7.5

IBM WebSphere MQ 관리

참고

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 153 페이지의 『주의사항』에 있는 정보를 확인하십시오.

This edition applies to version 7 release 5 of IBM® WebSphere® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

목차

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

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

- 5 페이지의 『IBM WebSphere MQ 제어 명령』
- 5 페이지의 『IBM WebSphere MQ 스크립트(MQSC) 명령』
- 6 페이지의 『프로그래밍 가능 명령 형식(PCF)』

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

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

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 WebSphere MQ Administration Interface)를 사용하여 단순화될 수 있습니다. 관리 태스크 자동화에 대한 자세한 정보는 8 페이지의 『관리 태스크 자동화』의 내용을 참조하십시오.

IBM WebSphere MQ 제어 명령

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

IBM WebSphere MQ for Windows, UNIX and Linux® systems provides the 제어 명령 that you issue at the system command line.

제어 명령은 큐 관리자 작성 및 관리에 설명되어 있습니다. 제어 명령을 위한 명령 참조의 경우, IBM WebSphere MQ 제어 명령을 참조하십시오.

IBM WebSphere MQ 스크립트(MQSC) 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 플랫폼 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. 자세한 내용은 MQSC 참조를 확인하십시오.

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 WebSphere 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 PCF(Programmable Command Formats) 참조서를 참조하십시오.

IBM WebSphere MQ Explorer

IBM WebSphere 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 는 시스템 메뉴, `MQExplorer` 실행 파일 또는 **`strmqcfg`** 명령을 사용하여 IBM WebSphere MQ Explorer 를 시작할 수 있습니다.

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

IBM WebSphere 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.

자세한 정보는 53 페이지의 『IBM WebSphere MQ Explorer를 사용하여 관리』의 내용을 참조하십시오.

Windows 기본 구성 애플리케이션

Windows 기본 구성 프로그램을 사용하여 IBM WebSphere MQ 오브젝트의 선발 (또는 기본) 세트를 작성할 수 있습니다. 작성된 기본 오브젝트의 요약이 표 1: Windows 기본 구성 애플리케이션에 의해 작성된 오브젝트에 나열되어 있습니다.

MSCS(Microsoft Cluster Service)

Microsoft Cluster Service(MSCS)를 사용하면 서버를 클러스터에 연결하여 데이터 및 애플리케이션의 가용성을 높이고 시스템 관리를 쉽게 만들 수 있습니다. MSCS는 서버 또는 애플리케이션 실패를 자동으로 감지하고 복구할 수 있습니다.

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

IBM WebSphere 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.

Service (MSCS) 지원 에서는 Windows 시스템이 MSCS를 사용하도록 IBM WebSphere MQ 를 구성하는 방법에 대한 자세한 정보를 제공합니다.

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

99 페이지의 『원격 IBM WebSphere MQ 오브젝트 관리』

XA 자원 관리자에 대한 연결이 끊길 경우 고려사항
관련 태스크
계획 중
구성
관련 참조
트랜잭션 지원 시나리오

로컬 및 원격 관리

WebSphere 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 있습니다. WebSphere MQ에서는 아무 채널도 포함되지 않았으므로, 즉, 운영 체제가 통신을 관리하므로 이를 로컬 관리로 간주할 수 있습니다.

WebSphere MQ는 원격 관리로 알려진 단일 접속 지점에서의 관리를 지원합니다. 그러면 다른 시스템에서 처리되는 명령을 로컬 시스템에서 실행할 수 있고 이를 WebSphere MQ Explorer에 적용할 수도 있습니다. 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. 이러한 제한사항은 WebSphere MQ Explorer에도 적용됩니다.

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

IBM WebSphere MQ 제어 명령 사용 방법

이 섹션은 IBM WebSphere MQ 제어 명령을 사용하는 방법을 설명합니다.

제어 명령을 실행하려면 사용자 ID가 mqm 그룹의 구성원이어야 합니다. 이에 대한 자세한 정보는 Authority to administer WebSphere MQ on UNIX, Linux and 윈도우 systems 의 내용을 참조하십시오. 또한, 다음 환경별 정보를 참고하십시오.

윈도우 에 대한 WebSphere MQ
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.

UNIX and Linux 시스템용 WebSphere MQ
모든 WebSphere MQ 제어 명령은 쉘에서 실행될 수 있습니다. 모든 명령은 대소문자를 구분합니다.

제어 명령의 서브세트는 IBM WebSphere MQ 탐색기를 사용하여 실행될 수 있습니다.

자세한 정보는 WebSphere 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 WebSphere MQ 오브젝트 관리에 경험이 있다고 가정합니다.

PCF 명령

WebSphere MQ PCF(Programmable Command Format)는 관리 태스크를 관리 프로그램으로 프로그래밍하는 데 사용할 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 WebSphere MQ 메시지의 애플리케이션 데이터 부분에 임베드된 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
표준 WebSphere 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 명령을 지정합니다 (예: *Change Queue* (MQCMD_CHANGE_Q)).

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 WebSphere MQ for AIX®
- HP-UX용 IBM WebSphere MQ
- Linux 용 IBM WebSphere MQ
- Solaris용 IBM WebSphere MQ
- IBM WebSphere MQ for 윈도우
- 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.

WebSphere MQ PCF 명령을 사용하여 큐 관리자 관리 및 기타 네트워크 관리를 단순화할 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. 시스템 관리 애플리케이션 프로그램에서 PCF 명령을 사용하여 WebSphere MQ 오브젝트(인증 정보 오브젝트, 채널, 채널 리스너, 이름 목록, 프로세스 정의, 큐 관리자, 큐, 서비스 및 스토리지 클래스)를 관리할 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

WebSphere MQ 원격 관리용 시스템 관리 프로그램에서 PCF를 사용할 수 있습니다.

이 섹션에는 다음이 포함됩니다.

- 10 페이지의 『PCF 명령 메시지』
- 12 페이지의 『응답』
- IBM WebSphere MQ 오브젝트 이름 지정 규칙
- 14 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/OS이외의 플랫폼에서는 PCF 명령이 전송되는 큐가 항상 SYSTEM.ADMIN.COMMAND.QUEUE라고 합니다.

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 목록은 WebSphere MQ COPY, 헤더, 포함 및 모듈 파일을 참조하십시오.

PCF 명령에 대한 메시지 디스크립터

WebSphere MQ 메시지 디스크립터는 MQMD - 메시지 디스크립터에 자세히 설명되어 있습니다.

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

CodedCharSetId
만약Windows, 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. 백의 컨텐츠는 호출 후에도 변경되지 않은 상태로 남아 있습니다.

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. 해당 명령은 z/OS를 제외한 모든 플랫폼에서 지원됩니다. 이전 애플리케이션은 z/OS의 PCF를 지원하지 않습니다. WebSphere MQ Windows Explorer는 이러한 애플리케이션 중 하나입니다. (그러나 버전 6.0 이상의 IBM WebSphere MQ Explorer는 z/OS에서 PCF를 지원합니다.)

명령 메시지의 PCF 헤더 유형이 MQCFT_COMMAND_XR인 경우, 확장된 응답 또는 표준 응답이 생성됩니다. 해당 명령은 z/OS 및 일부 기타 플랫폼에서 지원됩니다. z/OS에서 실행된 명령은 확장 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 응답의 정의(프로그래밍 가능한 명령 형식의 정의)에서 항상 리턴됨으로 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. 해당 명령은 z/OS를 제외한 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.

PCF 명령에 대한 권한 검사

PCF 명령이 처리될 때 명령 메시지 내에 있는 메시지 디스크립터의 *UserIdentifier*가 필수 WebSphere 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 WebSphere MQ for 윈도우, UNIX and Linux systems

PCF 명령을 처리하기 위해 사용자 ID에는 대상 시스템에 큐 관리자 오브젝트에 대한 *dsp* 권한이 있어야 합니다. 또한 WebSphere MQ 오브젝트 권한 검사가 15 페이지의 표 1에서 표시한 대로 특정 PCF 명령에 대해 수행됩니다.

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

참고: Windows의 경우에 **한해**, 사용자 ID가 그룹 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

HP Integrity NonStop Server 용 WebSphere MQ

PCF 명령을 처리하기 위해 사용자 ID에는 대상 시스템에 큐 관리자 오브젝트에 대한 *dsp* 권한이 있어야 합니다. 또한 IBM WebSphere MQ 오브젝트 권한 검사는 15 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

WebSphere MQ 오브젝트 권한

표 *1.* 윈도우, *HP Integrity NonStop Server, UNIX and Linux* 시스템-오브젝트 권한

명령	**WebSphere 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
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

표 *1.* 윈도우, *HP Integrity NonStop Server, UNIX and Linux* 시스템-오브젝트 권한 *(*계속*)*		
명령	**WebSphere MQ 오브젝트 권한**	**클래스 권한(오브젝트 유형의 경우)**
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
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

표 *1.* 윈도우, *HP Integrity NonStop Server, UNIX and Linux* 시스템-오브젝트 권한 *(계속)*		
명령	**WebSphere MQ 오브젝트 권한**	**클래스 권한(오브젝트 유형의 경우)**
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
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.

또한 WebSphere MQ에서는 사용자가 직접 보안 검사를 위해 엑시트 프로그램을 제공할 수 있도록 몇 가지 채널 보안 엑시트를 제공합니다. 세부사항은 채널 표시 매뉴얼에 제공됩니다.

MQAI를 사용하여 PCF의 사용 단순화

MQAI는 AIX, HP-UX, IBM i, Linux, Solaris및 플랫폼 사용 가능한 WebSphere 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.

다음과 같은 방법으로 MQAI를 사용할 수 있습니다.

PCF 메시지 사용 단순화
MQAI는 WebSphere MQ를 쉽게 관리할 수 있는 방법을 제공합니다. 사용자 자신의 PCF 메시지를 작성할 필요가 없으며 복잡한 데이터 구조와 연관 문제를 피할 수 있습니다.

MQI 호출을 사용하여 작성된 프로그램의 매개변수를 전달하려면 PCF 메시지에 문자열이나 정수 데이터의 세부사항 및 명령이 포함되어야 합니다. 이를 수행하려면 프로그램에 모든 구조에 대해 여러 명령문이 있어야 하며 메모리 공간이 할당되어야 합니다. 이는 길고 힘든 작업일 수 있습니다.

MQAI를 사용하여 작성된 프로그램은 매개변수를 해당 데이터 백으로 전달하므로 각 구조에 대해 하나의 명령문만 있으면 됩니다. MQAI 데이터 백을 사용하면 배열을 처리하고 스토리지를 할당할 필요가 없어지며 PCF를 단순화할 수 있습니다.

오류 조건을 보다 쉽게 처리
PCF 명령에서 리턴 코드를 다시 가져오는 것은 어렵지만 MQAI를 사용하면 프로그램이 오류 조건을 보다 쉽게 처리할 수 있습니다.

데이터 백을 작성하여 채운 다음, `mqExecute` 호출을 사용하여 관리 명령 메시지를 큐 관리자의 명령 서버로 보낼 수 있습니다. 그리고 이 호출은 응답 메시지를 대기합니다. `mqExecute` 호출은 명령 서버와의 교환을 처리하고 응답 백에 응답을 리턴합니다.

MQAI에 대한 자세한 정보는 18 페이지의 『MQAI(IBM WebSphere MQ Administration Interface)에 대한 소개』 의 내용을 참조하십시오.

MQAI(IBM WebSphere MQ Administration Interface)에 대한 소개

MQAI(IBM WebSphere MQ Administration Interface)는 IBM WebSphere 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 WebSphere MQ 큐 관리자에서 관리 태스크를 수행합니다.

MQAI 개념 및 용어

MQAI는 WebSphere MQ에 대한 프로그래밍 인터페이스로서, C 언어와 Windows용 Visual Basic을 사용합니다. z/OS 이외의 플랫폼에서도 사용 가능합니다.

이 MQAI는 데이터 백을 사용하여 WebSphere 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 44 페이지의 『데이터 백』 을 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 WebSphere MQ를 관리하는 쉬운 방법입니다. 사용자의 PCF 메시지를 작성할 필요가 없으므로 복잡한 데이터 구조와 연관된 문제를 피할 수 있습니다.
- 오류 조건을 더 쉽게 핸들링합니다. WebSphere MQ 스크립트(MQSC)에서 리턴 코드를 다시 가져오는 것은 어렵지만 MQAI를 사용하면 프로그램이 오류 조건을 보다 쉽게 핸들링할 수 있습니다.
- 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 WebSphere MQ의 기본 제공 데이터를 PCF 데이터로 변환할 수 있습니다. 이는 데이터 변환 엑시트를 작성할 필요가 없습니다. MQAI를 사용하여 WebSphere MQ를 관리하고 애플리케이션 간에 데이터를 교환하는 데 대한 자세한 정보는 18 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. 20 페이지의 『로컬 큐를 작성하기 위한 샘플 C 프로그램(amqsaicq.c)』
2. 단순 이벤트 모니터를 사용하여 화면에 이벤트를 표시합니다.23 페이지의 『이벤트 모니터를 사용하여 이벤트를 표시하는 샘플 C 프로그램(amqsaiem.c)』
3. 모든 로컬 큐 및 현재 용량 목록을 인쇄합니다.35 페이지의 『큐 및 인쇄 정보를 조회하기 위한 샘플 C 프로그램(amqsailq.c)』
4. 모든 채널 및 유형 목록을 인쇄합니다.30 페이지의 『채널 오브젝트를 조회하기 위한 샘플 C 프로그램(amqsaicl.c)』

MQAI 애플리케이션 빌드

MQAI를 사용하여 애플리케이션을 빌드하려면 WebSphere MQ에서 수행하는 동일한 라이브러리에 링크합니다. WebSphere MQ 애플리케이션을 빌드하는 방법에 대한 정보는 WebSphere MQ 애플리케이션 빌드 를 참조하십시오.

MQAI를 사용하여 WebSphere MQ를 구성하는 데 필요한 힌트와 팁

MQAI는 명령 서버 자체에서 직접 처리하지 않고 PCF 메시지를 사용하여 관리 명령을 명령 서버로 송신합니다. MQAI를 사용하여 WebSphere MQ를 구성하는 데 필요한 팁은 39 페이지의 『IBM WebSphere MQ 구성을 위한 힌트 및 팁』 에 있습니다.

MQAI(IBM WebSphere MQ Administration Interface)

IBM WebSphere MQ for 윈도우, AIX, Linux, HP-UX, and Solaris support the IBM WebSphere MQ Administration Interface (MQAI). MQAI는 PCF를 송신하고 수신하기 위해 MQI에 대한 대안을 주는 IBM WebSphere 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 서버로 PCF 메시지를 송신하고 응답을 기다림으로써 WebSphere MQ를 관리합니다.

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

로컬 큐를 작성하기 위한 샘플 C 프로그램(amqsaicq.c)

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

```
/*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICQ.C                                                  */
/*                                                                           */
/* Description:  Sample C program to create a local queue using the          */
/*               WebSphere 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, 2024.                         */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* 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 WebSphere 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,                    /* WebSphere 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.c은(는) MQAI를 사용하여 기본 이벤트 모니터를 보여줍니다.

```
*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAIEM.C                                                  */
/*                                                                           */
/* Description:  Sample C program to demonstrate a basic event monitor       */
/*               using the WebSphere MQ Admin Interface (MQAI).              */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 1999, 2024. 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 needs 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.c은(는) MQAI를 사용하여 채널 오브젝트를 조회합니다.

```
 /*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICL.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire channel objects                 */
/*               using the WebSphere MQ Administration Interface (MQAI)      */
/*                                                                           */
/* <N_OCO_COPYRIGHT>                                                         */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 2008, 2024. 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`은(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 WebSphere 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, 2024.                */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* 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 WebSphere 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,                   /* WebSphere 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 WebSphere MQ 구성을 위한 힌트 및 팁

MQAI 사용 시 프로그래밍 힌트 및 팁입니다.

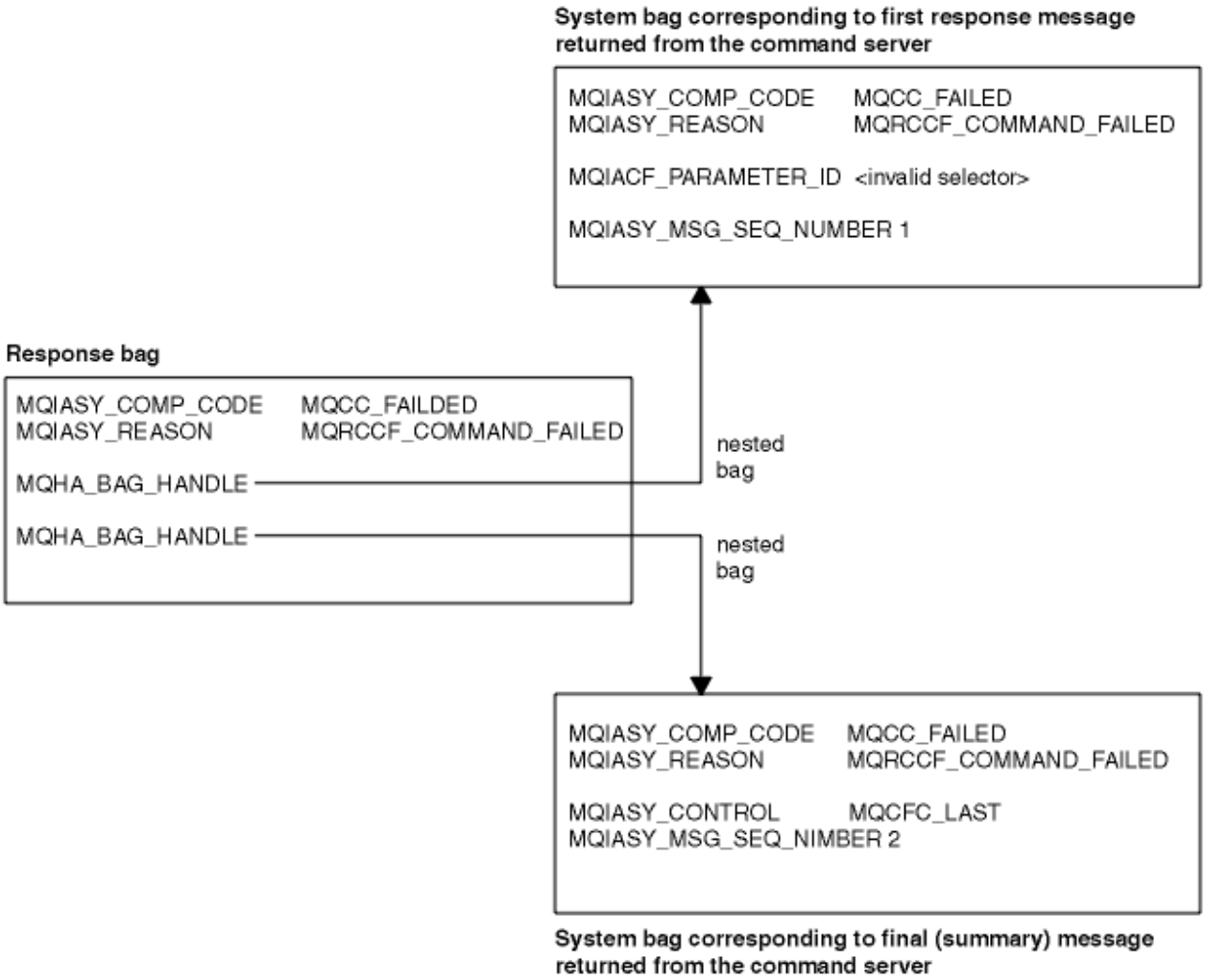
MQAI는 명령 서버 자체에서 직접 처리하지 않고 PCF 메시지를 사용하여 관리 명령을 명령 서버로 송신합니다. 다음은 MQAI를 사용하여 WebSphere MQ를 구성하는 데 대한 몇 가지 팁입니다.

- WebSphere MQ의 문자열은 고정 길이로 채워진 공백입니다. C를 사용하면, 보통 널(Null) 종료 문자열이 입력 매개변수로 WebSphere MQ 프로그래밍 인터페이스에 제공될 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 52 페이지의 『mqExecute 호출을 사용하여 명령 서버에 관리 명령 보내기』의 내용을 참조하십시오.

  다음 다이어그램에 이 시나리오가 표시됩니다.



고급 MQAI 주제

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 41 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 42 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 43 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 41 페이지의 그림 1에 표시된 대로 사용자 선택자, 시스템 선택자 또는 둘 다에 의한 색인 작성이 가능합니다.

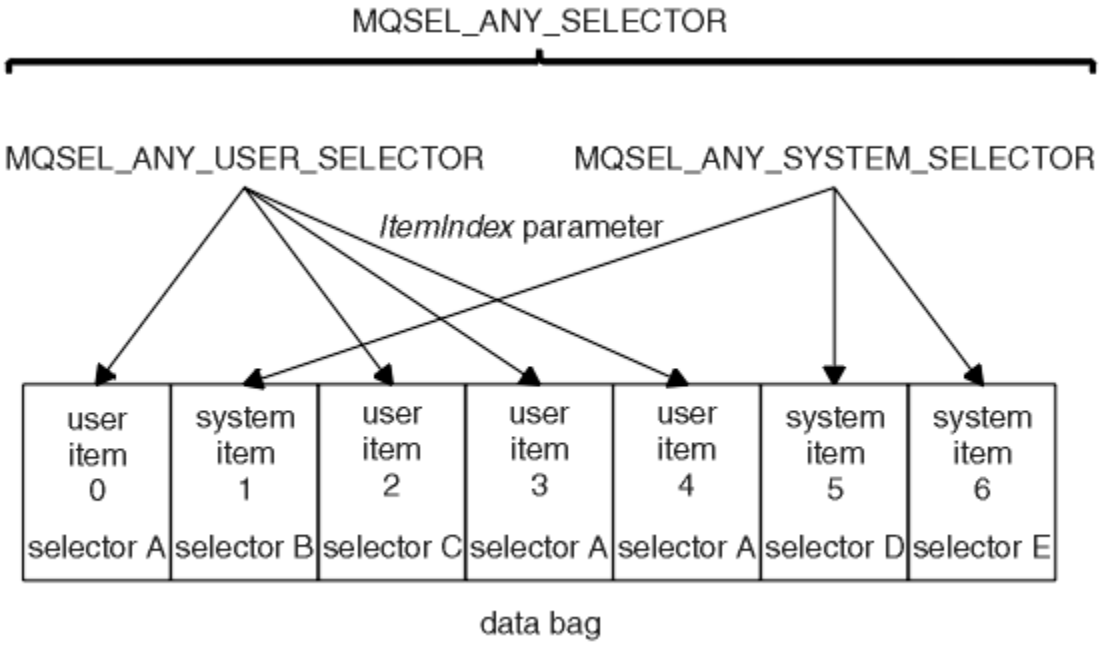


그림 *1.* 색인 작성

그림 41 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.

42 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	백 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	백 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	변경되지 않음(2)
mqPutBag	백 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 CCSID 설정(3)
mqGetBag	백 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 사용됨(4)	리턴된 데이터의 CCSID 설정(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

표 *2. CCSID* 처리 *(계속)*

MQAI 호출	**CCSID**	**호출할 입력**	**호출할 출력**

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 43 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

43 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 43 페이지의 표 5에 표시된 값을 사용합니다.

표 *5.* 메시지 디스크립터 값

매개변수	**가치**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE

표 5. 메시지 디스크립터 값 (계속)	
매개변수	**가치**
MsgType	43 페이지의 표 4 참조
Expiry	30초(44 페이지의 『1』 참고)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	CCSID 백에 따라 다름(44 페이지의 『2』 참고)
Format	43 페이지의 표 4 참조
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	44 페이지의 『3』 참조
ReplyToQMgr	공백
참고사항: 1. 이 값은 **OptionsBag** 매개변수를 사용하여 **mqExecute** 호출에서 대체될 수 있습니다. 이에 대한 정보는 **mqExecute**의 내용을 참조하십시오. 2. 42 페이지의 『MQAI에서 데이터 변환』의 내용을 참조하십시오. 3. **MQMT_REQUEST**유형의 메시지에 대한 사용자 지정 응답 큐 또는 MQAI 생성 임시 동적 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 47 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 41 페이지의 『MQAI의 색인 작성』에 설명되어 있습니다.

  이 개념의 계층은 그림 1에 표시되어 있습니다.

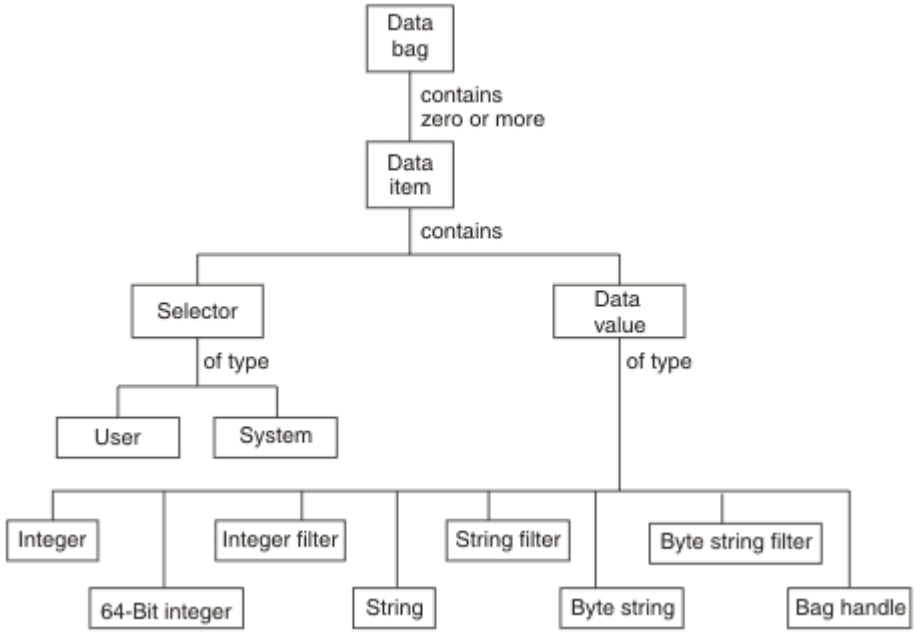


그림 *2. MQAI* 개념의 계층

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 서버로 송신하여 WebSphere MQ 오브젝트를 관리하는 데 사용되는 데이터에 맞게 작성된 백입니다. 관리 백은 46 페이지의 『데이터 백 작성 및 삭제』에 설명된 대로 특정 옵션을 자동으로 구현합니다.

명령 백(command bag)
WebSphere MQ 오브젝트를 관리하기 위한 명령을 위해 작성된 백입니다. 그러나, 관리 백과 달리, 명령 백은 이러한 옵션이 사용 가능할지라도 자동으로 특정 옵션을 의미하지는 않습니다. 옵션에 대한 자세한 정보는 46 페이지의 『데이터 백 작성 및 삭제』의 내용을 참조하십시오.

그룹 백
그룹화된 데이터 항목 세트에 사용되는 백. 그룹 백은 WebSphere MQ 오브젝트를 관리하는 데 사용할 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. 46 페이지의 『데이터 백 작성 및 삭제』
- 데이터 백을 사용하여 애플리케이션 사이에 데이터를 송신할 수 있습니다. 46 페이지의 『데이터 백 넣기 및 수신』
- 데이터 백에 데이터 항목을 추가할 수 있습니다. 47 페이지의 『백에 데이터 항목 추가』
- 데이터 백 내에서 조회 명령을 추가할 수 있습니다. 48 페이지의 『백에 조회 명령 추가』

- 데이터 백 내에서 조회할 수 있습니다. 49 페이지의 『데이터 백 내에서 조회』
- 데이터 백 내에서 데이터 항목을 셀 수 있습니다. 51 페이지의 『데이터 항목 계산』
- 데이터 백 내에서 정보를 변경할 수 있습니다. 49 페이지의 『백 내의 정보 변경』
- 데이터 백을 지울 수 있습니다. 50 페이지의 『mqClearBag 호출을 사용하여 백 지우기』
- 데이터 백을 자를 수 있습니다. 50 페이지의 『mqTruncateBag 호출을 사용하여 백 자르기』
- 백 및 버퍼를 변환할 수 있습니다. 51 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 47 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 52 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 11 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 항목을 사용하면 데이터 백 작성 시 이를 채울 수 있습니다. 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 47 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.

- 51 페이지의 『데이터 항목 계산』.
- 51 페이지의 『데이터 항목 삭제』.
- 47 페이지의 『백에 데이터 항목 추가』.
- 48 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(49 페이지의 『백 내의 정보 변경』의 내용 참조), 백 작성 옵션이 사용자에 의해 변경될 수 없습니다. 시스템 항목은 삭제할 수 없습니다 (51 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 47 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 48 페이지의 그림 3에 표시되어 있습니다. 항목은 선택자로 식

별됩니다. 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.

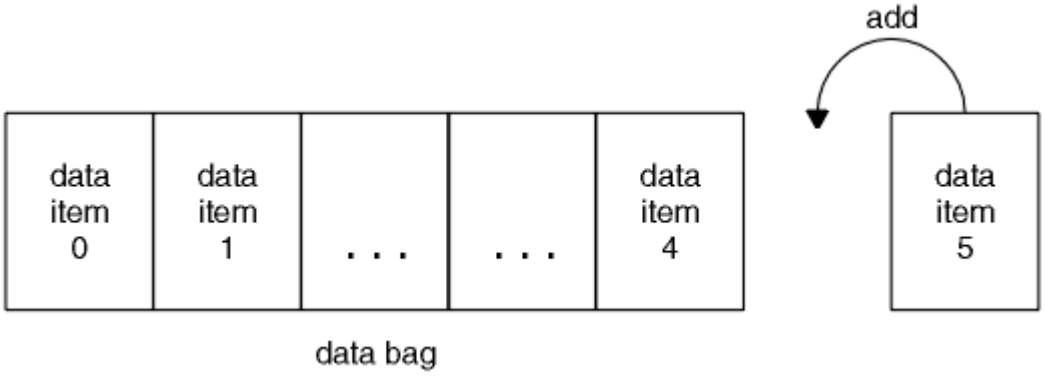


그림 3. 데이터 항목 추가

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 47 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. 호출을 사용하면 WebSphere MQ에서 조회할 속성의 선택자를 지정할 수 있습니다.

mqAddInquiry 호출에 대한 전체 설명은 mqAddInquiry를 참조하십시오.

데이터 항목 필터링 및 조회

MQAI를 사용하여 WebSphere MQ 오브젝트의 속성을 조회하는 경우, 두 가지 방식으로 프로그램에 리턴되는 데이터를 제어할 수 있습니다.

- mqAddInteger 및 mqAddString 호출을 사용하여 리턴되는 데이터를 **필터**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 사용하여 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 19 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(49 페이지의 그림 4의 내용 참조).

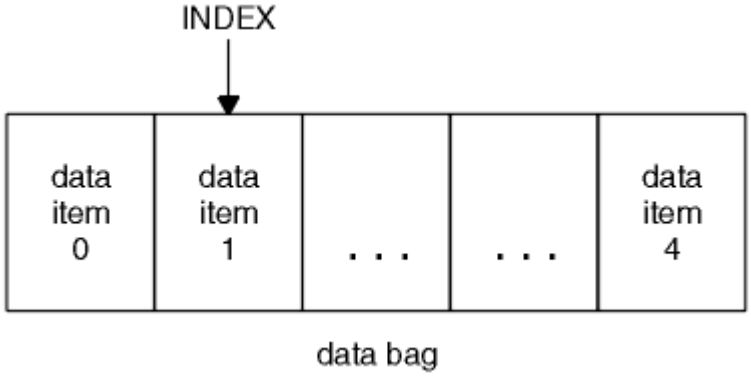


그림 *4.* 단일 데이터 항목 수정

2. 지정된 선택자의 기존 모든 발생을 삭제하고 백의 끝에 새로운 발생을 추가합니다. (50 페이지의 그림 5의 내용 참조.) 특수 색인 값은 바꾸려는 매개변수의 **모든** 인스턴스를 허용합니다.

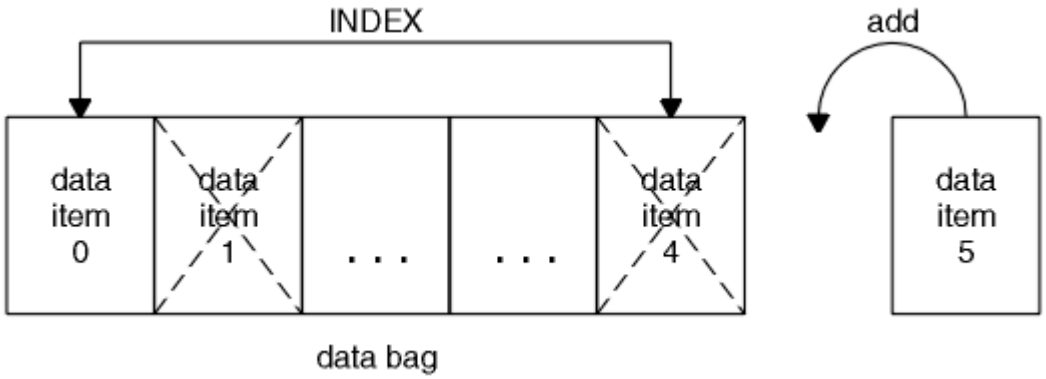


그림 5. 모든 데이터 항목 수정

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.

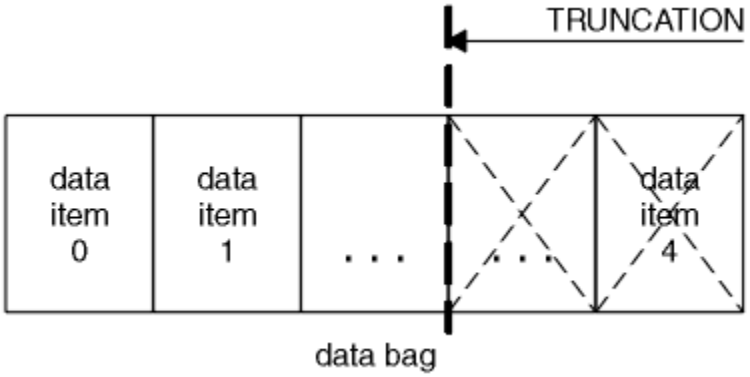


그림 6. 백 자르기

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 51 페이지의 그림 7에 표시되어 있습니다. mqBagToBuffer 호출에 대한 전체 설명은 mqBagToBuffer를 참조하십시오.

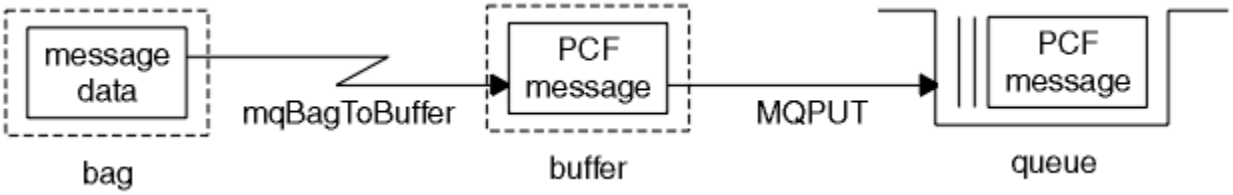


그림 *7. PCF* 메시지로 백 변환

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 51 페이지의 그림 8에 표시되어 있습니다. mqBufferToBag 호출에 대한 전체 설명은 mqBufferToBag을 참조하십시오.

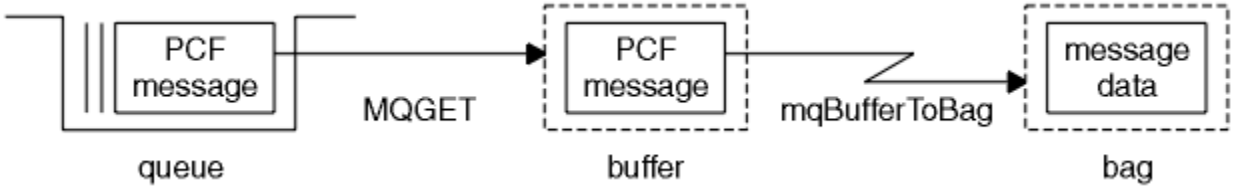


그림 *8.* 백 양식으로 *PCF* 메시지 변환

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 51 페이지의 『mqDeleteItem 호출을 사용하여 백에서 데이터 항목 삭제』의 내용을 참조하십시오.
- 백에서 **모든** 사용자 항목을 삭제합니다(백 지우기). 자세한 정보는 50 페이지의 『mqClearBag 호출을 사용하여 백 지우기』의 내용을 참조하십시오.
- 백의 끝에서 사용자 항목을 삭제하십시오(즉, 백 자르기). 자세한 정보는 50 페이지의 『mqTruncateBag 호출을 사용하여 백 자르기』의 내용을 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. (52 페이지의 그림 9의 내용 참조.)

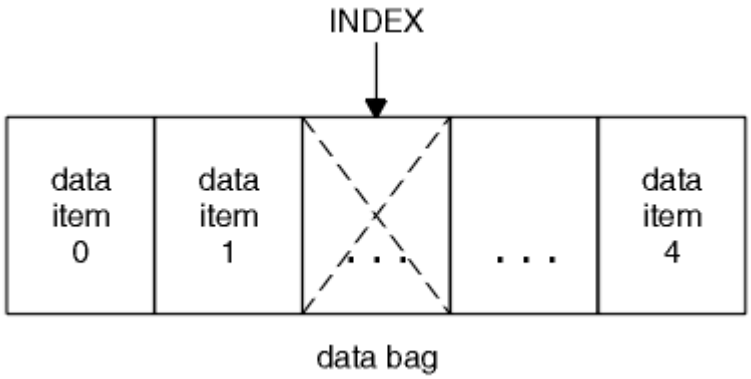


그림 *9.* 단일 데이터 항목 삭제

또는

2. 지정된 선택자의 모든 발생. (52 페이지의 그림 10의 내용 참조.)

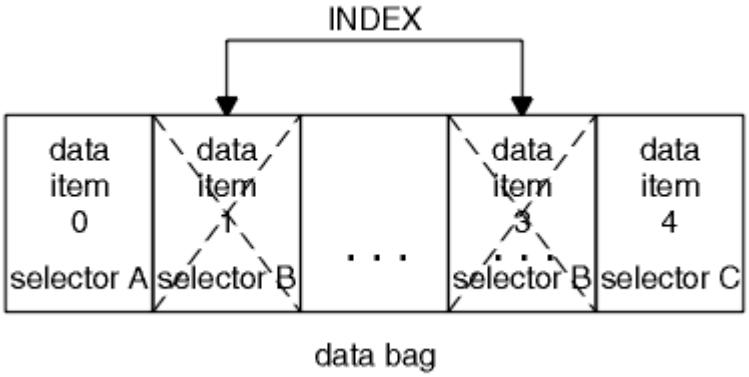


그림 *10.* 모든 데이터 항목 삭제

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. 이 응답에는 몇 가지 WebSphere MQ 오브젝트 또는 일련의 PCF 오류 응답 메시지 등과 관련된 속성 정보가 들어 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 53 페이지의 그림 11에 표시됩니다.

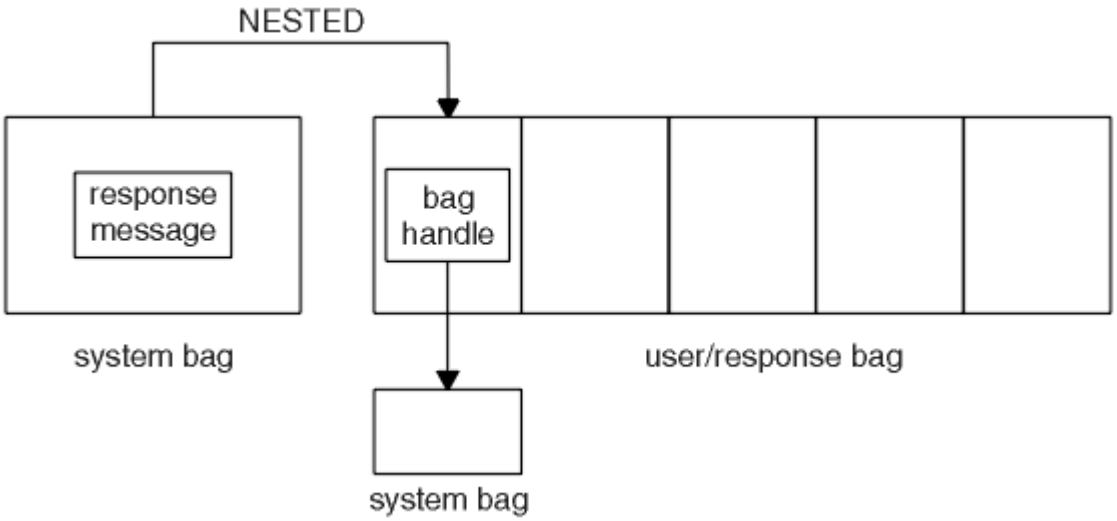


그림 *11.* 중첩

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.

IBM WebSphere MQ Explorer를 사용하여 관리

IBM WebSphere MQ Explorer를 사용하면 Windows 또는 Linux(x86 및 x86-64 플랫폼) 전용 컴퓨터에서 네트워크의 로컬 또는 리모트 관리를 수행할 수 있습니다.

Windows의 경우 IBM WebSphere MQ , Linux 용 IBM WebSphere MQ (x86 및 x86-64 플랫폼) 은 제어 또는 MQSC 명령을 사용하는 대신 관리 태스크를 수행하기 위해 IBM WebSphere MQ Explorer 라는 관리 인터페이스를 제공합니다. 명령 세트 비교에서는 IBM WebSphere MQ Explorer를 사용하여 수행할 수 있는 것을 표시합니다.

IBM WebSphere MQ Explorer에서는 관심있는 큐 관리자 및 클러스터에서 IBM WebSphere MQ Explorer를 지정함으로써 Windows 또는 Linux(x86-64 플랫폼)을 실행 중인 컴퓨터에서 네트워크의 로컬 또는 리모트 관리를 수행할 수 있습니다. IBM WebSphere MQ Explorer 를 사용하여 관리할 수 있는 IBM WebSphere MQ 의 플랫폼 및 레벨은 55 페이지의 『리모트 큐 관리자』에 설명되어 있습니다.

IBM WebSphere MQ Explorer 가 이를 관리할 수 있도록 원격 IBM WebSphere MQ 큐 관리자를 구성하려면 55 페이지의 『필수 소프트웨어 및 정의』 을 참조하십시오.

Windows 또는 Linux(x86 및 x86-64 플랫폼) 시스템 도메인 내에서 로컬로 또는 리모트로 IBM WebSphere MQ 의 작업 환경을 설정 및 세분화하는 것과 연관된 태스크를 수행할 수 있습니다.

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

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

IBM WebSphere MQ Explorer 를 사용하여 수행할 수 있는 작업

다음은 IBM WebSphere MQ Explorer를 사용하여 수행할 수 있는 태스크 목록입니다.

IBM WebSphere MQ 탐색기로 다음을 수행할 수 있습니다.

- 큐 관리자를 작성하고 삭제하십시오(로컬 시스템 전용).
- 큐 관리자를 시작하고 중지하십시오(로컬 시스템 전용).
- WebSphere MQ 오브젝트(예: 큐 및 채널)의 정의를 정의, 표시 및 대체합니다.
- 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.
- ikeyman GUI를 호출하여 SSL(secure sockets layer) 인증서를 관리하고 인증서를 큐 관리자와 연관시키고 인증서 저장소를 구성 및 설정하십시오(로컬 시스템 전용).
- WebSphere MQ 오브젝트에서 JMS 오브젝트를 작성하고, JMS 오브젝트에서 WebSphere MQ 오브젝트를 작성합니다.
- 현재 지원되는 모든 유형의 JMS 연결 팩토리를 작성합니다.
- 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.

- WebSphere MQ 자체에 관련된 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 WebSphere MQ 탐색기를 탐색합니다. 네비게이터를 사용하면 요청하는 컨텐츠 보기를 선택할 수 있습니다.

리모트 큐 관리자

연결할 수 있는 지원되는 큐 관리자에 두 가지 예외가 있습니다.

From a 윈도우 or Linux (x86 and x86-64 platforms) system, the WebSphere MQ Explorer can connect to all supported queue managers with the following exceptions:

- WebSphere MQ for z/OS queue managers earlier than Version 6.0.
- 현재 지원되는 MQSeries® V2 큐 관리자

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

If you intend to remotely administer a V6.0 or later queue manager on 윈도우 using the IBM WebSphere MQ Explorer on a WebSphere MQ V5.3 computer, you must install Fix Pack 9 (CSD9) or later on your WebSphere MQ for 윈도우 V5.3 computer.

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

IBM WebSphere MQ Explorer 사용 여부 결정

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

다음과 같은 점을 알고 있어야 합니다.

오브젝트 이름
MQSC 명령을 사용하여 오브젝트에 대해 작업할 때 IBM WebSphere MQ Explorer에서 큐 관리자 및 기타 오브젝트의 소문자 이름을 사용할 경우, 오브젝트 이름을 작은따옴표 안에 표시해야 합니다. 그렇지 않으면 WebSphere MQ가 이름을 인식하지 않습니다.

큰 큐 관리자
IBM WebSphere MQ Explorer는 작은 큐 관리자로 가장 잘 작동됩니다. 단일 큐 관리자에 많은 오브젝트가 있는 경우, WebSphere MQ Explorer가 보기에 표시하는 데 필요한 정보를 추출하는 시간이 지연될 수 있습니다.

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

IBM WebSphere MQ Explorer 설정

다음 절에서는 IBM WebSphere MQ Explorer를 설정하기 위해 수행해야 하는 단계를 간략하게 설명합니다.

- 55 페이지의 『필수 소프트웨어 및 정의』
- 56 페이지의 『보안』
- 59 페이지의 『큐 관리자와 클러스터 표시 및 숨기기』
- 60 페이지의 『클러스터 멤버십』
- 60 페이지의 『데이터 변환』

필수 소프트웨어 및 정의

IBM WebSphere MQ Explorer 사용을 시도하기 전에 다음 요구사항을 충족하는지 확인하십시오.

IBM WebSphere 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 WebSphere MQ 리스너이거나 UNIX and Linux 시스템에서는 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 DEFINE CHANNEL 을 참조하십시오.
4. 시스템 큐(SYSTEM.MQEXPLORER.REPLY.MODEL)가 존재해야 합니다.

보안

특정 오브젝트에 대한 사용자 액세스 제어가 중요한 환경에서 WebSphere MQ를 사용하는 경우, IBM WebSphere MQ 탐색기를 사용하는 보안 측면을 고려해야 합니다.

IBM WebSphere MQ Explorer 를 사용할 수 있는 권한

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

WebSphere MQ Explorer를 사용하여 로컬 관리 태스크를 수행하려면 사용자가 관리 태스크를 수행하는 데 필요한 권한이 있어야 합니다. 사용자가 mqm 그룹의 구성원인 경우, 사용자에게 모든 로컬 관리 태스크를 수행하기 위한 권한이 있습니다.

WebSphere MQ Explorer를 실행하는 사용자가 WebSphere MQ Explorer를 사용하여 리모트 큐 관리자에 연결하고 원격 관리 태스크를 수행하려면 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).

To connect to a remote queue manager on WebSphere MQ for z/OS and perform remote administrative tasks using the IBM WebSphere MQ Explorer, the following must be provided:

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

그 밖에 WebSphere MQ Explorer를 실행하는 사용자는 다음 권한을 가지고 있어야 합니다.

- 레이프 UPDATE authority to the system queue, SYSTEM.MQEXPLORER.REPLY.MODEL
- 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

WebSphere MQ 오브젝트에 권한을 부여하는 방법에 대한 정보는 UNIX 또는 Linux 시스템 및 Windows에서 WebSphere MQ 오브젝트에 대한 액세스 부여의 내용을 참조하십시오.

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

WebSphere MQ Explorer의 기본 필터는 모든 WebSphere MQ 오브젝트를 표시하기 위한 것입니다. 사용자가 DISPLAY 권한을 가지고 있지 않은 WebSphere MQ 오브젝트가 있을 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 WebSphere MQ Explorer와 각 리모트 큐 관리자 사이의 채널을 보안화해야 합니다.

IBM WebSphere 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.

*WebSphere MQ Explorer*에서 보안 엑시트 사용

WebSphere MQ Explorer를 사용하여 기본 보안 엑세트 및 큐 관리자에 특정되는 보안 엑시트를 지정할 수 있습니다.

WebSphere MQ Explorer에서 새로운 모든 클라이언트 연결에 사용할 수 있는 기본 보안 엑시트를 정의할 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. WebSphere MQ Explorer를 사용하여 엑시트를 지정하십시오. 자세한 정보는 WebSphere MQ 도움말 센터를 참조하십시오.

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

IBM WebSphere 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 WebSphere MQ MQI 클라이언트의 개요를 참조하십시오.

클라이언트 채널 정의 테이블을 사용하여 채널을 설정한 경우 57 페이지의 『리모트 큐 관리자를 호스트하는 시스템의 태스크』 및 58 페이지의 『IBM WebSphere MQ Explorer를 호스트하는 시스템의 태스크』에 설명된 대로 IBM WebSphere MQ Explorer를 사용하여 SSL 사용 가능 MQI 채널을 통해 리모트 큐 관리자에 연결할 수 있습니다.

리모트 큐 관리자를 호스트하는 시스템의 태스크

리모트 큐 관리자를 호스트하는 시스템에서 다음 태스크를 수행하십시오.

1. 서버 연결 및 채널의 클라이언트 연결 쌍을 정의하고 양쪽 채널의 서버 연결에서 *SSLCIPH* 변수에 대한 적절한 값을 지정하십시오. *SSLCIPH* 변수에 대한 자세한 정보는 SSL로 채널 보호를 참조하십시오.
2. 큐 관리자의 `@ipcc` 디렉토리에 있는 채널 정의 테이블 `AMQCLCHL.TAB`를 IBM WebSphere MQ Explorer를 호스트 중인 시스템으로 보내십시오.
3. 지정된 포트에서 TCP/IP 리스너를 시작하십시오.
4. 큐 관리자의 SSL 디렉토리에 CA 및 개인 SSL 인증서 모두를 배치하십시오.
   - UNIX and Linux 시스템의 경우 `/var/mqm/qmgrs/+QMNAME+/SSL`
   - Windows 시스템의 경우 `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL`

여기서, +QMNAME+은(는) 큐 관리자의 이름을 나타내는 토큰입니다.

5. 이름이 key.kdb인 CMS 유형의 키 데이터베이스 파일을 작성하십시오. iKeyman GUI에서 옵션을 선택하거나 -stash 옵션을 **runmqckm** 명령과 함께 사용하여 파일의 비밀번호를 스택하십시오.
6. 이전 단계에서 작성된 키 데이터베이스에 CA 인증서를 추가하십시오.
7. 큐 관리자에 대한 개인 인증서를 키 데이터베이스로 가져오십시오.

Windows 시스템에서 SSL (Secure Sockets Layer) 작업에 대한 자세한 정보는 UNIX, Linux 및 윈도우 시스템에서 SSL 또는 TLS에 대한 작업 의 내용을 참조하십시오.

IBM WebSphere MQ Explorer를 호스트하는 시스템의 태스크

IBM WebSphere MQ Explorer를 호스트하는 시스템에서 다음 태스크를 수행하십시오.

1. 이름이 key.jks인 JKS 유형의 키 데이터베이스 파일을 작성하십시오. 이 키 데이터베이스 파일에 대한 비밀번호를 설정하십시오.

   IBM WebSphere MQ Explorer는 SSL 보안을 위한 Java 키 저장소 파일(JKS)을 사용하고 IBM WebSphere 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 and Linux 시스템에서 시스템 메뉴, MQExplorer 실행 파일 또는 **strmqcfg** 명령을 사용하여 MQ Explorer를 시작하십시오.
5. IBM WebSphere MQ Explorer 도구 모음에서 **창 - > 환경 설정**을 클릭한 후 **WebSphere MQ 탐색기** 를 펼치고 **SSL 클라이언트 인증서 저장소**를 클릭하십시오. 신뢰성있는 인증서 저장소 및 개인 인증서 저장소 모두에 58 페이지의 『IBM WebSphere 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 57 페이지의 『리모트 큐 관리자를 호스트하는 시스템의 태스크』 의 2단계에서 리모트 큐 관리자로부터 전송한 채널 테이블 파일의 위치를 지정하십시오.
9. **마침**을 클릭하십시오. 이제 IBM WebSphere MQ Explorer에서 리모트 큐 관리자에 액세스할 수 있습니다.

다른 큐 관리자를 통한 연결

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

이 경우, IBM WebSphere 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 분석 을 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.

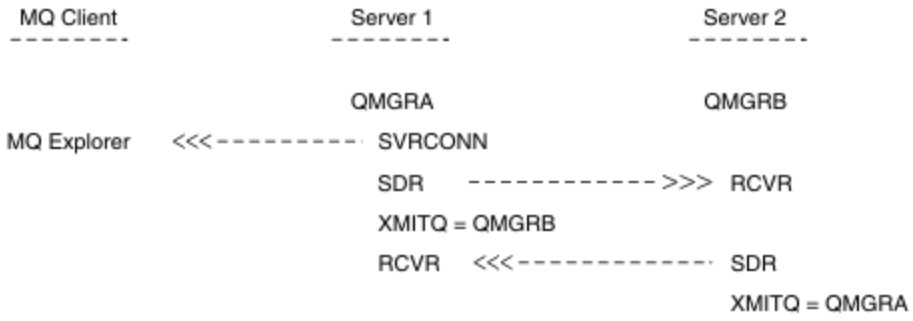
대상 큐 관리자의 명령 서버는 메시지 디스크립터(MQMD)에서 *UserIdentifier* 매개변수에 사용자 ID를 지정하여 메시지를 송신 큐에 넣습니다. 이러한 넣기가 성공하기 위해서는 대상 큐 관리자에 넣기 권한이 있는 사용자 ID가 이미 존재해야 합니다.

다음 예제는 중간 큐 관리자를 통해 WebSphere MQ Explorer에 큐 관리자를 연결하는 방법을 보여줍니다.

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 WebSphere 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 WebSphere MQ 탐색기에 연결될 수 있습니다.
- When connecting to QMGRB with WebSphere MQ Explorer, select QMGRA as the intermediate queue manager

이 상황에서는 IBM WebSphere 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. 101 페이지의 『원격 관리를 위한 채널 및 전송 큐 준비』 의 내용을 참조하십시오.

큐 관리자와 클러스터 표시 및 숨기기

IBM WebSphere MQ 탐색기는 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. Queue Managers 트리 노드를 마우스 오른쪽 단추로 누른 다음 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.

IBM WebSphere MQ 탐색기가 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. WebSphere MQ 탐색기의 탐색 분할창에서 맨 위 **IBM WebSphere MQ** 트리 노드를 마우스 오른쪽 단추로 누른 후 **MQ 탐색기 설정 내보내기** 를 선택하십시오.
3. **MQ 탐색기 > MQ 탐색기 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. WebSphere MQ 탐색기의 탐색 분할창에서 맨 위 **IBM WebSphere MQ** 트리 노드를 마우스 오른쪽 단추로 누른 후 **MQ 탐색기 설정 가져오기** 를 선택하십시오.
2. **MQ 탐색기 > MQ 탐색기 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, **마침**을 클릭하여 WebSphere MQ Explorer로 리모트 큐 관리자 연결 정보를 가져오십시오.

클러스터 멤버십

IBM WebSphere MQ Explorer에는 클러스터의 멤버인 큐 관리자에 대한 정보가 필요합니다.

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

IBM WebSphere 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 WebSphere MQ Explorer를 유지보수해야 합니다. 이를 수행하려면 WebSphere MQ Explorer에 다음 정보가 필요합니다.

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

이 정보를 사용하여 WebSphere 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. WebSphere MQ Explorer는 대체 저장소로 자동 전환하지 않습니다.
- 선택된 저장소가 TCP/IP를 통해 접속될 수 없습니다.
- 선택한 저장소가 WebSphere MQ Explorer가 지원하지 않는 플랫폼 및 명령 레벨에서 실행 중인 큐 관리자에서 실행 중입니다.

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

데이터 변환

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

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

지원되는 변환이 코드 페이지 변환에서 설명됩니다.

Windows의 보안

WebSphere MQ 준비 마법사는 특수 사용자 계정을 작성하여 Windows 서비스가 이를 사용해야 하는 프로세스에 공유할 수 있도록 합니다.

Windows 서비스는 IBM WebSphere MQ 설치를 위한 클라이언트 프로세스 사이에 공유됩니다. 한 가지 서비스가 각 설치를 위해 작성됩니다. 각 서비스의 이름은 `MQ_InstallationName`이고 표시 이름은 `IBM WebSphere MQ(InstallationName)`입니다. 서버에 하나의 설치만 있는 Version 7.1이전의 Windows 서비스에는 표시 이름 `IBM MQSeries`이 포함된 `MQSeriesServices` 이 (가) 지정되었습니다.

각 서비스가 비대화식 및 대화식 로그온 세션 사이에 공유되어야 하기 때문에, 특수 사용자 계정에서 각각을 실행해야 합니다. 모든 서비스에 대해 하나의 특수 사용자 계정을 사용하거나 다른 특수 사용자 계정을 작성할 수 있습니다. 각 특수 사용자 계정에는 "서비스로서 로그온"에 대한 사용자 권한이 있어야 합니다. 자세한 정보는 61 페이지의 『IBM WebSphere MQ Windows 서비스에 필요한 사용자 권한』의 내용을 참조하십시오. 사용자 ID에 서비스를 실행할 권한이 없으면 서비스가 시작되지 않고 Windows 시스템 이벤트 로그에서 오류가 리턴됩니다. 일반적으로 IBM WebSphere MQ 준비 마법사를 실행하고 사용자 ID를 올바르게 설정합니다. 그러나 사용자 ID를 수동으로 구성한 경우 해결해야 할 문제가 있을 수 있습니다.

IBM WebSphere MQ를 설치하고 IBM WebSphere MQ 준비 마법사를 처음 실행할 때, "서비스로 로그온"을 포함하여 필수 설정 및 권한으로 MUSR_MQADMIN이라는 서비스에 대한 로컬 사용자 계정을 작성합니다.

후속 설치의 경우, IBM WebSphere MQ 준비 마법사는 MUSR_MQADMINx라는 사용자 계정을 작성합니다, 여기서 x는 존재하지 않는 사용자 ID를 표시하는 다음에 사용 가능한 숫자입니다. MUSR_MQADMINx에 대한 비밀번호는 계정이 작성될 때 무작위로 생성되고 서비스에 대한 로그온 환경을 구성하기 위해 사용됩니다. 생성된 비밀번호는 만기되지 않습니다.

이 IBM WebSphere MQ 계정은 특정 기간 이후에 계정 비밀번호가 변경되도록 요청하기 위해 시스템에 설정되는 계정 정책에 의해 영향을 받지 않습니다.

비밀번호는 이 일회성 처리 외부에는 알려지지 않고 레지스트리의 보안 파트에 Windows 운영 체제에 의해 저장됩니다.

Active Directory 사용(Windows 전용)

사용자 계정이 Active Direcory를 사용 중인 도메인 제어기에 정의되어 있는 일부 네트워크 구성에서는 IBM WebSphere MQ가 실행 중인 로컬 사용자 계정에 기타 도메인 사용자 계정의 그룹 멤버십을 조회하는 데 필요한 권한이 없을 수 있습니다. IBM WebSphere MQ 준비 마법사는 네트워크 구성에 대한 사용자 질문을 하고 테스트를 실행하는 경우인지 여부를 식별합니다.

IBM WebSphere MQ가 실행 중인 로컬 사용자 계정에 필요한 권한이 없으면, IBM WebSphere MQ 준비 마법사가 특정 사용자 권한을 가진 도메인 사용자 계정의 계정 세부사항을 입력하도록 프롬프트를 표시합니다. 도메인 사용자 계정이 요구하는 사용자 권한의 경우61 페이지의 『IBM WebSphere MQ Windows 서비스에 필요한 사용자 권한』의 내용을 참조하십시오. 사용자가 도메인 사용자 계정에 대한 올바른 계정 세부사항을 IBM WebSphere MQ 준비 마법사에 입력할 때, 새 계정에서 실행하기 위한IBM WebSphere MQ Windows 서비스를 구성합니다. 계정 세부사항이 레지스트리의 보안 부분에 보유되고 사용자는 읽을 수 없습니다.

서비스가 실행 중일 때, IBM WebSphere MQ Windows 서비스가 실행되고 서비스가 실행 중인 한 실행으로 남습니다. Windows 서비스가 실행된 이후 서버에 로그온하는 IBM WebSphere MQ 관리자는 IBM WebSphere MQ Explorer를 사용하여 서버에서 큐 관리자를 관리할 수 있습니다. 이는 IBM WebSphere MQ Explorer를 기존 Windows 서비스 프로세스에 연결합니다. 이러한 두 개의 조치가 작동하려면 다른 레벨의 권한이 필요합니다.

- 시작 프로세스에는 실행 권한이 필요합니다.
- IBM WebSphere MQ 관리자에는 액세스 권한이 필요합니다.

IBM WebSphere MQ Windows 서비스에 필요한 사용자 권한

이 주제의 표는 IBM WebSphere MQ 설치를 위해 Windows 서비스가 실행되는 로컬 및 도메인 사용자 계정에 필요한 사용자 권한을 나열합니다.

배치 작업으로 로그온	이 사용자 계정에서 IBM WebSphere MQ Windows 서비스를 실행할 수 있습니다.
서비스로 로그온	사용자가 IBM WebSphere MQ Windows 서비스를 설정하여 구성된 계정으로 로그온합니다.
시스템 종료	IBM WebSphere MQ Windows 서비스가 서비스의 복구가 실패할 때 수행하도록 하기 구성되면 서버를 다시 시작할 수 있게 허용합니다.

쿼터 늘리기	운영 체제 `CreateProcessAsUser` 호출에 필요합니다.
운영 체제의 일부로 수행	운영 체제 `LogonUser` 호출에 필요합니다.
통과 검사 우회	운영 체제 `LogonUser` 호출에 필요합니다.
프로세스 레벨 토큰 대체	운영 체제 `LogonUser` 호출에 필요합니다.

참고: ASP 및 IIS 애플리케이션을 실행하는 환경에서는 디버그 프로그램 권리가 필요할 수 있습니다.

도메인 사용자 계정은 로컬 보안 정책 애플리케이션에 나열된 것과 같은 효력의 사용자 권한으로 이 Windows 사용자 권한을 설정해야 합니다. 그렇지 않으면, 서버에서 로컬 보안 정책 애플리케이션을 로컬로 사용하거나 도메인 보안 애플리케이션 도메인을 사용하여 이를 설정하십시오.

IBM WebSphere MQ 서비스와 연관된 사용자 이름 변경

IBM WebSphere MQ Service와 연관된 사용자 이름을 MUSR_MQADMIN에서 다른 것으로 변경해야 할 수도 있습니다. (예를 들어 큐 관리자가 8자 이상의 사용자 이름을 승인하지 않는 DB2®와 연관되어 있는 경우 이 조작을 수행해야 합니다.)

프로시저

1. 새 사용자 계정을 작성하십시오(예: **NEW_NAME**).
2. IBM WebSphere MQ 준비 마법사를 사용하여 새 사용자 계정의 자세한 내용을 입력하십시오.

IBM WebSphere MQ Windows 서비스 사용자 계정의 비밀번호 변경

이 태스크 정보

IBM WebSphere MQ Windows 서비스 로컬 사용자 계정의 비밀번호를 변경하려면 다음 단계를 수행하십시오.

프로시저

1. 서비스가 실행 중인 사용자를 식별하십시오.
2. 컴퓨터 관리 패널에서 IBMIBM WebSphere MQ 서비스를 중지하십시오.
3. 개인의 비밀번호를 변경하는 동일한 방법으로 필수 비밀번호를 변경하십시오.
4. 컴퓨터 관리 패널에서 IBM WebSphere MQ 서비스 특성으로 이동하십시오.
5. **로그온** 페이지를 선택하십시오.
6. 지정된 계정 이름이 비밀번호가 수정된 사용자와 일치하는지 확인하십시오.
7. 비밀번호를 **비밀번호** 및 **비밀번호 확인** 필드에 입력하고 **확인**을 클릭하십시오.

도메인 사용자 계정으로 실행 중인 설치의 *IBM WebSphere MQ Windows* 서비스

이 태스크 정보

설치를 위한 IBM WebSphere MQ Windows 서비스가 도메인 사용자 계정에서 실행 중인 경우, 다음과 같이 계정에 대한 비밀번호를 변경할 수도 있습니다.

프로시저

1. 도메인 제어기에서 도메인 계정에 대한 비밀번호를 변경하십시오. 도메인 관리자가 이를 수행하도록 요청해야 할 수도 있습니다.
2. 단계에 따라 IBMIBM WebSphere MQ 서비스에 대한 **로그온** 페이지를 수정하십시오.

   IBM WebSphere MQ Windows 서비스가 실행되는 사용자 계정은 사용자 인터페이스 애플리케이션에서 실행되거나 시스템 시동, 시스템 종료 또는 서비스 복구에서 자동으로 수행되는 MQSC 명령을 실행합니다. 따라서 이 사용자 계정에는 IBM WebSphere MQ 관리 권한이 있어야 합니다. 기본적으로 서버의 로컬 **mqm** 그룹에 추가됩니다. 이 멤버십을 제거하면 IBM WebSphere MQ Windows 서비스가 작동하지 않습니다. 사용자

권한에 대한 자세한 정보는 61 페이지의 『IBM WebSphere MQ Windows 서비스에 필요한 사용자 권한』의 내용을 참조하십시오

보안 문제점이 IBM WebSphere MQ Windows 서비스가 실행되는 사용자 계정과 함께 발생하면 오류 메시지 및 설명이 시스템 이벤트 로그에 나타납니다.

관련 개념

61 페이지의 『Active Directory 사용(Windows 전용)』
사용자 계정이 Active Direcory를 사용 중인 도메인 제어기에 정의되어 있는 일부 네트워크 구성에서는 IBM WebSphere MQ가 실행 중인 로컬 사용자 계정에 기타 도메인 사용자 계정의 그룹 멤버십을 조회하는 데 필요한 권한이 없을 수 있습니다. IBM WebSphere MQ 준비 마법사는 네트워크 구성에 대한 사용자 질문을 하고 테스트를 실행하는 경우인지 여부를 식별합니다.

IBM WebSphere MQ를 자원 관리자로 Db2와 통합

IBM WebSphere MQ Explorer로부터 사용자의 큐 관리자를 시작하거나, IBM WebSphere MQ V7을 사용 중이고 Db2를 통합할 때 문제가 발생하는 경우, 사용자의 큐 관리자 오류 로그를 확인하십시오.

다음과 같이 오류에 대한 큐 관리자 오류 로그를 검사하십시오.

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
    for xa_open. The queue manager is continuing without this resource manager.
```

설명: IBM WebSphere MQ 서비스 프로세스 `amqsvc.exe`를 실행하는 사용자 ID(기본 이름: MUSR_MQADMIN)가 DB2USERS 그룹에 대한 그룹 멤버십 정보를 포함하지 않은 액세스 토큰을 사용하여 계속 실행 중입니다.

해결: IBM WebSphere MQ 서비스 사용자 ID가 DB2USERS의 구성원인지 확인한 후 다음과 같은 순서로 명령을 사용하십시오.

- 서비스를 중지하십시오.
- 동일한 사용자 ID에서 실행되는 기타 프로세스를 중지하십시오.
- 이러한 프로세스를 재시작하십시오.

시스템을 다시 시작하면 위 단계가 확인되지만 반드시 다시 시작할 필요는 없습니다.

IBM WebSphere MQ Explorer 확장

IBM WebSphere MQ for 윈도우, and IBM WebSphere MQ for Linux (x86 and x86-64 platforms) provide an administration interface called the IBM WebSphere MQ Explorer to perform administration tasks as an alternative to using control or MQSC commands.

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

IBM WebSphere MQ Explorer는 Eclipse 프레임워크 및 Eclipse가 지원하는 기타 플러그인 애플리케이션과 일치하는 스타일로 정보를 표시합니다.

IBM WebSphere MQ Explorer 확장을 통해, 시스템 관리자는 WebSphere MQ 탐색기를 사용자 정의하는 기능이 있어 WebSphere MQ를 관리하는 방식을 개선할 수 있습니다.

자세한 정보는 IBM WebSphere MQ 탐색기 제품 문서에서 *IBM WebSphere MQ* 탐색기 확장 을 참조하십시오.

IBM WebSphere MQ 작업 표시줄 애플리케이션 사용(Windows 전용)

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

Windows에서 WebSphere MQ 아이콘은 서버의 시스템 트레이에 있으며 색상 코드화된 상태 기호로 대체되어 다음의 의미 중 하나를 나타낼 수 있습니다.

초록색
올바르게 작업 중이며 현재 경보가 작동하지 않음

파랑색
WebSphere MQ가 시작 중이거나 시스템 종료 중인지 확실치 않음

노랑색
경보; 하나 이상의 서비스가 실패 중이거나 이미 실패했음

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

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

IBM WebSphere MQ 경보 모니터 애플리케이션(Windows 전용)

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

경보 모니터는 WebSphere MQ 서버의 로컬 설치의 현재 상태에 대한 정보를 표시합니다. 또한 Windows ACPI(Advanced Configuration and Power Interface)를 모니터하고 ACPI 설정이 적용되는지 확인합니다.

WebSphere MQ 경보 모니터에서 다음을 수행할 수 있습니다.

- IBM WebSphere MQ 탐색기에 대한 직접적인 액세스
- 모든 미해결 경로와 관련하는 정보 보기
- 로컬 시스템의 WebSphere MQ 서비스 시스템 종료
- 네트워크를 통해 구성 가능한 사용자 계정 또는 Windows 워크스테이션이나 서버로 경보 메시지를 라우트함

로컬 IBM WebSphere MQ 오브젝트 관리

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

이 절에서 자세히 설명된 방법 외에도 IBM WebSphere MQ Explorer를 사용하여 로컬 WebSphere MQ 오브젝트를 관리할 수 있습니다. 53 페이지의 『IBM WebSphere 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
```

On WebSphere MQ for 윈도우 and WebSphere MQ for Linux (x86 and x86-64 platforms) systems, you can start a queue manager as follows:

1. IBM WebSphere MQ Explorer를 여십시오.
2. 네비게이터 보기에서 큐 관리자를 선택하십시오.
3. `Start`을(를) 클릭하십시오. 큐 관리자가 시작됩니다.

큐 관리자가 시작되는 데 몇 초 이상 걸리면 WebSphere MQ가 시작 진행 상태를 자세히 표시하는 정보 메시지를 간헐적으로 발행합니다.

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

큐 관리자 자동 시작

Windows용 WebSphere MQ에서는 WebSphere MQ Explorer를 사용하여 시스템이 시작될 때 큐 관리자를 자동으로 시작할 수 있습니다. 자세한 정보는 53 페이지의 『IBM WebSphere 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, `QMB`라는 큐 관리자를 중지하려면 다음 명령을 입력하십시오.

```
endmqm QMB
```

On WebSphere MQ for 윈도우 and WebSphere MQ for Linux (x86 and x86-64 platforms) systems, you can stop a queue manager as follows:

1. IBM WebSphere MQ Explorer를 여십시오.
2. 네비게이터 보기에서 큐 관리자를 선택하십시오.
3. `Stop...`을(를) 클릭하십시오. 큐 관리자 종료 패널이 표시됩니다.
4. 제어됨 또는 즉시를 선택하십시오.
5. `OK`을(를) 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 동등합니다.)

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 큐 관리자 작성 및 관리를 참조하십시오.

Windows에서 큐 관리자 중지

윈도우에 대한 IBM WebSphere MQ 의 큐 관리자를 중지하기 위해 프로세스 및 IBM WebSphere MQ 서비스를 종료하는 방법.

To stop a queue manager running under WebSphere MQ for 윈도우:

1. Windows 태스크 관리자를 사용하여 실행 중인 프로세스의 이름(ID)을 나열하십시오.
2. 다음 순서로 태스크 또는 **taskkill** 명령을 사용하여 프로세스를 종료하십시오 (실행 중인 경우).

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)
AMQZDMAA	지연된 메시지 프로세서
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
AMQZTRCN	추적

3. 제어판 **관리 도구** > **서비스** 에서 WebSphere MQ 서비스를 중지하십시오.
4. 모든 방법을 시도했지만 큐 관리자가 중지되지 않으면 시스템을 다시 시작하십시오.

Windows 태스크 관리자 및 **tasklist** 명령은 태스크에 관해 제한된 정보를 제공합니다. 특정 큐 관리자와 관련된 프로세스를 판별하는 데 도움이 되는 자세한 정보는 Microsoft웹 사이트 (https://www.microsoft.com) 에서 다운로드할 수 있는 *Process Explorer* (procexp.exe) 와 같은 도구 사용을 고려하십시오.

UNIX and Linux 시스템에서 큐 관리자 중지

UNIX and Linux에 대한 IBM WebSphere MQ 의 큐 관리자를 중지하기 위해 프로세스 및 IBM WebSphere MQ 서비스를 종료하는 방법. 큐 관리자를 중지하고 제거하기 위한 표준 메소드가 실패하는 경우 여기에 설명된 메소드를 시도할 수 있습니다.

UNIX and Linux 용 WebSphere MQ 에서 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.

   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)
amqzdmaa	지연된 메시지 프로세서
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 FFST가 취해질 수 있으며 FDC 파일은 `/var/mqm/errors.` 에 있습니다. 이를 큐 관리자의 결함으로 간주하지 마십시오.

큐 관리자는 이 메소드를 사용하여 이를 중지한 이후에도 일반적으로 재시작됩니다.

MQSC 명령을 사용하여 로컬 관리 태스크 수행

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

If you use IBM WebSphere MQ for 윈도우 or IBM WebSphere MQ for Linux (x86 and x86-64 platforms), you can also perform the operations described in this section using the IBM WebSphere MQ Explorer. 자세한 정보는 53 페이지의 『IBM WebSphere 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 절에서는 큐 관리자, 큐 및 프로세스 정의를 다룹니다. 채널, 클라이언트 연결 채널 및 리스너 오브젝트 관리에 대한 정보는 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 수행하거나 표준 입력 디바이스(`stdin`)를 경로 재지정하여 ASCII 텍스트 파일에서 일련의 명령을 실행할 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 및 z/OS를 포함하여 다른 플랫폼에서 사용할 수 있습니다.

MQSC 명령은 MQSC 참조 절의 토픽 콜렉션에 요약되어 있습니다.

스크립트(MQSC) 명령

MQSC 명령은 WebSphere MQ 플랫폼에서 사람이 이해할 수 있는 명령을 발행하는 균일한 방법을 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.
- 키워드, 괄호 및 값을 공백 및 쉼표 수로 분리할 수 있습니다. 구문 다이어그램에 표시된 쉼표는 항상 하나 이상의 공백으로 대체될 수 있습니다. 각 매개변수 바로 앞(1차 매개변수 뒤)에 공백이 하나 이상 있어야 합니다.
- 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.
  - 별표로 끝나는 일반 값
  - 단일 별표(예: TRACE(*))
  - 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.

- 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 WebSphere 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.

MQSC 명령의 전체 목록은 MQSC 명령을 참조하십시오.

관련 태스크
명령 스크립트 빌드

WebSphere 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. 일반적으로 로컬 큐의 이름에는 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 MQSC 참조를 참조하십시오.

runmqsc 명령 호출은 모든 WebSphere MQ 제어 명령과 마찬가지로 일부 WebSphere 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 WebSphere 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.manager`에서 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: WebSphere 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 올바로 입력하지 않은 경우, MQSC 참조에서 올바른 구문을 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 stdin을 경로 재지정하는 방법을 고려하십시오.

70 페이지의 『표준 입력 및 출력』 에는 stdin 및 stdout에 대한 정보가 포함됩니다. 텍스트 파일에서 stdin의 경로를 재지정하려면 일반 텍스트 편집기를 사용하여 MQSC 명령을 포함하는 텍스트 파일을 먼저 작성하십시오. 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. 입력에 대한 MQSC 명령이 포함된 파일을 *MQSC* 명령 파일이라고 합니다. 큐 관리자의 응답이 포함된 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 12 MQSC 명령 파일 myprog.in 및 74 페이지의 그림 13 에서 추출한 결과는 myprog.out에 있는 출력의 해당 추출을 표시합니다.

runmqsc 명령에서 stdin 및 stdout을 경로 재지정하려면 기본이 아닌 큐 관리자 (saturn.queue.manager)의 경우 다음 명령 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 12 는 MQSC 명령어 (DEFINE QLOCAL) 를 나타내는 MQSC 명령어 파일의 추출물이다. MQSC 참조에는 각 MQSC 명령과 해당 구문에 대한 설명이 있습니다.

```
 .
 .
 .
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

WebSphere MQ 환경 간의 이식성을 위해 MQSC 명령 파일의 행 길이를 72자로 제한하십시오. 더하기 부호는 명령이 다음 행에서 계속된다는 것을 표시합니다.

MQSC 명령 보고서

runmqsc 명령은 stdout으로 보내지는 보고서를 리턴합니다. 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 출력에서 각 명령에는 74 페이지의 그림 13에서와 같이 일련 번호가 접두부로 표시됩니다. 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: WebSphere 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: WebSphere MQ queue created.
       :
 .
 .
```

그림 *13. MQSC* 명령 보고서 파일에서 추출

제공된 MQSC 명령 파일 실행

다음 MQSC 명령 파일은 WebSphere MQ와 함께 제공됩니다.

amqscos0.tst
샘플 프로그램에서 사용한 오브젝트의 정의.

amqscic0.tst
CICS® 트랜잭션에 대한 큐의 정의.

Windows용 WebSphere MQ 에서 이러한 파일은 *MQ_INSTALLATION_PATH*\tools\mqsc\samples디렉토리에 있습니다. *MQ_INSTALLATION_PATH* WebSphere MQ 가 설치된 상위 레벨 디렉토리를 나타냅니다.

UNIX and Linux 시스템의 경우에는 이들 파일이 *MQ_INSTALLATION_PATH*/samp 디렉토리에 있습니다.*MQ_INSTALLATION_PATH* WebSphere MQ 가 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 WebSphere MQ가 리모트 큐 관리자의 응답을 대기하는 시간(초)입니다.

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.

MQSC 명령을 실행할 수 없는 경우 다음 정보를 사용하여 이러한 공통 문제점 중 적용되는 것이 있는지 확인하십시오. 생성된 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: WebSphere 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: WebSphere 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.
- runmqsc에서 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(750)                           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

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.*)
```

시스템 SYSTEM.*와 일치하는 큐 목록을 표시합니다. 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 참조

큐 관리자의 속성

로컬 큐에 대한 작업

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

이 명령에 대한 자세한 정보는 MQSC 참조의 내용을 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. 79 페이지의 『기본 오브젝트 속성 표시』 도 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. 80 페이지의 『로컬 큐 속성 변경』 의 내용을 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 대하여 큐 관리자에게 알려주어야 합니다.

큐 관리자에게 데드-레터 큐에 대해 알려주려면 `crtmqm` 명령에 데드-레터 큐 이름을 지정하거나(예: `crtmqm -u DEAD.LETTER.QUEUE`) ALTER QMGR 명령에 DEADQ 속성을 사용하여 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

WebSphere MQ는 데드-레터 큐 핸들러를 제공하는데, 이 핸들러를 사용하여 데드-레터 큐에서 발견된 메시지의 처리 또는 제거 방법을 지정할 수 있습니다. 자세한 정보는 WebSphere MQ 데드-레터 큐 핸들러를 사용하여 전달되지 않은 메시지 처리 를 참조하십시오.

기본 오브젝트 속성 표시

DISPLAY QUEUE 명령을 사용하여 WebSphere MQ 오브젝트 정의 시 기본 오브젝트에서 가져온 속성을 표시할 수 있습니다.

WebSphere MQ 오브젝트를 정의할 때 오브젝트는 기본 오브젝트에서 지정하지 않은 속성을 가져옵니다. 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 작성할 때 입력된 것과 **동일하게** 복사되도록 큐 이름을 입력하십시오. 이름에 소문자가 포함된 경우, 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

WebSphere MQ는 큐에 있는 메시지의 컨텐츠를 보는 데 사용할 수 있는 샘플 큐 브라우저를 제공합니다. 브라우저가 소스 및 실행 가능 형식 모두로 제공됩니다.

*MQ_INSTALLATION_PATH*는 WebSphere MQ가 설치되어 있는 상위 레벨 디렉토리를 표시합니다.

In WebSphere MQ for 윈도우, the file names and paths for the sample queue browser are as follows:

소스
MQ_INSTALLATION_PATH\tools\c\samples\

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

UNIX and Linux용 WebSphere MQ 에서 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 82 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

큰 큐를 사용으로 설정

IBM WebSphere MQ에서는 2GB보다 큰 큐를 지원합니다.

Windows 시스템에서 추가 실행 가능성 없이 큰 파일에 대한 지원을 사용할 수 있습니다. AIX, HP-UX, Linux 및 Solaris 시스템에서 2GB보다 큰 큐 파일을 작성할 수 있으려면 큰 파일 지원을 명시적으로 사용 설정할 필요가 있습니다. 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 스토리지 양을 계획하는 데 대한 정보는 IBM WebSphere MQ 웹 사이트를 방문하여 플랫폼별 성능 보고서를 참조하십시오.

```
https://www.ibm.com/software/integration/ts/mqseries/
```

알리어스 큐에 대한 작업

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

V 7.5.0.8

주의: 분배 목록에서는 토픽 오브젝트를 가리키는 알리어스 큐 사용을 지원하지 않습니다. Version 7.5.0, Fix Pack 8부터는 알리어스 큐가 분배 목록의 토픽 오브젝트를 가리키는 경우 IBM WebSphere 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(105 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.

TARGET 속성의 값을 변경하여 MQI 호출을 다른 큐 관리자의 다른 큐로 경로 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 QALIAS (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) TARGET(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 기타 알리어스 큐 명령에 대한 자세한 정보는 MQSC 참조 의 내용을 참조하십시오.

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 있습니다. (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.

이 명령에 대한 자세한 정보는 MQSC 참조의 내용을 참조하십시오.

관련 개념

관리 토픽 오브젝트

86 페이지의 『관리 토픽 정의』

MQSC 명령 **DEFINE TOPIC**을 사용하여 관리 토픽을 작성하십시오. 관리 토픽을 정의할 때,각 토픽 속성을 선택적으로 설정할 수 있습니다.

86 페이지의 『관리 토픽 오브젝트 속성 표시』

MQSC 명령 **DISPLAY TOPIC**을 사용하여 관리 토픽 오브젝트를 표시하십시오.

87 페이지의 『관리 토픽 속성 변경』

두 가지 방법(**ALTER TOPIC** 명령 또는 **REPLACE** 속성을 가지는 **DEFINE TOPIC** 명령)으로 토픽 속성을 변경할 수 있습니다.

87 페이지의 『관리 토픽 정의 복사』

DEFINE 명령에서 LIKE 속성을 사용하여 토픽 정의를 복사할 수 있습니다.

88 페이지의 『관리 토픽 정의 삭제』
MQSC 명령 **DELETE TOPIC**을 사용하여 관리 토픽을 삭제할 수 있습니다.

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. 토픽 트리에 대한 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 86 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, 87 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.예:

```
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

두 가지 방법(**ALTER TOPIC** 명령 또는 **REPLACE** 속성을 가지는 **DEFINE TOPIC** 명령)으로 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. 토픽 트리에 대한 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.

프록시
큐 관리자 사이에 발행물을 라우팅하기 위해 내부적으로 작성된 구독.

API
예를 들어, MQI MQSUB 호출을 사용하여 프로그래밍 방식으로 작성됩니다.

이 명령에 대한 자세한 정보는 MQSC 참조의 내용을 참조하십시오.

관련 개념

88 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.

89 페이지의 『구독의 속성 표시』
DISPLAY SUB 명령을 사용하여 큐 관리자에 알려진 구독의 구성된 속성을 표시할 수 있습니다.

90 페이지의 『로컬 구독 속성 변경』
ALTER SUB 명령 또는 **DEFINE SUB** 명령을 **REPLACE** 속성과 함께 사용하여 두 가지 방법으로 등록 속성을 변경할 수 있습니다.

90 페이지의 『로컬 구독 정의 복사』
DEFINE 명령에서 **LIKE** 속성을 사용하여 구독 정의를 복사할 수 있습니다.

90 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.
- 설명 및 토픽 문자열에 대한 값을 제외하고 표시되는 모든 속성 값이 기본값입니다.이는 설명으로만 여기에 표시됩니다. 기본값이 원하는 것이거나 변경되지 않는 것이라고 확인하는 경우 생략할 수 있습니다. 89 페이지의 『구독의 속성 표시』 도 참조하십시오.
- 이름이 TEST인 동일한 큐 관리자에 로컬 구독이 이미 있는 경우 이 명령이 실패합니다.큐의 기존 정의를 덮어쓰려는 경우 **REPLACE** 속성을 사용하지만, 90 페이지의 『로컬 구독 속성 변경』 의 내용도 참조하십시오.
- 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: WebSphere 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: WebSphere 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: WebSphere 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 89 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 WebSphere MQ 서비스는 큐 관리자를 시작한 사용자의 사용자 ID에서 항상 시작됩니다.

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 `runmqtrm`과 같이 트리거 모니터 프로세스를 실행하는 데 사용할 수 있습니다.

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
92 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

93 페이지의 『추가 환경 변수』

서비스가 시작될 때, 서비스 프로세스가 시작되는 환경은 큐 관리자의 환경으로부터 상속됩니다. 정의하려는 변수를 service.env 환경 대체 파일 중 하나에 추가하여 서비스 프로세스의 환경에서 설정할 추가 환경 변수를 정의할 수 있습니다.

추가 환경 변수

서비스가 시작될 때, 서비스 프로세스가 시작되는 환경은 큐 관리자의 환경으로부터 상속됩니다. 정의하려는 변수를 service.env 환경 대체 파일 중 하나에 추가하여 서비스 프로세스의 환경에서 설정할 추가 환경 변수를 정의할 수 있습니다.

참고:

환경 변수를 추가할 수 있는 두 개의 가능한 파일이 있습니다.

- 시스템 범위 service.env 파일. 이 파일은 UNIX and Linux 시스템의 /var/mqm에 있거나 Windows 시스템에 설치하는 동안 선택된 데이터 디렉토리에 있습니다.
- 큐 관리자 데이터 디렉토리에 있는 큐 관리자 범위 service.env 파일. 예를 들어, QMNAME이라는 큐 관리자에 대한 환경 대체 파일의 위치는 다음과 같습니다.
  - UNIX and Linux 시스템의 경우, /var/mqm/qmgrs/QMNAME/service.env
  - Windows 시스템의 경우, C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env

가능한 경우, 시스템 범위 파일에서 해당 정의보다 우선순위에 서는 큐 관리자 범위 파일의 정의로 양쪽 모든 파일이 처리됩니다.

service.env에서 환경 변수를 지정할 수 있습니다. 예를 들어, IBM WebSphere MQ 서비스가 여러 명령을 실행하는 경우 service.env 파일에서 PATH 사용자 변수를 설정하는 것이 유용할 수 있습니다. 변수를 설정하는 값은 환경 변수일 수 없습니다. 예를 들어 CLASSPATH=*%CLASSPATH%*는 올바르지 않습니다. 마찬가지로, Linux PATH=*$PATH*:/opt/mqm/bin인 경우 예상치 못한 결과가 나타날 수 있습니다.

CLASSPATH는 대문자로 표시되어야 하고 클래스 경로 명령문은 리터럴만을 포함할 수 있습니다. 일부 서비스(예: 텔레메트리)는 자체 클래스 경로를 설정합니다. service.env에 정의된 CLASSPATH가 해당 파일에 추가됩니다.

service.env 파일에 정의된 변수의 형식은 이름 및 값 변수 쌍의 목록입니다. 각 변수는 새 행에 정의되어야 하고 공백 문자를 포함하여 명백하게 정의된 것처럼 각 변수가 사용됩니다. service.env 파일의 예제는 다음과 같습니다.

```
  #*******************************************************************#
  #*                                                                 *#
  #* <N_OCO_COPYRIGHT>                                               *#
  #* Licensed Materials - Property of IBM                            *#
  #*                                                                 *#
  #* 63H9336                                                         *#
  #* (C) Copyright IBM Corporation 2005, 2024.                       *#
  #*                                                                 *#
  #* <NOC_COPYRIGHT>                                                 *#
  #*                                                                 *#
  #*******************************************************************#
  #*******************************************************************#
  #* Module Name: service.env                                        *#
  #* Type       : WebSphere 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

94 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
WebSphere MQ가 설치된 위치입니다.

MQ_DATA_PATH
WebSphere MQ 데이터 디렉토리의 위치는 다음과 같습니다.

- UNIX and Linux 시스템에서 WebSphere MQ 데이터 디렉토리 위치는 /var/mqm/입니다.
- Windows 시스템에서 WebSphere MQ 데이터 디렉토리의 위치는 WebSphere MQ를 설치하는 동안 선택한 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 WebSphere MQ 파일 이름 이해를 참조하십시오.

대체가능한 삽입을 사용하려면, 임의의 STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT 또는 STDERR 문자열로 + 문자 내에 토큰을 삽입하십시오. 이것의 예는 94 페이지의 『서비스 오브젝트 사용 예』의 내용을 참조하십시오.

서비스 오브젝트 사용 예

이 절의 서비스는 별도로 언급된 경우를 제외하고는 UNIX 스타일의 경로 구분 기호 문자로 작성되었습니다.

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.

amqsstop는 큐 관리자에게 프로세스 ID에 대한 모든 연결을 끊도록 요청하는 WebSphere 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 94 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. 서버 서비스 오브젝트가 대체되어 이니시에이션 큐가 JUPITER.INITIATION.QUEUE로 지정됩니다. 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 95 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 UNIX and 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:

logger는 운영 체제의 시스템 로그에 입력 항목을 기록할 수 있는 WebSphere MQ와 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 쓰여졌습니다.

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 공백이 포함된 문자열이며 단일 문자열로 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

WebSphere 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.

트리거에 대해서는 트리거를 사용하여 WebSphere MQ 애플리케이션 시작에서 자세히 설명합니다.

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 일반적인 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.

원격 IBM WebSphere MQ 오브젝트 관리

이 섹션은 사용자에게 MQSC 명령을 사용하여 리모트 큐 관리자에 IBM WebSphere MQ 오브젝트를 관리하는 방법 및 메시지와 응답 메시지의 목적지를 제어하기 위해 멀리 있는 큐 오브젝트를 사용하는 방법을 말합니다.

이 절에서는 다음을 설명합니다.

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
- WebSphere MQ가 지원하는 다른 플랫폼에서 실행 중입니다.

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 WebSphere MQ Explorer
- 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.

101 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 사용하는 WebSphere 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.

클러스터, 해당 속성 및 설정 방법에 대한 자세한 정보는 큐 관리자 클러스터를 참조하십시오.

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 사용한 관리 프로그램 작성에 대한 자세한 정보는 9 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. UNIX and Linux 시스템, Windows 시스템, IBM i 및 z/OS를 포함한 다른 WebSphere MQ 환경에 있는 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.

101 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: -w 옵션과 함께 **runmqsc**를 사용하는 경우, `source.queue.manager`는 기본 큐 관리자**이어야 합니다**. 큐 관리자 작성에 대한 추가 정보는 crtmqm을 참조하십시오.

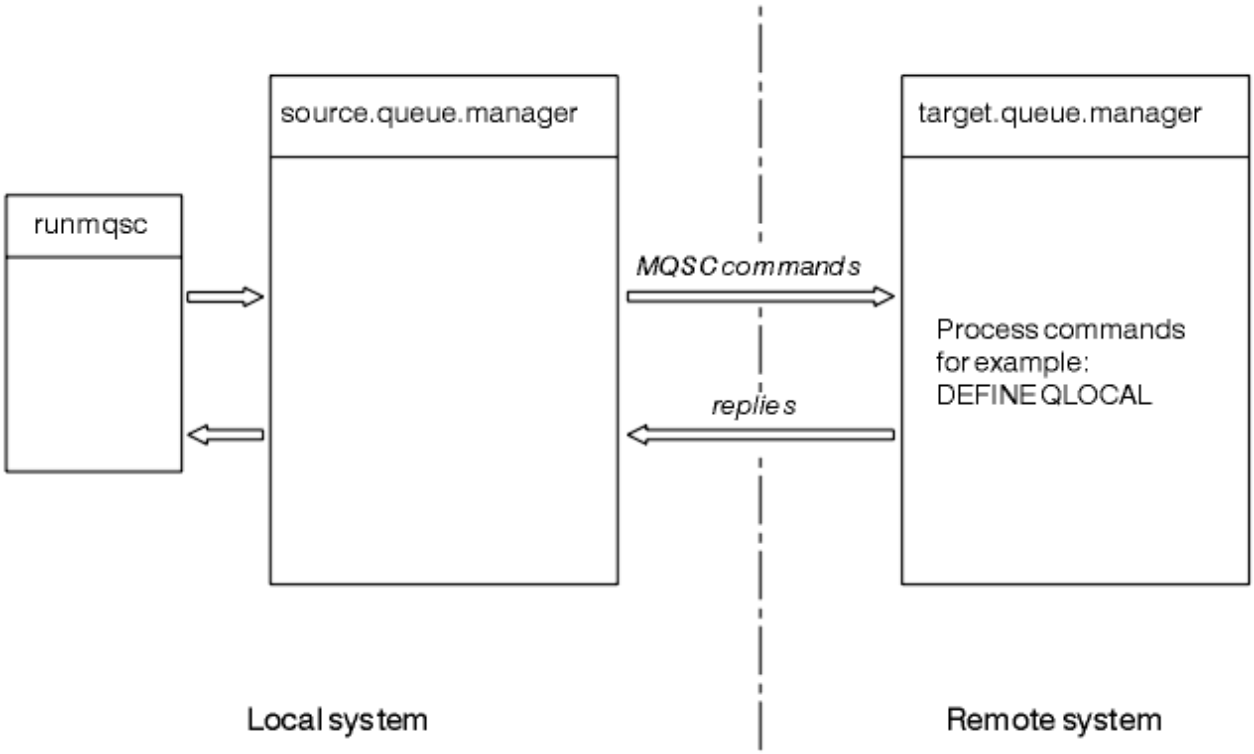


그림 *17. MQSC* 명령을 사용하는 원격 관리

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.102 페이지의 그림 18 은 이 구성을 요약한다.

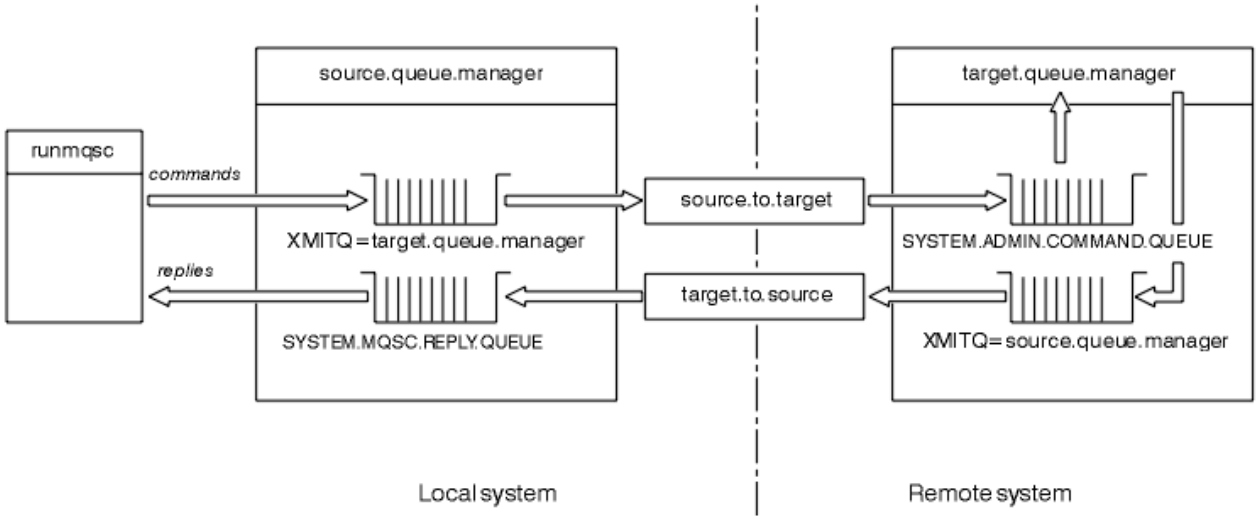


그림 *18.* 원격 관리를 위한 채널 및 큐 설정

채널 설정에 대한 자세한 정보는 분산 큐잉을 사용하여 애플리케이션 연결을 참조하십시오.

채널, 리스너 및 전송 큐 정의

소스 큐 관리자(source.queue.manager)에서 채널, 리스너 및 전송 큐를 정의하려면 다음 MQSC 명령을 실행하십시오.

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 정의하십시오.

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.

WebSphere MQ는 인바운드 접속 요청을 수신하고 적절한 수신자 또는 서버 연결 채널을 찾을 수 없는 경우 자동으로 채널을 작성합니다. 자동 정의는 WebSphere MQ: SYSTEM.AUTO.RECEIVER 및 SYSTEM.AUTO.SVRCONN을 참조하십시오.

채널 정의를 자동으로 작성하는 것에 대한 자세한 정보는 채널 준비를 참조하십시오. 클러스터의 채널 자동 정의에 대한 정보는 클러스터 채널의 자동 정의를 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 WebSphere MQ for 윈도우 or WebSphere MQ for Linux (x86 and x86-64 platforms) can perform the operations described in the following sections using the IBM WebSphere MQ Explorer. 자세한 정보는 53 페이지의 『IBM WebSphere 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 SCMDSERV매개변수를 지정하는 MQSC 명령 `ALTER QMGR` 를 사용하여 변경할 수 있습니다. 기본적으로, 명령 서버가 자동으로 시작됩니다.

*SCMDSERV*가 MANUAL로 설정되는 경우, 명령을 사용하여 명령 서버를 시작하십시오.

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

여기서 `saturn.queue.manager`는 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 처리하려면 명령 서버가 대상 큐 관리자에서 실행하고 **있어야** 합니다 (이는 소스 큐 관리자에서 필수는 아닙니다.) 큐 관리자에서 명령 서버를 시작하는 방법에 대한 정보는 103 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** 명령에 -m *LocalQmgrName* 옵션을 지정할 경우 로컬 큐 관리자를 통해 실행되는 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.

75 페이지의 『MQSC 명령으로 문제점 해결』의 내용을 참조하십시오.

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. 105 페이지의 『리모트 큐의 로컬 정의 작성』 도 참조하십시오.

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 별명 및 클러스터 를 참조하십시오.

코드화된 문자 세트 간의 데이터 변환

WebSphere MQ 정의 형식(내장 형식이라고도 함) 의 메시지 데이터는 큐 관리자에 의해 하나의 코드화된 문자 세트에서 다른 문자 세트로 변환될 수 있습니다(두 문자 세트가 모두 단일 언어 또는 유사 언어 그룹으로 관련된 경우).

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 변환의 경우 모든 큐 관리자를 참조하십시오.

지원되는 변환은 데이터 변환에 정의됩니다.

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 109 페이지의 『기본 데이터 변환』 에 설명되어 있습니다.

파일 ccsid.tbl

파일 ccsid.tbl은 다음 목적으로 사용됩니다.

- In WebSphere MQ for 윈도우 it records all the supported code sets.
- AIX 및 HP-UX 플랫폼에서 지원되는 코드 세트가 운영 체제에서 내부적으로 보유됩니다.

- 다른 모든 UNIX and Linux 플랫폼에서 지원되는 코드 세트는 WebSphere MQ가 제공하는 변환 테이블에 보유됩니다.
- 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.

In WebSphere MQ for 윈도우, ccsid.tbl is located in directory `C:\Program Files\IBM\WebSphere MQ\conv\table` by default.

UNIX and Linux 시스템의 WebSphere MQ 에서 ccsid.tbl 은 `/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.
- 한 CCSID는 ASCII 코드화된 문자 세트를 표시하고 다른 CCSID는 EBCDIC 코드화된 문자 세트를 표시할 경우, WebSphere MQ는 ccsid.tbl에 정의된 기본 데이터 변환 CCSID를 사용하여 데이터를 변환합니다.

참고: 메시지에 지정된 코드화 문자 세트 및 기본 코드화 문자 세트에 동일한 코드 값이 있는 문자로 변환되는 문자를 제한하십시오. WebSphere MQ 오브젝트 이름(IBM WebSphere MQ 오브젝트 이름 지정에서 설명함)으로 유효한 문자 세트만 사용하는 경우 일반적으로 이 요구사항을 충족합니다. 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 데이터를 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 WebSphere MQ Telemetry 관리

IBM WebSphere MQ Telemetry는 IBM WebSphere MQ Explorer를 사용하거나 또는 명령행에서 관리합니다. 탐색기를 사용하여 텔레메트리 채널을 구성하고, 텔레메트리 서비스를 제어하며, IBM WebSphere MQ에 연결된 MQTT 클라이언트를 모니터하십시오. JAAS, SSL및 IBM WebSphere MQ 오브젝트 권한 관리자를 사용하여 IBM WebSphere MQ Telemetry 의 보안을 구성하십시오.

IBM WebSphere MQ Explorer를 사용한 관리

탐색기를 사용하여 텔레메트리 채널을 구성하고, 텔레메트리 서비스를 제어하며, IBM WebSphere MQ에 연결된 MQTT 클라이언트를 모니터하십시오. JAAS, SSL및 IBM WebSphere MQ 오브젝트 권한 관리자를 사용하여 IBM WebSphere MQ Telemetry 의 보안을 구성하십시오.

명령행을 사용하여 관리

IBM WebSphere MQ Telemetry는 IBM WebSphere MQ MQSC 명령을 사용하여 명령행에서 완전하게 관리될 수 있습니다.

또한 IBM WebSphere MQ Telemetry 문서에는 MQ Telemetry Transport v3 클라이언트 애플리케이션의 기본 사용법을 보여주는 샘플 스크립트도 있습니다.

사용하기 전에 IBM WebSphere MQ Telemetry 에 대한 애플리케이션 개발 섹션의 IBM WebSphere MQ Telemetry 샘플 프로그램 에서 샘플을 읽고 이해하십시오.

관련 개념

WebSphere MQ Telemetry

114 페이지의 『MQTT 클라이언트에 메시지를 보내도록 분산 큐잉 구성』
IBM WebSphere MQ 애플리케이션은 클라이언트에서 작성된 구독을 발행하거나 메시지를 직접 송신하여 MQTT v3 클라이언트 메시지를 보낼 수 있습니다. 어떤 방법을 사용하든, 메시지는 SYSTEM.MQTT.TRANSMIT.QUEUE에 배치되며 텔레메트리 (MQXR) 서비스로 클라이언트에 전송됩니다. SYSTEM.MQTT.TRANSMIT.QUEUE에 메시지를 배치하는 방법은 여러 가지가 있습니다.

116 페이지의 『MQTT 클라이언트 ID, 권한 및 인증』

123 페이지의 『SSL을 사용하여 텔레메트리 채널 인증』

125 페이지의 『텔레메트리 채널의 발행물 개인정보 보호』

125 페이지의 『MQTT 클라이언트 및 텔레메트리 채널의 SSL 구성』

130 페이지의 『텔레메트리 채널 JAAS 구성』
클라이언트에서 보내는 Username을 인증하기 위해 JAAS를 구성하십시오.

131 페이지의 『디바이스용 IBM WebSphere MQ Telemetry 디먼 개념』
디바이스용 IBM WebSphere MQ Telemetry 디먼은 고급 MQTT V3 클라이언트 애플리케이션입니다. 다른 MQTT 클라이언트에서 온 메시지를 저장 후 전달하려면 이를 사용하십시오. MQTT 클라이언트처럼 IBM WebSphere MQ에 연결하지만 다른 MQTT 클라이언트도 여기 연결할 수 있습니다.

관련 태스크

110 페이지의 『Linux 및 AIX에서 텔레메트리에 대한 큐 관리자 구성』
IBM WebSphere MQ Telemetry를 실행하도록 큐 관리자를 구성하려면 다음 수동 단계를 수행하십시오. IBM WebSphere MQ Explorer에 대한 IBM WebSphere MQ Telemetry 지원을 사용하여 보다 간단한 구성을 설정할 수 있는 자동화된 프로시저를 실행할 수 있습니다.

112 페이지의 『Windows에서 텔레메트리용 큐 관리자 작성』
IBM WebSphere MQ Telemetry를 실행하도록 큐 관리자를 구성하려면 다음 수동 단계를 수행하십시오. IBM WebSphere MQ Explorer에 대한 IBM WebSphere MQ Telemetry 지원을 사용하여 보다 간단한 구성을 설정할 수 있는 자동화된 프로시저를 실행할 수 있습니다.

관련 참조

MQXR 특성

Linux 및 AIX에서 텔레메트리에 대한 큐 관리자 구성

IBM WebSphere MQ Telemetry를 실행하도록 큐 관리자를 구성하려면 다음 수동 단계를 수행하십시오. IBM WebSphere MQ Explorer에 대한 IBM WebSphere MQ Telemetry 지원을 사용하여 보다 간단한 구성을 설정할 수 있는 자동화된 프로시저를 실행할 수 있습니다.

시작하기 전에

1. IBM WebSphere MQ설치 방법 및 IBM WebSphere MQ Telemetry 기능에 대한 정보는 IBM WebSphere 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

IBM WebSphere MQ Explorer에 대한 IBM WebSphere MQ Telemetry 지원에는 마법사 및 샘플 명령 프로시저 `sampleMQM`이 포함됩니다. 게스트 사용자 ID를 사용하여 초기 구성을 설정합니다. IBM WebSphere MQ Explorer 을 사용하여 IBM WebSphere MQ Telemetry 설치 확인 및 IBM WebSphere MQ Telemetry 샘플 프로그램을 참조하십시오.

다른 권한 부여 설계를 사용하여 IBM WebSphere MQ Telemetry를 수동으로 구성하려면 이 태스크의 단계를 수행하십시오.

프로시저

1. 텔레메트리 샘플 디렉토리에서 명령 창을 여십시오.

   텔레메트리 샘플 디렉토리는 `/opt/mqm/mqxr/samples`입니다.
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.QUEUE`가 작성됩니다.

   이 태스크에서 `SYSTEM.MQTT.TRANSMIT.QUEUE`는 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.QUEUE`를 기본 전송 큐로 만들지 않습니다.

   `SYSTEM.MQTT.TRANSMIT.QUEUE`를 기본 전송 큐로 만들려면 기본 전송 큐 특성을 대체하십시오. IBM WebSphere 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.QUEUE`로 대체하는 이유는 MQTT 클라이언트로 직접 메시지를 보내는 작업을 더욱 쉽게 만들기 위해서입니다. 기본 전송 큐를 대체하지 않고 IBM WebSphere MQ 메시지를 수신하는 모든 클라이언트에 대한 리모트 큐 정의를 추가해야 합니다. 115 페이지의 『클라이언트에 직접 메시지 송신』의 내용을 참조하십시오.
4. 하나 이상의 사용자 ID를 작성하려면 117 페이지의 『WebSphere MQ 오브젝트에 액세스할 수 있도록 MQTT 클라이언트에 권한 부여』에서 프로시저를 수행하십시오. 사용자 ID에는 발행할 권한, 구독할 권한 및 MQTT 클라이언트로 발행을 보낼 권한이 있습니다.
5. 텔레메트리(MQXR) 서비스를 설치하십시오.

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   112 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. IBM WebSphere MQ Explorer를 사용하여 MQTT 클라이언트와의 연결을 승인하도록 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 WebSphere MQ Telemetry 샘플 프로그램도 참조하십시오.

예

112 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*

Windows에서 텔레메트리용 큐 관리자 작성

IBM WebSphere MQ Telemetry를 실행하도록 큐 관리자를 구성하려면 다음 수동 단계를 수행하십시오. IBM WebSphere MQ Explorer에 대한 IBM WebSphere MQ Telemetry 지원을 사용하여 보다 간단한 구성을 설정할 수 있는 자동화된 프로시저를 실행할 수 있습니다.

시작하기 전에

1. IBM WebSphere MQ설치 방법 및 IBM WebSphere MQ Telemetry 기능에 대한 정보는 IBM WebSphere 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 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

IBM WebSphere MQ Explorer에 대한 IBM WebSphere MQ Telemetry 지원에는 마법사 및 샘플 명령 프로시저 sampleMQM이 포함됩니다. 게스트 사용자 ID를 사용하여 초기 구성을 설정합니다. IBM WebSphere MQ Explorer 을 사용하여 IBM WebSphere MQ Telemetry 설치 확인 및 IBM WebSphere MQ Telemetry 샘플 프로그램을 참조하십시오.

다른 권한 부여 설계를 사용하여 IBM WebSphere 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. 기본 전송 큐 설정하기

```
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. IBM WebSphere MQ Explorer를 사용하거나 113 페이지의 그림 20의 명령을 통해 특성을 대체하십시오.

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 WebSphere MQ 메시지를 수신하는 모든 클라이언트에 대한 리모트 큐 정의를 추가해야 합니다. 115 페이지의 『클라이언트에 직접 메시지 송신』의 내용을 참조하십시오.

4. 하나 이상의 사용자 ID를 작성하려면 117 페이지의 『WebSphere MQ 오브젝트에 액세스할 수 있도록 MQTT 클라이언트에 권한 부여』에서 프로시저를 수행하십시오. 사용자 ID에는 발행할 권한, 구독할 권한 및 MQTT 클라이언트로 발행을 보낼 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. IBM WebSphere MQ Explorer를 사용하여 MQTT 클라이언트와의 연결을 승인하도록 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 WebSphere MQ Telemetry 샘플 프로그램도 참조하십시오.

수동으로 SYSTEM.MQXR.SERVICE 작성

114 페이지의 그림 21 는 **runmqsc** 명령을 표시하여 Windows에서 SYSTEM.MQXR.SERVICE 를 수동으로 작성합니다.

```
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 보내도록 분산 큐잉 구성

IBM WebSphere MQ 애플리케이션은 클라이언트에서 작성된 구독을 발행하거나 메시지를 직접 송신하여 MQTT v3 클라이언트 메시지를 보낼 수 있습니다. 어떤 방법을 사용하든, 메시지는 SYSTEM.MQTT.TRANSMIT.QUEUE에 배치되며 텔레메트리 (MQXR) 서비스로 클라이언트에 전송됩니다. SYSTEM.MQTT.TRANSMIT.QUEUE에 메시지를 배치하는 방법은 여러 가지가 있습니다.

MQTT 클라이언트 구독에 대한 응답으로 메시지 발행

텔레메트리(MQXR) 서비스는 MQTT 클라이언트를 대신하여 구독을 작성합니다. 클라이언트는 클라이언트가 보낸 구독과 일치하는 발행물에 대한 목적지입니다. 텔레메트리 서비스는 일치하는 발행을 클라이언트로 다시 전달합니다.

MQTT 클라이언트는 WebSphere MQ에 큐 관리자로 연결되며, 이러한 큐 관리자의 이름은 ClientIdentifier로 설정됩니다. 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 ClientIdentifier를 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 ClientIdentifier가 리모트 큐 관리자 이름으로 작성되므로 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 해석해야 합니다. 표준 WebSphere MQ 오브젝트 이름 확인을 사용하면 *ClientIdentifier*가 다음과 같이 해석됩니다. 115 페이지의 표 6의 내용을 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.QUEUE가 큐 이름으로 설정됩니다. MQTT v3 클라이언트는 큐를 지원하지 않으므로 해석된 큐 이름은 클라이언트에서 무시됩니다.
   로컬 큐 관리자 특성, 기본 전송 큐, 이름은 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정되어야 합니다. 따라서 발행이 SYSTEM.MQTT.TRANSMIT.QUEUE 에 전송되어 클라이언트로 전송됩니다.

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.QUEUE로 설정할 필요가 없습니다.

표 *6. MQTT* 큐 관리자 알리어스의 이름 확인					
	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.

모든 WebSphere MQ 프로그램은 동일한 토픽을 발행할 수 있습니다. 발행은 토픽에 대한 구독이 있는 MQTT v3 클라이언트를 포함해 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.QUEUE 클러스터 속성을 제공하지 마십시오.

MQTT 클라이언트 구독자 및 발행자는 다른 큐 관리자에 연결할 수 있습니다. 구독자 및 발행자는 일부 동일한 클러스터에 포함되거나 발행/구독 계층을 통해 연결될 수 있습니다. WebSphere MQ를 사용하여 발행이 발행자에서 구독자로 전달됩니다.

클라이언트에 직접 메시지 송신

클라이언트가 구독을 작성하고 구독 토픽과 일치하는 발행을 수신하는 다른 방법으로 MQTT v3 클라이언트에 직접 메시지를 보내는 방법이 있습니다. MQTT v3 클라이언트 애플리케이션은 직접 메시지를 보낼 수 없지만, WebSphere 애플리케이션 같은 다른 애플리케이션은 메시지를 보낼 수 있습니다.

WebSphere MQ 애플리케이션은 MQTT v3 클라이언트의 ClientIdentifier를 알고 있어야 합니다. MQTT v3 클라이언트에는 큐가 없으므로 대상 큐 이름은 토픽 이름으로 MQTT v3 애플리케이션 클라이언트 messageArrived 메소드로 전달됩니다. 예를 들어, MQI 프로그램에서 클라이언트가 포함된 오브젝트 디스크립터를 ObjectQmgrName으로 작성하십시오.

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

그림 *23. MQTT v3* 클라이언트 목적지로 메시지를 보내는 *MQI* 오브젝트 디스크립터

예를 들어, JMS를 사용하여 애플리케이션을 작성할 경우 지점간 목적지를 작성하십시오.

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

그림 *24. MQTT v3* 클라이언트로 메시지를 보내는 *JMS* 목적지

MQTT 클라이언트로 요청하지 않은 메시지를 보내려면 리모트 큐 정의를 사용하십시오. 리모트 큐 관리자 이름은 클라이언트의 `ClientIdentifier`로 해석되어야 합니다. 전송 큐는 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 해석되어야 합니다. 116 페이지의 표 7를 참조하십시오. 리모트 큐 이름은 어떤 것이든 가능합니다. 클라이언트는 이 이름을 토픽 문자열로 수신합니다.

표 *7. MQTT* 클라이언트 리모트 큐 정의의 이름 확인

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 연결되어 있으면 메시지는 직접 MQTT 클라이언트로 전송됩니다. 이는 `messageArrived` 메소드를 호출합니다. `messageArrived` 메소드를 참조하십시오.

클라이언트가 지속적 세션과 연결이 끊어진 경우 메시지는 `SYSTEM.MQTT.TRANSMIT.QUEUE`에 저장됩니다. MQTT Stateless및 Stateful 세션 을 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 더 낮출 수 있습니다. 클라이언트로 직접 보내지는 메시지를 위해 QoS(Quality of Service)를 낮추려면 토픽 `DEFAULT.QoS`에 대한 구독을 작성하십시오. 클라이언트가 지원할 수 있는 최대 서비스 품질(QoS)을 지정하십시오.

MQTT 클라이언트 ID, 권한 및 인증

텔레메트리(MQXR) 서비스는 MQTT 채널을 사용하여 MQTT 클라이언트 대신 WebSphere MQ 토픽을 발행하거나 구독합니다. WebSphere MQ 관리자는 WebSphere MQ 권한에 사용되는 MQTT 채널 ID를 구성합니다. 관리자는 채널에 대한 공통 ID를 정의하거나 채널에 연결된 클라이언트의 `Username` 또는 `ClientIdentifier`를 사용할 수 있습니다.

텔레메트리(MQXR) 서비스는 클라이언트 인증서를 사용하거나 클라이언트가 제공하는 `Username`을 사용하여 클라이언트를 인증할 수 있습니다. `Username`은 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 프로그램에서는 MQTT 채널을 통해 서버로 전송되는 `Username`과 `Password`를 설정합니다. 이는 연결을 암호화하고 인증하기 위해 필요한 SSL 특성을 설정할 수도 있습니다. 관리자는 MQTT 채널을 인증할지 선택하고 채널을 인증하는 방법을 결정합니다.

MQTT 클라이언트에 권한을 부여하여 WebSphere MQ 오브젝트에 액세스하려면 클라이언트의 `ClientIdentifier`나 `Username`에 권한을 부여하거나 공통 클라이언트 ID에 권한을 부여해야 합니다. 클라이언트가 WebSphere MQ에 연결할 수 있도록 하려면 `Username`을 인증하거나 클라이언트 인증서를 사용합니다. JAAS를 구성하여 `Username`을 인증하고 SSL을 구성하여 클라이언트 인증서를 인증하십시오.

클라이언트에서 `Password`를 설정할 경우 VPN을 통해 연결을 암호화하거나 SSL을 사용하도록 MQTT 채널을 구성하여 비밀번호를 개인용으로 유지합니다.

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

`ClientIdentifier`, `Username` 또는 권한 부여에 필요한 공통 클라이언트 ID를 사용하여 WebSphere MQ 오브젝트에 액세스합니다.

WebSphere MQ 관리자가 MQTT 채널의 ID를 선택할 수 있는 방법은 세 가지가 있습니다. 관리자는 클라이언트가 사용하는 MQTT 채널을 정의하거나 수정할 때 선택할 수 있습니다. ID는 WebSphere MQ 토픽에 대한 액세스 권한을 부여하는 데 사용됩니다. 선택 항목은 다음과 같습니다.

1. 클라이언트 ID
2. 관리자가 채널에 제공하는 ID.
3. MQTT 클라이언트에서 전달된 사용자 이름.

사용자 이름은 MqttConnectOptions 클래스의 속성입니다. 클라이언트가 서비스에 연결되기 전에 설정되어야 합니다. 기본값은 널입니다.

WebSphere MQ **setmqaut** 명령을 사용하여 MQTT 채널과 연관된 ID로 사용하도록 권한을 부여할 개체와 조치를 선택합니다. 예를 들어, 큐 관리자인 `QM1`의 관리자가 제공한 채널 ID `MQTTClient`에 권한을 부여하려면 다음을 수행하십시오.

```
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
```

WebSphere MQ 오브젝트에 액세스할 수 있도록 *MQTT* 클라이언트에 권한 부여

WebSphere MQ 오브젝트를 발행 및 구독할 수 있도록 MQTT 클라이언트에 권한을 부여하려면 다음 단계를 수행하십시오. 단계는 네 개의 대체 액세스 제어 패턴을 따릅니다.

시작하기 전에

MQTT 클라이언트가 Telemetry 채널에 연결할 때 ID를 지정하면 해당 MQTT 클라이언트가 WebSphere MQ의 오브젝트에 액세스할 수 있도록 권한이 부여됩니다. WebSphere MQ 관리자는 WebSphere MQ Explorer를 사용하여 클라이언트에게 세 가지 유형의 ID 중 하나를 부여하도록 Telemetry 채널을 구성합니다.

1. `ClientIdentifier`
2. 사용자 이름
3. 관리자가 채널에 지정하는 이름

사용하는 유형에 관계없이 ID는 설치된 권한 서비스를 통해 WebSphere MQ에 프린시펄로 정의되어야 합니다. Linux 에서 기본 권한 부여 서비스는 OAM (Object Authority Manager) 이라고 합니다. OAM을 사용 중인 경우, ID는 사용자 ID로 정의되어야 합니다.

ID를 사용하여 클라이언트 또는 클라이언트 콜렉션에 WebSphere MQ에 정의된 토픽을 발행 또는 구독할 권한을 부여하십시오. MQTT 클라이언트가 토픽을 구독하면 ID를 사용하여 해당 MQTT 클라이언트에 결과 발행을 수신할 권한을 부여하십시오.

많은 수의 MQTT 클라이언트가 있는 시스템을 관리하는 것은 어려운 작업이므로 각각의 MQTT 클라이언트에게는 별도의 액세스 권한이 필요합니다. 한 가지 해결 방법으로 공통 ID를 정의하여 개별 MQTT 클라이언트를 공통

ID 중 하나와 연결하는 방법이 있습니다. 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.

OAM에서 다음 두 가지 방법으로 MQTT 클라이언트를 공통 ID와 결합할 수 있습니다.

1. 관리자가 WebSphere MQ Explorer를 사용하여 할당하는 다양한 사용자 ID로 Telemetry 채널을 여러 개 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.

- 118 페이지의 『액세스 제어 없음』
- 118 페이지의 『거친 액세스 제어』
- 118 페이지의 『중간 단위 액세스 제어』
- 119 페이지의 『세밀한 액세스 제어』

모델을 사용하면 그 결과로 *mqttUsers*에게 WebSphere MQ를 발행 및 구독하고 WebSphere MQ에서 발행을 수신할 수 있는 권한 세트를 지정할 수 있습니다.

액세스 제어 없음
MQTT 클라이언트는 WebSphere MQ 관리 권한이 지정되며, 모든 오브젝트에 대해 모든 조치를 수행할 수 있습니다.

프로시저

1. 모든 MQTT 클라이언트의 ID로 사용할 사용자 ID *mqttUser*를 작성하십시오.
2. *mqttUser* 를 `mqm` 그룹에 추가하십시오. 의 그룹에 사용자또는 Linux의 그룹에 사용자 추가 를 참조하십시오.

거친 액세스 제어
MQTT 클라이언트는 발행 및 구독하고 MQTT 클라이언트에 메시지를 보낼 수 있는 권한이 있습니다. 기타 조치를 수행하거나 기타 오브젝트에 액세스하기 위한 권한이 없습니다.

프로시저

1. 모든 MQTT 클라이언트의 ID로 사용할 사용자 ID *mqttUser*를 작성하십시오.
2. *mqttUser*에 모든 토픽을 발행 및 구독하고 MQTT 클라이언트에게 발행을 보낼 수 있는 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 다양한 토픽 세트를 발행 및 구독하고 MQTT 클라이언트에 메시지를 보낼 수 있도록 서로 다른 그룹으로 나뉩니다.

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. *mqtt*에게 MQTT 클라이언트에 토픽을 보낼 수 있는 권한을 부여하십시오.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

세밀한 액세스 제어
MQTT 클라이언트는 그룹에 오브젝트에 대해 조치를 수행할 수 있는 권한을 부여하는 기존의 액세스 제어 시스템에 통합됩니다.

이 태스크 정보

사용자 ID가 필요한 권한 부여에 따라 하나 이상의 운영 체제 그룹으로 지정됩니다. WebSphere MQ 애플리케이션이 MQTT 클라이언트와 동일한 토픽 공간을 발행 및 구독하는 경우 다음 모델을 사용하십시오. 이 경우 그룹을 Publish*X*, Subscribe*Y* 및 mqtt라고 합니다.

Publish*X*
Publish*X* 그룹의 구성원은 *topicX*를 발행할 수 있습니다.

Subscribe*Y*
Subscribe*Y* 그룹의 구성원은 *topicY*에 구독할 수 있습니다.

mqtt
mqtt 그룹의 구성원은 MQTT 클라이언트에 발행을 보낼 수 있습니다.

프로시저

1. 발행/구독 토픽 트리의 여러 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. 여러 Publish*X* 및 Subscribe*X* 그룹에 다양한 토픽에 대한 권한을 부여하고 *mqtt* 그룹에는 MQTT 클라이언트에 메시지를 보낼 수 있는 권한을 부여하십시오.

```
setmqaut -m qMgr -t topic -n topic1 -p PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

비밀번호를 사용한 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 클라이언트가 제공하는 비밀번호를 사용합니다.

WebSphere MQ 관리자는 클라이언트가 연결된 MQTT 채널을 구성하여 사용자 이름을 인증할지 또는 인증하지 않을지에 대해 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. WebSphere MQ 관리자는 WebSphere MQ Explorer를 사용하여 이름에 MQTTClientUser 같은 MQTT 채널 ID를 설정합니다.
2. WebSphere MQ 관리자는 MQTTClient에 토픽을 발행하고 구독하기 위한 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 이름과 비밀번호를 설정합니다.
4. 보안 개발자는 비밀번호를 가지는 사용자 이름에 권한을 부여하기 위한 JAAS LoginModule을 작성하고 JAAS 구성 파일에 이를 포함시킵니다.
5. WebSphere MQ 관리자는 MQTT 채널을 구성하여 JAAS를 통해 클라이언트의 사용자 이름을 인증합니다.

SSL을 사용한 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 클라이언트에 제공하여 IBM WebSphere MQ에 대해 MQTT 클라이언트를 인증할 수 있습니다. IBM WebSphere MQ 관리자는 MQTT 클라이언트가 SSL을 사용하여 큐 관리자에 대해 인증하도록 강제 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.

-
-

*SSL*을 사용하여 *MQTT* 클라이언트 인증에 필요한 텔레메트리 채널 구성

IBM WebSphere 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 인증서는 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 `SSLCRLNL`입니다.

클라이언트가 보낸 모든 인증서가 서버 키 저장소에 있는 인증서에 의해 확인되면 클라이언트가 인증됩니다.

WebSphere MQ 관리자는 동일한 Telemetry 채널을 구성하여 JAAS를 통해 클라이언트의 `UserName`이나 `ClientIdentifier`는 물론 클라이언트 `Password`를 확인할 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 WebSphere MQ의 인증에만 사용됩니다. 클라이언트의 TCP/IP 주소를 검증하거나, 권한 부여 또는 회계를 위한 클라이언트 ID를 설정하는 데에는 사용되지 않습니다. 서버가 채택한 클라이언트 ID는 클라이언트의 `Username` 또는 `ClientIdentifier`, 또는 WebSphere MQ 관리자가 작성한 ID입니다.

클라이언트 인증에 SSL 암호 스위트를 사용할 수도 있습니다.다음은 현재 지원되는 SSL 암호 스위트 목록이며 알파벳순으로 나열되어 있습니다.

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_DH_anon_WITH_RC4_128_MD5`
- `SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_DSS_WITH_AES_128_CBC_SHA`
- `SSL_DHE_DSS_WITH_DES_CBC_SHA`
- `SSL_DHE_DSS_WITH_RC4_128_SHA`
- `SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`

- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 SHA-2 암호 스위트를 사용하려는 경우 SHA-2 암호 스위트를 MQTT 채널과 함께 사용하기 위한 시스템 요구사항을 참조하십시오.

관련 개념

123 페이지의 『SSL을 사용하여 채널 인증을 위한 텔레메트리 채널 구성』
IBM WebSphere 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.

CipherSpecs 및 CipherSuites

관련 참조

DEFINE CHANNEL(MQTT)

ALTER CHANNEL(MQTT)

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 인증을 위한 텔레메트리 채널 구성

IBM WebSphere 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 사용하려면 WebSphere MQ Explorer를 사용하여 텔레메트리 채널을 구성합니다. 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.다음은 현재 지원되는 SSL 암호 스위트 목록이며 알파벳 순으로 나열되어 있습니다.

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_DH_anon_WITH_RC4_128_MD5`
- `SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_DSS_WITH_AES_128_CBC_SHA`

- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 SHA-2 암호 스위트를 사용하려는 경우 SHA-2 암호 스위트를 MQTT 채널과 함께 사용하기 위한 시스템 요구사항을 참조하십시오.

관련 개념

121 페이지의 『SSL을 사용하여 MQTT 클라이언트 인증에 필요한 텔레메트리 채널 구성』
IBM WebSphere 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.

CipherSpecs 및 CipherSuites

관련 참조
DEFINE CHANNEL(MQTT)
ALTER CHANNEL(MQTT)

텔레메트리 채널의 발행물 개인정보 보호

텔레메트리 채널에서 방향에 관계 없이 송신된 MQTT 발행물의 개인 정보는 SSL로 연결에 대해 전송을 암호화하여 보호됩니다.

텔레메트리 채널에 연결하는 MQTT 클라이언트는 채널에서 대칭 키 암호화를 사용하여 전송된 발행물의 개인 정보를 보호하기 위해 SSL을 사용합니다. 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, 123 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.

MQTT 클라이언트 및 텔레메트리 채널의 SSL 구성

MQTT 클라이언트와 WebSphere MQ Telemetry (MQXR) 서비스는 SSL을 사용하여 텔레메트리 채널을 연결하기 위해 JSSE (Java Secure Socket Extension) 를 사용합니다. 디바이스용 IBM WebSphere MQ Telemetry 디먼은 SSL을 지원하지 않습니다.

Telemetry 채널 및 MQTT 클라이언트를 인증하고 클라이언트 및 Telemetry 채널 사이의 메시지 전송을 암호화하도록 SSL을 구성하십시오.

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 사용하기 위해 Java MQTT 클라이언트와 텔레메트리 채널 간의 연결을 구성할 수 있다. 보안 개념은 사용자가 JSSE를 사용하도록 SSL을 구성하는 방법에 따라 다릅니다. 가장 안전한 구성에서 시작하여 세 가지 다른 보안 레벨을 구성할 수 있습니다.

1. 신뢰할 수 있는 MQTT 클라이언트만 연결하도록 허용합니다. MQTT 클라이언트는 신뢰할 수 있는 텔레메트리 채널에만 연결합니다. 클라이언트와 큐 관리자 사이의 메시지를 암호화하십시오. 120 페이지의 『SSL을 사용한 MQTT 클라이언트 인증』의 내용을 참조하십시오
2. MQTT 클라이언트는 신뢰할 수 있는 텔레메트리 채널에만 연결합니다. 클라이언트와 큐 관리자 사이에 메시지를 암호화하십시오. 123 페이지의 『SSL을 사용하여 텔레메트리 채널 인증』의 내용을 참조하십시오.

3. 클라이언트와 큐 관리자 사이에 메시지를 암호화하십시오. 125 페이지의 『텔레메트리 채널의 발행물 개인 정보 보호』의 내용을 참조하십시오.

JSSE 구성 매개변수

SSL 연결 구성 방법을 대체하려면 JSSE 매개변수를 수정하십시오. JSSE 구성 매개변수는 세 가지 세트로 배열됩니다.

1. IBM WebSphere MQ 텔레메트리 채널
2. MQTT Java 클라이언트
3. JRE

IBM WebSphere 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 WebSphere MQ Telemetry 채널

WebSphere MQ Explorer를 사용하여 모든 Telemetry 채널 SSL 매개변수를 설정합니다.

ChannelName

`ChannelName`은 모든 채널의 필수 매개변수입니다.

채널 이름은 특정 포트 번호와 연관된 채널을 식별합니다. MQTT 클라이언트 세트를 관리하는 데 도움이 되도록 채널 이름을 지정하십시오.

PortNumber

`PortNumber`는 모든 채널에서 선택적 매개변수입니다. 기본값은 TCP 채널의 경우 1883으로 설정되고 SSL 채널의 경우 8883으로 설정됩니다.

이 채널과 연관된 TCP/IP 포트 번호입니다. 채널에 정의된 포트를 지정하면 MQTT 클라이언트가 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

`KeyFileName`은 SSL 채널의 필수 매개변수입니다. TCP 채널의 경우에는 생략되어야 합니다.

`KeyFile`이름 은 사용자가 제공하는 디지털 인증서가 포함된 Java키 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 `Required`로 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.

텔레메트리 서비스 및 MQTT 클라이언트는 각 끝에서 사용 가능한 모든 CipherSpec의 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 자바 클라이언트

MqttConnectionOptions.SSLProperties에서 Java 클라이언트의 SSL 특성을 설정하십시오. 예를 들어, 다음과 같습니다.

```
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. Windows 의 구성 파일은 *Java Installation Directory*\jre\lib\security에 있습니다. 다음 표에서는 IBM WebSphere 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 UNIX and 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 WebSphere MQ 명령 스트레이트 (strmqikm)을 사용하여 cacerts 를 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. **strmqikm** 명령을 사용하여 **ikeyman**을 시작하거나, 강력하지만 제한적인 관할 파일을 IBM developer kits, 보안 정보에서 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 국가의 법을 확인하십시오. 암호화 소프트웨어의 수입, 소유, 사용 및 재수출과 관련된 규정 및 정책을 조사하여 허용되는지 판별하십시오.

클라이언트가 서버에 연결할 수 있도록 신뢰 제공자 수정

다음 예제에서는 MQTT 클라이언트 코드에서 신뢰 제공자를 추가하고 참조하는 방법에 대해 설명합니다. 예는 클라이언트 또는 서버의 인증을 수행하지 않습니다. 따라서 SSL 연결은 인증되지 않고 암호화됩니다.

129 페이지의 그림 25의 코드 스니펫에서는 MQTT 클라이언트에 대한 `AcceptAllProviders` 신뢰 제공자와 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.

WebSphere MQ 관리자는 JAAS를 사용한 클라이언트 인증이 필요한 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.config에 정의되어 있습니다.

jaas.config 파일은 JAAS 구성 이름에 의해 조직됩니다. 로그인 구성의 목록은 각 구성 이름 아래에 있습니다. 131 페이지의 그림 29의 내용을 참조하십시오.

JAAS에서는 네 개의 표준 로그인 모듈을 제공합니다. 표준 NT 및 UNIX 로그인 모듈의 값은 제한적입니다.

JndiLoginModule
JNDI(Java Naming and Directory Interface) 아래에 구성된 디렉토리 서비스에 대해 인증합니다.

Krb5LoginModule
Kerberos 프로토콜을 사용하여 인증합니다.

NTLoginModule
현재 사용자에 대한 NT 보안 정보를 사용하여 인증합니다.

UnixLoginModule
현재 사용자의 UNIX 보안 정보를 사용하여 인증합니다.

NTLoginModule 또는 UnixLoginModule을 사용할 때 발생하는 문제점은 텔레메트리(MQXR) 서비스가 MQTT 채널의 ID가 아니라 mqm ID로 실행된다는 점입니다. 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.java가 WebSphere MQ Telemetry와 함께 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 WebSphere MQ 디렉토리에는 클래스를 배치할 수 없습니다. 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. 텔레메트

리(MQXR) 서비스는 자체 CLASSPATH를 설정합니다. service.env에 정의된 CLASSPATH는 해당 파일에 추가됩니다.

텔레메트리(MQXR) 서비스는 MQTT 채널에 연결된 클라이언트에 Username과 Password를 리턴하는 두 개의 콜백을 제공합니다. 사용자 이름 및 비밀번호 는 MqttConnectOptions 오브젝트에 설정됩니다. Username 및 Password에 액세스하는 방법에 대한 예제는 131 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

MQTT 클라이언트가 제공하는 Username 및 Password를 수신하도록 코딩된 JAAS 로그인 모듈의 예제입니다.

```
  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

디바이스용 IBM WebSphere MQ Telemetry 디먼 개념

디바이스용 IBM WebSphere MQ Telemetry 디먼은 고급 MQTT V3 클라이언트 애플리케이션입니다. 다른 MQTT 클라이언트에서 온 메시지를 저장 후 전달하려면 이를 사용하십시오. MQTT 클라이언트처럼 IBM WebSphere MQ에 연결하지만 다른 MQTT 클라이언트도 여기 연결할 수 있습니다.

디먼은 발행/구독 브로커입니다. MQTT V3 클라이언트는 발행할 토픽 문자열 및 구독할 토픽 필터를 사용하여 토픽에 발행 및 구독하기 위해 이에 연결합니다. 토픽 문자열은 /로 구분된 토픽 레벨이 있는 계층 구조입니다. 토픽 필터는 단일 레벨 + 와일드카드 및 다중 레벨 # 와일드카드를 토픽 문자열의 마지막 부분으로 포함할 수 있는 토픽 문자열입니다.

참고: 디먼의 와일드카드는 WebSphere Message Broker, v6의 보다 제한적인 규칙을 따릅니다. IBM WebSphere MQ는 다릅니다. 이는 여러 개의 다중 레벨 와일드카드를 지원합니다. 와일드카드는 토픽 문자열에 있는 임의의 계층 구조 레벨 수를 나타냅니다.

다중 MQTT v3 클라이언트는 리스너 포트를 사용하여 디먼에 연결됩니다. 기본 리스너 포트는 수정 가능합니다. 다중 리스너 포트를 정의하고 다른 네임스페이스를 할당할 수 있습니다(139 페이지의 『디바이스용 WebSphere MQ Telemetry 디먼 리스너 포트』 참조). 디먼은 그 자체가 MQTT v3 클라이언트입니다. 다른 디먼의 리스너 포트 또는 WebSphere MQ Telemetry(MQXR) 서비스에 디먼을 연결하도록 디먼 브릿지 연결을 구성하십시오.

디바이스용 WebSphere MQ Telemetry 디먼에 대해 다중 브릿지를 구성할 수 있습니다. 브릿지를 사용하여 발행물을 교환할 수 있는 디먼의 네트워크를 함께 연결하십시오.

각 브릿지는 해당 로컬 디먼에서 토픽을 발행 및 구독할 수 있습니다. 연결되어 있는 다른 디먼, WebSphere MQ 발행/구독 브로커 또는 다른 MQTT v3 브로커에서도 토픽을 발행하고 구독할 수 있습니다. 토픽 필터를 사용하면 브로커 간에 전파할 발행물을 선택할 수 있습니다. 다른 방향으로 발행물을 전파할 수 있습니다. 로컬 디먼에서 연결된 각 리모트 브로커로, 또는 연결된 브로커에서 로컬 디먼으로 발행물을 전파할 수 있습니다(132 페이지의 『디바이스용 IBM WebSphere MQ Telemetry 디먼 브릿지』 참조).

디바이스용 IBM WebSphere MQ Telemetry 디먼 브릿지

디바이스용 IBM WebSphere MQ Telemetry 디먼 브릿지는 MQTT v3 프로토콜을 사용하여 두 개의 발행/구독 브로커를 연결합니다. 브릿지는 브로커 간에 양방향으로 발행물을 전파합니다. 한쪽 끝은 디바이스용 WebSphere MQ Telemetry 디먼 브릿지 연결이고 다른 한쪽 끝은 큐 관리자 또는 다른 디먼이 될 수 있습니다. 큐 관리자는 텔레메트리 채널을 사용하여 브릿지 연결에 연결됩니다. 디먼은 디먼 리스너를 사용하여 브릿지 연결에 연결됩니다.

디바이스용 IBM WebSphere MQ Telemetry 디먼은 다른 브로커에 대한 하나 이상의 동시 연결을 지원합니다. 디먼으로부터의 연결을 브릿지라 하며 디먼 구성 파일에서 연결 항목별로 정의됩니다. IBM WebSphere MQ에 대한 연결은 다음 그림에 표시된 대로 IBM WebSphere MQ 텔레메트리 채널을 사용하여 설정됩니다.

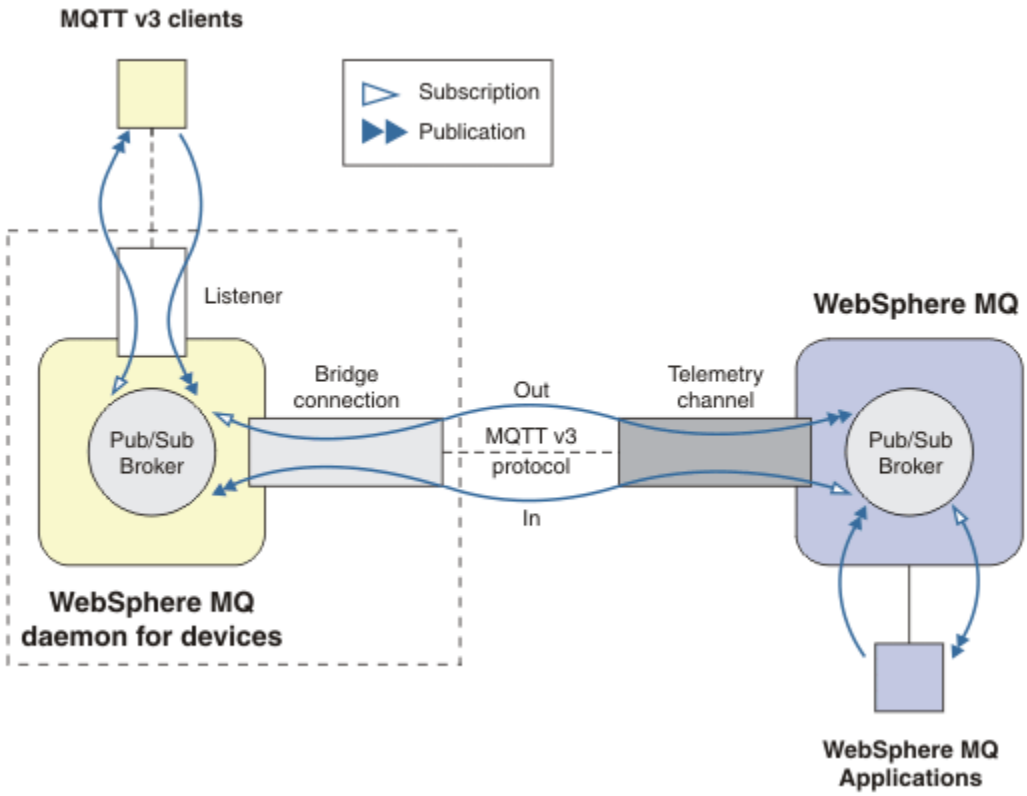


그림 *31. Connecting IBM WebSphere MQ Telemetry daemon for devices to IBM WebSphere MQ*

브릿지는 MQTT v3 클라이언트로 다른 브로커에 디먼을 연결합니다. 브릿지 매개변수는 MQTT v3 클라이언트의 속성을 미러링합니다.

브릿지는 둘 이상의 연결입니다. 이는 두 발행/구독 브로커 사이에서 발행 및 구독 에이전트의 역할을 합니다. 로컬 브로커는 디바이스용 IBM WebSphere MQ Telemetry 디먼이고 원격 브로커는 MQTT v3 프로토콜을 지원하는 발행/구독 브로커입니다. 일반적으로 원격 브로커는 다른 디먼 또는 IBM WebSphere MQ입니다.

브릿지의 역할은 두 브로커 사이에 발행물을 전파하는 것입니다. 브릿지는 양방향입니다. 다른 방향으로 발행물을 전파합니다. 133 페이지의 그림 31에서는 브릿지가 디바이스용 IBM WebSphere MQ Telemetry 디먼을 IBM WebSphere MQ에 연결하는 방식을 보여줍니다. 134 페이지의 『브릿지에 대한 토픽 설정의 예』에서는 토픽 매개변수를 사용하여 브릿지를 구성하는 방법을 보여주기 위한 예를 사용합니다.

133 페이지의 그림 31에서 In 및 Out 화살표는 브릿지의 양방향성을 나타냅니다. 화살표의 한쪽 끝에서 구독이 작성됩니다. 구독과 일치하는 발행물은 화살표 반대쪽 끝에 있는 브로커로 구독됩니다. 이 화살표는 발행물의 플로우를 따라 레이블 지정됩니다. 발행물은 디먼으로 플로우인(In)하고 디먼에서 플로우아웃(Out)합니다. 레이블의 중요성은 명령 구문에서 사용된다는 점입니다. In 및 Out은 발행물이 플로우하는 위치를 참조하고 구독이 송신되는 위치는 참조하지 않습니다.

다른 클라이언트, 애플리케이션 또는 브로커는 IBM WebSphere MQ 또는 디바이스용 WebSphere MQ Telemetry 디먼에 연결될 수 있습니다. 이는 연결되는 브로커에서 토픽을 발행 및 구독합니다. 브로커가 IBM WebSphere MQ인 경우 토픽이 클러스터되거나 분배될 수 있으며 로컬 큐 관리자에서 명시적으로 정의되지 않을 수 있습니다.

브릿지 사용

브릿지 연결 및 리스너를 사용하여 디먼을 함께 연결하십시오. 브릿지 연결 및 텔레메트리 채널을 사용하여 디먼과 큐 관리자를 함께 연결하십시오. 다중 브로커를 함께 연결할 경우 루프를 작성할 수 있습니다. 주의하십시오. 발행물은 끊임없이 발견되지 않은 브로커의 루프 주변을 순환할 수 있습니다.

IBM WebSphere MQ에 브릿지된 디먼을 사용하는 몇 가지 이유는 다음과 같습니다.

WebSphere MQ에 대한 MQTT 클라이언트 연결 수 줄이기
디먼의 계층 구조를 사용하면 단일 큐 관리자가 한 번에 연결할 수 있는 수보다 더 많은 클라이언트를 WebSphere MQ에 연결할 수 있습니다.

MQTT 클라이언트와 WebSphere MQ 간에 메시지 저장 후 전달
클라이언트가 자체 저장소를 가지고 있지 않을 경우 클라이언트와 IBM WebSphere MQ 간 지속적인 연결 유지를 피하도록 저장 후 전달을 사용할 수 있습니다. MQTT 클라이언트와 WebSphere MQ 사이에 여러 유형의 연결을 사용할 수 있습니다(모니터링 및 제어 텔레메트리 개념과 시나리오 참조).

MQTT 클라이언트와 WebSphere MQ 간에 교환된 발행물 필터
보통 발행물은 로컬로 처리되는 메시지와 다른 애플리케이션을 포함하는 메시지로 나뉩니다. 로컬 발행물에는 센서와 작동 장치 간의 제어 플로우가 포함될 수 있으며 원격 발행물에는 자료, 상태 및 구성 명령에 대한 요청이 포함될 수 있습니다.

발행물의 토픽 공간 변경
다른 리스너 포트에 연결된 클라이언트의 토픽 문자열이 다른 하나와 상충되지 않도록 하십시오. 다음 예에서는 다른 건물에서 들어오는 미터 자료에 레이블을 지정하기 위해 디먼을 사용합니다. 다른 클라이언트 그룹의 토픽 공간 분리를 참조하십시오.

브릿지에 대한 토픽 설정의 예

원격 브로커에 모든 항목 발행 - 기본값 사용

기본 방향은 out이라 하며, 브릿지는 원격 브로커에 대해 토픽을 발행합니다. topic 매개변수는 토픽 필터를 사용하여 전파되는 토픽을 제어합니다.

브릿지는 134 페이지의 그림 32에서 topic 매개변수를 사용하여 MQTT 클라이언트 또는 다른 브로커가 로컬 디먼에 발행한 모든 것을 구독합니다. 브릿지는 브릿지에 의해 연결된 리모트 브로커에 토픽을 발행합니다.

```
connection Daemon1
topic #
```

그림 *32.* 원격 브로커에 모든 항목 발행

원격 브로커에 모든 항목 발행 - 명시

다음 코드 단편의 topic 설정은 기본값을 사용하는 것과 동일한 결과를 나타냅니다. 차이점은 **direction** 매개변수가 명확하다는 점뿐입니다. 로컬 브로커인 디먼을 구독하고 원격 브로커에 발행하려면 out 방향을 사용하십시오. 브릿지가 구독하는 로컬 디먼에서 작성된 발행물은 원격 브로커에서 발행됩니다.

```
connection Daemon1
topic # out
```

그림 *33.* 원격 브로커에 모든 항목 발행 - 명시

로컬 브로커에 모든 항목 발행

out 방향을 사용하는 대신 반대 방향인 in을 설정할 수 있습니다. 다음 코드 단편은 브릿지로 연결된 원격 브로커에서 발행되는 모든 항목을 구독하도록 브릿지를 구성합니다. 브릿지는 로컬 브로커 디먼에 토픽을 발행합니다.

```
connection Daemon1
topic # in
```

그림 *34.* 로컬 브로커에 모든 항목 발행

로컬 브로커의 내보내기 토픽에서 원격 브로커의 가져오기 토픽으로 모든 항목 발행

local_prefix와 **remote_prefix**라는 두 개의 추가 토픽 매개변수를 사용하여 이전 예의 토픽 필터 #을 수정하십시오. 한 매개변수는 구독에서 사용되는 토픽 필터를 수정하는 데 사용되고, 다른 매개변수는 발행물이 발행되는 토픽을 수정하는 데 사용됩니다. 그 영향은 한 브로커에서 사용된 토픽 문자열의 시작을 다른 브로커의 다른 토픽 문자열로 대체하는 것입니다.

토픽 명령의 방향에 따라 **local_prefix** 및 **remote_prefix**의 의미가 뒤바뀝니다. 방향이 기본값인 out인 경우 **local_prefix**는 토픽 구독의 일부로 사용되고 **remote_prefix**가 원격 구독에서 토픽 문자열의 **local_prefix** 부분을 대체합니다. 방향이 in인 경우 **remote_prefix**는 원격 구독의 일부가 되며 **local_prefix**는 토픽 문자열의 **remote_prefix** 부분을 대체합니다.

토픽 문자열의 첫 번째 파트는 토픽 공간을 정의하는 것으로 볼 수 있습니다. 토픽이 발행되는 토픽 공간을 변경하려면 추가 매개변수를 사용하십시오. 전파 중인 토픽이 대상 브로커의 다른 토픽과 충돌하지 않도록 하거나 마운트 지점 토픽 문자열을 제거하기 위해 이를 수행할 수 있습니다.

예를 들어 다음 코드 단편에서는 디먼의 토픽 문자열 export/#에 대한 모든 발행물이 원격 브로커의 import/#에 다시 발행됩니다.

```
topic # out export/ import/
```

그림 *35.* 로컬 브로커의 내보내기 토픽에서 원격 브로커의 가져오기 토픽으로 모든 항목 발행

원격 브로커의 내보내기 항목에서 로컬 브로커의 가져오기 항목으로 모든 항목 발행

다음 코드 단편에서는 반대 구성을 보여줍니다. 브릿지는 원격 브로커에서 export/# 토픽 문자열이 포함되어 발행된 모든 항목을 구독하고 이를 로컬 브로커의 import/#에 발행합니다.

```
connection Daemon1
topic # in import/ export/
```

그림 *36.* 원격 브로커의 내보내기 항목에서 로컬 브로커의 가져오기 항목으로 모든 항목 발행

1884/ 마운트 지점에서 원래 토픽 문자열이 있는 원격 브로커로 모든 항목 발행

다음 코드 단편에서는 브릿지가 로컬 디먼의 마운트 지점 1884/에 연결된 클라이언트가 발행하는 모든 항목을 구독합니다. 브릿지는 마운트 지점에 발행된 모든 항목을 원격 브로커에 발행합니다. 마운트 지점 문자열 1884/는 원격 브로커에 발행된 토픽에서 제거됩니다. *local_prefix*는 마운트 지점 문자열 1884/와 동일하며, *remote_prefix*는 빈 문자열입니다.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

그림 *37. 1884/* 마운트 지점에서 원래 토픽 문자열이 있는 원격 브로커로 모든 항목 발행

다른 디먼에 연결된 다른 클라이언트의 토픽 공간 분리

빌딩에 대한 미터 읽기를 발행하기 위해 전원 미터에 대해 애플리케이션이 기록됩니다. 자료는 MQTT 클라이언트를 사용하여 동일한 건물에서 호스트된 디먼에 발행됩니다. 발행물을 위해 선택된 토픽은 power입니다. 동일한 애플리케이션이 많은 건물에 복잡하게 배치됩니다. 사이트 모니터링 및 데이터 저장을 위해 모든 건물의 자료는 브릿지 연결을 사용하여 집계됩니다. 연결은 건물 디먼을 중앙 위치의 WebSphere MQ로 링크합니다.

각 빌딩의 클라이언트 애플리케이션은 동일하지만 데이터는 빌딩마다 달라야 합니다. 각 읽기에는 power 토픽이 있으며 이를 구분하기 위해 접두부에 빌딩 번호를 지정해야 합니다. 첫 번째 건물의 브릿지는 접두부

meters/building01/를 사용하고 두 번째 건물에서는 meters/building02/라는 접두부를 사용합니다. 다른 건물의 자료도 동일한 패턴을 따릅니다. WebSphere MQ 는 meters/building01/power와 같은 주제로 표시값을 수신합니다.

이 예는 인위적인 예입니다. 사례에서 애플리케이션이 발행하는 토픽 공간은 구성 가능할 수 있습니다.

각 디먼의 구성 파일에는 다음 코드 단편의 패턴을 따르는 토픽 명령문이 있습니다.

```
connection Daemon1
topic power out "" meters/building01/
```

그림 *38.* 다른 디먼에 연결된 클라이언트의 토픽 공간 분리

사용하지 않는 local_prefix 매개변수의 플레이스홀더로 공백 문자열을 지정합니다.

동일한 디먼에 연결된 클라이언트의 토픽 공간 분리

단일 디먼이 모든 파워 미터를 연결하는 데 사용된다고 가정합니다. 애플리케이션에서 다른 포트에 연결하도록 구성할 수 있다고 가정하면 다음 코드 단편에서처럼 다른 건물에서 다른 리스너 포트로 미터를 연결하여 건물을 구분할 수 있습니다. 즉, 이 예에서는 마운트 지점이 사용되는 방법을 보여줍니다.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

그림 *39.* 동일한 디먼에 연결된 클라이언트의 토픽 공간 분리

두 방향으로 송신되는 발행물에 대해 다른 토픽 다시 맵핑

다음 코드 조각의 구성에서 브릿지는 원격 브로커의 단일 토픽 b 에 등록하고 b 에 대한 발행물을 로컬 디먼으로 전달하여 토픽을 a로 변경합니다. 브릿지는 로컬 브로커의 단일 토픽 x 에 등록하고 x 에 대한 발행물을 원격 브로커에 전달하여 토픽을 y로 변경합니다.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

그림 *40.* 두 방향으로 송신되는 발행물에 대해 다른 토픽 다시 맵핑

이 예에서 중요한 점은 다른 토픽이 두 브로커 모두에서 구독 및 발행된다는 점입니다. 두 브로커의 토픽 공간은 떨어져 있습니다.

두 방향으로 송신되는 발행물에 대해 동일한 토픽 다시 맵핑(루핑)

이전 예와 달리 일반적으로 137 페이지의 그림 41의 구성은 루프가 됩니다. 토픽 명령문 topic "" in a b에서는 브릿지가 원격으로 b를 구독하고 로컬로 a를 발행합니다. 다른 토픽 명령문에서는 브릿지가 로컬로 a를 구독하고 원격으로 b를 발행합니다. 137 페이지의 그림 42에 표시된 것처럼 동일한 구성을 작성할 수 있습니다.

일반적인 결과는 클라이언트가 b 에 원격으로 공개하는 경우 발행물이 a토픽에 대한 발행물로서 로컬 디먼으로 전송됩니다. 그러나 브릿지에 의해 a토픽의 로컬 디먼에 발행되는 경우 발행물은 브릿지가 로컬 토픽 a에 대해 작성한 구독과 일치합니다. 등록은 topic "" out a b입니다. 결과적으로 발행물은 b토픽에 대한 발행물로서 원격 브로커로 다시 전송됩니다. 이제 브릿지가 원격 토픽 b에 등록되고 순환이 다시 시작됩니다.

일부 브로커는 루프가 발생하지 않도록 하기 위해 루프 감지를 구현합니다. 그러나 루프 감지 메커니즘은 다른 유형의 브로커가 함께 브릿지될 때 작동해야 합니다. WebSphere MQ가 디바이스용 WebSphere MQ Telemetry 디먼에 브릿지될 경우 루프 감지가 작동하지 않습니다. 두 개의 디바이스용 IBM WebSphere MQ Telemetry 디먼이 함께 브릿지되는 경우에는 작동합니다. 기본적으로 루프 감지는 켜져 있습니다 (try_private 참조).

```
connection Daemon1
topic "" in a b
topic "" out a b
```

그림 *41. !*양방향으로 플로우되는 발행물에 대해 동일한 토픽을 다시 맵핑합니다.

```
connection Daemon1
topic "" both a b
```

그림 *42. !both*를 사용하여 양방향으로 플로우되는 발행물에 대해 동일한 주제를 다시 맵핑하십시오.

136 페이지의 그림 40의 구성은 137 페이지의 그림 41과 동일합니다.

IBM WebSphere MQ Telemetry daemon for devices 브릿지 연결의 가용성

사용 가능한 첫 번째 원격 브로커에 연결하도록 다중 IBM WebSphere MQ Telemetry daemon for devices 브릿지 연결 주소를 구성하십시오. 브로커가 멀티 인스턴스 큐 관리자인 경우 해당 TCP/IP 주소를 둘 다 제공하십시오. 사용 가능한 경우 1차 서버에 대해 연결 또는 다시 연결할 1차 연결을 구성하십시오.

연결 브릿지 매개변수 addresses는 TCP/IP 소켓 주소의 목록입니다. 브릿지는 연결에 성공할 때까지 각 주소에 대한 연결을 차례로 시도합니다. round_robin 및 start_type 연결 매개변수는 연결에 성공하고 난 후 주소 사용 방법을 제어합니다.

start_type이 auto, manual 또는 lazy인 경우, 이로 인해 연결이 실패할 경우, 브릿지가 재연결을 시도합니다. 이는 각 연결 시도 간에 약 20초 지연되며 차례로 각 주소를 사용합니다. `start_type`이 once인 경우, 이로 인해 연결이 실패할 경우, 브릿지는 자동으로 재연결을 시도하지 않습니다.

round_robin이 true이면 브릿지 연결 시도는 목록에 있는 첫 주소에서 시작하며 차례로 목록에 있는 각 주소를 시도합니다. 목록이 끝나면 첫 번째 주소에서 다시 시작합니다. 목록에 주소가 하나뿐인 경우 매 20초마다 다시 시도합니다.

round_robin이 false인 경우 1차 서버에서 호출되는 목록의 첫 번째 주소가 우선순위를 갖습니다. 1차 서버로의 연결을 위한 첫 번째 시도가 실패할 경우, 브릿지는 계속해서 백그라운드에서 1차 서버로 재연결을 시도합니다. 동시에 브릿지는 목록에 있는 다른 주소를 사용하여 연결을 시도합니다. 백그라운드가 1차 서버에 대한 연결 시도에 성공하면 브릿지가 현재 연결로부터 연결을 끊고 1차 서버 연결로 전환합니다.

연결이 자체적으로 끊어질 경우(예: **connection_stop** 명령에 의해) 연결이 재시작되면 동일한 주소를 다시 사용하도록 시도합니다. 연결 실패로 인해 또는 원격 브로커가 연결을 삭제하여 연결이 끊어질 경우 브릿지는 20초 동안 대기합니다. 그런 다음 목록에 있는 다음 주소로 연결을 시도하며, 목록에 주소가 하나뿐인 경우 동일한 주소로 연결을 시도합니다.

멀티 인스턴스 큐 관리자에 연결

멀티 인스턴스 큐 관리자 구성에서 큐 관리자는 다른 IP 주소를 가진 두 개의 다른 서버에서 실행됩니다. 일반적으로 텔레메트리 채널은 특정 IP 주소 없이 구성됩니다. 이러한 채널은 포트 번호로만 구성됩니다. 텔레메트리 채널이 시작되면 기본적으로 로컬 서버에서 사용 가능한 첫 번째 네트워크 주소를 선택합니다.

큐 관리자에서 사용된 두 개의 IP 주소로 브릿지 연결의 addresses 매개변수를 구성하십시오. round_robin을 true로 설정하십시오.

활성 큐 관리자 인스턴스가 실패하면 큐 관리자는 대기 인스턴스로 변경합니다. 디먼은 활성 인스턴스에 대한 연결이 중단되었음을 감지하고 대기 인스턴스로 다시 연결을 시도합니다. 이 경우 브릿지 연결용으로 구성된 주소 목록에 있는 다른 IP 주소를 사용합니다.

브릿지가 연결되는 큐 관리자는 여전히 동일한 큐 관리자입니다. 큐 관리자는 자체 상태를 복구합니다. cleansession이 False로 설정되면 브릿지 연결 세션이 장애 복구 이전과 동일한 상태로 복원됩니다. 지연 후 연결이 재개됩니다. "한 번 이상" 또는 "최대 한 번" 서비스 품질이 있는 메시지는 유실되지 않으며 등록은 계속 작동합니다.

재연결 횟수는 대기 인스턴스가 시작될 때 재시작하는 채널 및 클라이언트 수와 메시지가 발송되는 수에 따라 달라집니다. 브릿지 연결은 연결이 재설정되기 전에 여러 번 두 IP 주소 모두에 대해 다시 연결을 시도할 수 있습니다.

특정 IP 주소로 멀티 인스턴스 큐 관리자 텔레메트리 채널을 구성하지 마십시오. IP 주소는 하나의 서버에서만 유효합니다.

대체 고가용성 솔루션을 사용 중인 경우 이는 IP 주소를 관리하므로 특정 IP 주소로 텔레메트리 채널을 구성하는 것이 옳을 수 있습니다.

cleansession

브릿지 연결은 MQTT v3 클라이언트 세션입니다. 연결이 새 세션을 시작하는지 여부 또는 기존 세션을 복원하는지 여부를 제어할 수 있습니다. 기존 세션을 복원할 경우 브릿지 연결은 이전 세션의 구독 및 보유된 발행물을 보존합니다.

addresses에 다중 IP 주소가 나열되어 있고 IP 주소가 서로 다른 큐 관리자가 호스팅하는 텔레메트리 채널 또는 다른 텔레메트리 디먼에 연결되는 경우에는 cleansession을 false로 설정하지 마십시오. 세션 상태는 큐 관리자 또는 디먼 간에 전송되지 않습니다. 다른 큐 관리자 또는 디먼에서 기존 세션을 재시작하려고 하면 새 세션이 시작됩니다. 인다우트(in-doubt) 메시지가 손실되고 구독이 예상대로 작동하지 않을 수 있습니다.

notifications

애플리케이션은 브릿지 연결이 알림을 사용하여 실행 중인지 여부를 계속해서 추적할 수 있습니다. 알림은 연결된 경우 `1`, 연결되지 않은 경우 `0`이라는 값을 가진 발행입니다. 이는 `notification_topic` 매개변수로 정의되는 *topicString*에 발행됩니다. *topicString* 의 기본값은 `$SYS/broker/connection/` *clientIdentifier*`/state`입니다. 기본 *topicString* 에는 $SYS접두부가 포함되어 있습니다. $SYS로 시작하는 토픽 필터를 정의하여 $SYS로 시작하는 토픽을 구독하십시오. 토픽 필터 #(모두 구독)은 디먼에서 $SYS로 시작하는 토픽을 구독하지 않습니다. $SYS 를 응용프로그램 주제 영역과 구별되는 특수 시스템 주제 영역을 정의하는 것으로 생각하십시오.

브릿지가 연결되거나 연결이 끊길 때 알림을 사용하여 IBM WebSphere MQ Telemetry daemon for devices가 MQTT 클라이언트에 알립니다.

keepalive_interval

keepalive_interval 브릿지 연결 매개변수는 TCP/IP ping을 원격 서버로 송신하는 브릿지 사이의 간격을 설정합니다. 기본 간격은 60초입니다. ping은 TCP/IP 세션이 원격 서버 또는 방화벽에 의해 닫히지 않도록 해주며, 연결 시 비활성 기간을 감지합니다.

clientId

브릿지 연결은 MQTT v3 클라이언트 세션으로, 브릿지 연결 매개변수 clientid에 의해 설정된 `clientIdentifier`를 가집니다. cleansession 매개변수를 false로 설정하여 이전 세션을 재개하기 위해 다시 연결을 시도할 경우 각 세션에 사용된 `clientIdentifier`는 동일해야 합니다. clientid의 기본값은 동일하게 남아 있는 *hostname.connectionName*입니다.

디바이스용 WebSphere MQ Telemetry 디먼의 설치, 확인, 구성 및 제어

디먼의 설치, 구성 및 제어는 파일 기반입니다.

디먼 을 실행할 디바이스에 소프트웨어 개발 킷을 복사하여 디먼을 설치하십시오.

예를 들어, MQTT 클라이언트 유틸리티를 실행하고 디바이스용 WebSphere MQ 텔레메트리 디먼을 발행/구독 브로커로 연결하십시오. 특정한 MQTT v3 클라이언트에 메시지를 발행합니다. 를 참조하십시오.

구성 파일을 작성하여 디먼을 구성하십시오(디바이스용 MQ Telemetry 디먼 구성 파일 참조).

amqtdd.upd 파일에서 명령을 작성하여 실행 중인 디먼을 제어하십시오. 5초마다 디먼이 파일을 읽고, 명령을 실행하고 해당 파일을 삭제합니다(디바이스용 WebSphere MQ Telemetry 디먼 명령 파일 참조).

디바이스용 WebSphere MQ Telemetry 디먼 리스너 포트

리스너 포트를 사용하여 디바이스용 WebSphere MQ Telemetry 디먼에 MQTT V3 클라이언트를 연결하십시오. 마운트 지점 및 최대 연결 수로 리스너 포트에 권한을 부여할 수 있습니다.

리스너 포트는 이 포트에 연결되어 있는 클라이언트의 MQTT 클라이언트 connect(serverURI) 메소드에 지정된 포트 번호를 따라야 합니다. 이는 클라이언트와 디먼 모두에서 기본값을 1883로 지정합니다.

디먼 구성 파일에서 글로벌 정의 port를 설정하면 디먼에 대한 기본 포트를 변경할 수 있습니다. listener 정의를 디먼 구성 파일에 추가하여 특정 포트를 설정할 수 있습니다.

기본 포트가 아닌 각 리스너 포트의 경우 마운트 지점을 지정하여 클라이언트를 분리할 수 있습니다. 마운트 지점이 있는 포트에 연결된 클라이언트는 다른 클라이언트로부터 분리됩니다(139 페이지의 『디바이스용 WebSphere MQ Telemetry 디먼 마운트 지점』 참조).

임의의 포트에 연결할 수 있는 클라이언트의 수를 제한할 수 있습니다. 기본 포트에 대한 연결을 제한하도록 글로벌 정의 max_connections를 설정하거나 각 리스너 포트에 max_connections 권한을 부여하십시오.

예

기본 포트를 1883에서 1880으로 변경하고 포트 1880 - 10000으로 연결을 제한하는 구성 파일의 예입니다. 포트 1884에 대한 연결은 1000으로 제한됩니다. 포트 1884에 연결된 클라이언트는 다른 포트에 연결된 클라이언트로부터 분리됩니다.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

디바이스용 WebSphere MQ Telemetry 디먼 마운트 지점

MQTT 클라이언트에서 사용하는 리스너 포트와 마운트 지점을 연관시켜 디바이스용 WebSphere MQ Telemetry 디먼에 연결할 수 있습니다. 마운트 지점은 다른 리스너 포트에 연결된 MQTT 클라이언트의 리스너 포트 하나를 사용하여 MQTT 클라이언트에 의해 교환된 발행물 및 구독을 격리합니다.

마운트 지점이 있는 리스너 포트에 연결된 클라이언트는 다른 리스너 포트에 연결된 클라이언트와 절대로 직접 토픽을 교환할 수 없습니다. 마운트 지점이 없는 리스너 포트에 연결된 클라이언트는 클라이언트의 토픽을 발행 또는 구독할 수 있습니다. 클라이언트는 마운트 지점을 통해 연결되어 있는지 여부를 인식하지 못합니다. 이는 클라이언트에 의해 작성된 토픽 문자열과 차이가 없습니다.

마운트 지점은 발행물과 구독의 토픽 문자열에 접두부로 지정되는 텍스트의 문자열입니다. 마운트 지점이 있는 리스너 포트에 연결된 클라이언트에 의해 작성된 모든 토픽 문자열에는 접두부가 지정됩니다. 텍스트 문자열은 리스터 포트에 연결된 클라이언트로 송신된 모든 토픽 문자열에서 제거됩니다.

리스너 포트에 마운트 지점이 없으면 포트에 연결된 클라이언트에 의해 작성되고 수신된 발행물 및 구독의 토픽 문자열이 대체되지 않습니다.

후미 문자 /가 포함된 마운트 지점 문자열을 작성하십시오. 해당 마운트 지점은 마운트 지점에 대한 토픽 트리의 상위 토픽입니다.

예

구성 파일에는 다음과 같은 리스너 포트가 포함되어 있습니다.

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

포트 1883에 연결된 클라이언트는 MyTopic에 대한 구독을 작성합니다. 디먼은 1883/MyTopic으로서 구독을 등록합니다. 포트 1883에 연결된 다른 클라이언트는 토픽 MyTopic에 메시지를 발행합니다. 디먼은 토픽 문자열을 1883/MyTopic으로 변경하고 구독과의 일치를 검색합니다. 포트 1883의 구독자는 원래 토픽 문자열 MyTopic이 있는 발행물을 수신합니다. 디먼이 토픽 문자열에서 마운트 지점 접두부를 제거했습니다.

포트 1884에 연결된 다른 클라이언트도 토픽 MyTopic을 발행합니다. 이 때는 디먼이 토픽을 1884/MyTopic으로 구독합니다. 다른 마운트 지점에는 다른 토픽 문자열이 있는 구독이 있으므로 포트 1883의 구독자는 해당 발행물을 수신하지 않습니다.

포트 1885에 연결된 클라이언트는 토픽 1883/MyTopic을 발행합니다. 디먼은 토픽 문자열을 변경하지 않습니다. 포트 1883의 구독자는 MyTopic에 대한 발행물을 수신합니다.

디바이스용 WebSphere MQ Telemetry 디먼 QoS(Quality of Service), 지속적 구독 및 보유된 발행

QoS(Quality of Service) 설정은 실행 중인 디먼에만 적용됩니다. 디먼이 중지되면 제어되는 방식으로든 아니면 실패로 인해서든 전달 중인 메시지의 상태가 손실됩니다. 디먼이 중지되면 최소 한 번 또는 최대 한 번이라는 메시지 전달을 보장할 수 없습니다. 디바이스용 WebSphere MQ Telemetry 디먼은 제한된 지속성을 지원합니다. 디먼이 종료될 경우 보유된 발행 및 구독을 저장하도록 **retained_persistence** 구성 매개변수를 설정하십시오.

WebSphere MQ와 달리 용 WebSphere MQ Telemetry 디먼은 지속적 데이터를 저널하지 않습니다. 세션 상태, 메시지 상태 및 보유된 발행은 트랜잭션 방식으로 저장되지 않습니다. 기본적으로 디먼은 중지 시 모든 데이터를 제거합니다. 정기적으로 체크포인트 구독 및 보유된 발행에 대한 옵션을 설정할 수 있습니다. 메시지 상태는 디먼이 중지되면 항상 손실됩니다. 보유되지 않는 모든 발행은 손실됩니다.

정기적으로 보유된 발행을 파일에 저장하려면 디먼 구성 옵션 Retained_persistence를 true로 설정하십시오. 디먼이 재시작되면 마지막으로 자동 저장된 보유된 발행물이 복원됩니다. 기본적으로 클라이언트가 작성한 보유 메시지는 디먼 재시작 시 복원되지 않습니다.

정기적으로 지속 세션에서 작성된 구독을 파일에 저장하려면 디먼 구성 옵션 Retained_persistence를 true로 설정하십시오. Retained_persistence가 true로 설정되고 CleanSession이 있는 세션에서 클라이언트가 작성하는 구독이 false로 설정된 경우 "지속 세션"이 복원됩니다. 디먼이 재시작되면 구독을 복원하여 발행물 수신을 시작합니다. 클라이언트는 CleanSession을 false로 하여 다시 시작할 때 발행물을 수신합니다. 기본적으로 디먼이 중지하면 클라이언트 세션 상태가 저장되지 않습니다. 따라서 클라이언트가 CleanSession을 false로 설정하더라도 구독은 복원되지 않습니다.

Retained_persistence는 자동 저장 메커니즘입니다. 따라서 가장 최근에 보유된 발행물 또는 구독은 저장할 수 없습니다. 사용자는 보유된 발행물 및 구독이 저장되는 빈도를 변경할 수 있습니다. 구성 옵션 autosave_on_changes 및 autosave_interval을 사용하여 저장 사이의 간격 또는 저장 사이의 변경 수를 설정하십시오.

설정 지속성에 대한 구성의 예

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
port 1882
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

디바이스용 WebSphere MQ Telemetry 디먼 보안

디바이스용 WebSphere MQ Telemetry 디먼은 이에 연결하는 클라이언트를 인증하고, 신임 정보를 사용하여 다른 브로커에게 연결하며, 토픽에 대한 액세스를 제어합니다. 디먼이 제공하는 보안은 SSL 지원을 제공하지 않는 WebSphere MQ Telemetry C 클라이언트를 사용하여 빌드되도록 제한됩니다. 결과적으로 디먼에 대한 연결은 암호화되지 않으며 인증서를 사용하여 인증할 수 없게 됩니다.

기본적으로 보안은 켜져 있지 않습니다.

클라이언트 인증

MQTT 클라이언트는 메소드 `MqttConnectOptions.setUserName` 및 `MqttConnectOptions.setPassword`를 사용하여 사용자 이름 및 비밀번호를 설정할 수 있습니다.

비밀번호 파일의 항목에 대응하는 클라이언트에 의해 제공된 사용자 이름 및 비밀번호를 검사하여 디먼에 연결하는 클라이언트를 인증하십시오. 인증을 사용 가능하게 하려면 비밀번호 파일을 작성하고 디먼 구성 파일에서 `password_file` 매개변수를 설정하십시오(password_file 참조).

인증을 검사 중인 디먼에 연결하기 위해 사용자 이름 또는 비밀번호 없이 클라이언트 연결을 허용하도록 디먼 구성 파일의 `allow_anonymous` 매개변수를 설정하십시오(allow_anonymous 참조). 클라이언트가 사용자 이름 또는 비밀번호를 제공할 경우 `password_file` 매개변수가 설정되면 사용자 이름 또는 비밀번호가 비밀번호 파일에 대응하여 항상 검사됩니다.

특정 클라이언트에 대한 연결을 제한하려면 디먼 구성 파일의 `clientid_prefixes` 매개변수를 설정하십시오. 클라이언트는 `clientid_prefixes` 매개변수에 나열된 접두부 중 하나로 시작하는 `clientIdentifiers`가 있어야 합니다(clientid_prefixes 참조).

브릿지 연결 보안

디바이스용 각 WebSphere MQ Telemetry 디먼 브릿지 연결은 MQTT V3 클라이언트입니다. 디먼 구성 파일에서 브릿지 연결 매개변수로 각 브릿지 연결을 위한 사용자 이름 및 비밀번호를 설정할 수 있습니다(username 및 password 참조). 그러면 브릿지가 브로커에 대해 자체 인증할 수 있습니다.

토픽의 액세스 제어

클라이언트가 인증되는 중인 경우 디먼도 토픽에 대한 액세스 제어를 각 사용자에게 제공할 수 있습니다. 디먼은 클라이언트가 액세스 제어 파일에 액세스 토픽 문자열이 있는 발행 또는 구독 중 하나인 토픽과의 일치를 기반으로 액세스 제어 권한을 부여합니다(acl_file 참조).

액세스 제어 목록(ACL)은 두 부분으로 나눠집니다. 첫 번째 부분은 익명 클라이언트를 포함한 모든 클라이언트에 대한 액세스를 제어합니다. 두 번째 부분은 비밀번호 파일에 임의 사용자에 대한 섹션을 가집니다. 여기에는 각 사용자에 대한 특정 액세스 제어가 나열됩니다.

예

보안 매개변수는 다음 예에 표시됩니다.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

그림 *43.* 디먼 구성 파일

```
Fred:Fredpassword
Barney:Barneypassword
```

그림 *44. Password file, passwords.txt*

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

그림 *45.* 액세스 제어 파일 *acl.txt*

멀티캐스트 관리

이 정보를 사용하여 WebSphere MQ 멀티캐스트 관리 태스크(예: 멀티캐스트 메시지의 크기 줄이기 및 데이터 변환 사용)에 대해 학습할 수 있습니다.

멀티캐스트 시작하기

이 정보를 사용하여 WebSphere MQ 멀티캐스트 토픽 및 통신 정보 오브젝트를 시작할 수 있습니다.

이 태스크 정보

WebSphere MQ 멀티캐스트 메시징은 토픽을 그룹 주소로 맵핑하여 메시지를 전달하는 데 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 작성된 후 amqspubc 샘플 및 amqssubc 샘플을 사용하여 이러한 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 것입니다.
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 143 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 WebSphere MQ 멀티캐스트 토픽 토폴로지

이 예를 사용하여 IBM WebSphere MQ 멀티캐스트 토픽 토폴로지를 이해하십시오.

IBM WebSphere 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 - 239.255.255.255`지만, 이러한 주소 중 일부는 예약된 주소입니다. 예약된 주소 목록은 시스템 관리자에게 문의하거나 자세한 정보는 IPv4 멀티캐스트 주소 공간 레지스트리를 참조하십시오. `239.0.0.0 - 239.255.255.255`범위에서 로컬로 범위가 지정된 멀티캐스트 주소를 사용하는 것이 좋습니다.

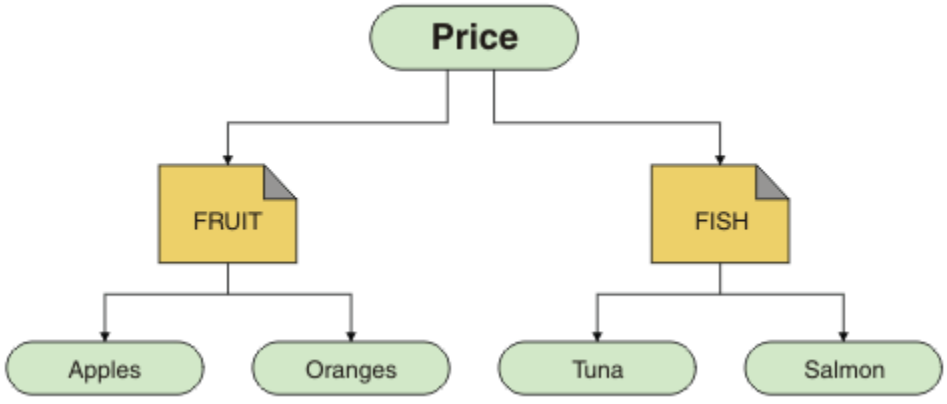
다음 다이어그램에는 사용 가능한 두 개의 멀티캐스트 데이터 스트림이 있습니다.

```
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 예제에서 FRUIT 토픽은 COMMINFO 오브젝트 MC1를 사용하도록 정의되고 FISH 토픽은 COMMINFO 오브젝트 MC2를 사용하도록 정의되고 Price 노드에는 멀티캐스트 정의가 없습니다.

WebSphereMQ 멀티캐스트에서는 토픽 문자열의 길이가 255자로 제한됩니다. 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

이 정보를 사용하여 WebSphere MQ 메시지 형식에 대해 학습하고 WebSphere MQ 메시지의 크기를 줄일 수 있습니다.

WebSphere MQ 메시지에 메시지 디스크립터에 포함된 메시지와 연관된 많은 속성이 있습니다. 작은 메시지의 경우, 이러한 속성은 대부분의 데이터 트래픽을 나타내고, 전송 속도에 대한 중요한 악영향을 가질 수 있습니다. WebSphere MQ 멀티캐스트를 사용하면 사용자가 이러한 속성 중 메시지와 함께 전송하는 속성을 구성할 수 있습니다.

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 144 페이지의 표 9에 설명되어 있습니다.

COMMINFO 오브젝트에는 메시지로 플로우하는 MQMD 필드 및 사용자 특성 수를 제어하는 MCPROP 속성이 포함됩니다. 이 속성 값을 적합한 레벨로 설정하여 WebSphere 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 값을 사용하면 RMM에 대한 호환 가능한 모드에서 메시지를 전송할 수 있으며 이로 인해 현재 XMS 애플리케이션 및 WebSphere Message Broker RMM 애플리케이션과의 상호 운영이 가능합니다.

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

표 9. 메시징 속성 및 멀티캐스트와의 관련 방법 (계속)		
속성	**멀티캐스트를 사용할 때 조치**	**전송되지 않는 경우 기본값**
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
UserProperties	포함됨	적용할 수 없음

관련 참조

ALTER COMMINFO

DEFINE COMMINFO

멀티캐스트 메시징에 대한 데이터 변환 사용 가능

이 정보를 사용하여 WebSphere MQ 멀티캐스트 메시징을 위해 데이터 변환을 사용하는 방법에 대해 학습할 수 있습니다.

WebSphere MQ 멀티캐스트는 공유된 연결되지 않은 프로토콜이기 때문에 각 클라이언트가 데이터 변환을 위한 특정 요청을 작성할 수 없습니다. 동일한 멀티캐스트 스트림을 구독하는 모든 클라이언트는 동일한 2진 데이터를 수신하므로, WebSphere MQ 데이터 변환이 필요한 경우, 각 클라이언트에서 로컬로 변환이 수행됩니다.

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.

WebSphere MQ 멀티캐스트에서는 다음 기본 제공 형식에 맞게 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 WebSphere MQ 멀티캐스트 관리 및 모니터링에 대해 학습할 수 있습니다.

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** 를 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 *reliability indicator*는 WebSphere MQ 멀티캐스트 발행자 및 구독자 모니터링에서 중요한 요소입니다. 신뢰도 표시기(발행 또는 구독 상태 명령에서 리턴되는 **MSGREL** 필드)는 오류가 없는 전송 백분율에 대해 보여주는 WebSphere MQ 표시기입니다. 때때로 전송 오류가 발생하면 메시지를 재전송해야 하며 이 오류는 **MSGREL** 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 작은 경우 이벤트 메시지가 작성됩니다. 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 WebSphere MQ 멀티캐스트 구독 및 메시지 실행 기록을 설정하는 방법을 학습할 수 있습니다.

멀티캐스트에 대한 전송 실패를 해결하기 위한 중요한 요소는 전송된 데이터의 WebSphere MQ 버퍼링입니다(링크의 전송 끝에 보관할 메시지의 실행 기록). 이 프로세스는 WebSphere MQ가 신뢰를 제공하므로 넣기 애플리케이션 프로세스에 메시지 버퍼링이 필요하지 않다는 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.

WebSphere MQ 멀티캐스트 메시지 실행 기록이 **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.

0 값은 최소 레벨의 신뢰도를 제공합니다. 기본값은 100KB입니다.

WebSphere MQ 멀티캐스트의 새 구독 실행 기록이 **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 값은 전송기가 등록 시간에서 작성된 발행물만 전송하도록 합니다. NONE은 기본값입니다.

모두
ALL 값을 사용하면 전송자가 알려진 것처럼 주제의 실행기록이 다시 전송됩니다. 어떤 경우, 이 상환은 보유된 발행물에 유사한 작동을 제공할 수 있습니다.

참고: ALL 값을 사용하면 모든 토픽 실행기록이 재전송되기 때문에 큰 토픽 실행기록이 있는 경우 성능에 해로운 영향을 줄 수 있습니다.

관련 참조
DEFINE COMMINFO
ALTER COMMINFO

고급 멀티캐스트 태스크

이 정보를 사용하여 고급 WebSphere MQ 멀티캐스트 관리 태스크(예: .ini 파일 구성) 및 WebSphere MQ LLM과의 상호 운용성에 대해 학습할 수 있습니다.

멀티캐스트 설치의 보안에 대한 고려사항은 멀티캐스트 보안 을 참조하십시오.

멀티캐스트 및 비멀티캐스트 발행/구독 도메인 사이의 브릿징

이 정보를 사용하여 비멀티캐스트 발행자가 WebSphere MQ 멀티캐스트 사용 토픽을 발행할 때 발생하는 상황을 파악할 수 있습니다.

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` 파일의 WebSphere MQ 멀티캐스트 필드를 이해할 수 있습니다.

`ini` 파일에서 추가 WebSphere MQ 멀티캐스트를 구성할 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 WebSphere MQ 데이터 디렉토리의 위치(/var/mqm/mqclient.ini)이며, *QMNAME*은 `.ini` 파일이 적용되는 큐 관리자의 이름입니다.

`.ini` 파일에는 WebSphere MQ 멀티캐스트의 동작을 미세 조정하는 데 사용되는 필드가 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 통신합니다. 이 값은 기본값입니다.

IPV6
IPv6 프로토콜만 사용하여 통신합니다.

ANY
사용 가능한 프로토콜에 따라 IPv4, IPv6 또는 두 가지 모두를 사용하여 통신합니다.

BOTH
IPv4와 IPv6 모두 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.

WebSphere MQ Low Latency Messaging과 멀티캐스트의 상호 운용성

이 정보를 사용하여 WebSphere MQ 멀티캐스트와 WebSphere MQ Low Latency Messaging(LLM) 사이의 상호 운용성을 파악할 수 있습니다.

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 WebSphere MQ 멀티캐스트를 모두 설치하고 두 제품을 따로 운영하고 서비스 제공할 수 있습니다.

멀티캐스트와 통신하는 LLM 애플리케이션은 메시지 특성을 보내고 수신해야 할 수도 있습니다. WebSphere MQ 메시지 특성 및 MQMD 필드가 다음 표에 표시된 특정 LLM 메시지 특성 코드를 가진 LLM 메시지 특성으로 전송됩니다.

표 *10. WebSphere MQ* 메시지 특성 대 *WebSphere MQ LLM* 특성 맵핑

WebSphere MQ 특성	**WebSphere 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(WebSphere MQ 낮은 대기 시간 메시징)를 참조하십시오.

HP Integrity NonStop Server 관리

이 정보를 사용하여 HP Integrity NonStop Server의 IBM WebSphere 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>
   ```

   애플리케이션은 새 등록 요청을 작성할 수 없으며 **THAW** 명령을 실행할 때까지 **START** 명령을 실행할 수 없습니다.

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.

라이센싱 IBM IBM 디렉터 North Castle Drive Armonk, NY 10504-1785 U.S.A.

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

지적재산권 라이센스 법률 및 지적 재산권에 관한 법률 IBM 일본, 1921-21, Nihonbashi-Hakozakzakicho, 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 비IBM 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.

IBM Corporation Software Interoperability Coordinator, Department 49XA 3605고속도로 52 N Rochester, MN 55901 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.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of IBM WebSphere MQ.

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: