS400) APPID(MQTEST/TESTPROG)
USRDATA('open, close, 235')
```

여기서 매개변수는 다음과 같습니다.

MQMNAME(MYQUEUEMANAGER)
대기열 관리자의 이름입니다.

PRCNAME('motor.insurance.quote.process')
프로세스 정의의 이름.

TEXT('Insurance request message processing')
이 정의가 관련되는 애플리케이션 프로그램에 대한 설명. **DSPMQMPRC** 명령을 사용할 때 이 텍스트가 표시됩니다. 이는 프로세스가 수행하는 것을 식별하는 데 도움이 될 수 있습니다. 문자열에서 공백을 사용하는 경우, 작은따옴표로 문자열을 묶어야 합니다.

APPTYPE(*OS400)
시작할 애플리케이션의 유형.

APPID(MQTEST/TESTPROG)
완전한 파일 이름으로 지정되는 애플리케이션 실행 파일의 이름.

USRDATA('open, close, 235')
애플리케이션이 사용할 수 있는 사용자 정의 데이터.

프로세스 정의 표시

DSPMQMPRC 명령을 사용하여 정의의 결과를 조사하십시오. 예를 들면, 다음과 같습니다.

```
MQMNAME(MYQUEUEMANAGER) DSPMQMPRC('motor.insurance.quote.process')
```

CHGMQMPRC 명령을 사용하여 기존 프로세스 정의를 변경하고 **DLTMQMPRC** 명령을 사용하여 프로세스 정의를 삭제할 수도 있습니다.

두 시스템 간 통신

서로 통신할 수 있도록 CL 명령을 사용하여 다음 예는 IBM MQ for IBM i 시스템을 설정하는 방법을 설명합니다.

시스템은 SYSTEMA 및 SYSTEMB라고 하며 사용되는 통신 프로토콜은 TCP/IP입니다.

다음 프로시저를 수행하십시오.

1. SYSTEMA에서 큐 관리자를 작성하십시오. 이를 QMGRA1이라고 합니다.

```
CRTMQM   MQMNAME(QMGRA1) TEXT('System A - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

2. 이 큐 관리자를 시작하십시오.

```
STRMQM   MQMNAME(QMGRA1)
```

3. SYSTEMB의 큐 관리자에 메시지를 보내야 하는 SYSTEMA에서 IBM MQ 오브젝트를 정의하십시오.

```
/* Transmission queue */
CRTMQMQ  QNAME(XMITQ.TO.QMGRB1) QTYPE(*LCL) +
MQMNAME(QMGRA1) TEXT('Transmission Queue +
to QMGRB1') MAXDEPTH(5000) USAGE(*TMQ)

/* Remote queue that points to a queue called TARGETB        */
/* TARGETB belongs to queue manager QMGRB1 on SYSTEMB        */
CRTMQMQ  QNAME(TARGETB.ON.QMGRB1) QTYPE(*RMT) +
MQMNAME(QMGRA1) TEXT('Remote Q pointing +
at Q TARGETB on QMGRB1 on Remote System +
SYSTEMB') RMTQNAME(TARGETB) +
RMTMQMNAME(QMGRB1) TMQNAME(XMITQ.TO.QMGRB1)

/* TCP/IP sender channel to send messages to the queue manager on SYSTEMB*/
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*SDR) +
MQMNAME(QMGRA1) TRPTYPE(*TCP) +
TEXT('Sender Channel From QMGRA1 on +
SYSTEMA to QMGRB1 on SYSTEMB') +
CONNAME(SYSTEMB) TMQNAME(XMITQ.TO.QMGRB1)
```

4. SYSTEMB에서 큐 관리자를 작성하십시오. 이를 QMGRB1이라고 합니다.

```
CRTMQM   MQMNAME(QMGRB1) TEXT('System B - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

5. SYSTEMB에서 큐 관리자를 시작하십시오.

```
STRMQM   MQMNAME(QMGRB1)
```

6. SYSTEMA의 큐 관리자로부터 메시지를 수신해야 하는 IBM MQ 오브젝트를 정의하십시오.

```
/* Local queue to receive messages on */
CRTMQMQ  QNAME(TARGETB) QTYPE(*LCL) MQMNAME(QMGRB1) +
TEXT('Sample Local Queue for QMGRB1')

/* Receiver channel of the same name as the sender channel on SYSTEMA */
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*RCVR) +
MQMNAME(QMGRB1) TRPTYPE(*TCP) +
TEXT('Receiver Channel from QMGRA1 to +
QMGRB1')
```

7. 마지막으로 채널이 시작될 수 있도록 SYSTEMB에서 TCP/IP 리스너를 시작하십시오. 이 예는 1414의 기본 포트를 사용합니다.

```
STRMQMLSR MQMNAME(QMGRB1)
```

지금 SYSTEMA 및 SYSTEMB 사이에 테스트 메시지를 보낼 준비가 됩니다. 제공된 샘플 중 하나를 사용함으로써 SYSTEMA의 리모트 큐에 일련의 메시지를 두십시오.

STRMQMCHL 명령을 사용하거나 WRKMQMCHL 명령을 사용하고 송신자 채널에 대한 시작 요청(옵션 14)을 입력하여 SYSTEMA에서 채널을 시작하십시오.

채널은 RUNNING 상태로 이동해야 하고 메시지는 SYSTEMB의 큐 TARGETB로 보내집니다.

명령을 실행하여 메시지를 검사하십시오.

```
WRKMQMMSG QNAME(TARGETB) MQMNAME(QMGRB1).
```

샘플 자원 정의

이 샘플은 AMQSAMP4 샘플 IBM i CL 프로그램을 포함합니다.

```
/*********************************************************************/
/*                                                                   */
/* Program name: AMQSAMP4                                            */
/*                                                                   */
/* Description: Sample CL program defining MQM queues                */
/*       to use with the sample programs                             */
/*       Can be run, with changes as needed, after                   */
/*       starting the MQM                                            */
/*                                                                   */
/* <N_OCO_COPYRIGHT>                                                 */
/* Licensed Materials - Property of IBM                              */
/*                                                                   */
/* 63H9336                                                           */
/* (c) Copyright IBM Corp. 1993, 2023. All Rights Reserved.          */
/*                                                                   */
/* US Government Users Restricted Rights - Use, duplication or       */
/* disclosure restricted by GSA ADP Schedule Contract with           */
/* IBM Corp.                                                         */
/* <NOC_COPYRIGHT>                                                   */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Function:                                                         */
/*                                                                   */
/*                                                                   */
/*  AMQSAMP4 is a sample CL program to create or reset the           */
/*  MQI resources to use with the sample programs.                   */
/*                                                                   */
/*  This program, or a similar one, can be run when the MQM          */
/*  is started - it creates the objects if missing, or resets        */
/*  their attributes to the prescribed values.                       */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*  Exceptions signaled: none                                        */
/*  Exceptions monitored: none                                       */
/*                                                                   */
/*  AMQSAMP4 takes a single parameter, the Queue Manager name        */
/*                                                                   */
/*********************************************************************/
QSYS/PGM PARM(&QMGRNAME)

/*********************************************************************/
/*  Queue Manager Name Parameter                                     */
/*********************************************************************/
QSYS/DCL VAR(&QMGRNAME) TYPE(*CHAR)

/*********************************************************************/
/*    EXAMPLES OF DIFFERENT QUEUE TYPES                              */
/*                                                                   */
```

```
/*  Create local, alias and remote queues                          */
/*                                                                 */
/*  Uses system defaults for most attributes                       */
/*                                                                 */
/*****************************************************************/
/*  Create a local queue                                           */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.LOCAL')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Sample local queue') /* description */+
SHARE(*YES)        /* Shareable  */+
DFTMSGPST(*YES) /* Persistent messages OK */

/*  Create an alias queue                                          */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.ALIAS')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*ALS) REPLACE(*YES)         +
+
TEXT('Sample alias queue')         +
DFTMSGPST(*YES) /* Persistent messages OK */+
TGTQNAME('SYSTEM.SAMPLE.LOCAL')

/*  Create a remote queue - in this case, an indirect reference    */
/*   is made to the sample local queue on OTHER queue manager      */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.REMOTE')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*RMT) REPLACE(*YES)         +
+
TEXT('Sample remote queue')/* description */+
DFTMSGPST(*YES) /* Persistent messages OK */+
RMTQNAME('SYSTEM.SAMPLE.LOCAL')       +
RMTMQMNAME(OTHER)  /* Queue is on OTHER  */

/*  Create a transmission queue for messages to queues at OTHER    */
/*  By default, use remote node name                               */
CRTMQMQ   QNAME('OTHER') /* transmission queue name */+
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES) +
TEXT('Transmision queue to OTHER') +
USAGE(*TMQ)  /* transmission queue   */

/*****************************************************************/
/*    SPECIFIC QUEUES AND PROCESS USED BY SAMPLE PROGRAMS          */
/*                                                                 */
/*  Create local queues used by sample programs                    */
/*  Create MQI process associated with sample initiation queue     */
/*                                                                 */
/*****************************************************************/
/*  General reply queue                                            */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.REPLY')        +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('General reply queue')         +
DFTMSGPST(*NO) /* Not Persistent     */

/*  Queue used by AMQSINQ4                                         */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.INQ')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSINQ4')         +
SHARE(*YES)        /* Shareable  */+
DFTMSGPST(*NO) /* Not Persistent     */+
+
TRGENBL(*YES) /* Trigger control on   */+
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.INQPROCESS')     +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Queue used by AMQSSET4                                         */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.SET')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSSET4')         +
SHARE(*YES)        /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent     */ +
+
TRGENBL(*YES) /* Trigger control on   */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
```

```
PRCNAME('SYSTEM.SAMPLE.SETPROCESS')     +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Queue used by AMQSECH4                                          */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.ECHO')         +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)         +
+
TEXT('Queue for AMQSECH4')          +
SHARE(*YES)        /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent     */ +
+
TRGENBL(*YES) /* Trigger control on   */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')    +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*  Initiation Queue used by AMQSTRG4, sample trigger process       */
CRTMQMQ   QNAME('SYSTEM.SAMPLE.TRIGGER') +
MQMNAME(&QMGRNAME)             +
QTYPE(*LCL) REPLACE(*YES)  +
TEXT('Trigger queue for sample programs')

/*  MQI Processes associated with triggered sample programs         */
/*                                                                  */
/*****  Note - there are versions of the triggered samples     ******/
/*****   in different languages - set APPID for these          ******/
/*****   process to the variation you want to trigger          ******/
/*                                                                  */
CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.INQPROCESS')     +
MQMNAME(&QMGRNAME)             +
REPLACE(*YES)               +
+
TEXT('Trigger process for AMQSINQ4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSINQ4')  /* C       +
/*    APPID('QMQM/AMQ0INQ4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3INQ4')  /* RPG - ILE */

CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.SETPROCESS')     +
MQMNAME(&QMGRNAME)             +
REPLACE(*YES)               +
+
TEXT('Trigger process for AMQSSET4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSSET4')  /* C   */    +
/*    APPID('QMQM/AMQ0SET4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3SET4')  /* RPG - ILE */

CRTMQMPRC  PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')    +
MQMNAME(&QMGRNAME)             +
REPLACE(*YES)               +
+
TEXT('Trigger process for AMQSECH4')    +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here       **/        +
APPID('QMQM/AMQSECH4')  /* C   */    +
/*    APPID('QMQM/AMQ0ECH4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3ECH4')  /* RPG - ILE */

/*********************************************************************/
/*                                                                   */
/*  Normal return.                                                   */
/*                                                                   */
/*********************************************************************/
SNDPGMMSG  MSG('AMQSAMP4 Completed creating sample +
objects for ' *CAT &QMGRNAME)
RETURN
ENDPGM

/*********************************************************************/
/*                                                                   */
/* END OF AMQSAMP4                                                   */
/*                                                                   */
/*********************************************************************/
```

IBM MQ for IBM i 관리의 대체 방법

이 정보를 사용하여 IBM MQ for IBM i, MQSC 명령, PCF 명령 및 원격 관리에 대해 학습하십시오.

IBM MQ 도구 이벤트를 사용하여 큐 관리자의 조작을 모니터할 수 있습니다. IBM MQ 도구 이벤트에 대한 정보 및 이를 사용하는 방법의 경우, 도구 이벤트를 참조하십시오.

일반적으로 IBM i CL 명령을 사용하여 IBM MQ for IBM i를 관리합니다. 이러한 명령의 개요의 경우 166 페이지의 『CL 명령을 사용하여 IBM MQ for IBM i 관리』의 내용을 참조하십시오.

CL 명령 사용은 시스템 관리의 선호되는 메소드입니다. 그러나, 다양한 기타 메소드를 사용할 수 있습니다. 이 섹션에서는 해당 메소드의 개요를 제공하고 다음 토픽이 포함됩니다.

로컬 및 원격 관리

IBM MQ for IBM i 오브젝트를 로컬 또는 원격으로 관리합니다.

로컬 관리는 로컬 시스템에 정의한 큐 관리자에서 관리 태스크를 실행한다는 것을 의미합니다. IBM MQ에서 IBM MQ 채널이 관련되지 않기 때문에 로컬 관리로서 이를 고려할 수 있습니다. 즉, 통신이 운영 체제에 의해 관리됩니다. 이 태스크 유형을 수행하려면 원격 시스템에 로그온하고 거기에서 명령을 실행하거나 사용자를 위해 명령을 실행할 수 있는 프로세스를 작성해야 합니다.

IBM MQ에서는 원격 관리로서 알려진 것을 통해 단일 지점에서 관리를 지원합니다. 원격 관리는 대상 큐 관리자에서 프로그래밍 가능 명령 형식(PCF) 제어 메시지를 `SYSTEM.ADMIN.COMMAND.QUEUE`로 보내는 것으로 구성됩니다.

PCF 메시지를 생성하는 여러 방법이 있습니다. 즉, 다음과 같습니다.

1. PCF 메시지를 사용하여 프로그램 쓰기. 180 페이지의 『PCF 명령을 사용하여 관리』의 내용을 참조하십시오.
2. PCF 메시지를 전송하는 MQAI를 사용하여 프로그램 쓰기. 20 페이지의 『MQAI를 사용하여 PCF의 사용 단순화』의 내용을 참조하십시오.
3. 그래픽 사용자 인터페이스(GUI)를 사용할 수 있게 허용하고, 정확한 PCF 메시지를 생성하는 IBM MQ for Windows로 사용 가능한 IBM MQ 탐색기 사용. 181 페이지의 『IBM MQ for IBM i 인 IBM MQ 탐색기 사용』의 내용을 참조하십시오.
4. **STRMQMMQSC**를 사용하여 명령을 리모트 큐 관리자에 직접적으로 송신하십시오. 179 페이지의 『MQSC 명령을 사용하여 관리』의 내용을 참조하십시오.

예를 들어, 리모트 큐 관리자에서 큐 정의를 변경하기 위한 리모트 명령을 실행할 수 있습니다.

일부 명령은 특히, 큐 관리자를 작성하거나 시작하고 명령 서버를 시작하는 방법으로 실행될 수 없습니다. 이 태스크 유형을 수행하려면 원격 시스템에 로그온하고 거기에서 명령을 실행하거나 사용자를 위해 명령을 실행할 수 있는 프로세스를 작성해야 합니다.

MQSC 명령을 사용하여 관리

이 정보를 사용하여 MQSC 명령 및 IBM MQ for IBM i를 관리하기 위해 이를 사용하는 방법에 대해 학습하십시오.

MQSC(IBM MQ Script) 명령이 사람이 읽을 수 있는 양식 즉, EBCDIC 텍스트로 작성됩니다. 큐 관리자 자체, 큐, 프로세스 정의, 이름 목록, 채널, 클라이언트 연결 채널, 리스너, 서비스, 토픽 및 인증 정보 오브젝트를 포함하여 큐 관리자 오브젝트를 관리하기 위해 MQSC 명령을 사용합니다.

STRMQMMQSC IBM MQ CL 명령을 사용하여 큐 관리자에 MQSC 명령을 발행합니다. 이 메소드는 서버 라이브러리 시스템의 소스 실제 파일에서 해당 입력을 가져오는 유일한 배치 메소드입니다. 이 소스 실제 파일에 대한 기본 이름은 QMQSC입니다.

주의: QTEMP 라이브러리의 사용법이 제한된 것처럼 STRMQMMQSC에 소스 라이브러리로서 QTEMP 라이브러리를 사용하지 마십시오. 명령의 입력 파일로 다른 라이브러리를 사용해야 합니다.

IBM MQ for IBM i는 QMQSC라는 소스 파일을 제공하지 않습니다. MQSC 명령을 처리하려면 다음 명령을 실행하여 사용자의 선택의 라이브러리에서 QMQSC 소스 파일을 작성해야 합니다.

```
CRTSRCPF FILE(MYLIB/QMQSC) RCDLEN(240) TEXT('IBM MQ - MQSC Source')
```

MQSC 소스는 이 소스 파일 내에 멤버에서 유지됩니다. 멤버와 함께 일하려면 다음 명령을 입력하십시오.

```
WRKMBRPDM MYLIB/QMQSC
```

지금 새 멤버를 추가하고 기존 멤버를 유지보수할 수 있습니다.

RUNMQSC를 실행하여 대화식으로 MQSC 명령도 입력할 수 있습니다. 또는

1. 큐 관리자 이름에 입력하고 Enter 키를 눌러서 **WRKMQM** 결과 패널에 액세스하십시오.
2. 이 패널에서 F23=More options 선택하십시오.
3. 180 페이지의 그림 31에 표시된 패널에서 활성 큐 관리자에 대응하는 옵션 26을 선택하십시오.

MQSC 세션을 종료하려면 end 을 입력하십시오.

180 페이지의 그림 31은 해당 속성으로 MQSC 명령(DEFINE QLOCAL)을 표시하는 MQSC 명령 파일로부터의 추출입니다.

```
.
.
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;
.
.
```

그림 *31. MQSC* 명령*(myprog.in)*으로부터의 추출

IBM MQ 환경 사이의 이식성의 경우, MQSC 명령 파일의 행 길이를 72자로 제한하십시오. 더하기 부호는 명령이 다음 행에서 계속된다는 것을 표시합니다.

비록 대소문자가 구분되지 않더라도 MQSC에 지정되는 오브젝트 속성이 대문자(예: RQMNAME)로 이 절에 표시됩니다.

참고:

1. MQSC 파일의 형식은 파일 시스템에서 해당 위치에 의존하지 않습니다.
2. MQSC 속성 이름은 8자로 제한됩니다.
3. MQSC 명령은 z/OS를 포함하여 기타 플랫폼에서 사용 가능합니다.

각 MQSC 명령 및 해당 구문의 설명의 경우, 69 페이지의 『스크립트(MQSC) 명령』의 내용을 참조하십시오.

PCF 명령을 사용하여 관리

PCF(IBM MQ programmable command format) 명령의 목적은 관리 태스크가 관리 프로그램으로 프로그래밍될 수 있도록 하는 것입니다. 이런 방식으로 프로그램에서 큐를 작성하고 정의를 처리하며 큐 관리자를 변경할 수 있습니다.

PCF 명령은 MQSC 명령에서 제공하는 동일한 범위의 기능을 포함합니다. 그러나, MQSC 명령과 달리 PCF 명령 및 응답은 읽을 수 있는 텍스트 형식에 없습니다.

단일 노드에서 네트워크의 임의의 큐 관리자에 PCF 명령을 발행하는 프로그램을 작성할 수 있습니다. 이러한 방법으로 관리 태스크를 집중시키고 자동화할 수 있습니다.

각 PCF 명령은 IBM MQ 메시지의 애플리케이션 데이터 부분에 임베드되는 데이터 구조입니다. 각 명령은 기타 메시지와 동일한 방식으로 MQI 기능 MQPUT을 사용하여 대상 큐 관리자로 보내집니다. 메시지를 수신하는 큐 관리자의 명령 서버는 명령 메시지로서 이를 해석하고 명령을 실행합니다. 응답을 얻기 위해 애플리케이션은 MQGET 호출을 실행하고 응답 데이터가 다른 데이터 구조로 리턴됩니다. 그런 다음, 애플리케이션은 응답을 처리하고 그에 따라 수행합니다.

간략하게, 이는 애플리케이션 프로그래머가 PCF 명령 메시지를 작성하기 위해 지정해야 하는 것 중 일부입니다.

메시지 디스크립터
이는 표준 IBM MQ 메시지 디스크립터입니다.

- 메시지 유형(*MsgType*)은 MQMT_REQUEST입니다.
- 메시지 형식(*Format*)은 MQFMT_ADMIN입니다.

애플리케이션 데이터
PCF 헤더를 포함하는 PCF 메시지가 들어 있습니다.

- PCF 메시지 유형(*Type*)은 MQCFT_COMMAND를 지정합니다.
- 명령 ID는 명령(예: *Change Queue*(MQCMD_CHANGE_Q)을 지정합니다.

PCF 나가기는 메시지 텍스트 내에 MQSC 명령을 포함하는 PCF 명령입니다. PCF를 사용하여 리모트 큐 관리자로 명령을 송신할 수 있습니다. 자세한 정보는 20 페이지의 『MQAI를 사용하여 PCF의 사용 단순화』의 내용을 참조하십시오.

PCF 데이터 구조의 전체 설명 및 이를 구현하는 방법은 명령 및 응답에 대한 구조를 참조하십시오.

IBM MQ for IBM i 인 IBM MQ 탐색기 사용

IBM MQ Explorer를 사용하여 IBM MQ for IBM i 를 관리하려면 이 정보를 사용하십시오.

IBM MQ for Windows(x86 플랫폼) 및 IBM MQ for Linux(x86 및 x86-64 플랫폼)에서는 CL, 제어 또는 MQSC 명령 사용에 대한 대체로서 관리 태스크를 수행하기 위해 IBM MQ 탐색기라는 관리 인터페이스를 제공합니다.

The IBM MQ Explorer allows you to perform local or remote administration of your network from a computer running Windows (x86 platform), or Linux (x86 and x86-64 platforms), by pointing the IBM MQ Explorer at the queue managers and clusters you are interested in. IBM MQ 탐색기를 사용하여 관리될 수 있는 IBM MQ의 플랫폼 및 레벨이 리모트 큐 관리자에서 설명됩니다.

IBM MQ 탐색기로 다음을 수행할 수 있습니다.

- 큐 관리자를 시작하고 중지하십시오(로컬 시스템 전용).
- 큐, 토픽 및 채널과 같은 IBM MQ 오브젝트의 정의를 정의, 표시 및 대체하십시오.
- 큐에서 메시지를 찾으십시오.
- 채널을 시작하고 중지하십시오.
- 채널에 대한 상태 정보를 보십시오.
- 클러스터에서 큐 관리자를 보십시오.
- 애플리케이션, 사용자 또는 채널이 특정 큐를 여는지 확인하기 위해 검사하십시오.
- **새 클러스터 작성** 마법사를 사용하여 새 큐 관리자 클러스터를 작성하십시오.
- **클러스터에 큐 관리자 추가** 마법사를 사용하여 클러스터에 큐 관리자를 추가하십시오.
- SSL(Secure Sockets Layer) 채널 보안과 함께 사용되는 인증 정보 오브젝트를 관리하십시오.

온라인 자세한 내용을 사용하여 다음을 수행할 수 있습니다.

- 큐 관리자, 큐, 채널, 프로세스 정의, 클라이언트 연결 채널, 리스너, 토픽, 서비스, 이름 목록 및 클러스터를 포함하여 다양한 자원을 정의하고 제어하십시오.
- 큐 관리자 및 연관된 프로세스를 시작 또는 중지하십시오.
- 워크스테이션 또는 기타 워크스테이션에서 큐 관리자 및 연관된 오브젝트를 보십시오.

- 큐 관리자, 클러스터 및 채널의 상태를 검사하십시오.

IBM MQ 탐색기를 사용하여 서버 시스템에서 IBM MQ를 관리하도록 시도하기 전에 다음 요구사항을 충족시켰다는 것을 확인하십시오. 다음을 확인하십시오.

1. 명령 서버는 관리되는 **임의의** 큐 관리자를 위해 실행되고 CL 명령 **STRMQMCSVR**에 의해 서버에 시작됩니다.
2. 적당한 TCP/IP 리스너는 모든 리모트 큐 관리자를 위해 존재합니다. 이는 **STRMQMLSR** 명령으로 시작된 IBM MQ 리스너입니다.
3. SYSTEM.ADMIN.SVRCONN라는 서버 연결 채널은 모든 리모트 큐 관리자에 존재합니다. 이 채널을 작성 **해야** 합니다. 모든 리모트 큐 관리자가 관리되는 것은 필수입니다. 그것 없이, 원격 관리는 가능하지 않습니다.
4. SYSTEM.MQEXPLORER.REPLY.MODEL 큐가 존재한다는 것을 확인하십시오.

원격 관리를 위한 명령 서버 관리

이 정보를 사용하여 IBM MQ IBM i 명령 서버의 원격 관리에 대해 학습하십시오.

각 큐 관리자에는 이와 연관되는 명령 서버가 있을 수 있습니다. 명령 서버는 리모트 큐 관리자에서 수신되는 명령 또는 애플리케이션에서 PCF 명령을 처리합니다. 이는 처리를 위해 큐 관리자에 명령을 나타내고 원래 명령에 따라 완료 코드 또는 운영자 메시지를 리턴합니다.

명령 서버는 PCF, MQAI를 포함하는 모든 관리 및 원격 관리를 위해 필수적입니다.

참고: 원격 관리의 경우, 대상 큐 관리자가 실행 중인지 확인해야 합니다. 그렇지 않으면, 명령을 포함하는 메시지는 발행되는 큐 관리자를 남길 수 없습니다. 그 대신, 이러한 메시지는 리모트 큐 관리자를 제공하는 로컬 전송 큐에서 큐잉됩니다. 조금이라도 가능하다면 이 상황을 피하십시오.

명령 서버를 시작하고 중지하기 위한 분리 제어 명령이 있습니다. IBM MQ 탐색기를 사용하여 다음 섹션에서 설명된 조작을 수행할 수 있습니다.

명령 서버 시작 및 중지

명령 서버를 시작하려면 이 CL 명령을 사용하십시오.

```
STRMQMCSVR MQMNAME('saturn.queue.manager')
```

여기서 saturn.queue.manager는 명령 서버가 시작되는 큐 관리자입니다.

명령 서버를 중지하려면 다음 CL 명령 중 하나를 사용하십시오.

1. ```
   ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*CNTRLD)
   ```

   제어된 중지를 수행합니다. 여기서 saturn.queue.manager는 명령 서버가 중지되는 큐 관리자입니다. 이는 기본 옵션이고, 이는 OPTION(*CNTRLD)이 생략될 수 있다는 것을 의미합니다.

2. ```
   ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*IMMED)
   ```

   즉시 중지를 수행합니다. 여기서 saturn.queue.manager는 명령 서버가 중지되는 큐 관리자입니다.

명령 서버의 상태 표시

원격 관리의 경우, 대상 큐 관리자의 명령 서버가 실행 중인지 확인하십시오. 실행 중이지 않은 경우, 원격 명령을 처리할 수 없습니다. 명령을 포함하는 메시지는 대상 큐 관리자의 명령 큐 SYSTEM.ADMIN.COMMAND.QUEUE에서 큐잉됩니다.

여기서 saturn.queue.manager라고 하는 큐 관리자에 대한 명령 서버의 상태를 표시하기 위한 CL 명령은 다음과 같습니다.

```
DSPMQMCSVR MQMNAME('saturn.queue.manager')
```

대상 시스템에서 이 명령을 실행하십시오. 명령 서버가 실행 중인 경우, 183 페이지의 그림 32에 표시된 패널이 표시됩니다.

```
Display MQM Command Server (DSPMQMCSVR)


Queue manager name . . . . . > saturn.queue.manager

MQM Command Server Status. . . . > RUNNING




F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

그림 *32. MQM* 명령 서버 패널 표시

작업 관리

이 정보는 IBM MQ가 작업 요청을 핸들링하는 방식을 설명하고 IBM MQ와 연관되는 작업을 우선시키고 제어하기 위해 사용할 수 있는 옵션에 대해 상세히 설명합니다.

경고

IBM i 및 IBM MQ 작업 관리의 개념을 완전히 이해하지 않는 한 **되지 않음** 작업 관리 IBM MQ 를 변경하십시오.

서브시스템 및 작업 설명에 대한 추가 정보는 IBM i 제품 문서의 작업 관리 에서 찾을 수 있습니다. 작업 시작 및 배치(Batch) 작업 절을 유의하여 살펴보십시오.

IBM MQ for IBM i는 IBM i UNIX 환경과 IBM i 스레드를 통합합니다. IFS(Integrated File System)에서 오브젝트를 변경하지 **마십시오**.

정상 작동 동안, IBM MQ 큐 관리자는 다른 태스크를 수행하기 위해 수많은 배치 작업을 시작합니다. 기본적으로 이러한 배치 작업은 IBM MQ가 설치될 때 작성되는 QMQM 서브시스템에서 실행됩니다.

작업 관리는 사용자의 시스템으로부터 최적의 성능을 얻고 관리를 더 단순하게 하기 위해 IBM MQ 태스크를 맞춤화하는 프로세스를 참조합니다.

예를 들어, 다음 사항을 수행할 수 있습니다.

- 하나의 큐 관리자가 다른 큐 관리자보다 더 응답하도록 작업의 실행 우선순위를 변경하십시오.
- 수많은 작업의 출력을 특정 출력 큐로 경로 재지정하십시오.
- 특정 유형의 모든 작업이 특정 서브시스템에서 실행되도록 하십시오.
- 오류를 서브시스템으로 분리하십시오.

작업 관리는 IBM MQ 작업과 연관되는 작업 설명을 작성하거나 변경하여 수행됩니다. 다음에 대한 작업 관리를 구성할 수 있습니다.

- 전체 IBM MQ 설치
- 개별 큐 관리자
- 개별 큐 관리자에 대한 개별 작업

IBM MQ 태스크의 설명

이는 IBM MQ 작업의 테이블 및 각각에 대한 간략한 설명입니다.

큐 관리자가 실행 중인 경우, IBM MQ 서브시스템에서 QMQM 사용자 프로파일에서 실행 중인 다음 배치 작업 중 일부 또는 전부를 참조하십시오. 작업이 184 페이지의 표 11에서 간략하게 설명됩니다.

큐 관리자에 대한 작업(WRKMQM) 패널에서 옵션 22를 사용하여 큐 관리자에 연결되는 모든 작업을 볼 수 있습니다. WRKMQMLSR 명령을 사용하여 리스너를 볼 수 있습니다.

표 *11. IBM MQ* 태스크.

작업 이름	**Function**
AMQALMPX	주기적으로 저널 체크포인트를 사용하는 체크포인트 프로세서.
AMQZMUC0	유틸리티 관리자. 이 작업은 결정적 큐 관리자 유틸리티를 실행합니다(예: 저널 체인 관리자).
AMQZXMA0	큐 관리자가 시작하는 첫 번째 작업인 실행 제어기. 이는 MQCONN 요청을 핸들링하고 IBM MQ API 호출을 처리하기 위해 에이전트 프로세스를 시작합니다.
AMQZFUMA	오브젝트 권한 관리자(OAM).
AMQZLAA0	MQCNO_STANDARD_BINDING을 사용하여 큐 관리자에 연결하는 애플리케이션에 대한 대부분의 작업을 수행하는 큐 관리자 에이전트.
AMQZLSA0	큐 매니저 에이전트.
AMQZMUF0	유틸리티 관리자
AMQZMGR0	프로세스 제어기. 작업은 리스너 및 서비스를 시작하고 관리하기 위해 사용됩니다.
AMQZMUR0	유틸리티 관리자. 이 작업은 결정적 큐 관리자 유틸리티를 실행합니다(예: 저널 체인 관리자).
AMQFQPUB	큐된 발행/구독 디먼.
AMQFCXBA	브로커 작업 프로그램 작업.
RUNMQBRK	브로커 제어 작업.
AMQRMPPA	채널 프로세스 풀링 작업.
AMQCRSTA	TCP/IP 호출 채널 응답자.
AMQCRS6B	LU62 수신자 채널 및 클라이언트 연결(노트 참조).
AMQRRMFA	클러스터를 위한 저장소 관리자.
AMQCLMAA	비스레드 TCP/IP 리스너.
AMQPCSEA	PCF를 핸들링하고 원격 관리가 요청하는 PCF 명령 프로세서.
RUNMQTRM	트리거 모니터.
RUNMQDLQ	데드 레터 큐 핸들러.
RUNMQCHI	채널 시작기
RUNMQCHL	각 송신자 채널에 대해 사용되는 송신자 채널 작업.
RUNMQLSR	스레드 TCP/IP 리스너.
AMQRCMLA	채널 MQSC 및 PCF 명령 프로세서.

참고: LU62 수신자 작업이 통신 서브시스템에서 실행되고 작업을 시작하기 위해 사용되는 통신 입력 항목 및 라우팅에서 해당 런타임 특성을 사용합니다. 자세한 정보는 시작된 한쪽 끝(수신자)을 참조하십시오.

IBM MQ for IBM i 작업 관리 오브젝트

IBM MQ가 설치될 때, 작업 관리자를 지원하기 위해 다양한 오브젝트가 QMQM 라이브러리에 제공됩니다. 이러한 오브젝트는 자체 서브시스템에서 실행하기 위해IBM MQ 작업에 필요한 것입니다.

두 개의 IBM MQ 배치 작업에 샘플 작업 설명이 제공됩니다. 특정 작업 설명이 IBM MQ 작업에 제공되지 않으면 기본 작업 설명 QMQMJOBD로 실행됩니다.

IBM MQ를 설치할 때 제공되는 작업 관리 오브젝트는 185 페이지의 표 12에 나열되고 큐 관리자에 작성된 오브젝트는 185 페이지의 표 13에 나열됩니다.

참고: MQMQ 라이브러리에서 작업 관리 오브젝트를 찾을 수 있고 큐 관리자 라이브러리에서 큐 관리자 오브젝트를 찾을 수 있습니다.

표 *12.* 작업 관리 오브젝트

이름	유형	설명
AMQALMPX	*JOBD	체크포인트 프로세스에서 사용하는 작업 설명
AMQZLAA0	*JOBD	IBM MQ 에이전트 프로세스에서 사용하는 작업 설명
AMQZLSA0	*JOBD	고립된 바인딩 큐 관리자 에이전트
AMQZXMA0	*JOBD	IBM MQ 실행 제어기에서 사용하는 작업 설명
QMQM	*SBSD	모든 IBM MQ 작업이 실행되는 서브시스템
QMQM	*JOBQ	제공된 서브시스템에 첨부된 작업 큐
QMQMJOBD	*JOBD	작업에 대한 특정 작업 설명이 아닌 경우 사용되는 기본 IBM MQ 작업 설명
QMQMMSG	*MSGQ	IBM MQ 작업에 대한 기본 메시지 큐.
QMQMRUN20	*CLS	상위 우선순위 IBM MQ 작업에 대한 클래스 설명
QMQMRUN35	*CLS	중간 우선순위 IBM MQ 작업에 대한 클래스 설명
QMQMRUN50	*CLS	하위 우선순위 IBM MQ 작업에 대한 클래스 설명

표 *13.* 큐 관리자에 작성된 작업 관리 오브젝트

이름	유형	설명
AMQA000000	*JRNRCV	로컬 저널 수신자
AMQAJRN	*JRN	로컬 저널
AMQJRNINF	*USRSPC	큐 관리자의 시동 및 매체 복원에 필요한 최근 저널 수신자로 업데이트되는 사용자 공간. 이 사용자 공간은 어느 저널 수신자가 파일 보관을 요구하고 어느 것이 안전하게 삭제될 수 있는지 판별하기 위해 애플리케이션에 의해 질문받을 수 있습니다.
AMQAJRNMSG	*MSGQ	로컬 저널 메시지 큐
AMQCRC6B	*PGM	LU6.2 연결을 시작하기 위한 프로그램
AMQRFOLD	* 파일	마이그레이션된 큐 관리자 채널 정의 파일
QMQMMSG	*MSGQ	큐 관리자 메시지 큐

IBM MQ가 작업 관리 오브젝트 사용 방법

이 정보는 IBM MQ가 작업 관리 오브젝트를 사용하고 구성 예제를 제공하는 방법에 대해 설명합니다.

주의: 우선순위에 따라 서브시스템에서 허용되는 작업 수를 제한하기 위해 QMQM 서브시스템에서 작업 큐 입력 항목 설정을 대체하지 마십시오. 이를 수행하려고 시도하는 경우, 제출되고 큐 관리자 시동을 실패하게 한 후 본질적IBM MQ 작업이 실행되는 것을 방지할 수 있습니다.

작업 관리자를 구성하는 방법에 대해 이해하려면IBM MQ가 작업 설명을 사용하는 방법에 대해 먼저 이해해야 합니다.

작업을 시작하기 위해 사용되는 작업 설명은 많은 작업 속성을 제어합니다. 예를 들면, 다음과 같습니다.

- 작업이 큐잉되는 작업 큐 및 작업이 실행되는 서브시스템의 작업 큐.

- 작업이 해당 런타임 매개변수를 위해 사용하는 클래스 및 작업을 시작하기 위해 사용되는 라우팅 데이터.
- 작업이 인쇄 파일에 사용하는 출력 큐.

IBM MQ 작업을 시작하는 프로세스는 세 가지 단계로 구성될 수 있습니다.

1. IBM MQ는 작업 설명을 선택합니다.

   IBM MQ는 배치 작업에 사용할 작업 설명을 판별하기 위해 다음 기술을 사용합니다.

   a. 작업과 동일한 이름을 가지는 작업 설명에 대해 큐 관리자 라이브러리에서 찾으십시오. 큐 관리자 라이브러리에 대한 자세한 내용은 IBM MQ for IBM i 큐 관리자 라이브러리 이름 이해를 참조하십시오.

   b. 기본 작업 설명 QMQMJOBD에 대해 큐 관리자 라이브러리에서 찾으십시오.

   c. 작업과 동일한 이름을 가지는 작업 설명에 대해 QMQM 라이브러리에서 찾으십시오.

   d. QMQM 라이브러리에서 기본 작업 설명(QMQMJOBD)을 사용하십시오.

2. 작업이 작업 큐에 제출됩니다.

   IBM MQ와 함께 제공되는 작업 설명은 기본적으로 라이브러리 QMQM의 작업 큐 QMQM에 작업을 배치시키기 위해 설정됩니다. QMQM 작업 큐는 제공된 QMQM 서브시스템에 첨부됩니다. 따라서 기본적으로 작업은 QMQM 서브시스템에서 실행을 시작됩니다.

3. 작업은 서브시스템을 입력하고 라우팅 단계를 통해 이동합니다.

   작업이 서브시스템을 입력할 때, 작업 설명에 지정된 라우팅 데이터는 작업에 대한 라우팅 입력 항목을 찾기 위해 사용됩니다.

   라우팅 데이터는 QMQM 서브시스템에 정의된 라우팅 입력 항목 중 하나와 일치하고 이는 작업에 사용하는 제공된 클래스(QMQMRUN20, QMQMRUN35 또는 QMQMRUN50)를 정의합니다.

참고: IBM MQ 작업이 시작되는 것으로 표시되지 않지만,서브시스템이 실행 중이고 작업 큐가 없다는 것을 확인하십시오.

IBM MQ 작업 관리 오브젝트를 수정한 경우, 모든 것이 올바르게 연관된다는 것을 확인하십시오. 예를 들어, 작업 설명에서 QMQM/QMQM 이외의 작업 큐를 지정하는 경우, ADDJOBQE가 서브시스템, 즉 QMQM에 대해 실행된다는 것을 확인하십시오.

예로서 다음 워크시트를 사용하여 184 페이지의 표 11에 문서화된 각 작업에 대해 작업 설명을 작성할 수 있습니다.

```
What is the queue manager library name? ___________
Does job description AMQZXMA0 exist in the queue manager library? Yes   No
Does job description QMQMJOBD exist in the queue manager library? Yes   No
Does job description AMQZXMA0 exist in the QMQM library?     Yes   No
Does job description QMQMJOBD exist in the QMQM library?     Yes   No
```

이러한 모든 질문에 No로 대답하는 경우, QMQM 라이브러리에서 글로벌 작업 설명 QMQMJOBD를 작성하십시오.

IBM MQ 메시지 큐

IBM MQ 메시지 큐(QMQMMSG)가 각 큐 관리자 라이브러리에서 작성됩니다. 운영 체제 메시지는 큐 관리자 작업이 종료되고 IBM MQ가 큐로 메시지를 보낼 때 이 큐로 보내집니다. 예: 시동 시 필요한 저널 수신자를 보고하기 위해. 모니터링하는 것을 보다 용이하게 하기 위한 관리 가능한 크기에 메시지의 수를 이 메시지 큐에 보존하십시오.

기본 시스템 예

이러한 예는 수정되지 않은 IBM MQ 설치가 일부 표준 작업이 큐 관리자 시동 시간에 제출될 때 작동하는 방법을 표시합니다.

먼저, AMQZXMA0 실행 제어기 작업이 시작됩니다.

1. 큐 관리자 TESTQM에 대한 **STRMQM** 명령을 실행하십시오.

2. IBM MQ는 먼저 작업 설명 AMQZXMA0 및 작업 설명 QMQMJOBD에 대해 큐 관리자 라이브러리 QMTESTQM을 검색합니다.

   이러한 작업 설명 중 어느 쪽도 존재하지 않아서 IBM MQ는 제품 라이브러리 QMQM에서 작업 설명 AMQZXMA0를 검색합니다. 이 작업 설명이 존재합니다. 따라서 작업을 제출하기 위해 사용됩니다.
3. 작업 설명은 IBM MQ 기본 작업 큐를 사용합니다. 따라서 작업이 작업 큐 QMQM/QMQM에 제출됩니다.
4. AMQZXMA0 작업 설명에 대한 라우팅 데이터는 QMQMRUN20입니다. 따라서 시스템은 해당 데이터와 일치하는 것에 대한 서브시스템 라우팅 입력 항목을 검색합니다.

   기본적으로, 순서 번호 9900을 가지는 라우팅 입력 항목에는 QMQMRUN20과 일치하는 비교 데이터가 있습니다. 따라서 작업은 QMQMRUN20이라고도 하는 해당 라우팅 입력 항목에 정의되는 클래스와 함께 시작됩니다.
5. QMQM/QMQMRUN20 클래스는 20으로 설정된 우선순위로 실행되었습니다. 따라서, AMQZXMA0 작업은 시스템에서 가장 대화식 작업과 동일한 우선순위로 서브시스템 QMQM에서 실행됩니다.

다음, AMQALMPX 체크포인트 프로세스 작업이 시작됩니다.

1. IBM MQ는 먼저 작업 설명 AMQALPMX에 대한 큐 관리자 라이브러리 QMTESTQM을 검색한 다음 작업 설명 QMQMJOBD를 검색하십시오.

   이러한 작업 설명 중 어느 쪽도 존재하지 않아서 IBM MQ는 제품 라이브러리 QMQM에서 작업 설명 AMQALMPX 및 QMQMJOBD를 검색합니다.

   직업 설명 AMQALMPX는 존재하지 않지만 QMQMJOBD는 존재합니다. 따라서 QMQMJOBD는 작업을 제출하는데 사용됩니다.

   참고: QMQMJOBD 작업 설명은 자체 작업 설명을 가지고 있지 않은 IBM MQ 작업에 항상 사용됩니다.
2. 작업 설명은 IBM MQ 기본 작업 큐를 사용합니다. 따라서 작업이 작업 큐 QMQM/QMQM에 제출됩니다.
3. QMQMJOBD 작업 설명에 대한 라우팅 데이터는 QMQMRUN35입니다. 따라서 시스템은 해당 데이터와 일치하는 것에 대한 서브시스템 라우팅 입력 항목을 검색합니다.

   기본적으로, 순서 번호 9910을 가지는 라우팅 입력 항목에는 QMQMRUN35와 일치하는 비교 데이터가 있습니다. 따라서 작업은 QMQMRUN35라고도 하는 해당 라우팅 입력 항목에 정의되는 클래스와 함께 시작됩니다.
4. QMQM/QMQMRUN35 클래스에는 35으로 설정된 우선순위가 실행됩니다. 따라서, AMQALMPX 작업은 시스템에서 가장 대화식 작업과 동일한 우선순위로 서브시스템 QMQM에서 실행됩니다.

작업 관리 구성 예

이 정보를 사용하여 IBM MQ 작업의 런타임 속성을 변경하기 위해 IBM MQ 작업 설명을 변경하고 작성할 수 있는 방법에 대해 학습하십시오.

IBM MQ 작업 관리의 유연성의 키는 IBM MQ이 작업 설명을 찾는 2단계 방법에 있습니다.

- 큐 관리자 라이브러리에서 작업 설명을 작성하거나 변경하는 경우, 해당 변경사항이 QMQM에서 글로벌 작업 설명을 대체하지만 변경사항은 로컬이고 해당 특정 큐 관리자에 단독으로 영향을 줍니다.
- QMQM 라이브러리에서 글로벌 작업 설명을 작성하거나 변경하는 경우, 개별 큐 관리자에 대해 로컬로 대체되지 않으면 해당 작업 설명이 시스템에서 모든 큐 관리자에 영향을 줍니다.

1. 다음 예는 개별 큐 관리자에 대한 채널 제어 작업의 우선순위를 증가시킵니다.

   저장소 관리자 및 채널 시작기 작업(AMQRRMFA 및 RUNMQCHI)을 작성하려면 큐 관리자 TESTQM에 대해 가능한 빨리 수행하고 다음 단계를 수행하십시오.

   a. 큐 관리자 라이브러리에서 제어하려는 IBM MQ 프로세스의 이름으로 QMQM/QMQMJOBD 작업 설명의 로컬 복제를 작성하십시오. 예를 들면, 다음과 같습니다.

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
NEWOBJ(RUNMQCHI)
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
NEWOBJ(AMQRRMFA)
```

b. 작업이 QMQMRUN20 클래스를 사용한다는 것을 확인하기 위해 작업 설명에서 라우팅 데이터 매개변수를 변경하십시오.

```
CHGJOBD JOBD(QMTESTQM/RUNMQCHI) RTGDTA('QMQMRUN20')
CHGJOBD JOBD(QMTESTQM/AMQRRMFA) RTGDTA('QMQMRUN20')
```

큐 관리자 TESTQM에 대한 AMQRRMFA 및 RUNMQCHI 작업.

- 큐 관리자 라이브러리에서 새로운 로컬 작업 설명을 사용하십시오.
- 작업이 서브시스템을 입력할 때 QMQMRUN20 클래스가 사용되므로 우선순위 20으로 실행하십시오.

2. 다음 예제는 QMQM 서브시스템에 대한 새 실행 우선순위 클래스를 정의합니다.

a. 기타 큐 관리자가 클래스에 액세스하도록 허용하려면 다음 명령을 실행하여 QMQM 라이브러리에서 복제 클래스를 작성하십시오.

```
CRTDUPOBJ OBJ(QMQMRUN20) FROMLIB(QMQM) OBJTYPE(*CLS) TOLIB(QMQM)
NEWOBJ(QMQMRUN10)
```

b. 다음 명령을 실행하여 새 실행 우선순위를 가지도록 클래스를 변경하십시오.

```
CHGCLS CLS(QMQM/QMQMRUN10) RUNPTY(10)
```

c. 다음 명령을 실행하여 서브시스템에 새 클래스 정의를 추가하십시오.

```
ADDRTGE SBSD(QMQM/QMQM) SEQNBR(8999) CMPVAL('QMQMRUN10') PGM(QSYS/QCMD)
CLS(QMQM/QMQMRUN10)
```

참고: 라우팅 순서 번호에 숫자 값을 지정할 수 있지만, 값은 순차 순서에 있어야 합니다. 이 순서 번호는 라우팅 입력 항목이 라우팅 데이터 일치를 검색할 순서를 서브시스템에 알려줍니다.

d. 다음 명령을 실행하여 새 우선순위 클래스를 사용하려면 로컬 또는 글로벌 작업 설명을 변경하십시오.

```
CHGJOBD JOBD(QMQMlibname/QMQMJOBD) RTGDTA('QMQMRUN10')
```

QMlibraryname과 연관되는 모든 큐 관리자 작업은 10의 실행 우선순위를 사용합니다.

3. 다음 예는 자체 서브시스템에서 큐 관리자를 실행합니다

큐 관리자 TESTQM에 대한 모든 작업이 QBATCH 서브시스템에서 실행되도록 하려면 다음 단계를 실행하십시오.

a. 큐 관리자 라이브러리에서 명령을 사용하여 QMQM/QMQMJOBD 작업 설명의 로컬 복제를 작성하십시오

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
```

b. 작업이 QBATCH 작업 큐를 사용한다는 것을 확인하기 위해 작업 설명에서 작업 큐 매개변수를 변경하십시오.

```
CHGJOBD JOBD(QMTESTQM/QMQMJOBD) JOBQ(*LIBL/QBATCH)
```

참고: 작업 큐는 서브시스템 설명과 연관됩니다. 작업이 작업 큐에 남아 있다는 것을 알면, 작업 큐 정의가 SBSD에서 정의되는지 확인하십시오. 서브시스템에 DSPSBSD 명령을 사용하고 옵션 **6**, "작업 큐 입력 항목"을 사용하십시오.

큐 관리자 TESTQM에 대한 모든 작업:

- 큐 관리자 라이브러리에서 새 로컬 기본 작업 설명을 사용하십시오
- 작업 큐 QBATCH로 제출됩니다.

작업이 라우팅되고 올바르게 우선순위 지정되는지 확인하려면 다음을 수행하십시오.

- 서브시스템 QBATCH에서 IBM MQ 작업에 대한 라우팅 입력 항목을 작성하거나
- 라우팅 데이터가 사용되는 것과 상관 없이 QCMD를 호출하는 광범위한 라우팅 입력 항목을 신뢰하십시오.

  작업 큐 QBATCH에 대한 최대 활성 작업 옵션이 *NOMAX로 설정되는 경우에만 이 옵션이 작동합니다. 시스템 기본값은 1입니다.

4. 다음 예는 다른 IBM MQ 서브시스템을 작성합니다

   a. 다음 명령을 실행하여 QMQM 라이브러리에서 복제 서브시스템을 작성하십시오.

   ```
   CRTDUPOBJ OBJ(QMQM) FROMLIB(QMQM) OBJTYPE(*SBSD) TOLIB(QMQM) NEWOBJ(QMQM2)
   ```

   b. 다음 명령을 실행하여 QMQM 작업 큐를 제거하십시오.

   ```
   RMVJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM)
   ```

   c. 다음 명령을 실행하여 서브시스템에 대한 새 작업 큐를 작성하십시오.

   ```
   CRTJOBQ JOBQ(QMQM/QMQM2) TEXT('Job queue for IBM MQ Queue Manager')
   ```

   d. 다음 명령을 실행하여 작업 큐 입력 항목을 서브시스템에 추가하십시오.

   ```
   ADDJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM2) MAXACT(*NOMAX)
   ```

   e. 다음 명령을 실행하여 큐 관리자 라이브러리에서 복제 QMQMJOBD를 작성하십시오.

   ```
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMlibraryname)
   ```

   f. 다음 명령을 실행하여 새 작업 큐를 사용하려면 작업 설명을 변경하십시오.

   ```
   CHGJOBD JOBD(QMlibraryname/QMQMJOBD) JOBQ(QMQM/QMQM2)
   ```

   g. 다음 명령을 실행하여 서브시스템을 시작하십시오.

   ```
   STRSBS SBSD(QMQM/QMQM2)
   ```

   참고:

   a. 라이브러리에서 서브시스템을 지정할 수 있습니다. 어떤 이유로도 제품이 재설치되거나 QMQM 라이브러리가 대체되는 경우, 사용자가 작성한 변경사항이 제거됩니다.

   b. QMlibraryname과 연관되는 모든 큐 관리자 작업은 서브시스템 QMQM2에서 실행됩니다.

5. 다음 예제는 작업 유형에 대한 모든 출력을 수집합니다.

   모든 체크포인트 프로세스를 수집하기 위해, AMQALMPX, 단일 출력 큐에서 다중 큐 관리자를 위한 작업 로그는 다음 단계를 수행합니다.

   a. 출력 큐를 작성하십시오. 예:

   ```
   CRTOUTQ OUTQ(MYLIB/CHCKPTLOGS)
   ```

   b. 제어하려는 IBM MQ 프로세스의 이름을 사용하여, QMQM/QMQMJOBD 작업 설명의 전체적 복제를 작성하십시오. 예:

   ```
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) NEWOBJ(AMQALMPX)
   ```

   c. 새 출력 큐를 가리키도록 작업 설명에서 출력 큐 매개변수를 변경하고, 모든 메시지가 작업 로그에 작성되도록 작업 로깅 레벨를 변경하십시오.

```
CHGJOBD JOBD(QMQM/AMQALMPX) OUTQ(MYLIB/CHKPTLOGS) LOG(4 00 *SECLVL)
```

모든 큐 관리자의 경우, 모든 IBM MQ AMQALMPX 작업은 로컬 큐 관리자 라이브러리에 로컬 대체 작업 설명이 없다면 새 글로벌 AMQALMPX 작업 설명을 사용합니다.

이러한 작업을 위한 모든 작업 로그 스풀 파일은 라이브러리 MYLIB의 큐 CHKPTLOGS를 출력하기 위해 지금 작성됩니다.

참고:

a. QPJOBLOG 또는 인쇄 파일이 해당 출력 큐 매개변수에 대해 *JOB의 값을 가지고 있는 경우에만 선행 예가 작동합니다. 앞선 예제에서, QSYS/QPDJOBLOG 파일에는 *JOB으로 설정되는OUTQ가 필요합니다.

b. 시스템 인쇄 파일을 변경하려면, CHGPRTF 명령을 사용하십시오. 예를 들면, 다음과 같습니다.

```
CHGPRTF PRTF(QJOBLOG) OUTQ(*JOB)
```

*JOB 옵션은 작업 설명이 사용되어야 한다는 것을 표시합니다.

c. IBM MQ 작업과 연관되는 스풀 파일을 특정 출력 큐로 보낼 수 있습니다. 그러나, 사용되는 인쇄 파일에 OUTQ 매개변수에 대한 적절한 값이 있다는 것을 확인하십시오.

가용성, 백업, 복구 및 재시작

이 정보를 사용하여 IBM MQ for IBM i가 해당 백업 및 복원 전략을 돕기 위한 IBM i 저널링 지원을 사용하는 방법에 대해 이해하십시오.

이 섹션을 읽기 전에 표준 IBM i 백업 및 복구 메소드와 IBM i에서 연관된 저널 및 저널의 사용에 익숙해야 합니다. 이러한 주제에 대한 정보는 백업 및 복구를 참조하십시오.

백업 및 복구 전략에 대해 이해하려면 먼저 IBM MQ for IBM i가 IBM i 파일 시스템 및 IFS(Integrated File System)에서 해당 데이터를 조직하는 방법에 대해 이해해야 합니다.

IBM MQ for IBM i는 IFS 파일 시스템의 스트림 파일 및 각 큐 관리자 인스턴스에 대한 개별 라이브러리에 해당 데이터를 보유합니다.

큐 관리자 특정 라이브러리에는 큐 관리자의 작업 관리를 제어하기 위해 필요한 저널, 저널 수신자 및 오브젝트가 포함됩니다. IFS 디렉토리 및 파일에는 IBM MQ 구성 파일, IBM MQ 오브젝트의 설명 및 포함하는 데이터가 포함됩니다.

시스템 실패에 대한 복구 가능한 이러한 오브젝트에 대한 변경은 적절한 오브젝트에 적용되기 전에 저널에 기록됩니다. 여기에는 저널에 기록된 정보를 재실행하여 그러한 변경사항이 복구될 수 있다는 효과가 있습니다.

서버 또는 큐 관리자 실패의 경우에 복구의 속도를 높이고 증가된 큐 관리자 가용성을 제공하기 위해 다른 서버에서 여러 큐 관리자 인스턴스를 사용하여 IBM MQ for IBM i를 구성할 수 있습니다.

IBM MQ for IBM i 저널

이 정보를 사용하여 IBM MQ for IBM i가 로컬 오브젝트에 대한 업데이트를 제어하기 위해 해당 조작에서 저널을 사용하는 방법에 대해 이해하십시오.

각 큐 관리자 라이브러리에는 해당 큐 관리자에 대한 저널이 있으며 저널에는 QM *GRLIB*/AMQ *A* JRN이름이 있습니다. 여기서 QM *GRLIB* 은 큐 관리자 라이브러리의 이름이고, *A* 는 큐 관리자 인스턴스에 고유한 단일 인스턴스 큐 관리자의 경우 A 입니다.

QM *GRLIB*은 QM 이름을 사용하고, 고유한 양식으로 큐 관리자의 이름이 뒤에 옵니다. 예를 들어, TEST라고 이름 지정된 큐 관리자에는 QMTEST로 이름 지정된 큐 관리자 라이브러리가 있습니다. **CRTMQM** 명령을 사용하여 큐 관리자를 작성할 때 큐 관리자 라이브러리가 지정될 수 있습니다.

저널은 저널링되는 정보를 포함하는 저널 수신자를 연관시킵니다. 수신자는 추가될 수 있고 결국 채워지는 정보에 대한 오브젝트입니다.

저널 수신자는 구식 정보로 가치 있는 디스크 공간을 다 사용합니다. 그러나, 이 문제를 최소화하기 위해 영구적인 스토리지에서 정보를 대체할 수 있습니다. 특정 시간에 하나의 저널 수신자가 저널에 첨부됩니다. 저널 수신자

가 사전정의된 임계값 크기에 도달하면, 새 저널 수신자에 의해 분리되고 대체됩니다. **CRTMQM** 및 **THRESHOLD** 매개변수를 사용하여 큐 관리자를 작성할 때 저널 수신자의 임계값을 지정할 수 있습니다.

로컬 IBM MQ for IBM i 저널과 연관된 저널 리시버가 각 큐 관리자 라이브러리에 있으며 다음과 같이 이름 지정 규칙을 채택합니다.

```
AMQ Arnnnnn
```

여기서,

A
는 문자 A-Z입니다. 이는 단일 인스턴스 큐 관리자에 대한 A입니다. 이는 여러 인스턴스 큐 관리자의 다른 인스턴스에 따라 다양합니다.

nnnnn
시퀀스의 다음 저널에 대해 1씩 증분되는 10진수 00000 to 99999 입니다.

r
은 리시버가 복원될 때마다 1씩 증가하는 10진수 0 to 9입니다.

저널 순서는 날짜를 기반으로 합니다. 그러나, 다음 저널의 이름 지정 규칙은 다음 규칙을 기반으로 합니다.

1. AMQArnnnnn은 AMQAr(nnnnn+1)로 이동하고 nnnnn은 99999에 도달할 때 줄 바꾸기를 합니다. 예를 들어, AMQA099999는 AMQA000000으로 이동하고 AMQA999999는 AMQA900000으로 이동합니다.
2. 규칙 1이 생성한 이름을 가지는 저널이 이미 존재하는 경우, CPI70E3이 QSYSOPR 메시지 큐로 보내지고 자동 수신자 전환이 중지됩니다.

   현재 첨부된 수신자는 문제점을 조사하고, 수동으로 새 수신자를 첨부시킬 때까지 계속 사용됩니다.
3. 어떤 새 이름도 시퀀스에 이용할 수 없으면(즉, 가능한 모든 저널 이름이 시스템에 있음), 다음 둘 다를 수행해야 합니다.
   a. 더 이상 필요하지 않는 저널을 삭제하십시오(195 페이지의 『저널 관리』 참조).
   b. 사용 중인 최근 저널 수신자로 변경한 저널을 기록하고(**RCDMQMIMG**) 이전 단계를 반복하십시오. 이는 오래된 저널 수신자 이름이 재사용될 수 있게 허용합니다.

AMQAJRN 저널은 MNGRCV(*SYSTEM) 옵션을 사용하여 임계값에 도달될 때 운영 체제가 자동으로 저널 수신자를 변경할 수 있도록 합니다. 시스템이 수신자를 관리하는 방법에 대한 자세한 정보는 *IBM i* 백업 및 복구를 참조하십시오.

저널 수신자의 기본 임계값은 100,000KB입니다. 이를 큐 관리자를 작성할 때 더 큰 값으로 설정할 수 있습니다. LogReceiverSize 속성의 초기값이 mqs.ini 파일의 LogDefaults 스탠자에 작성됩니다.

저널 수신자가 지정된 임계값을 넘어 연장될 때, 이전 수신자에서 속성을 상속하여 수신자가 분리되고 새 저널 수신자가 작성됩니다. 큐 관리자가 작성한 이후 LogReceiverSize 또는 LogASP 속성에 대한 변경사항은 시스템이 새 저널 수신자를 자동으로 첨부할 때 무시됩니다

시스템 구성에 대한 추가 세부사항은 IBM i에서 구성 정보 변경을 참조하십시오.

큐 관리자가 작성한 이후 저널 수신자의 크기를 변경해야 하는 경우, 새 저널 수신자를 작성하고 다음 명령을 사용하여 QMQM에 소유자를 설정하십시오.

```
CRTJRNRCV JRNRCV(QM GRLIB/AMQ Arnnnnn) THRESHOLD(xxxxxx) +
TEXT('MQM LOCAL JOURNAL RECEIVER')
CHGOBJOWN OBJ(QM GRLIB/AMQ Arnnnnn) OBJTYPE(*JRNRCV) NEWOWN(QMQM)
```

여기서,

QMGRLIB
큐 관리자 라이브러리의 이름입니다.

A
인스턴스 ID(일반적으로 A)입니다.

rnnnnn
이전에 설명된 이름 지정 순서에 있는 다음 저널 수신자입니다.

xxxxxx
새 수신자 임계값입니다(KB).

참고: 수신자의 최대 크기는 운영 체제에 의해 통제됩니다. 이 값을 확인하려면 **CRTJRNRCV** 명령에서 THRESHOLD 키워드를 보십시오.

새 수신자를 명령으로 AMQAJRN 저널에 첨부하십시오.

```
CHGJRN JRN(QMGRLIB/AMQ A JRN) JRNRCV(QMGRLIB/AMQ Annnnnn)
```

이러한 저널 수신자를 관리하는 방법에 대한 세부사항은 195 페이지의 『저널 관리』의 내용을 참조하십시오.

IBM MQ for IBM i 저널 사용법

이 정보를 사용하여 IBM MQ for IBM i가 로컬 오브젝트에 대한 업데이트를 제어하기 위해 해당 조작에서 저널을 사용하는 방법에 대해 이해하십시오.

메시지 큐에 대한 지속 업데이트는 두 단계로 발생합니다. 업데이트를 나타내는 레코드는 저널에 먼저 작성되고, 그런 다음 큐 파일이 업데이트됩니다.

따라서 저널 수신자는 큐 파일보다 최신 데이터가 될 수 있습니다. 일관성 지점에서 재시작 처리가 시작되도록 하기 위해 IBM MQ는 체크포인트를 사용합니다.

체크포인트는 저널에 설명된 레코드가 큐의 레코드와 동일할 때입니다. 체크포인트 자체는 큐 관리자를 재시작해야 하는 일련의 저널 레코드로 구성됩니다. 예를 들어, 모든 트랜잭션의 상태(즉, 작업 단위)는 체크포인트 당시에 활성화됩니다.

체크포인트는 IBM MQ에서 자동으로 생성됩니다. 큐 관리자가 시작되고 시스템이 종료될 때 사용되고 특정 수의 조작이 로그된 이후에 사용됩니다.

큐 관리자에서 모든 오브젝트에 대한 RCDMQMIMG 명령을 실행하고 결과를 표시하여 다음과 같이 큐 관리자가 체크포인트를 가져오도록 강제실행할 수 있습니다.

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>) DSPJRNDTA(*YES)
```

큐는 추가 메시지를 핸들링하기 때문에 체크포인트 레코드는 큐의 현재 상태와 불일치하게 됩니다.

IBM MQ가 재시작될 때, 로그에 최신 체크포인트 레코드를 배치합니다. 이 정보는 모든 체크포인트의 끝에서 업데이트되는 체크포인트 파일에 있습니다. 체크포인트 레코드는 로그와 데이터 사이의 최신 일관성 지점을 나타냅니다. 이 체크포인트로부터의 데이터는 체크포인트 시간에 존재하는 것처럼 큐를 다시 빌드하기 위해 사용됩니다. 큐가 재작성될 때, 로그는 시스템 장애 또는 닫기 전에 있는 상태로 큐를 다시 되돌리기 위해 전달됩니다.

IBM MQ가 저널을 사용하는 방법에 대해 이해하려면 큐 관리자 TEST에서 TESTQ라는 로컬 큐의 경우를 고려하십시오. 이는 다음 IFS 파일에 의해 표시됩니다.

```
/QIBM/UserData/mqm/qmgrs/TEST/queues
```

지정된 메시지가 이 큐에 배치되고 큐에서 검색되는 경우, 취해지는 조치가 193 페이지의 그림 33에서 표시됩니다.

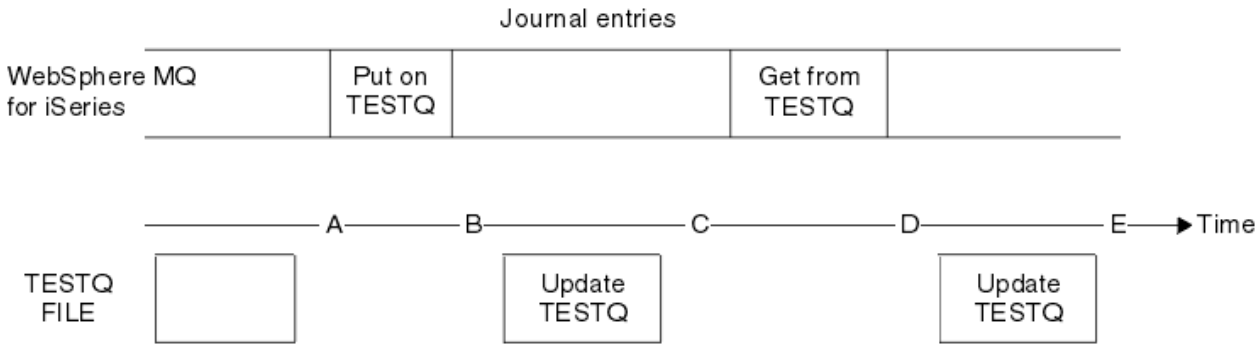


그림 *33. MQM* 오브젝트를 업데이트할 때 이벤트의 순서

다이어그램에 표시된 다섯 개의 지점(A-E)은 다음 상태를 정의하는 시간을 표시합니다.

A
큐의 IFS 파일 표현이 저널에 포함되는 정보와 일치합니다.

B
저널 입력 항목은 큐에 `Put` 조작을 정의하여 저널에 작성됩니다.

C
큐가 적절히 업데이트됩니다.

D
저널 입력 항목은 큐에서 `Get` 조작을 정의하여 저널에 작성됩니다.

E
큐가 적절히 업데이트됩니다.

IBM MQ for IBM i의 복구 기능에 대한 키는 사용자가 시간 A 당시의 TESTQ의 IFS 파일 표현을 저장할 수 있고, 계속해서 저장된 오브젝트를 복원하고 시간 A로부터 저널에 있는 입력 항목을 재실행하여 시간 E 당시의 TESTQ의 IFS 파일 표현을 복구할 수 있다는 것입니다.

이 전략은 시스템 실패 이후 지속 메시지를 복구하기 위해 IBM MQ for IBM i에서 사용됩니다. IBM MQ는 저널 수신자의 특정 입력 항목을 기억하고, 시동 시 이 지점으로부터 저널에 있는 입력 항목을 재실행하도록 합니다. 이 시동 입력 항목은 IBM MQ가 다음 시동 시 최소 필수 재실행만 수행하면 되도록 주기적으로 재계산됩니다.

IBM MQ에서는 오브젝트의 개별 복구가 제공됩니다. 오브젝트와 관련되는 모든 지속 정보는 로컬 IBM MQ for IBM i 저널에 기록됩니다. 손상되는 모든 IBM MQ 오브젝트는 저널에 보유된 정보로부터 완전하게 다시 빌드될 수 있습니다.

시스템이 수신자를 관리하는 방법에 대한 자세한 정보는 190 페이지의 『가용성, 백업, 복구 및 재시작』의 내용을 참조하십시오.

매체 이미지

이 정보를 사용하여 매체 이미지 및 매체 이미지로부터의 복구에 대해 이해하십시오.

긴 지속 기간의 IBM MQ 오브젝트는 작성된 지점으로 돌아가는 많은 수의 저널 입력 항목을 나타낼 수 있습니다. 이 문제를 방지하기 위해 IBM MQ for IBM i은 오브젝트의 매체 이미지의 개념을 가지고 있습니다.

이 매체 이미지는 저널에 기록된 IBM MQ 오브젝트의 완전한 사본입니다. 오브젝트의 이미지가 사용되는 경우, 오브젝트는 계속 이 이미지로부터 저널 입력 항목을 재실행하여 다시 빌드될 수 있습니다. 각 IBM MQ 오브젝트에 대한 재실행 지점을 나타내는 저널의 입력 항목은 해당 매체 복구 입력 항목으로서 참조됩니다. IBM MQ는 트랙을 유지합니다.

- 각 큐 관리자 오브젝트에 대한 매체 복원 입력 항목.
- 이 세트로부터의 가장 오래된 항목(자세한 내용은 195 페이지의 『저널 관리』의 오류 메시지 AMQ7462를 참조).

*CTLG 오브젝트 및 *MQM 오브젝트의 이미지는 이러한 오브젝트가 큐 관리자 재시작에 결정적이므로 규칙적으로 사용됩니다.

편리할 때 기타 오브젝트의 이미지가 사용됩니다. 기본적으로, **모든** 오브젝트 이미지는 큐 관리자가 매개변수 ENDCCTJOB(*YES)와 함께 **ENDMQM** 명령을 사용하여 시스템 종료될 때 사용됩니다. 이 조작은 매우 큰 큐 관리자에 대해 상당한 시간이 걸릴 수 있습니다. 빠르게 시스템 종료해야 하는 경우, ENDCCTJOB(*YES)으로 RCDMQMIMG(*NO) 매개변수를 지정하십시오. 그러한 경우, 다음 명령을 사용하여 큐 관리자가 재시작된 후 저널에서 완전한 매체 이미지를 기록하는 것이 좋습니다.

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>)
```

IBM MQ는 오브젝트가 저널에서 작은 입력 항목에 의해 설명될 수 있는 편리한 위치를 발견하는 경우 오브젝트의 이미지를 자동으로 기록합니다. 그러나, 이는 일부 오브젝트(예: 많은 수의 메시지를 지속적으로 포함하는 큐)에는 절대 발생할 수 없습니다.

가장 오래된 미디어 복구 항목의 날짜가 불필요하게 장 기간 동안 계속될 수 있게 허용하기보다, 수동으로 선택한 오브젝트의 이미지를 사용할 수 있게 하는 IBM MQ 명령 RCDMQMIMG을 사용하십시오.

매체 이미지로부터의 복구

IBM MQ는 손상되었다는 것이 발견되면 해당 매체 이미지로부터 일부 오브젝트를 자동으로 복구합니다. 특히, 이는 정상 큐 관리자 시동의 일부로서 특수 *MQM 및 *CTLG 오브젝트에 적용됩니다. 동기점 트랜잭션이 큐 관리자의 최근 시스템 종료에서 미완료된 경우, 영향을 받은 큐도 시동 조작을 완료하기 위해 자동으로 복구됩니다.

IBM MQ 명령 RCRMQMOBJ을 사용하여 다른 오브젝트를 수동으로 복구해야 합니다. 이 명령은 IBM MQ 오브젝트를 다시 작성하기 위해 저널에서 입력 항목을 재실행합니다. IBM MQ 오브젝트가 손상되면, 유일하게 올바른 조치는 이를 삭제하고 이 방법을 통해 이를 다시 작성하는 것입니다. 그러나, 비지속 메시지가 이 방식으로 복구될 수 없다는 것에 주목하십시오.

체크포인트

체크포인트는 복구를 위해 알려진 지속적 시작점을 제공하기 위해 여러 번 수행됩니다.

체크포인트 프로세스 AMQALMPX는 다음 지점에서 체크포인트를 수행할 책임이 있습니다.

- 큐 관리자 시동(STRMQM).
- 큐 관리자 시스템 종료(ENDMQM).
- 마지막 체크포인트(기본 기간은 30분임) 이후 일정 기간 경과 및 이전 체크포인트 이후 최소 수(기본값은 100임)의 로그 레코드가 작성된 후.
- 많은 로그 레코드가 작성된 이후. 기본값은 10,000입니다.
- 저널 임계값 크기가 초과되고 새 저널 수신자가 자동으로 작성된 후.
- 전체 매체 이미지가 작성될 때:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>) DSPJRNDTA(*YES)
```

IBM MQ for IBM i 데이터의 백업

이 정보를 사용하여 각 큐 관리자에 대한 IBM MQ 백업의 두 가지 유형을 이해하십시오.

각 큐 관리자의 경우, 고려할 두 가지 유형의 IBM MQ 백업이 있습니다.

- 데이터 및 저널 백업.

  두 세트의 데이터 모두 지속적인지 확인하려면 큐 관리자를 종료한 후에만 이를 수행하십시오.
- 저널 백업.

  큐 관리자가 활성인 동안 이를 수행할 수 있습니다.

두 방법 모두의 경우, 큐 관리자 IFS 디렉토리 및 큐 관리자 라이브러리의 이름을 찾아야 합니다. IBM MQ 구성 파일(mqs.ini)에서 이를 찾을 수 있습니다. 자세한 정보는 QueueManager 스탠자를 참조하십시오.

다음 프로시저를 사용하여 두 가지 유형 모두의 백업을 수행하십시오.

특정 큐 관리자의 데이터 및 저널 백업

참고: 큐 관리자가 실행 중일 때 활동 중 보관(save-while-active) 요청을 사용하지 마십시오. 그러한 요청은 보류 중인 변경 사항을 가지는 모든 커미트 정의가 커미트되거나 롤백되지 않는 경우 완료될 수 없습니다. 큐 관리자가 활성일 때 이 명령이 사용되는 경우, 채널 연결이 정상적으로 종료될 수 없습니다. 항상 다음 프로시저를 사용하십시오.

1. 다음 명령을 사용하여 비어 있는 저널 수신자를 작성하십시오.

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. **RCDMQMIMG** 명령을 사용하여 모든 IBM MQ 오브젝트에 대해 MQM 이미지를 기록한 후 다음 명령을 사용하여 체크포인트를 강제 실행하십시오.

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 채널을 종료하고 큐 관리자가 실행 중이지 않은지 확인하십시오. 큐 관리자가 실행 중인 경우, **ENDMQM** 명령으로 중지하십시오.
4. 다음 명령을 실행하여 큐 관리자 라이브러리를 백업하십시오.

```
SAVLIB LIB(QMTEST)
```

5. 다음 명령을 실행하여 큐 관리자 IFS 디렉토리를 백업하십시오.

```
SAV DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/test'))
```

특정 큐 관리자의 저널 백업

모든 관련 정보가 저널에 유지되므로, 언젠가 전체 저장을 수행하는 한 부분 백업이 저널 수신자를 저장하여 수행될 수 있습니다. 이는 전체 백업의 시간 이후 모든 변경 사항을 레코딩하고 다음 명령을 실행하여 수행됩니다.

1. 다음 명령을 사용하여 비어 있는 저널 수신자를 작성하십시오.

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. **RCDMQMIMG** 명령을 사용하여 모든 IBM MQ 오브젝트에 대해 MQM 이미지를 기록한 후 다음 명령을 사용하여 체크포인트를 강제 실행하십시오.

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 다음 명령을 사용하여 저널 수신자를 저장하십시오.

```
SAVOBJ OBJ(AMQ*) LIB(QMTEST) OBJTYPE(*JRNRCV) .........
```

단순 백업 전략은 매주 IBM MQ 라이브러리의 전체 백업을 수행하는 것이고 매일 저널 백업을 수행하는 것입니다. 물론, 이는 엔터프라이즈에 대한 백업 전략을 설정하는 방법에 따라 다릅니다.

저널 관리

사용자의 백업 전략의 일부로, 사용자의 저널 수신자에 주의하십시오. 다양한 이유로 IBM MQ 라이브러리에서 저널 수신자를 제거하는 것이 유용합니다.

- 공간 해제(이는 모든 저널 수신자에 적용됨)
- 시작할 때 성능 개선(STRMQM)
- 오브젝트 재작성 성능 개선(RCRMQMOBJ)

저널 수신자를 삭제하기 전에 백업 사본을 가지고 있고 더 이상 저널 수신자가 필요하지 않다는 것에 주의해야 합니다.

저널 수신자는 복구 조작에 필요한 경우 복원에 사용 가능하면 저널에서 분리되고 저장된 후 큐 관리자 라이브러리에서 제거될 수 있습니다.

저널 관리의 개념이 196 페이지의 그림 34에 표시됩니다.

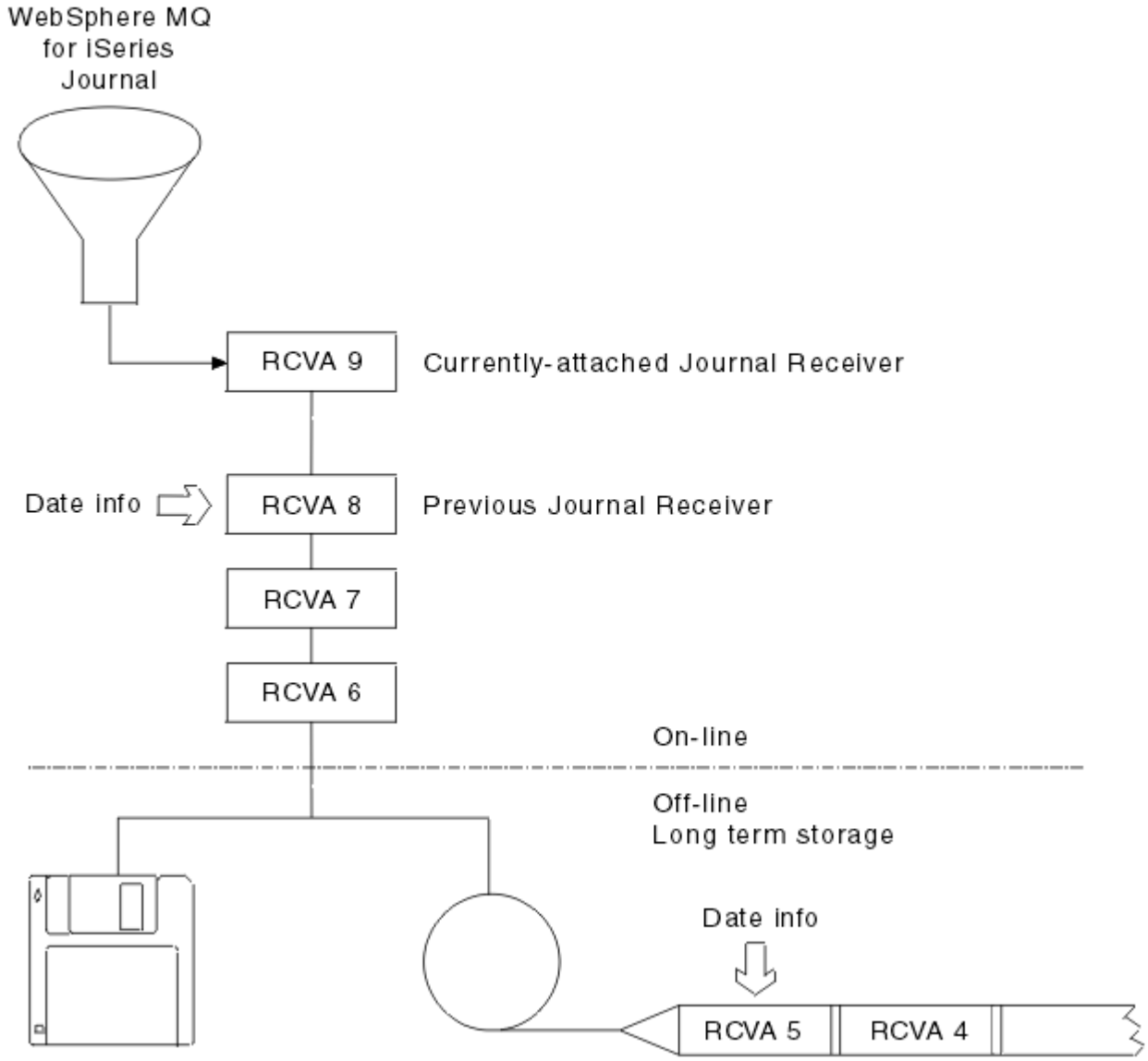


그림 *34. IBM MQ for IBM i* 저널링

백업된 저널 수신자가 큐 관리자 라이브러리에서 제거될 수 있을 때와 백업 자체가 제거될 수 있을 때를 판별하기 위해 IBM MQ가 저널에서 다시 뒤로 이동할 수 있는 범위를 아는 것이 중요합니다.

IBM MQ는 이 시간을 판별하는 데 도움을 주기 위해 큐 관리자 메시지 큐(큐 관리자 라이브러리의 QMQMMSG)에 두 개의 메시지를 발행합니다. 이러한 메시지는 시작될 때, 로컬 저널 수신자를 변경할 때, RCDMQIMG를 사용하여 체크포인트를 강제 실행할 때 발행됩니다. 두 개의 메시지는 다음과 같습니다.

AMQ7460
시동 복구 포인트. 이 메시지는 IBM MQ가 시동 복구 단계 이벤트에서 저널을 재실행하는 시동 입력 항목의 날짜 및 시간을 정의합니다. 이 레코드를 포함하는 저널 수신자를 IBM MQ 라이브러리에서 사용할 수 있는 경우, 이 메시지에는 레코드를 포함하는 저널 수신자의 이름도 포함됩니다.

AMQ7462
가장 오래된 매체 복원 입력 항목. 이 메시지는 해당 매체 이미지에서 오브젝트를 다시 작성하기 위해 사용할 가장 오래된 항목의 날짜 및 시간을 정의합니다.

식별된 저널 수신자는 요청되는 가장 오래된 것입니다. 더 오래된 작성 날짜를 가지는 다른 어떤 IBM MQ 저널 수신자도 더 이상 요구되지 않습니다. 별표(*)만 표시되면, 가장 오래된 저널 수신자를 판별하기 위해 표시되는 날짜에서 백업을 복원해야 합니다.

이러한 메시지가 로그될 때, IBM MQ는 하나의 입력 항목만 포함하는 큐 관리자 라이브러리에 사용자 공간 오브젝트를 작성합니다(시스템에서 유지되어야 하는 가장 오래된 저널 수신자의 이름). 이 사용자 공간은 AMQJRNINF라고 하고 데이터는 다음 형식으로 쓰여집니다.

```
JJJJJJJJJJLLLLLLLLLLYYYYMMDDHHMMSSmmm
```

설명:

JJJJJJJJJJ
IBM MQ가 여전히 필요로 하는 가장 오래된 수신자 이름입니다.

LLLLLLLLLL
저널 수신자 라이브러리 이름입니다.

YYYY
IBM MQ가 필요로 하는 가장 오래된 저널 입력 항목의 연도입니다.

MM
IBM MQ가 필요로 하는 가장 오래된 저널 입력 항목의 월입니다.

DD
IBM MQ가 필요로 하는 가장 오래된 저널 입력 항목의 일입니다.

HH
IBM MQ가 필요로 하는 가장 오래된 저널 입력 항목의 시간입니다.

SS
IBM MQ가 필요로 하는 가장 오래된 저널 입력 항목의 초입니다.

mmm
IBM MQ가 필요로 하는 가장 오래된 저널 입력 항목의 밀리초입니다.

가장 오래된 수신자가 시스템에서 삭제될 때, 이 사용자 공간에는 저널 수신자 이름에 별표(*)가 포함됩니다.

참고: 주기적으로 `RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES)`를 실행하면 IBM MQ의 시동 시간을 절약할 수 있고 복구를 위해 저장하고 복원해야 하는 로컬 저널 수신자 수를 감소시킬 수 있습니다.

IBM MQ for IBM i는 시동 또는 오브젝트를 다시 작성하기 위해 복구 단계를 수행하지 않는 경우 저널 수신자를 참조하지 않습니다. 요구하는 저널이 존재하지 않는다는 것을 발견하는 경우, 복구 단계를 완료하기 위해 필요한 저널 항목의 시간 및 날짜를 보고하는 큐 관리자 메시지 큐(QMQMMSG)에 AMQ7432 메시지를 발행합니다.

메시지가 발행되면, 복구 단계가 성공할 수 있도록 이 날짜 이후에 백업으로부터 분리된 모든 저널 수신자를 복원하십시오.

큐 관리자 라이브러리에서 사용 가능한 후속 저널 수신자 및 시동 항목을 포함하는 저널 수신자를 유지하십시오.

가장 오래된 `Media Recovery Entry`및 모든 후속 저널 수신자를 포함하는 저널 수신자를 항상 사용 가능하게 하고 큐 관리자 라이브러리에 있거나 백업하십시오.

체크포인트를 강제 실행할 때:

- AMQ7460에 이름 지정된 저널 수신자가 진행되지 않은 경우, 이는 커미트되거나 롤백 되어야 하는 미완료 작업 단위가 있다는 것을 나타냅니다.
- AMQ7462에 이름 지정된 저널 수신자가 진행되지 않은 경우, 이는 하나 이상의 손상된 오브젝트가 있다는 것을 나타냅니다.

전체 큐 관리자 복원(데이터 및 저널)

이 정보를 사용하여 백업 또는 원격 시스템에서 하나 이상의 큐 관리자를 복원하십시오.

백업에서 하나 이상의 IBM MQ 큐 관리자를 복원해야 하는 경우, 다음 단계를 수행하십시오.

1. IBM MQ 큐 관리자를 일시정지하십시오.
2. 가장 최근의 전체 백업 및 후속적으로 백업되는 저널 수신자로 구성되는 최근 백업 세트를 찾으십시오.

3. 다음 명령을 실행하여 전체 백업 시 해당 상태로 IBM MQ 데이터 라이브러리를 복원하기 위해 전체 백업에서 RSTLIB 조작을 수행하십시오.

```
RSTLIB LIB(QMQRLIB1) .........
RSTLIB LIB(QMQRLIB2) .........
```

   저널 수신자가 부분적으로 하나의 저널 백업에 저장되고 후속 백업에 완전히 저장되는 경우, 완전히 저장된 것만 복원하십시오. 시간 순서로 저널을 개별적으로 복원하십시오.

4. 다음 명령을 사용하여 IBM MQ IFS 디렉토리를 IFS 파일 시스템으로 복원하기 위해 RST 조작을 수행하십시오.

```
RST DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/testqm')) ...
```

5. 메시지 큐 관리자를 시작하십시오. 이는 전체 백업 이후 작성되는 모든 저널 레코드를 재실행하고 저널 백업 시 지속적 상태로 모든 IBM MQ 오브젝트를 복원합니다.

다른 시스템에서 전체 큐 관리자를 복원하려면 다음 프로시저를 사용하여 큐 관리자 라이브러리에서 모든 것을 복원하십시오. (샘플 큐 관리자 이름으로 TEST를 사용합니다.)

1. CRTMQM TEST
2. DLTLIB LIB(QMTEST)
3. RSTLIB SAVLIB(QMTEST) DEV(*SAVF) SAVF(QMGRLIBSAV)
4. 다음 IFS 파일을 삭제하십시오.

   /QIBM/UserData/mqm/qmgrs/TEST/QMQMCHKPT
   /QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMQMOBJCAT
   /QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMANAGER
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.AUTH.DATA.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CHANNEL.INITQ/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.COMMAND.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.REPOSITORY.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.TRANSMIT.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.PENDING.DATA.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.ADMIN.COMMAND.QUEUE/q

5. STRMQM TEST
6. RCRMQMOBJ OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(TEST)

특정 큐 관리자에 대한 저널 수신자 복원

저널 수신자를 복원하기 위한 다른 방법을 이해하기 위해 이 정보를 사용하십시오.

제거된 수신자가 후속 복구 기능에 다시 필요하면, 가장 공통적인 조치는 백업 저널 수신자를 큐 관리자 라이브러리로 복원하는 것입니다.

이는 단순 태스크이고, 저널 수신자가 표준 IBM i RSTOBJ 명령을 사용하여 복원되도록 요구합니다.

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .........
```

일련의 저널 수신자는 단일 수신자 보다 복원될 필요가 있을 수도 있습니다. 예를 들어, AMQA000007 은 IBM MQ 라이브러리에서 가장 오래된 수신자이며 AMQA000005 및 AMQA000006 둘 다 복원해야 합니다.

이 경우, 역시간 순서로 수신자를 개별적으로 복원하십시오. 이는 항상 필요하지만, 우수 사례는 아닙니다. 심각한 상황에서는 IBM i 명령 WRKJRNA 를 사용하여 복원된 저널 리시버를 저널과 연관시켜야 할 수도 있습니다.

저널을 복원할 때, 시스템은 저널 수신자 순서로 새 이름을 가지는 첨부된 저널 수신자를 자동으로 작성합니다. 그러나, 생성된 새 이름은 복원해야 하는 저널 수신자와 동일할 수도 있습니다. 매뉴얼 개입은 이 문제를 극복할 필요가 있습니다. 저널 수신자를 복원하기 전에 시퀀스와 새 저널에서 새 이름 저널 수신자를 작성합니다.

예를 들어, 저장된 저널 AMQAJRN과 다음 저널 수신자의 문제를 고려하십시오.

- AMQA000000
- AMQA100000
- AMQA200000
- AMQA300000
- AMQA400000
- AMQA500000
- AMQA600000
- AMQA700000
- AMQA800000
- AMQA900000

저널 AMQAJRN을 큐 관리자 라이브러리로 복구할 때, 시스템은 자동으로 저널 수신자 AMQA000000을 작성합니다. 이는 복원하려는 기존 저널 수신자(AMQA000000) 중 하나와의 수신자 충돌을 자동으로 생성하며, 이는 복원할 수 없습니다.

솔루션을 다음과 같습니다.

1. 다음 저널 수신자를 수동으로 작성하십시오(190 페이지의 『IBM MQ for IBM i 저널』 참조).

```
CRTJRNRCV JRNRCV(QMQRLIB/AMQA900001) THRESHOLD(XXXXX)
```

2. 저널 수신자로 저널을 수동으로 작성하십시오.

```
CRTJRN JRN(QMGRLIB/AMQAJRN) MNGRCV(*SYSTEM) +
JRNRCV(QMGRLIB/AMQA9000001) MSGQ(QMGRLIB/AMQAJRNMSG)
```

3. 로컬 저널 수신자 AMQA000000을 AMQA900000으로 복원하십시오.

다중 인스턴스 큐 관리자

다중 인스턴스 큐 관리자는 활성 서버가 실패하면 자동으로 대기 서버로 전환하여 가용성을 향상시킵니다. 활성 서버와 대기 서버는 동일한 큐 관리자의 다중 인스턴스로 동일한 큐 관리자 데이터를 공유합니다. 활성 인스턴스가 실패하면 큐 관리자가 해당 큐를 다시 빌드할 수 있도록 인계받는 대기 인스턴스로 해당 저널을 전송해야 합니다.

활성 큐 관리자 인스턴스가 실패하면 사용하는 저널을 인계받는 대기 인스턴스가 사용할 수 있도록 다중 인스턴스 큐 관리자를 실행하는 IBM i 시스템을 구성합니다. 활성 인스턴스의 저널을 인계받는 인스턴스가 사용할 수 있도록 사용자 고유의 구성 및 관리 태스크를 설계할 수 있습니다. 메시지를 잃지 않으려면 실패 시점에 대기 저널이 활성 저널과 일치하도록 설계해야 합니다. 일관성을 유지하는 후속 토픽의 예로 설명되는 두 가지 구성 중 하나에서 설계를 수정할 수 있습니다.

1. 활성 큐 관리자 인스턴스를 실행하는 시스템에서 대기 인스턴스를 실행하는 시스템으로 저널을 미러링합니다.
2. 활성 인스턴스를 실행하는 시스템에서 대기 인스턴스로 전송 가능한 독립 보조 기억장치 풀(IASP)에 저널을 배치합니다.

첫 번째 솔루션은 기본 ASP를 사용하므로 추가 하드웨어 또는 소프트웨어가 필요하지 않습니다. 두 번째 솔루션에서는 별도 가격의 IBM i 라이센스 제품 5761-SS1 옵션 41로 제공되는 IBM i 클러스터링 지원이 필요한 전환 가능 IASP가 필요합니다.

신뢰성 및 가용성

다중 인스턴스 큐 관리자의 목표는 애플리케이션 가용성 향상입니다. 기술 및 물리적 제한조건은 큐 관리자와 연속 조작을 백업하여 재해 복구 요구를 충족시키기 위해 다른 솔루션이 필요함을 의미합니다.

신뢰성과 가용성을 고려하여 구성하는 경우 많은 요인의 균형을 유지해야 하므로 다음과 같은 네 가지 설계점이 나타납니다.

재해 복구
모든 로컬 자산을 영구 삭제하는 중대 재해 후 복구를 위해 최적화됩니다.

IBM i의 재해 복구는 종종 IASP의 지리적 미러링을 기반으로 합니다.

백업
일반적으로 사용자 오류나 예측하지 못한 일부 기술적 문제점에 해당하는 국지적 실패 후 복구를 위해 최적화됩니다.

IBM MQ는 큐 관리자를 주기적으로 백업하기 위해 백업 큐 관리자를 제공합니다. 큐 관리자 저널의 비동기 복제를 사용하여 백업의 통화를 향상시킬 수도 있습니다.

가용성
서버 또는 디스크 고장과 같은 예측 불가능한 기술적 장애 이후 거의 중단되지 않는 서비스를 빠르게 제공할 수 있게 해주는 조작 복원을 위해 최적화됩니다.

복구는 일반적으로 분 단위로 측정되며 때때로 감지 시간이 복구 프로세스보다 오래 걸립니다. 다중 인스턴스 큐 관리자는 가용성을 고려한 구성을 지원합니다.

연속 조작
중단되지 않는 서비스 제공을 위해 최적화됩니다.

연속 조작 솔루션은 감지 문제점을 해결해야 하므로 거의 모든 경우 여러 시스템을 통해 동일한 작업을 제출하고 첫 번째 결과를 사용하거나 최소 두 가지 결과물을 비교(주요 고려 사항이 정확성인 경우)하는 작업이 포함됩니다.

다중 인스턴스 큐 관리자는 가용성을 고려한 구성을 지원합니다. 큐 관리자의 한 가지 인스턴스는 한 번에 하나씩 활성화됩니다. 대기 인스턴스로 전환하는 데는 시스템 구성, 로드, 조정 방법에 따라 10초에서 15초 이상이 소요됩니다.

다중 인스턴스 큐 관리자는 애플리케이션 프로그램이 큐 관리자 중단을 인식하지 않고도 계속 처리할 수 있는 다시 연결 가능한 IBM MQ MQI clients와 함께 사용되는 경우 거의 중단되지 않는 서비스로 보일 수 있습니다(자동 클라이언트 다시 연결 주제 참조).

고가용성 솔루션의 컴포넌트

강력한 큐 관리자 데이터용 네트워크 스토리지, 저널 복제 또는 강력한 큐 관리자 저널용 IASP 스토리지를 제공하여 다중 인스턴스 큐 관리자를 사용하거나 재시작 가능한 큐 관리자 서비스로 구성되는 애플리케이션의 다시 연결 가능한 클라이언트를 사용하여 고가용성 솔루션을 구성합니다.

다중 인스턴스 큐 관리자가 다른 서버에서 다른 큐 관리자 인스턴스 시작을 재개하여 큐 관리자 실패 감지에 반응합니다. 시작을 완료하려면 네트워크 스토리지의 공유 큐 관리자 데이터와 로컬 큐 관리자 저널의 해당 사본에 대한 액세스 권한이 인스턴스에 필요합니다.

고가용성 솔루션을 작성하려면 큐 관리자 데이터의 가용성, 로컬 큐 관리자 저널의 통화를 관리해야 하며 다시 연결 가능한 클라이언트 애플리케이션을 빌드하거나 애플리케이션을 큐 관리자가 재개될 때 자동으로 다시 시작되는 큐 관리자 서비스로 배치해야 합니다. 자동 클라이언트 다시 연결은 IBM MQ classes for Java에서 지원되지 않습니다.

큐 관리자 데이터

RAID 레벨 1 이상 디스크를 사용하여 공유되고 신뢰할 수 있는 고가용성 네트워크 스토리지에 큐 관리자 데이터를 배치합니다. 파일 시스템은 다중 인스턴스 큐 관리자의 공유 파일 시스템에 대한 요구 사항을 충족시켜야 합니다. 공유 파일 시스템 관련 요구 사항에 대한 자세한 정보는 공유 파일 시스템 요구 사항을 참조하십시오. 네트워크 파일 시스템 버전 4(NFS4)는 이러한 요구 사항을 충족시키는 프로토콜입니다.

큐 관리자 저널

또한 대기 인스턴스가 해당 큐 관리자 데이터를 일치 상태로 복원할 수 있도록 큐 관리자 인스턴스가 사용하는 IBM i 저널을 구성해야 합니다. 중단되지 않는 서비스의 경우 이는 저널을 활성 인스턴스가 실패한 시점의 상태로 복원해야 함을 의미합니다. 백업 또는 재해 복구 솔루션과 달리 이전 체크포인트로 저널을 복원하는 것으로는 충분하지 않습니다.

네트워크 스토리지의 여러 IBM i 시스템 간에 물리적으로 저널을 공유할 수 없습니다. 큐 관리자 저널을 실패 시점의 일치 상태로 복원하려면 실패 시점에 활성 큐 관리자 인스턴스와 로컬 상태였던 물리적 저널을 활성화된 새

인스턴스로 전송하거나 실행 중인 대기 인스턴스에서 저널 미러를 유지해야 합니다. 미러링된 저널은 실패 인스턴스에 속하는 로컬 저널과 정확히 일치하도록 유지된 원격 저널 복제본입니다.

다음 세 가지 구성은 다중 인스턴스 큐 관리자에 대한 저널 관리 방법을 설계하기 위한 시작점입니다.

1. 활성 인스턴스 ASP에서 대기 인스턴스 ASP로의 동기화된 저널 복제(저널 미러링) 사용
2. 큐 관리자 저널을 유지하도록 구성한 IASP를 활성 인스턴스에서 활성 인스턴스로 인계되는 대기 인스턴스로 전송
3. 동기화된 보조 IASP 미러 사용

IBM MQ IBM i CRTMQM 명령에서 큐 관리자 데이터를 iASP에 넣는 방법에 대한 자세한 정보는 ASP 옵션을 참조하십시오.

또한 IBM Documentation의 IBM i 정보에 있는 고가용성 및 관리자 > 고가용성을 참조하십시오.

애플리케이션

대기 큐 관리자가 재개될 때 큐 관리자에 자동으로 다시 연결되는 클라이언트를 빌드하려면 MQCONNX를 사용하여 큐 관리자로 애플리케이션을 연결하고 **MQCNO** 옵션 필드에서 MQCNO_RECONNECT_Q_MGR을 지정하십시오. 다시 연결 가능한 클라이언트를 사용하는 세 가지 샘플 프로그램을 보려면 고가용성 샘플 프로그램을, 복구용 클라이언트 애플리케이션 설계에 대한 정보는 애플리케이션 복구를 참조하십시오.

*NetServer*를 사용하여 큐 관리자 데이터의 네트워크 공유 작성

IBM i 서버에서 큐 관리자 데이터를 저장하기 위한 네트워크 공유를 작성합니다. 큐 관리자 인스턴스를 호스트하는 두 서버에서 네트워크 공유에 액세스하기 위한 연결을 설정합니다.

시작하기 전에

- 이 태스크에는 세 가지 IBM i 서버가 필요합니다. 네트워크 공유는 그 중 한 가지 서버인 GAMMA에 정의됩니다. 다른 두 서버인 ALPHA와 BETA는 GAMMA에 연결됩니다.
- 세 가지 서버에 모두 IBM MQ를 설치합니다.
- System i® Navigator를 설치합니다(System i Navigator 참조).

이 태스크 정보

- GAMMA에서 큐 관리자 디렉토리를 작성하고 사용자 프로파일 QMQM 및 QMQMADM에 올바른 소유권과 권한을 설정합니다. 디렉토리와 권한은 GAMMA에 IBM MQ 를 설치하여 쉽게 작성됩니다.
- System i Navigator를 사용하여 GAMMA의 큐 관리자 데이터 디렉토리에 대한 공유를 작성합니다.
- ALPHA 및 BETA에서 공유를 가리키는 디렉토리를 작성합니다.

프로시저

1. GAMMA에서 QMQM 사용자 프로파일을 소유자로 지정하고 QMQMADM을 기본 그룹으로 지정하여 큐 관리자 데이터를 호스트할 디렉토리를 작성하십시오.

   팁:

   올바른 권한으로 디렉토리를 빠르고 안전하게 작성할 수 있는 방법은 GAMMA에 IBM MQ를 설치하는 것입니다.

   나중에 GAMMA에서 IBM MQ를 실행하지 않으려면 IBM MQ를 설치 제거하십시오. 설치 제거한 후에는 소유자 QMQM 사용자 프로필 및 QMQMADM 기본 그룹과 함께 /QIBM/UserData/mqm/qmgrs 디렉토리가 GAMMA에 남아 있습니다.

   태스크는 GAMMA에서 공유용으로 /QIBM/UserData/mqm/qmgrs 디렉토리를 사용합니다.
2. System i Navigator **연결 추가** 마법사를 시작하고 GAMMA 시스템에 연결하십시오.
   a) Windows 데스크탑에서 **System i Navigator** 아이콘을 두 번 클릭하십시오.
   b) **예**를 클릭하여 연결을 작성하십시오.
   c) **연결 추가** 마법사의 지시사항에 따라 IBM i 시스템에서 GAMMA로의 연결을 작성하십시오.

GAMMA에 대한 연결이 **내 연결**에 추가됩니다.

3. GAMMA에 새 파일 공유를 추가하십시오.
   a) **System i Navigator** 창에서 My Connections/GAMMA/File Systems의 File Shares 폴더를 클릭하십시오.
   b) **내 태스크** 창에서 **IBM i NetServer 공유 관리**를 클릭하십시오.

      새 창, **IBM i NetServer - GAMMA**가 데스크탑에서 열리고 공유 오브젝트가 표시됩니다.
   c) Shared Objects 폴더를 마우스의 오른쪽 단추로 클릭하고 **파일** > **새로 작성** > **파일**을 클릭하십시오.

      새 창, **IBM i NetServer 파일 공유 - GAMMA**가 열립니다.
   d) 예를 들어, 공유 이름으로 WMQ를 지정하십시오.
   e) 액세스 제어를 Read/Write로 설정하십시오.
   f) 이전에 작성한 /QIBM/UserData/mqm/qmgrs 디렉토리로 이동하여 **경로 이름**을 선택하고 **확인**을 클릭하십시오.

      IBM i NetServer 파일 공유 - GAMMA 창이 닫히고 공유 오브젝트 창에 WMQ가 나열됩니다.
4. 공유 오브젝트 창에서 **WMQ**를 마우스 오른쪽 단추로 클릭하십시오. **파일** > **권한**을 클릭하십시오.

   /QIBM/UserData/mqm/qmgrs 오브젝트에 대한 **Qmgrs 권한 - GAMMA** 창이 열립니다.
   a) 아직 설정되지 않은 경우 QMQM에 대한 다음 권한을 확인하십시오.

      ```
      Read
      Write
      Execute
      Management
      Existence
      Alter
      Reference
      ```
   b) 아직 설정되지 않은 경우 QMQMADM에 대한 다음 권한을 확인하십시오.

      ```
      Read
      Write
      Execute
      Reference
      ```
   c) /QIBM/UserData/mqm/qmgrs에 대한 권한을 부여하려는 다른 사용자 프로파일을 추가하십시오.

      예를 들어, /QIBM/UserData/mqm/qmgrs에 대한 기본 사용자 프로파일(공용) Read 및 Execute 권한을 부여할 수 있습니다.
5. GAMMA에서 /QIBM/UserData/mqm/qmgrs에 대한 액세스 권한이 부여된 모든 사용자 프로파일이 GAMMA에 액세스하는 서버에서와 동일한 비밀번호를 사용하는지 확인하십시오.

   특히 공유에 액세스하려는 다른 서버의 QMQM 사용자 프로파일의 비밀번호가 GAMMA의 QMQM 사용자 프로파일 비밀번호와 같은지 확인하십시오.

   팁: 비밀번호를 설정하려면 System i Navigator에서 My Connections/GAMMA/Users and Groups 폴더를 클릭하십시오. 또는 **CHFUSRPRF** 및 **CHGPWD** 명령을 사용하십시오.

결과

공유를 사용하는 다른 서버에서 GAMMA에 액세스할 수 있는지 확인하십시오. 다른 태스크를 수행하는 경우 /QNTC/GAMMA/WMQ 경로를 사용하여 ALPHA 및 BETA에서 GAMMA에 액세스할 수 있는지 확인하십시오. /

QNTC/GAMMA 디렉토리가 ALPHA 또는 BETA에 없는 경우에는 디렉토리를 작성해야 합니다. NetServer 도메인에 따라, 디렉토리를 작성하기 전에 ALPHA 또는 BETA에 IPL을 수행해야 할 수 있습니다.

```
CRTDIR DIR('/QNTC/GAMMA')
```

ALPHA 또는 BETA에서 /QNTC/GAMMA/WMQ에 액세스할 수 있는 것으로 확인한 경우 CRTMQM MQMNAME('QM1') MQMDIRP('/QNTC/GAMMA/WMQ') 명령을 실행하면 GAMMA에서 /QIBM/UserData/mqm/qmgrs/QM1이 작성됩니다.

다음에 수행할 작업

212 페이지의 『저널 미러링 및 NetServer를 사용하여 다중 인스턴스 큐 관리자 작성』 또는 216 페이지의 『NetServer 및 저널 미러링을 사용하여 단일 인스턴스 큐 관리자를 다중 인스턴스 큐 관리자로 변환』 태스크의 단계에 따라 다중 인스턴스 큐 관리자를 작성하십시오.

장애 복구 성능

큐 관리자 인스턴스가 실패한 사실을 감지하고 대기 인스턴스에서 처리를 재개하는 데 소요되는 시간은 구성에 따라 수십 초에서 15분 이상까지 다양합니다. 고가용성 솔루션을 설계하고 테스트할 때는 성능을 중점적으로 고려해야 합니다.

다중 인스턴스 큐 관리자가 저널 복제를 사용하거나 IASP를 사용하도록 구성할지 여부를 결정하는 데 있어 고려해야 할 장점과 단점이 있습니다. 미러링을 위해서는 큐 관리자가 원격 저널에 동기식으로 쓰기를 수행해야 합니다. 하드웨어 관점에서는 성능에 영향을 주지 않아도 되지만 소프트웨어 관점에서는 로컬 저널에만 쓰는 것보다 원격 저널에 쓰는 경우의 경로 길이가 더 길므로 실행 큐 관리자의 성능이 어느 정도 저하되는 것으로 예상할 수 있습니다. 그러나 대기 큐 관리자가 인계하면, 실패 전에 활성 인스턴스가 유지하는 원격 저널로부터의 로컬 저널 동기화 지연이 IBM i가 IASP를 감지하여 큐 관리자의 대기 인스턴스를 실행하는 서버로 전송하는 데 소요되는 시간보다 일반적으로 더 짧습니다. IASP 전송 시간은 몇 초 내에 완료되지 않고 최대 10분에서 15분입니다. IASP 전송 시간은 IASP가 대기 시스템으로 전송될 때 상이해야 하는 오브젝트의 수, 액세스 경로의 크기 또는 병합되어야 하는 색인에 따라 다릅니다.

대기 큐 관리자가 인계하면, 실패 전에 활성 인스턴스가 유지하는 원격 저널로부터의 로컬 저널 동기화 지연이 IBM i가 독립 ASP를 감지하여 큐 관리자의 대기 인스턴스를 실행하는 서버로 전송하는 데 소요되는 시간보다 일반적으로 더 짧습니다. 독립 APS 전송 시간은 몇 초 내에 완료되지 않고 최대 10분에서 15분입니다. 독립 APS 전송 시간은 독립 APS가 대기 시스템으로 전송될 때 상이해야 하는 오브젝트의 수, 액세스 경로의 크기 또는 병합되어야 하는 색인에 따라 다릅니다.

그러나 저널 전송이 대기 인스턴스가 완전히 재개되는 데 소요되는 시간에 영향을 주는 유일한 요인은 아닙니다. 네트워크 파일 시스템이 대기 인스턴스가 시동을 계속 시도하도록 신호를 보내는 큐 관리자 데이터에 대한 잠금을 해제하는 데 소요되는 시간과 인스턴스가 메시지 처리를 다시 시작할 수 있도록 저널에서 큐를 복구하는 데 소요되는 시간 또한 고려해야 합니다. 이러한 기타 지연 소스가 모두 대기 인스턴스를 시작하는 데 소요되는 시간에 추가됩니다. 총 전환 시간은 다음 컴포넌트로 구성됩니다.

실패 감지 시간
NFS가 큐 관리자 데이터에 대한 잠금을 해제하고 대기 인스턴스가 시동 프로세스를 계속 진행하는 데 소요되는 시간

전송 시간
HA 클러스터의 경우 IBM i가 활성 인스턴스를 호스트하는 시스템에서 대기 인스턴스로 IASP를 전송하는 데 소요되는 시간과, 저널 복제의 경우 대기 인스턴스의 로컬 저널을 원격 복제본의 데이터로 업데이트하는 데 소요되는 시간

재시작 시간
새로 활성화된 큐 관리자 인스턴스가 복원된 저널의 최신 체크포인트에서 큐를 재빌드하고 메시지 처리를 재개하는 데 소요되는 시간

참고:

인계된 대기 인스턴스가 이전 활성 인스턴스에 동시에 복제되도록 구성되는 경우에는 시동이 지연될 수 있습니다. 원격 저널이 이전 활성 인스턴스를 호스트하는 서버에 있고 해당 서버가 실패한 경우에는 새로 활성화된 인스턴스가 원격 저널에 복제되지 못할 수 있습니다.

동기 응답을 대기하는 기본 시간은 1분입니다. 복제 제한시간이 초과되기 전에 최대 지연을 구성할 수 있습니다. 또는 실패한 활성 인스턴스에 대한 비동기 복제를 사용하여 대기 인스턴스가 시작되도록 구성할 수 있습니다. 실패한 인스턴스가 대기 인스턴스에서 다시 실행될 때 나중에 동기 복제로 전환합니다. 동기 독립 ASP 미러링을 사용하는 경우에도 동일한 고려사항이 적용됩니다.

전체 장애 복구 시간을 평가하는 데 도움을 줄 수 있도록, 또한 사용할 구성 접근 방식을 결정하는 데 영향을 줄 수 있도록 이 컴포넌트의 기준선을 별도로 측정할 수 있습니다. 가장 적합한 구성을 결정하려면 또한 동일한 서버에서 다른 애플리케이션이 장애를 복구하는 방법과 이미 IASP를 사용하는 백업 또는 재해 복구 프로세스가 있는지 여부를 고려해야 합니다.

다음과 같이 클러스터 구성을 조정하여 IASP 전송 시간을 줄일 수 있습니다.

1. 가변 프로세스에서 UID 및 GID를 변경하지 않아도 되도록 클러스터 내 시스템의 사용자 프로파일이 동일한 GID 및 UID를 갖습니다.
2. 디스크 풀 그룹에 대한 교차 참조 테이블을 작성하기 위해 병합되어야 하므로 시스템 및 기본 사용자 디스크 풀에서 데이터베이스 오브젝트 수를 최소화합니다.
3. 추가 성능 팁은 IBM Redbook, *PowerHA® for IBM i* 구현, *SG24-7405*를 참조하십시오.

기본 ASP, 저널 미러링, 소규모 구성을 사용하는 구성은 수십초 단위로 전환되어야 합니다.

IBM i 클러스팅 기능과 *IBM MQ* 클러스팅 결합의 개요

IBM i에서 IBM MQ를 실행하고 IBM i 클러스터링 기능을 활용하면 IBM MQ 클러스터링만 사용하는 경우보다 훨씬 더 포괄적인 고가용성 솔루션을 제공할 수 있습니다.

이 기능을 사용하려면 다음을 설정해야 합니다.

1. IBM i 시스템의 클러스터. 204 페이지의 『IBM i 클러스터』의 내용을 참조
2. 독립 보조 스토리지 풀(IASP). 여기로 큐 관리자를 이동시키십시오. 204 페이지의 『독립 보조 스토리지 풀(IASP)』의 내용을 참조
3. 클러스터 자원 그룹(CRG). 205 페이지의 『디바이스 클러스터 자원 그룹』의 내용 참조. 여기에서 다음을 정의할 수 있습니다.
   - 복구 도메인
   - IASP
   - 엑시트 프로그램. 205 페이지의 『디바이스 CRG 엑시트 프로그램』의 내용 참조

IBM i 클러스터

IBM i 클러스터는 로컬에서 서로 링크된 IBM i 컴퓨터 또는 파티션인 인스턴스의 콜렉션입니다.

이 그룹화의 목적은 각 인스턴스를 백업할 수 있게 하여 단일 장애점을 제거하고 애플리케이션과 데이터의 탄력성을 높이기 위한 것입니다. 클러스터를 작성한 경우, 클러스터의 애플리케이션, 데이터 및 디바이스를 관리하도록 다양한 클러스터 자원 그룹(CRG) 유형을 구성할 수 있습니다.

자세한 정보는 클러스터 작성 및 클러스터 작성(CRTCLU) 명령을 참조하십시오.

독립 보조 스토리지 풀(IASP)

IASP는 단일 레벨 스토리지의 확장으로 기능하는 사용자 ASP의 유형입니다. 시스템 스토리지에서 독립되어 있으므로 시스템 IPL 없이 쉽게 조작할 수 있는 스토리지 부분입니다.

IASP는 다른 운영 체제 인스턴스로 쉽게 전환되거나 다른 운영 체제 인스턴스의 대상 IASP로 복제될 수 있습니다. 두 방법 모두 인스턴스 간 IASP를 전환하는 데 사용할 수 있습니다.

- 첫 번째 방법에는 클러스터의 모든 컴퓨터 및 IASP를 포함하는 전환 가능 디스크 타워가 고속 링크(HSL) 루프를 사용하여 연결되어야 합니다.
- 두 번째 방법에는 입출력 프로세서(IOS)가 두 파티션 사이에 전환될 수 있는 IBM i 컴퓨터에서 운영 체제 인스턴스가 파티셔닝되어야 합니다. IASP를 복제하는 데 특별한 하드웨어가 필요하지 않습니다. 이 복제는 네트워크에서 TCP/IP를 통해 수행됩니다.

자세한 정보는 디바이스 ASP 구성(CFGDEVASP) 명령을 참조하십시오.

디바이스 클러스터 자원 그룹

여러 개의 클러스터 자원 그룹(CRG) 유형이 있습니다. 사용 가능한 여러 CRG 유형에 대한 자세한 정보는 클러스터 자원 그룹을 참조하십시오.

이 주제는 디바이스 CRG를 집중 조명합니다. 디바이스 CRG는 다음과 같습니다.

- 독립 보조 스토리지 풀(IASP)과 같은 디바이스 자원을 기술하고 관리합니다.
- 클러스터 노드의 복구 도메인을 정의합니다.
- 디바이스를 지정합니다. 그리고
- 클러스터 이벤트를 처리할 엑시트 프로그램을 지정합니다.

복구 도메인은 어떤 클러스터 노드를 기본 노드로 간주할지를 나타냅니다. 나머지 노드는 백업 노드로 간주됩니다. 백업 노드는 또한 복구 도메인에서 순서가 지정되는데 복구 도메인에 얼마나 많는 노드가 있는지에 따라 첫 번째 백업, 두 번째 백업 노드 등으로 지정됩니다.

기본 노드에 장애가 발생하면 엑시트 프로그램이 복구 도메인의 모든 노드에서 실행됩니다. 첫 번째 백업 노드에서 실행되는 엑시트 프로그램은 이 노드를 새로운 기본 노드로 만들기 위해 필요한 초기화를 작성할 수 있습니다.

자세한 정보는 디바이스 CRG 작성 및 클러스터 자원 그룹 작성(CRTCRG) 명령을 참조하십시오.

디바이스 CRG 엑시트 프로그램

장애 복구 또는 전환 이벤트와 같이 복구 도메인이 정의한 노드 중 하나에서 이벤트가 발생하면 운영 체제 클러스터 자원 서비스가 디바이스 CRG 엑시트 프로그램을 호출합니다.

장애 복구 이벤트는 클러스터 기본 노드가 실패하고 CRG가 관리하는 모든 자원으로 전환된 경우 발생하며 특정 CRG가 기본 노드에서 백업 노드로 수동으로 전환될 때 전환 이벤트가 발생합니다.

어느 쪽이든 엑시트 프로그램은 이전의 기본 노드에서 실행 중이었던 모든 프로그램의 초기화와 시작을 담당하며 이는 첫 번째 백업 노드를 새 기본 노드로 변환합니다.

예를 들어 IBM MQ에서 엑시트 프로그램은 IBM MQ 서브시스템(QMQM) 및 큐 관리자 시작을 담당해야 합니다. 큐 관리자는 트리거 모니터 같은 자동으로 리스너와 서비스를 시작하도록 구성되어야 합니다.

전환 가능한 IASP 구성

IBM MQ는 IBM i의 클러스터링 기능을 활용하도록 설정될 수 있습니다. 이를 수행하려면:

1. 데이터 센터 시스템 간에 IBM i 클러스터를 작성하십시오.
2. IASP로 큐 관리자를 이동하십시오.

   206 페이지의 『독립 보조 스토리지 풀로 큐 관리자 이동 및 풀에서 큐 관리자 제거』는 이 조작을 수행하는 데 도움이 되는 일부 샘플 코드가 포함되어 있습니다.
3. 복구 도메인, IASP, 엑시트 프로그램을 정의하는 CRG를 작성해야 합니다.

   205 페이지의 『디바이스 클러스터 자원 그룹 구성』에는 이 조작을 수행하는 데 도움이 되는 일부 샘플 코드가 포함되어 있습니다.

관련 개념

224 페이지의 『독립 ASP 및 고가용성』
독립 ASP를 사용하여 서버 간에 애플리케이션과 데이터를 이동할 수 있습니다. 독립 ASP의 유연성은 ASP가 일부 IBM i 고가용성 솔루션의 기반임을 의미합니다. 큐 관리자 저널에 ASP 또는 독립 ASP를 사용할지 여부를 고려할 때는 독립 ASP를 기반으로 하는 다른 고가용성 구성을 고려해야 합니다.

디바이스 클러스터 자원 그룹 구성
디바이스 클러스터 자원 그룹(CRG)을 설정하는 예제 프로그램입니다.

이 태스크 정보

다음 예에서 다음 사항에 유의하십시오.

- [PRIMARY SITE NAME] 및 [BACKUP SITE NAME]은 8자 이하의 두 개별 문자열이 될 수 있습니다.

- [PRIMARY IP] 및 [BACKUP IP]는 미러링에 사용될 수 있는 IP입니다.

프로시저

1. 클러스터의 이름을 식별하십시오.
2. CRG 엑시트 프로그램 이름 및 라이브러리를 식별하십시오.
3. 이 CRG로 정의할 기본 노드 및 백업 노드의 이름을 판별하십시오.
4. 이 CRG로 관리할 IASP를 식별하고 기본 노드 아래에서 작성되었는지 확인하십시오.
5. 다음 명령을 사용하여 백업 노드에 디바이스 설명을 작성하십시오.

```
 CRTDEVASP DEVD([IASP NAME]) RSRCNAME([IASP NAME])
```

6. 다음 명령을 사용하여 인계 IP 주소를 모든 노드에 추가하십시오.

```
ADDTCPIFC INTNETADR(' [TAKEOVER IP]') LIND([LINE DESC])
SUBNETMASK('[SUBNET MASK]') AUTOSTART(*NO)
```

7. 다음 명령을 사용하여 기본 노드에서만 인계 IP 주소를 시작하십시오.

```
STRTCPIFC INTNETADR('[TAKEOVER IP')
```

8. 옵션: IASP가 전환 가능하면 다음 명령을 호출하십시오.

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG( [CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT
NAME])
USRPRF([EXIT PROFILE]) RCYDMN(( [PRIMARY NODE] *PRIMARY) ([BACKUP NAME] *BACKUP))
EXITPGMFMT(EXTP0200) CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]')
```

9. 옵션: IASP를 미러링하려면 다음 명령을 호출하십시오.

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG([CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT NAME])
USRPRF([EXIT PROFILE]) RCYDMN(([PRIMARY NODE] *PRIMARY *LAST [PRIMARY SITE NAME] ('[PRIMARY
IP]'))
[BACKUP NAME] *BACKUP *LAST [BACKUP SITE NAME] ('[BACKUP IP]'))) EXITPGMFMT(EXTP0200)
CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]'))
```

독립 보조 스토리지 풀로 큐 관리자 이동 및 풀에서 큐 관리자 제거
큐 관리자를 독립 보조 스토리지 풀(IASP)로 이동하기 위한 예제 프로그램 및 IASP에서 큐 관리자를 제거하는 명령입니다.

이 태스크 정보

다음 예에서 다음 사항에 유의하십시오.

- [MANAGER NAME]은 큐 관리자의 이름입니다.
- [IASP NAME]은 IASP의 이름입니다.
- [MANAGER LIBRARY]는 큐 관리자 라이브러리의 이름입니다.
- [MANAGER DIRECTORY]는 큐 관리자 디렉토리의 이름입니다.

프로시저

1. 기본 노드 및 백업 노드를 식별하십시오.
2. 기본 노드에서 다음 프로시저를 수행하십시오.
   a) 큐 관리자가 종료되었는지 확인하십시오.
   b) 다음 명령을 사용하여 IASP가 `vary on`인지 확인하십시오.

```
VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)
```

   c) IASP 아래에 큐 관리자 디렉토리를 작성하십시오.

다음과 같이 IASP 이름의 루트 아래에 있는 디렉토리일 수 있습니다.

```
QSH CMD('mkdir -p /[IASP_NAME]/QIBM/UserData/mqm/qmgrs/')
```

d) 다음 명령을 사용하여 방금 IASP 아래에 작성된 큐 관리자 디렉토리로 관리자의 IFS 오브젝트를 이동하십시오.

```
QSH CMD('mv /QIBM/UserData/mqm/qmgrs/[MANAGER NAME]
/[IASP NAME]/QIBM/UserData/mqm/qmgrs')
```

e) 다음 명령을 사용하여 MGRLIB 이름의 임시 저장 파일을 작성하십시오.

```
CRTSAVF QGPL/MGRLIB
```

f) 다음 명령을 사용하여 큐 관리자 라이브러리를 MGRLIB 저장 파일로 저장하십시오.

```
SAVLIB LIB([MANGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
```

g) 다음 명령을 사용하여 큐 관리자 라이브러리를 삭제하고 모든 조회 메시지를 무시하십시오.

```
DLTLIB [MANAGER LIBRARY]
```

h) 다음 명령을 사용하여 큐 관리자 라이브러리를 IASP로 복원하십시오.

```
RSTLIB SAVLIB([MANAGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
RSTASPDEV([IASP NAME])
```

i) 다음 명령을 사용하여 임시 저장 파일을 삭제하십시오.

```
 DLTF FILE(QGPL/MGRLIB)
```

j) 다음 명령을 사용하여 IASP 아래에 큐 관리자 IFS 오브젝트에 대한 기호 링크를 작성하십시오.

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

k) 다음 명령을 사용하여 IASP에 첨부하십시오.

```
SETASPGRP [IASP NAME]
```

l) 다음 명령을 사용하여 큐 관리자를 시작하십시오.

```
STRMQM [MANAGER NAME]
```

3. 백업 노드에서 다음 프로시저를 수행하십시오.

a) 다음 명령을 사용하여 임시 큐 관리자 디렉토리를 작성하십시오.

```
QSH CMD('mkdir -p /[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

b) 다음 명령을 사용하여 큐 관리자 임시 디렉토리에 대한 기호 링크를 작성하십시오.

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

c) 다음 명령을 사용하여 임시 디렉토리를 삭제하십시오.

```
QSH CMD('rm -r /[IASP NAME]')
```

d) /QIBM/UserData/mqm/mqs.ini 파일의 끝에 다음을 추가하십시오.

```
QueueManager:
Name=[MANAGER NAME]
Prefix=/QIBM/UserData/mqm
```

```
Library=[MANAGER LIBRARY]
Directory=[MANAGER DIRECTORY]
```

4. IASP에서 큐 관리자를 제거하려면 다음 명령을 실행하십시오.
   a) VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)
   b) SETASPGRP [IASP NAME]
   c) ENDMQM [MANAGER NAME]
   d) DLTMQM [MANAGER NAME]

*ASP*의 미러링된 저널 구성

미러링된 저널 간 동기 복제를 사용하여 강력한 다중 인스턴스 큐 관리자를 구성합니다.

미러링된 큐 관리자 구성은 기본 또는 독립 보조 기억장치 풀(ASP)에서 작성되는 저널을 사용합니다.

IBM i의 경우에는 큐 관리자 데이터가 저널과 파일 시스템에 기록됩니다. 저널에는 큐 관리자 데이터의 마스터 사본이 포함됩니다. 저널은 동기 또는 비동기 저널 복제를 사용하여 시스템 간에 공유됩니다. 큐 관리자 인스턴스를 재시작하려면 로컬 저널과 원격 저널의 믹스가 필요합니다. 큐 관리자 재시작은 서버의 로컬 및 원격 저널 믹스의 저널 레코드와 공유 네트워크 파일 시스템의 큐 관리자 데이터를 읽습니다. 파일 시스템의 데이터로 인해 큐 관리자가 빨리 재시작됩니다. 파일 시스템과 저널 사이의 재동기화 지점을 표시하는 체크포인트가 파일 시스템에 저장됩니다. 일반 큐 관리자 재시작에는 체크포인트 이전에 저장된 저널 레코드가 필요하지 않습니다. 그러나 파일 시스템의 데이터가 최신 상태가 아닐 수 있으므로 체크포인트 이후 저널 레코드를 사용하여 큐 관리자 재시작을 완료합니다. 인스턴스에 첨부된 저널의 데이터는 최신 상태로 유지되므로 재시작이 성공적으로 완료될 수 있습니다.

그러나 대기 서버의 원격 저널이 비동기식으로 복제되고 동기화되기 전에 실패한 경우에는 저널 레코드도 최신 상태가 아닐 수 있습니다. 동기화되지 않은 원격 저널을 사용하여 큐 관리자를 재시작하려는 경우, 대기 큐 관리자 인스턴스가 활성 인스턴스가 실패하기 전에 삭제된 메시지를 재처리하거나 활성 인스턴스가 실패하기 전에 수신된 메시지를 처리하지 않을 수 있습니다.

드물지만 또 다른 경우 파일 시스템에는 최신 체크포인트 레코드가 포함되고 대기 서버에서 동기화되지 않은 원격 저널에는 포함되지 않는 경우입니다. 이러한 경우 큐 관리자가 자동으로 재시작되지 않습니다. 원격 저널이 동기화될 때까지 대기하거나 파일 시스템에서 대기 큐 관리자를 콜드 스타트할 수 있습니다. 이 경우 파일 시스템에 원격 저널보다 최신 상태의 큐 관리자 데이터 체크포인트가 포함되더라도 활성 인스턴스가 실패하기 전에 처리된 모든 메시지가 포함되지 않을 수 있습니다. 저널과 동기화되지 않은 콜드 리스타트 이후에 일부 메시지는 재처리되고 일부는 처리되지 않을 수 있습니다.

다중 인스턴스 큐 관리자를 사용하는 경우 활성 상태의 큐 관리자 인스턴스와 대기 인스턴스를 제어하기 위해 파일 시스템을 사용할 수도 있습니다. 활성 인스턴스는 큐 관리자 데이터에 대한 잠금을 획득합니다. 대기 인스턴스는 잠금 획득을 대기하며 획득 시 활성 인스턴스가 됩니다. 활성 인스턴스가 정상적으로 종료되면 잠금을 해제합니다. 파일 시스템이 활성 인스턴스가 실패했거나 파일 시스템에 액세스할 수 없는 것으로 감지하면 파일 시스템이 잠금을 해제합니다. 파일 시스템은 실패 감지를 위한 요구사항을 충족시켜야 합니다. 공유 파일 시스템에 대한 요구사항을 참조하십시오.

IBM i에서 다중 인스턴스 큐 관리자의 아키텍처는 서버 또는 큐 관리자 실패 후 자동 재시작을 제공합니다. 또한 큐 관리자 데이터가 저장되는 파일 시스템이 실패한 후 큐 관리자 데이터 복원을 지원합니다.

209 페이지의 그림 35에서 ALPHA가 실패하는 경우 미러링된 저널을 사용하여 BETA에서 QM1을 수동으로 재시작할 수 있습니다. QM1에 다중 인스턴스 큐 관리자 기능을 추가하면 ALPHA의 활성 인스턴스가 실패하는 경우 QM1의 대기 인스턴스가 BETA에서 자동으로 재개됩니다. QM1은 또한 QM1의 활성 인스턴스뿐 아니라 실패한 ALPHA 서버인 경우 자동으로 재개될 수 있습니다. BETA가 활성 큐 관리자 인스턴스의 호스트가 되면 대기 인스턴스가 ALPHA에서 시작될 수 있습니다.

209 페이지의 그림 35은 NetServer를 사용하여 큐 관리자 데이터를 저장하는 큐 관리자의 두 인스턴스 간에 저널을 미러링하는 구성을 보여줍니다. 보다 많은 저널과 그에 따라 보다 많은 인스턴스를 포함하도록 패턴을 확장할 수 있습니다. 190 페이지의 『IBM MQ for IBM i 저널』 주제에서 설명하는 저널 이름 지정 규칙을 따르십시오. 현재 큐 관리자의 실행 인스턴스 수는 두 개(하나의 활성 인스턴스와 하나의 대기 인스턴스)로 제한됩니다.

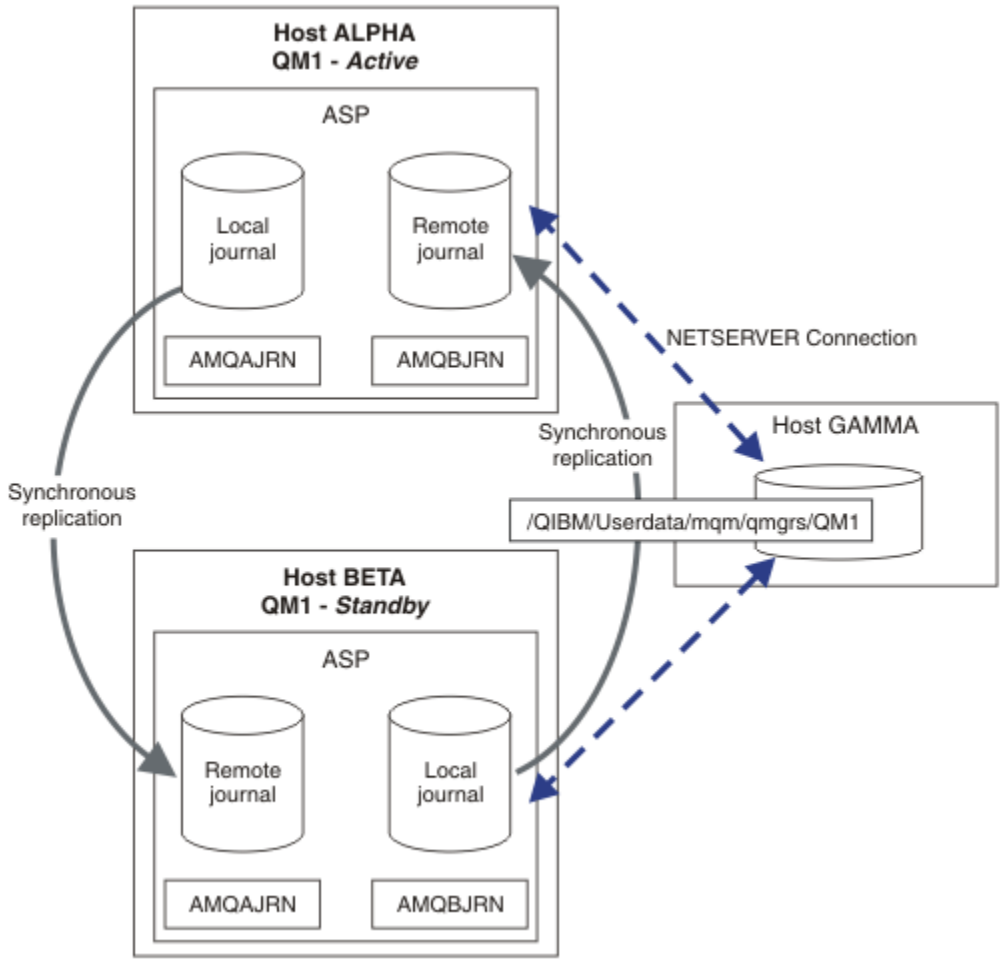


그림 *35.* 큐 관리자 저널 미러링

호스트 ALPHA의 QM1에 대한 로컬 저널을 AMQAJRN(보다 정확하게는 QMQM1/AMQAJRN)이라고 하며 BETA의 저널은 QMQM1/AMQBJRN입니다. 각 로컬 저널은 큐 관리자의 다른 모든 인스턴스에 있는 원격 저널에 복제됩니다. 큐 관리자가 두 개 인스턴스로 구성되면 로컬 저널이 하나의 원격 저널에 복제됩니다.

*SYNC 또는 *ASYNC 원격 저널 복제

IBM i 저널은 동기(*SYNC) 또는 비동기(*ASYNC) 저널링을 사용하여 미러링됩니다. 원격 저널 관리를 참조하십시오.

209 페이지의 그림 35 의 복제 모드는 *ASYNC가 아니고 *SYNC입니다. *ASYNC 는 더 빠릅니다. 그러나 원격 저널 상태가 *ASYNCPEND일 때 실패가 발생하면 로컬 및 원격 저널이 일관되지 않습니다. 원격 저널이 로컬 저널을 따라잡아야 합니다. *SYNC를 선택하면 완료된 쓰기가 필요한 호출로부터 리턴되기 전에 로컬 시스템이 원격 저널을 대기합니다. 로컬 및 원격 저널은 일반적으로 서로 일치 상태를 유지합니다. *SYNC 조작이 지정된 시간보다 오래 걸리는 경우에만[1]원격 저널링이 비활성화되면 저널이 동기화되지 않습니다. 저널 메시지 큐와 QSYSOPR에 오류가 로깅됩니다. 큐 관리자가 이 메시지를 감지하면 큐 관리자 오류 로그에 오류를 기록하고 큐 관리자 저널의 원격 복제를 비활성화합니다. 활성 큐 관리자 인스턴스는 이 저널에 대한 원격 저널링 없이 재개됩니다. 원격 서버를 다시 사용할 수 없게 되면 동기 원격 저널 복제를 수동으로 재활성화해야 합니다. 그러면 저널이 재동기화됩니다.

209 페이지의 그림 35에 나와 있는 *SYNC / *SYNC 구성 관련 문제점은 BETA의 대기 큐 관리자 인스턴스가 제어를 수용하는 방식입니다. BETA의 큐 관리자 인스턴스가 첫 번째 지속 메시지를 쓰자마자 ALPHA에서 원격 저널 업데이트를 시도합니다. ALPHA에서 BETA로 전달되는 제어의 원인이 ALPHA 실패이고 ALPHA가 계속 작동

[1] 지정된 시간은 IBM i 버전 5에서 60초이고 IBM i 6.1 이상에서는 1-3600초 범위입니다.

중지 상태이면 ALPHA에 대한 원격 저널링이 실패합니다. BETA는 ALPHA가 응답할 때까지 대기한 다음 원격 저널링을 비활성화하며 로컬 저널링만으로 메시지 처리를 재개합니다. BETA는 ALPHA가 작동 중지 상태임을 감지하는 동안 대기해야 하므로 일정 기간 동안 비활성화됩니다.

원격 저널링을 *SYNC 또는 *ASYNC로 설정할지 선택하는 것은 상호 보완적입니다. 210 페이지의 표 14에는 한 쌍의 큐 관리자 간에 *SYNC 및 *ASYNC 저널링을 사용하는 데 따른 상호 보완성이 요약 설명되어 있습니다.

표 *14.* 원격 저널링 옵션

활성	대기	***SYNC**	***ASYNC**
***SYNC**		1. 지속적 전환 및 장애 복구 2. 대기 인스턴스가 장애 복구 직후 재개되지 않습니다. 3. 원격 저널링은 항상 사용 가능해야 합니다. 4. 큐 관리자 성능은 원격 저널링에 따라 다릅니다.	1. 지속적 전환 및 장애 복구 2. 원격 저널링은 대기 서버를 사용할 수 있을 때 *SYNC로 전환되어야 합니다. 3. 재시작된 후에는 원격 저널링을 계속 사용할 수 있어야 합니다. 4. 큐 관리자 성능은 원격 저널링에 따라 다릅니다.
***ASYNC**		1. 바람직한 조합이 아님	1. 장애 복구 또는 전환 후 일부 메시지가 손실 또는 복제될 수 있습니다. 2. 대기 인스턴스가 항상 사용 가능한 상태가 아니어도 활성 인스턴스가 지연 없이 계속 진행됩니다. 3. 성능이 원격 저널링에 따라 영향을 받지 않습니다.

***SYNC / *SYNC**

활성 큐 관리자 인스턴스는 *SYNC 저널링을 사용하므로 대기 큐 관리자 인스턴스가 시작되면 즉시 *SYNC 저널링을 사용하려고 시도합니다.

1. 원격 저널은 트랜잭션 관점에서 활성 큐 관리자의 로컬 저널과 일치합니다. 큐 관리자가 대기 인스턴스로 전환되면 즉시 재개될 수 있습니다. 대기 인스턴스는 일반적으로 메시지 손실 또는 복제 없이 재개됩니다. 메시지는 마지막 체크포인트 이후 원격 저널링이 실패한 경우에만 손실 또는 복제되므로 이전에 활성화된 큐 관리자는 재시작할 수 없습니다.
2. 큐 관리자가 대기 인스턴스로 장애를 복구하면 즉시 시작하지 못할 수 있습니다. 대기 큐 관리자 인스턴스는 *SYNC 저널링으로 활성화됩니다. 장애 복구의 원인으로 인해 대기 인스턴스를 호스팅하는 서버에 대한 원격 저널링을 방해할 수 있습니다. 큐 관리자는 문제점이 감지될 때까지 대기한 후 지속 메시지를 처리합니다.저널 메시지 큐와 QSYSOPR에 오류가 로깅됩니다. 큐 관리자가 이 메시지를 감지하면 큐 관리자 오류 로그에 오류를 기록하고 큐 관리자 저널의 원격 복제를 비활성화합니다. 활성 큐 관리자 인스턴스는 이 저널에 대한 원격 저널링 없이 재개됩니다. 원격 서버를 다시 사용할 수 없게 되면 동기 원격 저널 복제를 수동으로 재활성화해야 합니다. 그러면 저널이 재동기화됩니다.
3. 원격 저널을 유지하려면 원격 저널이 복제되는 서버가 항상 사용 가능해야 합니다. 원격 저널은 일반적으로 대기 큐 관리자를 호스팅하는 동일한 서버에 복제됩니다. 서버는 사용하지 못하게 될 수 있습니다.저널 메시지 큐와 QSYSOPR에 오류가 로깅됩니다. 큐 관리자가 이 메시지를 감지하면 큐 관리자 오류 로그에 오류를 기록하고 큐 관리자 저널의 원격 복제를 비활성화합니다. 활성 큐 관리자 인스턴스는 이 저널에 대한 원격 저널링 없이 재개됩니다. 원격 서버를 다시 사용할 수 없게 되면 동기 원격 저널 복제를 수동으로 재활성화해야 합니다. 그러면 저널이 재동기화됩니다.
4. 원격 저널링은 로컬 저널링보다 느리며 서버가 먼 거리에 분산되어 있는 경우에는 속도가 더 느립니다. 큐 관리자는 원격 저널링을 대기해야 하므로 큐 관리자 성능이 저하됩니다.

한 쌍의 서버 간 *SYNC / *SYNC 구성은 장애 복구 후 대기 인스턴스 재개가 지연되는 단점이 있습니다. *SYNC / *ASYNC 구성에는 이러한 문제점이 없습니다.

*SYNC / *SYNC는 원격 저널을 사용할 수 있는 한 전환 또는 장애 복구 후 메시지가 손실되지 않는 것으로 보장하지 않습니다. 장애 복구 또는 전환 후 메시지 손실 위험을 줄이려면 두 가지 방법 중 하나를 선택할 수 있

습니다. 원격 저널이 비활성화되는 경우 활성 인스턴스를 중지하거나 여러 서버에서 원격 저널을 작성하면 됩니다.

***SYNC / *ASYNC**
활성 큐 관리자 인스턴스는 *SYNC 저널링을 사용하므로 대기 큐 관리자 인스턴스가 시작되면 *ASYNC 저널링을 사용합니다. 새 대기 인스턴스를 호스팅하는 서버를 사용할 수 있게 된 직후 시스템 운영자가 활성 인스턴스의 원격 저널을 *SYNC로 전환해야 합니다. 운영자가 원격 저널링을 *ASYNC에서 *SYNC로 전환하면 원격 저널 상태가 *ASYNCPEND인 경우 활성 인스턴스가 일시정지됩니다. 활성 큐 관리자 인스턴스는 나머지 저널 항목이 원격 저널로 전송될 때까지 대기합니다. 원격 저널이 로컬 저널과 동기화되면 새 대기 인스턴스가 트랜잭션 관점에서 새 활성 인스턴스와 다시 일치하게 됩니다. 다중 인스턴스 큐 관리자 관리 관점에서 볼 때 *SYNC / *ASYNC 구성에서는 IBM i 시스템 운영자에게 추가 태스크가 있습니다. 운영자는 실패한 큐 관리자 인스턴스를 재시작하는 것뿐 아니라 원격 저널링을 *SYNC로 전환해야 합니다.

1. 원격 저널은 트랜잭션 관점에서 활성 큐 관리자의 로컬 저널과 일치합니다. 활성 큐 관리자 인스턴스가 대기 인스턴스로 장애가 복구되거나 전환되면 대기 인스턴스가 즉시 재개될 수 있습니다.대기 인스턴스는 일반적으로 메시지 손실 또는 복제 없이 재개됩니다. 메시지는 마지막 체크포인트 이후 원격 저널링이 실패한 경우에만 손실 또는 복제되므로 이전에 활성화된 큐 관리자는 재시작할 수 없습니다.
2. 시스템 운영자는 활성 인스턴스를 호스팅하는 시스템을 다시 사용할 수 있게 되는 즉시 원격 저널을 *ASYNC에서 *SYNC로 전환해야 합니다. 운영자는 원격 저널을 *SYNC로 전환하기 전에 원격 저널이 따라잡을 때까지 대기할 수 있습니다. 또는 운영자는 원격 인스턴스를 *SYNC로 즉시 전환하고 대기 인스턴스 저널이 따라잡을 때까지 활성 인스턴스가 대기하도록 강제 실행할 수 있습니다. 원격 저널링이 *SYNC로 설정되면 일반적으로 대기 인스턴스가 활성 인스턴스와 트랜잭션 관점에서 일치하게 됩니다.메시지는 마지막 체크포인트 이후 원격 저널링이 실패한 경우에만 손실 또는 복제되므로 이전에 활성화된 큐 관리자는 재시작할 수 없습니다.
3. 전환 또는 장애 복구에서 구성이 복원되면 원격 저널을 호스팅하는 서버를 항상 사용할 수 있어야 합니다.

장애 복구 직후 대기 큐 관리자를 재개하려면 *SYNC / *ASYNC를 선택하십시오. 새 활성 인스턴스에서 원격 저널 설정을 *SYNC로 수동 복원해야 합니다. *SYNC / *ASYNC 구성은 한 쌍의 다중 인스턴스 큐 관리자를 관리하는 정상 패턴과 일치합니다. 하나의 인스턴스가 실패하면 일정 시간 후 대기 인스턴스가 재시작되며 이 시간 동안에는 활성 인스턴스가 장애를 복구할 수 없습니다.

***ASYNC / *ASYNC**
활성 큐 관리자와 대기 큐 관리자를 호스팅하는 두 서버 모두 *ASYNC 원격 저널링을 사용하도록 구성됩니다.

1. 전환 또는 장애 복구가 수행되면 큐 관리자가 새 서버의 저널을 계속 처리합니다. 전환 또는 장애 복구가 수행될 때 저널이 동기화되지 않을 수 있습니다. 결과적으로 메시지가 손실 또는 복제될 수 있습니다.
2. 대기 큐 관리자를 호스팅하는 서버를 사용할 수 없는 경우라도 활성 인스턴스는 실행됩니다. 로컬 저널은 사용 가능한 경우 대기 서버와 비동기식으로 복제됩니다.
3. 로컬 큐 관리자의 성능은 원격 저널링으로 영향을 받지 않습니다.

성능이 주된 요구사항이고 장애 복구 또는 전환 후 일부 메시지가 손실 또는 복제되어도 무방한 경우에는 *ASYNC / *ASYNC를 선택하십시오.

***ASYNC / *SYNC**
이 옵션 조합은 사용하지 않아도 됩니다.

원격 저널에서 큐 관리자 활성화

저널은 동기 또는 비동기식으로 복제됩니다. 원격 저널은 활성화되지 않거나 로컬 저널을 따라잡을 수 있습니다. 원격 저널은 최근에 활성화되지 않았을 수 있으므로 동기식으로 복제되더라도 따라잡을 수 있습니다. 큐 관리자가 시작 시 사용하는 원격 저널의 상태에 적용하는 규칙은 다음과 같습니다.

1. 대기가 대기의 원격 저널에서 재생해야 하고 저널 상태가 *FAILED 또는 *INACTPEND인 경우 대기 시작이 실패합니다.
2. 대기 활성화가 시작되면 대기의 원격 저널 상태가 *ACTIVE 또는 *INACTIVE여야 합니다. 상태가 *INACTIVE이고 일부 저널 데이터가 복제되지 않은 경우에는 활성화가 실패할 수 있습니다.

네트워크 파일 시스템의 큐 관리자 데이터에 원격 저널에 있는 것보다 최신 체크포인트 레코드가 있으면 실패가 발생합니다. 원격 저널이 체크포인트 사이 기본 최대 간격인 30분 이내에 활성화되면 실패가 발생할 가능성이 낮습니다. 대기 큐 관리자가 파일 시스템에서 최신 체크포인트 레코드를 읽으면 시작되지 않습니다.

활성 서버의 로컬 저널을 복원할 수 있을 때까지 기다리거나 대기 큐 관리자를 콜드 스타트할 수 있습니다. 콜드 스타트를 선택하면 큐 관리자가 저널 데이터 없이 시작되며 파일 시스템에서 큐 관리자 데이터의 일관성과 완전성에 의존합니다.

참고: 큐 관리자를 콜드 스타트하는 경우 마지막 체트포인트 후 메시지 손실 또는 복제의 위험이 있습니다. 메시지 트랜잭션은 저널에 기록되었지만 일부 트랜잭션이 파일 시스템에서 큐 관리자 데이터에 기록되지 않을 수 있습니다. 큐 관리자를 콜드 스타트하는 경우 새 저널이 시작되며 파일 시스템에서 큐 관리자 데이터에 기록되지 않은 트랜잭션이 손실됩니다.

3. 대기 큐 관리자 활성화는 대기의 원격 저널 상태가 *ASYNCPEND 또는 *SYNCPEND에서 *ASYNC 또는 *SYNC로 변경될 때까지 대기합니다. 실행 제어기의 작업 로그에 주기적으로 메시지가 기록됩니다.

   참고: 이 경우 활성화되는 대기 큐 관리자에 로컬인 원격 저널에서 활성화가 대기합니다. 큐 관리자는 또한 원격 저널 없이 일정 시간 대기한 후 계속 진행합니다. 하나 이상의 원격 저널에 동기식 쓰기를 시도하고 저널을 사용할 수 없을 때 대기합니다.
4. 저널 상태가 *FAILED 또는 *INACTPEND로 변경되면 활성화가 중지됩니다.

활성화에 사용될 로컬 및 원격 저널의 이름과 상태는 큐 관리자 오류 로그에 기록됩니다.

저널 미러링 및 *NetServer*를 사용하여 다중 인스턴스 큐 관리자 작성
두 개의 IBM i 서버에서 실행될 다중 인스턴스 큐 관리자를 작성합니다. 큐 관리자 데이터는 NetServer를 사용하는 세 번째 IBM i 서버에 저장됩니다. 큐 관리자 저널은 원격 저널링을 사용하여 두 서버 간에 미러링됩니다. **ADDMQMJRN** 명령은 원격 저널 작성을 단순화하는 데 사용됩니다.

시작하기 전에

1. 이 태스크에는 IBM i 서버 세 개가 필요합니다. 이 중 두 개 서버(예에서 ALPHA 및 BETA)에 IBM MQ를 설치하십시오. IBM MQ는 버전 7.0.1.1 이상이어야 합니다.
2. 세 번째 서버는 IBM i 서버로, NetServer가 ALPHA와 BETA에 연결합니다. 이 서버는 큐 관리자 데이터를 공유하는 데 사용됩니다. IBM MQ 설치는 필요하지 않습니다. 큐 관리자 디렉토리와 권한을 설정하기 위해 임시 조치로 서버에 IBM MQ를 설치하는 것이 좋습니다.
3. QMQM 사용자 프로필의 비밀번호가 세 서버에서 모두 동일한지 확인하십시오.
4. IBM i NetServer를 설치하십시오(i5/OS NetServer 참조).

이 태스크 정보

215 페이지의 그림 36에 표시된 구성을 작성하려면 다음 단계를 수행하십시오. 큐 관리자 데이터는 IBM i NetServer를 사용하여 연결됩니다.

- ALPHA 및 BETA에서 큐 관리자 데이터를 저장할 GAMMA의 디렉토리 공유로의 연결을 작성하십시오. 이 태스크는 또한 필요한 권한, 사용자 프로파일, 비밀번호를 설정합니다.
- 큐 관리자 인스턴스를 실행할 IBM i 시스템에 관계형 데이터베이스 항목(RDBE)을 추가하십시오. RDBE 항목은 원격 저널링에 사용되는 IBM i 시스템에 연결하는 데 사용됩니다.
- IBM i 서버의 ALPHA에서 큐 관리자 QM1 를 작성하십시오.
- 다른 IBM i 서버, BETA에 QM1에 대한 큐 관리자 제어 정보를 추가하십시오.
- 두 IBM i 서버에서 두 가지 큐 관리자 인스턴스 모두에 대한 원격 저널을 작성하십시오. 각 큐 관리자는 로컬 저널에 쓰기를 수행합니다. 로컬 저널은 원격 저널에 복제됩니다. **ADDMQMJRN** 명령은 저널 및 연결 추가를 단순화합니다.
- 대시 인스턴스를 허용하여 큐 관리자를 시작하십시오.

프로시저

1. 201 페이지의 『NetServer를 사용하여 큐 관리자 데이터의 네트워크 공유 작성』 태스크를 수행하십시오.

결과적으로, ALPHA와 BETA는 GAMMA의 /QIBM/UserData/mqm/qmgrs를 가리키는 공유, /QNTC/GAMMA/WMQ를 갖게 됩니다. 사용자 프로파일 QMQM 및 QMQMADM은 필요한 권한을 갖고 QMQM은 세 개 시스템 모두에서 일치하는 비밀번호를 갖습니다.

2. 큐 관리자 인스턴스를 호스트할 IBM i 시스템에 관계형 데이터베이스 항목(RDBE)을 추가하십시오.

   a) ALPHA에서 BETA에 대한 연결을 작성하십시오.

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) BETA에서 ALPHA에 대한 연결을 작성하십시오.

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```

3. GAMMA에 큐 관리자 데이터를 저장하여 ALPHA에서 큐 관리자 QM1을 작성하십시오.

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   /QNTC/GAMMA/WMQ 경로는 NetServer 를 사용하여 /QIBM/UserData/mqm/qmgrs에 큐 관리자 데이터를 작성합니다.

4. ALPHA에서 **ADDMQMJRN**을 실행하십시오. 이 명령은 BETA에서 QM1의 원격 저널을 추가합니다.

   ```
   ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
   ```

   QM1은 QM1의 활성 인스턴스가 ALPHA에 있을 때 ALPHA의 해당 로컬 저널에서 저널 항목을 작성합니다. ALPHA의 로컬 저널은 BETA의 원격 저널에 복제됩니다.

5. ALPHA에서 QM1 에 대해 **CRTMQM** 에 의해 작성된 IBM MQ 구성 데이터를 검사하려면 **DSPF**명령을 사용하십시오.

   이 정보는 다음 단계에서 필요합니다.

   이 예에서 다음 구성은 QM1에 대해 ALPHA의 /QIBM/UserData/mqm/mqs.ini에서 작성됩니다.

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```

6. **ADDMQMINF** 명령을 사용하여 BETA에서 QM1의 큐 관리자 인스턴스를 작성하십시오. BETA의 /QIBM/UserData/mqm/mqs.ini에서 큐 관리자 제어 정보를 수정하려면 BETA에서 다음 명령을 실행하십시오.

   ```
   ADDMQMINF MQMNAME(QM1)
   PREFIX('/QIBM/UserData/mqm')
   MQMDIR(QM1)
   MQMLIB(QMQM1)
   DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
   ```

   팁: 구성 정보를 복사하여 붙여넣으십시오. 큐 관리자 스탠자는 ALPHA와 BETA에서 동일합니다.

7. BETA에서 **ADDMQMJRN**을 실행하십시오. 이 명령은 BETA에서 QM1의 로컬 저널을 추가하고 ALPHA에서 원격 저널을 추가합니다.

   ```
   ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
   ```

   QM1은 QM1의 활성 인스턴스가 BETA에 있을 때 BETA의 해당 로컬 저널에서 저널 항목을 작성합니다. BETA의 로컬 저널은 ALPHA의 원격 저널에 복제됩니다.

참고: 또는 비동기 저널링을 사용하여 BETA에서 ALPHA로의 원격 저널링을 설정할 수 있습니다.

단계 213 페이지의 『7』의 명령 대신 이 명령을 사용하여 BETA에서 ALPHA로의 비동기 저널링을 설정하십시오.

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

ALPHA의 서버 또는 저널링이 실패의 소스인 경우에는 새 저널 항목이 ALPHA에 복제될 때까지 대기하지 않고 BETA가 시작됩니다.

ALPHA가 다시 온라인 상태가 되면 **CHGMQMJRN** 명령을 사용하여 복제 모드를 *SYNC로 전환하십시오.

208 페이지의 『ASP의 미러링된 저널 구성』의 정보를 사용하여 저널을 동기, 비동기 또는 두 가지 방식으로 모두 미러링되는지 여부를 결정하십시오. 기본값은 원격 저널로부터의 응답을 60초 동안 대기하고 동기식으로 복제하는 것입니다.

8. ALPHA 및 BETA의 저널을 사용하고 원격 저널 복제 상태가 *ACTIVE인지 확인하십시오.
   a) ALPHA:

   ```
   WRKMQMJRN MQMNAME(QM1)
   ```

   b) BETA:

   ```
   WRKMQMJRN MQMNAME(QM1)
   ```

9. ALPHA와 BETA에서 큐 관리자 인스턴스를 시작하십시오.
   a) ALPHA에서 첫 번째 인스턴스를 시작하고 활성화하십시오. 대기 인스턴스로의 전환을 사용하십시오.

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

   b) BETA에서 두 번째 인스턴스를 시작하고 활성화하십시오.

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

결과

WRKMQM을 사용하여 큐 관리자 상태를 확인하십시오.

1. ALPHA의 큐 관리자 인스턴스 상태는 *ACTIVE여야 합니다.
2. BETA의 큐 관리자 인스턴스 상태는 *STANDBY여야 합니다.

예

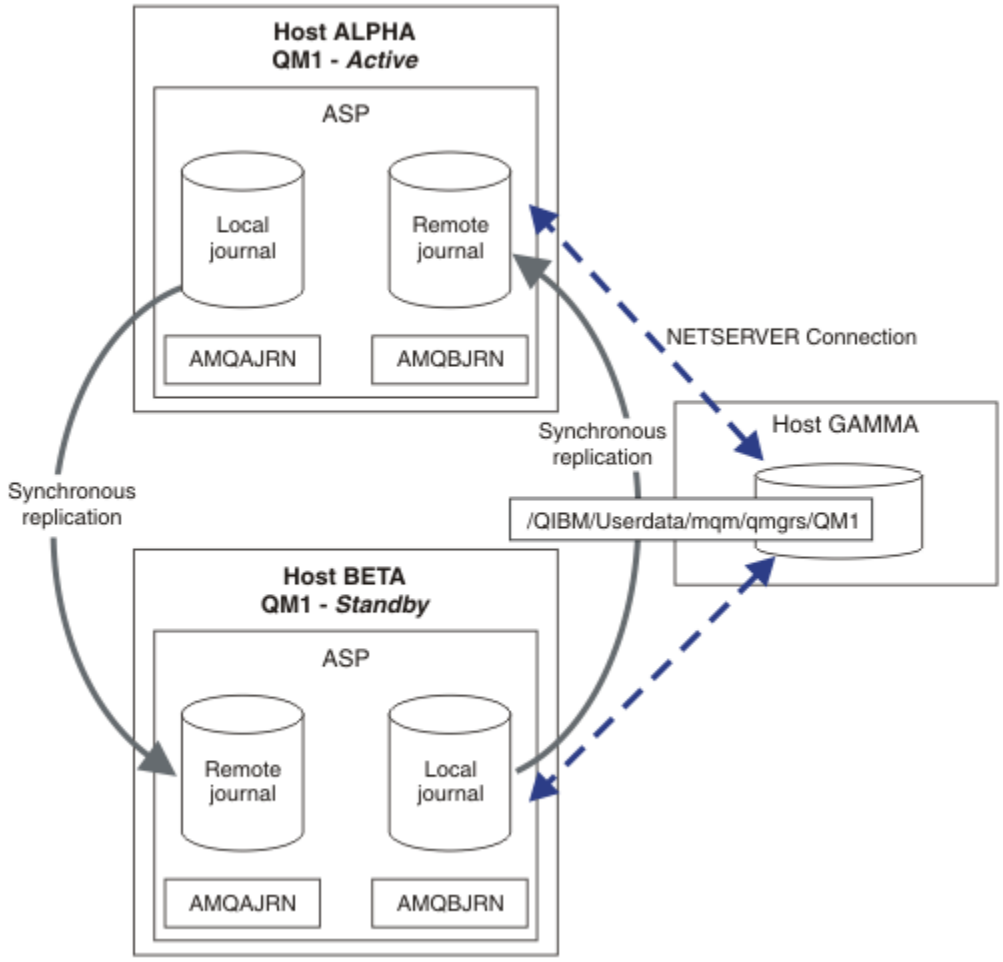


그림 *36.* 미러링된 저널 구성

다음에 수행할 작업

- 활성 및 대기 인스턴스가 자동으로 전환되는지 확인하십시오. 샘플 고가용성 샘플 프로그램을 실행하여 전환을 테스트할 수 있습니다(고가용성 샘플 프로그램 참조). 샘플 프로그램은 'C' 클라이언트입니다. 이 프로그램은 Windows 또는 Unix 플랫폼에서 실행할 수 있습니다.
  1. 고가용성 샘플 프로그램을 시작하십시오.
  2. ALPHA에서 전환을 요청하는 큐 관리자를 종료하십시오.

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. BETA의 QM1 인스턴스가 활성 상태인지 확인하십시오.
  4. ALPHA에서 QM1을 재시작하십시오.

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 대체 고가용성 구성을 확인하십시오.
  1. NetServer를 사용하여 Windows Server에 큐 관리자 데이터를 배치하십시오.

2. 원격 저널링을 사용하여 큐 관리자 저널을 미러링하는 대신 독립 ASP에 저널을 저장하십시오. 독립 ASP를 ALPHA에서 BETA로 전송하려면 IBM i 클러스터링을 사용하십시오.

NetServer 및 저널 미러링을 사용하여 단일 인스턴스 큐 관리자를 다중 인스턴스 큐 관리자로 변환
단일 인스턴스 큐 관리자를 다중 인스턴스 큐 관리자로 변환합니다. 큐 관리자 데이터를 NetServer가 연결한 네트워크 공유로 이동합니다. 원격 저널링을 사용하여 큐 관리자 저널을 두 번째 IBM i 서버로 미러링합니다.

시작하기 전에

1. 이 태스크에는 IBM i 서버 세 개가 필요합니다. 예의 ALPHA 서버에서 기존 IBM MQ 설치는 버전 7.0.1.1 이상이어야 합니다. 예에서 ALPHA는 큐 관리자 QM1을 실행합니다.
2. IBM MQ를 두 번째 IBM i 서버(예의 BETA)에 설치하십시오.
3. 세 번째 서버는 IBM i 서버로, NetServer가 ALPHA와 BETA에 연결합니다. 이 서버는 큐 관리자 데이터를 공유하는 데 사용됩니다. IBM MQ 설치는 필요하지 않습니다. 큐 관리자 디렉토리와 권한을 설정하기 위해 임시 조치로 서버에 IBM MQ를 설치하는 것이 좋습니다.
4. QMQM 사용자 프로필의 비밀번호가 세 서버에서 모두 동일한지 확인하십시오.
5. IBM i NetServer를 설치하십시오(i5/OS NetServer 참조).

이 태스크 정보

단일 인스턴스 큐 관리자를 219 페이지의 그림 37에 표시된 다중 인스턴스 큐 관리자로 변환하려면 다음 단계를 수행하십시오. 단일 인스턴스 큐 관리자가 태스크에서 삭제된 다음 재작성되며 NetServer가 연결하는 네트워크 공유에 큐 관리자 데이터를 저장합니다. 이 프로시저는 **CPY** 명령을 사용하여 큐 관리자 디렉토리와 파일을 네트워크 공유로 이동하는 것보다 신뢰할 수 있는 방법입니다.

- ALPHA 및 BETA에서 큐 관리자 데이터를 저장할 GAMMA의 디렉토리 공유로의 연결을 작성하십시오. 이 태스크는 또한 필요한 권한, 사용자 프로파일, 비밀번호를 설정합니다.
- 큐 관리자 인스턴스를 실행할 IBM i 시스템에 관계형 데이터베이스 항목(RDBE)을 추가하십시오. RDBE 항목은 원격 저널링에 사용되는 IBM i 시스템에 연결하는 데 사용됩니다.
- 큐 관리자 로그와 정의를 저장하고 큐 관리자를 중지한 후 삭제하십시오.
- 큐 관리자를 재작성하고 GAMMA의 네트워크 공유에 큐 관리자 데이터를 저장하십시오.
- 큐 관리자의 두 번째 인스턴스를 다른 서버에 추가하십시오.
- 두 IBM i 서버에서 두 가지 큐 관리자 인스턴스 모두에 대한 원격 저널을 작성하십시오. 각 큐 관리자는 로컬 저널에 쓰기를 수행합니다. 로컬 저널은 원격 저널에 복제됩니다. **ADDMQMJRN** 명령은 저널 및 연결 추가를 단순화합니다.
- 대시 인스턴스를 허용하여 큐 관리자를 시작하십시오.

참고:

태스크의 단계 217 페이지의 『4』에서 단일 인스턴스 큐 관리자, QM1을 삭제합니다. 큐 관리자를 삭제하면 큐의 모든 지속 메시지가 삭제됩니다. 따라서 큐 관리자로 변환하기 전에 큐 관리자가 저장한 모든 메시지 처리를 완료하십시오. 모든 메시지 처리가 불가능한 경우에는 단계 217 페이지의 『4』 이전에 큐 관리자 라이브러리를 백업하십시오. 단계 217 페이지의 『5』 이후 큐 관리자 라이브러리를 복원하십시오.

참고:

태스크의 단계 217 페이지의 『5』에서 QM1을 재작성합니다. 큐 관리자의 이름은 같더라도 큐 관리자 ID는 다릅니다. 큐 관리자 클러스터링은 큐 관리자 ID를 사용합니다. 클러스터에서 큐 관리자를 삭제하고 재작성하려면 먼저 클러스터에서 큐 관리자를 제거해야 합니다. 클러스터에서 큐 관리자 제거: 대체 메소드 또는 클러스터에서 큐 관리자 제거를 참조하십시오. 큐 관리자를 재작성했으면 클러스터에 추가하십시오. 이전과 이름은 같지만 클러스터의 다른 큐 관리자가 새 큐 관리자로 인식합니다.

프로시저

1. 201 페이지의 『NetServer를 사용하여 큐 관리자 데이터의 네트워크 공유 작성』 태스크를 수행하십시오.

결과적으로, ALPHA와 BETA는 GAMMA의 /QIBM/UserData/mqm/qmgrs를 가리키는 공유, /QNTC/GAMMA/WMQ를 갖게 됩니다. 사용자 프로파일 QMQM 및 QMQMADM은 필요한 권한을 갖고 QMQM은 세 개 시스템 모두에서 일치하는 비밀번호를 갖습니다.

2. 큐 관리자 인스턴스를 호스트할 IBM i 시스템에 관계형 데이터베이스 항목(RDBE)을 추가하십시오.
   a) ALPHA에서 BETA에 대한 연결을 작성하십시오.

```
ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
```

   b) BETA에서 ALPHA에 대한 연결을 작성하십시오.

```
ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
```

3. 큐 관리자 오브젝트를 재작성하는 스크립트를 작성하십시오.

```
QSAVEQMGR LCLQMGRNAM(QM1) FILENAME('*CURLIB/QMQSC(QM1)')
OUTPUT(*REPLACE) MAKEAUTH(*YES) AUTHFN('*CURLIB/QMAUT(QM1)')
```

4. 큐 관리자를 중지하고 삭제하십시오.

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*YES) TIMEOUT(15)
DLTMQM MQMNAME(QM1)
```

5. GAMMA에 큐 관리자 데이터를 저장하여 ALPHA에서 큐 관리자 QM1을 작성하십시오.

```
CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
MQMDIRP(' /QNTC/GAMMA/WMQ ')
```

   /QNTC/GAMMA/WMQ 경로는 NetServer 를 사용하여 /QIBM/UserData/mqm/qmgrs에 큐 관리자 데이터를 작성합니다.

6. 저장된 정의에서 QM1의 큐 관리자 오브젝트를 재작성하십시오.

```
STRMQMMQSC SRCMBR(QM1) SRCFILE(*CURLIB/QMQSC) MQMNAME(QM1)
```

7. 저장된 정보에서 권한을 적용하십시오.
   a) 저장된 권한 부여 프로그램 컴파일하십시오.

```
CRTCLPGM PGM(*CURLIB/QM1) SRCFILE(*CURLIB/QMAUT)
SRCMBR(QM1) REPLACE(*YES)
```

   b) 프로그램을 실행하여 권한을 적용하십시오.

```
CALL PGM(*CURLIB/QM1)
```

   c) QM1에 대한 보안 정보를 새로 고치십시오.

```
RFRMQMAUT MQMNAME(QM1)
```

8. ALPHA에서 **ADDMQMJRN**을 실행하십시오. 이 명령은 BETA에서 QM1의 원격 저널을 추가합니다.

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
```

   QM1은 QM1의 활성 인스턴스가 ALPHA에 있을 때 ALPHA의 해당 로컬 저널에서 저널 항목을 작성합니다. ALPHA의 로컬 저널은 BETA의 원격 저널에 복제됩니다.

9. ALPHA에서 QM1 에 대해 **CRTMQM** 에 의해 작성된 IBM MQ 구성 데이터를 검사하려면 **DSPF**명령을 사용하십시오.

   이 정보는 다음 단계에서 필요합니다.

   이 예에서 다음 구성은 QM1에 대해 ALPHA의 /QIBM/UserData/mqm/mqs.ini에서 작성됩니다.

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```

10. **ADDMQMINF** 명령을 사용하여 BETA에서 QM1의 큐 관리자 인스턴스를 작성하십시오. BETA의 /QIBM/UserData/mqm/mqs.ini에서 큐 관리자 제어 정보를 수정하려면 BETA에서 다음 명령을 실행하십시오.

    ```
    ADDMQMINF MQMNAME(QM1)
    PREFIX('/QIBM/UserData/mqm')
    MQMDIR(QM1)
    MQMLIB(QMQM1)
    DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
    ```

    팁: 구성 정보를 복사하여 붙여넣으십시오. 큐 관리자 스탠자는 ALPHA와 BETA에서 동일합니다.

11. BETA에서 **ADDMQMJRN**을 실행하십시오. 이 명령은 BETA에서 QM1의 로컬 저널을 추가하고 ALPHA에서 원격 저널을 추가합니다.

    ```
    ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
    ```

    QM1은 QM1의 활성 인스턴스가 BETA에 있을 때 BETA의 해당 로컬 저널에서 저널 항목을 작성합니다. BETA의 로컬 저널은 ALPHA의 원격 저널에 복제됩니다.

    참고: 또는 비동기 저널링을 사용하여 BETA에서 ALPHA로의 원격 저널링을 설정할 수 있습니다.

    단계 213 페이지의 『7』 의 명령 대신 이 명령을 사용하여 BETA에서 ALPHA로의 비동기 저널링을 설정하십시오.

    ```
    ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
    ```

    ALPHA의 서버 또는 저널링이 실패의 소스인 경우에는 새 저널 항목이 ALPHA에 복제될 때까지 대기하지 않고 BETA가 시작됩니다.

    ALPHA가 다시 온라인 상태가 되면 **CHGMQMJRN** 명령을 사용하여 복제 모드를 *SYNC로 전환하십시오.

    208 페이지의 『ASP의 미러링된 저널 구성』 의 정보를 사용하여 저널을 동기, 비동기 또는 두 가지 방식으로 모두 미러링되는지 여부를 결정하십시오. 기본값은 원격 저널로부터의 응답을 60초 동안 대기하고 동기식으로 복제하는 것입니다.

12. ALPHA 및 BETA의 저널을 사용하고 원격 저널 복제 상태가 *ACTIVE인지 확인하십시오.

    a) ALPHA:

    ```
    WRKMQMJRN MQMNAME(QM1)
    ```

    b) BETA:

    ```
    WRKMQMJRN MQMNAME(QM1)
    ```

13. ALPHA와 BETA에서 큐 관리자 인스턴스를 시작하십시오.

    a) ALPHA에서 첫 번째 인스턴스를 시작하고 활성화하십시오. 대기 인스턴스로의 전환을 사용하십시오.

    ```
    STRMQM MQMNAME(QM1) STANDBY(*YES)
    ```

b) BETA에서 두 번째 인스턴스를 시작하고 활성화하십시오.

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

결과

WRKMQM을 사용하여 큐 관리자 상태를 확인하십시오.

1. ALPHA의 큐 관리자 인스턴스 상태는 *ACTIVE여야 합니다.
2. BETA의 큐 관리자 인스턴스 상태는 *STANDBY여야 합니다.

예

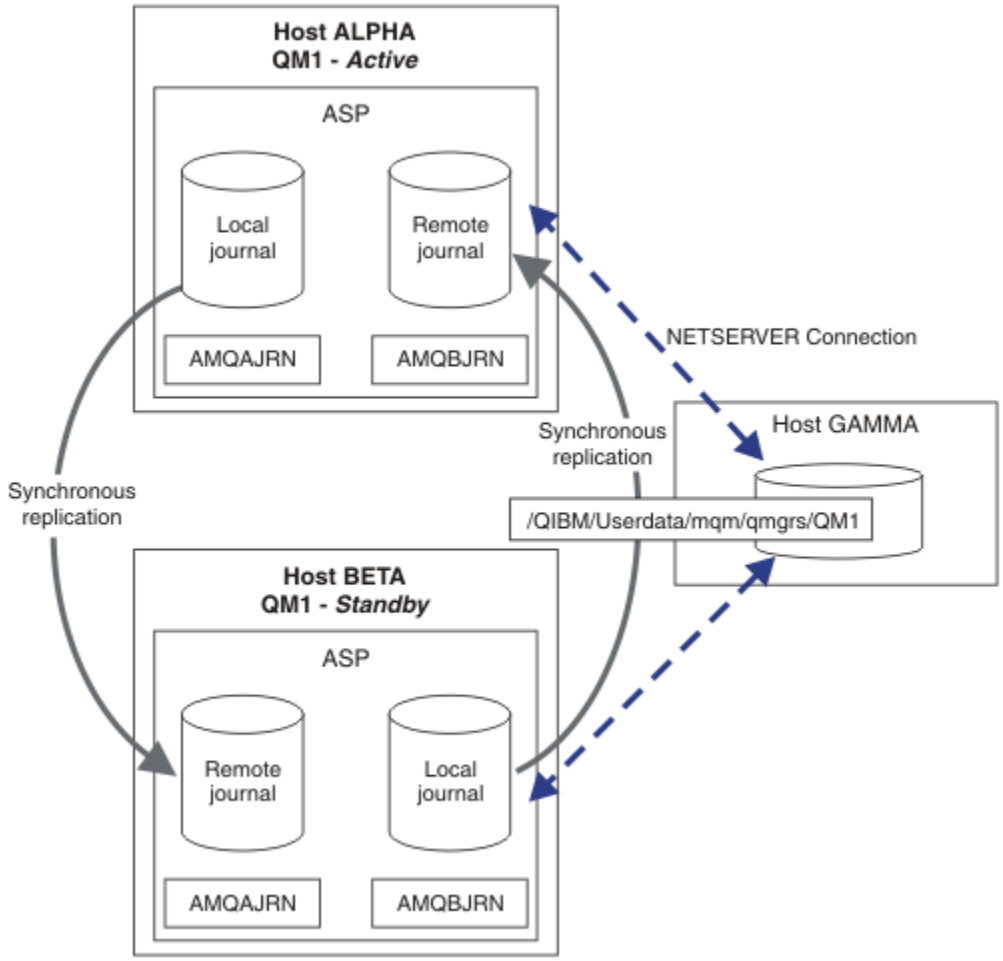


그림 *37.* 미러링된 저널 구성

다음에 수행할 작업

- 활성 및 대기 인스턴스가 자동으로 전환되는지 확인하십시오. 샘플 고가용성 샘플 프로그램을 실행하여 전환을 테스트할 수 있습니다(고가용성 샘플 프로그램 참조). 샘플 프로그램은 'C' 클라이언트입니다. 이 프로그램은 Windows 또는 Unix 플랫폼에서 실행할 수 있습니다.
  1. 고가용성 샘플 프로그램을 시작하십시오.
  2. ALPHA에서 전환을 요청하는 큐 관리자를 종료하십시오.

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
```

3. BETA의 QM1 인스턴스가 활성 상태인지 확인하십시오.
4. ALPHA에서 QM1을 재시작하십시오.

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

- 대체 고가용성 구성을 확인하십시오.
  1. NetServer를 사용하여 Windows Server에 큐 관리자 데이터를 배치하십시오.
  2. 원격 저널링을 사용하여 큐 관리자 저널을 미러링하는 대신 독립 ASP에 저널을 저장하십시오. 독립 ASP를 ALPHA에서 BETA로 전송하려면 IBM i 클러스터링을 사용하십시오.

전환된 독립 *ASP* 저널 구성

다중 인스턴스 큐 관리자 구성을 작성하기 위해 독립 ASP 저널을 복제하지 않아도 됩니다. 활성 큐 관리자에서 대기 큐 관리자로 독립 ASP를 전송하기 위한 수단을 자동화해야 합니다. 독립 ASP를 사용하여 가능한 대체 고가용성 솔루션이 있으며 일부 솔루션은 다중 인스턴스 큐 관리자를 사용하지 않아도 됩니다.

독립 ASP를 사용하는 경우 큐 관리자 저널을 미러링하지 않아도 됩니다. 클러스터 관리를 설치하고 큐 관리자 인스턴스를 호스트하는 서버가 동일한 클러스터 자원 그룹에 있는 경우 활성 인스턴스를 실행하는 호스트가 실패하면 큐 관리자 저널이 근거리에 있는 활성 서버 내 다른 서버에 자동으로 전송될 수 있습니다. 계획된 전환의 일부로 저널을 수동으로 전송하거나 독립 ASP를 프로그래밍 방식으로 전송하는 명령 프로시저를 작성할 수 있습니다.

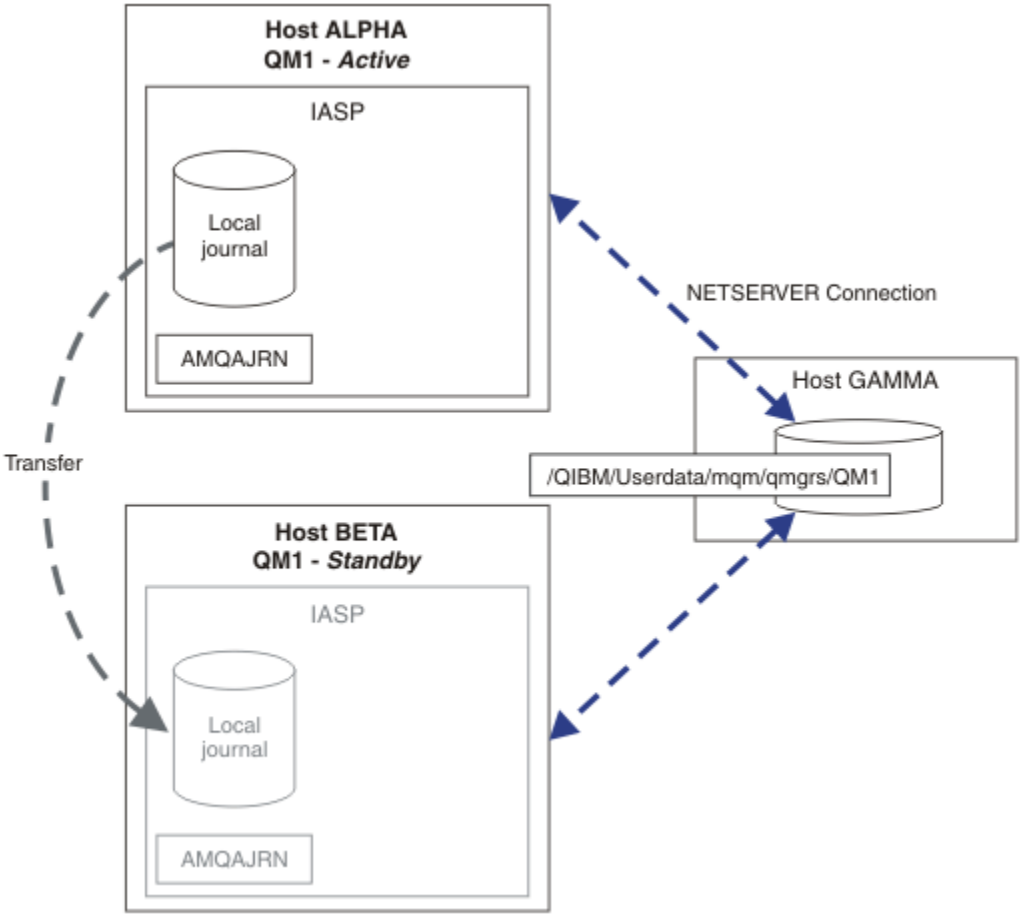


그림 *38.* 독립 *ASP*를 사용하여 큐 관리자 저널 전송

다중 인스턴스 큐 관리자 조작의 경우, 큐 관리자 데이터를 공유 파일 시스템에 저장해야 합니다. 파일 시스템은 다양한 플랫폼에서 호스트할 수 있습니다. 다중 인스턴스 큐 관리자 데이터는 ASP 또는 독립 ASP에 저장할 수 없습니다.

공유 파일 시스템은 구성에서 두 가지 역할을 수행합니다. 큐 관리자의 모든 인스턴스 간에 동일한 큐 관리자 데이터를 공유합니다. 파일 시스템에는 시작 시 큐 관리자의 인스턴스가 하나만 큐 관리자 데이터에 액세스할 수 있도록 하는 강력한 잠금 프로토콜이 필요합니다. 큐 관리자가 실패하거나 파일 서버에 대한 통신이 끊어지면 파일 시스템이 더 이상 파일 시스템과 통신하지 않는 활성 인스턴스가 보유한 큐 관리자 데이터에 대한 잠금을 해제해야 합니다. 그러면 대기 큐 관리자 인스턴스가 큐 관리자 데이터에 대한 읽기/쓰기 액세스 권한을 얻을 수 있습니다. 파일 시스템 프로토콜은 다중 인스턴스 큐 관리자와 올바르게 작업을 수행하기 위한 규칙 세트를 준수해야 합니다(200 페이지의 『고가용성 솔루션의 컴포넌트』 참조).

잠금 메커니즘은 큐 관리자 시작 명령을 직렬화하고 활성 상태인 큐 관리자 인스턴스를 제어합니다. 큐 관리자가 활성화되면 사용자 또는 HA 클러스터가 대기 서버로 전송한 로컬 저널로부터 해당 큐를 재빌드합니다. 동일한 큐 관리자로의 재연결을 대기하는 다시 연결 가능한 클라이언트가 다시 연결되고 인플라이트 트랜잭션은 백아웃됩니다. 큐 관리자 서비스로 시작되도록 구성된 애플리케이션이 시작됩니다.

클러스터 자원 관리자를 구성하거나 독립 ASP를 수동으로 전송하여, 독립 ASP의 실패한 활성 큐 관리자 인스턴스로부터의 로컬 저널이 새로 활성화된 대기 큐 관리자 인스턴스를 호스트하는 서버로 전송되는지 확인해야 합니다. 백업 및 재해 복구를 위해 독립 ASP를 사용하려는 경우 독립 ASP를 사용하더라도 원격 저널을 구성하고 미러링하며 다중 인스턴스 큐 관리자 구성을 위해 원격 저널 미러링을 사용합니다.

독립 ASP를 사용하도록 선택한 경우에는 다른 고가용성 구성을 고려할 수 있습니다. 이 솔루션의 배경은 224 페이지의 『독립 ASP 및 고가용성』에서 설명됩니다.

1. 다중 인스턴스 큐 관리자를 사용하는 대신, 독립 ASP에서 단일 인스턴스 큐 관리자를 설치 및 구성하고 IBM i 고가용성 서비스를 사용하여 큐 관리자 장애를 복구하십시오. 큐 관리자가 서버와 관계 없이 실패했는지 여부를 감지하기 위해 큐 관리자 모니터로 솔루션을 강화해야 할 수 있습니다. 이는 *Supportpac MC41:* 고가용성을 위한 *IBM MQ for iSeries* 구성에 제공된 솔루션의 기반이 됩니다.
2. 로컬 버스의 독립 ASP를 전환하는 대신 독립 ASP와 교차 사이트 미러링(XSM)을 사용하여 독립 ASP를 미러링하십시오. 이는 장거리 로그 레코드 쓰기에 소요되는 시간이 허용하는 한 독립 ASP 솔루션의 지리적 범위를 확장합니다.

독립 *ASP* 및 *NetServer*를 사용하여 다중 인스턴스 큐 관리자 작성
두 개의 IBM i 서버에서 실행될 다중 인스턴스 큐 관리자를 작성합니다. 큐 관리자 데이터는 NetServer를 사용하는 IBM i 서버에 저장됩니다. 큐 관리자 저널은 독립 ASP에 저장됩니다. 큐 관리자 저널을 포함하는 독립 ASP를 다른 IBM i 서버로 전송하려면 IBM i 클러스터링 또는 수동 프로시저를 사용하십시오.

시작하기 전에

1. 이 태스크에는 IBM i 서버 세 개가 필요합니다. 이 중 두 개 서버(예에서 ALPHA 및 BETA)에 IBM MQ를 설치하십시오. IBM MQ는 버전 7.0.1.1 이상이어야 합니다.
2. 세 번째 서버는 IBM i 서버로, NetServer가 ALPHA와 BETA에 연결합니다. 이 서버는 큐 관리자 데이터를 공유하는 데 사용됩니다. IBM MQ 설치는 필요하지 않습니다. 큐 관리자 디렉토리와 권한을 설정하기 위해 임시 조치로 서버에 IBM MQ를 설치하는 것이 좋습니다.
3. `QMQM` 사용자 프로필의 비밀번호가 세 서버에서 모두 동일한지 확인하십시오.
4. IBM i NetServer를 설치하십시오(i5/OS NetServer 참조).
5. 실패한 큐 관리자에서 인계받는 대기 큐 관리자로 독립 ASP를 전송하는 프로시저를 작성하십시오. *SupportPac MC41:* 고가용성을 위한 *IBM MQ for iSeries* 구성에서 독립 ASP 전송 프로시저 설계에 유용한 일부 기법을 참조할 수 있습니다.

이 태스크 정보

223 페이지의 그림 39에 표시된 구성을 작성하려면 다음 단계를 수행하십시오. 큐 관리자 데이터는 IBM i NetServer를 사용하여 연결됩니다.

- ALPHA 및 BETA에서 큐 관리자 데이터를 저장할 GAMMA의 디렉토리 공유로의 연결을 작성하십시오. 이 태스크는 또한 필요한 권한, 사용자 프로파일, 비밀번호를 설정합니다.
- IBM i 서버의 ALPHA에서 큐 관리자 `QM1` 를 작성하십시오.

- 다른 IBM i 서버, BETA에 QM1에 대한 큐 관리자 제어 정보를 추가하십시오.
- 대시 인스턴스를 허용하여 큐 관리자를 시작하십시오.

프로시저

1. 201 페이지의 『NetServer를 사용하여 큐 관리자 데이터의 네트워크 공유 작성』 태스크를 수행하십시오.

   결과적으로, ALPHA와 BETA는 GAMMA의 /QIBM/UserData/mqm/qmgrs를 가리키는 공유, /QNTC/GAMMA/WMQ를 갖게 됩니다. 사용자 프로파일 QMQM 및 QMQMADM은 필요한 권한을 갖고 QMQM은 세 개 시스템 모두에서 일치하는 비밀번호를 갖습니다.
2. GAMMA에 큐 관리자 데이터를 저장하여 ALPHA에서 큐 관리자 QM1을 작성하십시오.

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   /QNTC/GAMMA/WMQ 경로는 NetServer 를 사용하여 /QIBM/UserData/mqm/qmgrs에 큐 관리자 데이터를 작성합니다.
3. ALPHA에서 QM1 에 대해 **CRTMQM** 에 의해 작성된 IBM MQ 구성 데이터를 검사하려면 **DSPF**명령을 사용하십시오.

   이 정보는 다음 단계에서 필요합니다.

   이 예에서 다음 구성은 QM1에 대해 ALPHA의 /QIBM/UserData/mqm/mqs.ini에서 작성됩니다.

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```
4. **ADDMQMINF** 명령을 사용하여 BETA에서 QM1의 큐 관리자 인스턴스를 작성하십시오. BETA의 /QIBM/UserData/mqm/mqs.ini에서 큐 관리자 제어 정보를 수정하려면 BETA에서 다음 명령을 실행하십시오.

   ```
   ADDMQMINF MQMNAME(QM1)
   PREFIX('/QIBM/UserData/mqm')
   MQMDIR(QM1)
   MQMLIB(QMQM1)
   DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
   ```

   팁: 구성 정보를 복사하여 붙여넣으십시오. 큐 관리자 스탠자는 ALPHA와 BETA에서 동일합니다.
5. ALPHA와 BETA에서 큐 관리자 인스턴스를 시작하십시오.

   a) ALPHA에서 첫 번째 인스턴스를 시작하고 활성화하십시오. 대기 인스턴스로의 전환을 사용하십시오.

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

   b) BETA에서 두 번째 인스턴스를 시작하고 활성화하십시오.

   ```
   STRMQM MQMNAME(QM1) STANDBY(*YES)
   ```

결과

WRKMQM을 사용하여 큐 관리자 상태를 확인하십시오.

1. ALPHA의 큐 관리자 인스턴스 상태는 *ACTIVE여야 합니다.
2. BETA의 큐 관리자 인스턴스 상태는 *STANDBY여야 합니다.

예

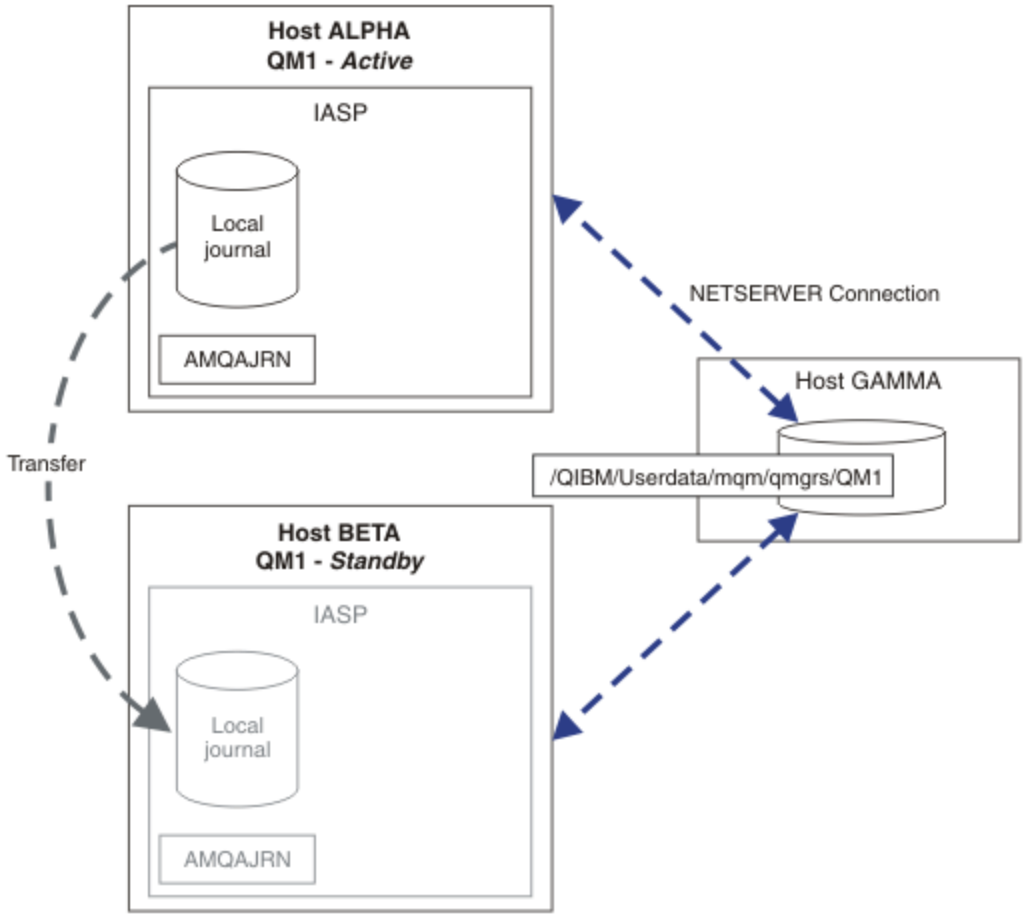


그림 *39.* 독립 *ASP*를 사용하여 큐 관리자 저널 전송

다음에 수행할 작업

- 활성 및 대기 인스턴스가 자동으로 전환되는지 확인하십시오. 샘플 고가용성 샘플 프로그램을 실행하여 전환을 테스트할 수 있습니다(고가용성 샘플 프로그램 참조). 샘플 프로그램은 'C' 클라이언트입니다. 이 프로그램은 Windows 또는 Unix 플랫폼에서 실행할 수 있습니다.
  1. 고가용성 샘플 프로그램을 시작하십시오.
  2. ALPHA에서 전환을 요청하는 큐 관리자를 종료하십시오.

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. BETA의 QM1 인스턴스가 활성 상태인지 확인하십시오.
  4. ALPHA에서 QM1을 재시작하십시오.

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 대체 고가용성 구성을 확인하십시오.
  1. NetServer를 사용하여 IBM i 서버에 큐 관리자 데이터를 배치하십시오.
  2. 독립 ASP를 사용하여 대기 서버로 큐 관리자 저널을 전송하는 대신 원격 저널링을 사용하여 대기 서버에 저널을 미러링하십시오.

독립 *ASP* 및 고가용성
독립 ASP를 사용하여 서버 간에 애플리케이션과 데이터를 이동할 수 있습니다. 독립 ASP의 유연성은 ASP가 일부 IBM i 고가용성 솔루션의 기반임을 의미합니다. 큐 관리자 저널에 ASP 또는 독립 ASP를 사용할지 여부를 고려할 때는 독립 ASP를 기반으로 하는 다른 고가용성 구성을 고려해야 합니다.

보조 기억장치 풀(ASP)은 IBM i 아키텍처의 빌딩 블록입니다. 디스크 장치가 함께 그룹화되어 단일 ASP를 구성합니다. 오브젝트를 여러 ASP에 배치함으로써 다른 ASP의 디스크 장애로 특정 ASP의 데이터가 영향을 받지 않도록 보호할 수 있습니다.

모든 IBM i 서버는 하나 이상의 기본 ASP(시스템 ASP라고 함)를 갖습니다. 이 ASP는 ASP1로 지정되며 *SYSBAS라고도 합니다. 애플리케이션 관점에서 시스템 ASP와 구분할 수 없는 최대 31개의 추가 기본 사용자 ASP를 구성할 수 있습니다. ASP는 동일한 네임스페이스를 공유하기 때문입니다. 여러 기본 ASP를 사용하여 많은 디스크에 애플리케이션을 분산시킴으로써 성능을 향상시키고 복구 시간을 줄일 수 있습니다. 여러 기본 ASP를 사용하는 경우 또한 디스크 장애로부터 어느 정도 격리될 수 있지만 전반적인 신뢰성은 향상되지 않습니다.

독립 ASP는 특수 유형의 ASP입니다. 이 ASP는 독립 디스크 풀이라고도 합니다. 독립 디스크 풀은 IBM i 고가용성의 핵심 구성요소입니다. 스스로를 독립 디스크 스토리지 장치에서 연결되는 현재 시스템에 독립적인 것으로 간주하는 애플리케이션과 데이터를 저장할 수 있습니다. 전환 가능 또는 전환 불가능 독립 ASP를 구성할 수 있습니다. 가용성 관점에서는 일반적으로 서버에서 서버로 자동 전송될 수 있는 전환 가능 독립 ASP만 고려합니다. 결과적으로, 독립 ASP의 애플리케이션과 데이터를 서버에서 서버로 이동할 수 있습니다.

기본 사용자 ASP와 달리 독립 ASP는 시스템 ASP와 동일한 네임스페이스를 공유하지 않습니다. 사용자 APS에 대한 작업을 수행하는 애플리케이션에는 독립 ASP에 대한 작업을 수행하기 위해 변경사항이 필요하지 않습니다. 소프트웨어와 사용하는 써드파티 소프트웨어가 독립 ASP 환경에서 작동하는지 확인해야 합니다.

독립 ASP가 다른 서버에 연결될 때 독립 ASP의 네임스페이스가 시스템 ASP의 네임스페이스와 결합되어야 합니다. 이 프로세스를 독립 ASP 변환이라고 합니다. 서버 IPL을 수행하지 않고 독립 ASP를 변환할 수 있습니다. 특정 서버에서 다른 서버로 독립 ASP를 자동으로 전송하려면 클러스터링 지원이 필요합니다.

독립 ASP로 신뢰할 수 있는 솔루션 빌드

ASP에 저널링한 후 저널 복제를 사용하지 않고 독립 ASP에 저널링하는 경우 실패한 큐 관리자 인스턴스에서 로컬 저널의 사본을 대기 큐 관리자에 제공할 수 있는 대체 방법을 제공합니다. 독립 ASP를 다른 서버에 자동으로 전송하려면 클러스터링 지원을 설치 및 구성한 상태여야 합니다. 다중 인스턴스 큐 관리자를 사용하여 결합 또는 대체할 수 있는 클러스터 지원 기반의 독립 ASP와 하위 레벨 디스크 미러링에 대한 여러 가지 고가용성 솔루션이 존재합니다.

다음 목록은 독립 ASP 기반의 신뢰할 수 있는 솔루션을 빌드하는 데 필요한 구성요소를 설명합니다.

저널링
큐 관리자와 다른 애플리케이션이 로컬 저널을 사용하여 지속 데이터를 디스크에 안전하게 기록함으로써 서버 장애로 인한 메모리에 데이터 손실을 방지합니다. 이를 특정 시점 일관성이라고도 합니다. 이는 일정 기간 수행되는 여러 업데이트의 일관성을 보장하지 않습니다.

커미트 제어
글로벌 트랜잭션을 사용함으로써 저널에 기록된 데이터가 일관성을 유지하도록 메시지 및 데이터베이스에 대한 업데이트를 조정할 수 있습니다. 2단계 커미트 프로토콜을 사용하여 일정 시간 동안 일관성을 유지할 수 있습니다.

전환된 디스크

전환된 디스크는 HA 클러스터의 디바이스 클러스터 자원 그룹(CRG)이 관리합니다. 계획되지 않은 가동 중단 시 CRG가 독립 ASP를 새 서버로 자동 전환합니다. CRG는 지리적으로 로컬 IO 버스 범위로 제한됩니다.

전환 가능 독립 ASP에 로컬 저널을 구성하여 다른 서버로 저널을 전송하고 메시지 처리를 재개할 수 있습니다. 독립 ASP가 실패하지 않으면 동기점 제어 없이 작성되거나 동기점 제어와 함께 커미트된 지속 메시지에 대한 변경사항이 손실되지 않습니다.

전환 가능 독립 ASP에 저널링과 커미트 제어를 둘 다 사용하는 경우, 데이터베이스 저널과 큐 관리자 저널을 다른 서버로 전송할 수 있으며 일관성 또는 커미트된 트랜잭션 손실 없이 트랜잭션 처리를 재개할 수 있습니다.

사이트 간 미러링(XSM)

XSM은 1차 독립 ASP를 TCP/IP 네트워크에서 지리적으로 멀리 떨어져 있는 2차 독립 ASP로 미러링하고 실패 시 제어를 자동으로 전송합니다. 동기 또는 비동기 미러를 구성하도록 선택할 수 있습니다. 동기 미러링은 프로덕션 시스템에 대한 쓰기 조작이 완료되기 전에 데이터가 미러링되므로 큐 관리자의 성능을 저하시키지만 2차 독립 ASP가 최신 상태로 유지되도록 보장합니다. 반면에 비동기 미러링을 사용하는 경우에는 2차 독립 ASP가 최신 상태로 유지되도록 보장할 수 없습니다. 비동기 미러링은 2차 독립 ASP의 일관성을 유지합니다.

XSM 기술은 세 가지가 있습니다.

지리적 미러링

지리적 미러링은 클러스터링의 확장으로, 넓은 영역에서 독립 ASP를 전환할 수 있습니다. 동기 모드와 비동기 모드가 둘 다 있습니다. 고가용성은 동기 모드에서만 보장할 수 있지만 독립 ASP를 분리해도 성능에 큰 영향을 주지 않습니다. 지리적 미러링과 전환된 디스크를 결합하여 로컬로는 고가용성을 제공하고 원격으로는 재해 복구를 제공할 수 있습니다.

메트로 미러링

메트로 미러링은 로컬 버스보다 먼 거리에서 빠른 로컬 동기 미러링을 제공하는 디바이스 레벨 서비스입니다. 이 서비스와 다중 인스턴스 큐 관리자를 결합하여 큐 관리자의 고가용성을 제공할 수 있으며 독립 ASP의 두 개 사본을 작성함으로써 큐 관리자 저널의 고가용성을 제공할 수 있습니다.

글로벌 미러링

글로벌 미러링은 비동기 미러링을 제공하는 디바이스 레벨 서비스이므로 장거리 백업 및 재해 복구에 적합하지만 고가용성을 위한 일반적인 선택사항은 아닙니다. 동시성이 아닌 특정 시점 일관성만 유지하기 때문입니다.

고려해야 하는 핵심 의사결정 사항은 다음과 같습니다.

ASP 또는 독립 ASP 여부

다중 인스턴스 큐 관리자를 사용하기 위해 IBM i HA 클러스터를 실행하지 않아도 됩니다. 이미 독립 ASP를 사용하고 있거나 독립 ASP가 필요한 다른 애플리케이션에 대한 가용성 요구사항이 있는 경우 독립 ASP를 선택할 수 있습니다. 큐 관리자 모니터링을 큐 관리자 실패 감지 수단으로 바꾸기 위해 독립 ASP와 다중 인스턴스 큐 관리자를 결합할 수도 있습니다.

가용성

복구 시간 목표(RTO)란 무엇입니까? 거의 중단되지 않는 동작이 나타나야 하는 경우 복구 시간이 가장 빠른 솔루션은 무엇입니까?

저널 가용성

저널을 단일 장애 지점으로서 제거하기 위한 방법. RAID 1 이상 디바이스를 사용하여 하드웨어 솔루션을 채택하거나, 복제 저널 또는 디스크 미러링을 사용하는 소프트웨어 솔루션을 사용할 수 있습니다.

거리

활성 큐 관리자 인스턴스와 대기 큐 관리자 인스턴스 사이의 거리. 사용자가 약 250미터 이상의 거리에서 동기식 복제에 따른 성능 저하를 허용할 수 있습니까?

스킬

솔루션 유지보수 및 실행과 관련된 관리 태스크를 정기적으로 자동화하기 위해 수행해야 할 작업이 있습니다. 자동화 수행을 위해 필요한 스킬은 ASP를 기반으로 하는 솔루션과 독립 ASP를 기반으로 하는 솔루션에 각각 다릅니다.

다중 인스턴스 큐 관리자 삭제

다중 인스턴스 큐 관리자를 삭제하기 전에 원격 저널링을 중지하고 큐 관리자 인스턴스를 제거하십시오.

시작하기 전에

1. 이 예에서는 QM1 큐 관리자의 두 개 인스턴스가 ALPHA 및 BETA 서버에서 정의됩니다. ALPHA는 활성 인스턴스이고 BETA는 대기입니다. 큐 관리자 QM1 와 연관된 큐 관리자 데이터는 NetServer를 사용하여 IBM i 서버 GAMMA에 저장됩니다. 212 페이지의 『저널 미러링 및 NetServer를 사용하여 다중 인스턴스 큐 관리자 작성』의 내용을 참조하십시오.
2. 정의되는 원격 저널을 IBM MQ가 삭제할 수 있도록 ALPHA와 BETA를 연결해야 합니다.
3. 시스템 명령 **EDTF** 또는 **WRKLNK**를 사용하여 /QNTC 디렉토리 및 서버 디렉토리 파일 공유에 액세스할 수 있는지 확인하십시오.

이 태스크 정보

DLTMQM 명령을 사용하여 서버에서 다중 인스턴스 큐 관리자를 삭제하려면 **RMVMQMINF** 명령을 사용하여 다른 서버의 큐 관리자 인스턴스를 제거하십시오.

RMVMQMINF 명령을 사용하여 큐 관리자 인스턴스를 제거하면 AMQ가 앞에 오고 인스턴스와 연관된 로컬 및 원격 저널이 삭제됩니다. 서버에 로컬인 큐 관리자 인스턴스에 대한 구성 정보도 삭제됩니다.

큐 관리자의 나머지 인스턴스가 있는 서버에서는 **RMVMQMINF** 명령을 실행하지 마십시오. 명령을 실행하면 **DLTMQM**이 올바르게 작동하지 않습니다.

DLTMQM 명령을 사용하여 큐 관리자를 삭제하십시오. 네트워크 공유에서 큐 관리자 데이터가 제거됩니다. 앞에 AMQ가 표시되고 인스턴스와 연관된 로컬 및 원격 저널은 삭제됩니다. **DLTMQM**은 또한 서버에 로컬인 큐 관리자 인스턴스에 대한 구성 정보를 삭제합니다.

예에는 큐 관리자 인스턴스가 두 가지만 있습니다. IBM MQ는 하나의 활성 큐 관리자 인스턴스와 하나의 대기 인스턴스가 있는 실행 다중 인스턴스 구성을 지원합니다. 실행 구성에서 사용할 추가 큐 관리자 인스턴스를 작성한 경우, 나머지 인스턴스를 삭제하기 전에 **RMVMQMINF** 명령을 사용하여 제거하십시오.

프로시저

1. 각 서버에서 **CHGMQMJRN RMTJRNSTS** (*INACTIVE) 명령을 실행하여 큐 관리자 인스턴스 간에 원격 저널링을 비활성화하십시오.

   a) ALPHA:

   ```
   CHGMQMJRN MQMNAME('QM1')
   RMTJRNRDB('BETA') RMTJRNSTS(*INACTIVE)
   ```

   b) BETA:

   ```
   CHGMQMJRN MQMNAME('QM1')
   RMTJRNRDB('ALPHA') RMTJRNSTS(*INACTIVE)
   ```

2. 활성 큐 관리자 인스턴스인 ALPHA에서 **ENDMQM** 명령을 실행하여 QM1 인스턴스를 둘 다 중지하십시오.

   ```
   ENDMQM MQMNAME(QM1) OPTION(*IMMED) INSTANCE(*ALL) ENDCCTJOB(*YES)
   ```

3. ALPHA에서 **RMVMQMINF** 명령을 실행하여 ALPHA 및 BETA에서 인스턴스에 대한 큐 관리자 자원을 제거하십시오.

   ```
   RMVMQMINF MQMNAME(QM1)
   ```

   RMVMQMINF는 QM1에 대한 큐 관리자 구성 정보를 ALPHA에서 제거합니다. 저널 이름 앞에 AMQ가 있으면 QM1과 연관된 로컬 저널을 ALPHA에서 삭제합니다. 저널 이름 앞에 AMQ가 오고 원격 저널이 작성된 경우에는 BETA에서도 원격 저널을 제거합니다.

4. BETA에서 **DLTMQM** 명령을 실행하여 QM1을 삭제하십시오.

   ```
   DLTMQM MQMNAME(QM1)
   ```

   DLTMQM은 GAMMA의 네트워크 공유에서 큐 관리자 데이터를 삭제합니다. BETA에서 QM1에 대한 큐 관리자 구성 정보를 제거합니다. 저널 이름 앞에 AMQ가 있고 QM1과 연관된 로컬 저널을 BETA에서 삭제합니다. 저널 이름 앞에 AMQ가 오고 원격 저널이 작성된 경우에는 ALPHA에서도 원격 저널을 제거합니다.

결과

DLTMQM과 **RMVMQMINF**는 **CRTMQM**과 **ADDMQJRN**으로 작성된 로컬 및 원격 저널을 삭제합니다. 이 명령은 또한 저널 수신자를 삭제합니다. 저널과 저널 수신자는 이름이 AMQ로 시작하는 이름 지정 규칙을 따라야 합니다.

DLTMQM과 **RMVMQMINF**는 mqs.ini에서 큐 관리자 오브젝트, 큐 관리자 데이터, 큐 관리자 구성 정보를 제거합니다.

다음에 수행할 작업

또 다른 방법은 226 페이지의 『1』 단계에서 저널링을 비활성화한 후 및 큐 관리자 인스턴스를 종료하기 전에 다음 명령을 실행하는 것입니다. 또는 이름 지정 규칙을 따르지 않은 경우 이름별로 저널과 저널 수신자를 삭제해야 합니다.

1. ALPHA:

```
RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('BETA')
```

2. BETA:

```
RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('ALPHA')
```

저널을 삭제한 후 나머지 단계를 계속 진행하십시오.

다중 인스턴스 큐 관리자 백업

이 프로시저는 로컬 서버의 큐 관리자 오브젝트와 네트워크 파일 서버의 큐 관리자 데이터를 백업하는 방법을 보여줍니다. 다른 큐 관리자의 데이터를 백업하려면 예를 수정하십시오.

시작하기 전에

이 예에서 큐 관리자 QM1과 연관된 큐 관리자 데이터는 NetServer를 사용하여 IBM i 서버 GAMMA에 저장됩니다. 212 페이지의 『저널 미러링 및 NetServer를 사용하여 다중 인스턴스 큐 관리자 작성』의 내용을 참조하십시오. IBM MQ는 서버, ALPHA 및 BETA에 설치됩니다. 큐 관리자, QM1은 ALPHA와 BETA에 구성됩니다.

이 태스크 정보

IBM i는 원격 디렉토리로부터의 데이터 저장을 지원하지 않습니다. 파일 시스템 서버의 로컬 백업 프로시저를 사용하여 원격 파일 시스템에 큐 관리자 데이터를 저장하십시오. 이 태스크에서 네트워크 파일 시스템은 IBM i 서버, GAMMA에 있습니다. 큐 관리자 데이터는 GAMMA의 저장 파일에 백업됩니다.

네트워크 파일 시스템이 Windows 또는 Linux에 있는 경우에는 압축 파일에 큐 관리자 데이터를 저장한 다음 저장할 수 있습니다. Tivoli Storage Manager와 같은 백업 시스템이 있는 경우에는 해당 시스템을 사용하여 큐 관리자 데이터를 백업하십시오.

프로시저

1. QM1과 연관된 큐 관리자 라이브러리의 경우 ALPHA에 저장 파일을 작성하십시오.

   큐 관리자 라이브러리 이름을 사용하여 저장 파일 이름을 지정하십시오.

```
CRTSAVF FILE(QGPL/QMQM1)
```

2. ALPHA의 저장 파일에 큐 관리자 라이브러리를 저장하십시오.

```
SAVLIB LIB(QMQM1) DEV(*SAVF) SAVF(QGPL/QMQM1)
```

3. GAMMA의 큐 관리자 데이터 디렉토리에 대한 저장 파일을 작성하십시오.

   큐 관리자 이름을 사용하여 저장 파일 이름을 지정하십시오.

```
CRTSAVF FILE(QGPL/QMDQM1)
```

4. GAMMA의 로컬 디렉토리에서 큐 관리자 데이터의 사본을 저장하십시오.

```
SAV DEV('/QSYS.LIB/QGPL.LIB/QMDQM1.FILE') OBJ('/QIBM/Userdata/mqm/qmgrs/QM1')
```

다중 인스턴스 큐 관리자를 설정하는 명령

IBM MQ에는 저널 복제 구성, 새 큐 관리자 인스턴스 추가, 큐 관리자가 독립 ASP를 사용하도록 구성 작업을 단순화하는 명령이 있습니다.

로컬 및 원격 저널을 작성하고 관리하는 저널 명령은 다음과 같습니다.

ADDMQMJRN
이 명령을 사용하면 큐 관리자 인스턴스의 이름 지정된 로컬 및 원격 저널을 작성하고 복제가 동기 또는 비동기인지 여부, 동기 제한시간 초과한 것, 원격 저널이 즉시 활성화될 수 있는지 여부를 구성할 수 있습니다.

CHGMQMJRN
이 명령은 복제 저널에 영향을 주는 제한시간, 상태, 전달 매개변수를 수정합니다.

RMVMQMJRN
큐 관리자 인스턴스에서 이름 지정된 원격 저널을 제거합니다.

WRKMQMJRN
로컬 큐 관리자 인스턴스의 로컬 및 원격 저널 상태를 나열합니다.

`mqs.ini` 파일을 수정하는 다음 명령을 사용하여 추가 큐 관리자 인스턴스를 추가하고 관리하십시오.

ADDMQMINF
이 명령은 DSPMQMINF 명령으로 `mqs.ini` 파일에서 추출한 정보를 사용하여 다른 IBM i 서버에서 새 큐 관리자 인스턴스를 추가합니다.

RMVMQMINF
큐 관리자 인스턴스를 제거합니다. 기존 큐 관리자의 인스턴스를 제거하거나 다른 서버에서 삭제된 큐 관리자에 대한 구성 정보를 제거하려면 이 명령을 사용하십시오.

CRTMQM 명령은 다중 인스턴스 큐 관리자 구성을 지원하기 위한 세 가지 매개변수를 갖습니다.

MQMDIRP (*DFT | *directory-prefix*)
네트워크 스토리지의 큐 관리자 데이터에 맵핑되는 마운트 포인트를 선택하려면 이 매개변수를 사용하십시오.

ASP (*SYSTEM | *ASPDEV | *auxiliary-storage-pool-number*)
시스템 또는 기본 사용자 ASP에 큐 관리자 저널을 배치하려면 *SYSTEM 또는 *auxiliary-storage-pool-number*를 지정하십시오. 독립 ASP에 큐 관리자 저널을 배치하려면 *ASPDEV 옵션을 선택하고 또한 **ASPDEV** 매개변수를 사용하여 디바이스 이름을 설정하십시오.

ASPDEV (*ASP | *device-name*)
1차 또는 2차 독립 ASP 디바이스의 *device-name*를 지정하십시오. *ASP를 선택한 결과는 **ASP** (*SYSTEM)을 지정하는 경우와 같습니다.

성능 및 디스크 장애 복구 고려사항

다양한 보조 기억장치 풀을 사용하여 성능과 신뢰성을 향상시킬 수 있습니다.

애플리케이션에서 많은 지속 메시지 또는 대형 메시지를 사용하는 경우 이러한 메시지를 디스크에 쓰는 데 소요되는 시간이 시스템 성능을 결정하는 중대 요인이 됩니다.

이러한 가능성에 대응할 수 있을 정도로 디스크가 충분하게 활성화되었는지 확인하거나 큐 관리자 저널 수신자를 저장하는 별도 보조 기억장치 풀(ASP)을 고려하십시오.

CRTMQM의 ASP 매개변수를 사용하여 큐 관리자를 작성할 때 큐 관리자 라이브러리와 저널이 저장되는 ASP를 지정할 수 있습니다. 기본적으로 큐 관리자 라이브러리와 저널 및 IFS 데이터는 시스템 ASP에 저장됩니다.

ASP는 하나 이상의 특정 디스크 장치에서 오브젝트 격리를 허용합니다. 따라서 디스크 매체 고장으로 인한 데이터 손실을 줄일 수도 있습니다. 대부분의 경우, 영향을 받는 ASP의 디스크 장치에 저장된 데이터만 손실됩니다.

장애 복구를 제공하고 디스크 경합을 줄이려면 별도 사용자 ASP의 큐 관리자 라이브러리 및 저널 데이터를 루트 IFS 파일 시스템의 해당 위치에 저장하는 것이 좋습니다.

자세한 정보는 백업 및 복구를 참조하십시오.

SAVLIB를 사용하여 IBM MQ 라이브러리 저장

You cannot use SAVLIB LIB(*ALLUSR) to save the IBM MQ libraries, because these libraries have names beginning with Q.

SAVLIB LIB(QM*)을 사용하여 모든 큐 관리자 라이브러리를 저장하려면 *SAVF 이외의 저장 디바이스를 사용해야만 합니다. DEV(*SAVF)의 경우, 시스템의 각각 모든 큐 관리자 라이브러리에 SAVLIB 명령을 사용해야 합니다.

IBM MQ for IBM i 일시정지

이 절에서는 IBM MQ for IBM i를 일시정지하는 방법(우아한 종료)을 설명합니다.

IBM MQ for IBM i를 일시정지하려면 다음을 수행하십시오.

1. 오브젝트에 액세스하지 않은 상태로 새 대화식 IBM MQ for IBM i 세션에 사인온하십시오.
2. 다음 권한이 있는지 확인하십시오.
   - QMQM 라이브러리에 대한 오브젝트 관리 권한 또는 *ALLOBJ 권한
   - ENDSBS 명령을 사용할 수 있는 충분한 권한
3. 모든 사용자에게 IBM MQ for IBM i를 중지할 예정임을 알리십시오.
4. 다음 진행 단계는 단일 큐 관리자(다른 큐 관리자가 존재할 수 있는 경우)(229 페이지의 『IBM MQ for IBM i의 단일 큐 관리자 종료』 참조) 또는 모든 큐 관리자(231 페이지의 『IBM MQ for IBM i의 모든 큐 관리자 종료』 참조)를 종료(일시정지)하려는지에 따라 결정됩니다.

ENDMQM 매개변수 ENDCCTJOB(*YES)

ENDMQM 매개변수 ENDCCTJOB(*YES)는 이전 버전과 비교할 때 IBM MQ for IBM i V6.0 이상에서 다르게 작동합니다.

이전 버전의 경우 ENDCCTJOB(*YES)를 지정하면 MQ가 애플리케이션을 강제로 종료합니다.

IBM MQ for IBM i V6.0 이상의 경우 ENDCCTJOB(*YES)를 지정하면 애플리케이션은 종료되지 않지만 대신 큐 관리자와의 연결이 끊어집니다.

ENDCCTJOB(*YES)을 지정하고 큐 관리자가 종료 중임을 감지하기 위해 작성되지 않은 애플리케이션이 있는 경우, 다음에 새 MQI 호출이 발행되면 호출은 MQRC_CONNECTION_BROKEN(2009) 오류를 리턴합니다.

ENDCCTJOB(*YES) 사용에 대한 대안으로, ENDCCTJOB(*NO) 매개변수와 WRKMQM 옵션 22(작업 처리)를 사용하여 큐 관리자가 재시작되는 것을 방해하는 애플리케이션 작업을 수동으로 종료합니다.

IBM MQ for IBM i의 단일 큐 관리자 종료

이 정보를 통해 세 가지 종료 유형을 이해할 수 있습니다.

다음 프로시저에서는 샘플 큐 관리자 이름 QMgr1과 샘플 서브시스템 이름 SUBX를 사용합니다. 필요한 경우 이 이름을 원하는 값으로 바꿀 수 있습니다.

계획된 종료

IBM i에서의 계획된 큐 관리자 종료

1. 종료하기 전에 다음을 실행하십시오.

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

2. 큐 관리자를 종료하려면 다음을 실행하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

   QMgr1이 종료되지 않으면 채널 또는 애플리케이션 사용량이 증가할 수 있습니다.

3. QMgr1을 즉시 종료해야 하는 경우에는 다음을 실행하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

계획되지 않은 종료

1. 큐 관리자를 종료하려면 다음을 실행하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   QMgr1이 종료되지 않으면 채널 또는 애플리케이션 사용량이 증가할 수 있습니다.
2. QMgr1을 즉시 종료해야 하는 경우에는 다음을 실행하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

비정상적인 조건하에서의 종료

1. 큐 관리자를 종료하려면 다음을 실행하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   다음과 같은 경우 QMgr1이 종료되지 않으면 단계 3을 계속 진행하십시오.

   - QMgr1이 자체 서브시스템 내에 있습니다. 또는
   - QMgr1과 동일한 서브시스템을 공유하는 모든 큐 관리자를 종료할 수 있습니다. 그러한 모든 큐 관리자에 대해 계획되지 않은 종료 프로시저를 사용하십시오.
2. 서브시스템(예에서 SUBX)을 공유하는 모든 큐 관리자에 대해 프로시저의 모든 단계를 수행한 경우 다음을 실행하십시오.

```
ENDSBS SUBX *IMMED
```

   이 명령이 완료되지 않으면 계획되지 않은 종료 프로시저를 사용하여 모든 큐 관리자를 종료하고 시스템에서 IPL을 수행하십시오.

   경고: ENDJOB 또는 ENDSBS의 결과로 종료되지 않는 IBM MQ 작업에는 ENDJOBABN 를 사용하지 마십시오. 즉시 시스템에서 IPL을 수행할 준비가 되어 있지 않은 경우에는 사용하지 마십시오.
3. 다음을 실행하여 서브시스템을 시작하십시오.

```
STRSBS SUBX
```

4. 다음을 실행하여 큐 관리자를 즉시 종료하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(10)
```

5. 다음을 실행하여 큐 관리자를 재시작하십시오.

```
STRMQM MQMNAME(QMgr1)
```

   실패하는 경우 원인은 다음과 같습니다.

   - IPL을 수행하여 시스템을 다시 시작했습니다. 또는
   - 단일 큐 관리자만 있습니다.

다음을 실행하여 IBM MQ 공유 메모리를 정리하십시오.

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

단계 5를 반복하기 전에

큐 관리자를 재시작하는 데 몇 초 이상이 소요되는 경우 IBM MQ가 시동 진행 상황을 설명하는 상태 메시지를 작업 로그에 간헐적으로 추가합니다.

큐 관리자 재시작 문제점이 계속 해결되지 않으면 IBM 지원 부서에 문의하십시오. 추가 조치를 수행하는 경우 큐 관리자가 손상되어 IBM MQ를 복구하지 못할 수 있습니다.

IBM MQ for IBM i의 모든 큐 관리자 종료

이 정보를 통해 세 가지 종료 유형을 이해할 수 있습니다.

이 프로시저는 단일 큐 관리자의 경우와 거의 동일하지만 가능한 경우 큐 관리자 이름 대신 *ALL을 사용하거나 각 큐 관리자 이름을 차례로 사용하여 명령을 반복적으로 사용합니다. 프로시저 전체에서 샘플 큐 관리자 이름 QMgr1과 샘플 서브시스템 이름 SUBX를 사용합니다. 이 이름은 원하는 대로 바꿀 수 있습니다.

계획된 종료

1. 종료 1시간 전에 다음을 실행하십시오.

   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
   ```

   종료하려는 모든 큐 관리자에 이 작업을 반복하십시오.
2. 큐 관리자를 종료하려면 다음을 실행하십시오.

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
   ```

   종료하려는 모든 큐 관리자에 이 작업을 반복하십시오. 개별 명령을 동시에 실행할 수 있습니다.

   큐 관리자가 적정 시간(예를 들어, 10분) 내에 종료되지 않으면 단계 3으로 진행하십시오.
3. 모든 큐 관리자를 즉시 종료하려면 다음을 실행하십시오.

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

계획되지 않은 종료

1. 큐 관리자를 종료하려면 다음을 실행하십시오.

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   종료하려는 모든 큐 관리자에 이 작업을 반복하십시오. 개별 명령을 동시에 실행할 수 있습니다.

   큐 관리자가 종료되지 않으면 채널 또는 애플리케이션 사용량이 증가할 수 있습니다.
2. 큐 관리자를 즉시 종료해야 하는 경우에는 다음을 실행하십시오.

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

비정상적인 조건하에서의 종료

1. 큐 관리자를 종료하려면 다음을 실행하십시오.

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

종료하려는 모든 큐 관리자에 이 작업을 반복하십시오. 개별 명령을 동시에 실행할 수 있습니다.

2. 다음을 실행하여 서브시스템(예의 SUBX)을 종료하십시오.

```
ENDSBS SUBX *IMMED
```

종료하려는 모든 서브시스템에 이 작업을 반복하십시오. 개별 명령을 동시에 실행할 수 있습니다.

이 명령이 완료되지 않으면 시스템에서 IPL을 수행하십시오.

경고: 시스템에서 즉시 IPL을 수행할 수 있도록 준비되지 않은 경우에는 ENDJOB 또는 ENDSBS의 결과로 종료되지 않는 작업에 ENDJOBABN을 사용하지 마십시오.

3. 다음을 실행하여 서브시스템을 시작하십시오.

```
STRSBS SUBX
```

시작하려는 모든 서브시스템에 이 작업을 반복하십시오.

4. 다음을 실행하여 큐 관리자를 즉시 종료하십시오.

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

5. 다음을 실행하여 큐 관리자를 다시 시작하십시오.

```
STRMQM MQMNAME(QMgr1)
```

시작하려는 모든 큐 관리자에 이 작업을 반복하십시오.

큐 관리자 재시작을 위해 몇 초 이상이 소요되는 경우 IBM MQ는 시작 진행 상황을 나타내는 상태 메시지를 간헐적으로 표시합니다.

큐 관리자 재시작 문제점이 계속 해결되지 않으면 IBM 지원 부서에 문의하십시오. 추가 조치를 수행하는 경우 큐 관리자가 손상되어 MQSeries 또는 IBM MQ를 복구하지 못할 수 있습니다.

z/OS IBM MQ for z/OS 관리

큐 관리자 및 연관된 자원 관리에는 해당 자원을 활성화하고 관리하기 위해 자주 수행하는 태스크가 포함됩니다. 큐 관리자 및 연관된 자원을 관리할 때 선호하는 메소드를 선택하십시오.

IBM MQ for z/OS는 제품과 함께 제공되는 유틸리티 및 프로그램 세트로 제어 및 관리할 수 있습니다. MQSC(IBM MQ Script) 명령 또는 프로그래밍 가능 명령 형식(PCF)을 사용하여 IBM MQ for z/OS를 관리할 수 있습니다. IBM MQ for z/OS의 명령 사용에 대한 정보는 233 페이지의 『IBM MQ for z/OS에 명령 실행』의 내용을 참조하십시오.

IBM MQ for z/OS는 또한 시스템 관리를 도와줄 유틸리티 프로그램 세트를 제공합니다. 여러 유틸리티 프로그램 및 사용 방법에 대한 정보는 240 페이지의 『IBM MQ for z/OS 유틸리티』의 내용을 참조하십시오.

IBM MQ for z/OS 관리 방법 및 수행해야 하는 여러 관리 태스크에 대한 자세한 내용은 다음 링크를 참조하십시오.

관련 개념

64 페이지의 『로컬 IBM MQ 오브젝트 관리』
이 섹션은 메시지 큐 인터페이스(MQI)를 사용하는 애플리케이션 프로그램을 지원하기 위해 로컬 IBM MQ 오브젝트를 관리하는 방법에 대해 알려줍니다. 이 컨텍스트에서 로컬 관리는 IBM MQ 오브젝트 작성, 표시, 변경, 복사 및 삭제를 의미합니다.

118 페이지의 『원격 IBM MQ 오브젝트 관리』

5 페이지의 『IBM MQ 관리』

큐 관리자 및 연관된 자원 관리에는 해당 자원을 활성화하고 관리하기 위해 자주 수행하는 태스크가 포함됩니다. 큐 관리자 및 연관된 자원을 관리할 때 선호하는 메소드를 선택하십시오.

관련 정보

IBM MQ for z/OS 개념
계획 중
z/OS에서 IBM MQ 환경 계획
구성
z/OS 구성
프로그래밍 가능한 명령 포맷 참조
MQSC 참조
IBM MQ for z/OS 유틸리티 사용

IBM MQ for z/OS에 명령 실행

IBM MQ 스크립트 명령(MQSC)을 배치 또는 대화식 모드로 사용하여 큐 관리자를 제어할 수 있습니다.

IBM MQ for z/OS에서는 다음 소스에서 실행할 수 있는 MQSC 명령을 지원합니다.

- z/OS 콘솔 또는 다른 해당 콘솔(예: SDSF/TSO)
- 초기화 입력 데이터 세트
- 순차적 데이터 세트에서 명령 목록을 처리하는 제공된 배치 유틸리티, CSQUTIL
- 명령 입력 큐에 메시지로 명령을 송신하여 적합한 권한이 부여된 애플리케이션. 애플리케이션은 다음 중 하나일 수 있습니다.
  - 배치 리젼 프로그램
  - CICS 애플리케이션
  - IMS 애플리케이션
  - TSO 애플리케이션
  - 다른 IBM MQ 시스템의 애플리케이션 프로그램 또는 유틸리티

236 페이지의 표 16에는 실행할 수 있는 MQSC 명령 및 소스가 요약되어 있습니다.

이러한 명령의 기능은 상당수 IBM MQ for z/OS 운영 및 제어판에서 편리하게 사용할 수 있습니다.

명령을 사용한 큐 관리자의 자원 정의 변경 사항(직접 또는 간접)은 IBM MQ 서브시스템을 다시 시작해도 유지됩니다.

IBM MQ for z/OS는 또한 프로그래밍 가능 명령 형식(PCF) 명령을 지원합니다. 이러한 명령은 IBM MQ 관리를 위한 애플리케이션 작성을 단순화합니다. MQSC 명령은 사용자가 읽을 수 있는 텍스트 양식이지만, PCF를 사용하면 애플리케이션이 텍스트 문자열을 구문 분석하지 않고도 요청을 작성하고 응답을 읽을 수 있습니다. MQSC 명령과 마찬가지로 애플리케이션은 명령 입력 큐에 메시지로 이들을 송신하여 PCF 명령을 실행합니다. PCF 명령 사용에 대한 자세한 정보 및 명령의 자세한 내용은 프로그램 가능한 명령 형식 참조 문서를 참조하십시오.

개인용 및 글로벌 정의

IBM MQ for z/OS에서 오브젝트를 정의할 때 해당 정의를 다른 큐 관리자와 공유할지(글로벌 정의) 또는 오브젝트 정의를 한 명의 큐 관리자만 사용하는지(개인용 정의) 여부를 선택할 수 있습니다. 이를 오브젝트 속성 지정이라고 합니다.

글로벌 정의

큐 관리자가 큐 공유 그룹에 속하는 경우 그룹의 다른 멤버에서 작성한 오브젝트 정의를 공유하도록 선택할 수 있습니다. 즉, 전체 시스템에 필요한 총 정의 수를 줄여 한 번만 오브젝트를 정의해야 함을 의미합니다.

글로벌 오브젝트 정의는 공유 저장소(Db2® 공유 데이터베이스)에 저장되며 큐 공유 그룹의 모든 큐 관리자가 사용할 수 있습니다. 이러한 오브젝트에는 GROUP 속성 지정이 있습니다.

개인용 정의

하나의 큐 관리자에서만 필요한 오브젝트 정의를 작성하려는 경우 또는 큐 관리자가 큐 공유 그룹의 멤버가 아닌 경우 큐 공유 그룹의 다른 멤버와 공유되지 않는 오브젝트 정의를 작성할 수 있습니다.

개인용 오브젝트 정의는 정의하는 큐 관리자의 페이지 세트 0에 보유됩니다. 이러한 오브젝트에는 QMGR 속성 지정이 있습니다.

CF 구조를 제외한 모든 IBM MQ 오브젝트 유형(즉, 채널, 이름 목록, 프로세스 정의, 큐, 큐 관리자, 스토리지 클래스 정의, 인증 정보 오브젝트)에는 개인용 정의를 작성하고 큐 관리자를 제외한 모든 오브젝트 유형에는 글로벌 정의를 작성할 수 있습니다.

IBM MQ는 그룹 오브젝트 정의를 해당 정의를 사용하는 각 큐 관리자의 페이지 세트 0에 자동으로 복사합니다. 원하는 경우 정의 사본을 임시로 대체하고 필요한 경우 IBM MQ를 사용하여 저장소 사본에서 페이지 세트 사본을 새로 고칠 수 있습니다.

IBM MQ는 시작 시 또는 그룹 오브젝트가 변경되는 경우 항상 저장소 사본에서 페이지 세트 사본 새로 고치기를 시도합니다. 채널 명령의 경우에는 채널 이니시에이터가 다시 시작될 때 이를 수행합니다.

참고: 정의 사본은 정의 사본을 작성한 후 그룹 정의가 변경된 경우에만 그룹 정의에서 새로 고쳐집니다.

이를 통해 큐 관리자가 비활성 상태일 때 수행된 변경사항을 포함한 저장소의 버전이 페이지 세트 사본에 반영됩니다. 사본은 DEFINE REPLACE 명령을 생성하여 새로 고쳐지므로 새로 고치기가 수행되지 않는 경우가 있습니다. 예를 들어 다음과 같습니다.

- 큐의 사본이 열린 경우 큐 사용을 변경하는 새로 고치기에 실패합니다.
- 큐의 사본에 이에 대한 메시지가 있는 경우 해당 큐를 삭제하는 새로 고치기에 실패합니다.
- 큐의 사본에서 이를 변경하려면 FORCE를 포함하는 ALTER가 필요합니다.

이러한 상황에서 새로 고치기는 해당 사본에서 수행되지 않으며 다른 모든 큐 관리자의 사본에서 수행됩니다.

큐 관리자가 종료된 후 독립적으로 재시작되면 예를 들어 큐에 연관된 메시지가 없는 한, 오브젝트의 로컬 사본이 삭제됩니다.

로컬 큐에만 적용되는 세 번째 오브젝트 속성 지정이 있습니다. 이를 통해 공유 큐를 작성할 수 있습니다. 공유 큐의 정의는 공유 저장소에 보유되며 큐 공유 그룹의 모든 큐 관리자에서 사용 가능합니다. 또한 공유 큐의 메시지는 큐 공유 그룹의 모든 큐 관리자에서도 사용 가능합니다. 이 내용은 공유 큐 및 큐 공유 그룹에서 설명됩니다. 공유 큐에는 SHARED 오브젝트 속성 지정이 있습니다.

다음 표에서는 큐 공유 그룹의 멤버로, 큐 관리자가 시작된 독립형에 대한 오브젝트 속성 지정 옵션의 영향을 요약하여 보여줍니다.

처리	독립형 큐 관리자	큐 공유 그룹의 구성원
큐 관리자	페이지 세트 0에 보유되는 오브젝트 정의입니다.	페이지 세트 0에 보유되는 오브젝트 정의입니다.
GROUP	허용되지 않습니다.	공유 저장소에 보유되는 오브젝트 정의입니다. 그룹에서 각 큐 관리자의 페이지 세트 0에 보유되는 로컬 사본입니다.
SHARED	허용되지 않습니다.	공유 저장소에 보유된 큐 정의입니다. 그룹의 큐 관리자에서 사용 가능한 메시지입니다.

글로벌 정의 조작

공유 저장소에 저장된 오브젝트의 정의를 변경하려는 경우 페이지 세트 0에서 로컬 사본 또는 저장소의 버전을 변경할지 여부를 지정해야 합니다. 명령의 일부로 오브젝트 속성 지정을 사용하여 이를 수행하십시오.

서로 다른 큐 관리자에 명령 전달

명령 범위를 사용하여 명령이 실행되는 큐 관리자를 제어할 수 있습니다.

큐 공유 그룹에서 서로 다른 큐 관리자 또는 명령이 입력된 큐 관리자에서 명령을 실행하도록 선택할 수 있습니다. 또한 큐 공유 그룹에 있는 모든 큐 관리자에서 병렬로 특정 명령을 실행하도록 선택할 수 있습니다. MQSC 명령 및 PCF 명령에 모두 가능합니다.

이는 명령 범위로 판별됩니다. 명령 범위는 오브젝트 속성 지정과 함께 사용되어 사용하려는 오브젝트의 버전을 판별합니다.

예를 들어 공유 저장소에 보유된 항목에 대한 정의와 같은 오브젝트에 대한 일부 속성을 대체할 수 있습니다.

- 하나의 큐 관리자에서만 버전을 변경하고 저장소의 버전이나 다른 큐 관리자가 사용하는 버전은 변경하지 않을 수 있습니다.
- 향후 사용자를 위해 공유 저장소에서 버전을 변경하지만 기존 사본은 그대로 둘 수 있습니다.
- 공유 저장소에서 버전을 변경할 수 있뿐 아니라 해당 페이지 세트 0에서 오브젝트의 사본을 보유하는 큐 공유 그룹의 모든 큐 관리자에서 변경사항을 즉시 반영할 수 있습니다.

명령 범위를 사용하여 해당 큐 관리자, 다른 큐 관리자 또는 모든 큐 관리자에서 명령이 실행되는지 여부를 지정하십시오. 오브젝트 속성 지정을 사용하여 조작하는 오브젝트가 공유 저장소(그룹 오브젝트)에 있는지 또는 페이지 세트 0에 있는 로컬 사본(큐 관리자 오브젝트)인지 여부를 지정하십시오.

큐 공유 그룹의 모든 큐 관리자가 단일 큐로 공유 큐를 처리하므로 공유 큐에서 작동하기 위해 명령 범위 및 오브젝트 속성 지정을 지정하지 않아도 됩니다.

명령 요약

이 주제는 기본 MQSC 및 PCF 명령의 참조로 사용하십시오.

235 페이지의 표 15 은 IBM MQ 오브젝트를 변경, 정의, 삭제 및 표시하기 위해 IBM MQ for z/OS 에서 사용 가능한 MQSC및 PCF 명령을 요약합니다.

표 *15.* 오브젝트 유형별 기본 *MQSC* 및 *PCF* 명령 요약

MQSC 명령어	**ALTER**	**DEFINE**	**DISPLAY**	**DELETE**
PCF 명령	**변경**	**작성/복사**	**조회**	**삭제**
AUTHINFO	X	X	X	X
CFSTATUS			X	
CFSTRUCT	X	X	X	X
CHANNEL	X	X	X	X
CHSTATUS			X	
이름 목록	X	X	X	X
PROCESS	X	X	X	X
QALIAS	M	M	M	M
QCLUSTER			M	
QLOCAL	M	M	M	M
큐 관리자	X		X	
QMODEL	M	M	M	M
QREMOTE	M	M	M	M
큐	P	P	X	P
QSTATUS			X	
STGCLASS	X	X	X	X

표 기호 키:

- M = MQSC만 해당
- P = PCF만 해당
- X = 둘 다 해당

다른 IBM MQ 자원을 관리하고 235 페이지의 표 15에 요약되어 있는 것 이외에 추가 조치를 수행할 수 있는 많은 다른 MQSC 및 PCF 명령이 있습니다.

236 페이지의 표 16는 모든 MQSC 명령과 각 명령을 실행할 수 있는 위치를 보여줍니다.

- CSQINP1 초기화 입력 데이터 세트
- CSQINP2 초기화 입력 데이터 세트
- z/OS 콘솔(또는 이에 상응)
- SYSTEM.COMMAND.INPUT 큐 및 명령 서버(애플리케이션, CSQUTIL 또는 CSQINPX 초기화 입력 데이터 세트에 있음)

표 *16. MQSC* 명령을 실행하는 소스

명령	**CSQINP1**	**CSQINP2**	**z/OS 콘솔**	**명령 입력 큐와 서버**
ALTER AUTHINFO		X	X	X
ALTER BUFFPOOL		X	X	X
ALTER CFSTRUCT		X	X	X
ALTER CHANNEL		X	X	X
ALTER NAMELIST		X	X	X
ALTER PSID			X	X
ALTER PROCESS		X	X	X
ALTER QALIAS		X	X	X
ALTER QLOCAL		X	X	X
ALTER QMGR		X	X	X
ALTER QMODEL		X	X	X
ALTER QREMOTE		X	X	X
ALTER SECURITY	X	X	X	X
ALTER STGCLASS		X	X	X
ALTER SUB		X	X	X
ALTER TOPIC		X	X	X
ALTER TRACE	X	X	X	X
ARCHIVE LOG	X	X	X	X
BACKUP CFSTRUCT			X	X
CLEAR QLOCAL		X	X	X
DEFINE AUTHINFO		X	X	X
DEFINE BUFFPOOL	X	X		
DEFINE CFSTRUCT		X	X	X
DEFINE CHANNEL		X	X	X

표 *16. MQSC* 명령을 실행하는 소스 *(계속)*

명령	CSQINP1	CSQINP2	z/OS 콘솔	명령 입력 큐와 서버
DEFINE LOG			X	X
DEFINE NAMELIST		X	X	X
DEFINE PROCESS		X	X	X
DEFINE PSID	X		X	X
DEFINE QALIAS		X	X	X
DEFINE QLOCAL		X	X	X
DEFINE QMODEL		X	X	X
DEFINE QREMOTE		X	X	X
DEFINE STGCLASS		X	X	X
DEFINE SUB			X	X
DEFINE TOPIC		X	X	X
DELETE AUTHINFO		X	X	X
DELETE BUFFPOOL			X	X
DELETE CFSTRUCT		X	X	X
DELETE CHANNEL			X	X
DELETE NAMELIST		X	X	X
DELETE PROCESS		X	X	X
DELETE PSID			X	X
DELETE QALIAS		X	X	X
DELETE QLOCAL		X	X	X
DELETE QMODEL		X	X	X
DELETE QREMOTE		X	X	X
DELETE STGCLASS		X	X	X
DELETE SUB		X	X	X
DELETE TOPIC		X	X	X
DISPLAY ARCHIVE	X	X	X	X
DISPLAY AUTHINFO		X	X	X
DISPLAY CFSTATUS			X	X
DISPLAY CFSTRUCT		X	X	X
DISPLAY CHANNEL		X	X	X
DISPLAY CHSTATUS			X	X
DISPLAY CLUSQMGR			X	X
DISPLAY CMDSERV	X	X	X	X
DISPLAY CONN		X	X	X

표 *16. MQSC* 명령을 실행하는 소스 *(계속)*

명령	CSQINP1	CSQINP2	z/OS 콘솔	명령 입력 큐와 서버
DISPLAY CHINIT		X	X	X
DISPLAY GROUP		X	X	X
DISPLAY LOG	X	X	X	X
DISPLAY NAMELIST		X	X	X
DISPLAY PROCESS		X	X	X
DISPLAY QALIAS		X	X	X
DISPLAY QCLUSTER		X	X	X
DISPLAY QLOCAL		X	X	X
DISPLAY QMGR		X	X	X
DISPLAY QMODEL		X	X	X
DISPLAY QREMOTE		X	X	X
DISPLAY QSTATUS		X	X	X
DISPLAY QUEUE		X	X	X
DISPLAY SECURITY			X	X
DISPLAY STGCLASS		X	X	X
DISPLAY SUB		X	X	X
DISPLAY TOPIC		X	X	X
DISPLAY SYSTEM	X	X	X	X
DISPLAY THREAD		X	X	X
DISPLAY TRACE	X	X	X	X
DISPLAY USAGE		X	X	X
MOVE QLOCAL		X	X	X
PING CHANNEL			X	X
RECOVER BSDS	X	X	X	X
RECOVER CFSTRUCT			X	X
REFRESH CLUSTER		X	X	X
REFRESH QMGR		X	X	X
REFRESH SECURITY		X	X	X
RESET CHANNEL			X	X
RESET CLUSTER		X	X	X
RESET QSTATS		X	X	X
RESET TPIPE			X	X
RESOLVE CHANNEL			X	X
RESOLVE INDOUBT		X	X	X

표 16. *MQSC* 명령을 실행하는 소스 *(계속)*

명령	CSQINP1	CSQINP2	z/OS 콘솔	명령 입력 큐와 서버
RESUME QMGR			X	X
RVERIFY SECURITY		X	X	X
SET ARCHIVE	X	X	X	X
SET LOG	X	X	X	X
SET SYSTEM	X	X	X	X
START CHANNEL			X	X
START CHINIT		X	X	X
START CMDSERV	X	X	X	
START LISTENER			X	X
START QMGR			X	
START TRACE	X	X	X	X
STOP CHANNEL			X	X
STOP CHINIT			X	X
STOP CMDSERV	X	X	X	
STOP LISTENER			X	X
STOP QMGR			X	X
STOP TRACE	X	X	X	X
SUSPEND QMGR			X	X

MQSC 명령의 경우, 각 명령 설명은 해당 명령이 실행될 수 있는 소스를 식별합니다.

초기화 명령

초기화 명령은 큐 관리자 시작을 제어하는 데 사용할 수 있습니다.

초기화 입력 데이터 세트의 명령은 큐 관리자 시작 시 IBM MQ가 초기화될 때 처리됩니다. 초기화 입력 데이터 세트에서는 다음 세 가지 유형의 명령이 실행될 수 있습니다.

- 다른 곳에서 정의할 수 없는 IBM MQ 엔티티를 정의하기 위한 명령(예: DEFINE BUFFPOOL).

  이러한 명령은 DD 이름 CSQINP1로 식별되는 데이터 세트에 상주해야 합니다. 이러한 명령은 초기화의 재시작 단계 전에 처리됩니다. 콘솔, 조작 및 제어판 또는 애플리케이션 프로그램을 통해 실행할 수 없습니다. 이러한 명령에 대한 응답은 시작된 태스크 프로시저의 CSQOUT1 문에서 참조하는 순차 데이터 세트에 기록됩니다.
- 재시작 후 복구 가능한 IBM MQ 오브젝트를 정의하기 위한 명령. 이러한 정의는 DD 이름 CSQINP2로 식별되는 데이터 세트에 지정되어야 합니다. 이러한 정의는 페이지 세트 0에 저장됩니다. CSQINP2는 초기화의 재시작 단계 이후에 처리됩니다. 이러한 명령에 대한 응답은 시작된 태스크 프로시저의 CSQOUT2 문에서 참조하는 순차 데이터 세트에 기록됩니다.
- IBM MQ 오브젝트를 조작하기 위한 명령. 이러한 명령은 또한 DD 이름 CSQINP2로 식별되는 데이터 세트에 지정되어야 합니다. 예를 들어, IBM MQ 제공 샘플에는 서브시스템의 데드-레터 큐를 지정하기 위한 ALTER QMGR 명령이 포함됩니다. 이러한 명령에 대한 응답은 CSQOUT2 출력 데이터 세트에 기록됩니다.

참고: IBM MQ 오브젝트가 CSQINP2에 정의되는 경우, IBM MQ는 큐 관리자가 시작될 때마다 오브젝트를 재정의하려고 시도합니다. 오브젝트가 이미 있는 경우에는 재정의 시도가 실패합니다. CSQINP2에서 오브젝트를 정

의해야 하는 경우 DEFINE 명령의 REPLACE 매개변수를 사용하여 이 문제점을 방지할 수 있습니다. 그러나 이 경우 큐 관리자의 이전 실행 중에 수행된 변경을 대체합니다.

IBM MQ for z/OS에는 샘플 초기화 데이터 세트 멤버가 함께 제공됩니다. IBM MQ와 함께 제공되는 샘플 정의의 설명을 참조하십시오.

분산 큐잉에 대한 초기화 명령

CSQINP2 초기화 데이터 세트는 START CHINIT 명령에도 사용할 수 있습니다. 분산 큐잉 환경을 정의하기 위해 (예를 들어, 리스너 시작) 일련의 다른 명령이 필요한 경우 IBM MQ가 채널 시작기 시작된 태스크 프로시저의 일부로 처리되는 세 번째 시작 입력 데이터 세트, CSQINPX를 제공합니다.

데이터 세트에 포함된 MQSC 명령은 채널 시작기 초기화의 끝에 실행되며 CSQOUTX DD 문에서 지정된 데이터 세트에 출력이 작성됩니다. 예를 들어 CSQINPX 초기화 데이터 세트를 사용하여 리스너를 시작할 수 있습니다.

IBM MQ for z/OS에는 샘플 채널 시작기 초기화 데이터 세트 멤버가 함께 제공됩니다. IBM MQ와 함께 제공되는 샘플 정의의 설명을 참조하십시오.

발행/구독에 대한 초기화 명령

발행/구독 환경을 정의하기 위해 일련의 명령이 필요한 경우(예를 들어, 구독을 정의하는 경우) IBM MQ가 네 번째 시작 초기화 입력 데이터 세트, CSQINPT를 제공합니다.

데이터 세트에 포함된 MQSC 명령은 발행/구독 초기화의 끝에 실행되며 CSQOUTT DD 문에서 지정된 데이터 세트에 출력이 작성됩니다. 예를 들어 CSQINPT 초기화 데이터 세트를 사용하여 구독을 정의할 수 있습니다.

IBM MQ for z/OS에는 샘플 발행/구독 초기화 데이터 세트 멤버가 함께 제공됩니다. IBM MQ와 함께 제공되는 샘플 정의의 설명을 참조하십시오.

IBM MQ for z/OS 유틸리티

IBM MQ for z/OS는 간편한 시스템 관리를 위해 사용할 수 있는 유틸리티 프로그램 세트를 제공합니다.

IBM MQ for z/OS는 다음과 같은 다양한 관리 태스크를 수행하는 데 도움이 되는 유틸리티 프로그램 세트를 제공합니다.

- 메시지 보안 정책을 관리합니다.
- 백업, 복원, 재구성 태스크를 수행합니다.
- 명령을 실행하고 오브젝트 정의를 처리합니다.
- 데이터 변환 엑시트를 생성합니다.
- 부트스트랩 데이터 세트를 수정합니다.
- 로그에 대한 정보를 나열합니다.
- 로그를 인쇄합니다.
- Db2 테이블과 기타 Db2 유틸리티를 설정합니다.
- 데드-레터 큐의 메시지를 처리합니다.

메시지 보안 정책 유틸리티

메시지 보안 정책 유틸리티(CSQ0UTIL)는 메시지 보안 정책을 관리하기 위한 독립형 유틸리티로 실행됩니다. 자세한 정보는 메시지 보안 정책 유틸리티(CSQ0UTIL)를 참조하십시오.

CSQUTIL 유틸리티

태스크를 백업, 복원, 재구성하는 데 도움을 주기 위해 제공되는 유틸리티 프로그램입니다. 자세한 정보는 CSQUTIL 유틸리티를 참조하십시오.

데이터 변환 엑시트 유틸리티

IBM MQ for z/OS 데이터 변환 엑시트 유틸리티(**CSQUCVX**)는 데이터 변환 엑시트 루틴을 작성하기 위해 독립형 유틸리티로 실행됩니다.

로그 인벤토리 변경 유틸리티

IBM MQ for z/OS 로그 인벤토리 변경 유틸리티 프로그램(**CSQJU003**)은 부트스트랩 데이터 세트(BSDS) 변경을 위한 독립형 유틸리티로 실행됩니다. 이 유틸리티를 사용하여 다음 기능을 수행할 수 있습니다.

- 활성 또는 아카이브 로그 데이터 세트를 추가하거나 삭제합니다.
- 아카이브 로그의 비밀번호를 제공합니다.

로그 맵 인쇄 유틸리티

IBM MQ for z/OS 로그 맵 인쇄 유틸리티 프로그램(**CSQJU004**)은 다음 정보를 나열하기 위한 독립형 유틸리티로 실행됩니다.

- 모든 활성 및 아카이브 로그 데이터 세트의 두 사본에 대한 로그 데이터 세트 이름과 로그 RBA 연관. 이중 로깅이 활성화되지 않으면 데이터 세트 사본이 하나만 있습니다.
- 새 로그 데이터에 사용 가능한 활성 로그 데이터 세트
- 부트스트랩 데이터 세트(BSDS)에 있는 체크포인트 레코드 큐의 컨텐츠
- 아카이브 로그 명령 실행 기록 레코드의 컨텐츠
- 시스템 및 유틸리티 시간소인

로그 인쇄 유틸리티

로그 인쇄 유틸리티 프로그램(**CSQ1LOGP**)은 독립형 유틸리티로 실행됩니다. 유틸리티는 다음을 지정하여 실행할 수 있습니다.

- 부트스트랩 데이터 세트(BSDS)
- 활성 로그(BSDS 없음)
- 아카이브 로그(BSDS 없음)

큐 공유 그룹 유틸리티

큐 공유 그룹 유틸리티 프로그램(**CSQ5PQSG**)은 Db2 테이블을 설정하고 큐 공유 그룹에 필요한 다른 Db2 태스크를 수행하기 위한 독립형 유틸리티로 실행됩니다.

활성 로그 사전 형식화 유틸리티

활성 로그 사전 형식화 유틸리티(**CSQJUFMT**)는 큐 관리자가 사용하기 전에 활성 로그 데이터 세트를 형식화합니다. 유틸리티가 활성 로그 데이터 세트를 사전에 형식화한 경우, 큐 관리자가 활성 로그를 통해 처음으로 전달할 때 로그 쓰기 성능이 향상됩니다.

데드-레터 큐 핸들러 유틸리티

데드-레터 큐 핸들러 유틸리티 프로그램(**CSQUDLQH**)은 독립형 유틸리티로 실행됩니다. 데드-레터 큐에 있는 메시지를 확인하고 유틸리티에 제공하는 규칙 세트에 따라 처리합니다.

CSQUTIL 유틸리티

CSQUTIL 유틸리티 프로그램은 백업, 복원, 재구성 태스크를 수행하는 데 도움을 주고 명령을 실행하며 오브젝트 정의를 처리하기 위해 IBM MQ for z/OS에 함께 제공됩니다.

CSQUTIL 유틸리티 프로그램에 대한 자세한 정보는 IBM MQ 유틸리티 프로그램(CSQUTIL)을 참조하십시오. 이 유틸리티 프로그램을 사용하여 다음 함수를 호출할 수 있습니다.

COMMAND
MQSC 명령을 실행하고 오브젝트 정의를 기록하며 클라이언트 채널 정의 파일을 작성합니다.

COPY
이름 지정된 IBM MQ for z/OS 메시지 큐의 컨텐츠 또는 이름 지정된 페이지 세트의 모든 큐의 컨텐츠를 읽고 순차 파일에 넣으며 원래 큐를 유지합니다.

COPYPAGE
전체 페이지 세트를 더 큰 페이지 세트에 복사합니다.

EMPTY
이름 지정된 IBM MQ for z/OS 메시지 큐 또는 이름 지정된 페이지 세트의 모든 큐의 컨텐츠를 삭제하고 큐의 정의는 유지합니다.

FORMAT
IBM MQ for z/OS 페이지 세트를 형식화합니다.

LOAD
이름 지정된 IBM MQ for z/OS 메시지 큐의 컨텐츠 또는 이름 지정된 페이지 세트의 모든 큐의 컨텐츠를 COPY 함수로 작성된 순차 파일에서 복원합니다.

PAGEINFO
하나 이상의 페이지 세트에서 페이지 세트 정보를 추출합니다.

RESETPAGE
전체 페이지 세트를 다른 페이지 세트 데이터 세트에 복사하고 사본에서 로그 정보를 재설정합니다.

SCOPY
큐 관리자가 오프라인 상태일 때 큐의 컨텐츠를 데이터 세트에 복사합니다.

SDEFS
큐 관리자가 오프라인 상태일 때 오브젝트의 정의 명령 세트를 생성합니다.

SLOAD
이전 COPY 또는 SCOPY 조작의 목적지 데이터 세트에서 메시지를 복원합니다. SLOAD는 단일 큐를 처리합니다.

SWITCH
클러스터-송신자 채널과 연관된 전송 큐를 전환 또는 조회합니다.

XPARM
채널 시작기 매개변수 로드 모듈을 큐 관리자 속성으로 변환합니다(마이그레이션용).

IBM MQ for z/OS 운영

이 기본 프로시저를 사용하여 IBM MQ for z/OS를 조작할 수 있습니다.

You can also perform the operations described in this section using the IBM MQ Explorer, which is distributed with IBM MQ for Windows, IBM MQ for Linux (x86 and x86-64 platforms) and SupportPac MS0T. 자세한 정보는 56 페이지의 『MQ Explorer를 사용하여 관리』 및 IBM 지원 및 다운로드를 참조하십시오.

이 절에는 다음 주제에 대한 정보가 포함되어 있습니다.

큐 관리자 명령 실행

z/OS 콘솔에서 또는 유틸리티 프로그램 CSQUTIL을 사용하여 IBM MQ 제어 명령을 실행할 수 있습니다. 명령은 명령 접두부 문자열(CPF)을 사용하여 명령을 처리하는 IBM MQ 서브시스템을 표시할 수 있습니다.

IBM MQ 명령을 사용하여 IBM MQ의 운영 환경을 대부분 제어할 수 있습니다. IBM MQ for z/OS는 이러한 명령의 MQSC 및 PCF 유형을 모두 지원합니다. 이 주제는 MQSC 명령을 사용하여 속성을 지정하는 방법을 설명하므로 RCF 이름이 아닌 MQSC 명령 이름을 사용하여 해당 명령과 속성을 나타냅니다. MQSC 명령 구문에 대한 자세한 내용은 MQSC 명령을 참조하십시오. PCF 명령 구문에 대한 자세한 내용은 10 페이지의

『PCF(Programmable Command Format) 사용』의 내용을 참조하십시오. 적정 권한이 부여된 사용자의 경우 다음을 사용하여 IBM MQ 명령을 실행할 수 있습니다.

- 초기화 입력 데이터 세트(239 페이지의 『초기화 명령』의 설명 참조)
- z/OS 콘솔 또는 다른 해당 콘솔(예: SDSF)
- z/OS 마스터 가져오기 명령 루틴, MGCRE (SVC 34)
- IBM MQ 유틸리티, CSQUTIL(IBM MQ 유틸리티 프로그램의 설명 참조)
- 다음과 같은 사용자 애플리케이션:
  - CICS 프로그램
  - TSO 프로그램
  - z/OS 배치 프로그램
  - IMS 프로그램

  해당 정보는 260 페이지의 『IBM MQ 관리 프로그램 작성』의 내용을 참조하십시오.

이러한 명령의 기능은 상당수 TSO 및 ISPF에서 액세스 가능한 조작 및 제어판(248 페이지의 『조작 및 제어판 소개』의 설명 참조)에서 편리하게 제공됩니다.

추가적인 정보는 다음을 참조하십시오.

- 243 페이지의 『z/OS 콘솔 또는 다른 해당 콘솔에서 명령 실행』
  - 명령 접두부 문자열
  - z/OS 콘솔을 사용하여 명령 실행
  - 명령 응답
- 유틸리티 프로그램 CSQUTIL에서 명령 실행

z/OS 콘솔 또는 다른 해당 콘솔에서 명령 실행

모든 IBM MQ 명령은 z/OS 콘솔 또는 다른 해당 콘솔에서 실행할 수 있습니다. z/OS 명령을 실행할 수 있는 다른 위치(예: SDSF)에서 또는 MGCRE 매크로를 사용하는 프로그램으로 IBM MQ 명령을 실행할 수도 있습니다.

콘솔에서 입력된 명령의 결과로 표시될 수 있는 최대 데이터량은 32KB입니다.

참고:

1. IMS 터미널에서 IMS/SSR 명령을 사용하여 IBM MQ 명령을 실행할 수 없습니다. 이 기능은 IMS 어댑터가 지원하지 않습니다.
2. 일부 명령, 특히 채널에 대한 명령의 경우 SDSF가 제공하는 입력 필드의 길이가 충분하지 않을 수 있습니다.

명령 접두부 문자열

각 IBM MQ 명령에는 명령 접두부 문자열(CPF)이 앞에 와야 합니다(244 페이지의 그림 40 참조).

z/OS에서 여러 IBM MQ 서브시스템이 실행될 수 있으므로 CPF를 사용하여 명령을 처리하는 IBM MQ 서브시스템을 표시합니다. 예를 들어, 서브시스템 CSQ1의 큐 관리자를 시작하려면(CPF가 ' +CSQ1 '인 경우) 운영자 콘솔에서 +CSQ1 START QMGR 명령을 실행합니다. 이 CPF는 서브시스템 CSQ1의 서브시스템 이름 테이블에서 정의되어야 합니다. 이 내용은 명령 접두부 문자열(CPF) 정의에 설명되어 있습니다. 예에서 문자열 ' +CSQ1 '은 명령 접두부로 사용됩니다.

z/OS 콘솔을 사용하여 명령 실행

z/OS 콘솔에서 단순 명령을 입력할 수 있습니다(예를 들어, 244 페이지의 그림 40의 DISPLAY 명령). 그러나 복잡한 명령이나 자주 실행하는 명령 세트의 경우에는 다름 명령 실행 방법이 더 효과적입니다.

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
```

그림 *40. z/OS* 콘솔에서 *DISPLAY* 명령 실행

명령 응답

명령에 대한 직접 응답은 명령을 실행한 콘솔로 전송됩니다. IBM MQ는 z/OS에서 사용 가능한 확장 콘솔 지원(EMCS) 기능을 지원하므로 4바이트 ID를 사용하는 콘솔을 사용할 수 있습니다. 또한 MGCRE 매크로를 사용하는 프로그램이 명령을 실행하는 경우 START QMGR 및 STOP QMGR을 제외한 모든 명령이 명령 및 응답 토큰(CART) 사용을 지원합니다.

유틸리티 프로그램 CSQUTIL에서 명령 실행

유틸리티 프로그램 CSQUTIL의 COMMAND 기능을 사용하여 순차 데이터 세트에서 명령을 실행할 수 있습니다. 이 유틸리티는 명령을 메시지로서 시스템 명령 입력 큐에 전송하며 응답을 대기합니다. 응답은 원래 명령과 함께 SYSPRINT에 인쇄됩니다. 자세한 내용은 IBM MQ 유틸리티 프로그램을 참조하십시오.

큐 관리자 시작 및 중지

이 주제는 큐 관리자 중지 및 시작에 대한 소개로 사용합니다.

이 절에서는 큐 관리자를 시작 및 중지하는 방법을 설명합니다. 다음 주제에 대한 정보가 포함되어 있습니다.

- 244 페이지의 『IBM MQ를 시작하기 전에』
- 244 페이지의 『큐 관리자 시작』
- 246 페이지의 『큐 관리자 중지』

큐 관리자 시작 및 중지는 상대적으로 간단합니다. 큐 관리자가 정상 조건에서 중지되면 마지막 조치는 종료 체크포인트를 사용하는 것입니다. 이 체크포인트 및 로그는 큐 관리자가 재시작하는 데 필요한 정보를 제공합니다.

이 절에는 START 및 STOP 명령에 대한 정보가 들어 있고 비정상 종료가 발생한 후 시작에 대한 간략한 개요가 들어 있습니다.

IBM MQ를 시작하기 전에

IBM MQ를 설치하면 정식 z/OS 서브시스템으로 정의됩니다. 이 메시지는 z/OS:의 초기 프로그램 로드(IPL) 중에 나타납니다.

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

여기서 *ssnm*은 IBM MQ 서브시스템 이름입니다.

이제부터 해당 서브시스템 시스템 제어 명령을 실행할 수 있는 권한이 부여된 모든 *z/OS* 콘솔에서 에 대한 큐 관리자를 시작할 수 있습니다. 즉, z/OS SYS 명령 그룹입니다. START 명령은 권한이 부여된 콘솔에서 실행해야 합니다. JES 또는 TSO를 통해 실행할 수 없습니다.

큐 공유 그룹을 사용하는 경우, 큐 관리자를 시작하기 전에 먼저 RRS를 시작한 다음 Db2를 시작해야 합니다.

큐 관리자 시작

START QMGR 명령을 실행하여 큐 관리자를 시작합니다. 그러나 적절한 권한이 없으면 START 명령을 사용할 수 없습니다. IBM MQ 보안에 대한 정보는 z/OS의 보안 설정을 참조하십시오. 245 페이지의 그림 41에서는 START 명령 예를 보여줍니다. IBM MQ 명령 앞에는 명령 접두부 문자열(CPF)이 와야 합니다.

```
+CSQ1  START QMGR
+CSQ1  START QMGR PARM(NEWLOG)
```

그림 *41. z/OS* 콘솔에서 큐 관리자 시작

START QMGR 명령 구문에 대한 정보는 START QMGR을 참조하십시오.

큐 관리자는 배치 작업으로 실행하거나 z/OS 명령 START를 사용하여 시작할 수 없습니다. 이 방법은 비정상적으로 종료되는 IBM MQ의 주소 공간을 시작할 수 있습니다. CSQUTIL 유틸리티 프로그램 또는 유사한 사용자 애플리케이션에서 큐 관리자를 시작할 수도 없습니다.

그러나 z/OS MGCRE (SVC 34) 서비스로 START QMGR 명령을 전달하여 APF 권한 프로그램에서 큐 관리자를 시작할 수 있습니다.

큐 공유 그룹을 사용하는 경우에는 큐 관리자를 시작할 때, 연관된 Db2 시스템과 RRS가 활성 상태여야 합니다.

시작 옵션

큐 관리자를 시작하면 시스템 매개변수 모듈이 로드됩니다. 다음 두 가지 방법 중 하나를 사용하여 시스템 매개변수 모듈의 이름을 지정할 수 있습니다.

- /cpf START QMGR 명령의 PARM 매개변수를 사용합니다. 예를 들어 다음과 같습니다.

  ```
  /cpf START QMGR PARM(CSQ1ZPRM)
  ```

- 시작 프로시저에서 매개변수를 사용합니다. 예를 들어, JCL EXEC 문을 다음과 같이 코딩합니다.

  ```
  //MQM  EXEC PGM=CSQYASCP,PARM='ZPARM(CSQ1ZPRM)'
  ```

시스템 매개변수 모듈은 큐 관리자가 사용자 정의될 때 지정한 정보를 제공합니다.

ENVPARM 옵션을 사용하여 큐 관리자의 JCL 프로시저에서 하나 이상의 매개변수를 대체할 수도 있습니다.

예를 들어, DDname CSQINP2가 변수가 되도록 큐 관리자 시작 프로시저를 업데이트할 수 있습니다. 이는 시작 프로시저를 변경하지 않고 CSQINP2 DDname을 변경할 수 있음을 의미합니다. 이는 변경사항을 구현하고 운영자 및 큐 관리자 조작에 백아웃을 제공할 때 유용합니다.

큐 관리자 CSQ1의 시작 프로시저가 245 페이지의 그림 42와 유사하다고 가정하십시오.

```
//CSQ1MSTR PROC INP2=NORM
//MQMESA  EXEC PGM=CSQYASCP
//STEPLIB DD  DISP=SHR,DSN=thlqual.SCSQANLE
//     DD  DISP=SHR,DSN=thlqual.SCSQAUTH
//     DD  DISP=SHR,DSN=db2qual.SDSNLOAD
//BSDS1  DD  DISP=SHR,DSN=myqual.BSDS01
//BSDS2  DD  DISP=SHR,DSN=myqual.BSDS02
//CSQP0000 DD  DISP=SHR,DSN=myqual.PSID00
//CSQP0001 DD  DISP=SHR,DSN=myqual.PSID01
//CSQP0002 DD  DISP=SHR,DSN=myqual.PSID02
//CSQP0003 DD  DISP=SHR,DSN=myqual.PSID03
//CSQINP1 DD  DISP=SHR,DSN=myqual.CSQINP(CSQ1INP1)
//CSQINP2 DD  DISP=SHR,DSN=myqual.CSQINP(CSQ1&INP2.)
//CSQOUT1 DD  SYSOUT=*
//CSQOUT2 DD  SYSOUT=*
```

그림 *42.* 샘플 시작 프로시저

그런 후 다음 명령으로 큐 관리자를 시작하는 경우:

```
+CSQ1  START QMGR
```

사용된 CSQINP2는 CSQ1NORM이라는 멤버입니다.

그러나 다음에 큐 관리자 CSQ1을 시작할 때 CSQINP2 정의를 CSQ1NEW 구성원에서 가져올 수 있도록 새 프로그램 제품군을 프로덕션에 배치하는 중이라고 가정하십시오. 이를 수행하려면 다음 명령으로 큐 관리자를 시작해야 합니다.

```
+CSQ1  START QMGR ENVPARM('INP2=NEW')
```

CSQ1NEW가 CSQ1NORM 대신에 사용됩니다. 참고: z/OS는 기호 매개변수에 대한 KEYWORD=value 스펙을 255자로 제한합니다(INP2=NEW에서 처럼).

비정상 종료 후 시작

IBM MQ는 정상 종료 또는 비정상 종료 후 재시작되는지 여부를 자동으로 감지합니다.

큐 관리자가 비정상 종료된 후에 시작하는 것은 STOP QMGR 명령이 실행된 후에 시작하는 것과 다릅니다. STOP QMGR 후에 시스템은 작업을 순서대로 완료하고 중지하기 전에 종료 체크포인트를 수행합니다. 큐 관리자를 재시작할 때 이는 시스템 체크포인트 및 복구 로그의 정보를 사용하여 시스템 종료 시 시스템 상태를 판별합니다.

그러나 큐 관리자가 비정상 종료된 경우에는 작업을 완료하거나 종료 체크포인트를 수행하지 않고 종료합니다. 이상종료 후에 큐 관리자를 재시작하는 경우에는 로그에 있는 정보를 사용하여 종료 시에 상태에 대한 지식을 새로 고치고 다양한 태스크의 상태를 사용자에게 알려줍니다. 일반적으로 재시작 프로세스는 모든 불일치 상태를 해결합니다. 그러나 경우에 따라 불일치를 해결하기 위해 특정 단계를 수행해야 합니다.

시작 시 사용자 메시지

큐 관리자가 시작되면 큐 관리자가 일련의 시작 메시지를 생성합니다.

큐 관리자 중지

큐 관리자를 중지하기 전에, 모든 IBM MQ 관련 응답과 함께 운영자에게 쓰기(WTOR) 메시지가 응답을 수신해야 합니다(예를 들어, 로그 요청 가져오기). 246 페이지의 그림 43의 각 명령은 실행 큐 관리자를 종료합니다.

```
+CSQ1  STOP QMGR
+CSQ1  STOP QMGR MODE(QUIESCE)
+CSQ1  STOP QMGR MODE(FORCE)

+CSQ1  STOP QMGR MODE(RESTART)
```

그림 *43.* 큐 관리자 중지

STOP QMGR 명령은 기본적으로 STOP QMGR MODE(QUIESCE)입니다.

QUIESCE 모드에서는 IBM MQ가 새 연결 스레드 작성을 허용하지 않으며 기존 스레드가 계속되는 것은 허용합니다. 모든 스레드가 종료된 경우에만 종료됩니다. 애플리케이션은 큐 관리자가 일시정지되는 경우 알림을 받도록 요청할 수 있습니다. 따라서 알림을 요청한 애플리케이션이 연결을 끊을 기회를 가질 수 있도록 가능하면 QUIESCE 모드를 사용하십시오. 세부사항은 종료 중 발생하는 사항을 참조하십시오.

큐 관리자가 STOP QMGR MODE(QUIESCE) 명령에 대한 응답으로 합당한 시간 내에 종료하지 않으면 DISPLAY CONN 명령을 사용하여 연결 스레드가 존재하는지 여부를 판별하고 연관된 애플리케이션을 종료하기 위해 필요한 단계를 취하십시오. 스레드가 없으면 STOP QMGR MODE(FORCE) 명령을 실행하십시오.

STOP QMGR MODE(QUIESCE) 및 STOP QMGR MODE(FORCE) 명령은 MVS 자동 재시작 관리자(ARM)에서 IBM MQ 등록을 취소하여 ARM이 큐 관리자를 자동으로 재시작하는 것을 방지합니다. STOP QMGR MODE(RESTART) 명령은 ARM에서 IBM MQ 등록을 취소하지 않는다는 것을 제외하고는 STOP QMGR MODE(FORCE) 명령과 동일하게 작동합니다. 이는 큐 관리자가 즉각적인 자동 재시작에 적합함을 의미합니다.

IBM MQ 서브시스템이 ARM에 등록되지 않은 경우에는 STOP QMGR MODE(RESTART) 명령이 거부되며 z/OS 콘솔로 다음 메시지를 보냅니다.

```
CSQY205I ARM element arm-element is not registered
```

이 메시지가 실행되지 않으면 큐 관리자가 자동으로 재시작됩니다. ARM에 대한 자세한 정보는 313 페이지의 『z/OS 자동 재시작 관리자(ARM) 사용』의 내용을 참조하십시오.

STOP QMGR MODE(FORCE)가 큐 관리자를 종료하지 않는 경우에만 큐 관리자 주소 공간을 취소하십시오.

주소 공간을 취소하거나 STOP QMGR MODE(FORCE) 명령을 사용하여 큐 관리자가 중지되는 경우에는 연결된 CICS 또는 IMS 시스템과의 일관성이 유지됩니다. 자원 재동기화는 큐 관리자가 재시작될 때 시작되고 CICS 또는 IMS 시스템에 대한 연결이 설정되면 완료됩니다.

참고: 큐 관리자를 중지할 때 IEF352I 메시지가 실행되었음을 확인할 수도 있습니다. z/OS는 주소 공간을 사용 불가능으로 표시하지 못해 무결성이 노출된 것으로 감지하는 경우 이 메시지를 실행합니다. 이 메시지는 무시해도 됩니다.

중지 메시지

STOP QMGR 명령을 실행한 후 다음과 같은 CSQY009I 및 CSQY002I 메시지가 나타납니다.

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

여기서 userid는 STOP QMGR 명령을 실행한 사용자 ID이고 MODE 매개변수는 명령에 지정된 값에 따라 다릅니다.

STOP 명령이 완료되면 z/OS 콘솔에 다음 메시지가 표시됩니다.

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

ARM을 사용 중이고 MODE(RESTART)를 지정하지 않은 경우 다음 메시지 또한 표시됩니다.

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type
arm-element-type successful
```

다음 메시지가 표시될 때까지는 큐 관리자를 재시작할 수 없습니다.

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

조작 및 제어판 소개

IBM MQ 운영 및 제어판을 사용하여 IBM MQ 오브젝트에 대한 관리 태스크를 수행할 수 있습니다. 이 주제는 명령과 제어판에 대한 소개로 사용합니다.

이 패널은 IBM MQ 오브젝트를 정의, 표시, 대체 또는 삭제하기 위해 사용할 수 있습니다. 패널은 일상적인 관리 목적으로 또한 오브젝트를 약간 변경하는 경우 사용합니다. 많은 오브젝트를 설정 또는 변경하는 경우에는 CSQUTIL 유틸리티 프로그램의 COMMAND 함수를 사용하십시오.

조작 및 제어판은 채널 이니시에이터(예를 들어, 채널 또는 TCP/IP 리스너를 시작하기 위해), 클러스터링, 보안 목적의 제어를 지원합니다. 스레드와 페이지 세트 사용에 대한 정보를 표시할 수도 있습니다.

패널은 시스템 명령 입력 큐를 통해 MQSC 유형 IBM MQ 명령을 큐 관리자로 보냄으로써 작동합니다.

참고:

1. z/OS IBM MQ 운영 및 제어판(CSQOREXX)은 버전 7 이상에서 추가된 새 함수와 매개변수를 일부 지원하지 않을 수 있습니다. 예를 들어, 토픽 오브젝트 또는 구독을 직접 조작할 수 있는 패널은 없습니다.

   다음 지원 메커니즘 중 하나를 사용함으로써 발행/구독 정의와 다른 패널에서 직접 사용할 수 없는 다른 시스템 제어를 관리할 수 있습니다.

   a. IBM MQ 탐색기

   b. z/OS 콘솔

   c. 프로그래밍 가능 명령 형식(PCF) 메시지

   d. CSQUTIL의 COMMAND 함수

   CSQOREXX 패널의 일반 **Command** 조치를 사용하면 SMDS 관련 명령을 포함하는 유효한 MQSC 명령을 실행할 수 있습니다. CSQUTIL의 COMMAND 함수가 발행하는 모든 명령을 사용할 수 있습니다.
2. 패널에서는 명령행에서 직접 IBM MQ 명령을 실행할 수 없습니다.
3. 조작 및 제어판을 사용하려면 올바른 보안 권한이 있어야 합니다(명령 보안 및 명령 자원 보안을 위한 사용자 ID의 설명 참조).
4. CSQUTIL 또는 CSQOREXX 패널을 사용하여 사용자 ID와 비밀번호를 제공할 수 없습니다. 대신 사용자 ID에 MQCONN의 BATCH 프로파일에 대한 UPDATE 권한이 있는 경우 **CHCKLOCL**(*REQUIRED* 설정을 무시할 수 있습니다. 자세한 정보는 로컬 바운드 애플리케이션에서 **CHCKLOCL** 사용 을 참조하십시오.

조작 및 제어판에 대한 호출과 규칙

ISPF 패널에서 IBM MQ를 제어하고 제어 명령을 실행할 수 있습니다.

IBM MQ 조작 및 제어판에 액세스하는 방법

IBM MQ에 대한 ISPF/PDF 기본 옵션 메뉴가 업데이트된 경우 해당 메뉴에서 IBM MQ 운영 및 제어판에 액세스할 수 있습니다. 메뉴 업데이트에 대한 자세한 내용은 태스크 20: 조작 및 제어판 설정을 참조하십시오.

IBM MQ 운영 및 제어판은 TSO 명령 프로세서 패널에서 액세스할 수 있습니다(일반적으로 ISPF/PDF 기본 옵션 메뉴의 옵션 6). 이를 수행하기 위해 실행하는 exec의 이름은 CSQOREXX입니다. 두 개의 매개변수가 있습니다. `thlqual`은 사용될 IBM MQ 라이브러리의 상위 레벨 규정자이고 `langletter`는 사용될 자국어 라이브러리를 식별하는 문자입니다(예: 미국 영어의 경우 E). 영어). IBM MQ 라이브러리가 ISPF 설정에 영구적으로 설치되는 경우 이 매개변수를 생략할 수 있습니다. 또는 TSO 명령행에서 CSQOREXX를 발행할 수 있습니다.

이러한 패널은 운영자와 관리자가 최소의 정규 교육으로 사용할 수 있도록 설계되었습니다. 실행 패널과 함께 이 지시사항을 읽고 다른 제안 태스크를 시도하십시오.

참고: 패널을 사용하는 동안 SYSTEM.CSQOREXX.* 작성되었다.

조작 및 제어판에 대한 규칙

IBM MQ 문자열 및 이름에 대한 일반 규칙은 IBM MQ 오브젝트 이름 지정 규칙을 참조하십시오. 그러나 다음과 같은 일부 규칙은 조작 및 제어판에만 적용됩니다.

- 문자열(예를 들어, 설명)을 작은따옴표 또는 큰따옴표로 묶지 마십시오.

- 텍스트 필드에 아포스트로피 또는 인용 부호를 포함하는 경우에는 반복하거나 이스케이프 문자를 추가하지 않아도 됩니다. 문자는 입력한 대로 정확하게 저장됩니다. 예를 들어, 다음과 같습니다.

```
This is Maria's queue
```

  패널 프로세서는 문자를 복사하여 IBM MQ로 전달합니다. 그러나 이 작업을 수행하기 위해 데이터를 잘라야 하는 경우 데이터를 자릅니다.
- 대문자 또는 소문자는 대부분의 필드에서 사용할 수 있지만 Enter를 누르면 대문자로 변환됩니다. 예외사항은 다음과 같습니다.
  - 스토리지 클래스 이름과 커플링 기능 구조 이름의 경우에는 A에서 Z까지의 대문자로 시작해야 하며 그 뒤에 A에서 Z까지의 대문자 또는 숫자 문자가 와야 합니다.
  - 변환되지 않는 특정 필드. 다음이 포함됩니다.
    - 애플리케이션 ID
    - 설명
    - 환경 데이터.
    - 오브젝트 이름(그러나 소문자 오브젝트 이름을 사용하는 경우에는 z/OS 콘솔에서 입력하지 못할 수 있습니다.)
    - 원격 시스템 이름
    - 트리거 데이터
    - 사용자 데이터
- 이름에서는 선행 공백과 선행 밑줄을 무시합니다. 따라서 오브젝트 이름은 공백 또는 밑줄로 시작할 수 없습니다.
- 밑줄은 공백 필드의 범위를 표시하는 데 사용됩니다. Enter를 누르면 후행 밑줄이 공백으로 바뀝니다.
- 많은 설명과 텍스트 필드가 복수 파트로 표시되며 IBM MQ는 각 파트를 독립적으로 처리합니다. 이는 후행 공백이 유지되며 텍스트가 인접하지 않음을 의미합니다.

공백 필드

IBM MQ 오브젝트에 대한 **정의** 조치를 지정하면 정의 패널의 각 필드에 값이 포함됩니다. IBM MQ가 값을 가져오는 위치에 대한 정보는 표시 패널에 대한 일반 도움말(확장 도움말)을 참조하십시오. 필드에 공백을 입력하고 공백이 허용되지 않으면 IBM MQ가 필드에 설치 기본값을 지정하거나 필수값을 입력하라는 프롬프트를 표시합니다.

IBM MQ 오브젝트에 대한 **대체** 조치를 지정하면 대체 패널의 각 필드에 해당 필드의 현재 값이 포함됩니다. 필드에 공백을 입력하고 공백이 허용되지 않으면 해당 필드의 값이 변경되지 않습니다.

오브젝트 및 조치

조작 및 제어판은 다양한 유형의 오브젝트와 오브젝트에 대해 수행할 수 있는 여러 조치를 제공합니다.

초기 패널에는 조치가 나열되며 오브젝트를 조작하고 관련 정보를 표시할 수 있습니다. 이러한 오브젝트에는 모든 IBM MQ 오브젝트가 일부 예비 오브젝트가 함께 포함됩니다. 오브젝트는 다음 범주로 분류됩니다.

- 큐, 프로세스, 인증 정보 오브젝트, 이름 목록, 스토리지 클래스, CF 구조
- 채널
- 클러스터 오브젝트
- 큐 관리자 및 보안
- 연결
- 시스템

IBM MQ 오브젝트와 함께 수행할 수 있는 조치의 상호 참조 테이블은 조치 를 참조하십시오.

큐, 프로세스, 인증 정보 오브젝트, 이름 목록, 스토리지 클래스, CF 구조

기본 IBM MQ 오브젝트입니다. 각 유형별로 많은 오브젝트가 있을 수 있습니다. 나열, 필터로 나열, 정의 및 삭제될 수 있으며 LIST 또는 DISPLAY, LIST with FILTER, DEFINE LIKE, MANAGE 및 ALTER 조치를 사용하여 표시 및 대체될 수 있는 속성을 갖습니다. 오브젝트는 MANAGE 조치를 사용하여 삭제됩니다.

이 범주는 다음 오브젝트로 구성됩니다.

QLOCAL	로컬 큐
QREMOTE	리모트 큐
QALIAS	큐에 대한 간접 참조의 알리어스 큐
QMODEL	동적 큐 정의를 위한 모델 큐
큐	모든 유형의 큐
QSTATUS	로컬 큐 상태
PROCESS	트리거 이벤트가 발생할 때 시작될 애플리케이션에 대한 정보
AUTHINFO	인증 정보: LDAP 서버를 사용한 인증서 폐기 목록(CRL) 검사를 수행하는 데 필요한 정의
이름 목록	큐 또는 클러스터와 같은 이름 목록
STGCLASS	스토리지 클래스
CFSTRUCT	커플링 기능(CF) 구조
CFSTATUS	CF 구조 상태

채널

채널은 분산 큐잉에 사용됩니다. 각 유형별로 다수가 존재할 수 있으며 나열, 필터로 나열, 정의, 삭제, 표시, 대체될 수 있습니다. 또한 START, STOP, PERFORM 조치를 사용하여 사용 가능한 다른 기능을 가질 수 있습니다. PERFORM은 채널 재설정, ping, 해결 기능을 제공합니다.

이 범주는 다음 오브젝트로 구성됩니다.

CHANNEL	모든 유형의 채널
송신자	송신자 채널
SERVER	서버 채널
수신자	수신자 채널
요청자	요청자 채널
CLUSRCVR	클러스터 수신자 채널
CLUSSDR	클러스터-송신자 채널
SVRCONN	서버 연결 채널
CLNTCONN	클라이언트 연결 채널
CHSTATUS	채널 연결 상태

클러스터 오브젝트

클러스터에 속하는 큐와 채널에 대해 클러스터 오브젝트가 자동으로 작성됩니다. 기본 큐 및 채널 정의는 다른 큐 관리자에 있을 수 있습니다. 각 유형별로 다수가 존재할 수 있으며 이름은 중복될 수 있습니다. 나열, 필터로 나열, 표시될 수 있습니다. PERFORM, START, STOP은 LIST 조치를 통해서도 사용할 수 있습니다.

이 범주는 다음 오브젝트로 구성됩니다.

CLUSQ	클러스터 큐, 클러스터에 속하는 큐용으로 작성됨
CLUSCHL	클러스터 채널, 클러스터에 속하는 채널용으로 작성됨

CLUSQMGR 클러스터 큐 관리자, 클러스터 채널과 같지만 큐 관리자 이름으로 식별됨

클러스터 채널과 클러스터 채널 관리자에는 PERFORM, START, STOP 조치가 있지만 ISPLAY 조치를 통해 간접적으로만 가능합니다.

큐 관리자 및 보안

큐 관리자 및 보안 오브젝트는 단일 인스턴스를 갖습니다. 나열될 수 있으며, LIST 또는 DISPLAY 및 ALTER 조치를 사용하여 표시, 대체될 수 있는 속성과 함께 PERFORM 조치를 사용하여 사용 가능한 다른 기능을 갖습니다.

이 범주는 다음 오브젝트로 구성됩니다.

관리자	큐 관리자: PERFORM 조치는 클러스터 일시중단 및 재개 기능을 제공함
보안	보안 기능: PERFORM 조치는 새로 고치기 및 재확인 기능을 제공함

연결

연결은 나열, 필터로 나열, 표시될 수 있습니다.

이 범주는 연결 오브젝트, CONNECT로만 구성됩니다.

시스템

다른 기능의 콜렉션. 이 범주는 다음 오브젝트로 구성됩니다.

SYSTEM	시스템 기능
CONTROL	SYSTEM의 동의어

사용할 수 있는 기능은 다음과 같습니다.

LIST 또는 DISPLAY	큐 공유 그룹, 분산 큐잉, 페이지 세트 또는 데이터 세트 사용 정보를 표시합니다.
PERFORM	클러스터링 새로 고치기 또는 재설정
시작	채널 시작기 또는 리스너 시작
STOP	채널 시작기 또는 리스너 중지

조치

다음 표에는 각 오브젝트 유형에 수행할 수 있는 조치가 표시되어 있습니다.

표 *17. IBM MQ* 오브젝트에 올바른 조작 및 제어판 조치

오브젝트	대체	유사 정의	관리(1)	나열 또는 표시	필터로 나열	수행	시작	중지
AUTHINFO	X	X	X	X	X			
CFSTATUS				X				
CFSTRUCT	X	X	X	X	X			
CHANNEL	X	X	X	X	X	X	X	X
CHSTATUS				X	X			
CLNTCONN	X	X	X	X	X			
CLUSCHL				X	X	X(2)	X(2)	X(2)
CLUSQ				X	X			
CLUSQMGR				X	X	X(2)	X(2)	X(2)

표 17. *IBM MQ* 오브젝트에 올바른 조작 및 제어판 조치 *(계속)*								
오브젝트	**대체**	**유사 정의**	**관리(1)**	**나열 또는 표시**	**필터로 나열**	**수행**	**시작**	**중지**
CLUSRCVR	X	X	X	X	X	X	X	X
CLUSSDR	X	X	X	X	X	X	X	X
CONNECT				X	X			
CONTROL				X		X	X	X
관리자	X			X		X		
이름 목록	X	X	X	X	X			
PROCESS	X	X	X	X	X			
QALIAS	X	X	X	X	X			
QLOCAL	X	X	X	X	X			
QMODEL	X	X	X	X	X			
QREMOTE	X	X	X	X	X			
QSTATUS				X	X			
큐	X	X	X	X	X			
수신자	X	X	X	X	X	X	X	X
요청자	X	X	X	X	X	X	X	X
보안	X			X		X		
송신자	X	X	X	X	X	X	X	X
SERVER	X	X	X	X	X	X	X	X
SVRCONN	X	X	X	X	X		X	X
STGCLASS	X	X	X	X	X			
SYSTEM				X		X	X	X

참고:

1. `Delete` 및 기타 기능을 제공합니다.
2. `List or Display` 조치 사용

오브젝트 속성 지정

작업을 수행해야 하는 오브젝트의 속성 지정을 지정할 수 있습니다. 속성 지정은 오브젝트 **정의**의 보관 위치와 오브젝트 작동 방식을 나타냅니다.

속성 지정은 다음 오브젝트 유형에 대한 작업을 수행하는 경우에만 의미를 갖습니다.

- 큐
- 채널
- 프로세스
- 이름 목록
- 스토리지 클래스
- 인증 정보 오브젝트

다른 오브젝트 유형에 대한 작업을 수행하는 경우에는 속성 지정을 무시합니다.

허용되는 값은 다음과 같습니다.

Q
QMGR. 오브젝트 정의는 큐 관리자의 페이지 세트에 있으며 큐 관리자만 액세스할 수 있습니다.

C
COPY. 오브젝트 정의는 큐 관리자의 페이지 세트에 있으며 큐 관리자만 액세스할 수 있습니다. GROUP 속성 지정이 있는 것으로 정의되는 오브젝트의 로컬 사본입니다.

P
PRIVATE. 오브젝트 정의는 큐 관리자의 페이지 세트에 있으며 큐 관리자만 액세스할 수 있습니다. 이 오브젝트는 QMGR 또는 COPY 속성 지정이 있는 것으로 정의되었습니다.

G
그룹. 오브젝트 정의는 공유 저장소에 있으며 큐 공유 그룹의 모든 큐 관리자가 액세스할 수 있습니다.

S
SHARED. 이 속성 지정은 로컬 큐에만 적용됩니다. 큐 정의는 공유 저장소에 있으며 큐 공유 그룹의 모든 큐 관리자가 액세스할 수 있습니다.

A
모두. 조치 큐 관리자가 대상 큐 관리자 또는 *인 경우에는 **모든** 속성 지정 오브젝트가 포함되고 그렇지 않으면 QMGR 및 COPY 속성 지정 오브젝트만 포함됩니다. 기본값입니다.

ISPF 제어판을 사용하여 큐 관리자, 기본값, 레벨 선택

ISPF에서 CSQOREXX exec를 사용하여 큐 관리자를 제어할 수 있습니다.

초기 패널을 보는 동안에는 큐 관리자에 연결되지 않습니다. 그러나 Enter를 누르면 큐 관리자 또는 **연결 이름** 필드에서 이름이 지정된 큐 공유 그룹의 큐 관리자로 연결됩니다. 이 필드는 공백으로 둘 수 있습니다. 이는 배치 애플리케이션에 기본 큐 관리자를 사용함을 의미합니다. 이는 CSQBDEFV에 정의됩니다. 해당 정보는 태스크 19: 배치, TSO, RRS 어댑터 설정을 참조하십시오.

요청하는 조치가 수행될 큐 관리자를 지정하려면 **대상 큐 관리자** 필드를 사용하십시오. 이 필드를 공백으로 두는 경우 기본값은 **연결 이름** 필드에 지정된 큐 관리자입니다. 연결하지 않은 대상 큐 관리자를 지정할 수 있습니다. 이 경우 일반적으로 큐 관리자 알리어스 정의를 제공하는 리모트 큐 관리자 오브젝트의 이름을 지정합니다(이름은 명령 입력 큐를 열 때 *ObjectQMgrName*으로 사용됨). 이 작업을 수행하려면 리모트 큐 관리자에 액세스하는 데 적당한 큐와 채널이 설정되어 있어야 합니다.

조치 큐 관리자를 사용하여 **대상 큐 관리자** 필드에서 지정된 큐 관리자와 동일한 큐 공유 그룹에 있는 큐 관리자를 사용자가 요청하는 조치가 수행될 큐 관리자로 지정할 수 있습니다. 이 필드에서 *를 지정하면 요청하는 조치가 큐 공유 그룹의 모든 큐 관리자에서 수행됩니다. 이 필드를 공백으로 두는 경우 기본값은 **대상 큐 관리자** 필드에 지정된 값입니다. **조치 큐 관리자** 필드는 MQSC 명령에서 설명하는 CMDSCOPE 명령 수정자를 사용하는 것과 같습니다.

큐 관리자 기본값

큐 관리자 필드를 공백으로 두거나 큐 공유 그룹에 연결하도록 선택하는 경우 Enter를 누르면 보조 창이 열립니다. 이 창에서 사용할 큐 관리자의 이름을 확인합니다. Press Enter to continue. 일부 요청을 작성한 후 초기 패널로 돌아가면 필드가 실제 이름으로 완성되어 있습니다.

큐 관리자 레벨

조작 및 제어판은 z/OS에서 실행되고 명령 레벨이 패널의 명령 레벨(현재 710 또는 800)과 일치하는 큐 관리자에만 정상적으로 작동합니다.

이러한 조건이 충족되지 않으면 조치가 부분적으로 또는 잘못 실행되거나 전혀 실행되지 않을 수 있으며 큐 관리자의 복제본이 인식되지 않을 수 있습니다.

조치 큐 관리자가 명령 레벨 800에 없으면 일부 필드가 표시되지 않아 일부 값을 입력할 수 없습니다. 몇몇 오브젝트와 조치는 허용되지 않습니다. 이러한 경우 보조 창이 열려 계속 진행할지 여부를 확인하는 질문이 표시됩니다.

ISPF 제어판에서 기능 키와 명령행 사용

패널을 사용하려면 기능 키를 사용하거나 ISPF 제어판 명령 영역에 해당 명령을 입력해야 합니다.

- 기능 키
  - 조치 처리
  - 254 페이지의 『IBM MQ 사용자 메시지 표시』
  - 조치 취소
  - 도움말 보기
- 명령행 사용

기능 키

기능 키는 IBM MQ에 대한 특수 설정을 갖습니다. 이는 기능 키에 ISPF 기본값을 사용할 수 없음을 의미합니다. 이전에 KEYLIST OFF ISPF 명령을 다른 곳에서 사용한 경우에는 조작 및 제어판의 명령 영역에 KEYLIST ON을 입력한 다음 Enter를 눌러 IBM MQ 설정을 사용 가능하도록 지정해야 합니다.

이러한 기능 키 설정은 패널에 표시될 수 있습니다(255 페이지의 그림 44 참조). 설정이 표시되지 않으면 조작 및 제어판의 명령 영역에 PFSHOW를 입력한 다음 Enter를 누르십시오. 설정 표시를 제거하려면 FSHOW OFF 명령을 사용하십시오.

조작 및 제어판의 기능 키 설정은 CUA 표준을 준수합니다. 일반 ISPF 프로시저(예: KEYLIST 유틸리티)를 통해 키 설정을 변경할 수 있지만 권장되지는 않습니다.

참고: PFSHOW 및 KEYLIST 명령 사용은 다른 모든 논리적 ISPF 화면에 영향을 주므로 조작 및 제어판을 떠나도 해당 설정은 유지됩니다.

조치 처리
패널에서 요청된 조치를 수행하려면 Enter를 누르십시오. 패널의 정보는 큐 관리자로 보내 처리됩니다.

패널에서 Enter를 누를 때마다 IBM MQ가 하나 이상의 운영자 메시지를 생성합니다. 조작이 성공하면 확인 메시지 CSQ9022I이 나타나고 그렇지 않으면 오류 메시지가 나타납니다.

IBM MQ 사용자 메시지 표시
아무 패널에서나 기능 키 F10을 누르면 IBM MQ 사용자 메시지가 나타납니다.

조치 취소
초기 패널에서 F3 및 F12는 둘 다 조작 및 제어판을 종료하고 ISPF로 돌아갑니다. 큐 관리자에게 정보를 보내지 않습니다.

다른 패널에서 기능 키 F3 또는 F12를 누르면 현재 패널이 종료되고 **마지막으로 Enter를 누른 후 입력한 모든 데이터는 무시됩니다**. 강조하건대, 큐 관리자에게 정보를 보내지 않습니다.

- F3을 누르면 초기 패널로 다시 돌아갑니다.
- F12를 누르면 이전 패널로 돌아갑니다.

도움말 가져오기

각 패널에는 연관된 도움말 패널이 있습니다. 도움말 패널은 ISPF 프로토콜을 사용합니다.

- 아무 패널에서나 기능 키 F1을 누르면 태스크에 대한 일반 도움말(확장 도움말)이 나타납니다.
- 아무 필드에나 커서를 두고 기능 키 F1을 누르면 해당 필드에 대한 특정 도움말이 나타납니다.
- 아무 필드에서나 기능 키 F5를 누르면 일반 도움말이 나타납니다.
- 기능 키 F3을 누르면 기능 키 F1을 눌렀던 기본 패널로 돌아갑니다.
- 아무 도움말 패널에서나 기능 키 F6을 누르면 기능 키에 대한 도움말이 나타납니다.

도움말 정보가 두 번째 또는 후속 페이지까지 이어지면 패널의 오른쪽 상단에 **계속** 표시기가 나타납니다. 다음 기능 키를 사용하면 도움말 페이지를 탐색할 수 있습니다.

- F11: 다음 도움말 페이지로 이동합니다(다음 페이지가 있는 경우).
- F10: 이전 도움말 페이지로 돌아갑니다(이전 페이지가 있는 경우).

명령행 사용

조작 및 제어판이 사용하는 명령은 기능 키에서 사용할 수 있으므로 이 명령을 실행하기 위해 명령행을 사용하지 않아도 됩니다. 일반 ISPF 명령(예: PFSHOW)을 입력할 수 있는 명령행이 제공됩니다.

ISPF 명령 PANELID ON은 현재 CSQOREXX 패널의 이름을 표시합니다.

명령행은 처음에 ISPF 설정에 관계 없이 패널 맨 아래에 있는 기본 위치에 표시됩니다. 조작 및 제어판에서 SETTINGS ISPF 명령을 사용하여 명령행의 위치를 변경할 수 있습니다. 이 설정은 조작 및 제어판과의 후속 세션을 위해 유지됩니다.

조작 및 제어판 사용

CSQOREXX에서 표시되는 초기 제어판을 조사하려면 이 주제를 사용합니다.

255 페이지의 그림 44에서는 패널 세션을 시작할 때 표시되는 패널을 보여줍니다.

```
                  IBM MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . . 1     0. List with filter   4. Manage
                                    1. List or Display    5. Perform
                                    2. Define like        6. Start
                                    3. Alter              7. Stop
                                    8. Command
Object type . . . . . . . . . CHANNEL       +
Name  . . . . . . . . . . . . *
Disposition . . . . . . . . . A  Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . . MQ1C  - local queue manager or group
Target queue manager  . . . MQ1C
          - connected or remote queue manager for command input
Action queue manager  . . . MQ1C  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993, 2023. All rights reserved.

Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

그림 *44. IBM MQ* 운영 및 제어 초기 패널

이 패널에서는 다음 조치를 수행할 수 있습니다.

- 원하는 로컬 큐 관리자와 명령을 해당 큐 관리자, 리모트 큐 관리자 또는 로컬 큐 관리자와 동일한 큐 공유 그룹의 다른 큐 관리자에서 실행하려는지 여부를 선택합니다. 큐 관리자 이름을 변경해야 하는 경우 이름을 입력합니다.
- **조치** 필드에 적절한 번호를 입력하여 실행하려는 조치를 선택하십시오.
- 작업을 수행하려는 오브젝트 유형을 지정하십시오. 오브젝트 유형을 모르는 경우 오브젝트 유형에 대한 관련 도움말을 보려면 기능 키 F1을 누르십시오.
- 작업을 수행하려는 오브젝트 유형의 속성을 지정하십시오.
- 지정된 유형의 오브젝트 목록을 표시하십시오. **이름** 필드에 별표(*)를 입력하고 Enter를 누르면 이미 조치 큐 관리자에 정의된 지정된 유형의 오브젝트 목록이 표시됩니다. 그런 다음 작업을 수행하려는 하나 이상의 오브젝트를 차례로 선택할 수 있습니다. 모든 조치는 목록에서 선택할 수 있습니다.

참고: 오브젝트 목록이 표시되도록 선택한 다음 해당 목록에서 작업을 선택하는 것이 좋습니다. 모든 오브젝트 유형에 허용되는 **표시** 조치를 사용하십시오.

명령 기능 사용

편집기를 사용하여 큐 관리자로 전달될 MQSC 명령을 입력 또는 수정할 수 있습니다.

기본 패널, CSQOPRIA에서 옵션 **8 명령**을 사용하여 명령 기능을 시작하십시오.

CSQUTIL COMMAND 함수에 대한 입력으로 사용되는 순차 파일, *prefix*.CSQUTIL.COMMANDS의 편집 세션이 표시됩니다. IBM MQ에 명령 실행을 참조하십시오.

명령 앞에 명령 접두부 문자열(CPF)이 오지 않아도 됩니다.

연속 문자 **+** 또는 **-**로 현재 행을 종료하여 후속 행에서 MQSC 명령을 계속할 수 있습니다. 또는 행 편집 모드를 사용하여 긴 MQSC 명령 또는 명령 내의 긴 속성 값을 제공하십시오.

행 편집

행 편집을 사용하려면 편집 패널에서 적절한 행으로 커서를 옮기고 **F4**를 사용하여 스크롤 가능 패널에 단일 행을 표시하십시오. 단일 행의 최대 데이터 크기는 32 760바이트입니다.

행 편집 종료 방법:

- **F3 종료**는 행 변경사항을 저장하고 종료합니다.
- **F12 취소**는 행 변경사항을 제거하고 편집 패널로 돌아갑니다.

이 세션에서 수행된 변경사항을 제거하려면 **F12 취소**를 사용하여 파일 컨텐츠를 변경하지 않고 편집 세션을 종료하십시오. 명령은 실행되지 않습니다.

명령 실행

MQSC 명령 입력을 완료하면 **F3 종료**로 편집 세션을 종료하여 파일 컨텐츠를 저장하고 CSQUTIL을 호출하여 큐 관리자로 명령을 전달하십시오. 명령 처리 출력은 *prefix*.CSQUTIL.OUTPUT 파일에 저장됩니다. 이 파일에서 편집 세션이 자동으로 열려 응답을 볼 수 있습니다. 이 세션을 종료하고 기본 메뉴로 돌아가려면 **F3 종료**를 누르십시오.

IBM MQ 오브젝트에 대한 작업

이 문서에서 설명하는 많은 태스크에는 IBM MQ 오브젝트 조작이 포함됩니다. 오브젝트 유형에는 큐 관리자, 큐, 프로세스 정의, 이름 목록, 채널, 클라이언트 연결 채널, 리스너, 서비스, 인증 정보 오브젝트가 있습니다.

- 단순 큐 오브젝트 정의
- 다른 유형의 오브젝트 정의
- 오브젝트 정의에 대한 작업
- 이름 목록에 대한 작업

단순 큐 오브젝트 정의

새 오브젝트를 정의하려면 기존 정의를 기반으로 사용하십시오. 다음 세 가지 방법 중 하나로 수행할 수 있습니다.

- 초기 패널에서 선택한 옵션의 결과로 표시되는 목록의 멤버인 오브젝트를 선택합니다. 그런 다음 선택한 오브젝트 옆에 있는 조치 필드에 조치 유형 2(**유사 정의**)를 입력합니다. 이제 오브젝트는 속성 지정을 제외하고 선택한 오브젝트의 속성을 갖게 됩니다. 그러면 필요에 따라 새 오브젝트에서 속성을 변경할 수 있습니다.
- 초기 패널에서 **유사 정의** 조치 유형을 선택하고 **오브젝트 유형** 필드에 정의하는 오브젝트 유형을 입력하고 **이름** 필드에 특정 기존 오브젝트의 이름을 입력합니다. 새 오브젝트는 속성 지정을 제외하고 **이름** 필드에 이름을 지정한 오브젝트와 동일한 속성을 갖습니다. 그러면 필요에 따라 새 오브젝트 정의에서 속성을 변경할 수 있습니다.
- 오브젝트 유형을 지정하고 **이름** 필드를 공백으로 두어 **유사 정의** 조치 유형을 선택합니다. 그런 다음 새 오브젝트를 정의할 수 있으며 설치에 정의된 기본 속성을 갖습니다. 그러면 필요에 따라 새 오브젝트 정의에서 속성을 변경할 수 있습니다.

참고: 초기 패널에서 정의하는 오브젝트의 이름을 입력하지 마십시오. **정의** 패널에서는 입력할 수 있습니다.

다음 예는 기존 큐를 템플리트로 사용하여 로컬 큐를 정의하는 방법을 보여줍니다.

로컬 큐 정의

조작 및 제어판에서 로컬 큐 오브젝트를 정의하려면 기존 큐 정의를 새 정의의 기반으로 사용하십시오. 여러 패널을 완료해야 합니다. 모든 패널을 완료하고 속성이 올바른 것으로 확인한 후에는 Enter를 눌러 큐 관리자로 정의를 보내십시오. 그러면 실제 큐가 작성됩니다.

초기 패널에서 또는 초기 패널에서 선택한 옵션의 결과로 표시되는 목록의 오브젝트 항목에 대해 **유사 정의** 조치를 사용하십시오.

예를 들어, 초기 패널부터 다음 필드를 완료하십시오.

조치 2(유사 정의)

오브젝트 유형	QLOCAL
이름	QUEUE.YOU.LIKE. 새 큐의 속성을 제공하는 큐의 이름입니다.

Enter를 눌러 **로컬 큐 정의** 패널을 표시하십시오. 큐 이름 필드는 공백이므로 새 큐의 이름을 입력할 수 있습니다. 설명은 이 새 정의의 기반이 되는 큐에 대한 설명입니다. 이 필드에는 새 큐에 대한 사용자 고유의 설명을 입력하십시오.

속성 지정을 제외한 다른 필드의 값은 이 새 큐의 기반이 되는 큐의 값입니다. 이 필드는 필요에 따라 입력할 수 있습니다. 예를 들어, 적합한 권한이 부여된 애플리케이션이 이 큐에 메시지를 넣을 수 있는 경우에는 **넣기 사용** 필드에 Y를 (아직 Y가 없는 경우) 입력하십시오.

커서를 필드로 이동하고 기능 키 F1을 누르면 필드 도움말이 나타납니다. 필드 도움말은 각 속성에 사용할 수 있는 값에 대한 정보를 제공합니다.

첫 번째 패널을 완료하면 기능 키 F8을 눌러 두 번째 패널을 표시하십시오.

힌트:

1. 이 단계에서는 Enter를 누르지 마십시오. Enter를 누르면 나머지 필드를 완료할 수 있기 전에 큐가 작성됩니다. Enter를 먼저 누르더라도 나중에 언제라도 정의를 변경할 수 있으므로 걱정하지 않아도 됩니다.
2. 기능 키 F3 또는 F12는 누르지 마십시오. 이 키를 누르면 입력한 데이터를 잃게 됩니다.

기능 키 F8을 계속 누르면 트리거 정의, 이벤트 제어, 백아웃 보고 패널을 포함하는 나머지 패널이 나타나 완료할 수 있습니다.

로컬 큐 정의가 완료되는 경우

정의가 완료되면 Enter를 눌러 큐 관리자로 처리할 정보를 보내십시오. 큐 관리자는 제공한 정의에 따라 큐를 작성합니다. 큐를 작성하지 않으려면 기능 키 F3을 눌러 종료하고 정의를 취소하십시오.

다른 오브젝트 유형 정의

다른 오브젝트 유형을 정의하려면 기존 정의를 새 정의의 기반으로 사용하십시오(로컬 큐 정의의 설명 참조).

초기 패널에서 또는 초기 패널에서 선택한 옵션의 결과로 표시되는 목록의 오브젝트 항목에 대해 **유사 정의** 조치를 사용하십시오.

예를 들어, 초기 패널부터 다음 필드를 완료하십시오.

조치	2(유사 정의)
오브젝트 유형	QALIAS, NAMELIST, PROCESS, CHANNEL 및 기타 자원 오브젝트
이름	공백으로 남겨두거나 동일한 유형의 기존 오브젝트 이름을 입력합니다.

Enter를 눌러 해당 DEFINE 패널을 표시하십시오. 필요에 따라 필드를 완료한 다음 Enter를 다시 눌러 큐 관리자로 정보를 보내십시오.

로컬 큐 정의와 같이 다른 유형의 오브젝트를 정의하려면 일반적으로 여러 패널을 완료해야 합니다. 이름 목록을 정의하려면 일부 추가 작업이 필요합니다(258 페이지의 『이름 목록에 대한 작업』의 설명 참조).

오브젝트 정의에 대한 작업

오브젝트가 정의되면 **조치** 필드에 오브젝트를 대체, 표시 또는 관리하는 조치를 지정할 수 있습니다.

각각의 경우 다음 중 하나를 수행할 수 있습니다.

- 초기 패널에서 선택한 옵션의 결과로 표시되는 목록에서 작업을 수행하려는 오브젝트를 선택합니다. 예를 들어, **조치** 필드에 1을 입력하여 오브젝트를 표시하고 **오브젝트 유형** 필드에 Queue, **이름** 필드에 *를 입력하면 시스템에 정의된 모든 큐의 목록이 나타납니다. 그런 다음 작업을 수행해야 하는 큐를 이 목록에서 선택할 수 있습니다.
- **오브젝트 유형** 및 **이름** 필드를 완료하여 작업을 수행하는 오브젝트를 지정하는 초기 패널부터 시작합니다.

오브젝트 정의 대체
오브젝트 정의를 대체하려면 조치 3을 지정하고 Enter를 눌러 ALTER 패널을 표시하십시오. 이러한 패널은 DEFINE 패널과 매우 유사합니다. 원하는 값을 대체할 수 있습니다. 변경사항이 완료되면 Enter를 눌러 큐 관리자로 처리할 정보를 보내십시오.

오브젝트 정의 표시
오브젝트 세부사항을 변경하지 않고 표시하려면 조치 1을 지정하고 Enter를 눌러 DISPLAY 패널을 표시하십시오. 앞에서 말한 것처럼 이 패널은 필드를 변경할 수 없다는 것을 제외하고는 DEFINE 패널과 유사합니다. 다른 오브젝트의 세부사항을 표시하려면 오브젝트 이름을 변경하십시오.

오브젝트 삭제
오브젝트를 삭제하려면 조치 4(관리)를 지정하십시오. **삭제** 조치는 결과 메뉴에 나타나는 조치 중 하나입니다. **삭제** 조치를 선택하십시오.

요청을 확인해야 합니다. 기능 키 F3 또는 F12를 누르면 요청이 취소됩니다. Enter를 누르면 요청을 확인하고 큐 관리자로 전달됩니다. 그런 다음 지정한 오브젝트가 삭제됩니다.

참고: 채널 시작기가 시작되지 않으면 대부분 유형의 채널 오브젝트를 삭제할 수 없습니다.

이름 목록에 대한 작업

이름 목록에 대한 작업을 수행하는 경우 다른 오브젝트와 같이 진행하십시오.

DEFINE LIKE 또는 ALTER 조치의 경우, 기능 키 F11을 눌러 목록에 이름을 추가하거나 목록에서 이름을 변경하십시오. 여기에는 ISPF 편집기에 대한 작업이 포함되며 모든 일반 ISPF 편집 명령을 사용할 수 있습니다. 이름 목록의 각 이름을 개별 행에 입력하십시오.

ISPF 편집기를 이러한 방식으로 사용하는 경우 기능 키 설정은 일반 ISPF 설정이며 다른 조작 및 제어판이 사용하는 설정이 **아닙니다**.

목록에서 소문자 이름을 지정해야 하는 경우 편집기 패널 명령행에서 CAPS(OFF)를 지정하십시오. 이 작업을 수행하는 경우, 향후 편집하는 모든 이름 목록은 CAPS(ON)을 지정할 때까지 소문자입니다.

이름 목록 편집을 완료하면 기능 키 F3을 눌러 ISPF 편집 세션을 종료하십시오. 그런 다음 Enter를 입력하여 큐 관리자로 변경사항을 보내십시오.

주의: 이 단계에서 Enter를 누르지 않고 대신 기능 키 F3을 누르면 입력한 업데이트 내용을 잃게 됩니다.

다중 클러스터 전송 큐를 사용하여 시스템 구현

채널이 단일 클러스터에서 사용되거나 중첩 클러스터에서 사용되더라도 차이는 없습니다. 채널을 선택하여 시작되면 채널이 정의에 따라 전송 큐를 선택합니다.

이 태스크 정보

DEFCLXQ 옵션을 사용하는 경우 258 페이지의 『큐와 전환의 자동 정의 사용』의 내용을 참조하십시오.

단계별 접근법을 사용하는 경우 259 페이지의 『단계식 접근 방법을 사용하여 클러스터-송신자 채널 변경』의 내용을 참조하십시오.

큐와 전환의 자동 정의 사용

DEFCLXQ 옵션 사용을 계획하는 경우 이 옵션을 사용합니다. 모든 채널과 모든 새 채널의 큐가 작성됩니다.

프로시저

1. SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE의 정의를 검토하고 필요한 경우 속성을 변경하십시오.
   이 큐는 SCSQPROC(csq4insx) 멤버에서 정의됩니다.
2. SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 모델 큐를 작성하십시오.
3. 이 모델 큐 및 SYSTEM.CLUSTER.TRANSMIT. * * 큐이다.
   z/OS의 경우 채널 시작기 시작된 태스크 사용자 ID에는 다음이 필요합니다.
   - CLASS(MQADMIN)에 대한 제어 액세스

```
ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channelname
```

- CLASS(MQQUEUE)에 대한 업데이트 액세스

```
ssid.SYSTEM.CLUSTER.TRANSMIT.channelname
```

단계식 접근 방법을 사용하여 클러스터-송신자 채널 변경

이 프로세스를 통해 엔터프라이즈 요구에 맞게 다양한 시점에 새 클러스터-송신자 채널로 이동할 수 있습니다.

시작하기 전에

- 비즈니스 애플리케이션과 사용되는 채널을 식별하십시오.
- 사용하는 큐의 경우, 큐가 있는 클러스터를 표시하십시오.
- 연결 이름을 표시할 채널, 리모트 큐 관리자의 이름, 채널이 지원하는 클러스터를 표시하십시오.

이 태스크 정보

- 전송 큐를 작성하십시오. z/OS의 경우 큐에 사용하는 페이지 세트를 고려해야 할 수 있습니다.
- 큐에 대한 보안 정책을 설정하십시오.
- 이 큐 이름을 포함하도록 큐 모니터링을 변경하십시오.
- 이 전송 큐를 사용할 채널을 결정하십시오. 채널은 유사한 이름을 가져야 하므로 CLCHNAME의 일반 문자 '*'가 채널을 식별합니다.
- 새 기능을 사용할 수 있는 경우, 전송 큐를 대체하여 이 전송 큐를 사용할 채널의 이름을 지정하십시오(예: CLUSTER1.TOPARIS또는 CLUSTER1.* 또는 *.TOPARIS
- 채널 시작

프로시저

1. DIS CLUSQMGR(xxxx) XMITQ 명령을 사용하여 클러스터에 정의된 클러스터-송신자 채널을 표시하십시오. 여기서 *xxxx*는 리모트 큐 관리자의 이름입니다.
2. 전송 큐에 대한 보안 프로파일을 설정하고 채널 시작기에 큐 액세스 권한을 부여하십시오.
3. 사용될 전송 큐를 정의하고 USAGE(XMITQ) INDXTYPE(CORRELID) SHARE 및 CLCHNAME(*value*)을 지정하십시오.

   채널 시작기 시작된 태스크 사용자 ID에는 다음 액세스 권한이 필요합니다.

   ```
   alter class(MQADMIN) ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channel
   update class(MQQUEUE ssid.SYSTEM.CLUSTER.TRANSMIT.channel
   ```

   또한 SWITCH 명령을 사용하는 사용자 ID에는 다음 액세스 권한이 필요합니다.

   ```
   alter cl(MQADMIN) ssid.QUEUE.queuename
   ```

4. 채널을 중지하고 다시 시작하십시오.

   채널은 채널이 MQSC 명령을 사용하기 시작하거나 CSQUTIL을 사용할 때 변경됩니다. CSQUTIL의 SWITCH CHANNEL(*) STATUS를 사용하여 재시작해야 하는 채널을 식별할 수 있습니다.

   채널이 시작될 때 문제점이 있는 경우 채널을 중지하고 문제점을 해결한 후 채널을 재시작하십시오.

   CLCHNAME 속성은 필요할 때마다 변경할 수 있습니다.

   사용되는 CLCHNAME의 값은 채널이 시작될 때의 값이므로 채널이 시작된 시점부터 계속 정의를 사용하는 동안 CLCHNAME 정의를 변경할 수 있습니다. 채널은 재시작될 때 새 정의를 사용합니다.

변경 실행 취소

결과가 예상과 다른 경우 변경사항을 백아웃하는 프로세스가 필요합니다.

잠재적인 문제점

새 전송 큐가 예상과 다른 경우:

1. CLCHNAME이 예상과 같은지 확인하십시오.
2. 작업 로그를 검토하여 전환 프로세스가 완료되었는지 여부를 확인하십시오. 완료되지 않은 경우 대기 후 나중에 채널의 새 전송 큐를 확인하십시오.

여러 클러스터 전송 큐를 사용하는 경우에는 전송 큐 정의를 명시적으로 설계하여 복잡한 중첩 구성을 방지해야 합니다. 이러한 방식으로 문제점이 있는 경우 원래 큐와 구성으로 되돌릴 수 있습니다.

다른 전송 큐를 사용하기 위해 이동하는 동안 문제점이 발생하는 경우에는 문제점을 해결해야 변경을 계속 진행할 수 있습니다.

새 변경 요청을 수행하려면 기존 변경 요청이 완료되어야 합니다. 예를 들어, 다음 작업을 수행할 수 있습니다.

1. 최대 용량이 1인 새 전송 큐를 정의하십시오. 전송 대기 중인 메시지는 10개입니다.
2. 전송 큐를 변경하여 CLCHNAME 매개변수에서 채널 이름을 지정하십시오.
3. 채널을 중지하고 다시 시작하십시오. 메시지 이동 시도가 실패하고 문제점을 보고합니다.
4. 전송 큐의 CLCHNAME 매개변수가 공백이 되도록 변경하십시오.
5. 채널을 중지하고 다시 시작하십시오. 채널이 원래 요청을 완료하려고 계속 시도하므로 채널이 새 전송 큐를 계속 사용합니다.
6. 문제점을 해결하고 채널을 재시작해야 하므로 메시지 이동이 성공적으로 완료됩니다.

다음에 채널이 재시작될 때 변경사항이 적용되므로 CLCHNAME을 공백으로 설정한 경우 채널이 지정된 전송 큐를 사용하지 않습니다.

이 예에서 전송 큐의 CLCHNAME을 공백으로 변경한다고 해서 반드시 채널이 SYSTEM.CLUSTER.TRANSMIT 큐를 사용하는 것은 아닙니다. CLCHNAME 매개변수가 채널 이름과 일치하는 다른 전송 큐가 있을 수 있기 때문입니다. 예를 들어, 일반 이름 또는 큐 관리자 속성 DEFCLXQ가 채널로 설정될 수 있으므로 채널은 SYSTEM.CLUSTER.TRANSMIT 큐 대신 동적 큐를 사용합니다.

IBM MQ 관리 프로그램 작성

큐 관리자를 관리하기 위한 사용자 고유의 애플리케이션 프로그램을 작성할 수 있습니다. 이 주제를 통해 사용자 고유의 관리 프로그램을 작성하기 위한 요구사항을 이해할 수 있습니다.

일반 용도 프로그래밍 인터페이스 시작 정보

이 주제 세트에는 IBM MQ 애플리케이션 프로그램에서 IBM MQ 명령을 실행할 수 있는 힌트와 지침이 포함되어 있습니다.

참고: 이 주제에서는 C 언어 표기를 사용하여 MQI 호출을 설명합니다. COBOL, PL/I, 어셈블러 언어에서 수행하는 일반적인 호출은 함수 호출 매뉴얼을 참조하십시오.

작동 방법 이해

대략적으로 애플리케이션 프로그램에서 명령을 실행하기 위한 프로시저는 다음과 같습니다.

1. IBM MQ 메시지 유형, 요청 메시지에 IBM MQ 명령을 빌드하십시오. 명령은 MQSC 또는 PCF 형식일 수 있습니다.
2. 특수 큐, 시스템 명령 입력 큐로 이 메시지를 보내십시오(MQPUT 사용). IBM MQ 명령 프로세서가 명령을 실행합니다.
3. 명령의 결과를 응답 대상 큐에서 응답 메시지로 검색하십시오(MQGET 사용). 이러한 메시지에는 명령 성공 여부와 명령이 성공한 경우 해당 결과를 판별하는 데 필요한 사용자 메시지가 포함됩니다.

그런 다음 결과를 처리하는 것은 애플리케이션 프로그램의 역할입니다.

이 주제 세트에 포함되는 내용은 다음과 같습니다.

관리 프로그램을 위한 큐 준비

관리 프로그램에는 시스템 명령 입력에 대한 여러 개의 사전 정의된 큐와 수신 응답이 필요합니다.

이 절의 내용은 MQSC 형식의 명령에 적용됩니다. PCF 형식의 명령은 10 페이지의 『PCF(Programmable Command Format) 사용』의 내용을 참조하십시오.

MQPUT 또는 MQGET 호출을 실행하려면 먼저 사용하려는 큐를 정의한 다음 열어야 합니다.

시스템 명령 입력 큐 정의

시스템 명령 입력 큐는 로컬 큐 SYSTEM.COMMAND.INPUT입니다. 제공된 CSQINP2 초기화 데이터 세트, thlqual.SCSQPROC(CSQ4INSG)에는 시스템 명령 입력 큐에 대한 기본 정의가 포함됩니다. 다른 플랫폼의 IBM MQ 와의 호환성을 위해 SYSTEM.ADMIN.COMMAND.QUEUE 라는 이 큐의 별명도 제공됩니다. 자세한 정보는 IBM MQ와 함께 제공되는 샘플 정의를 참조하십시오.

응답 대상 큐 정의

IBM MQ 명령 프로세서에서 응답 메시지를 수신하려면 응답 대상 큐를 정의해야 합니다. 이 큐는 응답 메시지를 큐에 넣을 수 있는 속성을 가진 큐입니다. 그러나 정상 작동을 위해서는 다음 속성을 지정하십시오.

- USAGE(NORMAL)
- NOTRIGGER(애플리케이션이 트리거를 사용하지 않는 경우)

명령의 지속 메시지를 사용해서는 안 됩니다. 지속 메시지를 사용하는 경우에는 응답 대상 큐가 임시 동적 큐가 아니어야 합니다.

제공된 CSQINP2 초기화 데이터 세트, thlqual.SCSQPROC(CSQ4INSG)에는 모델 큐 SYSTEM.COMMAND.REPLY.MODE에 대한 정의가 포함됩니다. 이 모델을 사용하여 동적 응답 대상 큐를 작성할 수 있습니다.

참고: 명령 프로세서로 생성되는 응답의 최대 길이는 15 000바이트입니다.

영구적 동적 큐를 응답 대상 큐로 사용하는 경우, 애플리케이션은 큐 삭제를 시도하기 전에 모든 PUT 및 GET 조작을 완료하기 위한 시간을 허용해야 하며 그렇지 않으면 MQRC2055 (MQRC_Q_NOT_EMPTY)가 리턴될 수 있습니다. 이러한 경우에는 몇 초 후에 큐 삭제를 다시 시도하십시오.

시스템 명령 입력 큐 열기

시스템 명령 입력 큐를 열려면 애플리케이션 프로그램이 큐 관리자에 연결되어야 합니다. 이를 수행하려면 MQI 호출 MQCONN 또는 MQCONNX를 사용하십시오.

그런 다음 MQI 호출 MQOPEN을 사용하여 시스템 명령 입력 큐를 여십시오. 이 호출을 사용하려면 다음을 수행하십시오.

1. *Options* 매개변수를 MQOO_OUTPUT로 설정하십시오.
2. MQOD 오브젝트 디스크립터 필드를 다음과 같이 설정하십시오.

   ObjectType
   MQOT_Q(오브젝트가 큐임)

   ObjectName
   SYSTEM.COMMAND.INPUT

   ObjectQMgrName
   로컬 큐 관리자로 요청 메시지를 보내려면 이 필드를 공백으로 두십시오. 이는 명령이 로컬로 처리됨을 의미합니다.

   IBM MQ 명령을 리모트 큐 관리자에서 처리하려면 여기에 이름을 입력하십시오. 또한 분산 큐잉 및 클러스터에 설명된 대로 올바른 큐 및 링크가 설정되어 있어야 합니다.

응답 대상 큐 열기

IBM MQ 명령에서 응답을 검색하려면 응답 대상 큐를 열어야 합니다. 이를 수행하기 위한 한 가지 방법은 MQOPEN 호출에 모델 큐, SYSTEM.COMMAND.REPLY.MODEL을 지정하여 영구 동적 큐를 응답 대상 큐로 작성하는 것입니다. 이 호출을 사용하려면 다음을 수행하십시오.

1. *Options* 매개변수를 MQOO_INPUT_SHARED로 설정하십시오.
2. MQOD 오브젝트 디스크립터 필드를 다음과 같이 설정하십시오.

   ObjectType
   MQOT_Q(오브젝트가 큐임)

   ObjectName
   응답 대상 큐의 이름. 지정하는 큐 이름이 모델 큐 오브젝트의 이름이면 큐 관리자는 동적 큐를 작성합니다.

   ObjectQMgrName
   로컬 큐 관리자에서 응답을 수신하려면 이 필드를 공백으로 두십시오.

   DynamicQName
   작성될 동적 큐의 이름을 지정합니다.

명령 서버 사용

명령 서버는 명령 프로세서 컴포넌트 관련 작업을 수행하는 IBM MQ 컴포넌트입니다. 형식화된 메시지를 메시지를 해석하는 명령 서버로 보내고 관리 요청을 실행한 후 관리 애플리케이션으로 다시 응답을 보낼 수 있습니다.

이 명령 서버는 시스템 명령 입력 큐로부터 요청 메시지를 읽고 확인한 다음 올바른 메시지를 명령 프로세서에 명령으로 전달합니다. 명령 프로세서는 명령을 처리하며 응답을 관리자가 지정하는 응답 대상 큐에 응답 메시지로 넣습니다. 첫 번째 응답 메시지에는 사용자 메시지 CSQN205I이 포함됩니다. 자세한 정보는 265 페이지의 『명령 서버에서 응답 메시지 해석』의 내용을 참조하십시오. 명령 서버는 또한 채널 시작기와 큐 공유 그룹 명령이 실행된 모든 위치에서 명령을 처리합니다.

명령을 처리하는 큐 관리자 식별

관리 프로그램에서 실행하는 명령을 처리하는 큐 관리자는 메시지를 넣는 시스템 명령 입력 큐를 소유하는 큐 관리자입니다.

명령 서버 시작

일반적으로 명령 서버는 큐 관리자가 시작될 때 자동으로 시작됩니다. 이 서버는 START QMGR 명령에서 메시지 CSQ9022I 'START QMGR' NORMAL COMPLETION이 리턴되는 즉시 사용할 수 있습니다. 시스템 종료 단계에서 연결된 모든 태스크 연결이 끊어지면 명령 서버가 중지됩니다.

START CMDSERV 및 STOP CMDSERV 명령을 사용하여 명령 서버를 직접 제어할 수 있습니다. IBM MQ가 재시작될 때 명령 서버가 자동으로 시작되는 것을 방지하려면 CSQINP1 또는 CSQINP2 초기화 데이터 세트에 STOP CMDSERV 명령을 추가하면 됩니다. 그러나 이 명령은 채널 시작기 또는 큐 공유 그룹 명령 처리를 방해하므로 권장되지는 않습니다.

STOP CMDSERV 명령은 현재 메시지 처리를 완료하자마자 또는 처리하는 메시지가 없는 경우 즉시 명령 서버를 중지합니다.

명령 서버가 프로그램에서 STOP CMDSERV 명령으로 중지된 경우에는 프로그램의 다른 명령을 처리할 수 없습니다. 명령 서버를 재시작하려면 z/OS 콘솔에서 START CMDSERV 명령을 실행해야 합니다.

큐 관리자가 실행 중인 동안 명령 서버를 중지했다가 재시작하면 명령 서버가 중지될 때 시스템 명령 입력 큐에 있는 모든 메시지가 명령 서버가 재시작될 때 처리됩니다. 그러나 명령 서버가 중지된 후 큐 관리자를 중지했다가 재시작하면 명령 서버가 재시작될 때 시스템 명령 입력 큐의 지속 메시지만 처리됩니다. 시스템 명령 입력 큐의 모든 비지속 메시지는 손실됩니다.

명령 서버로 명령 보내기

각 명령마다 명령을 포함하는 메시지를 빌드한 다음 시스템 명령 입력 큐에 넣습니다.

IBM MQ 명령을 포함하는 메시지 빌드

필수 명령을 포함하는 요청 메시지를 빌드하여 IBM MQ 명령을 애플리케이션 프로그램에 통합할 수 있습니다. 해당 명령마다 다음을 수행하십시오.

1. 명령을 나타내는 문자열을 포함하는 버퍼를 작성하십시오.
2. 호출의 *buffer* 매개변수에 버퍼 이름을 지정하는 MQPUT 호출을 발행하십시오.

C로 이 작업을 수행할 수 있는 가장 간단한 방법은 'char'을 사용하여 버퍼를 정의하는 것입니다. 예를 들면, 다음과 같습니다.

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

명령을 빌드할 때 널(Null) 종료 문자열을 사용하십시오. 이 방식으로 정의된 명령 시작 위치에서 명령 접두부 문자열(CPF)을 지정하지 마십시오. 이는 다른 큐 관리자에서 실행하려는 경우에는 명령 스크립트를 대체하지 않아도 됨을 의미합니다. 그러나 응답 대상 큐에 넣는 응답 메시지에 CPF가 포함된다는 사실을 고려해야 합니다.

명령 서버는 인용 부호로 묶지 않은 경우 모든 소문자를 대문자로 변환합니다.

명령의 최대 길이는 32 762자입니다.

시스템 명령 입력 큐에 메시지 넣기

MQPUT 호출을 사용하여 명령을 포함하는 요청 메시지를 시스템 명령 입력 큐에 넣을 수 있습니다. 이 호출에서는 이미 연 응답 대상 큐의 이름을 지정합니다.

MQPUT 호출을 사용하려면 다음을 수행하십시오.

1. 다음 MQPUT 매개변수를 설정하십시오.

   Hconn
   MQCONN 또는 MQCONNX 호출로 리턴된 연결 핸들

   Hobj
   시스템 명령 입력 큐에 대한 MQOPEN 호출로 리턴된 오브젝트 핸들

   BufferLength
   형식화된 명령의 길이

   Buffer
   명령을 포함하는 버퍼의 이름
2. 다음 MQMD 필드를 설정하십시오.

   MsgType
   MQMT_REQUEST

   Format
   MQFMT_STRING 또는 MQFMT_NONE

   큐 관리자와 동일한 코드 페이지를 사용하지 않는 경우, 명령 서버가 메시지를 변환할 수 있도록 *CodedCharSetId*를 적절하게 설정하고 MQFMT_STRING을 설정하십시오. MQFMT_ADMIN을 설정하면 명령이 PCF로 해석되므로 설정하지 마십시오.

   ReplyToQ
   응답 대상 큐의 이름

   ReplyToQMgr
   로컬 큐 관리자로 응답을 보내려면 이 필드를 공백으로 두십시오. IBM MQ 명령을 리모트 큐 관리자로 보내려면 여기에 이름을 입력하십시오. 또한 분산 큐잉 및 클러스터에 설명된 대로 올바른 큐 및 링크가 설정되어 있어야 합니다.
3. 필요한 경우 다른 MQMD 필드를 설정하십시오. 일반적으로 명령의 비지속 메시지를 사용해야 합니다.

4. 필요에 따라 *PutMsgOpts* 옵션을 설정하십시오.

   MQPMO_SYNCPOINT(기본값)를 지정하는 경우 MQPUT 호출 다음에 동기점 호출이 와야 합니다.

MQPUT1 및 시스템 명령 입력 큐 사용
시스템 명령 입력 큐에 메시지를 하나만 넣으려면 MQPUT1 호출을 사용하면 됩니다. 이 호출은 MQOPEN 의 함수, 단일 메시지의 MQPUT, MQCLOSE 순으로 모두 하나의 호출에 결합합니다. 이 호출을 사용하는 경우 그에 따라 매개변수를 수정하십시오. 자세한 내용은 MQPUT1 호출을 사용하여 큐에 단일 메시지 넣기를 참조하십시오.

명령에 대한 응답 검색

명령 서버는 수신한 각 요청 메시지에 대한 응답 큐로 응답을 보냅니다. 관리자 애플리케이션이 응답 메시지를 수신 및 처리해야 합니다.

명령 프로세서가 명령을 처리할 때 MQPUT 호출에서 지정된 응답 대상 큐에 응답 메시지를 넣습니다. 명령 서버는 수신한 명령 메시지와 지속성이 같은 응답 메시지를 보냅니다.

응답 대기

MQGET 호출을 사용하여 요청 메시지에서 응답을 검색하십시오. 하나의 요청 메시지가 여러 응답 메시지를 생성할 수 있습니다. 자세한 내용은 265 페이지의 『명령 서버에서 응답 메시지 해석』의 내용을 참조하십시오.

MQGET 호출이 응답 메시지가 생성될 때까지 대기하는 시간 간격을 지정할 수 있습니다. 응답을 수신하지 않으면 266 페이지의 『응답을 수신하지 않는 경우』 주제에서 시작되는 체크리스트를 사용하십시오.

MQGET 호출을 사용하려면 다음을 수행하십시오.

1. 다음 매개변수를 설정하십시오.

   Hconn
   MQCONN 또는 MQCONNX 호출로 리턴된 연결 핸들

   Hobj
   응답 대상 큐에 대한 MQOPEN 호출로 리턴된 오브젝트 핸들

   Buffer
   응답을 수신할 영역의 이름입니다.

   BufferLength
   응답을 수신할 버퍼의 길입니다. 최소 80바이트여야 합니다.
2. 실행한 명령에서만 응답을 가져오려면 해당 *MsgId* 및 *CorrelId* 필드를 지정해야 합니다. 이 필드는 MQPUT 호출에 지정한 보고서 옵션, MQMD_REPORT에 따라 다릅니다.

   MQRO_NONE
   2진 0, '00...00'(24개 널)

   MQRO_NEW_MSG_ID
   2진 0, '00...00'(24개 널)

   이들 옵션이 지정되지 않은 경우 기본값입니다.

   MQRO_PASS_MSG_ID
   MQPUT 의 *MsgId*

   MQRO_NONE
   MQPUT 호출의 *MsgId*.

   MQRO_COPY_MSG_ID_TO_CORREL_ID
   MQPUT 호출의 *MsgId*.

   이들 옵션이 지정되지 않은 경우 기본값입니다.

MQRO_PASS_CORREL_ID
MQPUT 호출의 *CorrelId*.

보고서 옵션에 대한 자세한 정보는 보고서 옵션 및 메시지 플래그를 참조하십시오.

3. 다음 *GetMsgOpts* 필드를 설정하십시오.

Options
MQGMO_WAIT

큐 관리자와 동일한 코드 페이지를 사용하지 않는 경우, MQGMO_CONVERT를 설정하고 *CodedCharSetId*를 MQMD에서 적절하게 설정하십시오.

WaitInterval
로컬 큐 관리자로부터의 응답의 경우, 5초를 시도합니다. 밀리초 단위로 코드화되는 경우에는 5 000이 됩니다. 리모트 큐 관리자로부터의 응답과 채널 제어 및 상태 명령의 경우, 30초를 시도합니다. 밀리초 단위로 코드화되는 경우에는 30 000이 됩니다.

제거된 메시지

명령 서버가 요청 메시지가 올바르지 않은 것을 발견하면 이 메시지를 제거하고 CSQN205I 메시지를 이름 지정된 응답 대상 큐에 기록합니다. 응답 대상 큐가 없으면 CSQN205I 메시지를 데드-레터 큐에 넣습니다. 이 메시지의 리턴 코드는 원래 요청 메시지가 올바르지 않은 이유를 보여줍니다.

00D5020F	MQMT_REQUEST 유형이 아닙니다.
00D50210	길이가 0입니다.
00D50212	32 762바이트보다 깁니다.
00D50211	모든 공백을 포함합니다.
00D5483E	변환이 필요했지만 *Format*이 MQFMT_STRING이 아닙니다.
기타	명령 서버 코드를 참조하십시오.

명령 서버 응답 메시지 디스크립터

응답 메시지의 경우, 다음 MQMD 메시지 디스크립터 필드가 설정됩니다.

MsgType	MQMT_REPLY
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
Priority	발행한 메시지에서 MQMD의 경우
Persistence	발행한 메시지에서 MQMD의 경우
CorrelId	MQPUT 보고서 옵션에 따라 다릅니다.
ReplyToQ	없음

명령 서버는 MQPMO 구조의 *Options* 필드를 MQPMO_NO_SYNCPOINT로 설정합니다. 이는 다음 동기점에서 그룹으로서가 아닌 응답이 작성될 때 검색할 수 있음을 의미합니다.

명령 서버에서 응답 메시지 해석

IBM MQ가 올바르게 처리하는 각 요청 메시지는 두 개 이상의 응답 메시지를 생성합니다. 각 응답 메시지에는 단일 IBM MQ 사용자 메시지가 포함됩니다.

응답 길이는 실행된 명령에 따라 다릅니다. 가장 긴 응답은 DISPLAY NAMELIST에서 가져올 수 있으며 최대 길이는 15 000바이트입니다.

첫 번째 사용자 메시지, CSQN205I에는 항상 다음 항목이 포함됩니다.

- 나머지 응답을 가져오기 위해 루프에서 카운터로 사용할 수 있는 응답 개수(10진수). 개수에는 이 첫 번째 메시지가 포함됩니다.
- 명령 프리프로세서로부터의 리턴 코드
- 명령 프로세서의 이유 코드인 이유 코드입니다.

이 메시지는 CPF를 포함하지 않습니다.

예를 들면, 다음과 같습니다.

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
```

COUNT 필드의 길이는 8바이트이고 오른쪽으로 정렬됩니다. 항상 위치 18에서 시작됩니다. 즉, 'COUNT =' 다음에 시작됩니다. RETURN 필드는 문자 16진수로 길이가 8바이트이고 위치 35에서 'RETURN =' 바로 뒤에 있습니다. REASON 필드의 길이는 8바이트 16진 문자이며 위치 52에서 'REASON=' 바로 뒤에 옵니다.

RETURN= 값이 00000000이고 REASON= 값이 00000004이면 응답 메시지 세트가 불완전한 상태입니다. CSQN205I 메시지로 표시된 응답을 검색한 후 추가 MQGET 호출을 발행하여 추가 응답 세트를 대기하십시오. 다음 응답 세트의 첫 번째 메시지 역시 CSQN205I이며 이는 현재 응답 수와 남아 있는 응답 수를 표시합니다.

개별 메시지에 대한 자세한 정보는 IBM MQ for z/OS 메시지, 완료 및 이유 코드 문서를 참조하십시오.

영어 이외의 언어 기능을 사용하는 경우에는 응답의 텍스트와 레이아웃이 여기에 표시되는 것과 다릅니다. 그러나 CSQN205I 메시지에 나타나는 개수 및 리턴 코드의 크기와 위치는 같습니다.

응답을 수신하지 않는 경우

명령 서버에 대한 요청 응답을 수신하지 않는 경우 수행할 수 있는 일련의 단계가 있습니다.

요청 메시지에 대한 응답을 수신하지 않는 경우 다음 체크리스트를 수행하십시오.

- 명령 서버가 실행 중입니까?
- *WaitInterval* 값이 충분히 깁니까?
- 시스템 명령 입력 및 응답 대상 큐가 올바르게 정의되었습니까?
- 해당 큐에 대한 MQOPEN 호출이 성공적입니까?
- MQPUT 및 MQGET 호출에 시스템 명령 입력 및 응답 대상 큐를 둘 다 사용할 수 있습니까?
- 큐의 MAXDEPTH 및 MAXMSGL 속성 늘리기를 고려했습니까?
- *CorrelId* 및 *MsgId* 필드를 올바르게 사용하고 있습니까?
- 큐 관리자가 계속 실행 중입니까?
- 명령이 올바르게 빌드되었습니까?
- 모든 원격 링크가 정의되고 올바르게 작동합니까?
- MQPUT 호출이 올바르게 정의되었습니까?
- 응답 대상 큐가 영구적 동적 큐가 아닌 임시 동적 큐로 정의되었습니까? 요청 메시지가 지속적인 경우에는 응답에 영구적 동적 큐를 사용해야 합니다.

명령 서버가 응답을 생성하지만 사용자가 지정하는 응답 대상 큐에 응답을 쓸 수 없는 경우에는 데드-레터 큐에 응답을 씁니다.

MGCRE를 사용하여 명령 전달

올바른 권한이 있으면 z/OS 서비스 루틴을 사용하여 여러 큐 관리자에 대한 요청을 작성할 수 있습니다.

올바른 권한이 있으면 MGCRE (SVC 34) z/OS 서비스로 사용자 프로그램에서 여러 큐 관리자로 IBM MQ 명령을 전달할 수 있습니다. CPF 값은 명령이 지정되는 특정 큐 관리자를 식별합니다. CPF에 대한 정보는 명령 보안 및 명령 자원 보안을 위한 사용자 ID 및 242 페이지의 『큐 관리자 명령 실행』의 내용을 참조하십시오.

MGCRE를 사용하면 명령 및 응답 토큰(CART)으로 명령에 대한 직접 응답을 가져올 수 있습니다.

명령 및 해당 응답 예

이 주제를 명령 서버로의 명령과 명령 서버로부터의 응답을 보여주는 일련의 예로 사용합니다.

다음은 IBM MQ 메시지에 빌드할 수 있는 명령과 응답인 사용자 메시지의 예입니다. 달리 명시되지 않는 한 응답의 각 행은 개별 메시지입니다.

- DEFINE 명령의 메시지
- DELETE 명령의 메시지
- DISPLAY 명령의 메시지
- CMDSCOPE가 있는 명령의 메시지
- CMDSCOPE 명령을 생성하는 명령의 메시지

DEFINE 명령의 메시지

다음 명령은

```
DEFINE QLOCAL(Q1)
```

다음 메시지를 생성합니다.

```
CSQN205I    COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

이 응답 메시지는 정상 완료 시 생성됩니다.

DELETE 명령의 메시지

다음 명령은

```
DELETE QLOCAL(Q2)
```

다음 메시지를 생성합니다.

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
CSQM125I +CSQ1 CSQMUQLC QLOCAL (Q2) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E +CSQ1 CSQMUQLC FAILURE REASON CODE X'00D44002'
CSQ9023E +CSQ1 CSQMUQLC ' DELETE QLOCAL' ABNORMAL COMPLETION
```

이 메시지는 로컬 큐 Q2가 존재하지 않음을 나타냅니다.

DISPLAY 명령의 메시지

다음 예는 일부 DISPLAY 명령의 응답을 표시합니다.

데드-레터 큐 이름 찾기

큐 관리자의 데드-레터 큐 이름을 찾으려면 애플리케이션 프로그램에서 다음 명령을 실행하십시오.

```
DISPLAY QMGR DEADQ
```

필수 이름을 추출할 수 있는 다음 세 개 사용자 메시지가 리턴됩니다.

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE          )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

DISPLAY QUEUE 명령의 메시지

다음 예는 명령의 결과가 해당 명령에 지정된 속성에 따라 어떻게 다른지를 보여줍니다.

예제 1

명령을 사용하여 로컬 큐를 정의합니다.

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

애플리케이션 프로그램에서 다음 명령을 실행하는 경우:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

다음 세 개 사용자 메시지가 리턴됩니다.

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1                       ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
DESCR(A sample queue
) SHARE  GET(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

참고: 여기에는 두 번째 메시지, CSQM401I가 네 행에 표시됩니다.

예제 2

다음 두 가지 큐의 이름은 문자 A로 시작됩니다.

- A1은 해당 PUT 속성이 DISABLED로 설정된 로컬 큐입니다.
- A2는 해당 PUT 속성이 ENABLED로 설정된 리모트 큐입니다.

애플리케이션 프로그램에서 다음 명령을 실행하는 경우:

```
DISPLAY QUEUE(A*) PUT
```

다음 네 개 사용자 메시지가 리턴됩니다.

```
CSQN205I    COUNT=    4, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(A1                      ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
PUT(DISABLED )
CSQM406I +CSQ1 QUEUE(A2                      ) TYPE(
QREMOTE ) PUT(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

참고: 여기서 두 번째, 세 번째 메시지인 CSQM401I 및 CSQM406I는 세 행, 두 행에 표시됩니다.

DISPLAY NAMELIST 명령의 메시지

다음 명령을 사용하여 이름 목록을 정의합니다.

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

애플리케이션 프로그램에서 다음 명령을 실행하는 경우:

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

다음 세 개 사용자 메시지가 리턴됩니다.

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                      ) QS
GDISP(QMGR   ) NAMCOUNT(     2) NAMES(Q1
,SAMPLE_QUEUE                 )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY NAMELIST' NORMAL COMPLETION
```

참고: 여기에는 두 번째 메시지, CSQM407I가 네 행에 표시됩니다.

CMDSCOPE 명령 메시지

다음 예는 CMDSCOPE 속성과 함께 입력된 명령의 응답을 보여줍니다.

ALTER PROCESS 명령의 메시지

다음 명령은

```
ALT PRO(V4) CMDSCOPE(*)
```

다음 메시지를 생성합니다.

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'ALT PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ26
CSQM125I !MQ26 CSQMMSGP PROCESS(V4) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E !MQ26 CSQMMSGP FAILURE REASON CODE X'00D44002'
CSQ9023E !MQ26 CSQMMSGP ' ALT PRO' ABNORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' ALT PRO' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=0000000C, REASON=00000008
CSQN123E !MQ25 'ALT PRO' command for CMDSCOPE(*) abnormal completion
```

이 메시지는 큐 관리자 MQ25에서 명령이 입력되고 두 개 큐 관리자(MQ25 및 MQ26)로 전송되었음을 알려줍니다. 명령은 MQ25에서는 성공했지만 MQ26에는 프로세스 정의가 존재하지 않아 해당 큐 관리자에서 명령이 실패했습니다.

DISPLAY PROCESS 명령의 메시지

다음 명령은

```
DIS PRO(V*) CMDSCOPE(*)
```

다음 메시지를 생성합니다.

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ26
CSQM408I !MQ26 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ26 PROCESS(V3) QSGDISP(QMGR)
CSQ9022I !MQ26 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    7, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ25
CSQM408I !MQ25 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ25 PROCESS(V2) QSGDISP(GROUP)
CSQM408I !MQ25 PROCESS(V3) QSGDISP(QMGR)
CSQM408I !MQ25 PROCESS(V4) QSGDISP(QMGR)
CSQ9022I !MQ25 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS PRO' command for CMDSCOPE(*) normal completion
```

이 메시지는 큐 관리자 MQ25에서 명령이 입력되고 두 개 큐 관리자(MQ25 및 MQ26)로 전송되었음을 알려줍니다. 이름이 문자 V로 시작되는 각 큐 관리자에서 모든 프로세스에 대한 정보가 표시됩니다.

DISPLAY CHSTATUS 명령의 메시지

다음 명령은

```
DIS CHS(VT) CMDSCOPE(*)
```

다음 메시지를 생성합니다.

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS CHS' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM422I !MQ25 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ25 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=    4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM422I !MQ26 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ26 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS CHS' command for CMDSCOPE(*) normal completion
```

이 메시지는 큐 관리자 MQ25에서 명령이 입력되고 두 개 큐 관리자(MQ25 및 MQ26)로 전송되었음을 알려줍니다. 각 큐 관리자의 채널 상태에 대한 정보가 표시됩니다.

STOP CHANNEL 명령의 메시지

다음 명령은

```
STOP CHL(VT) CMDSCOPE(*)
```

다음 메시지를 생성합니다.

```
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'STOP CHL' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQM134I !MQ25 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
SQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQM134I !MQ26 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQ9022I !MQ26 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQ9022I !MQ25 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'STOP CHL' command for CMDSCOPE(*) normal completion
```

이 메시지는 큐 관리자 MQ25에서 명령이 입력되고 두 개 큐 관리자(MQ25 및 MQ26)로 전송되었음을 알려줍니다. 각 큐 관리자에서 채널 VT가 중지되었습니다.

CMDSCOPE 명령을 생성하는 명령의 메시지

다음 명령은

```
DEF PRO(V2) QSGDISP(GROUP)
```

다음 메시지를 생성합니다.

```
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQM122I !MQ25 CSQMMSGP ' DEF PRO' COMPLETED FOR QSGDISP(GROUP)
CSQN138I !MQ25 'DEFINE PRO' command generated for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ26
CSQ9022I !MQ26 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DEFINE PRO' command for CMDSCOPE(*) normal completion
```

이 메시지는 큐 관리자 MQ25에서 명령이 입력되었음을 알려줍니다. 공유 저장소에서 오브젝트가 작성될 때 다른 명령이 생성되고 큐 공유 그룹의 모든 활성 큐 관리자로 전송되었습니다(MQ25 및 MQ26).

z/OS z/OS에서 IBM MQ 자원 관리

이 주제의 링크를 사용하여 IBM MQ for z/OS가 사용하는 자원 관리 방법(예를 들어, 로그 파일, 데이터 세트, 페이지 세트, 버퍼 풀, 커플링 기능 구조 관리)을 이해할 수 있습니다.

IBM MQ for z/OS를 사용할 때 완료해야 하는 다른 관리 태스크에 대한 자세한 내용은 다음 링크를 참조하십시오.

- 272 페이지의 『로그 관리』
- 279 페이지의 『부트스트랩 데이터 세트(BSDS) 관리』
- 287 페이지의 『페이지 세트 관리』
- 292 페이지의 『페이지 세트 백업 및 복구 방법』
- 296 페이지의 『CSQUTIL을 사용하여 큐 백업 및 복원 방법』
- 296 페이지의 『버퍼 풀 관리』
- 297 페이지의 『큐 공유 그룹 및 공유 큐 관리』

관련 개념

232 페이지의 『IBM MQ for z/OS 관리』
큐 관리자 및 연관된 자원 관리에는 해당 자원을 활성화하고 관리하기 위해 자주 수행하는 태스크가 포함됩니다. 큐 관리자 및 연관된 자원을 관리할 때 선호하는 메소드를 선택하십시오.

233 페이지의 『IBM MQ for z/OS에 명령 실행』
IBM MQ 스크립트 명령(MQSC)을 배치 또는 대화식 모드로 사용하여 큐 관리자를 제어할 수 있습니다.

304 페이지의 『복구 및 재시작』
이 주제를 통해 IBM MQ가 사용하는 복구 및 재시작 메커니즘을 이해할 수 있습니다.

관련 참조

240 페이지의 『IBM MQ for z/OS 유틸리티』
IBM MQ for z/OS는 간편한 시스템 관리를 위해 사용할 수 있는 유틸리티 프로그램 세트를 제공합니다.

관련 정보

IBM MQ for z/OS 개념
z/OS에서 IBM MQ 환경 계획
z/OS 구성
프로그래밍 가능한 명령 포맷 참조
MQSC 참조
IBM MQ for z/OS 유틸리티 사용

로그 관리

이 주제를 통해 로그 아카이브 프로세스 등 로그 레코드 압축, 로그 레코드 복구, 로그 레코드 인쇄를 사용하여 IBM MQ 로그 파일을 관리하는 방법을 이해할 수 있습니다.

이 주제는 IBM MQ 로그 관리와 관련된 태스크를 설명합니다. 여기에는 다음 절이 포함되어 있습니다.

ARCHIVE LOG 명령으로 로그 아카이브

권한이 부여된 운영자가 ARCHIVE LOG 명령을 사용하여 필요할 때마다 현재 IBM MQ 활성 로그 데이터 세트를 아카이브할 수 있습니다.

ARCHIVE LOG 명령을 실행하면 IBM MQ가 현재 활성 로그 데이터 세트를 자른 다음 비동기 오프로드 프로세스를 실행하고 오프로드 프로세스의 레코드로 BSDS를 업데이트합니다.

ARCHIVE LOG 명령은 MODE(QUIESCE) 옵션을 갖습니다. 이 옵션을 지정하면 커미트 지점 이후에 IBM MQ 작업과 사용자가 일시정지되며 오프로드되기 전에 결과 일관성 지점이 현재 활성 로그에서 캡처됩니다.

오프 사이트 복구를 위한 백업 전략을 계획할 때 MODE(QUIESCE) 옵션 사용을 고려하십시오. 복구 중에 최신 백업 페이지 세트 사본과 함께 아카이브 로그를 사용할 때 데이터 일관성 수를 최소화하는 시스템 전체 일관성 지점을 작성합니다. 예를 들면, 다음과 같습니다.

```
ARCHIVE LOG MODE(QUIESCE)
```

ARCHIVE LOG 명령을 실행할 때 TIME 매개변수를 지정하지 않는 경우 일시정지 기간은 기본적으로 CSQ6ARVP 매크로의 QUIESCE 매개변수 값입니다. ARCHIVE LOG MODE(QUIESCE)를 완료하는 데 필요한 시간이 지정된 시간보다 짧으면 명령이 완료되고 그렇지 않으면 기간이 만기될 때 명령이 실패합니다. 예를 들어 다음과 같이 TIME 옵션을 사용하여 명시적으로 기간을 지정할 수 있습니다.

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

이 명령은 ARCHIVE LOG 처리가 발생하기 전 최대 일시정지 기간(60초)을 지정합니다.

주의: 시간이 중요할 때 TIME 옵션을 사용하면 IBM MQ 자원을 사용하는 모든 작업과 사용자에 대한 IBM MQ 가용성에 크게 영향을 줄 수 있습니다.

기본적으로 명령은 명령을 제출하는 시점부터 비동기식으로 처리됩니다. 명령을 다른 IBM MQ 명령과 동시에 처리하려면 QUIESCE에 WAIT(YES) 옵션을 사용하십시오. z/OS 콘솔은 전체 QUIESCE 기간 동안 IBM MQ 명령 입력에서 잠깁니다.

일시정지 기간 동안:

- 큐 관리자의 작업과 사용자는 커미트 처리를 계속 수행할 수 있지만 커미트 후 IBM MQ 자원 업데이트를 시도하면 일시중단됩니다.
- 데이터를 읽기만 하는 작업과 사용자는 일시중단된 작업 또는 사용자가 유지하는 잠금을 대기할 수 있으므로 영향을 받을 수 있습니다.
- 새 태스크를 시작할 수 있지만 데이터를 업데이트할 수는 없습니다.

DISPLAY LOG 명령의 출력은 CSQV400I 메시지를 사용하여 일시정지가 유효함을 표시합니다. 예를 들면, 다음과 같습니다.

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     4000
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
WRTHRSH     20
DEALLCT     0
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full DSName
1      68  VICY.CSQ1.LOGCOPY1.DS01
2      68  VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2014-04-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

모든 업데이트가 일시정지되면 BSDS의 일시정지 실행 기록 레코드가 활성 로그 데이터 세트가 잘린 날짜 및 시간과 현재 활성 로그 데이터 세트에서 마지막으로 기록된 RBA로 업데이트됩니다. IBM MQ는 현재 활성 로그 데이터 세트를 자르고 사용 가능한 다음 활성 로그 데이터 세트로 전환되며 오프로드 프로세스가 시작되었음을 나타내는 CSQJ311I 메시지를 발행합니다.

일시정지 기간이 만기되기 전에 업데이트를 일시정지할 수 없으면 IBM MQ가 CSQJ317I 메시지를 발행하고 ARCHIVE LOG 처리가 종료됩니다. 현재 활성 로그 데이터 세트는 잘리지 않고 사용 가능한 다음 로그 데이터 세트로 전환되지 않으며 오프로드 프로세스가 시작되지 않습니다.

일시정지 성공 여부에 관계 없이, 일시정지된 모든 사용자와 작업이 재개되며, IBM MQ가 일시정지가 종료되고 업데이트 활동이 재개되었음을 나타내는 CSQJ312I 메시지를 발행합니다.

현재 활성 로그가 마지막으로 사용 가능한 활성 로그 데이터 세트일 때 ARCHIVE LOG가 실행되면 명령이 처리되지 않고 IBM MQ가 다음 메시지를 발행합니다.

```
CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
AVAILABLE ACTIVE LOG DATA SET. ARCHIVE LOG PROCESSING
WILL BE TERMINATED
```

다른 ARCHIVE LOG 명령이 이미 진행 중일 때 ARCHIVE LOG가 실행되면 새 명령이 처리되지 않고 IBM MQ가 다음 메시지를 실행합니다.

```
CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS
```

아카이브 도중 실행된 메시지에 대한 정보는 IBM MQ for z/OS 메시지를 참조하십시오.

실패 후 로그 아카이브 프로세스 재시작

로그 아카이브 프로세스에 문제점(예를 들어, 할당 또는 테이프 마운트 관련 문제점)이 있는 경우 활성 로그 아카이브가 일시중단될 수 있습니다. ARCHIVE LOG CANCEL OFFLOAD 명령을 사용하여 아카이브 프로세스를 취소하고 재시작할 수 있습니다. 이 명령은 현재 진행 중인 오프로드 처리를 취소하고 아카이브 프로세스를 재시작합니다. 아카이브되지 않은 가장 오래된 로그 데이터 세트로 시작되며 오프로딩이 필요한 모든 활성 로그 데이터 세트를 통해 진행됩니다. 일시중단된 모든 로그 아카이브 조작이 재시작됩니다.

이 명령은 현재 로그 아카이브 태스크가 더 이상 작동하지 않거나 실패한 이전 시도를 재시작하려는 경우에만 사용합니다. 명령이 비정상적인 오프로드 태스크 종료를 야기하여 덤프가 생성될 수 있기 때문입니다.

아카이브 및 로깅 제어

CSQ6LOGP, CSQ6ARVP, CSQ6SYSP 매크로를 사용하여 압축, 인쇄, 아카이브, 복구, 로깅을 제어할 수 있습니다. 개인용 오브젝트에 대한 변경사항은 IBM MQ 로그에 기록됩니다. 정의가 그룹 전체에 대해 전파되며 로컬로 유지되기 때문에 GROUP 오브젝트(공유 인바운드 채널과 같이)에대한 변경사항도 로깅됩니다.

아카이브 및 로깅의 많은 측면이 큐 관리자가 사용자 정의될 때 시스템 매개변수 모듈의 CSQ6LOGP, CSQ6ARVP, CSQ6SYSP 매크로를 사용하여 설정된 매개변수로 제어됩니다. 이러한 매크로에 대한 자세한 내용은 태스크 17: 시스템 매개변수 모듈 조정을 참조하십시오.

이러한 매개변수 중 일부는 큐 관리자가 IBM MQ MQSC SET LOG, SET SYSTEM, SET ARCHIVE 명령을 사용하여 실행되는 동안 변경될 수 있습니다. 275 페이지의 표 18에 해당 매개변수가 표시되어 있습니다.

표 *18.* 큐 관리자가 실행 중인 동안 변경될 수 있는 아카이브 및 로깅 매개변수

SET 명령	**매개변수**
로그	WRTHRSH, MAXARCH, DEALLCT, MAXRTU, COMPLOG
아카이브	모두
SYSTEM	LOGLOAD

MQSC DISPLAY LOG, DISPLAY ARCHIVE, DISPLAY SYSTEM 명령을 사용하여 모든 매개변수의 설정을 표시할 수 있습니다. 이러한 명령은 또한 아카이브 및 로깅에 대한 상태 정보를 보여줍니다.

로그 압축 제어

다음 중 하나를 사용하여 로그 레코드 압축을 사용 또는 사용하지 않도록 설정할 수 있습니다.

- MQSC의 SET 및 DISPLAY LOG 명령(MQSC 명령 참조)
- PCF 인터페이스 호출 (9 페이지의 『프로그래밍 가능 명령 형식 소개』 참조)
- 시스템 매개변수 모듈에서 CSQ6LOGP 매크로 사용(CSQ6LOGP 사용 참조)

로그 레코드 인쇄

CSQ1LOGP 유틸리티를 사용하여 로그 레코드를 추출 및 인쇄할 수 있습니다. 지시사항은 로그 인쇄 유틸리티를 참조하십시오.

로그 복구

특히 이중 로깅을 사용하는 경우 일반적으로 IBM MQ 로그를 백업 및 복원하지 않아도 됩니다. 그러나 로그의 I/O 오류와 같은 드문 상황에서 로그를 복구해야 할 수 있습니다. 액세스 방법 서비스를 사용하여 데이터 세트를 삭제하고 재정의한 다음 해당 이중 로깅을 복사하십시오.

아카이브 로그 데이터 세트 제거

아카이브 로그 데이터 세트를 제거하고 로그를 자동 또는 수동으로 제거하도록 선택할 수 있습니다.

작업 단위 복구, 페이지 세트 매체 복원(페이지 세트가 손실된 경우) 또는 CF 구조 매체 복원(CF 구조가 손실된 경우)를 수행할 수 있는 충분한 로그 데이터를 유지해야 합니다. 복구에 필요할 수 있는 아카이브 로그 데이터 세트는 제거하지 마십시오. 이러한 아카이브 로그 데이터 세트를 제거하면 필요한 복구 조작을 수행하지 못할 수 있습니다.

아카이브 로그 데이터 세트를 제거할 수 있는 것으로 확인한 경우 다음 중 한 가지 방법으로 이를 수행할 수 있습니다.

- 자동 아카이브 로그 데이터 세트 삭제
- 아카이브 로그 데이터 세트 수동 삭제

자동 아카이브 로그 데이터 세트 삭제

DASD 또는 테이프 관리 시스템을 사용하여 아카이브 로그 데이터 세트를 자동으로 삭제할 수 있습니다. IBM MQ 아카이브 로그 데이터 세트의 보존 기간은 CSQ6ARVP 설치 매크로의 보존 기간 필드 ARCRETN으로 지정됩니다. 자세한 정보는 CSQ6ARVP 사용을 참조하십시오.

보존 기간 기본값은 아카이브 로그가 9999일(최대값) 동안 보존되는 것으로 지정합니다. **보존 기간은 변경할 수 있지만 계획한 백업 주기 수를 수용할 수 있는지 확인해야 합니다**.

IBM MQ는 보존 기간 값을 아카이브 로그 데이터 세트가 작성될 때 JCL 매개변수 RETPD의 값으로 사용합니다.

MVS™/DFP 스토리지 관리 서브시스템(SMS)에서 설정된 보존 기간은 이 IBM MQ 매개변수로 대체할 수 있습니다. 일반적으로 보존 기간은 IBM MQ 또는 SMS로 지정된 작은 값으로 설정됩니다. 스토리지 관리자와 IBM MQ 관리자는 IBM MQ에 적절한 보존 기간 값에 동의해야 합니다.

참고: IBM MQ에는 BSDS에서 아카이브 로그 데이터 세트에 대한 정보를 자동으로 삭제하는 방법이 없습니다. 일부 테이프 관리 시스템이 보존 기간의 외부 수동 오버라이드를 제공하기 때문입니다. 따라서 데이터 세트 보존 기간이 만기되고 테이프 관리 시스템이 데이터 세트를 취소한 후 한참 동안에도 아카이브 로그 데이터 세트에 대한 정보가 BSDS에서 계속 보존될 수 있습니다. 반대로, 최대 아카이브 로그 데이터 세트 수가 초과되어 데이터 세트가 만기 날짜에 도달하기 전에 BSDS 데이터가 삭제될 수 있습니다.

아카이브 로그 데이터 세트가 자동으로 삭제되면 조작으로 BSDS에서 아카이브 로그 목록이 업데이트되지 않습니다. 로그 인벤토리 변경 유틸리티로 BSDS를 업데이트할 수 있습니다(281 페이지의 『BSDS 변경』의 설명 참조). 업데이트가 반드시 수행되지는 않습니다. 오래된 아카이브 로그를 기록하면 BSDS 공간은 부족해지지만 다른 손해는 없습니다.

아카이브 로그 데이터 세트 수동 삭제

CSQI024I 및 CSQI025I 메시지에 식별된 가장 낮은 RBA까지 모든 로그 레코드를 보존해야 합니다. 이 RBA는 방법 1: 전체 백업을 사용하여 복구점을 작성할 때 실행한 DISPLAY USAGE 명령을 사용하여 확보합니다.

로그를 제거하기 전에 비공유 자원의 복구점 작성을 읽어보십시오.

아카이브 로그 데이터 세트 찾기 및 제거

복구에 필요한 최소 로그 RBA를 설정한 경우 다음 프로시저를 수행하여 이전 로그 레코드만 포함하는 아카이브 로그 데이터 세트를 찾을 수 있습니다.

1. 로그 맵 인쇄 유틸리티를 사용하여 BSDS의 컨텐츠를 인쇄하십시오. 출력 예는 로그 맵 인쇄 유틸리티를 참조하십시오.
2. "ARCHIVE LOG COPY n DATA SETS" 출력의 섹션을 찾으십시오. 이중 로깅을 사용하는 경우에는 두 개의 섹션이 있습니다. STARTRBA 및 ENDRBA 열은 각 볼륨에 포함된 RBA의 범위를 보여줍니다. CSQI024I 및 CSQI025I 메시지로 찾은 최소 RBA를 포함하는 범위의 볼륨을 찾으십시오. 이 볼륨은 보존해야 하는 가장 오래된 볼륨입니다. 이중 로깅을 사용하는 경우에는 이러한 볼륨이 두 개 있습니다.

   적절한 범위의 볼륨이 없으면 다음 중 한 가지 경우입니다.

   - 최소 RBA가 아직 아카이브되지 않아 모든 아카이브 로그 볼륨을 제거할 수 있습니다.
   - 볼륨 수가 CSQ6LOGP 매크로의 MAXARCH 매개변수가 허용하는 수를 초과할 때 BSDS의 아카이브 로그 볼륨 목록 줄이 바뀌었습니다. BSDS가 아카이브 로그 볼륨을 등록하지 않으면 해당 볼륨을 복구에 사용할 수 없습니다. 따라서 기존 볼륨에 대한 정보를 BSDS에 추가하는 방법을 고려하십시오. 지시사항은 283 페이지의 『아카이브 로그에 대한 변경사항』의 내용을 참조하십시오.

     또한 MAXARCH 값 늘리기를 고려하십시오. 자세한 정보는 CSQ6LOGP 사용을 참조하십시오.
3. 보존하려는 가장 오래된 볼륨의 STARTRBA 값보다 작은 ENDRBA 값을 갖는 아카이브 로그 데이터 세트 또는 볼륨을 삭제하십시오. 이중 로깅을 사용하는 경우에는 해당 사본을 둘 다 삭제하십시오.

BSDS 항목은 줄이 바뀌므로 BSDS 아카이브 로그 섹션의 처음 몇몇 항목이 맨 아래 항목보다 최신 항목일 수 있습니다. 날짜 및 시간의 조합을 보고 해당 수명을 비교하십시오. 최소 LOGRBA를 포함하는 아카이브 로그에 대한 항목보다 위에 있는 모든 항목을 제거할 수 있는 것으로 가정하지 마십시오.

데이터 세트를 삭제하십시오. 아카이브가 테이프에 있으면 테이프를 지우십시오. DASD에 있으면 z/OS 유틸리티를 실행하여 각 데이터 세트를 삭제하십시오. 그런 다음 BSDS로 기존 아카이브 볼륨만 나열하려면 로그 인벤토리 변경 유틸리티(CSQJU003)를 사용하여 제거된 볼륨에 대한 항목을 삭제하십시오. 283 페이지의 『아카이브 로그에 대한 변경사항』의 예를 참조하십시오.

로그 처리 지연 효과

트랜잭션이 길게 실행되면 작업 단위 로그 레코드가 로그 데이터 세트에 포함될 수 있습니다. IBM MQ는 보유한 로그 데이터의 양과 큐 관리자 재시작 시간을 최적화하기 위해 로그 레코드를 이동하는 기술인 로그 처리 지연을 사용하여 이 시나리오를 처리합니다.

작업 단위가 길다고 간주되면 각 로그 레코드의 표현이 로그 아래에 기록됩니다. 이를 로그 처리 지연이라고 합니다. 이는 로그 파일에 자세히 설명되어 있습니다.

큐 관리자는 실패 후 원본 대신 이러한 처리 지연 로그 레코드를 사용하여 작업 단위 무결성을 보장합니다. 여기에는 다음과 같은 두 가지 이점이 있습니다.

- 작업 단위 조정을 위해 보유해야 하는 로그 데이터의 양이 감소합니다.
- 큐 관리자 재시작 시간에 전달되어야 하는 로그 데이터가 적으므로 큐 관리자가 보다 빨리 재시작됩니다.

처리 지연 로그 레코드에는 매체 복원 조작을 위한 충분한 정보가 포함되지 않습니다.

로그에 저장된 데이터는 두 가지 다른 목적(매체 복원 및 작업 단위 조정)으로 사용됩니다. CF 구조 또는 페이지 세트에 영향을 주는 매체 실패가 발생하면 큐 관리자가 오래된 사본을 복원하고 로그에 포함된 데이터를 사용하여 이를 업데이트함으로써 매체를 장애 지점으로 복구할 수 있습니다. 작업 단위로 수행되는 지속 활동이 로그에 기록되므로 장애 발생 시 백아웃되거나 변경된 자원에서 잠금이 복구될 수 있습니다. 큐 관리자 복구를 사용하기 위해 보유해야 하는 로그 데이터의 양은 이 두 요소의 영향을 받습니다.

매체 복원을 위해서는 최소한 최신 매체 사본에서 매체 복원을 수행하고 백아웃할 수 있는 정도의 충분한 로그 데이터를 보유해야 합니다. 사이트에서 이전 백업에서 복구하는 기능을 규정할 수 있습니다. 작업 단위 무결성을 위해서는 가장 오래된 인플라이트 또는 인다우트(in-doubt) 작업 단위에 대한 로그 데이터를 보유해야 합니다.

시스템 관리를 지원하기 위해 큐 관리자는 각 로그 아카이브에서 오래된 작업 단위를 감지하고 메시지 CSQJ160 및 CSQJ161에 보고합니다. 내부 태스크가 이처럼 오래된 작업 단위에 대한 작업 단위 로그 정보를 읽고 보다 간결한 양식으로 로그 내 현재 위치에 재작성합니다. CSQR026 메시지는 이 시점을 표시합니다. MQSC 명령 DISPLAY USAGE TYPE(DATASET)는 또한 로그 데이터 보유를 관리하는 데 도움을 줄 수 있습니다. 이 명령은 다음과 같은 세 가지 복구 정보를 보고합니다.

1. 작업 단위 복구를 위해 보유해야 하는 로그의 양
2. 페이지 세트의 매체 복원을 위해 보유해야 하는 로그의 양
3. 큐 공유 그룹의 큐 관리자의 경우, CF 구조의 매체 복원을 위해 보유해야 하는 로그의 양

이들 각각에 대해 필요한 가장 오래된 로그 데이터를 데이터 세트에 맵핑하기 위해 시도합니다. 새 작업 단위가 시작되고 중지됨에 따라 (1)이 로그에서 보다 최신 위치로 이동하게 됩니다. 이동하지 않는 경우에는 장기 실행 UOW 메시지가 문제가 있음을 경고합니다. (2)는 큐 관리자가 현재 종료되었다가 재시작된 경우 페이지 세트 매체 복원과 관련됩니다. 페이지 세트를 마지막으로 백업한 시점 또는 페이지 세트 실패 시 사용해야 하는 백업을 알지 못합니다. 버퍼 풀에 보유된 변경사항이 페이지 세트에 기록됨에 따라 일반적으로 체크포인트 처리 중에 로그에서 보다 최신 위치로 이동합니다. (3)의 경우에는 큐 관리자가 큐 공유 그룹에서 이 큐 관리자 또는 다른 큐 관리자에 대해 수행된 CF 구조 백업을 인식합니다. 그러나 CF 구조 복구를 위해서는 마지막 백업 이후 CF 구조와 상호작용한 큐 공유 그룹 내 모든 큐 관리자로부터의 로그 데이터 병합이 필요합니다. 이는 로그 데이터가 시간소인을 기반으로 하는 LRSN(Log Record Sequence Number)으로 식별되며 큐 공유 그룹의 큐 관리자마다 다른 RBA가 아닌 전체 큐 공유 그룹에도 적용됨을 의미합니다. BACKUP CFSTRUCT 명령은 큐 공유 그룹 내 이 큐 관리자 또는 다른 큐 관리자에서 수행되므로 일반적으로 로그에서 보다 최신 위치로 이동합니다.

큐 관리자 로그 재설정

이 주제를 통해 큐 관리자 로그 재설정 방법을 이해할 수 있습니다.

큐 관리자 로그 RBA에서 로그 RBA 범위의 끝에서 0까지 줄 바꾸기를 허용하지 않아야 합니다. 그렇지 않으면 큐 관리자가 가동 중단되며 모든 지속 데이터가 복구할 수 없는 상태가 됩니다. 로그 RBA의 끝은 값이

FFFFFFFFFFFF(6바이트 RBA를 사용 중인 경우)이거나 FFFFFFFFFFFFFFFF(8바이트 RBA를 사용 중인 경우)입니다.

큐 관리자는 사용된 로그 범위가 넓고 계획되지 않은 가동 중단을 방지하기 위한 조치 수행을 계획해야 함을 나타내는 CSQI045I, CSQI046E, CSQI047E, CSQJ031D 및 CSQJ032E 메시지를 발행합니다.

RBA 값이 FFF800000000(6바이트 로그 RBA를 사용 중인 경우) 또는 FFFFFFC000000000(8바이트 로그 RBA를 사용 중인 경우)인 경우 큐 관리자는 이유 코드 00D10257을 표시하며 종료됩니다.

6바이트 로그 RBA를 사용 중인 경우에는 대형 로그 상대 바이트 주소 구현에서 설명하는 프로세스에 따라, 큐 관리자 로그를 재설정하지 않고 큐 관리자가 8바이트 로그 RBA를 사용하도록 변환하는 것을 고려하십시오. 큐 관리자가 8바이트 로그 RBA를 사용하도록 변환하려면 로그를 재설정하는 경우보다 가동 중단 시간이 짧아지므로 로그를 재설정해야 하는 시점까지의 시간이 늘어날 수 있습니다.

큐 관리자 초기화 중에 발행된 CSQJ034I 메시지는 큐 관리자에 대해 구성된 로그 RBA 범위의 끝을 표시하며 사용 중인 로그 RBA가 6바인트인지 아니면 8바인트인지를 판별하는 데 사용할 수 있습니다.

큐 관리자 로그를 재설정하는 절차는 다음과 같습니다.

1. 해석되지 않은 작업 단위를 해석하십시오. 해석되지 않은 작업 단위 수는 메시지 CSQR005I의 큐 관리자 시작 시 INDOUBT 카운트로 표시됩니다. 각 체크포인트에서 또한 큐 관리자가 종료될 때마다 큐 관리자가 자동으로 명령을 실행합니다.

   DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED): 해석되지 않은 작업 단위에 대한 정보를 제공합니다.

   복구 단위 해석에 대한 정보는 복구 인다우트 단위를 해석하는 방법을 참조하십시오. 궁극의 조치는 **RESOLVE INDOUBT** MQSC 명령을 사용하여 복구 인다우트 단위를 수동으로 해석하는 것입니다.

2. 큐 관리자를 문제 없이 종료하십시오.

   STOP QMGR 또는 **STOP QMGR MODE(FORCE)**를 사용할 수 있습니다. 이 두 명령 모두 버퍼풀의 변경된 페이지를 페이지 세트로 보내기 때문입니다.

3. 큐 관리자가 큐 공유 그룹에 포함된 경우 큐 공유 그룹에 있는 모든 구조에 대해 다른 큐 관리자에 대한 CFSTRUCT 백업을 수행하십시오. 이렇게 하면 최신 백업이 이 큐 관리자 로그에 포함되지 않고 CFSTRUCT 복구 시 이 큐 관리자의 로그가 필요하지 않습니다.

4. CSQJU003을 사용하여 새 로그와 BSDS를 정의하십시오(로그 인벤토리 변경 유틸리티 사용에 대한 자세한 정보는 로그 인벤토리 변경 유틸리티 참조).

5. 이 큐 관리자의 모든 페이지 세트에 대해 **CSQUTIL RESETPAGE**를 실행하십시오. 이 기능 사용에 대한 자세한 정보는 페이지 복사 및 로그 재설정을 참조하십시오. 페이지 세트 RBA는 독립적으로 재설정될 수 있으므로 여러 동시 작업(예를 들어, 페이지 세트당 하나)을 제출하여 이 단계의 경과 시간을 줄일 수 있습니다.

6. 큐 관리자를 재시작하십시오.

관련 개념

278 페이지의 『대형 로그 상대 바이트 주소 구현』

IBM MQ for z/OS의 이전 릴리스에서는 6바이트 로그 RBA를 사용하여 로그에서의 데이터 위치를 식별합니다. IBM MQ 8.0에서 로그 RBA의 길이는 8바이트이므로 로그를 재설정해야 하는 시점까지의 기간을 늘릴 수 있습니다.

대형 로그 상대 바이트 주소 구현

IBM MQ for z/OS의 이전 릴리스에서는 6바이트 로그 RBA를 사용하여 로그에서의 데이터 위치를 식별합니다. IBM MQ 8.0에서 로그 RBA의 길이는 8바이트이므로 로그를 재설정해야 하는 시점까지의 기간을 늘릴 수 있습니다.

이 새 기능은 명시적으로 사용하게 설정해야 합니다. 8바이트 로그 RBA 사용을 계획하는 경우 주소 지정 가능 최대 로그 범위 증가 계획을 참조하십시오.

표시된 순서대로 이 지시사항을 수행하여 단일 IBM MQ for z/OS 큐 관리자에서 8바이트 로그 RBA를 사용 가능하게 하십시오.

1. OPMODE를 사용하여 IBM MQ 8.0 새 기능을 사용 가능하게 하십시오.

큐 공유 그룹에 있는 큐 관리자의 경우 전체 큐 공유 그룹을 가동 중단하지 않아도 됩니다. 각 큐 관리자를 차례대로 중지하고 OPMODE=(NEWFUNC,800)에 사용 가능하게 한 후 다시 시작할 수 있습니다.

큐 공유 그룹에 있는 모든 큐 관리자를 OPMODE(NEWFUNC,800)로 실행한 다음 모든 큐 관리자가 새로운 BSDS로 실행될 때까지 큐 공유 그룹에 있는 각 큐 관리자에 대해 다음 단계를 수행하십시오.

2. 현재 BSDS와 비슷한 속성을 사용하여 새 BSDS 데이터 세트를 할당하십시오. 샘플 CSQ4BSDS를 구성하고 관련이 없는 명령문을 삭제하거나 기존 JCL을 사용하되 BSDS 이름을 ++HLQ++.NEW.BSDS01과 같이 변경할 수 있습니다.

   참고사항:

   a. 새 BSDS의 속성을 확인하십시오. 변경할 수 있는 유일한 속성은 BSDS의 크기입니다.

   b. 새 BSDS에는 현재 BSDS보다 많은 데이터가 포함되어 있으므로 새 데이터 세트에 공간이 충분히 할당되었는지 확인해야 합니다. 새 BSDS를 정의할 때의 권장 값은 로깅 환경 계획 및 연관된 주제를 참조하십시오.

3. 큐 관리자를 문제 없이 종료하십시오.
4. BSDS 변환 유틸리티(CSQJUCNV)를 실행하여 기존 BSDS를 새 BSDS 데이터 세트로 변환하십시오. 일반적으로 이 작업은 실행하는 데 몇 분이 소요됩니다.

   기존 BSDS는 이 프로세스에서 변경되지 않으므로 변환에 실패하는 경우 큐 관리자 초기화를 위해 사용할 수 있습니다.

5. 다음 번에 큐 관리자를 다시 시작할 때 새 데이터 세트를 사용하도록 현재 BSDS가 이전 BSDS가 되고 새 BSDS가 현재 BSDS가 되게 이름을 바꾸십시오. 예를 들어, 다음과 같은 DFSMS 액세스 방법 서비스 ALTER 명령을 사용할 수 있습니다.

   ```
   ALTER '++HLQ++.BSDS01' NEWNAME('++HLQ++.OLD.BSDS01')
   ALTER '++HLQ++.NEW.BSDS01' NEWNAME('++HLQ++.BSDS01')
   ```

   VSAM 클러스터의 데이터 및 색인 부분의 이름을 모두 바꾸는 명령도 실행하십시오.

6. 큐 관리자를 재시작하십시오. 6바이트 로그 RBA를 사용할 때와 동일한 시간 내에 시작되어야 합니다.

   변환된 BSDS에 액세스하지 못해 큐 관리자를 재시작하지 못하는 경우 실패의 원인을 식별하고 문제점을 해결한 다음 조작을 재시도하십시오. 필요한 경우 IBM 지원 센터에 지원을 요청하십시오.

   필요한 경우 다음을 수행하여 변경을 취소할 수 있습니다.

   a. 현재 BSDS가 새 BSDS가 되도록 이름 바꾸기

   b. 이전 BSDS가 현재 BSDS가 되도록 이름 바꾸기

   c. 큐 관리자 재시작

   변환된 BSDS로 큐 관리자를 재시작한 경우 이전 BSDS를 사용하여 큐 관리자를 시작하지 마십시오.

7. 큐 관리자에 대해 구성된 로그 RBA의 끝을 표시하도록 큐 관리자 초기화 중에 CSQJ034I 메시지가 발행됩니다. 표시된 로그 RBA 범위의 끝이 FFFFFFFFFFFFFFFF인지 확인하십시오. 이는 8바이트 로그 RBA가 사용 중임을 나타냅니다.

참고: 새 IBM MQ 8.0 큐 관리자에서 8바이트 로그 RBA를 사용하려면 먼저 시작되기 전에 비어 있는 버전 1 형식 BSDS를 먼저 작성하고 BSDS 변환 유틸리티에 대한 입력으로 사용하여 버전 2 형식 BSDS를 생성해야 합니다. 이 프로세스 수행 방법에 대한 정보는 부트스트랩 및 로그 데이터 세트 작성을 참조하십시오.

관련 정보

주소 지정 가능 최대 로그 범위 증가 계획
대형 로그 상대 바이트 주소
BSDS 변환 유틸리티(CSQJUCNV)

부트스트랩 데이터 세트(BSDS) 관리

부트스트랩 데이터 세트(BSDS)는 로그 데이터 세트와 로그 레코드를 참조하는 데 사용됩니다. 이 주제를 사용하여 BSDS 조사, 변경, 복구 방법을 이해할 수 있습니다.

자세한 정보는 부트스트랩 데이터 세트를 참조하십시오.

이 주제는 부트스트랩 데이터 세트 관리와 관련된 태스크를 설명합니다. 여기에는 다음 절이 포함되어 있습니다.

BSDS 컨텐츠 찾기

로그 맵 인쇄 유틸리티(CSQJU004)를 사용하여 BSDS의 컨텐츠를 조사할 수 있습니다.

로그 맵 인쇄 유틸리티(CSQJU004)는 BSDS에 저장된 정보를 나열하는 배치 유틸리티입니다. 실행 지시사항은 로그 맵 인쇄 유틸리티를 참조하십시오.

BSDS에는 다음이 포함됩니다.

- 시간소인
- 활성 로그 데이터 세트 상태

BSDS의 시간소인

로그 맵 인쇄 유틸리티 출력은 다양한 시스템 이벤트의 날짜 및 시간을 기록하기 위해 사용하고 BSDS에 저장되는 시간소인을 보여줍니다.

보고서의 헤더 섹션에는 다음 시간소인이 포함됩니다.

SYSTEM TIMESTAMP
BSDS가 마지막으로 업데이트된 날짜 및 시간을 반영합니다. BSDS 시간소인을 업데이트할 수 있는 경우는 다음과 같습니다.

- 큐 관리자가 시작됩니다.
- 로그 쓰기 활동 중에 쓰기 임계값에 도달합니다. 지정한 출력 버퍼 수와 시스템 활동 비율에 따라, BSDS가 1초에 여러 번 업데이트될 수도 있고 몇 초, 몇 분 또는 몇 시간 동안도 업데이트되지 않을 수도 있습니다. 쓰기 임계값에 대한 자세한 내용은 CSQ6LOGP 사용에서 CSQ6LOGP 매크로의 WRTHRSH 매개변수를 참조하십시오.
- 오류로 인해 IBM MQ가 일반 이중 BSDS 모드에서 단일 BSDS 모드로 전환됩니다. 이는 BSDS 레코드 가져오기, 삽입, 가리키기, 업데이트 또는 삭제 요청이 실패할 때 발생할 수 있습니다. 이 오류가 발생하면 IBM MQ가 나머지 BSDS에서 시간소인을 업데이트하여 사용하지 않는 BSDS와의 시간소인 불일치를 강제 실행합니다.

UTILITY TIMESTAMP
로그 인벤토리 변경 유틸리티(CSQJU003)에 의해 BSDS의 컨텐츠가 대체된 날짜 및 시간입니다.

다음 시간소인은 보고서의 활성 및 아카이브 로그 데이터 세트 부분에 포함되어 있습니다.

활성 로그 날짜
BSDS에서 활성 로그 항목이 작성된 날짜(즉, CSQJU003 NEWLOG가 수행된 시점)

활성 로그 시간
BSDS에서 활성 로그 항목이 작성된 시간(즉, CSQJU003 NEWLOG가 수행된 시점)

아카이브 로그 날짜
BSDS에서 아카이브 로그 항목이 작성된 날짜(즉, CSQJU003 NEWLOG가 수행되었거나 아카이브 자체가 수행된 시점)

아카이브 로그 시간
BSDS에서 아카이브 로그 항목이 작성된 시간(즉, CSQJU003 NEWLOG가 수행되었거나 아카이브 자체가 수행된 시점)

활성 로그 데이터 세트 상태

BSDS는 활성 로그 데이터 세트의 상태를 다음 중 하나로 기록합니다.

NEW
데이터 세트가 정의되었지만 IBM MQ가 사용하지 않았거나 데이터 세트가 처음 사용되기 전 시점까지 로그가 잘렸습니다. 두 가지 경우 모두 데이터 세트 시작 및 종료 RBA 값이 0으로 재설정됩니다.

REUSABLE
데이터 세트가 정의되었지만 IBM MQ가 사용하지 않았거나 데이터 세트가 오프로드되었습니다. 로그 맵 인쇄 출력에서, 마지막 REUSABLE 데이터 세트의 시작 RBA 값은 마지막 아카이브 로그 데이터 세트의 시작 RBA 값과 같습니다.

NOT REUSABLE
데이터 세트에 오프로드되지 않은 레코드가 포함되어 있습니다.

STOPPED
레코드를 읽는 도중 오프로드 프로세서에 오류가 발생하여 활성 로그의 다른 사본에서 해당 레코드를 확보할 수 없습니다.

TRUNCATED
다음 중 하나입니다.

- I/O 오류가 발생하여 IBM MQ가 이 데이터 세트에 쓰기를 중지했습니다. 시작 RBA에서부터 잘린 활성 로그 데이터 세트의 올바른 마지막 레코드 세그먼트까지 계속 활성 로그 데이터 세트가 오프로드되었습니다. 올바른 마지막 레코드 세그먼트의 RBA가 활성 로그 데이터 세트의 종료 RBA보다 낮습니다. 로깅은 사용 가능한 다음 활성 로그 데이터 세트로 전환되며 계속 중단되지 않습니다.

또는

- ARCHIVE LOG 함수가 호출되어 활성 로그가 잘렸습니다.

상태는 로그 맵 인쇄 유틸리티의 출력에 나타납니다.

BSDS 변경

로깅 이벤트 레코드로 BSDS를 계속 업데이트하기 위해 특별한 단계를 수행하지 않아도 됩니다. IBM MQ가 해당 작업을 자동으로 수행하기 때문입니다.

그러나 다음을 수행하는 경우 BSDS를 변경해야 할 수 있습니다.

- 추가 활성 로그 데이터 세트를 추가합니다.
- 예를 들어, 보다 큰 활성 로그 할당을 제공할 때 새로 할당된 데이터 세트에 활성 로그 데이터 세트를 복사합니다.
- 로그 데이터 세트를 다른 디바이스로 이동합니다.
- 손상된 BSDS를 복구합니다.
- 오래된 아카이브 로그 데이터 세트를 제거합니다.

로그 인벤토리 변경 유틸리티(CSQJU003)를 실행하여 BSDS를 변경할 수 있습니다. 이 유틸리티는 큐 관리자가 비활성 상태인 경우에만 실행하십시오. 그렇지 않으면 불일치 결과를 얻게 됩니다. 유틸리티 조치는 SYSIN 데이터 세트의 명령문으로 제어됩니다. 이 절에서는 몇 가지 예를 보여줍니다. 전체 지시사항은 로그 인벤토리 변경 유틸리티를 참조하십시오.

IBM MQ는 큐 관리자 시작 시 활성 로그 데이터 세트를 독점(DISP=OLD)으로 할당하므로 큐 관리자가 비활성인 경우에만 활성 로그 데이터 세트를 복사할 수 있습니다.

활성 로그에 대한 변경사항

이 주제를 통해 BSDS를 사용한 활성 로그 변경 방법을 이해할 수 있습니다.

변경 로그 유틸리티를 사용하여 BSDS에서 활성 로그에 대한 항목을 추가, 삭제, 기록할 수 있습니다. 여기에 표시된 것은 예시용이므로 표시된 데이터 세트 이름을 사용하려는 이름으로 바꾸십시오. 유틸리티에 대한 자세한 내용은 로그 인벤토리 변경 유틸리티를 참조하십시오.

자세한 정보는 다음 절을 참조하십시오.

- BSDS에 레코드 항목 추가
- BSDS에서 활성 로그 데이터 세트에 대한 정보 삭제
- BSDS에서 로그 데이터 세트에 대한 정보 기록

- 활성 로그 크기 늘리기
- CSQJUFMT 사용

BSDS에 레코드 항목 추가

활성 로그가 "중지됨" 플래그가 지정된 경우에는 로깅 목적으로 재사용되지 않습니다. 그러나 읽기용으로 계속 사용됩니다. 액세스 방법 서비스를 사용하여 새 활성 로그 데이터 세트를 정의한 다음 로그 인벤토리 변경 유틸리티를 사용하여 BSDS에서 새 데이터 세트를 등록하십시오. 예를 들어, 다음을 사용하십시오.

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

이전 활성 로그 데이터 세트의 컨텐츠를 새 활성 로그 데이터 세트에 복사하는 경우 또한 RBA 범위와 함께 시작 및 종료 시간소인을 NEWLOG 기능에 제공할 수 있습니다.

BSDS에서 활성 로그 데이터 세트에 대한 정보 삭제

BSDS에서 활성 로그 데이터 세트에 대한 정보를 삭제하기 위해 다음을 사용할 수 있습니다.

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

BSDS에서 로그 데이터 세트에 대한 정보 기록

BSDS에서 기존 활성 로그 데이터 세트에 대한 정보를 기록하려면 다음을 사용하십시오.

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

다음과 같은 이유로 BSDS에서 이 정보 유형을 포함하는 레코드를 삽입해야 할 수 있습니다.

- 데이터 세트에 대한 항목이 삭제되었지만 다시 필요합니다.
- 특정 활성 로그 데이터 세트의 컨텐츠를 다른 데이터 세트에 복사합니다.
- BSDS를 백업 사본에서 복구합니다.

활성 로그 크기 늘리기

이 프로세스를 수행하는 방법은 두 가지가 있습니다.

1. 큐 관리자가 활성 상태인 경우:
   a. JCL을 사용하여 새 대형 로그 데이터 세트를 정의하십시오.
   b. MQSC DEFINE LOG 명령을 사용하여 활성 큐 관리자에 새 로그 데이터 세트를 추가하십시오.
   c. MQSC ARCHIVE LOG 명령을 사용하여 현재 활성 로그가 새 대형 로그가 되도록 이동하십시오.
   d. 소형 활성 로그 데이터 세트 아카이브가 완료될 때까지 기다리십시오.
   e. CSQJU003 유틸리티로 이전 소형 활성 로그를 제거하여 큐 관리자를 종료하십시오.
   f. 큐 관리자를 재시작하십시오.
2. 큐 관리자가 비활성 상태인 경우:
   a. 큐 관리자를 중지합니다. 이 단계가 필요한 이유는 IBM MQ가 모든 활성 로그 데이터 세트를 활성 상태에서 독점적으로 사용하도록 할당하기 때문입니다.
   b. 액세스 방법 서비스 ALTER를 NEWNAME 옵션과 함께 사용하여 활성 로그 데이터 세트의 이름을 바꾸십시오.
   c. 액세스 방법 서비스 DEFINE을 사용하여 대형 활성 로그 데이터 세트를 정의하십시오.

이전 데이터 세트 이름을 재사용함으로써 BSDS에서 새 이름을 설정하기 위해 로그 인벤토리 변경 유틸리티를 실행하지 않아도 됩니다. 이전 데이터 세트 이름과 올바른 RBA 범위는 이미 BSDS에 있습니다.

d. 액세스 방법 서비스 REPRO를 사용하여 이전(이름을 바꾼) 데이터 세트를 해당 새 데이터 세트에 복사하십시오.

참고: 이 단계는 시간이 오래 걸릴 수 있으므로 이 기간 동안 엔터프라이즈가 작동하지 않을 수 있습니다.

e. 큐 관리자를 시작하십시오.

모든 로그 데이터 세트의 크기가 같으면 시스템이 보다 일관적이고 효율적으로 작동합니다. 로그 데이터 세트의 크기가 같지 않으면 시스템 로그를 추적하기가 더 어려우므로 공간이 낭비될 수 있습니다.

CSQJUFMT 사용

활성 로그 크기를 늘릴 때 CSQJUFMT 형식을 실행하지 마십시오.

큐 관리자가 새 활성 로그에 처음 쓰기를 시도할 때 성능 이점을 제공하기 위해 CSQJUFMT를 실행하면 다음 메시지를 수신합니다.

```
IEC070I 203-204,XS95GTLX,REPRO02,OUTPUT,B857,SPMG02, 358
IEC070I MG.W.MG4E.LOGCOPY1.DS02,MG.W.MG4E.LOGCOPY1.DS02.DATA,
IDC3302I ACTION ERROR ON MG.W.MG4E.LOGCOPY1.DS02
IDC3351I ** VSAM I/O RETURN CODE IS 28 - RPLFDBWD = X'2908001C'
IDC31467I MAXIMUM ERROR LIMIT REACHED.

IDC0005I NUMBER OF RECORDS PROCESSED WAS 0
```

또한 액세스 방법 서비스 REPRO를 사용하는 경우 비어 있는 로그를 새로 정의합니다.

REPRO를 사용하여 이전(이름을 바꾼) 데이터 세트를 해당 새 데이터 세트에 복사하는 경우 기본값은 NOREPLACE입니다.

이는 REPRO가 이미 지정된 데이터 세트에 있는 레코드를 바꾸지 않음을 의미합니다. 데이터 세트에서 형식화가 완료되면 RBA 값이 재설정됩니다. 최종 결과는 형식화 이후 비어 있지 않은 데이터 세트입니다.

아카이브 로그에 대한 변경사항

이 주제를 통해 아카이브 로그 변경 방법을 이해할 수 있습니다.

BSDS에서 아카이브 로그에 대한 입력 항목을 추가, 삭제하고 비밀번호를 변경할 수 있습니다. 여기에 표시된 것은 예시용이므로 표시된 데이터 세트 이름을 사용하려는 이름으로 바꾸십시오. 유틸리티에 대한 자세한 내용은 로그 인벤토리 변경 유틸리티를 참조하십시오.

- 아카이브 로그 추가
- 아카이브 로그 삭제
- 아카이브 로그의 비밀번호 변경

아카이브 로그 추가

오브젝트 복구가 기존 아카이브 로그 데이터 세트 읽기에 따라 결정되는 경우, 해당 데이터 세트에 대한 정보가 BSDS에 포함되어야 IBM MQ가 찾을 수 있습니다. BSDS에서 기존 아카이브 로그 데이터 세트에 대한 정보를 등록하려면 다음을 사용하십시오.

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

아카이브 로그 삭제

하나 이상의 볼륨에서 전체 아카이브 로그 데이터 세트를 삭제하려면 다음을 사용하십시오.

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

아카이브 로그의 비밀번호 변경

기존 아카이브 로그 데이터 세트의 비밀번호를 변경하려면 BSDS에서도 정보를 변경해야 합니다.

1. 로그 맵 인쇄 유틸리티를 사용하여 BSDS를 나열하십시오.
2. CSQJU003 유틸리티의 DELETE 함수를 사용하여 변경된 비밀번호로 아카이브 로그 데이터 세트에 대한 입력 항목을 삭제하십시오(로그 인벤토리 변경 유틸리티 주제 참조).
3. 새 아카이브 로그 데이터 세트에 대한 데이터 세트 이름을 지정하십시오. CSQJU003 유틸리티(로그 인벤토리 변경 유틸리티 주제 참조)의 NEWLOG 함수를 사용하여 새 비밀번호, 시작 및 종료 RBA, 볼륨 일련 번호(로그 맵 인쇄 유틸리티 출력에서 찾을 수 있음, 로그 맵 인쇄 유틸리티 참조)를 지정하십시오.

새 아카이브 로그 데이터 세트의 비밀번호를 변경하려면 다음을 사용하십시오.

```
ARCHIVE PASSWORD= password
```

새 아카이브 로그 데이터 세트에 비밀번호 배치를 중지하려면 다음을 사용하십시오.

```
ARCHIVE NOPASSWD
```

참고: ARCHIVE 유틸리티 함수는 외부 보안 관리자가 없는 경우에만 사용하십시오.

로그 및 *BSDS*의 상위 레벨 규정자*(HLQ)* 변경
이 주제를 통해 상위 레벨 규정자(HLQ)를 변경하는 데 필요한 프로시저를 이해할 수 있습니다.

시작하기 전에

로그 또는 데이터 세트를 새 데이터 세트에 복사하기 전에 큐 관리자를 정상적으로 종료해야 합니다. 이를 통해 데이터의 일관성을 유지하고 재시작 과정에서 복구가 필요하지 않게 됩니다.

이 태스크 정보

이 태스크는 로그 및 BSDS의 HLQ 변경 방법에 대한 정보를 제공합니다. 이를 수행하려면 다음 단계를 수행하십시오.

프로시저

1. 로그 인쇄 유틸리티 CSQJU004를 실행하여 로그 데이터 세트 정보를 기록하십시오. 이 정보는 나중에 필요합니다.
2. 다음 중 한 가지 방법을 수행할 수 있습니다.
   a) DSS 백업을 실행하고 이름을 바꿀 로그 및 BSDS 데이터 세트에서 이름 바꾸기로 복원합니다. 또는
   b) AMS DEFINE 및 REPRO를 사용하여 HLQ 데이터 세트를 작성하고 이전 데이터 세트에서 데이터를 복사합니다.
3. 새 데이터 세트를 가리키도록 MSTR 및 CHIN 프로시저를 수정하십시오.
4. CSQJU003을 사용하여 BSDS의 새 사본에서 이전 로그 정보를 삭제하십시오.
5. CSQJU003의 NEWLOG 함수를 사용하여 새 BSDS에 새 로그 데이터 세트를 정의하십시오.
   HLQ와 별도로 각 로그에 대한 모든 정보를 동일하게 유지합니다.
6. 새 BSDS에는 이전 BSDS에서 이전 로그에 대해 기록된 것과 동일한 정보가 반영되어야 합니다.
   HLQ만 변경 가능해야 합니다.

다음에 수행할 작업

이전 BSDS와 새 BSDS에 대한 CSQJU004 출력을 비교하여 큐 관리자를 시작하기 전과 정확하게 일치하는지 (HLQ 제외) 확인하십시오.

참고: 이 조작을 수행할 때는 주의를 기울여야 합니다. 잘못된 조치는 복구 불가능한 상황을 초래할 수 있습니다. PRINT LOG MAP UTILITY 출력을 보고 복구 또는 재시작에 필요한 모든 정보가 포함되었는지 확인하십시오.

BSDS 복구

IBM MQ가 이중 BSDS 모드로 작동할 때 하나의 BSDS가 손상되는 경우 IBM MQ를 단일 BSDS 모드로 강제 실행하면 IBM MQ가 문제점 없이 계속 작동합니다(다음에 재시작할 때까지).

환경을 이중 BSDS 모드로 되돌리려면 다음을 수행하십시오.

1. 액세스 방법 서비스를 사용하여 손상된 BSDS의 이름을 바꾸거나 삭제하고 손상된 BSDS와 같은 이름으로 새 BSDS를 정의하십시오. 제어 명령문의 예는 thlqual.SCSQPROC의 작업 CSQ4BREC에서 찾을 수 있습니다.
2. IBM MQ 명령 RECOVER BSDS를 실행하여 새로 할당된 데이터 세트에서 올바른 BSDS의 사본을 작성하고 이중 BSDS 모드를 복원하십시오.

IBM MQ가 단일 BSDS 모드에서 작동하고 BSDS가 손상되거나, IBM MQ가 이중 BSDS 모드에서 작동하고 BSDS가 둘 다 손상된 경우에는 큐 관리자가 중지되고 BSDS 데이터 세트가 복구될 때까지 재시작되지 않습니다. 이 경우 다음과 같은 상황이 발생합니다.

1. 최신 아카이브 로그 데이터 세트와 연관된 BSDS를 찾으십시오. 최신 아카이브 로그의 데이터 세트 이름은 작업 로그에서 CSQJ003I 메시지(오프로드 처리가 성공적으로 완료되었음을 나타냄)의 마지막 발생에 나타납니다. 이 프로시저의 나머지 부분을 준비하기 위해 해당 메시지에 표시된 모든 성공적인 아카이브의 로그를 보존하는 것이 좋습니다.
   - 아카이브 로그가 DASD에 있으면 사용 가능한 DASD에서 BSDS가 할당됩니다. BSDS 이름은 해당 아카이브 로그 데이터 세트 이름과 유사합니다. 다음 예와 같이 다음 마지막 규정자의 첫 번째 문자만 A에서 B로 변경하십시오.

     아카이브 로그 이름
     CSQ.ARCHLOG1. ***A*** 0000001

     BSDS 사본 이름
     CSQ.ARCHLOG1. ***B*** 0000001
   - 아카이브 로그가 테이프에 있으면 BSDS가 첫 번째 아카이브 로그 볼륨의 첫 번째 데이터 세트입니다. 이후 볼륨에서는 BSDS가 반복되지 않습니다.
2. 최신 아카이브 로그 데이터 세트에 BSDS의 사본이 없으면(예를 들어, 오프로드 시 오류가 발생한 경우) 이전 오프로드 처리에서 BSDS의 이전 사본을 찾으십시오.
3. 액세스 방법 서비스 ALTER 명령과 NEWNAME 옵션을 함께 사용하여 손상된 BSDS의 이름을 바꾸십시오. 손상된 BSDS를 삭제하려면 액세스 방법 서비스 DELETE 명령을 사용하십시오. 손상된 BSDS마다 액세스 방법 서비스를 사용하여 새 BSDS를 대체 데이터 세트로 정의하십시오. thlqual.SCSQPROC의 CSQ4BREC 작업에는 새 BSDS를 정의하는 액세스 방법 서비스 제어 명령문이 포함되어 있습니다.
4. 액세스 방법 서비스 REPRO 명령을 사용하여 아카이브 로그의 BSDS를 단계 285 페이지의 『3』에서 정의한 대체 BSDS 중 하나에 복사하십시오. 두 번째 대체 BSDS에는 데이터를 복사하지 마십시오. 이 작업은 단계 286 페이지의 『5』에서 수행합니다.

   a. 대체 BSDS의 컨텐츠를 인쇄하십시오.

      로그 맵 인쇄 유틸리티(CSQJU004)를 사용하여 대체 BSDS의 컨텐츠를 인쇄하십시오. 이렇게 하면 복구 작업을 계속 수행하기 전에 대체 BSDS의 컨텐츠를 검토할 수 있습니다.

   b. 대체 BSDS에서 아카이브 로그 데이터 세트 인벤토리를 업데이트하십시오.

      로그 맵 인쇄 유틸리티의 출력을 조사하여 대체 BSDS에 BSDS가 복사된 아카이브 로그의 레코드가 포함되지 않았는지 확인하십시오. 대체 BSDS가 이전 사본인 경우에는 해당 인벤토리에 보다 최근에 작성된 모든 아카이브 로그 데이터 세트가 포함되지 않았을 수 있습니다. 아카이브 로그 데이터 세트의 BSDS 인벤토리는 현재 서브시스템 인벤토리를 반영하여 업데이트되어야 합니다.

      로그 인벤토리 변경 유틸리티(CSQJU003) NEWLOG 명령문으로 BSDS가 복사된 아카이브 로그의 레코드를 추가하여 대체 BSDS를 업데이트하십시오. 아카이브 로그 데이터 세트가 비밀번호로 보호되는 경우에는 NEWLOG 함수의 PASSWORD 옵션을 사용하십시오. 또한 아카이브 로그 데이터 세트가 카탈로그화된

경우에는 NEWLOG 함수의 CATALOG 옵션이 CATALOG=YES로 올바르게 설정되었는지 확인하십시오. BSDS 사본보다 나중에 작성된 추가 아카이브 로그 데이터 세트를 추가하려면 NEWLOG 명령문을 사용하십시오.

c. 대체 BSDS에서 비밀번호를 업데이트하십시오.

BSDS에는 아카이브 로그 데이터 세트와 활성 로그 데이터 세트에 대한 비밀번호가 포함됩니다. 대체 BSDS의 비밀번호가 설치 시 사용된 현재 비밀번호를 반영하도록 하려면 로그 인벤토리 변경 ARCHIVE 유틸리티 함수에 PASSWORD 옵션을 함께 사용하십시오.

d. 대체 BSDS에서 활성 로그 데이터 세트 인벤토리를 업데이트하십시오.

특별한 경우, BSDS가 복사된 이후 설치에서 활성 로그 데이터 세트가 추가, 삭제되거나 이름이 바뀌었을 수 있습니다. 이러한 경우에는 설치에 현재 사용 중인 활성 로그 데이터 세트의 하나 이상의 실제 이름이 대체 BSDS에 반영되지 않습니다.

대체 BSDS 로그 인벤토리에서 활성 로그 데이터 세트를 삭제해야 하는 경우에는 로그 인벤토리 변경 유틸리티 DELETE 함수를 사용하십시오.

대체 BSDS 로그 인벤토리에 활성 로그 데이터 세트를 추가해야 하는 경우에는 로그 인벤토리 변경 유틸리티 NEWLOG 함수를 사용하십시오. RBA 범위가 NEWLOG 함수에 올바르게 지정되었는지 확인하십시오. 활성 로그 데이터 세트가 비밀번호로 보호되는 경우에는 PASSWORD 옵션을 사용하십시오.

대체 BSDS 로그 인벤토리에서 활성 로그 데이터 세트를 이름을 바꿔야 하는 경우에는 로그 인벤토리 변경 유틸리티 DELETE 함수와 NEWLOG 함수를 차례로 사용하십시오. RBA 범위가 NEWLOG 함수에 올바르게 지정되었는지 확인하십시오. 활성 로그 데이터 세트가 비밀번호로 보호되는 경우에는 PASSWORD 옵션을 사용하십시오.

e. 대체 BSDS에서 활성 로그 RBA 범위를 업데이트하십시오.

나중에 큐 관리자가 재시작되면 BSDS에 나열된 활성 로그 데이터 세트의 RBA를 실제 활성 로그 데이터 세트에 있는 RBA와 비교합니다. RBA가 동의하지 않으면 큐 관리자가 재시작되지 않습니다. BSDS의 이전 사본을 사용하는 경우에는 문제점이 커집니다. 이 문제점을 해결하려면 로그 인벤토리 변경 유틸리티(CSQJU003)를 사용하여 실제 활성 로그 데이터 세트의 RBA를 사용하여 BSDS에 있는 RBA를 조정하십시오. 방법은 다음과 같습니다.

- 로그 레코드 인쇄 유틸리티(CSQ1LOGP)를 사용하여 활성 로그 데이터 세트의 요약 보고서를 인쇄합니다. 이 보고서는 시작 및 종료 RBA를 보여줍니다.
- 실제 RBA 범위와 방금 인쇄한 RBA 범위를 비교합니다(모든 활성 로그 데이터 세트의 RBA를 알 수 있는 경우).

  RBA 범위가 모든 활성 로그 데이터 세트에 동일한 경우에는 추가 작업 없이 다음 복구 단계로 진행할 수 있습니다.

  RBA 범위가 동일하지 않은 경우에는 실제 값을 반영하여 BSDS에서 값을 조정하십시오. RBA 범위를 조정해야 하는 각 활성 로그 데이터 세트마다 로그 인벤토리 변경 유틸리티 DELETE 함수를 사용하여 대체 BSDS의 인벤토리에서 활성 로그 데이터 세트를 삭제하십시오. 그런 다음 NEWLOG 함수를 사용하여 활성 로그 데이터 세트를 BSDS로 재정의하십시오. 활성 로그 데이터 세트가 비밀번호로 보호되는 경우에는 NEWLOG 함수의 PASSWORD 옵션을 사용하십시오.

f. 각 활성 로그 사본마다 활성 로그 데이터 세트가 두 개만 지정된 경우에는 IBM MQ가 큐 관리자 재시작 시 어려움을 겪을 수 있습니다. 이 문제점은 활성 로그 데이터 세트 중 하나가 가득 차고 오프로드되지 않았으며 두 번째 활성 로그 데이터 세트가 거의 채워진 경우 발생할 수 있습니다. 이러한 경우에는 활성 로그의 각 사본에 대한 새 활성 로그 데이터 세트를 추가하고 대체 BSDS 로그 인벤토리에서 새 활성 로그 데이터 세트를 각각 정의하십시오.

액세스 방법 서비스 DEFINE 명령을 사용하여 활성 로그의 각 사본에 대한 새 활성 로그 데이터 세트를 정의하고 로그 인벤토리 변경 유틸리티 NEWLOG 함수를 사용하여 대체 BSDS에서 새 활성 로그 데이터 세트를 정의하십시오. NEWLOG 명령문에는 RBA 범위를 지정하지 않아도 됩니다. 그러나 활성 로그 데이터 세트가 비밀번호로 보호되는 경우에는 NEWLOG 함수의 PASSWORD 옵션을 사용하십시오. 이 태스크 수행을 위한 제어 명령문 예는 thlqual.SCSQPROC의 CSQ4LREC 작업에 나와 있습니다.

5. 업데이트된 BSDS를 두 번째 새 BSDS 데이터 세트에 복사하십시오. 이제 BSDS는 같아집니다.

   로그 맵 인쇄 유틸리티(CSQJU004)를 사용하여 이 시점에서 두 번째 대체 BSDS의 컨텐츠를 인쇄하십시오.

6. 현재 활성 로그 데이터 세트가 손실된 경우 수행 조치에 대한 정보는 활성 로그 문제점을 참조하십시오.

7. 새로 구성된 BSDS를 사용하여 큐 관리자를 재시작하십시오. IBM MQ는 현재 RBA와 아카이브되어야 하는 활성 로그를 판별합니다.

페이지 세트 관리

이 주제를 통해 큐 관리자와 연관된 페이지 세트 관리 방법을 이해할 수 있습니다.

이 주제는 큐 관리자와 연관된 페이지 세트를 추가, 복사하고 일반적으로 관리하는 방법을 설명합니다. 여기에는 다음 절이 포함되어 있습니다.

페이지 세트, 스토리지 클래스, 버퍼 및 버퍼 풀에 대한 설명과 적용되는 일부 성능 고려사항은 페이지 세트를 참조하십시오.

페이지 세트의 상위 레벨 규정자(HLQ) 변경 방법

이 태스크는 페이지 세트의 HLQ를 변경하는 방법에 대한 정보를 제공합니다. 이 태스크를 수행하려면 다음을 수행하십시오.

1. 새 HLQ 페이지 세트를 정의하십시오.
2. 크기 할당이 이전 페이지 세트와 같은 경우 REPRO를 사용하여 기존 페이지 세트를 비어 있는 새 HLQ 페이지 세트에 복사하십시오. 페이지 세트의 크기를 늘리는 경우 CSQUTIL의 FORMAT 함수를 사용하여 대상 페이지 세트를 형식화하십시오. 자세한 정보는 페이지 세트 형식화(FORMAT)를 참조하십시오.
3. CSQUTIL의 COPYPAGE 함수를 사용하여 소스 페이지 세트에서 대상 페이지 세트로 모든 메시지를 복사하십시오. 자세한 정보는 페이지 세트 확장(COPYPAGE)을 참조하십시오.
4. 새 HLQ 페이지 세트를 가리키도록 큐 관리자 프로시저에서 CSQP00xx DD문을 변경하십시오.

큐 관리자를 재시작하고 페이지 세트의 변경사항을 확인하십시오.

페이지 세트를 큐 관리자에 추가하는 방법

이 설명에서는 이미 실행 중인 큐 관리자가 있다고 가정합니다. 예를 들어 큐 관리자가 새 큐를 사용하여 새 애플리케이션을 처리해야 하는 경우 페이지 세트를 추가해야 합니다.

새 페이지 세트를 추가하려면 다음 프로시저를 사용하십시오.

1. 새 페이지 세트를 정의하고 형식화하십시오. thlqual.SCSQPROC(CSQ4PAGE)의 샘플 JCL을 기초로 사용할 수 있습니다. 자세한 정보는 페이지 세트 형식화(FORMAT)를 참조하십시오.

   의도한 경우를 제외하고는 사용 중인 페이지 세트를 형식화하지 않도록 주의하십시오. 형식화하려는 경우 FORMAT 유틸리티 함수의 FORCE 옵션을 사용하십시오.
2. DSN 옵션과 함께 DEFINE PSID 명령을 사용하여 페이지 세트를 버퍼 풀과 연관시키십시오.
3. DEFINE STGCLASS 명령을 실행하여 페이지 세트에 적절한 스토리지 클래스 정의를 추가하십시오.

4. 또는 큐 관리자 구성 방법을 문서화하려면 다음을 수행하십시오.
   a. 큐 관리자의 시작된 태스크 프로시저에 새 페이지 세트를 추가하십시오.
   b. 새 페이지 세트의 정의를 CSQINP1 초기화 데이터 세트에 추가하십시오.
   c. 새 스토리지 클래스의 정의를 CSQ4INYR 초기화 데이터 세트 구성원에 추가하십시오.

DEFINE PSID 및 DEFINE STGCLASS 명령에 대한 자세한 내용은 DEFINE PSID 및 DEFINE STGCLASS를 참조하십시오.

페이지 세트 중 하나가 가득 차면 수행할 사항

IBM MQ 명령 DISPLAY USAGE를 사용하여 페이지 세트 이용에 대한 정보를 얻을 수 있습니다. 예를 들어,

```
DISPLAY USAGE PSID(03)
```

명령은 페이지 세트 03의 현재 상태를 표시합니다. 이 페이지 세트에 있는 사용 가능한 페이지 수를 표시합니다.

페이지 세트의 보조 범위를 정의한 경우 범위가 가득 찰 때마다 동적으로 확장됩니다. 결국 모든 보조 범위가 사용되거나 추가 디스크 공간이 사용 가능하지 않게 됩니다. 이 경우 애플리케이션에 MQRC_STORAGE_MEDIUM_FULL 리턴 코드가 수신됩니다.

애플리케이션이 MQI 호출로부터 MQRC_STORAGE_MEDIUM_FULL 리턴 코드를 수신하면 페이지 세트에 남아 있는 공간이 충분하지 않음을 나타냅니다. 문제점이 지속되거나 반복될 가능성이 큰 경우 해결하도록 조치를 수행해야 합니다.

다음과 같은 여러 방법으로 이 문제점에 접근할 수 있습니다.

- 한 페이지 세트에서 다른 페이지 세트로 큐를 이동하여 페이지 세트 간 로드의 밸런스를 맞추십시오.
- 페이지 세트를 확장하십시오. 지시사항은 290 페이지의 『페이지 세트의 크기를 늘리는 방법』의 내용을 참조하십시오.
- 4GB를 넘어 최대 64GB로 크기가 확장될 수 있도록 페이지 세트를 다시 정의하십시오. 지시사항은 4GB보다 크게 페이지 세트 정의를 참조하십시오.

페이지 세트의 로드 밸런싱 방법

페이지 세트의 로드 밸런싱은 하나 이상의 큐와 연관된 메시지를 한 페이지 세트에서 덜 사용되는 다른 페이지 세트로 이동하는 작업입니다. 페이지 세트 확장이 실용적이지 않은 경우 이 기술을 사용하십시오.

페이지 세트를 사용하는 큐를 식별하려면 해당 IBM MQ 명령을 사용하십시오. 예를 들어 페이지 세트 02에 맵핑된 큐를 찾으려면, 먼저 다음 명령을 사용하여 페이지 세트 02에 맵핑된 스토리지 클래스를 찾으십시오.

```
DISPLAY STGCLASS(*) PSID(02)
```

그런 후 다음 명령을 사용하여 각 큐가 사용하는 스토리지 클래스를 찾으십시오.

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

비공유 큐 이동

큐와 해당 메시지를 특정 페이지 세트에서 다른 페이지 세트로 이동하려면 MQSC MOVE QLOCAL 명령을 사용하십시오(MOVE QLOCAL의 설명 참조). 새 페이지 세트로 이동할 하나 이상의 큐를 식별한 경우 각 큐에 대해 이 프로시저를 수행하십시오.

1. 이동할 큐를 다른 애플리케이션에서 사용 중이지 않으며(즉, DISPLAY QSTATUS 명령의 IPPROCS 및 OPPROCS 값이 0이 아님) 커미트된 메시지가 없는지(DISPLAY QSTATUS 명령의 UNCOM 값이 NO임) 확인하십시오.

   참고: 이 상태가 계속되도록 하는 유일한 방법은 큐의 보안 권한을 임시로 변경하는 것입니다. 자세한 정보는 큐 보안을 위한 프로파일을 참조하십시오.

   이 작업을 수행할 수 없는 경우 PUT(DISABLED) 설정과 같은 예방조치 단계를 수행해도 애플리케이션이 큐를 사용하기 시작하면 이 프로시저의 나중 단계가 실패할 수 있습니다. 그러나 이 프로시저에서 메시지는 손실되지 않습니다.
2. 애플리케이션이 MQPUT 을 사용하지 않도록 큐 정의를 변경하여 큐에 메시지를 넣지 않도록 합니다. 큐 정의를 PUT(DISABLED)로 변경하십시오.
3. 다음 명령을 사용하여 이동 중인 큐와 속성이 동일한 임시 큐를 정의하십시오.

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
```

   참고: 이전 실행에서 이 임시 큐를 이미 작성한 경우 정의하기 전에 이 큐를 삭제하십시오.
4. 다음 명령을 사용하여 임시 큐로 메시지를 이동하십시오.

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. 다음 명령을 사용하여 이동 중인 큐를 삭제하십시오.

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. 예를 들어 다음과 같이 필요한 페이지 세트에 맵핑되는 새 스토리지 클래스를 정의하십시오.

```
DEFINE STGCLASS(NEW) PSID(nn)
```

   다음 큐 관리자 재시작에 사용 가능하도록 CSQINP2 데이터 세트에 새 스토리지 클래스 정의를 추가하십시오.
7. 다음과 같이 스토리지 클래스 속성을 변경하여 이동 중인 큐를 다시 정의하십시오.

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
```

   큐를 다시 정의할 때 이 큐는 289 페이지의 『3』 단계에서 작성된 임시 큐를 기반으로 합니다.
8. 다음 명령을 사용하여 메시지를 새 큐로 다시 이동하십시오.

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

9. 289 페이지의 『3』 단계에서 작성한 큐는 더 이상 필요하지 않습니다. 다음 명령을 사용하여 해당 큐를 삭제하십시오.

```
DELETE QL(TEMP_QUEUE)
```

10. 이동 중인 큐가 CSQINP2 데이터 세트에 정의된 경우 CSQINP2 데이터 세트에서 적절한 DEFINE QLOCAL 명령의 STGCLASS 속성을 변경하십시오. 기존 큐 정의를 바꾸도록 REPLACE 키워드를 추가하십시오.

290 페이지의 그림 45은 로드 밸런싱 작업에서의 추출을 표시합니다.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE STGCLASS(NEW) PSID(2)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

그림 *45.* 페이지 세트의 로드 밸런싱 작업에서 추출

페이지 세트의 크기를 늘리는 방법

처음에 4GB보다 큰 페이지를 할당할 수 있습니다. 4GB보다 크게 페이지 세트 정의를 참조하십시오.

페이지 세트가 가득 참에 따라 자동으로 확장되도록 EXPAND(SYSTEM) 또는 EXPAND(USER)를 지정하여 페이지 세트를 정의할 수 있습니다. EXPAND(NONE)을 사용하여 페이지 세트를 정의한 경우 다음 두 방법 중 하나로 확장할 수 있습니다.

- 자동 확장될 수 있도록 정의를 대체하십시오. 자동 확장될 수 있도록 페이지 세트 대체를 참조하십시오.
- 보다 큰 새 페이지 세트를 작성하고 이전 페이지 세트에서 새 페이지 세트로 메시지를 복사하십시오. 보다 큰 새 페이지 세트로 메시지 이동을 참조하십시오.

4GB보다 크게 페이지 세트 정의

데이터 세트가 VSAM에 대한 '확장된 주소 지정 가능성'으로 정의된 경우 IBM MQ는 최대 64GB 크기의 페이지 세트를 사용할 수 있습니다. 확장된 주소 지정 가능성은 SMS 데이터 클래스에서 제공하는 속성입니다. 다음 샘플 JCL에 표시된 예에서는 관리 클래스 'EXTENDED'가 '확장된 주소 지정 가능성'을 사용하여 SMS에 정의됩니다. 기존 페이지 세트가 현재 확장된 주소 지정 가능성이 있도록 정의되지 않은 경우 다음 메소드를 사용하여 확장된 주소 지정 가능성 형식 데이터 세트로 마이그레이션하십시오.

1. 큐 관리자를 중지합니다.
2. AMS를 사용하여 기존 페이지 세트의 이름을 바꾸십시오.
3. 기존 페이지 세트와 크기는 같지만 DATACLAS(EXTENDED)가 있는 대상 페이지 세트를 정의하십시오.

   참고: 확장된 데이터 세트는 SMS 관리여야 합니다. 다음은 VSAM 데이터 세트의 확장된 형식을 요청하는 메커니즘입니다.

   - DSNTYPE 값이 EXT이고 하위 매개변수가 R 또는 P인 데이터 클래스를 사용하여 필수 또는 선호를 표시합니다.
   - DD 명령문에서 DSNTYPE=EXTREQ(확장된 형식이 필수임) 또는 DSNTYPE=EXTPREF(확장된 형식이 선호임)를 코딩하십시오.
   - 기존 확장된 형식 데이터 세트를 참조하기 위해 DD 명령문에 LIKE= 매개변수를 코딩하십시오.

   자세한 정보는 확장된 형식 데이터 세트에 대한 제한사항을 참조하십시오.

4. CSQUTIL의 COPYPAGE 함수를 사용하여 소스 페이지 세트에서 대상 페이지 세트로 모든 메시지를 복사하십시오. 자세한 정보는 페이지 세트 확장(COPYPAGE)을 참조하십시오.
5. 큐 관리자를 재시작하십시오.
6. 시스템 확장을 사용하도록 페이지 세트를 대체하여 현재 할당 크기보다 계속 증가될 수 있게 하십시오.

다음 JCL은 AMS 명령 예를 표시합니다.

```
//S1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN   DD  *
ALTER 'VICY.CSQ1.PAGE01' -
NEWNAME('VICY.CSQ1.PAGE01.OLD')
ALTER 'VICY.CSQ1.PAGE01.DATA' -
NEWNAME('VICY.CSQ1.PAGE01.DATA.OLD')
DEFINE CLUSTER (NAME('VICY.CSQ1.PAGE01') -
MODEL('VICY.CSQ1.PAGE01.OLD') -
DATACLAS(EXTENDED))
/*
```

자동 확장될 수 있도록 페이지 세트 대체

ALTER PSID 명령을 EXPAND(USER) 또는 EXPAND(SYSTEM) 옵션과 함께 사용하십시오. 페이지 세트 확장에 대한 일반 정보는 ALTER PSID 및 페이지 세트 확장(COPYPAGE)을 참조하십시오.

보다 큰 새 페이지 세트로 메시지 이동

이 기술을 사용하려면 큐 관리자를 중지하고 재시작해야 합니다. 그러면 재시작 시 공유 큐에 없는 비지속 메시지를 삭제합니다. 삭제하지 않을 비지속 메시지가 있는 경우 대신 로드 밸런싱을 사용하십시오. 자세한 정보는 288 페이지의 『페이지 세트의 로드 밸런싱 방법』의 내용을 참조하십시오. 이 설명에서 확장할 페이지 세트는 소스 페이지 세트라고 하며, 보다 큰 새 페이지 세트는 대상 페이지 세트라고 합니다.

다음 단계를 수행하십시오.

1. 큐 관리자를 중지합니다.
2. 보다 큰 보조 범위 값의 목적지 페이지 세트를 정의하여 소스 페이지 세트보다 크도록 하십시오.
3. CSQUTIL의 FORMAT 함수를 사용하여 대상 페이지 세트를 형식화하십시오. 자세한 정보는 페이지 세트 형식화(FORMAT)를 참조하십시오.
4. CSQUTIL의 COPYPAGE 함수를 사용하여 소스 페이지 세트에서 대상 페이지 세트로 모든 메시지를 복사하십시오. 자세한 정보는 페이지 세트 확장(COPYPAGE)을 참조하십시오.
5. 다음 중 하나를 수행함으로써 목적지 페이지를 사용하여 큐 관리자를 재시작하십시오.
   - 대상 페이지 세트를 참조하도록 큐 관리자 시작된 태스크 프로시저를 변경하십시오.
   - AMS를 사용하여 소스 페이지 세트를 삭제한 다음 소스 페이지 세트의 이름과 같도록 대상 페이지 세트의 이름을 바꾸십시오.

주의:

IBM MQ 페이지 세트를 삭제하기 전에 필요한 백업 사본을 작성해야 합니다.

페이지 세트 축소 방법

IBM MQ 관리자 이외의 모든 사용자가 큐 관리자를 사용하지 못하게 하십시오. 예를 들어, 액세스 보안 설정을 변경하여 수행합니다.

거의 비어 있는 큰 페이지 세트가 있는 경우(DISPLAY USAGE 명령으로 표시) 해당 크기를 줄일 수 있습니다. 이를 수행하기 위한 프로시저에는 CSQUTI의 COPY, FORMAT, LOAD 함수를 사용해야 합니다(IBM MQ 유틸리티 프로그램 참조). 페이지 세트 0에는 이 프로시저가 적용되지 않습니다. 이 페이지 세트의 크기를 줄이는 것은 현실성이 없기 때문입니다. 이를 수행할 수 있는 유일한 방법은 큐 관리자를 다시 초기화하는 것입니다(311 페이지의 『큐 관리자 재초기화』 참조). 이 프로시저의 필수조건은 모든 UOW가 완료되고 페이지 세트가 일관되도록 시스템에서 모든 사용자를 제거하는 작업입니다.

1. STOP QMGR 명령을 QUIESCE 또는 FORCE 속성과 함께 사용하여 큐 관리자를 중지하십시오.

2. CSQUTIL의 SCOPY 함수를 PSID 옵션과 함께 실행하여 대형 페이지 세트의 모든 메시지 데이터를 복사하여 순차 데이터 세트에 저장하십시오.
3. 대형 페이지 세트를 바꾸도록 더 작은 새 페이지 세트 데이터 세트를 정의하십시오.
4. 292 페이지의 『3』 단계에서 작성한 페이지 세트에 대해 CSQUTIL의 FORMAT TYPE(NEW) 함수를 실행하십시오.
5. 292 페이지의 『3』 단계에서 작성한 페이지 세트를 사용하여 큐 관리자를 재시작하십시오.
6. CSQUTIL의 LOAD 함수를 실행하여 292 페이지의 『2』 단계 중에 저장한 모든 메시지를 다시 로드하십시오.
7. 모든 사용자에게 큐 관리자에 대한 액세스를 허용하십시오.
8. 이전 대형 페이지 세트를 삭제하십시오.

페이지 세트 재도입 방법

특정 시나리오에서는 이전 페이지 세트를 큐 관리자에 다시 온라인이 되도록 전환할 수 있는 것이 좋습니다. 특정 조치를 수행하지 않는 한, 이전 페이지 세트가 온라인이 되면 큐 관리자는 페이지 세트 자체와 체크포인트 레코드에 저장된 페이지 세트 복구 RBA가 이전 버전임을 인식하므로 페이지 세트가 최신이 되도록 자동으로 매체 복원을 시작합니다.

해당 매체 복원은 큐 관리자를 재시작할 때만 수행될 수 있으며, 테이프에 보유된 아카이브 로그를 읽어야 하는 경우 특히 상당한 시간이 걸립니다. 그러나 일반적으로 이 경우 페이지 세트가 사이 기간 동안 오프라인이므로 로그에는 페이지 세트 복구에 관련된 정보가 없습니다.

다음과 같은 세 가지 선택사항이 있습니다.

전체 매체 복원을 수행하도록 허용하십시오.

1. 큐 관리자를 중지합니다.
2. 큐 관리자의 시작된 태스크 프로시저와 CSQINP1 초기화 데이터 세트 둘 다에서 페이지 세트의 정의가 사용 가능하도록 하십시오.
3. 큐 관리자를 재시작하십시오.

페이지 세트의 모든 메시지를 삭제하도록 허용하십시오.
이 선택은 페이지 세트가 장기간(예: 몇 달) 오프라인이며 다른 용도로 재사용하도록 결정한 경우 유용합니다.

1. CSQUTIL의 FORMAT 함수를 TYPE(NEW) 옵션과 함께 사용하여 페이지 세트를 형식화하십시오.
2. 큐 관리자의 시작된 태스크 프로시저 및 CSQINP1 초기화 데이터 세트 둘 다에서 페이지 세트의 정의를 추가하십시오.
3. 큐 관리자를 재시작하십시오.

형식화를 수행하는 데 TYPE(NEW) 옵션을 사용하면 페이지 세트의 현재 컨텐츠를 지우고 큐 관리자에게 체크포인트에서 페이지 세트에 대한 실행 기록 정보를 무시하도록 지시합니다.

페이지 세트를 온라인으로 전환하여 매체 복원 프로세스를 방지하십시오.
큐 관리자를 문제 없이 시스템 종료한 후로 페이지 세트가 계속 오프라인인 경우에만 이 기술을 사용하십시오. 이 선택은 일반적으로 운영 문제(예: 큐 관리자를 시작하는 동안 실행되는 백업)로 인해 페이지 세트가 단기간 오프라인인 경우에 가장 적합합니다.

1. CSQUTIL의 FORMAT 함수를 TYPE(REPLACE) 옵션과 함께 사용하여 페이지 세트를 형식화하십시오.
2. DEFINE PSID 명령을 DSN 옵션과 함께 사용하여 동적으로 페이지 세트를 큐 관리자에 다시 추가하거나 큐 관리자가 재시작할 때 추가될 수 있도록 하십시오.

형식화를 수행하는 데 TYPE(REPLACE) 옵션을 사용하면 큐 관리자가 페이지 세트를 문제 없이 종료했는지 확인한 다음 이를 표시하여 매체 복원이 수행되지 않도록 합니다. 페이지 세트의 컨텐츠는 변경되지 않습니다.

페이지 세트 백업 및 복구 방법

백업 및 복구에 사용 가능한 여러 메커니즘이 있습니다. 이 주제를 통해 이러한 메커니즘을 이해할 수 있습니다.

이 절은 다음 주제를 설명합니다.

- 293 페이지의 『비공유 자원의 복구점 작성』
- 294 페이지의 『페이지 세트 백업』
- 294 페이지의 『페이지 세트 복구』
- 페이지 세트 삭제 방법

공유 자원의 복구점 작성 방법에 대한 자세한 정보는 299 페이지의 『공유 큐 복구』의 내용을 참조하십시오.

비공유 자원의 복구점 작성

다음 조건이 둘 다 해당되는 경우 IBM MQ가 오브젝트와 비공유 지속 메시지를 현재 상태로 복구할 수 있습니다.

1. 이전 지점의 페이지 세트 사본이 존재하는 경우
2. 해당 복구점부터 복구를 수행하기 위해 모든 IBM MQ 로그를 사용할 수 있습니다.

이는 비공유 자원의 복구점을 나타냅니다.

오브젝트와 메시지는 둘 다 페이지 세트에 보관됩니다. 다른 큐의 여러 오브젝트와 메시지가 동일한 페이지 세트에 존재할 수 있습니다. 복구 목적으로, 오브젝트와 메시지를 분리하여 백업할 수 없으므로 올바른 데이터 복구를 위해서는 페이지 세트 전체를 백업해야 합니다.

IBM MQ 복구 로그에는 모든 지속 메시지의 레코드와 오브젝트에 대한 변경사항이 포함됩니다. IBM MQ가 실패하는 경우(예를 들어, 페이지 세트의 I/O 오류로 인해) 백업 사본을 복원하고 큐 관리자를 재시작하여 페이지 세트를 복구할 수 있습니다. IBM MQ는 백업 사본 지점부터 페이지 세트에 로그 변경사항을 적용합니다.

복구점은 다음과 같은 두 가지 방법으로 작성합니다.

전체 백업
페이지 세트에서 모든 업데이트를 강제 실행하는 큐 관리자를 중지합니다.

이 방법을 사용하면 백업된 페이지 세트 데이터 세트 및 해당 지점의 로그만을 사용하여 복구점에서 재시작할 수 있습니다.

퍼지 백업
큐 관리자를 중지하지 않고 페이지 세트의 퍼지 백업 사본을 작성합니다.

이 메소드를 사용하고 연관된 로그가 나중에 손상되거나 손실되는 경우에는 퍼지 페이지 세트 백업 사본을 사용하여 복구할 수 없습니다. 이는 퍼지 페이지 세트 백업 사본에 큐 관리자의 상태에 대한 일관되지 않은 보기가 들어 있고 사용 가능한 로그에 종속되어 있기 때문입니다. 로그를 사용할 수 없는 경우, 서브시스템이 비활성 상태일 때 작성된 백업 페이지 세트 사본의 마지막 세트로 돌아가야 하며(방법 1) 해당 시점으로부터의 데이터 손실을 허용해야 합니다.

방법 1: 전체 백업

이 방법은 큐 관리자 종료와 관련이 있습니다. 이 경우 페이지 세트가 일관된 상태가 되도록 페이지 세트에 대한 모든 업데이트를 강제 실행합니다.

1. 큐 관리자를 사용하는 모든 IBM MQ 애플리케이션을 중지하십시오(애플리케이션이 먼저 완료되도록 허용). 예를 들어, 액세스 보안 또는 큐 설정을 변경하여 이를 수행할 수 있습니다.
2. 모든 활동이 완료되면 복구 인다우트 단위를 표시 및 해석하십시오. DISPLAY CONN 및 RESOLVE INDOUBT에 설명된 대로 DISPLAY CONN 및 RESOLVE INDOUBT 명령을 사용하십시오.

   그러면 페이지 세트가 일관된 상태가 됩니다. 이를 수행하지 않으면 페이지 세트가 일관되지 않고 퍼지 백업을 효과적으로 수행하게 됩니다.
3. ARCHIVE LOG 명령을 실행하여 최신 로그 데이터가 로그 데이터 세트에 기록되도록 하십시오.
4. STOP QMGR MODE(QUIESCE) 명령을 실행하십시오. CSQI024I 또는 CSQI025I 메시지에 가장 낮은 RBA 값을 기록하십시오(자세한 정보는 CSQI024I 및 CSQI025I 참조). RBA 값으로 표시된 로그 데이터 세트에서 시작하여 현재 로그 데이터 세트까지 로그 데이터 세트를 유지해야 합니다.
5. 모든 큐 관리자 페이지 세트의 백업 사본을 작성하십시오(294 페이지의 『페이지 세트 백업』 참조).

방법 2: 퍼지 백업

이 방법은 큐 관리자 종료와 관련이 없습니다. 따라서 백업 프로세스에서는 업데이트가 가상 스토리지 버퍼에 있을 수 있습니다. 이는 페이지 세트가 일관된 상태가 아니며 로그를 사용한 복구에만 사용할 수 있음을 의미합니다.

1. DISPLAY USAGE TYPE(ALL) 명령을 실행하고 CSQI024I 또는 CSQI025I 메시지의 RBA 값을 기록하십시오(자세한 정보는 CSQI024I 및 CSQI025I 참조).
2. 페이지 세트의 백업 사본을 작성하십시오(294 페이지의 『페이지 세트 백업』 참조).
3. ARCHIVE LOG 명령을 실행하여 최신 로그 데이터가 로그 데이터 세트에 기록되도록 하십시오. 복구점에서 재시작하려면 RBA 값으로 표시된 로그 데이터 세트에서 시작하여 현재 로그 데이터 세트까지 로그 데이터 세트를 유지해야 합니다.

페이지 세트 백업

페이지 세트를 복구하려면 IBM MQ가 로그 내 복구점을 알아야 합니다. IBM MQ는 각 페이지 세트의 페이지 0에서 복구 로그 순서 번호(LSN)라는 로그 RBA 번호를 유지합니다. 이 번호는 로그에서 IBM MQ가 페이지 세트를 복구할 수 있는 시작 RBA입니다. 페이지 세트를 백업하면 이 번호 또한 복사됩니다.

나중에 페이지 세트를 복구하기 위해 사본을 사용하는 경우, IBM MQ가 이 RBA 값에서 현재 RBA까지 모든 로그 레코드에 액세스할 수 있어야 합니다. 이는 IBM MQ가 사용자가 보존하려는 페이지 세트의 가장 오래된 백업 사본부터 복구할 수 있는 충분한 로그 레코드를 보존해야 함을 의미합니다.

페이지 세트를 복사하려면 ADRDSSU COPY 함수를 사용하십시오.

자세한 정보는 논리 데이터 세트의 COPY DATASET 명령 구문 문서를 참조하십시오.

예를 들면, 다음과 같습니다.

```
//STEP2 EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
 COPY -
 DATASET(INCLUDE(SCENDATA.MQPA.PAGESET.*)) -
 RENAMEU(SCENDATA.MQPA.PAGESET.**,SCENDATA.MQPA.BACKUP1.**) -
 SPHERE -
 REPUNC -
 FASTREPLICATION(PREF )-
 CANCELERROR -
 TOL(ENQF)
/*
//
```

큐 관리자가 실행되는 동안 페이지 세트를 복사하는 경우 먼저 페이지 세트의 페이지 0을 복사하는 복사 유틸리티를 사용해야 합니다. 이를 수행하지 않으면 페이지 세트에서 데이터가 손상될 수 있습니다.

예를 들어 시스템 정전으로 인해 페이지 세트를 동적으로 확장하는 프로세스가 중단되더라도 ADRDSSU를 사용하여 계속 페이지 세트 백업을 작성할 수 있습니다.

액세스 메소드 서비스 IDCAMS `LISTCAT ENT('page set data set name') ALLOC`를 수행하는 경우 `HI-ALLOC-RBA` 가 `HI-USED-RBA`보다 높음을 알 수 있습니다.

다음에 이 페이지 세트가 가득 차면 다시 확장되고, 가능한 경우 자주 사용되는 RBA와 가장 높이 할당된 RBA 사이의 페이지가 또 다른 새 범위와 함께 사용됩니다.

오브젝트 정의 백업

오브젝트 정의의 사본을 백업해야 합니다. 이를 수행하려면 CCSQUTIL COMMAND 함수의 MAKEDEF 기능을 사용하십시오(IBM MQ에 명령 실행(COMMAND)에 설명되어 있음).

큐 관리자의 백업 사본을 작성할 때마다 오브젝트 정의를 백업하고 최신 버전을 유지하십시오.

페이지 세트 복구

큐 관리자가 장애로 인해 종료된 경우 큐 관리자는 재시작 중에 수행된 모든 복구와 함께 정상적으로 재시작할 수 있습니다. 그러나 페이지 세트 또는 로그 데이터 세트가 사용 불가능한 경우에는 이러한 복구가 불가능합니다. 현재 복구할 수 있는 범위는 페이지 세트와 로그 데이터 세트의 백업 사본의 가용성에 따라 다릅니다.

복구점에서 재시작하려면 다음이 필요합니다.

- 복구될 페이지 세트의 백업 사본
- 294 페이지의 『방법 2: 퍼지 백업』에 설명된 "퍼지" 백업 프로세스를 사용한 경우에는 레코드된 RBA 값을 포함한 로그 데이터 세트, ARCHIVE LOG 명령으로 작성된 로그 데이터 세트 및 이들 사이의 모든 로그 데이터 세트
- 전체 백업을 사용했지만 ARCHIVE LOG 명령으로 작성된 로그 데이터 세트 다음에 로그 데이터 세트가 없는 경우에는 모든 페이지 세트에 대해 CSQUTIL 유틸리티의 FORMAT TYPE(REPLACE) 함수를 실행하지 **않아도** 됩니다.

페이지 세트를 현재 상태로 복구하려면 ARCHIVE LOG 명령 이후의 모든 로그 데이터 세트와 레코드도 있어야 합니다.

페이지 세트를 복구하는 방법은 두 가지가 있습니다. 어떤 방법을 사용해도 큐 관리자가 중지되어야 합니다.

단순 복구

좀 더 간단한 방법이며 대부분의 복구 상황에 적합합니다.

1. 백업에서 복원할 페이지 세트를 삭제하십시오.
2. ADRDSSU COPY 함수를 사용하여 백업 사본에서 페이지 세트를 복구하십시오.

   또는 백업 사본을 원래 이름으로 바꾸거나 큐 관리자 프로시저에서 SQP00xx DD 명령문을 백업 페이지 세트를 가리키도록 변경할 수 있습니다. 그러나 이후 페이지 세트가 손실되거나 손상되면 더 이상 복원할 백업 사본이 없게 됩니다.
3. 큐 관리자를 재시작하십시오.
4. 큐 관리자가 재시작된 경우 애플리케이션을 재시작할 수 있습니다.
5. 복원된 페이지의 정상 백업 프로시저를 복원시키십시오.

고급 복구

이 방법은 복구할 페이지 세트가 크거나 마지막 백업 사본이 작성된 이후 페이지 세트에 활동이 많았던 경우 성능상의 이점을 제공합니다. 그러나 단순한 방법보다 수동 개입이 더 많이 필요하므로 오류 위험과 복구를 수행하는 데 소요되는 시간이 증가할 수 있습니다.

1. 백업에서 복원할 페이지 세트를 삭제 및 재정의하십시오.
2. ADRDSSU를 사용하여 페이지 세트의 백업 사본을 새 페이지 세트에 복사하십시오. 새 페이지 세트가 동적으로 확장될 수 있도록 보조 범위 값으로 새 페이지 세트를 정의하십시오.

   또는 백업 사본을 원래 이름으로 바꾸거나 큐 관리자 프로시저에서 SQP00xx DD 명령문을 백업 페이지 세트를 가리키도록 변경할 수 있습니다. 그러나 이후 페이지 세트가 손실되거나 손상되면 더 이상 복원할 백업 사본이 없게 됩니다.
3. 복구 중인 페이지 세트와 연관된 버퍼 풀을 가능한 크게 만들도록 큐 관리자의 CSQINP1 정의를 변경하십시오. 버퍼 풀을 크게 만들면 변경된 페이지를 모두 버퍼 풀에서 보존하여 페이지 세트에 대한 I/O 양을 줄일 수 있습니다.
4. 큐 관리자를 재시작하십시오.
5. 큐 관리자가 재시작되면 일시정지를 사용하여 중지한 다음 해당 페이지 세트의 정상 버퍼 풀 정의를 사용하여 재시작하십시오. 이 두 번째 재시작이 완료된 후 애플리케이션을 재시작할 수 있습니다.
6. 복원된 페이지의 정상 백업 프로시저를 복원시키십시오.

큐 관리자가 재시작될 때 발생하는 사항

큐 관리자가 재시작되면 페이지 세트의 재시작 지점부터 시작하여 로그에 등록된 페이지 세트에 작성된 모든 변경사항을 적용합니다. IBM MQ는 이 방법으로 여러 페이지 세트를 복구할 수 있습니다. 페이지 세트는 필요한 경우 매체 복원 중에 동적으로 확장됩니다.

재시작 동안 IBM MQ는 다음에서 가장 작은 값을 가져와 시작할 로그 RBA를 판별합니다.

- 각 페이지 세트의 체크포인트 로그 레코드에서 복구 LSN
- 각 페이지 세트의 페이지 0에서 복구 LSN
- 백업 작성 시 시스템에서 가장 오래된 불완전 복구 단위의 RBA

모든 오브젝트 정의는 페이지 세트 0에 저장됩니다. 메시지는 사용 가능한 모든 페이지 세트에 저장될 수 있습니다.

참고: 페이지 세트 0을 사용할 수 없으면 큐 관리자를 재시작할 수 없습니다.

페이지 세트 삭제 방법

DELETE PSID 명령을 사용하여 페이지 세트를 삭제합니다. 이 명령에 대한 자세한 내용은 DELETE PSID를 참조하십시오.

스토리지 클래스가 계속 참조하는 페이지 세트는 삭제할 수 없습니다. DISPLAY STGCLASS를 사용하여 페이지 세트를 참조하는 스토리지 클래스를 확인할 수 있습니다.

데이터 세트는 IBM MQ에서 할당이 취소되지만 삭제되지는 않습니다. 나중에 계속 사용하거나 z/OS 기능을 사용하여 삭제될 수 있습니다.

큐 관리자의 시작된 태스크 프로시저에서 페이지 세트를 제거하십시오.

CSQINP1 초기화 데이터 세트에서 페이지 세트 정의를 제거하십시오.

CSQUTIL을 사용하여 큐 백업 및 복원 방법

이 주제는 CSQUTIL을 사용한 백업 및 복원에 대한 추가 정보를 얻기 위한 참조로 사용하십시오.

큐 백업과 복원을 위해 CSQUTIL 유틸리티 함수를 사용할 수 있습니다. 큐를 백업하려면 COPY 또는 SCOPY 함수를 사용하여 메시지를 큐에서 데이터 세트로 복사하십시오. 큐를 복원하려면 보완 함수 LOAD 또는 SLOAD를 사용하십시오. 자세한 정보는 IBM MQ 유틸리티 프로그램을 참조하십시오.

버퍼 풀 관리

버퍼 풀을 변경 또는 삭제하려는 경우 이 주제를 사용합니다.

이 주제는 버퍼 풀을 대체 및 삭제하는 방법을 설명합니다. 여기에는 다음 절이 포함되어 있습니다.

- 296 페이지의 『버퍼 풀에서 버퍼 수를 변경하는 방법』
- 297 페이지의 『버퍼 풀을 삭제하는 방법』

버퍼 풀은 큐 관리자 초기화 중에 초기화 입력 데이터 세트 CSQINP1에서 실행된 DEFINE BUFFPOOL 명령을 사용하여 정의됩니다. 해당 속성은 비즈니스 요구사항에 따라 큐 관리자가 실행되는 동안 이 주제에서 설명하는 프로세스를 사용하여 대체될 수 있습니다. 큐 관리자는 체크포인트 로그 레코드에 현재 버퍼 풀 속성을 기록합니다. 이 속성은 CSQINP1의 버퍼 풀 정의에 REPLACE 속성이 포함되지 않는 경우 다음에 큐 관리자를 재시작할 때 자동으로 복원됩니다.

현재 버퍼 속성을 표시하려면 DISPLAY USAGE 명령을 사용하십시오.

DEFINE PSID 명령과 DSN 옵션을 사용하여 버퍼 풀을 동적으로 정의할 수도 있습니다.

버퍼 풀을 동적으로 변경하는 경우 초기화 데이터 세트 CSQINP1에서도 해당 정의를 업데이트해야 합니다.

페이지 세트, 스토리지 클래스, 버퍼, 버퍼 풀에 대한 설명과 적용되는 일부 성능 고려사항은 z/OS 에 대한 계획의 내용을 참조하십시오.

참고: 버퍼 풀은 많은 저장 공간을 사용합니다. 버퍼 풀 크기를 늘리거나 새 버퍼 풀을 정의할 때는 충분한 저장 공간을 사용할 수 있는지 확인해야 합니다. 자세한 정보는 주소 공간 스토리지를 참조하십시오.

버퍼 풀에서 버퍼 수를 변경하는 방법

버퍼 풀이 너무 작은 경우 콘솔에 CSQP020E 메시지가 표시되며 ALTER BUFFPOOL 명령을 다음과 같이 사용하여 버퍼를 더 할당할 수 있습니다.

1. 로그에서 CSQY220I 메시지를 보고 새 버퍼에 사용 가능한 공간을 판별하십시오. 사용 가능한 공간은 MB 단위로 보고됩니다. 버퍼 하나의 크기는 4KB이므로 사용 가능한 1MB 공간마다 256개 버퍼를 할당할 수 있습니다. 여유 공간 중 일부는 다른 태스크에 필요하므로 모든 여유 공간을 버퍼에 할당하지는 마십시오.

   버퍼 풀이 고정된 4KB 페이지를 사용하는 경우, 즉 PAGECLAS 속성이 FIXED4KB인 경우에는 LPAR에서 사용 가능한 실제 스토리지가 충분한지 확인하십시오.
2. 보고된 여유 공간이 적절하지 않은 경우에는 다음 명령을 사용하여 다른 버퍼 풀에서 일부 버퍼를 해제하십시오.

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   여기서 *buf-pool-id*는 공간을 재확보할 버퍼 풀이고 *integer*는 이 버퍼 풀에 할당될 새 버퍼 수로, 할당된 원래 버퍼 수보다 작아야 합니다.
3. 다음 명령을 사용하여 확장하려는 버퍼 풀에 버퍼를 추가하십시오.

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   여기서 *buf-pool-id*는 확장될 버퍼 풀이고 *integer*는 이 버퍼 풀에 할당될 새 버퍼 수로, 할당된 원래 버퍼 수보다 커야 합니다.

버퍼 풀을 삭제하는 방법

페이지 세트가 더 이상 사용하지 않는 버퍼 풀은 삭제하여 할당된 가상 스토리지를 해제하십시오.

DELETE BUFFPOOL 명령을 사용하여 버퍼 풀을 삭제합니다. 페이지 세트가 이 버퍼 풀을 사용하는 경우에는 명령이 실패합니다.

페이지 세트 삭제 방법에 대한 정보는 296 페이지의 『페이지 세트 삭제 방법』의 내용을 참조하십시오.

큐 공유 그룹 및 공유 큐 관리

IBM MQ는 다양한 유형의 공유 자원(예를 들어, 큐 공유 그룹, 공유 큐, 커플링 기능)을 사용할 수 있습니다. 이 주제를 사용하여 이러한 공유 자원을 관리하는 데 필요한 프로시저를 검토합니다.

이 절에는 다음 주제에 대한 정보가 포함되어 있습니다.

큐 공유 그룹 관리

큐 공유 그룹에 큐 관리자를 추가하거나 제거하고 연관된 Db2 테이블을 관리할 수 있습니다.

이 주제에는 다음 태스크에 대한 절이 포함됩니다.

Db2 테이블에 큐 공유 그룹 추가

Db2 테이블에 큐 공유 그룹을 추가하려면 큐 공유 그룹 유틸리티(CSQ5PQSG)의 ADD QSG 함수를 사용하십시오. 이 프로그램은 큐 공유 그룹 유틸리티에 설명되어 있습니다. thlqual.SCSQPROC(CSQ45AQS)에 샘플이 제공됩니다.

큐 공유 그룹에 큐 관리자 추가

큐 공유 그룹에 큐 관리자가 추가될 수 있으며 이 주제는 몇몇 제한사항을 설명합니다.

큐 공유 그룹에 큐 관리자를 추가하려면 큐 공유 그룹 유틸리티(CSQ5PQSG)의 ADD QMGR 함수를 사용하십시오. 이 프로그램은 큐 공유 그룹 유틸리티에 설명되어 있습니다. thlqual.SCSQPROC(CSQ45AQM)에 샘플이 제공됩니다.

큐 공유 그룹에 큐 관리자를 추가하려면 큐 공유 그룹이 존재해야 합니다.

큐 공유 그룹은 최대 4자의 이름을 가집니다. 이 이름은 네트워크에서 고유해야 하며 큐 관리자 이름과 달라야 합니다.

큐 관리자는 하나의 큐 공유 그룹의 구성원만 될 수 있습니다.

참고: IBM MQ의 이전 버전을 실행하는 큐 관리자가 있는 기존 큐 공유 그룹에 큐 관리자를 추가하려면 먼저 그룹에서 IBM MQ의 최상위 버전에 대한 공존 PTF를 그룹의 모든 이전 버전 큐 관리자에 적용해야 합니다.

큐 공유 그룹에서 큐 관리자 제거

다른 프로세스에서 큐 관리자의 로그가 필요하지 않은 경우에만 큐 공유 그룹에서 큐 관리자를 제거할 수 있습니다. 로그는 다음을 포함하는 경우 필요합니다.

- 큐 공유 그룹이 사용하는 커플링 기능(CF) 애플리케이션 구조 중 하나의 최신 백업
- 향후 복원 프로세스에 필요한 데이터. 즉, 마지막 백업 제외 간격 값으로 설명된 시간 이후에 큐 관리자가 복구 가능한 구조를 사용했습니다.

이 두 가지 사항 중 하나 또는 모두가 적용되는 경우에는 큐 관리자를 제거할 수 없습니다. 향후 복원 프로세스에 필요한 큐 관리자 로그를 판별하려면 MQSC DISPLAY CFSTATUS 명령과 TYPE(BACKUP) 옵션을 함께 사용하십시오. 이 명령에 대한 자세한 내용은 DISPLAY CFSTATUS를 참조하십시오.

큐 관리자 로그가 필요하지 않은 경우에는 다음 단계를 수행하여 큐 공유 그룹에서 큐 관리자를 제거하십시오.

1. 이 큐 관리자와 관련된 인다우트(in-doubt) 작업 단위를 해석하십시오.
2. STOP QMGR MODE(QUIESCE)를 사용하여 큐 관리자를 완전히 종료하십시오.
3. 다음 단계에서 BACKUP CFSTRUCT 명령에 지정할 EXCLINT 매개변수 값에 상당하는 간격 이상을 기다리십시오.
4. 또 다른 큐 관리자에서 MQSC BACKUP CFSTRUCT 명령을 사용하고 이전 단계에서 필요한 대로 EXCLINT 값을 지정하여 복구 가능한 각 CF 구조의 CF 구조 백업을 실행하십시오.
5. CSQ5PQSG 유틸리티의 REMOVE QMGR 함수를 사용하여 큐 공유 그룹에서 큐 관리자를 제거하십시오. 이 프로그램은 큐 공유 그룹 유틸리티에 설명되어 있습니다. thlqual.SCSQPROC(CSQ45RQM)에 샘플이 제공됩니다.
6. 큐 관리자를 재시작하기 전에 QSGDATA 시스템 매개변수를 기본값으로 재설정하십시오. 시스템 매개변수 구성 방법에 대한 정보는 CSQ6SYSP 사용을 참조하십시오.

큐 공유 그룹에서 마지막 큐 관리자를 제거할 때는 REMOVE가 아닌 FORCE 옵션을 사용해야 합니다. 그러면 큐 공유 그룹에서 큐 관리자가 제거되며 복구에 필요한 큐 관리자 로그에 대한 일관성 검사를 수행하지 않습니다. 이 조작은 큐 공유 그룹을 삭제하는 경우에만 수행해야 합니다.

Db2 테이블에서 큐 공유 그룹 제거

Db2 테이블에서 큐 공유 그룹을 제거하려면 큐 공유 그룹 유틸리티(CSQ5PQSG)의 REMOVE QSG 함수를 사용하십시오. 이 프로그램은 큐 공유 그룹 유틸리티에 설명되어 있습니다. thlqual.SCSQPROC(CSQ45RQS)에 샘플이 제공됩니다.

큐 공유 그룹에서 모든 큐 관리자를 제거한 후에만 공용 Db2 데이터 공유 그룹 테이블에서 큐 공유 그룹을 제거할 수 있습니다(298 페이지의 『큐 공유 그룹에서 큐 관리자 제거』의 설명 참조).

큐 공유 그룹 레코드가 큐 공유 그룹 관리 테이블에서 삭제되면 해당 큐 공유 그룹과 관련된 관리 정보와 모든 오브젝트가 다른 IBM MQ Db2 테이블에서 삭제됩니다. 여기에는 공유 큐 및 그룹 오브젝트 정보가 포함됩니다.

Db2 정의 일관성에 대한 유효성 검사

어떤 이유로 Db2 오브젝트 정의가 불일치하게 되면 큐 공유 그룹 내 공유 그룹에 문제점에 발생할 수 있습니다.

큐 관리자, CF 구조, 공유 큐에 대한 Db2 오브젝트 정의 일관성의 유효성을 검증하려면 큐 공유 그룹 유틸리티(CSQ5PQSG)의 VERIFY QSG 함수를 사용하십시오. 이 프로그램은 큐 공유 그룹 유틸리티에 설명되어 있습니다.

공유 큐 관리

이 주제를 통해 공유 큐 복구, 이동, 마이그레이션 방법을 이해할 수 있습니다.

이 절에서는 다음 태스크를 설명합니다.

- 299 페이지의 『공유 큐 복구』
- 299 페이지의 『공유 큐 이동』
- 302 페이지의 『공유 큐로 비공유 큐 마이그레이션』
- Db2 연결 일시중단

공유 큐 복구

IBM MQ는 다음 조건이 모두 충족되는 경우 공유 큐에서 지속 메시지를 복구할 수 있습니다.

- 메시지를 포함하는 CF 구조 백업이 수행되었습니다.
- 큐 공유 그룹의 모든 큐 관리자에 대한 모든 로그를 사용하여 백업이 생성된 지점부터 복구를 수행할 수 있습니다.
- Db2를 사용할 수 있으며 구조 백업 테이블이 최신 CF 구조 백업보다 최신 상태입니다.

공유 큐의 메시지는 커플링 기능(CF) 구조에 저장됩니다. 지속 메시지는 공유 큐에 넣을 수 있으며 비공유 큐의 지속 메시지처럼 큐 관리자 로그에 복사됩니다. MQSC BACKUP CFSTRUCT 및 RECOVER CFSTRUCT 명령은 자주 발생하지는 않지만 커플링 기능이 실패하는 경우 CF 구조 복구를 허용하기 위해 제공됩니다. 이러한 경우, 영향을 받은 구조에 저장된 모든 비지속 메시지는 손실되지만 지속 메시지는 복구될 수 있습니다. 구조가 복구될 때까지는 구조를 사용하는 추가 애플리케이션 활동이 방지됩니다.

복구를 사용하려면 MQSC BACKUP CFSTRUCT 명령을 사용하여 커플링 기능 목록 구조를 자주 백업해야 합니다. CF 구조의 메시지는 백업을 작성하는 큐 관리자의 활성 로그 데이터 세트에 기록됩니다. Db2에 백업 레코드(백업되는 CF 구조의 이름, 백업을 수행하는 큐 관리자의 이름, 해당 큐 관리자 로그에서 이 백업에 대한 RBA 범위, 백업 시간)를 기록합니다. CF 목록 구조는 공유 큐를 적극적으로 사용하지 않더라도 백업하십시오(예를 들어, 향후 사용하도록 큐 공유 그룹을 설정한 경우).

복구를 수행할 수 있는 큐 관리자에 MQSC RECOVER CFSTRUCT 명령을 실행하여 CF 구조를 복구할 수 있습니다. 큐 공유 그룹에서 임의 큐 관리자를 사용할 수 있습니다. 복구할 하나의 CF 구조를 지정하거나 동시에 여러 CF 구조를 복구할 수 있습니다.

앞에서 언급한 것처럼, CF 목록 구조는 자주 백업해야 하며 그렇지 않으면 CF 구조를 복구하는 데 시간이 오래 소요될 수 있습니다. 또한 복구 프로세스는 취소할 수 없습니다.

공유 큐의 정의는 Db2 데이터베이스에 보관되므로 필요한 경우 표준 Db2 데이터베이스 프로시저를 사용하여 복구될 수 있습니다. 자세한 정보는 공유 큐 및 큐 공유 그룹을 참조하십시오.

공유 큐 이동

이 절은 공유 큐를 특정 커플링 기능 구조에서 다른 커플링 기능 구조로 이동하여 로드 밸런싱을 수행하는 방법을 설명합니다. 또한 비공유 큐를 공유 큐로 이동하는 방법과 공유 큐를 비공유 큐로 이동하는 방법을 설명합니다.

큐를 이동할 때 프로시저의 일부로 임시 큐를 정의해야 합니다. 이는 모든 큐의 이름이 고유하므로 큐의 큐 속성 지정이 다르더라도 같은 이름의 큐를 두 개 가질 수 없기 때문입니다. IBM MQ는 이름이 같은 두 개의 큐를 가질 수 있도록 허용하지만(300 페이지의 『2』 단계 참조) 큐를 사용할 수는 없습니다.

- 특정 커플링 기능 구조에서 다른 커플링 기능 구조로 큐 이동
- 비공유 큐를 공유 큐로 이동
- 비공유 큐로 공유 큐 이동

특정 커플링 기능 구조에서 다른 커플링 기능 구조로 큐 이동

특정 CF 구조에서 다른 CF 구조로 큐와 해당 메시지를 이동하려면 MQSC MOVE QLOCAL 명령을 사용하십시오. 새 CF 구조로 이동하려는 하나 이상의 큐를 식별했으면 다음 프로시저를 사용하여 각 큐를 이동하십시오.

1. 이동하려는 큐를 애플리케이션이 사용하지 않는지, 즉 큐 속성 IPPROCS 및 OPPROCS가 큐 공유 그룹의 모든 큐 관리자에서 0인지 확인하십시오.
2. 애플리케이션이 MQPUT 을 사용하지 않도록 큐 정의를 변경하여 큐에 메시지를 넣지 않도록 합니다. 큐 정의를 PUT(DISABLED)로 변경하십시오.
3. 다음 명령을 사용하여 이동 중인 큐와 속성이 동일한 임시 큐를 정의하십시오.

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```

   참고: 이전 실행에서 이 임시 큐를 작성한 경우 정의하기 전에 이 큐를 삭제하십시오.
4. 다음 명령을 사용하여 임시 큐로 메시지를 이동하십시오.

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. 다음 명령을 사용하여 이동 중인 큐를 삭제하십시오.

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. 다음 명령을 사용하여 CFSTRUCT 속성을 변경하여 이동 중인 큐를 재정의하십시오.

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```

   큐를 다시 정의할 때 이 큐는 300 페이지의 『3』 단계에서 작성된 임시 큐를 기반으로 합니다.
7. 다음 명령을 사용하여 메시지를 새 큐로 다시 이동하십시오.

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

8. 300 페이지의 『3』 단계에서 작성한 큐는 더 이상 필요하지 않습니다. 다음 명령을 사용하여 해당 큐를 삭제하십시오.

```
DELETE QL(TEMP_QUEUE)
```

9. 이동 중인 큐가 CSQINP2 데이터 세트에 정의된 경우 CSQINP2 데이터 세트에서 적절한 DEFINE QLOCAL 명령의 CFSTRUCT 속성을 변경하십시오. 기존 큐 정의를 바꾸도록 REPLACE 키워드를 추가하십시오.

301 페이지의 그림 46은 특정 CF 구조에서 다른 CF 구조로 큐를 이동하기 위한 샘플 작업을 보여줍니다.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

그림 *46.* 특정 *CF* 구조에서 다른 *CF* 구조로 큐를 이동하기 위한 샘플 작업

비공유 큐를 공유 큐로 이동

비공유 큐에서 공유 큐로 이동하는 프로시저는 특정 CF 구조에서 다른 CF 구조로 큐를 이동하는 프로시저와 유사합니다(300 페이지의 『특정 커플링 기능 구조에서 다른 커플링 기능 구조로 큐 이동』 참조). 301 페이지의 그림 47는 이를 수행하기 위한 샘플 작업을 제공합니다.

참고: 공유 큐의 메시지에는 최대 메시지 크기, 메시지 지속성, 큐 색인 유형에 대한 특정 제한사항이 적용되므로 일부 비공유 큐를 공유 큐로 이동하지 못할 수 있습니다.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

그림 *47.* 비공유 큐에서 공유 큐로 이동하기 위한 샘플 작업

비공유 큐로 공유 큐 이동

공유 큐에서 비공유 큐로 이동하는 프로시저는 특정 CF 구조에서 다른 CF 구조로 큐를 이동하는 프로시저와 유사합니다(300 페이지의 『특정 커플링 기능 구조에서 다른 커플링 기능 구조로 큐 이동』 참조).

302 페이지의 그림 48는 이를 수행하기 위한 샘플 작업을 제공합니다.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW) QSGDISP(QMGR)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

그림 *48.* 공유 큐에서 비공유 큐로 이동하기 위한 샘플 작업

공유 큐로 비공유 큐 마이그레이션

비공유 큐를 공유 큐로 마이그레이션하려면 다음 두 단계가 필요합니다.

- 큐 공유 그룹의 첫 번째(또는 유일한) 큐 관리자 마이그레이션
- 큐 공유 그룹의 다른 큐 관리자 마이그레이션

큐 공유 그룹의 첫 번째(또는 유일한) 큐 관리자 마이그레이션

301 페이지의 그림 47는 비공유 큐를 공유 큐로 이동하기 위한 작업 예를 보여줍니다. 마이그레이션이 필요한 각 큐에 이 작업을 수행합니다.

참고:

1. 공유 큐의 메시지에는 최대 메시지 크기, 메시지 지속성, 큐 색인 유형에 대한 특정 제한사항이 적용되므로 일부 비공유 큐를 공유 큐로 이동하지 못할 수 있습니다.
2. 공유 큐의 올바른 색인 유형을 사용해야 합니다. 공유 큐로 전송 큐를 마이그레이션하는 경우에는 색인 유형이 MSGID여야 합니다.

큐가 비어 있거나 큐에 있는 메시지를 보관하지 않아도 되는 경우에는 큐 마이그레이션이 보다 단순합니다. 302 페이지의 그림 49는 이러한 상황에서 사용할 작업 예를 보여줍니다.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN  DD  *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD  *
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
DELETE QL(TEMP_QUEUE)
/*
```

그림 *49.* 메시지가 없는 비공유 큐를 공유 큐로 이동하기 위한 샘플 작업

큐 공유 그룹의 다른 큐 관리자 마이그레이션

1. 기존 공유 큐와 이름이 다른 각 큐마다 301 페이지의 그림 47 또는 302 페이지의 그림 49에서 설명하는 대로 큐를 이동하십시오.

2. 기존 공유 큐와 이름이 같은 큐의 경우에는 303 페이지의 그림 50에 표시된 명령을 사용하여 공유 큐로 메시지를 이동하십시오.

```
MOVE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR)
```

그림 *50.* 비공유 큐에서 기존 공유 큐로 메시지 이동

Db2에 대한 연결 일시중단

큐 관리자를 중지하지 않고 공유 큐와 관련된 Db2 테이블 또는 패키지에 유지보수 또는 서비스를 적용하려면 Db2에서 데이터 공유 그룹(DSG)에 있는 큐 관리자의 연결을 일시적으로 끊어야 합니다.

이를 수행하려면:

1. MQSC 명령 SUSPEND QMGR FACILITY(Db2)를 사용하십시오.
2. 바인딩을 수행하십시오.
3. MQSC 명령 RESUME QMGR FACILITY(Db2)를 사용하여 Db2에 다시 연결하십시오.

이 명령을 사용하는 데는 제한사항이 있습니다.

주의: Db2 연결이 일시중단되어 있는 동안에는 다음과 같은 조작을 사용할 수 없습니다. 따라서 엔터프라이즈가 가장 한가할 때 이 작업을 수행해야 합니다.

- 관리(정의, 삭제, 변경)를 위해 공유 큐 오브젝트에 액세스
- 공유 채널 시작
- Db2에서 메시지 저장
- CFSTRUCT 백업 또는 복구

그룹 오브젝트 관리

이 주제를 통해 그룹 오브젝트에 대한 작업을 수행하는 방법을 이해할 수 있습니다.

IBM MQ는 그룹 오브젝트 정의를 해당 정의를 사용하는 각 큐 관리자의 페이지 세트 0에 자동으로 복사합니다. 정의 사본을 임시로 대체하고 IBM MQ를 사용하여 저장소 사본에서 페이지 세트 사본을 새로 고칠 수 있습니다. IBM MQ는 시작 시 항상 저장소 사본에서 페이지 세트 사본 새로 고치기를 시도합니다. 채널 오브젝트의 경우에는 채널 이니시에이터가 다시 시작될 때 이를 수행합니다. 이를 통해 큐 관리자가 비활성 상태일 때 수행된 변경사항을 포함한 저장소의 버전이 페이지 세트 사본에 반영됩니다.

예를 들어, 다음과 같은 경우에는 새로 고치기가 수행되지 않습니다.

- 큐의 사본이 열려 있으면 큐 사용을 변경하는 새로 고치기가 실패합니다.
- 큐의 사본에 해당 메시지가 있으면 해당 큐를 삭제하는 새로 고치기가 실패합니다.

이러한 상황에서 새로 고치기는 해당 사본에서 수행되지 않으며 다른 모든 큐 관리자의 사본에서 수행됩니다. 그룹 오브젝트를 추가, 변경 또는 삭제한 후, 큐 관리자에서 또는 채널 이니시에이터가 다시 시작될 때 복사 오브젝트 관련 문제점을 확인하고 정정하십시오.

커플링 기능 관리

이 주제를 통해 커플링 기능(CF) 구조를 추가하거나 제거하는 방법을 이해할 수 있습니다.

이 절에서는 다음 태스크를 설명합니다.

- 304 페이지의 『커플링 기능 구조 추가』
- 304 페이지의 『커플링 기능 구조 제거』

커플링 기능 구조 추가

커플링 기능 구조를 추가할 때는 IBM MQ 조치가 필요하지 않습니다. 태스크 10: 커플링 기능 설정의 커플링 기능 설정에 대한 정보는 커플링 기능 구조 이름 지정 규칙과 CFRM 정책 데이터 세트에서 구조를 정의하는 방법을 설명합니다.

커플링 기능 구조 제거

커플링 기능 구조를 제거하려면 다음 프로시저를 따르십시오.

- 다음 명령을 사용하여 삭제하려는 커플링 기능 구조를 사용하는 모든 큐의 목록을 가져오십시오.

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

- 구조를 사용하는 모든 큐를 삭제하십시오.
- 큐 공유 그룹에서 각 큐 관리자를 차례로 중지 및 재시작하여 구조에서 IBM MQ 및 Db2 연결을 끊고 해당 정보를 삭제하십시오. 모든 큐 관리자를 한 번에 중지할 필요는 없습니다. 한 번에 하나씩 중지합니다.
- CFRM 정책 데이터 세트에서 구조 정의를 제거하고 IXCMIAPU 유틸리티를 실행하십시오. 이는 태스크 10: 커플링 기능 설정에서 설명하는 사용자 정의 태스크 10(커플링 기능 설정)의 반대 작업입니다.

복구 및 재시작

이 주제를 통해 IBM MQ가 사용하는 복구 및 재시작 메커니즘을 이해할 수 있습니다.

IBM MQ 재시작

큐 관리자가 종료된 후에는 큐 관리자 종료 방식에 따라 다른 재시작 프로시저가 필요합니다. 이 주제를 통해 사용할 수 있는 다양한 재시작 프로시저를 이해할 수 있습니다.

이 주제에는 다음 상황에서 큐 관리자를 재시작하는 방법에 대한 정보가 포함되어 있습니다.

- 304 페이지의 『정상 종료 후 재시작』
- 304 페이지의 『비정상 종료 후 재시작』
- 305 페이지의 『페이지 세트가 손실된 경우 재시작』
- 305 페이지의 『로그 데이터 세트가 손실된 경우 재시작』
- CF 구조가 손실된 경우 재시작

정상 종료 후 재시작

STOP QMGR 명령으로 큐 관리자가 중지된 경우, 시스템은 순서대로 작업을 종료하며 중지하기 전에 종료 체크포인트를 수행합니다. 큐 관리자를 재시작할 때 이는 시스템 체크포인트 및 복구 로그의 정보를 사용하여 시스템 종료 시 시스템 상태를 판별합니다.

큐 관리자를 재시작하려면 START QMGR 명령을 실행하십시오(244 페이지의 『큐 관리자 시작 및 중지』의 설명 참조).

비정상 종료 후 재시작

IBM MQ는 정상 종료 또는 비정상 종료 후 재시작되는지 여부를 자동으로 감지합니다.

비정상 종료 후에 큐 관리자를 시작하는 것은 STOP QMGR 명령이 실행된 후 시작하는 것과 다릅니다. 큐 관리자가 비정상적으로 종료된 경우에는 작업을 완료하거나 종료 체크포인트를 수행하지 않고 종료합니다.

큐 관리자를 재시작하려면 START QMGR 명령을 실행하십시오(244 페이지의 『큐 관리자 시작 및 중지』의 설명 참조). 비정상 종료 후에 큐 관리자를 재시작하는 경우에는 로그에 있는 정보를 사용하여 종료 시 상태에 대한 정보를 새로 고치고 다양한 태스크의 상태를 사용자에게 알려줍니다.

일반적으로 재시작 프로세스는 모든 불일치 상태를 해결합니다. 그러나 경우에 따라 불일치를 해결하기 위해 특정 단계를 수행해야 합니다. 이 프로그램은 317 페이지의 『작업 단위 수동 복구』에서 설명합니다.

페이지 세트가 손실된 경우 재시작

페이지 세트가 손실된 경우 백업 사본에서 복원해야 큐 관리자를 재시작할 수 있습니다. 이 프로그램은 292 페이지의 『페이지 세트 백업 및 복구 방법』에서 설명합니다.

이러한 상황에서는 매체 복원에 필요한 시간으로 인해 큐 관리자를 재시작하는 데 시간이 오래 소요될 수 있습니다.

로그 데이터 세트가 손실된 경우 재시작

STOP QMGR 명령을 사용하여 큐 관리자를 중지한 후 로그 사본이 둘 다 손실 또는 손상되는 경우 방법 1: 전체 백업을 사용하여 생성된 페이지 세트의 일치 세트가 있으면 큐 관리자를 재시작할 수 있습니다.

다음 프로시저를 수행하십시오.

1. 큐 관리자의 각 기존 페이지 세트에 해당하는 새 페이지 세트를 정의하십시오. 페이지 세트 정의에 대한 정보는 태스크 15: 페이지 세트 정의를 참조하십시오.

   각 새 페이지 세트가 해당 소스 페이지 세트보다 큰지 확인하십시오.
2. CSQUTIL의 FORMAT 함수를 사용하여 대상 페이지 세트를 형식화하십시오. 자세한 정보는 페이지 세트 형식화를 참조하십시오.
3. CSQUTIL의 RESETPAGE 함수를 사용하여 기존 페이지 세트를 복사하거나 올바른 위치에서 재설정하고 각 페이지에서 로그 RBA를 재설정하십시오. 이 함수에 대한 자세한 정보는 페이지 세트 복사 및 로그 재설정을 참조하십시오.
4. CSQJU003을 사용하여 큐 관리자 로그 데이터 세트와 BSDS를 재정의하십시오(로그 인벤토리 변경 유틸리티 참조).
5. 새 페이지 세트를 사용하여 큐 관리자를 재시작하십시오. 이를 수행하려면 다음 중 하나를 수행하십시오.
   - 새 페이지 세트를 참조하도록 큐 관리자 시작된 태스크 프로시저를 변경합니다. 자세한 정보는 태스크 6: IBM MQ 큐 관리자용 프로시저 작성을 참조하십시오.
   - 액세스 방법 서비스를 사용하여 이전 페이지 세트를 삭제한 다음 이전 페이지 세트와 같은 이름을 지정하여 새 페이지 세트의 이름을 바꿉니다.

주의: IBM MQ 페이지 세트를 삭제하기 전에 필요한 백업 사본을 작성해야 합니다.

큐 관리자가 큐 공유 그룹의 멤버인 경우에는 로그가 손실 또는 손상되더라도 일반적으로 GROUP 및 SHARED 오브젝트 정의가 영향을 받지 않습니다. 그러나 공유 큐 메시지가 손실되거나 손상된 로그에 포함된 작업 단위와 관련이 있는 경우에는 커미트되지 않은 해당 메시지에 대한 영향을 예측할 수 없습니다.

참고: 로그가 손상되고 큐 관리자가 큐 공유 그룹의 멤버인 경우에는 공유 지속 메시지를 복구하지 못할 수 있습니다. RECOVER(YES) 속성을 갖는 모든 CF 구조의 큐 공유 그룹에 있는 다른 활성 큐 관리자에서 즉시 BACKUP CFSTRUCT 명령을 실행하십시오.

CF 구조가 손실된 경우 재시작

큐 관리자는 종료되지 않으므로 CF 기능이 손실되는 경우 재시작하지 않아도 됩니다.

z/OS 대체 사이트 복구

단일 큐 관리자 또는 큐 공유 그룹을 복구하거나 디스크 미러링을 고려할 수 있습니다.

자세한 정보는 다음 절을 참조하십시오.

- 대체 사이트에서 단일 큐 관리자 복구
- 큐 공유 그룹 복구
  - CF 구조 매체 복구
  - 기본 사이트에서 큐 공유 그룹 백업
  - 대체 사이트에서 큐 공유 그룹 복구
- 디스크 미러링 사용

대체 사이트에서 단일 큐 관리자 복구

IBM MQ 컴퓨팅 센터가 완전히 손실되는 경우 복구 사이트에서 다른 큐 관리자 또는 큐 공유 그룹을 복구할 수 있습니다. 큐 공유 그룹에 대한 대체 사이트 복구 프로시저는 309 페이지의 『대체 사이트에서 큐 공유 그룹 복구』의 내용을 참조하십시오.

복구 사이트에서 다른 큐 관리자를 복구하려면 페이지 세트와 로그를 정기적으로 백업해야 합니다. 모든 데이터 복구 조작과 마찬가지로, 재해 복구의 목표는 데이터, 워크로드 처리(업데이트), 시간을 최대한 적게 잃는 것입니다.

복구 사이트에서:

- 복구 큐 관리자는 **반드시** 손실된 큐 관리자와 이름이 같아야 합니다.
- 각 복구 큐 관리자에서 사용된 시스템 매개변수 모듈(예: CSQZPARM)에 손실된 해당 큐 관리자와 같은 매개변수가 포함되어야 합니다.

이 작업을 완료한 경우, 다음 프로시저에 설명된 대로 모든 큐 관리자를 재설정합니다. 이는 단일 큐 관리자의 복구 사이트에서 재해 복구를 수행하는 데 사용될 수 있습니다. 이 경우 사용 가능한 모든 사본은 다음과 같은 것으로 가정합니다.

- 기본 사이트에서 정상적으로 실행하여 작성된 아카이브 로그와 BSDS의 사본(활성 로그는 기본 사이트에서 큐 관리자와 함께 손실됨)
- 기본 사이트에서 사용 가능한 최신 아카이브 로그 사본보다 오래되거나 수명이 같은 큐 관리자의 페이지 세트 사본

활성 및 아카이브 로그에는 이중 로깅을 사용할 수 있으며 이 경우 다음과 같이 두 사본에 모두 BSDS 업데이트를 적용해야 합니다.

1. 새 페이지 세트 데이터 세트를 정의하고 기본 사이트의 페이지 세트 사본에 있는 데이터와 함께 로드하십시오.
2. 새 활성 로그 데이터 세트를 정의하십시오.
3. 새 BSDS 데이터 세트를 정의하고 액세스 방법 서비스 REPRO를 사용하여 가장 최근에 아카이브된 BSDS를 복사하십시오.
4. 로그 맵 인쇄 유틸리티 CSQJU004를 사용하여 이 최신 BSDS에서 정보를 인쇄하십시오. 이 BSDS가 아카이브된 시점에는 가장 최근에 아카이브된 로그가 활성 로그로서 방금 잘려 아카이브된 로그로 나타나지 않습니다. 이 로그의 STARTRBA 및 ENDRBA를 기록하십시오.
5. 로그 인벤토리 변경 유틸리티, CSQJU003을 사용하여 306 페이지의 『4』 단계에서 기록된 STARTRBA 및 ENDRBA으로 방금 복원한 BSDS에서 이 최신 아카이브 로그 데이터 세트를 등록하십시오.
6. CSQJU003의 DELETE 옵션을 사용하여 BSDS에서 모든 활성 로그 정보를 제거하십시오.
7. CSQJU003의 NEWLOG 옵션을 사용하여 BSDS에 활성 로그를 추가하십시오. STARTRBA 또는 ENDRBA는 지정하지 마십시오.
8. CSQJU003을 사용하여 재시작 제어 레코드를 BSDS에 추가하십시오. `CRESTART CREATE,ENDRBA=highrba`를 지정하십시오. 여기서 `highrba`는 사용 가능한 최신 아카이브 로그의 높은 RBA(306 페이지의 『4』 단계 참조)에 1을 더한 값입니다.

   이제 BSDS는 모든 활성 로그를 비어 있는 것으로, 아카이브된 모든 로그를 사용 가능한 것으로 또한 로그 끝 뒤에 체크포인트가 없는 것으로 설명합니다.
9. START QMGR 명령으로 큐 관리자를 재시작하십시오. 초기화 중에 다음과 같은 운영자 응답 메시지가 발행됩니다.

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
REPLY Y TO CONTINUE, N TO CANCEL
```

큐 관리자를 시작하려면 Y를 입력하십시오. 큐 관리자가 시작되고 CRESTART 문에서 지정된 ENDRBA까지 데이터를 복구합니다.

CSQJU003 및 CSQJU004 사용에 대한 정보는 IBM MQ 유틸리티 사용을 참조하십시오.

307 페이지의 그림 51은 6, 7, 8 단계에서 CSQJU003의 샘플 입력 명령문을 보여줍니다.

```
* Step 6
DELETE DSNAME=MQM2.LOGCOPY1.DS01
DELETE DSNAME=MQM2.LOGCOPY1.DS02
DELETE DSNAME=MQM2.LOGCOPY1.DS03
DELETE DSNAME=MQM2.LOGCOPY1.DS04
DELETE DSNAME=MQM2.LOGCOPY2.DS01
DELETE DSNAME=MQM2.LOGCOPY2.DS02
DELETE DSNAME=MQM2.LOGCOPY2.DS03
DELETE DSNAME=MQM2.LOGCOPY2.DS04

* Step 7
NEWLOG DSNAME=MQM2.LOGCOPY1.DS01,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS02,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS03,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY1.DS04,COPY1
NEWLOG DSNAME=MQM2.LOGCOPY2.DS01,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS02,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS03,COPY2
NEWLOG DSNAME=MQM2.LOGCOPY2.DS04,COPY2

* Step 8
CRESTART CREATE,ENDRBA=063000
```

그림 *51. CSQJU003*에 대한 샘플 입력 명령문

복구 사이트에서 채널 시작기를 재시작하기 위해 고려해야 할 사항은 다른 z/OS 이미지에서 채널 시작기를 재시작하기 위해 ARM을 사용할 때 고려해야 할 사항과 유사합니다. 자세한 정보는 315 페이지의 『IBM MQ 네트워크에서 ARM 사용』의 내용을 참조하십시오. 또한 복구 전략은 IBM MQ 제품 라이브러리와 IBM MQ를 사용하는 애플리케이션 프로그래밍 환경(예: CICS)의 복구를 다루어야 합니다.

로그 인벤토리 변경 유틸리티(CSQJU003)의 다른 함수 또한 재해 복구 시나리오에서 사용할 수 있습니다. HIGHRBA 함수는 부트스트랩 데이터 세트 내에서 기록된 가장 높은 RBA 값과 오프로드된 가장 높은 RBA 값의 업데이트를 허용합니다. CHECKPT 함수는 새 체크포인트 큐 레코드 추가 또는 BSDS의 기존 체크포인트 큐 레코드 삭제를 허용합니다.

주의: 이러한 함수는 IBM MQ 데이터의 무결성에 영향을 줄 수 있습니다. 이러한 함수는 IBM 서비스 담당자의 지시가 있는 경우에만 재해 복구 시나리오에서 사용합니다.

빠른 복사 기술

큐 관리자가 보류된 상태에서 모든 페이지 세트와 로그의 사본이 작성되는 경우에는 해당 사본이 일치 세트가 되므로 대체 사이트에서 큐 관리자를 재시작하는 데 사용할 수 있습니다. 일반적으로는 수행될 매체 복원이 거의 없으므로 큐 관리자를 훨씬 더 빠르게 재시작할 수 있습니다.

큐 관리자를 보류시키려면 SUSPEND QMGR LOG 명령을 사용하십시오. 이 명령은 버퍼 풀을 페이지 세트로 비우고 체크포인트를 생성하며 추가 로그 쓰기 활동을 중지합니다. 로그 쓰기 활동이 일시중단되면 RESUME QMGR LOG 명령을 실행할 때까지 큐 관리자가 효과적으로 보류 상태를 유지합니다. 큐 관리자가 보류 상태인 동안 페이지 세트와 로그가 복사될 수 있습니다.

FLASHCOPY 또는 SNAPSHOT과 같은 복사 도구를 사용하여 페이지 세트와 로그를 빠르게 복사함으로써 큐 관리자가 보류 상태인 시간을 최소한으로 줄일 수 있습니다.

그러나 큐 공유 그룹 내에서는 SUSPEND QMGR LOG 명령이 그렇게 좋은 솔루션이 아닐 수 있습니다. 효과적인 솔루션이 되려면 로그 사본에 모두 동일한 복구 시점이 포함되어야 하며 이는 큐 공유 그룹 내 모든 큐 관리자에 대해 SUSPEND QMGR LOG 명령이 실행되어야 함을 의미하므로 전체 큐 공유 그룹이 일정 시간 동안 보류 상태가 됩니다.

큐 공유 그룹 복구

기본 사이트에 재해가 발생하는 경우 기본 사이트의 백업 데이터 세트를 사용하여 원격 사이트에서 큐 공유 그룹을 재시작할 수 있습니다. 큐 공유 그룹을 복구하려면 큐 공유 그룹 내 모든 큐 관리자 간에 복구를 통합해야 하며 다른 자원, 주로 Db2와 통합되어야 합니다. 이 절에서는 이러한 태스크를 자세히 설명합니다.

- CF 구조 매체 복구
- 기본 사이트에서 큐 공유 그룹 백업
- 대체 사이트에서 큐 공유 그룹 복구

CF 구조 매체 복원

공유 큐에서 지속 메시지를 보관하는 데 사용되는 CF 구조의 매체 복원을 수행하려면 로그된 업데이트를 적용하여 정방향으로 복구될 수 있는 매체의 백업이 있어야 합니다. MQSC BACKUP CFSTRUCT 명령을 사용하여 CF 구조의 백업을 주기적으로 생성하십시오. 공유 큐에 대한 모든 업데이트(MQGET 및 MQPUT)는 업데이트가 수행되는 큐 관리자의 로그에 기록됩니다. CF 구조의 매체 복원을 수행하려면 해당 CF 구조를 사용한 **모든** 큐 관리자의 로그에서 해당 백업에 로그된 업데이트를 적용해야 합니다. MQSC RECOVER CFSTRUCT 명령을 사용하는 경우, IBM MQ는 관련 큐 관리자의 로그를 자동으로 병합하고 업데이트 사항을 최신 백업에 적용합니다.

CF 구조 백업은 BACKUP CFSTRUCT 명령을 처리한 큐 관리자의 로그에 기록되므로 수집 후 대체 사이트로 전송될 추가 데이터 세트는 없습니다.

기본 사이트에서 큐 공유 그룹 백업

기본 사이트에서는 일관적인 백업 세트를 정기적으로 설정해야 하며 재해 발생 시 대체 사이트에서 큐 공유 그룹을 재빌드하는 데 사용될 수 있습니다. 단일 큐 관리자의 경우, 임의 시점(일반적으로 원격 사이트에서 사용 가능한 로그 끝)까지 복구될 수 있습니다. 그러나 지속 메시지가 공유 큐에 저장된 경우에는 큐 공유 그룹 내 모든 큐 관리자의 로그가 병합되어야 공유 큐를 복구할 수 있습니다. 큐 공유 그룹의 큐 관리자가 큐에서 업데이트(MQPUT 또는 MQGET)를 수행했을 수도 있기 때문입니다.

큐 공유 그룹을 복구하려면 모든 큐 관리자의 로그 데이터 로그 범위에 해당하는 특정 시점을 설정해야 합니다. 그러나 로그에서는 매체를 **정방향**으로만 복구할 수 있으므로 이 시점은 BACKUP CFSTRUCT 명령이 실행되고 페이지 세트 백업이 수행된 이후여야 합니다. 일반적으로 복구 시점은 평일 자정이나 주말에 해당할 수 있습니다.

다음 다이어그램은 큐 공유 그룹의 두 관리자에 대한 시간표를 보여줍니다. 각 큐 관리자마다 페이지 세트의 퍼지 백업이 생성됩니다(방법 2: 퍼지 백업 참조). 큐 관리자 A에서는 BACKUP CFSTRUCT 명령이 실행됩니다. 그 뒤에 각 큐 관리자에서 ARCHIVE LOG 명령이 실행되어 활성 로그를 잘라 큐 관리자로부터 오프라인인 매체에 복사합니다. 이 활성 로그는 대체 사이트로 전송될 수 있습니다. 로그 끝은 ARCHIVE LOG 명령이 실행된 시점을 식별하므로 일반적으로 대체 사이트에서 사용 가능한 로그 데이터의 범위를 표시합니다. 복구 시점은 페이지 세트 또는 CF 구조 백업의 끝과 대체 사이트에서 사용 가능한 가장 빠른 로그 끝 사이여야 합니다.

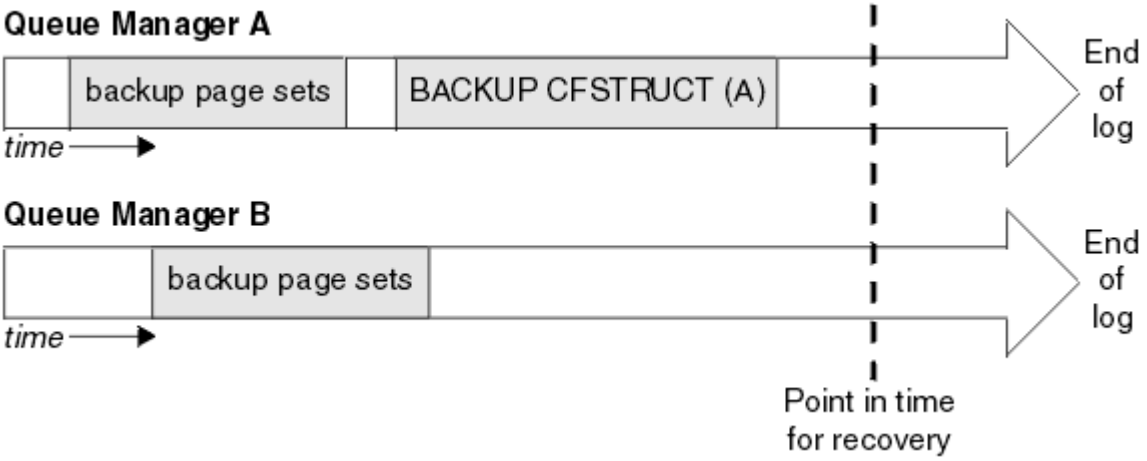


그림 52. 큐 공유 그룹 내 두 개의 큐 관리자를 위한 복구 시점

IBM MQ는 Db2의 테이블에 CF 구조 백업과 연관된 정보를 기록합니다. 요구사항에 따라, IBM MQ의 복구 시점과 Db2의 복구 시점을 통합할 수도 있고 또는 BACKUP CFSTRUCT 명령이 완료된 후 IBM MQ CSQ.ADMIN_B_STRBACKUP 테이블의 사본을 작성하는 것만으로 충분할 수 있습니다.

복구를 준비하려면 다음을 수행하십시오.

1. 큐 공유 그룹의 각 큐 관리자에 대한 페이지 세트 백업을 작성하십시오.
2. RECOVER(YES) 속성과 함께 각 CF 구조에 대한 BACKUP CFSTRUCT 명령을 실행하십시오. 이러한 명령은 단일 큐 관리자에서 또는 워크로드 균형을 맞추기 위해 큐 공유 그룹 내 다른 큐 관리자로부터 실행할 수 있습니다.
3. 모든 백업이 완료되면 ARCHIVE LOG 명령을 실행하여 활성 로그를 전환하고 큐 공유 그룹 내 각 큐 관리자의 로그 및 BSDS의 사본을 작성하십시오.
4. 큐 공유 그룹 내 모든 큐 관리자의 페이지 세트 백업, 아카이브된 로그, 아카이브된 BSDS와 선택한 Db2 백업 정보를 오프사이트로 전송하십시오.

대체 사이트에서 큐 공유 그룹 복구

큐 공유 그룹을 복구하려면 다음과 같이 환경을 준비해야 합니다.

1. 큐 공유 그룹을 설치할 때 연습 시작에서 커플링 기능에 이전 정보가 있는 경우 먼저 이를 정리해야 합니다.

   참고: 커플링 기능에 오래된 정보가 없는 경우에는 이 단계를 생략할 수 있습니다.

   a. 다음 z/OS 명령을 입력하여 이 큐 공유 그룹의 CF 구조를 표시하십시오.

   ```
   D XCF,STRUCTURE,STRNAME= qsgname
   ```

   b. 큐 공유 그룹 이름으로 시작되는 모든 구조의 경우, z/OS 명령 SETXCF FORCE CONNECTION을 사용하여 해당 구조와의 연결 끊기를 강제 실행하십시오.

   ```
   SETXCF FORCE,CONNECTION,STRNAME= strname,CONNAME=ALL
   ```

   c. 각 구조에 다음 명령을 사용하여 모든 CF 구조를 삭제하십시오.

```
SETXCF FORCE,STRUCTURE,STRNAME= strname
```

2. Db2 시스템과 데이터 공유 그룹을 복원하십시오.
3. 중요 사이트에서 생성된 최신 구조 백업에 대한 정보가 포함되도록 CSQ.ADMIN_B_STRBACKUP 테이블을 복구하십시오.

   참고: STRBACKUP 테이블에는 최신 구조 백업 정보가 포함되어야 합니다. 오래된 구조 백업 정보의 경우 최신 DISPLAY USAGE TYPE(DATASET) 명령으로 제공된 정보의 결과로 제거한 데이터 세트가 필요할 수 있으며 이는 복구된 CF 구조에 정확한 정보가 포함되지 않게 된다는 사실을 의미합니다.
4. 큐 공유 그룹의 모든 큐 관리자에 대해 CSQ5PQSG 유틸리티의 ADD QMGR 명령을 실행하십시오. 각 큐 관리자의 XCF 그룹 항목이 복원됩니다.

   이 시나리오에서 유틸리티를 실행할 때는 다음 메시지가 정상적인 것입니다.

```
CSQU566I Unable to get attributes for admin structure, CF not found
or not allocated
CSQU546E Unable to add QMGR queue_manager_name entry,
already exists in DB2 table CSQ.ADMIN_B_QMGR
CSQU148I CSQ5PQSG Utility completed, return code=4
```

큐 공유 그룹의 큐 관리자를 복구하려면 다음을 수행하십시오.

1. 새 페이지 세트 데이터 세트를 정의하고 기본 사이트의 페이지 세트 사본에 있는 데이터와 함께 로드하십시오.
2. 새 활성 로그 데이터 세트를 정의하십시오.
3. 새 BSDS 데이터 세트를 정의하고 액세스 방법 서비스 REPRO를 사용하여 가장 최근에 아카이브된 BSDS를 복사하십시오.
4. 로그 맵 인쇄 유틸리티 CSQJU004를 사용하여 이 최신 BSDS에서 정보를 인쇄하십시오. 이 BSDS가 아카이브된 시점에는 가장 최근에 아카이브된 로그가 활성 로그로서 방금 잘려 아카이브된 로그로 나타나지 않습니다. 이 로그의 STARTRBA, STARTLRSN, ENDRBA, ENDLRSN 값을 기록하십시오.
5. 로그 인벤토리 변경 유틸리티, CSQJU003을 사용하여 310 페이지의 『4』 단계에서 기록된 값으로 방금 복원한 BSDS에서 이 최신 아카이브 로그 데이터 세트를 등록하십시오.
6. CSQJU003의 DELETE 옵션을 사용하여 BSDS에서 모든 활성 로그 정보를 제거하십시오.
7. CSQJU003의 NEWLOG 옵션을 사용하여 BSDS에 활성 로그를 추가하십시오. STARTRBA 또는 ENDRBA는 지정하지 마십시오.
8. 큐 공유 그룹의 *recoverylrsn*을 계산하십시오. *recoverylrsn* 는 큐 공유 그룹에 있는 모든 큐 관리자의 ENDLRSNs (310 페이지의 『4』 단계에서 기록됨), -1을 뺀 가장 낮은 ENDLRSNs입니다. 예를 들어, 큐 공유 그룹에 두 개의 큐 관리자가 있고 이들 중 하나에 대한 ENDLRSN이 B713 3C72 22C5이고 다른 큐 관리자가 B713 3D45 2123이면, *recoverylrsn* 은 B713 3C72 22C4입니다.
9. CSQJU003을 사용하여 재시작 제어 레코드를 BSDS에 추가하십시오. 다음을 지정하십시오.

```
CRESTART CREATE,ENDLRSN= recoverylrsn
```

   여기서 *recoverylrsn*은 단계 310 페이지의 『8』에서 기록한 값입니다.

   이제 BSDS는 모든 활성 로그를 비어 있는 것으로, 아카이브된 모든 로그를 사용 가능한 것으로 또한 로그 끝 뒤에 체크포인트가 없는 것으로 설명합니다.

   큐 공유 그룹 내 각 큐 관리자의 BSDS에 CRESTART 레코드를 추가해야 합니다.
10. START QMGR 명령으로 큐 공유 그룹의 각 큐 관리자를 재시작하십시오. 초기화 중에 다음과 같은 운영자 응답 메시지가 발행됩니다.

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
REPLY Y TO CONTINUE, N TO CANCEL
```

큐 관리자를 시작하려면 Y로 응답하십시오. 큐 관리자가 시작되고 CRESTART 문에서 지정된 ENDRBA까지 데이터를 복구합니다.

IBM WebSphere MQ 7.0.1 이상의 경우, 처음 시작된 큐 관리자가 큐 공유 그룹의 다른 멤버 및 자체 관리 구조 파티션을 재빌드할 수 있으므로 이 단계에서는 더 이상 큐 공유 그룹의 각 큐 관리자를 재시작하지 않아도 됩니다.

11. 모든 큐 관리자의 관리 구조 데이터가 재빌드되면 각 CF 애플리케이션 구조에 RECOVER CFSTRUCT 명령을 실행하십시오.

    단일 큐 관리자의 모든 구조에 대해 RECOVER CFSTRUCT 명령을 실행하는 경우에는 로그 병합 프로세스가 한 번만 수행되므로 각 CF 구조마다 다른 큐 관리자에서 명령을 실행하는 것보다 빠릅니다. 이 경우 각 큐 관리자가 로그 병합 단계를 수행해야 합니다.

큐 공유 그룹에서 조건부 재시작 처리가 사용되는 경우에는 IBM WebSphere MQ 7.0.1 이상 큐 관리자가 피어 관리 재빌드를 수행하여 피어 BSDS에 자체와 동일한 CRESTART LRSN이 포함되는지 확인합니다. 이는 재빌드된 관리 구조의 무결성을 확인하기 위해서입니다. 따라서 QSG에서 다른 피어를 재시작해야 하며 이를 통해 다음에 그룹의 멤버가 무조건적으로 재시작되기 전에 자체 CRESTART 정보를 처리할 수 있습니다.

디스크 미러링 사용

많은 설치에서는 이제 IBM 메트로 미러(이전 PPRC)와 같은 디스크 미러링 기술을 사용하여 대체 사이트에서 데이터 세트의 동기 사본을 작성합니다. 이러한 경우 많은 세부 단계가 불필요하게 됩니다. 대체 사이트의 IBM MQ 페이지 세트와 로그가 기본 사이트의 해당 페이지 세트 및 로그와 사실상 동일하기 때문입니다. 이러한 기술을 사용되는 경우 대체 사이트에서 큐 공유 그룹을 재시작하기 위한 단계는 다음과 같이 요약될 수 있습니다.

- 대체 사이즈에서 IBM MQ CF 구조를 지우십시오. 이러한 구조에는 일반적으로 이전 재해 복구 연습에서의 나머지 정보가 포함되어 있습니다.
- IBM MQ 큐 공유 그룹이 사용하는 데이터베이스에서 모든 테이블과 Db2 시스템을 복원하십시오.
- 큐 관리자를 재시작하십시오. IBM WebSphere MQ 7.0.1 이전의 경우에는 큐 공유 그룹에 정의된 각 큐 관리자를 재시작해야 합니다. 각 큐 관리자가 관리 구조의 자체 파티션을 큐 관리자 재시작 중에 복구하기 때문입니다. 각 큐 관리자가 재시작된 후 해당 '홈' LPAR에 없는 큐 관리자는 다시 종료될 수 있습니다. IBM WebSphere MQ 7.0.1 이상의 경우, 처음 시작된 큐 관리자가 큐 공유 그룹의 다른 멤버 및 자체 관리 구조 파티션을 재빌드하므로 더 이상 큐 공유 그룹의 각 큐 관리자를 재시작하지 않아도 됩니다.
- 관리 구조가 재빌드된 후 애플리케이션 구조를 복구하십시오.

큐 관리자 재초기화

큐 관리자가 비정상적으로 종료되면 큐 관리자를 재시작하지 못할 수 있습니다. 페이지 세트 또는 로그가 손실, 절단 또는 손상되었기 때문일 수 있습니다. 이 경우 큐 관리자를 다시 초기화해야 합니다(콜드 스타트 수행).

주의

콜드 스타트는 큐 관리자를 다른 방법으로 재시작할 수 없는 경우에만 수행하십시오. 콜드 스타트를 수행하면 큐 관리자와 오브젝트 정의를 복구할 수 있습니다. 메시지 데이터는 복구할 수 **없습니다**. 이 작업을 수행하기 전에 이 주제에 설명된 기타 재시작 시나리오 중 해당되는 시나리오가 없는지 확인하십시오.

재시작하면 모든 IBM MQ 오브젝트가 정의되고 사용할 수 있지만 메시지 데이터는 없습니다.

참고: 큐 관리자가 클러스터의 일부인 경우 다시 초기화하지 마십시오. 우선 클러스터에서 큐 관리자를 제거(클러스터의 다른 큐 관리자에서 RESET CLUSTER 명령 사용)하고 다시 초기화한 다음 새 큐 관리자로 클러스터에 다시 도입해야 합니다.

다시 초기화하는 동안 큐 관리자 ID(QMID)가 변경되므로 이전 큐 관리자 ID를 사용하는 클러스터 오브젝트를 클러스터에서 제거해야 합니다.

추가 정보는 다음 절을 참조하십시오.

- 큐 공유 그룹에 없는 큐 관리자 재초기화
- 큐 공유 그룹의 큐 관리자 재초기화

큐 공유 그룹에 없는 큐 관리자 재초기화

큐 관리자를 재초기화하려면 다음 프로시저를 따르십시오.

1. 큐 관리자를 재시작할 때 사용할 오브젝트 정의문을 준비하십시오. 이 작업을 수행하려면 다음 중 하나를 수행하십시오.
   - 페이지 세트 0을 사용할 수 있는 경우, CSQUTIL SDEFS 함수를 사용하십시오(IBM MQ 정의 명령 목록 생성 참조). 모든 오브젝트 유형(인증 정보 오브젝트, CF 구조, 채널, 이름 목록, 프로세스, 큐, 스토리지 클래스)의 정의를 가져와야 합니다.
   - 페이지 세트 0을 사용할 수 없는 경우에는 오브젝트 정의를 마지막으로 백업한 시점부터 정의를 사용하십시오.
2. 큐 관리자 데이터 세트를 재정의하십시오(312 페이지의 『1』 단계를 완료할 때까지는 재정의하지 않음).

   자세한 정보는 부트스트랩 및 로그 데이터 세트 작성과 페이지 세트 정의를 참조하십시오.
3. 새로 정의되고 초기화된 로그 데이터 세트, BSDS와 페이지 세트를 사용하여 큐 관리자를 재시작하십시오. 312 페이지의 『1』 단계에서 작성한 오브젝트 정의 입력 명령문을 CSQINP2 초기화 입력 데이터 세트의 입력으로 사용하십시오.

큐 공유 그룹에서 큐 관리자 재초기화

큐 공유 그룹에서 큐 관리자를 재초기화하는 작업은 더 복잡합니다. 페이지 세트 또는 로그 문제점으로 인해 하나 이상의 큐 관리자를 재초기화해야 할 수도 있지만 처리해야 할 커플링 기능 또는 Db2 관련 문제점이 있을 수도 있습니다. 이 경우 다음과 같은 여러 가지 대안이 있습니다.

콜드 스타트
: 전체 큐 공유 그룹을 재초기화하려면 모든 커플링 기능 구조를 강제 실행하고 Db2에서 큐 공유 그룹에 대한 모든 오브젝트 정의를 지워야 하며 로그와 BSDS를 삭제 또는 재정의하고 큐 공유 그룹에서 모든 큐 관리자의 페이지 세트를 형식화해야 합니다.

공유 정의 보유
: 로그 및 BSDS를 삭제하거나 재정의하고, 큐 공유 그룹에 있는 모든 큐 관리자의 페이지 세트를 형식화하며, 커플링 기능 구조를 모두 강제 실행합니다. 재시작 시 모든 메시지가 삭제됩니다. 큐 관리자는 Db2 데이터베이스에 계속 존재하는 GROUP 오브젝트에 해당하는 COPY 오브젝트를 재작성합니다. 공유 큐는 계속 존재하며 사용할 수 있습니다.

단일 큐 관리자 재초기화
: 로그와 BSDS를 삭제하거나 재정의하고, 단일 큐 관리자의 페이지 세트를 형식화합니다(모든 개인용 오브젝트와 메시지가 삭제됨). 재시작 시 큐 관리자는 Db2 데이터베이스에 계속 존재하는 GROUP 오브젝트에 해당하는 COPY 오브젝트를 재작성합니다. 공유 큐는 메시지와 같이 계속 존재하며 사용할 수 있습니다.

큐 공유 그룹의 시점 복구
: 대체 사이트 재해 복구 시나리오입니다.

  공유 오브젝트는 Db2 복구로 달성된 시점으로 복구됩니다(A Db2 시스템 실패의 설명 참조). 각 큐 관리자는 대체 사이트에서 사용 가능한 백업 사본에서 달성할 수 있는 시점으로 복구할 수 있습니다.

  지속 메시지는 큐 공유 그룹에서 사용할 수 있으며 MQSC RECOVER CFSTRUCT 명령을 사용하여 복구할 수 있습니다. 이 명령은 실패한 시간으로 복구합니다. 그러나 비지속 공유 큐 메시지는 복구하지 않습니다. CSQUTIL 유틸리티 프로그램의 COPY 기능을 사용하여 개별적으로 백업 사본을 작성하지 않은 경우 손실됩니다.

  다른 큐 관리자의 로컬 오브젝트(실제로 복구되는 대상) 간에는 상호 의존성이 없으므로 각 큐 관리자를 동일한 시점으로 복원하려도 시도하지 않아도 되며 재시작 시 Db2와의 큐 관리자 재동기화를 통해 큐 관리자별로 필요에 따라 COPY 오브젝트를 작성 또는 삭제합니다.

z/OS 자동 재시작 관리자(ARM) 사용

이 주제를 통해 ARM을 사용하여 큐 관리자를 자동으로 재시작할 수 있는 방법을 이해할 수 있습니다.

이 절에는 다음 주제에 대한 정보가 포함되어 있습니다.

ARM의 개념

z/OS 자동 재시작 관리자(ARM)는 큐 관리자의 가용성을 향상시킬 수 있는 z/OS 복구 기능입니다. 작업 또는 태스크가 실패하거나 작업 또는 태스크가 실행 중인 시스템이 실패하는 경우 ARM이 운영자 개입 없이 작업 또는 태스크를 재시작할 수 있습니다.

큐 관리자 또는 채널 시작기가 실패한 경우 ARM은 이를 동일한 z/OS 이미지에서 재시작합니다. z/OS에서 관련 서브시스템과 애플리케이션의 전체 그룹이 실패하는 경우에는 sysplex 내 다른 z/OS 이미지에서 사전 정의된 순서대로 ARM이 모든 실패 시스템을 자동으로 재시작할 수 있습니다. 이를 시스템 간 재시작이라고 합니다.

예외적인 상황에서만 ARM으로 채널 시작기를 재시작하십시오. 큐 관리자를 ARM으로 시작하는 경우에는 CSQINP2 초기화 데이터 세트에서 채널 시작기를 재시작하십시오(315 페이지의 『IBM MQ 네트워크에서 ARM 사용』 참조).

ARM을 사용하여 z/OS 장애 시에 sysplex 내의 다른 z/OS 이미지에서 큐 관리자를 재시작할 수 있습니다. 다른 z/OS 이미지에서 IBM MQ ARM 다시 시작의 네트워크 영향은 315 페이지의 『IBM MQ 네트워크에서 ARM 사용』에 설명되어 있습니다.

자동 재시작을 사용하려면 다음을 수행하십시오.

- ARM 커플 데이터 세트를 설치하십시오.
- z/OS가 *ARM* 정책에서 수행하려는 자동 재시작 조치를 정의하십시오.
- ARM 정책을 시작하십시오.

또한 시작 시 IBM MQ가 ARM에 등록해야 합니다(자동으로 진행).

참고: 큐 관리자를 다른 z/OS 이미지에서 자동으로 재시작하려면 sysplex 범위의 고유한 4자 서브시스템 이름과 함께 해당 큐 관리자가 재시작될 수 있는 각 z/OS 이미지에서 모든 큐 관리자를 서브시스템으로 정의해야 합니다.

ARM 커플 데이터 세트

ARM에 필요한 커플 데이터 세트를 정의하고 ARM 지원을 원하는 큐 관리자를 시작하기 전에 해당 데이터 세트가 온라인이고 활성 상태인지 확인하십시오. 큐 관리자 시작 시 커플 데이터 세트를 사용할 수 없는 경우 IBM MQ 자동 ARM 등록이 실패합니다. 이러한 경우 IBM MQ는 커플 데이터 세트가 없는 경우 ARM 지원을 원하지 않음을 의미한다고 가정하여 초기화가 계속 진행됩니다.

ARM 커플 데이터 세트에 대한 정보는 *z/OS MVS Sysplex* 설정 매뉴얼을 참조하십시오.

ARM 정책

자동 재시작 관리자 정책은 큐 관리자 재시작을 제어할 수 있는 ARM 기능을 제어하는 사용자 정의 규칙입니다.

ARM 기능은 사용자 정의 *ARM* 정책으로 제어됩니다. ARM으로 재시작되는 큐 관리자 인스턴스를 실행하는 각 z/OS 이미지는 활성 ARM 정책으로 ARM 커플 데이터 세트에 연결되어야 합니다.

IBM은 기본 ARM 정책을 제공합니다. 새 정책을 정의하거나, z/OS와 함께 제공되는 관리 데이터 유틸리티(IXCMIAPU)를 사용하여 정책 기본값을 대체할 수 있습니다. *z/OS MVS Sysplex* 설정 매뉴얼은 이 유틸리티에 대해 설명하고 ARM 정책 정의 방법에 대한 자세한 내용을 다룹니다.

314 페이지의 그림 53은 ARM 정책의 예를 보여줍니다. 이 샘플 정책은 큐 관리자가 실패했거나 전체 시스템이 실패한 경우 SYSPLEX 내부에서 큐 관리자를 재시작합니다.

```
//IXCMIAPU EXEC PGM=IXCMIAPU,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN  DD *
DATA TYPE(ARM)
DEFINE POLICY NAME(ARMPOL1) REPLACE(YES)
RESTART_GROUP(DEFAULT)
ELEMENT(*)
RESTART_ATTEMPTS(0)  /* Jobs not to be restarted by ARM  */
RESTART_GROUP(GROUP1)
ELEMENT(SYSMQMGRMQ*)  /* These jobs to be restarted by ARM */
/*
```

그림 *53.* 샘플 *ARM* 정책

자세한 정보는 다음을 참조하십시오.

- ARM 정책 정의
- ARM 정책 활성화
- ARM에 등록

ARM 정책 정의

ARM 정책은 다음과 같이 설정하십시오.

- 각 큐 관리자 인스턴스마다 해당 큐 관리자 인스턴스에 연결되는 CICS 또는 IMS 서브시스템도 포함하는 RESTART_GROUP을 정의하십시오. 서브시스템 이름 지정 규칙을 사용하는 경우, 요소 이름에 '?' 및 '*' 와일드 카드 문자를 사용하여 최소 정의 노력으로 RESTART_GROUP을 정의할 수 있습니다.
- 채널 시작기가 실패하고 z/OS 이미지가 실패하지 않은 경우에만 재시작됨을 표시하는 TERMTYPE(ELEMTERM)을 채널 시작기에 대해 지정하십시오.
- 큐 관리자가 실패했거나 z/OS 이미지가 실패한 경우 재시작됨을 표시하는 TERMTYPE(ALLTERM)을 큐 관리자에 대해 지정하십시오.
- 큐 관리자와 채널 시작기에 모두 RESTART_METHOD(BOTH, PERSIST)를 지정하십시오. 그러면 마지막 시작 중에 시스템 기호를 해석한 후 ARM이 저장한 JCL을 사용하여 재시작하도록 ARM에 지시합니다. 또한 개별 요소가 실패했는지 또는 z/OS 이미지가 실패했는지 여부에 관계 없이 이를 수행하도록 ARM에 지시합니다.
- 다른 모든 ARM 정책 옵션에는 기본값을 승인하십시오.

ARM 정책 활성화

자동 재시작 관리 정책을 시작하려면 다음 z/OS 명령을 실행하십시오.

```
SETXCF START,POLICY,TYPE=ARM,POLNAME= mypol
```

정책이 시작되면 ARM 커플 데이터 세트에 연결된 모든 시스템이 동일한 활성 정책을 사용합니다.

자동 재시작을 사용하지 않으려면 SETXCF STOP 명령을 사용하십시오.

ARM에 등록

IBM MQ는 ARM 가용성에 따라 큐 관리자 시작 동안 *ARM* 요소로서 자동으로 등록됩니다. 반대 요청이 없는 경우 종료 단계에서 등록을 취소합니다.

시작 시 큐 관리자는 ARM 사용 가능 여부를 판별합니다. 사용 가능한 경우 IBM MQ는 SYSMQMGR *ssid*라는 이름을 사용하여 등록합니다. 여기서 *ssid*는 4자 길이의 큐 관리자 이름이고 SYSMQMGR은 요소 유형입니다.

STOP QMGR MODE(QUIESCE) 및 STOP QMGR MODE(FORCE) 명령은 ARM에서 큐 관리자 등록을 취소합니다(시작 시 ARM에 등록된 경우). 그러면 ARM이 이 큐 관리자를 재시작하지 않습니다. STOP QMGR MODE(RESTART) 명령은 ARM에서 큐 관리자를 등록 취소하지 않으므로 즉시 자동으로 재시작할 수 있습니다.

각각의 채널 시작기 주소 공간은 ARM 사용 가능 여부를 판별하며 사용 가능한 경우 요소 이름 SYSMQCH *ssid*를 사용하여 등록합니다. 여기서 *ssid*는 큐 관리자 이름이고 SYSMQCH는 요소 유형입니다.

채널 시작기는 정상적으로 중지되는 경우 항상 ARM에서 등록 취소되며 비정상적으로 종료되는 경우에만 등록 상태를 유지합니다. 큐 관리자가 실패하면 채널 시작기가 항상 등록 취소됩니다.

IBM MQ 네트워크에서 *ARM* 사용

큐 관리자가 재시작될 때 채널 시작기 및 연관된 리스너가 자동으로 시작될 수 있도록 큐 관리자를 설정할 수 있습니다.

LU 6.2 및 TCP/IP 통신 프로토콜 모두에 대해 동일한 z/OS 이미지에서 큐 관리자를 완전 자동으로 재시작하려면 다음을 수행하십시오.

- 적합한 START LISTENER 명령을 CSQINPX 데이터 세트에 추가하여 리스너를 자동으로 시작하십시오.
- 적합한 START CHINIT 명령을 CSQINP2 데이터 세트에 추가하여 채널 시작기를 자동으로 시작하십시오.

TCP/IP 또는 LU6.2를 사용하여 큐 관리자를 재시작하려면 다음을 참조하십시오.

- 315 페이지의 『TCP/IP를 사용하여 다른 z/OS 이미지에서 재시작』
- 316 페이지의 『LU 6.2를 사용하여 다른 z/OS 이미지에서 재시작』

CSQINP2 및 CSQINPX 데이터 세트에 대한 정보는 태스크 13: 초기화 입력 데이터 세트 사용자 정의를 참조하십시오.

TCP/IP를 사용하여 다른 z/OS 이미지에서 재시작

TCP/IP를 통신 프로토콜로 사용하고 가상 IP 주소를 사용하는 경우, 해당 큐 관리자에 연결되는 채널이 변경사항 없이 다시 연결될 수 있도록 허용하여 다른 z/OS 이미지에서 복구되도록 구성할 수 있습니다. 또는 클러스터를 사용하는 경우 또는 WLM 동적 DNS(Domain Name System) 논리 그룹 이름을 사용하여 큐 공유 그룹에 연결하는 경우 큐 관리자를 다른 z/OS 이미지로 이동한 후 TCP/IP 주소를 재할당할 수 있습니다.

- 클러스터링을 사용하는 경우
- 큐 공유 그룹에 연결하는 경우

클러스터링을 사용하는 경우

z/OS ARM은 동일한 SYSPLEX 내 다른 z/OS 이미지에서 큐 관리자를 재시작하여 시스템 장애에 대응합니다. 이 시스템은 원래 z/OS 이미지와 다른 TCP/IP 주소를 갖습니다. 다음은 IBM MQ 클러스터를 사용하여 ARM 재시작으로 다른 z/OS 이미지로 이동된 후 큐 관리자의 TCP/IP 주소를 재할당하는 방법을 설명합니다.

클라이언트 큐 관리자가 큐 관리자 실패를 채널 실패로 감지하면 클러스터 전송 큐에서 적합한 메시지를 대상 클러스터 큐의 다른 인스턴스를 호스트하는 다른 서버 큐 관리자에 재할당하여 대응합니다. 그러나 연관 관계 제한조건으로 인해 원래 서버에 바인딩된 메시지 또는 배치 종료 처리 중에 서버 큐 관리자가 실패하여 인다우트(in-doubt) 상태가 된 메시지는 재할당할 수 없습니다. 이러한 메시지를 처리하려면 다음을 수행하십시오.

1. 각 z/OS 큐 관리자에 다른 클러스터 수신자 채널 이름과 다른 TCP/IP 포트를 할당하십시오. 각 큐 관리자에는 두 시스템이 z/OS 이미지에서 단일 TCP/IP 스택을 공유할 수 있도록 다른 포트가 필요합니다. 그 중 하나는 원래 해당 z/OS 이미지에서 실행 중인 큐 관리자이고 나머지 하나는 ARM이 시스템 장애 이후 해당 z/OS 이미지에서 재시작하게 될 큐 관리자입니다. ARM이 임의 z/OS 이미지에서 큐 관리자를 재시작할 수 있도록 각 z/OS 이미지에 각 포트를 구성하십시오.
2. 각 큐 관리자와 z/OS 이미지 조합에 대해 채널 시작기 시작 중에 참조될 다른 채널 시작기 명령 입력 파일(CSQINPX)을 작성하십시오.

   각 CSQINPX 파일에는 해당 큐 관리자의 고유한 START LISTENER PORT(port) 명령과 해당 큐 관리자와 z/OS 이미지 조합에 고유한 클러스터 수신자 채널에 대한 ALTER CHANNEL 명령이 포함되어야 합니다.

ALTER CHANNEL 명령은 연결 이름을 재시작되는 z/OS 이미지의 TCP/IP 이름으로 설정해야 합니다. 여기에는 연결 이름의 일부로서 재시작된 큐 관리자의 고유한 포트 번호가 포함되어야 합니다.

각 큐 관리자의 시작 JCL에는 이 CSQINPX 파일의 고정 데이터 세트 이름을 가질 수 있으며 각 z/OS 이미지는 비공유 DASD 볼륨에 각 CSQINPX 파일의 다른 버전을 갖고 있어야 합니다.

ARM 재시작이 발생하면 IBM MQ가 클러스터 저장소에 변경된 채널 정의를 알려주며 해당 정의는 서버 큐 관리자에서 관심을 표현한 모든 클라이언트 큐 관리자에 발행됩니다.

클라이언트 큐 관리자는 서버 큐 관리자 실패를 채널 실패로 간주하여 실패한 채널 재시작을 시도합니다. 클라이언트 큐 관리자가 새 서버 연결 이름을 알게 되면 채널 재시작이 클라이언트 큐 관리자를 재시작된 서버 큐 관리자에 다시 연결합니다. 그러면 클라이언트 큐 관리자가 해당 메시지를 재동기화하고 클라이언트 큐 관리자의 전송 큐에 있는 인다우트(in-doubt) 메시지를 해석하며 정상 처리가 계속 진행될 수 있습니다.

큐 공유 그룹에 연결할 때

TCP/IP 동적 DNS(Domain Name System) 논리 그룹 이름을 통해 큐 공유 그룹에 연결하는 경우, 채널 정의의 연결 이름은 물리적 시스템의 호스트 이름 또는 IP 주소가 아닌 큐 공유 그룹의 논리 그룹 이름을 지정합니다. 이 채널이 시작되면 동적 DNS에 연결된 다음 큐 공유 그룹에 있는 큐 관리자 중 하나에 연결됩니다. 이 프로세스는 큐 공유 그룹을 사용한 IBM MQ for z/OS에 대한 통신 설정에 설명되어 있습니다.

자주 발생하지는 않지만 이미지 실패의 경우 다음 중 하나가 발생합니다.

- 실패한 이미지의 큐 관리자가 SYSPLEX에서 실행 중인 동적 DNS에서 등록 취소됩니다. 채널은 RETRYING 상태를 입력하여 연결 장애에 대응하고 SYSPLEX에서 실행 중인 동적 DNS에 연결합니다. 동적 DNS는 나머지 이미지에서 계속 실행 중인 큐 공유 그룹의 나머지 멤버 중 하나에 인바운드 요청을 할당합니다.
- 큐 공유 그룹에 활성 상태인 다른 큐 관리자가 없고 ARM이 큐 관리자와 채널 시작기를 서로 다른 이미지에서 재시작하는 경우 그룹 리스너는 이 새 이미지에서 동적 DNS에 등록합니다. 이는 채널의 연결 이름 필드에 있는 논리 그룹 이름이 동적 DNS에 연결된 다음 현재 다른 이미지에서 실행 중인 동일 큐 관리자에 연결됨을 의미합니다. 채널 정의에 대한 변경사항은 필요하지 않습니다.

이러한 유형의 복구가 발생하려면 다음 사항에 유의해야 합니다.

- z/OS에서, 동적 DNS는 SYSPLEX 내 z/OS 이미지 중 하나에서 실행됩니다. 이 이미지가 실패하는 경우, 기본 이름 서버의 대체 역할을 하는 보조 이름 서버가 SYSPLEX에서 활성화되도록 동적 DNS가 구성되어야 합니다. 기본 및 보조 동적 DNS 서버에 대한 정보는 *OS/390® SecureWay CS IP* 구성 매뉴얼에 나와 있습니다.
- 이 z/OS 이미지에서 사용 불가능한 특정 IP 주소에서 TCP/IP 그룹 리스너가 시작되었을 수 있습니다. 이러한 경우 새 이미지의 다른 IP 주소에서 리스너가 시작되어야 할 수 있습니다. 가상 IP 주소를 사용하는 경우, START LISTENER 명령에 대한 변경사항이 필요하지 않도록 다른 z/OS 이미지에서 복구되도록 구성할 수 있습니다.

LU 6.2를 사용하여 다른 z/OS 이미지에서 재시작

LU 6.2 통신 프로토콜만 사용하는 경우에는 다음 프로시저를 수행하여 SYSPLEX 내 다른 z/OS 이미지에서 큐 관리자를 자동으로 재시작한 후 네트워크가 다시 연결될 수 있도록 설정하십시오.

- SYSPLEX 내에서 각 큐 관리자를 고유한 서브시스템 이름으로 정의하십시오.
- SYSPLEX 내에서 각 채널 시작기를 고유한 LUNAME으로 정의하십시오. 이는 큐 관리자 속성과 START LISTENER 명령에서 둘 다 지정됩니다.

  참고: LUNAME은 APPC 사이드 테이블에서 항목에 이름을 지정하고 그런 다음 여기에서 실제 LUNAME으로 맵핑됩니다.
- SYSPLEX 내에서 각 z/OS 이미지가 참조하는 공유 APPC 측 테이블을 설정하십시오. 여기에는 각 채널 시작기의 LUNAME의 항목이 포함되어야 합니다. 해당 정보는 *MVS 계획: APPC/MVS* 관리 매뉴얼을 참조하십시오.
- 해당 채널 시작기의 APPC 측 테이블 항목을 활성화하기 위해 LUADD를 포함하도록 SYSPLEX 내 각 채널 시작기에 대해 SYS1.PARMLIB의 APPCPM *xx* 멤버를 설정하십시오. 이러한 멤버는 각 z/OS 이미지가 공유해야 합니다. 해당 SYS1.PARMLIB 멤버는 z/OS 명령 SET APPC= *xx*로 활성화됩니다. 이 명령은 다음 텍스트에 설명된 대로 다른 z/OS 이미지에서 큐 관리자와 해당 채널 시작기를 ARM 재시작하는 동안 자동으로 실행됩니다.

- LU62ARM 큐 관리자 속성을 사용하여 각 채널 시작기에 대해 이 SYS1.PARMLIB 멤버의 *xx* 접미부를 지정하십시오. 그러면 채널 시작기가 필요한 z/OS 명령 SET APPC= *xx*를 실행하여 해당 LUNAME을 활성화합니다.

해당 z/OS 이미지가 유효한 동안 실패하는 경우에만 채널 시작기를 재시작하도록 ARM 정책을 정의하십시오. XCFAS 주소 공간과 연관된 사용자 ID에는 IBM MQ 명령 START CHINIT를 실행할 수 있는 권한이 부여되어야 합니다. z/OS 이미지 또한 실패할 때는 채널 시작기를 자동으로 재시작하지 마십시오. 대신 CSQINP2 및 CSQINPX 데이터 세트의 명령을 사용하여 채널 시작기와 리스너를 시작하십시오.

작업 단위 수동 복구

작업 단위 CICS, IMS, RRS 또는 큐 공유 그룹의 다른 큐 관리자를 수동으로 복구할 수 있습니다. 큐 관리자 명령을 사용하여 큐 관리자에 대한 각 연결과 연관된 작업 단위의 상태를 표시할 수 있습니다.

이 주제에는 다음 주제에 대한 정보가 포함되어 있습니다.

- 317 페이지의 『연결 및 스레드 표시』
- 317 페이지의 『CICS 복구 단위 수동 복구』
- 321 페이지의 『IMS 복구 단위 수동 복구』
- 323 페이지의 『RRS 복구 단위 수동 복구』
- 323 페이지의 『큐 공유 그룹의 다른 큐 관리자에서 복구 단위 복구』

연결 및 스레드 표시

DISPLAY CONN 명령을 사용하면 큐 관리자에 대한 연결과 연관 작업 단위에 대한 정보를 얻을 수 있습니다. 활성 작업 단위를 표시하여 현재 상태를 확인하거나 큐 관리자가 종료할 수 있어 종료되어야 하는 작업 단위를 표시할 수 있으며 복구에 도움이 되는 해석되지 않은 작업 단위를 표시할 수 있습니다.

활성 작업 단위

활성 작업 단위만 표시하려면 다음을 사용하십시오.

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

해석되지 않은 작업 단위

해석되지 않은 작업 단위("인다우트(in-doubt) 스레드"라고도 함)는 2단계 커미트 조작의 두 번째 패스에 있는 작업 단위입니다. 자원이 대신 IBM MQ에 저장됩니다. 해석되지 않은 작업 단위를 표시하려면 다음을 사용하십시오.

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

해석되지 않은 작업 단위의 상태를 해석하려면 외부 개입이 필요합니다. 여기에는 복구 코디네이터(CICS, IMS 또는 RRS) 시작만 포함될 수도 있고 그 이상이 포함될 수도 있습니다(다음 절의 설명 참조).

CICS 복구 단위 수동 복구

이 주제를 통해 CICS 어댑터가 재시작될 때 발생하는 상황을 이해하고 해석되지 않은 복구 단위를 처리하는 방법을 설명합니다.

CICS 어댑터가 재시작될 때 발생하는 상황

연결이 끊어질 때마다 어댑터가 다시 연결 프로세스에서 재시작 단계를 진행해야 합니다. 재시작 단계는 자원을 재동기화합니다. CICS와 IBM MQ 간의 재동기화를 통해 인다우트(in-doubt) 작업 단위를 식별하고 해석할 수 있습니다.

가능한 재동기화 원인은 다음과 같습니다.

- 분산 큐잉 컴포넌트로부터의 명시적 요청
- IBM MQ에 연결할 때 암시적 요청

IBM MQ에 연결하여 재동기화가 발생한 경우 이벤트 순서는 다음과 같습니다.

1. 연결 프로세스가 IBM MQ에서 인다우트(in-doubt) 작업 단위(UOW) ID의 목록을 검색합니다.
2. 콘솔에서 CSQC313I 메시지에 UOW ID가 표시됩니다.
3. CICS로 UOW ID가 전달됩니다.
4. CICS는 각 인다우트(in-doubt) UOW ID의 재동기화 태스크(CRSY)를 시작합니다.
5. 각 인다우트(in-doubt) UOW에 대한 태스크의 결과가 콘솔에 표시됩니다.

연결 프로세스에서 표시되는 메시지를 확인해야 합니다.

CSQC313I
UOW가 인다우트(in-doubt) 상태임을 표시합니다.

CSQC400I
UOW를 식별하고 다음 메시지 중 하나가 뒤에 표시됩니다.

- CSQC402I 또는 CSQC403I는 UOW가 해석되었음을 보여줍니다(커미트 또는 백아웃).
- CSQC404E, CSQC405E, CSQC406E 또는 CSQC407E는 UOW가 해석되지 않았음을 보여줍니다.

CSQC409I
모든 UOW가 해석되었음을 보여줍니다.

CSQC408I
일부 UOW가 해석되지 않았음을 보여줍니다.

CSQC314I
*로 강조표시된 UOW ID가 자동으로 해석되지 않음을 경고합니다. 이러한 UOW는 재시작될 때 분산 큐잉 컴포넌트에 의해 명시적으로 해석되어야 합니다.

319 페이지의 그림 54은 z/OS 콘솔에 표시되는 재시작 메시지의 예 설정을 보여줍니다.

```
CSQ9022I +CSQ1 CSQYASCP ' START QMGR' NORMAL COMPLETION
+CSQC323I VICIC1 CSQCQCON CONNECT received from TERMID=PB62 TRANID=CKCN
+CSQC303I VICIC1 CSQCCON CSQCSERV loaded. Entry point is 850E8918
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F0E2178D25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F055B2AC25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6EFFD60D425 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F07AB56D22 is in doubt
+CSQC307I VICIC1 CSQCCON Successful connection to subsystem VC2
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAD18) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAA10) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BA708) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAE88) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAB80) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA878) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA570) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA268) connect
successful
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F0E2178D25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F055B2AC25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F07AB56D22
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6EFFD60D425
+CSQC409I VICIC1 CSQCTRUE Resynchronization completed successfully
```

그림 *54.* 재시작 메시지 예

CSQC313I 메시지의 총수는 CSQC402I 메시지와 CSQC403I 메시지의 총수와 같아야 합니다. 합계가 일치하지 않으면 연결 프로세스로 해석할 수 없는 UOW가 있기 때문입니다. 해석할 수 없는 UOW는 CICS(예를 들어, 콜드 스타트) 또는 IBM MQ 관련 문제점이나 큐잉 분산에 따른 것입니다. 이러한 문제가 해결되면 연결을 끊었다가 다시 연결하여 다른 재동기화를 시작할 수 있습니다.

또는 RESOLVE INDOUBT 명령과 CSQC400I 메시지에 표시된 UOW ID를 사용하여 미해결 UOW를 각각 직접 해석할 수 있습니다. 그런 다음 연결 끊기와 연결을 시작하여 CICS에서 복구 단위 디스크립터를 정리해야 합니다. UOW를 수동으로 해석하려면 UOW의 정확한 결과를 알아야 합니다.

미해결 UOW와 연관된 모든 메시지는 IBM MQ로 잠그며 배치, TSO 또는 CICS 태스크가 액세스할 수 없습니다.

CICS가 실패하고 긴급 재시작이 필요한 경우에는 CICS 시스템의 GENERIC APPLID를 변경하지 마십시오. 변경한 후 IBM MQ에 다시 연결하면 IBM MQ 관련 데이터 무결성을 보장할 수 없습니다. 이는 APPLID가 달라 IBM MQ가 CICS의 새 인스턴스를 다른 CICS로 간주하기 때문입니다. 그러면 인다우트(in-doubt) 해석이 잘못된 CICS 로그를 기반으로 하게 됩니다.

CICS 복구 단위 수동 해석 방법

어댑터가 비정상적으로 종료되면 이상종료를 야기한 서브시스템에 따라 CICS 및 IBM MQ가 인다우트(in-doubt) 목록을 동적으로 또는 재시작 중에 빌드합니다.

참고: DFH$INDB 샘플 프로그램을 사용하여 작업 단위를 표시하는 경우 IBM MQ UOW를 올바르게 표시하지 않을 수 있습니다.

CICS가 IBM MQ에 연결되면 해석되지 않은 하나 이상의 작업 단위가 존재할 수 있습니다.

다음 메시지 중 하나를 콘솔로 보냅니다.

- CSQC404E
- CSQC405E

- CSQC406E
- CSQC407E
- CSQC408I

이러한 메시지의 의미에 대한 자세한 내용은 CICS 어댑터 및 브릿지 메시지 메시지를 참조하십시오.

CICS는 연결 시작 중에 해석되지 않은 복구 단위의 세부사항을 유지합니다. IBM MQ가 제공하는 목록에 더 이상 나타나지 않을 때 항목은 영구 제거됩니다.

CICS가 해석할 수 없는 복구 단위는 IBM MQ 명령을 사용하여 수동으로 해석해야 합니다. 이 수동 프로시저는 작동 오류 또는 소프트웨어 문제점으로 인해 자동 해석을 방해하는 경우에만 필요하므로 설치 내에서 거의 사용되지 않습니다. 인다우트*(in-doubt)* 해석 중에 발견된 불일치는 조사해야 합니다.

복구 단위를 해석하려면 다음을 수행하십시오.

1. 다음 명령을 사용하여 IBM MQ에서 복구 단위 목록을 확보하십시오.

   ```
   +CSQ1  DISPLAY CONN( * ) WHERE(UOWSTATE EQ UNRESOLVED)
   ```

   다음과 같은 메시지가 표시됩니다.

   ```
   CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
   CONN(BC85772CBE3E0001)
   EXTCONN(C3E2D8C3C7D9F0F94040404040404040)
   TYPE(CONN)
   CONNOPTS(
   MQCNO_STANDARD_BINDING
   )
   UOWLOGDA(2005-02-04)
   UOWLOGTI(10.17.44)
   UOWSTDA(2005-02-04)
   UOWSTTI(10.17.44)
   UOWSTATE(UNRESOLVED)
   NID(IYRCSQ1 .BC8571519B60222D)
   EXTURID(BC8571519B60222D)
   QMURID(0000002BDA50)
   URTYPE(CICS)
   USERID(MQTEST)
   APPLTAG(IYRCSQ1)
   ASID(0000)
   APPLTYPE(CICS)
   TRANSID(GP02)
   TASKNO(0000096)
   END CONN DETAILS
   ```

   CICS 연결의 경우, NID는 CICS applid와 동기점 로그 항목이 기록될 때 CICS가 제공한 고유한 숫자로 구성됩니다. 이 고유한 숫자는 동기점 처리 시점에 CICS 시스템 로그와 IBM MQ 로그에 모두 기록된 레코드에 저장됩니다. 이 값을 CICS에서는 복구 토큰이라고 합니다.
2. CICS 로그에서 특정 복구 단위와 관련된 항목을 스캔하십시오.

   복구 토큰 필드(JCSRMTKN)가 네트워크 ID에서 확보한 값과 같은 태스크 관련 설치에 대한 PREPARE 레코드를 찾아보십시오. 네트워크 ID는 IBM MQ가 DISPLAY CONN 명령 출력에서 제공합니다.

   복구 단위에 대한 CICS 로그의 PREPARE 레코드는 CICS 태스크 번호를 제공합니다. 이 CICS 태스크의 로그에서 다른 모든 항목은 이 번호를 사용하여 찾을 수 있습니다.

   로그를 스캔할 때 CICS 저널 인쇄 유틸리티 DFHJUP을 사용할 수 있습니다. 이 프로그램 사용에 대한 자세한 내용은 *CICS* 작동 및 유틸리티 가이드를 참조하십시오.
3. IBM MQ 로그에서 특정 복구 단위와 관련된 NID 레코드를 스캔하십시오. 그런 다음 이 레코드의 URID를 사용하여 이 복구 단위에 대한 로그 레코드의 나머지를 확보하십시오.

   IBM MQ 로그를 스캔할 때 IBM MQ 시작 메시지 CSQJ001I는 이 세션의 시작 RBA를 제공함을 참고하십시오.

로그 레코드 인쇄 프로그램(CSQ1LOGP)을 이 목적으로 사용할 수 있습니다.

4. 필요한 경우 IBM MQ에서 인다우트(in-doubt) 해석을 수행하십시오.

   IBM MQ RESOLVE INDOUBT 명령을 사용하여 복구 단위에 대한 복구 조치를 수행하도록 IBM MQ에 지시할 수 있습니다.

   특정 *connection-name* 과 연관된 모든 스레드를 복구하려면 NID(*) 옵션을 사용하십시오.

   이 명령은 스레드가 커미트 또는 백아웃되었는지 여부를 보여주는 다음 메시지 중 하나를 생성합니다.

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

인다우트(in-doubt) 해석을 수행할 때, IBM MQ 자원만 영향을 받으므로 CICS와 어댑터는 복구 단위를 커미트 또는 백아웃하기 위한 IBM MQ에 대한 명령을 인식하지 못합니다. 그러나 CICS는 IBM MQ가 해석할 수 없는 인다우트(in-doubt) 스레드에 대한 세부사항을 유지합니다. 이 정보는 제공된 목록이 비어 있거나 CICS가 세부사항을 유지하는 복구 단위가 목록에 포함되지 않는 경우 영구 제거됩니다.

IMS 복구 단위 수동 복구

이 주제를 통해 IMS 어댑터가 재시작될 때 발생하는 상황을 이해하고 해석되지 않은 복구 단위를 처리하는 방법을 설명합니다.

IMS 어댑터가 재시작될 때 발생하는 상황

IBM MQ에 대한 연결이 재시작될 때마다 큐 관리자 재시작 또는 IMS /START SUBSYS 명령 다음에 IMS가 다음 재동기화 프로세스를 시작합니다.

1. IMS는 IBM MQ IMS 어댑터에 확실하지 않은 작업 단위(UOW)ID의 목록을 제공합니다(커미트 또는 백아웃의 해석 매개변수와 함께 한 번에 하나 ID).
2. IMS 어댑터는 IBM MQ로 해석 보고서를 전달하고 해당 결과를 다시 IMS에 보고합니다.
3. 모든 IMS 해석 요청을 처리하면 IMS 어댑터가 IBM MQ에서 IMS 시스템이 시작하고 IBM MQ가 계속 인다우트(in-doubt) 상태를 유지하는 모든 UOW의 목록을 가져옵니다. 이는 CSQQ008I 메시지로 IMS 마스터 터미널에 보고됩니다.

참고: UOW가 인다우트(in-doubt) 상태인 경우, 모든 연관된 IBM MQ 메시지를 IBM MQ가 잠궈 애플리케이션이 사용할 수 없습니다.

IMS 복구 단위 수동 해석 방법

IMS가 IBM MQ에 연결되면 IBM MQ가 해석되지 않은 하나 이상의 복구 인다우트(in-doubt) 단위를 가질 수 있습니다.

IBM MQ에 IMS가 해석하지 못한 복구 인다우트 단위가 있으면 IMS 마스터 터미널에서 다음 메시지가 발행됩니다.

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

이 메시지가 발행되면 IMS가 콜드 스타트되었거나 불완전한 로그 테이프로 시작된 것입니다. 이 메시지는 소프트웨어 오류 또는 기타 서브시스템 실패로 인해 IBM MQ 또는 IMS가 비정상적으로 종료되는 경우에도 발행될 수 있습니다.

CSQQ008I 메시지를 수신한 후:

- 연결이 활성 상태를 유지합니다.
- IMS 애플리케이션은 IBM MQ 자원에 계속 액세스할 수 있습니다.

- 일부 IBM MQ 자원은 계속 잠겨 있습니다.

인다우트(in-doubt) 스레드가 해석되지 않으면 IMS 메시지 큐가 축적되기 시작할 수 있습니다. IMS 큐가 용량까지 채워지면 IMS가 종료됩니다. 이러한 잠재적인 어려움을 파악하고 있어야 하므로 복구 인다우트 단위가 완전히 해석될 때까지 IMS를 모니터해야 합니다.

복구 프로시저

IMS 작업 단위를 복구하려면 다음 프로시저를 사용하십시오.

1. /SWI OLDS를 사용하여 IMS 로그를 닫힌 상태로 강제 유지한 다음 IMS 로그를 아카이브하십시오. DFSERA10 유틸리티를 사용하여 이전 IMS 로그 테이프에서 레코드를 인쇄하십시오. 2단계 커미트 요청을 표시하려면 X'3730' 로그 레코드를 입력하고 중지 요청을 표시하려면 X'38' 로그 레코드를 입력하십시오. 각 종속 영역의 마지막 트랜잭션에 요청된 조치를 기록하십시오.
2. DL/I 배치 작업을 실행하여 커미트 지점에 도달하지 않은 각 관련 PSB를 백아웃하십시오. 트랜잭션이 계속 처리되므로 이 프로세스는 시간이 소요될 수 있습니다. 또한 여러 레코드를 백업하여 나머지 처리 작업과 나머지 메시지 큐에 영향을 줄 수 있습니다.
3. 다음 명령을 사용하여 IBM MQ에서 복구 인다우트 단위의 목록을 생성하십시오.

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

   다음과 같은 메시지가 표시됩니다.

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3E2C5C3F24040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-15)
UOWLOGTI(16.39.43)
UOWSTDA(2005-02-15)
UOWSTTI(16.39.43)
UOWSTATE(UNRESOLVED)
NID(IM8F  .BC45A794D3810344)
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0000)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

   IMS의 경우, NID는 IMS 연결 이름과 IMS가 제공한 고유한 수로 구성됩니다. 이 값을 IMS에서는 복구 토큰이라고 합니다. 자세한 정보는 *IMS* 사용자 정의 안내서를 참조하십시오.
4. DISPLAY THREAD 메시지에 표시되는 NID(IMSID와 16진수 OASN)와 DFSERA10 출력에 표시되는 OASN(4바이트 10진수)을 비교하십시오. 커미트 또는 백아웃 여부를 결정하십시오.
5. 다음과 같이 IBM MQ에서 RESOLVE INDOUBT 명령을 사용하여 인다우트(in-doubt) 해석을 수행하십시오.

```
RESOLVE INDOUBT( connection-name )
ACTION(COMMIT|BACKOUT)
NID( network-id )
```

connection-name 과 연관된 모든 스레드를 복구하려면 NID(*) 옵션을 사용하십시오. 이 명령은 스레드가 커미트 또는 백아웃되었는지 여부를 보여주는 다음 메시지 중 하나를 생성합니다.

```
CSQV414I  THREAD network-id COMMIT SCHEDULED
CSQV415I  THREAD network-id BACKOUT SCHEDULED
```

인다우트(in-doubt) 해석을 수행할 때, IBM MQ 자원만 영향을 받으므로 IMS와 어댑터는 복구 인다우트 단위를 커미트 또는 백아웃하기 위한 IBM MQ에 대한 명령을 인식하지 못합니다.

RRS 복구 단위 수동 복구

이 주제를 통해 인다우트(in-doubt) RRS 복구 단위가 있는지 여부를 판별하는 방법과 해당 복구 단위를 수동으로 해석하는 방법을 이해할 수 있습니다.

RRS가 IBM MQ에 연결되면 IBM MQ가 해석되지 않은 하나 이상의 복구 인다우트 단위를 가질 수 있습니다. IBM MQ에 RRS가 해석하지 않은 복구 인다우트 단위가 있는 경우 z/OS 콘솔에 다음 메시지 중 하나가 실행됩니다.

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

IBM MQ 및 RRS 모두 인다우트 복구 단위와 해당 복구 단위를 수동으로 해석하는 기술에 대한 정보를 표시하는 도구를 제공합니다.

IBM MQ에서 DISPLAY CONN 명령을 사용하여 인다우트(in-doubt) IBM MQ 스레드에 대한 정보를 표시하십시오. 명령 출력에는 RSS가 조정자인 IBM MQ 스레드에 대한 RRS 복구 단위 ID가 포함됩니다. 이 ID를 사용하여 복구 단위의 결과를 판별할 수 있습니다.

IBM MQ 인다우트(in-doubt) 스레드를 수동으로 해석하려면 RESOLVE INDOUBT 명령을 사용하십시오. 이 명령은 올바른 결정 사항을 판별한 후 복구 단위를 커미트 또는 백아웃하는 데 사용할 수 있습니다.

큐 공유 그룹의 다른 큐 관리자에서 복구 단위 복구

이 주제를 통해 큐 공유 그룹에 있는 다른 큐 관리자에서 복구 단위를 식별하고 수동으로 복구할 수 있습니다.

큐 공유 그룹의 멤버인 큐 관리자가 실패하여 재시작할 수 없게 되면 그룹의 다른 큐 관리자가 피어 복구를 수행하고 인계받을 수 있습니다. 그러나 해당 복구 단위의 최종 속성 지정을 실패한 큐 관리자만 알고 있어 큐 관리자가 피어 복구로 해석할 수 없는 복구 인다우트 단위를 가질 수 있습니다. 이러한 복구 단위는 큐 관리자가 마침내 재시작될 때 해석되지만 그 때까지는 인다우트(in-doubt) 상태를 유지합니다.

이는 특정 자원(예를 들어, 메시지)이 잠겨 그룹의 다른 큐 관리자가 사용하지 못하게 될 수 있음을 의미합니다. 이러한 경우에는 DISPLAY THREAD 명령을 사용하여 비활성 큐 관리자에서 해당 작업 단위를 표시할 수 있습니다. 해당 복구 단위를 수동으로 해석하여 그룹의 다른 큐 관리자가 메시지를 사용할 수 있도록 하려면 RESOLVE INDOUBT 명령을 사용하면 됩니다.

DISPLAY THREAD 명령을 실행하여 인다우트(in-doubt) 복구 단위를 표시하는 경우 QMNAME 키워드를 사용하여 비활성 큐 관리자의 이름을 지정할 수 있습니다. 예를 들어, 다음 명령을 실행하면

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

다음 메시지를 수신합니다.

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -
NAME   THREAD-XREF    URID NID
USER1  00000000000000000000000000 CSQ:0001.0
USER2  00000000000000000000000000 CSQ:0002.0
DISPLAY THREAD REPORT COMPLETE
```

지정된 큐 관리자가 활성 상태인 경우에는 IBM MQ가 인다우트(in-doubt) 스레드에 대한 정보를 리턴하지 않지만 다음 메시지를 실행합니다.

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

인다우트(in-doubt) 스레드를 수동으로 해석하려면 IBM MQ 명령 RESOLVE INDOUBT를 사용하십시오. QMNAME 키워드를 사용하면 명령에서 비활성 큐 관리자의 이름을 지정할 수 있습니다.

이 명령은 복구 단위를 커미트 또는 백아웃하는 데 사용할 수 있습니다. 이 명령은 복구 단위의 공유 부분만 해석합니다. 로컬 메시지는 영향을 받지 않으며 큐 관리자가 재시작되거나 CICS, IMS 또는 RRS 배치에 다시 연결될 때까지 잠긴 상태를 유지합니다.

IBM MQ 및 IMS

IBM MQ는 IMS, IBM MQ - IMS 어댑터, IBM MQ - IMS 브릿지와 상호작용하는 두 가지 컴포넌트를 제공합니다. 이러한 컴포넌트를 일반적으로 IMS 어댑터, IMS 브릿지라고 합니다.

IMS 어댑터 작동

이 주제를 통해 IBM MQ를 IMS 시스템에 연결하는 IMS 어댑터의 작동 방법을 이해할 수 있습니다.

참고: IMS 어댑터는 조작과 제어판을 통합하지 않습니다.

이 주제에는 다음 절이 포함되어 있습니다.

IMS 연결 제어

이 주제를 통해 IBM MQ에 대한 연결을 제어하고 모니터하는 IMS 운영자 명령을 이해할 수 있습니다.

IMS는 IBM MQ에 대한 연결을 제어하고 모니터하기 위해 다음 운영자 명령을 제공합니다.

/CHANGE SUBSYS
IMS에서 복구 인다우트 단위를 삭제합니다.

/DISPLAY OASN SUBSYS
미해결 복구 요소를 표시합니다.

/DISPLAY SUBSYS
연결 상태와 스레드 활동을 표시합니다.

/START SUBSYS
IMS 제어 리젼을 큐 관리자에 연결합니다.

/STOP SUBSYS
큐 관리자에서 IMS 연결을 끊습니다.

/추적
IMS 추적을 제어합니다.

이러한 명령에 대한 자세한 정보는 사용 중인 IMS의 레벨에 해당하는 *IMS/ESA®* 운영자 참조서 매뉴얼을 참조하십시오.

IMS 명령 응답은 명령이 실행된 터미널로 보냅니다. IMS 명령을 실행할 수 있는 권한은 IMS 보안을 기반으로 합니다.

IMS 제어 영역에서 연결

이 주제를 통해 IMS에서 IBM MQ에 연결하는 데 사용할 수 있는 메커니즘을 이해할 수 있습니다.

IMS는 제어 영역에서 IMS를 사용하는 각 큐 관리자로의 단일 연결을 작성합니다. IMS가 다음 중 한 가지 방법으로 연결을 작성할 수 있어야 합니다.

- 다음 과정에서 자동으로
  - 콜드 스타트 초기화
  - IMS의 웜 스타트(IMS가 종료될 때 IBM MQ 연결이 활성 상태인 경우)
- IMS 명령에 대한 응답으로

```
/START SUBSYS sysid
```

여기서 *sysid*는 큐 관리자 이름입니다.

명령은 큐 관리자가 활성 상태인지 여부에 관계 없이 실행될 수 있습니다.

큐 관리자에 대한 첫 번째 MQ API 호출이 작성될 때까지는 연결이 작성되지 않습니다. 해당 시점까지는 IMS 명령 /DIS SUBSYS가 상태를 'NOT CONN'으로 표시합니다.

IMS와 큐 관리자를 시작하는 순서는 중요하지 않습니다.

큐 관리자가 STOP QMGR 명령, IMS 명령 /STOP SUBSYS 또는 비정상 종료로 인해 중지되면 IMS가 큐 관리자에 대한 연결을 자동으로 다시 시작할 수 없습니다. 따라서 IMS 명령 /START SUBSYS를 사용하여 연결을 작성해야 합니다.

어댑터 초기화 및 큐 관리자에 연결

어댑터는 IMS 제어 및 종속 영역에 로드되는 모듈 세트이며 IMS 외부 서브시스템 첨부 기능을 사용합니다.

이 프로시저는 어댑터를 초기화하고 큐 관리자에 연결됩니다.

1. IMS.PROCLIB에서 서브시스템 멤버(SSM)를 읽으십시오. 선택한 SSM은 IMS EXEC 매개변수입니다. 각 큐 관리자의 멤버에는 IMS가 연결할 수 있는 항목이 하나 있습니다. 각 항목에는 IBM MQ 어댑터에 대한 제어 정보가 포함됩니다.
2. IMS 어댑터를 로드하십시오.

   참고: IMS는 SSM 멤버에 정의된 각 IBM MQ 인스턴스마다 하나의 어댑터 모듈 사본을 로드합니다.
3. IBM MQ의 외부 서브시스템 태스크를 첨부하십시오.
4. CTL EXEC 매개변수(IMSID)를 연결 이름으로 지정하여 어댑터를 실행하십시오.

프로세스는 연결이 IMS 명령 /START SUBSYS의 결과이거나 초기화의 일부인지 여부에 관계 없이 동일합니다.

IMS가 연결 작성을 시도할 때 큐 관리자가 활성 상태인 경우에는 다음 메시지가 전송됩니다.

- z/OS 콘솔:

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

- IMS 마스터 터미널:

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

IMS가 연결 작성을 시도하고 큐 관리자가 활성 상태가 아닌 경우, 애플리케이션이 MQI 호출을 작성할 때마다 IMS 마스터 터미널로 다음 메시지가 전송됩니다.

```
CSQQ001I IMS/TM imsid not connected to queue manager ssnm.
Notify message accepted
DFS3607I MQM1   SUBSYSTEM ID EXIT FAILURE, FC = 0286, RC = 08,
JOBNAME = IMSEMPR1
```

IMS에 대한 연결을 시작하거나 시스템 시작 시 DFS3607I 메시지를 수신하면 큐 관리자를 사용할 수 없음을 의미합니다. 메시지가 많이 생성되는 것을 방지하려면 다음 중 하나를 수행해야 합니다.

1. 관련 큐 관리자를 시작하십시오.
2. IMS 명령을 실행하십시오.

   ```
   /STOP SUBSYS
   ```

   그러면 IMS가 큐 관리자에 연결되도록 기대하지 않습니다.

둘 다 수행하지 않으면 리전에서 작업이 스케줄될 때마다 그리고 애플리케이션이 큐 관리자에 대한 연결 요청을 작성할 때마다 DFS3607I 메시지 및 연관된 CSQQ001I 메시지가 발행됩니다.

스레드 첨부

MPP 또는 IFP 영역에서, 해당 리전에 첫 번째 애플리케이션 프로그램이 스케줄될 때 IMS가 스레드 연결을 작성하며 이는 해당 애플리케이션 프로그램이 IBM MQ 호출을 작성하지 않는 경우에도 해당됩니다. BMP 영역에서는 애플리케이션이 첫 번째 IBM MQ 호출(MQCONN 또는 MQCONNX)을 작성할 때 스레드 연결이 작성됩니다. 이 스레드는 리전 지속 기간 동안 또는 연결이 중지될 때까지 보유됩니다.

메시지 지향 리전과 비메시지 지향 리전 모두, 스레드와 연관된 복구 스레드 교차 참조 ID, *Thread-xref*는 다음과 같습니다.

```
PSTid + PSBname
```

설명:

PSTid
파티션 스펙 테이블 리전 ID

PSBname
프로그램 스펙 블록 이름

연결 ID를 IBM MQ 명령에서 고유 ID로 사용할 수 있으며 이 경우 IBM MQ가 생성된 운영자 메시지에 자동으로 해당 ID를 삽입합니다.

복구 인다우트 단위 표시

복구 인다우트 단위를 표시하고 복구를 시도할 수 있습니다.

이 주제에서 복구 인다우트 단위를 나열하고 복구하는 데 사용되는 조작 단계는 상대적으로 단순한 경우에만 해당됩니다. 큐 관리자가 IMS에 연결된 상태에서 비정상적으로 종료되면 IMS가 IBM MQ를 인식하지 못하고 작업을 커미트 또는 백아웃할 수 있습니다. 큐 관리자가 재시작되면 해당 작업은 인다우트*(in doubt)* 상태가 됩니다. 작업 상태를 결정해야 합니다.

복구 인다우트 단위 목록을 표시하려면 다음 명령을 실행하십시오.

```
+CSQ1  DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

IBM MQ는 다음과 같은 메시지로 응답합니다.

```
CSQM201I +CSQ1 CSQMDRTC DIS CONN DETAILS
CONN(BC0F6125F5A30001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-11-02)
UOWLOGTI(12.27.58)
UOWSTDA(2004-11-02)
UOWSTTI(12.27.58)
UOWSTATE(UNRESOLVED)
NID(CSQ1CHIN.BC0F5F1C86FC0766)
EXTURID(000000000000001F000000007472616E5F6964547565204E6F762020...)
QMURID(000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNAME( )
END CONN DETAILS
```

이 메시지의 속성에 대한 설명은 DISPLAY CONN 명령에 대한 설명을 참조하십시오.

복구 인다우트 단위 복구

복구 인다우트 단위를 복구하려면 다음 명령을 실행하십시오.

```
+CSQ1  RESOLVE INDOUBT( connection-name ) ACTION(COMMIT|BACKOUT)
NID( net-node.number )
```

설명:

connection-name
IMS 시스템 ID

조치
이 복구 단위를 커미트(COMMIT) 또는 백아웃(BACKOUT)할지 여부를 표시합니다.

net-node.number
연관된 net-node.number

RESOLVE INDOUBT 명령을 실행한 경우에는 다음 메시지 중 하나가 표시됩니다.

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED

CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

나머지 복구 항목 해석

지정된 시간에 IMS가 RRE(Residual Recovery Entry) 목록을 빌드합니다. RRE는 인다우트(in-doubt) 상태인 IBM MQ에 대한 복구 단위이며 다음과 같은 여러 상황에서 발생합니다.

- 큐 관리자가 활성 상태가 아닌 경우 IMS에는 큐 관리자가 활성 상태가 될 때까지 해석할 수 없는 RRE가 있습니다. 이러한 RRE는 문제점이 아닙니다.
- 큐 관리자가 활성 상태이고 IMS에 연결되어 있는 경우 및 IMS가 IBM MQ가 커미트한 작업을 백아웃하는 경우, IMS 어댑터가 CSQQ010E 메시지를 발행합니다. 두 시스템의 데이터가 일치해야 하는 경우에는 문제점이 발생합니다. 이러한 문제점 해결에 대한 정보는 321 페이지의 『IMS 복구 단위 수동 복구』의 내용을 참조하십시오.
- 큐 관리자가 활성 상태이고 IMS에 연결된 경우에는 해당 문제점을 알려준 메시지가 없더라도 RRE가 계속 존재할 수 있습니다. IMS에 대한 IBM MQ 연결이 설정된 후에는 다음 IMS 명령을 실행하여 문제점이 있는지 여부를 확인할 수 있습니다.

```
/DISPLAY OASN SUBSYS sysid
```

RRE를 영구 제거하려면 다음 IMS 명령 중 하나를 실행하십시오.

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

여기서 *nnnn*은 +CSQ1 DISPLAY 명령에 대한 응답에 나열되는 시작 애플리케이션 순서 번호입니다. 이 번호는 프로그램 인스턴스의 스케줄 번호로, 마지막 IMS 콜드 스타트 이후 해당 프로그램의 호출 순서에서의 위치를 나타냅니다. IMS는 스케줄 번호가 같은 두 개의 복구 인다우트 단위를 가질 수 없습니다.

이러한 명령은 IMS의 상태를 재설정합니다. IBM MQ와의 통신은 초래하지 않습니다.

IMS 종속 영역 연결 제어

IMS와 IBM MQ 간의 연결을 제어, 모니터하고 필요한 경우 종료할 수 있습니다.

IMS 종속 영역 연결 제어에는 다음 활동이 포함됩니다.

- 종속 영역에서 연결
- 영역 오류 옵션
- 연결에 대한 활동 모니터링
- 종속 영역에서 연결 끊기

종속 영역에서 연결

제어 영역에서 사용되는 IMS 어댑터는 또한 종속 영역에 로드됩니다. 각 종속 영역에서 IBM MQ로의 연결이 작성됩니다. 이 연결은 IBM MQ 및 IMS 작업 확약을 조정하는 데 사용됩니다. 연결을 초기화하고 작성하기 위해 IMS가 다음을 수행합니다.

1. IMS.PROCLIB에서 서브시스템 멤버(SSM)를 읽습니다.

   종속 영역 EXEC 매개변수에서 서브시스템 멤버가 지정될 수 있습니다. 지정되지 않으면 제어 영역 SSM이 사용됩니다. 영역이 IBM MQ에 연결되지 않는 경우 어댑터 로드를 방지하려면 항목 없이 멤버를 지정하십시오.
2. IBM MQ 어댑터를 로드합니다.

   배치 메시지 프로그램의 경우에는 애플리케이션이 첫 번째 메시지 명령을 실행할 때까지 로드가 수행되지 않습니다. 이 경우 IMS가 연결 작성을 시도합니다.

메시지 처리 프로그램 영역 또는 IMS 고속 경로 영역의 경우, 영역이 초기화될 때 시도합니다.

영역 오류 옵션

큐 관리자가 활성 상태가 아니거나 애플리케이션 프로그램에서 첫 번째 메시징 명령을 보낼 때 자원을 사용할 수 없는 경우 수행되는 조치는 SSM 항목에 지정된 오류 옵션에 따라 다릅니다. 해당 옵션은 다음과 같습니다.

R
해당 리턴 코드가 애플리케이션으로 전송됩니다.

Q
애플리케이션이 이상종료 코드 U3051과 함께 비정상적으로 종료됩니다. 입력 메시지를 다시 큐에 넣습니다.

A
애플리케이션이 이상종료 코드 U3047과 함께 비정상적으로 종료됩니다. 입력 메시지는 제거됩니다.

연결에 대한 활동 모니터링

애플리케이션이 첫 번째 IBM MQ 요청에 성공하면 종속 영역으로부터 스레드가 설정됩니다. IBM MQ에서 다음 명령을 실행하여 연결 및 현재 연결을 사용하는 애플리케이션에 대한 정보를 표시할 수 있습니다.

```
+CSQ1 DISPLAY CONN(*) ALL
```

이 명령은 다음과 같은 메시지를 생성합니다.

```
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-12-15)
UOWLOGTI(16.39.43)
UOWSTDA(2004-12-15)
UOWSTTI(16.39.43)
UOWSTATE(ACTIVE)
NID( )
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0049)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

제어 영역의 경우 *thread-xref*는 특수값 CONTROL입니다. 종속 영역의 경우에는 PSBname과 연결된 PSTid입니다. *auth-id*는 작업 카드의 사용자 필드 또는 z/OS 시작 프로시저 테이블의 ID입니다.

표시된 목록에 대한 설명은 IBM MQ for z/OS 메시지, 완료 및 이유 코드 문서에서 CSQV402I 메시지에 대한 설명을 참조하십시오.

IMS는 IBM MQ에 대한 연결을 모니터하기 위한 표시 명령을 제공합니다. 각 종속 영역 연결에서 활성 상태인 프로그램, LTERM 사용자 이름, 제어 영역 연결 상태를 보여줍니다. 명령은 다음과 같습니다.

```
/DISPLAY SUBSYS name
```

IMS와 IBM MQ 간 연결 상태는 다음 중 하나로 표시됩니다.

```
CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME= name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
```

각 종속 영역의 스레드 상태는 다음 중 하나입니다.

```
CONN
CONN, ACTIVE (includes LTERM of user)
```

종속 영역에서 연결 끊기

IMS.PROCLIB의 SSM 멤버에서 값을 변경하려면 종속 영역의 연결을 끊습니다. 이를 위해서는 다음을 수행해야 합니다.

1. IMS 명령을 실행하십시오.

   ```
   /STOP REGION
   ```

2. SSM 멤버를 업데이트하십시오.
3. IMS 명령을 실행하십시오.

   ```
   /START REGION
   ```

IMS 연결 끊기

IMS 또는 큐 관리자가 종료되면 연결이 종료됩니다. 또는 IMS 마스터 터미널 운영자가 명시적으로 연결을 끊을 수 있습니다.

IMS와 IBM MQ 간의 연결을 종료하려면 다음 IMS 명령을 사용하십시오.

```
/STOP SUBSYS sysid
```

이 명령은 다음 메시지를 메시지를 실행한 터미널(일반적으로 마스터 터미널 운영자(MTO))로 보냅니다.

```
DFS058I STOP COMMAND IN PROGRESS
```

다음 IMS 명령은

```
/START SUBSYS sysid
```

연결을 재설정하는 데 필요합니다.

참고: IMS 트리거 모니터가 실행되는 경우에는 IMS 명령 /STOP SUBSYS가 완료되지 않습니다.

IMS 트리거 모니터 제어

CSQQTRMN 트랜잭션을 사용하여 IMS 트리거 모니터를 중지 및 시작할 수 있습니다.

IMS 트리거 모니터(CSQQTRMN 트랜잭션)는 IMS 트리거 모니터 설정에 설명되어 있습니다.

IMS 트리거 모니터를 제어하려면 다음을 참조하십시오.

- CSQQTRMN 시작
- CSQQTRMN 중지

CSQQTRMN 시작

1. 모니터하려는 각 이니시에이션 큐마다 프로그램 CSQQTRMN을 실행하는 배치 지향 BMP를 시작하십시오.
2. 배치 JCL을 수정하여 다음 정보를 포함하는 데이터 세트를 가리키는 CSQQUT1의 DDname을 추가하십시오.

```
QMGRNAME=q_manager_name    Comment: queue manager name
INITQUEUENAME=init_q_name   Comment: initiation queue name
LTERM=lterm           Comment: LTERM to remove error messages
CONSOLEMESSAGES=YES      Comment: Send error messages to console
```

   설명:

q_manager_name	큐 관리자의 이름(공백이면 CSQQDEFV에 지정된 기본값으로 가정)
init_q_name	모니터할 이니시에이션 큐의 이름
lterm	오류 메시지 목적지의 IMS LTERM 이름(공백인 경우 기본값은 MASTER)
CONSOLEMESSAGES=YES	메시지가 지정된 IMS LTERM으로 보낸 요청 또한 z/OS 콘솔로 보냅니다. 이 매개변수가 생략되거나 철자가 올바르지 않는 경우 기본값은 콘솔로 메시지를 보내지 않는 것입니다.

3. CSQQUT1 입력 처리의 인쇄된 보고서가 필요한 경우 CSQQUT2의 DD 이름을 추가하십시오.

참고:

1. 데이터 세트 CSQQUT1은 LRECL=80으로 정의됩니다. 다른 DCB 정보는 데이터 세트에서 가져옵니다. 데이터 세트 CSQQUT2의 DCB는 RECFM=VBA 및 LRECL=125입니다.
2. 레코드마다 키워드를 하나만 넣을 수 있습니다. 키워드 값은 키워드 뒤의 첫 번째 공백으로 구분됩니다. 이는 주석을 포함할 수 있음을 의미합니다. 열 1의 별표는 전체 입력 레코드가 주석임을 의미합니다.
3. QMGRNAME 또는 LTERM 키워드 중 하나의 철자가 올바르지 않으면 CSQQTRMN이 해당 키워드의 기본값을 사용합니다.
4. 트리거 모니터 BMP 작업을 제출하기 전에 서브시스템이 IMS에서 시작되었는지(/START SUBSYS 명령으로) 확인하십시오. 시작되지 않은 경우에는 이상종료 코드 U3042와 함께 트리거 모니터 작업이 종료됩니다.

CSQQTRMN 중지

시작된 CSQQTRMN은 다음 이벤트 중 하나로 인해 IBM MQ와 IMS 간의 연결이 끊어질 때까지 실행됩니다.

- 큐 관리자 종료
- IMS 종료

또는 z/OS STOP **jobname** 명령이 입력될 때까지 실행됩니다.

IMS 브릿지 제어

이 주제를 통해 IMS 브릿지를 제어하기 위해 사용할 수 있는 IMS 명령을 이해할 수 있습니다.

IBM MQ-IMS 브릿지를 제어하는 IBM MQ 명령은 없습니다. 그러나 다음과 같은 방식으로 IMS에 메시지가 전달되지 못하게 할 수 있습니다.

- 비공유 큐의 경우, 모든 브릿지 큐에 ALTER QLOCAL(xxx) GET(DISABLED) 명령을 사용합니다.
- 클러스터된 큐의 경우, SUSPEND QMGR CLUSTER(xxx) 명령을 사용합니다. 이 방법은 다른 큐 관리자 또한 클러스터 브릿지 큐를 호스트하는 경우에만 유효합니다.
- 클러스터된 큐의 경우, SUSPEND QMGR FACILITY(IMSBRIDGE) 명령을 사용합니다. IMS로는 추가 메시지가 전송되지 않지만 미해결 트랜잭션에 대한 응답은 IMS에서 수신됩니다.

  IMS로 다시 메시지 송신을 시작하려면 RESUME QMGR FACILITY(IMSBRIDGE) 명령을 실행하십시오.

MQSC 명령 DISPLAY SYSTEM을 사용하여 브릿지가 일시중지되었는지 여부를 표시할 수도 있습니다.

이러한 명령에 대한 자세한 내용은 MQSC 명령을 참조하십시오.

추가 정보는 다음을 참조하십시오.

- 332 페이지의 『IMS 브릿지 시작 및 중지』
- 332 페이지의 『IMS 연결 제어』
- 브릿지 큐 제어
- 334 페이지의 『IMS 브릿지 재동기화』
- tpipe 이름에 대한 작업
- IMS에서 메시지 삭제
- tpipe 삭제
- 335 페이지의 『IMS 트랜잭션 만기』

IMS 브릿지 시작 및 중지

OTMA를 시작하여 IBM MQ 브릿지를 시작하십시오. 다음과 같이 IMS 명령을 사용하거나

```
/START OTMA
```

또는 IMS 시스템 매개변수에서 OTMA=YES를 지정하여 자동으로 시작하십시오. OTMA가 이미 시작된 경우에는 큐 관리자 시작이 완료될 때 자동으로 브릿지가 시작됩니다. OTMA가 시작되면 IBM MQ 이벤트 메시지가 생성됩니다.

다음 IMS 명령을 사용하여

```
/STOP OTMA
```

OTMA 통신을 중지하십시오. 이 명령이 실행되면 IBM MQ 이벤트 메시지가 생성됩니다.

IMS 연결 제어

IMS는 IBM MQ에 대한 연결을 제어하고 모니터하기 위해 다음 운영자 명령을 제공합니다.

/DEQUEUE TMEMBER *tmember* TPIPE *tpipe*
Tpipe에서 메시지를 제거합니다. 모든 메시지를 제거하려면 PURGE를 지정하고 첫 번째 메시지만 제거하려면 PURGE1을 지정하십시오.

/DISPLAY OTMA
OTMA 서버 및 클라이언트와 클라이언트 상태에 대한 요약 정보를 표시합니다.

/DISPLAY TMEMBER *name*
OTMA 클라이언트에 대한 정보를 표시합니다.

/DISPLAY TRACE TMEMBER *name*
추적 대상에 대한 정보를 표시합니다.

/SECURE OTMA
보안 옵션을 설정합니다.

/START OTMA
OTMA를 통한 통신을 사용 가능하게 설정합니다.

/START TMEMBER *tmember* TPIPE *tpipe*
이름 지정된 Tpipe를 시작합니다.

/STOP OTMA
OTMA를 통한 통신을 중지합니다.

/STOP TMEMBER *tmember* TPIPE *tpipe*
이름 지정된 Tpipe를 중지합니다.

/추적
IMS 추적을 제어합니다.

이러한 명령에 대한 자세한 정보는 사용 중인 IMS의 레벨에 해당하는 *IMS/ESA* 운영자 참조서 매뉴얼을 참조하십시오.

IMS 명령 응답은 명령이 실행된 터미널로 보냅니다. IMS 명령을 실행할 수 있는 권한은 IMS 보안을 기반으로 합니다.

브릿지 큐 제어

브릿지를 통해 XCF 멤버 이름 *tmember*로 큐 관리자와의 통신을 중지하려면 다음 IMS 명령을 실행하십시오.

```
/STOP TMEMBER tmember TPIPE ALL
```

통신을 재개하려면 다음 IMS 명령을 실행하십시오.

```
/START TMEMBER tmember TPIPE ALL
```

큐에 대한 Tpipe는 MQ DISPLAY QUEUE 명령을 사용하여 표시될 수 있습니다.

단일 Tpipe에서 큐 관리자와의 통신을 중지하려면 다음 IMS 명령을 실행하십시오.

```
/STOP TMEMBER tmember TPIPE tpipe
```

각 활성 브릿지 큐마다 하나 또는 두 개의 Tpipe가 작성되므로 이 명령을 실행하면 IBM MQ 큐와의 통신이 중지됩니다. 통신을 재개하려면 다음 IMS 명령을 사용하십시오.

```
/START TMEMBER tmember TPIPE tpipe
```

또는 IBM MQ 큐의 속성을 대체하여 금지할 수 있습니다.

IMS 브릿지 재동기화

IMS 브릿지는 큐 관리자, IMS 또는 OTMA가 재시작될 때마다 자동으로 재시작됩니다.

IMS 브릿지가 수행하는 첫 번째 태스크는 IMS와의 재동기화입니다. 여기에는 동기화된 모든 Tpipe에서의 IBM MQ 및 IMS 검사 순서 번호가 포함됩니다. 커미트 모드 0을 사용하여 IBM MQ - IMS 브릿지 큐에서 IMS로 지속 메시지가 전송될 때 동기화된 Tpipe가 사용됩니다(커미트 후 전송).

브릿지가 IMS와 재동기화될 수 없으면 CSQ2023E 메시지에 IMS 감지 코드가 리턴되고 OTMA에 대한 연결이 중지됩니다. 브릿지가 개별 IMS Tpipe와 재동기화될 수 없으면 CSQ2025E 메시지에 IMS 감지 코드가 리턴되고 Tpipe가 중지됩니다. Tpipe가 콜드 스타트되면 복구 가능한 순서 번호가 자동으로 1로 재설정됩니다.

브릿지가 Tpipe와 재동기화할 때 불일치 순서 번호를 발견하면 CSQ2020E 메시지가 발행됩니다. IMS Tpipe와의 재동기화를 시작하려면 IBM MQ 명령 RESET TPIPE를 사용하십시오. XCF 그룹 및 멤버 이름과 Tpipe의 이름을 제공해야 합니다. 이 정보는 메시지로 제공됩니다.

또한 다음을 지정할 수 있습니다.

- IBM MQ가 송신한 메시지에 대해 Tpipe에서 설정되고 파트너의 수신 순서 번호로 설정될 복구 가능한 새 순서 번호. 이 번호를 지정하지 않으면 파트너의 수신 순서 번호가 현재 IBM MQ 송신 순서 번호로 설정됩니다.
- IBM MQ가 수신한 메시지에 대해 Tpipe에서 설정되고 파트너의 송신 순서 번호로 설정될 복구 가능한 새 순서 번호. 이 번호를 지정하지 않으면 파트너의 송신 순서 번호가 현재 IBM MQ 수신 순서 번호로 설정됩니다.

Tpipe와 연관된 해석되지 않은 복구 단위가 있으면 메시지로도 알립니다. 복구 단위를 커미트 또는 백아웃할지 여부를 지정하려면 IBM MQ 명령 RESET TPIPE를 사용하십시오. 복구 단위를 커미트하는 경우 메시지 배치를 이미 IMS로 보내고 브릿지 큐에서 삭제된 상태입니다. 복구 단위를 백아웃하는 경우에는 메시지가 브릿지 큐로 리턴된 다음 나중에 IMS로 송신됩니다.

커미트 모드 1(송신 후 커미트) Tpipe는 동기화되지 않습니다.

커미트 모드 1 트랜잭션에 대한 고려사항

IMS에서는 커미트 모드 1(CM1) 트랜잭션이 동기점 이전에 해당 출력 응답을 보냅니다.

예를 들어 다음과 같은 이유로 CM1 트랜잭션이 응답을 보내지 못할 수 있습니다.

- 응답을 보내는 Tpipe가 중지되었습니다.
- OTMA가 중지되었습니다.
- OTMA 클라이언트(즉, 큐 관리자)가 없습니다.
- 응답 대상 큐와 데드-레터 큐를 사용할 수 없습니다.

이러한 이유로, 메시지를 보내는 IMS 애플리케이션이 코드 U0119와 함께 의사 이상종료됩니다. 이 경우 IMS 트랜잭션과 프로그램은 중지되지 않습니다.

이러한 이유로 인해 IMS로 메시지가 전송되지 않거나 IMS에서 메시지가 전달되지 않는 경우가 자주 있습니다. 다음과 같은 경우 U0119 이상종료가 발생할 수 있습니다.

- 메시지가 IMS에 있는 동안 Tpipe, OTMA 또는 큐 관리자가 중지되는 경우
- IMS가 수신 메시지와 다른 Tpipe에서 응답하고 해당 Tpipe가 중지되는 경우
- IMS가 다른 OTMA 클라이언트에 응답하고 해당 클라이언트를 사용할 수 없는 경우

U0119 이상종료가 발생할 때마다 IMS로의 수신 메시지와 IBM MQ로의 재생 메시지를 둘 다 잃게 됩니다. 이러한 이유로 CM0 트랜잭션의 출력을 전달할 수 없는 경우에는 IMS 내 Tpipe에 큐잉됩니다.

tpipe 이름 처리

IBM MQ - IMS 브릿지를 제어하는 데 사용되는 많은 명령에는 *tpipe* 이름이 필요합니다. 이 주제를 통해 tpipe 이름에 대한 추가 세부사항을 찾을 수 있는 방법을 이해할 수 있습니다.

IBM MQ - IMS 브릿지를 제어하는 많은 명령에는 *tpipe* 이름이 필요합니다. DISPLAY QUEUE 명령에서 tpipe 이름을 가져올 수 있으며 다음 사항에 유의합니다.

- tpipe 이름은 로컬 큐를 정의할 때 지정됩니다
- 로컬 큐에는 두 개의 tpipe 이름(동기용과 비동기용으로 하나씩)이 지정됩니다.
- 해당 특정 로컬 큐에 고유한 IMS와 IBM MQ 간의 통신이 발생할 때까지 IMS가 tpipe 이름을 알지 못합니다.
- IBM MQ - IMS 브릿지가 tpipe를 사용하려면 올바른 XCF 그룹이 있는 스토리지 클래스에 연관된 큐 이름이 지정되어야 하며 멤버 이름 필드가 완성되어야 합니다.

IMS에서 메시지 삭제

Tmember/Tpipe가 중지되면 IMS 브릿지를 통해 IBM MQ가 목적지로 지정된 메시지를 삭제할 수 있습니다. XCF 멤버 이름이 *tmember*인 큐 관리자에 대한 하나의 메시지를 삭제하려면 다음 IMS 명령을 실행하십시오.

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

Tpipe에서 모든 메시지를 삭제하려면 다음 IMS 명령을 실행하십시오.

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

tpipe 삭제

IMS tpipe는 직접 삭제할 수 없습니다. 다음과 같은 경우 IMS가 삭제합니다.

- IMS가 콜드 스타트될 때 동기화된 tpipe가 삭제됩니다.
- IMS가 재시작될 때 동기화되지 않은 tpipe가 삭제됩니다.

IMS 트랜잭션 만기

만기 시간은 트랜잭션과 연관이 있습니다. 모든 IBM MQ 메시지에는 연관된 만기 시간이 있습니다. 만기 간격은 MQMD.Expiry 필드를 사용하여 애플리케이션에서 IBM MQ로 전달됩니다. 이 시간은 만기 전 메시지 지속 시간으로, 해당 값은 0.1초 단위로 표시됩니다. 만기된 후 나중에 메시지 MQGET을 수행하려고 시도하면 수행된 큐 및 만기 처리에서 메시지가 제거됩니다. 만기 시간은 IBM MQ 네트워크에서 큐 관리자 간에 메시지가 이동함에 따라 감소합니다. IMS 브릿지에서 OTMA로 IMS 메시지가 전달되면 나머지 메시지 만기 시간이 트랜잭션 만기 시간으로서 OTMA로 전달됩니다.

트랜잭션에 만기 시간이 지정된 경우 OTMA는 다음과 같은 IMS의 세 가지 다른 시점에 입력 트랜잭션을 만기시킵니다.

- XCF에서 입력 메시지 수신
- 입력 메시지를 큐에 넣는 시간
- 애플리케이션 GU 시간

GU 시간 이후에는 만기가 수행되지 않습니다.

트랜잭션 EXPRTIME은 다음에 의해 제공될 수 있습니다.

- IMS 트랜잭션 정의

- IMS OTMA 메시지 헤더
- IMS DFSINSX0 사용자 엑시트
- IMS CREATE 또는 UPDATE TRAN 명령

IMS는 0243으로 트랜잭션을 이상종료하고 메시지를 발행하여 트랜잭션이 만기되었음을 표시합니다. 발행된 메시지는 비공유 큐 환경의 DFS555I 또는 공유 큐 환경의 DFS2224I입니다.

IBM MQ Advanced Message Security 운영

IBM MQ Advanced Message Security 주소 공간에 대한 명령을 입력하려면 z/OS MODIFY 명령을 사용하십시오.

예를 들면 다음과 같습니다.

```
F qmgr AMSM, cmd
```

허용되는 MODIFY 명령은 다음과 같습니다.

표 *19. IBM MQ Advanced Message Security* 주소 공간 *MODIFY* 명령

명령	**옵션**	**설명**
DISPLAY		버전 정보 표시
REFRESH	KEYRING POLICY 모두	키 링 인증서나 보안 정책 또는 둘 다 새로 고칩니다.
SMFAUDIT	SUCCESS FAILURE 모두	AMS가 메시지를 보호/보호 해제하거나 AMS가 메시지를 보호/보호 해제하지 않거나 이 두 가지 경우에 모두 해당할 때 SMF 감사가 필요한지 여부를 설정합니다.
SMFTYPE	0 - 255	AMS가 메시지를 보호/보호 해제할 때 생성될 SMF 레코드 유형을 설정합니다. SMF 감사를 사용하지 않으려면 레코드 유형 0을 지정하십시오.

참고: 옵션을 지정하려면 쉼표로 분리해야 합니다. 예를 들면, 다음과 같습니다.

```
F qmgrAMSM,REFRESH KEYRING
F qmgrAMSM,SMFAUDIT ALL
F qmgrAMSM,SMFTYPE 180
```

REFRESH 명령

MQOPEN 호출을 발행하는 애플리케이션에는 변경사항이 적용됩니다. 기존 애플리케이션에서는 애플리케이션이 큐를 열 때 제공되는 옵션을 계속 사용합니다. 변경사항을 적용하려면 애플리케이션이 큐를 닫았다 다시 열어야 합니다.

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

150-945
서울특별시 영등포구
국제금융로 10, 3IFC
한국 아이.비.엠 주식회사
U.S.A.

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

지적 재산권 라이센스 부여
2-31 Roppongi 3-chome, Minato-Ku
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 명시적 또는 묵시적인 일체의 보증 없이 이 책을 "현상태대로" 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

서울특별시 영등포구
서울특별시 강남구 도곡동 467-12,
군인공제회관빌딩
한국 아이.비.엠 주식회사
U.S.A.

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등)하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 단계의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을

통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이들 샘플 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다.

이 정보를 소프트카피로 확인하는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

프로그래밍 인터페이스 정보

프로그래밍 인터페이스 정보는 본 프로그램과 함께 사용하기 위한 응용프로그램 소프트웨어 작성을 돕기 위해 제공됩니다.

이 책에는 고객이 프로그램을 작성하여 WebSphere MQ서비스를 얻을 수 있도록 하는 계획된 프로그래밍 인터페이스에 대한 정보가 포함되어 있습니다.

그러나 본 정보에는 진단, 수정 및 성능 조정 정보도 포함되어 있습니다. 진단, 수정 및 성능 조정 정보는 응용프로그램 소프트웨어의 디버그를 돕기 위해 제공된 것입니다.

중요사항: 이 진단, 수정 및 튜닝 정보는 변경될 수 있으므로 프로그래밍 인터페이스로 사용하지 마십시오.

상표

IBM, IBM 로고, ibm.com®는 전세계 여러 국가에 등록된 IBM Corporation의 상표입니다. 현재 IBM 상표 목록은 웹 "저작권 및 상표 정보"(www.ibm.com/legal/copytrade.shtml) 에 있습니다. 기타 제품 및 서비스 이름은 IBM 또는 타사의 상표입니다.

Microsoft 및 Windows는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.

Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.

이 제품에는 Eclipse 프로젝트 (http://www.eclipse.org/) 에서 개발한 소프트웨어가 포함되어 있습니다.

Java 및 모든 Java 기반 상표와 로고는 Oracle 및/또는 그 계열사의 상표 또는 등록상표입니다.

부품 번호:

(1P) P/N: