9.3

IBM MQ 기술 개요

참고

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 273 페이지의 『주의사항』에 있는 정보를 확인하십시오.

이 개정판은 새 개정판에 별도로 명시하지 않는 한, IBM® MQ 의 버전 9릴리스 3및 모든 후속 릴리스와 수정에 적용됩니다.

IBM은 귀하가 IBM으로 보낸 정보를 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 사용하거나 배포할 수 있습니다.

목차

IBM MQ 기술 개요

IBM MQ를 사용하여 애플리케이션을 연결하고 조직에서 정보 분배를 관리하십시오.

IBM MQ에서는 프로그램이 일관된 API(Application Programming Interface)를 사용하여 서로 다른 컴포넌트(프로세서, 운영 체제, 서브시스템 및 통신 프로토콜)가 존재하는 네트워크에서 서로 다른 항목과 통신할 수 있습니다. 이 인터페이스를 사용하여 설계 및 기록된 애플리케이션은 메시지 큐잉 애플리케이션이라고 합니다.

다음 하위 주제를 사용하여 메시지 큐잉 및 IBM MQ에서 제공되는 다른 기능에 대해 확인하십시오.

관련 개념

IBM MQ 소개

제품 요구사항 및 지원 정보를 제공하는 위치

관련 태스크

IBM MQ 아키텍처 계획

관련 참조

6 페이지의 『메시지 큐잉의 기본 기능 및 이점』

이 정보는 메시지 큐잉의 일부 기능 및 이점을 강조합니다. 이 정보에서는 메시지 큐잉의 보안 및 데이터 무결성과 같은 기능을 설명합니다.

메시지 큐잉 소개

IBM MQ 제품을 사용하면 프로그램이 일관된 API(Application Programming Interface)를 사용하여 다른 컴포넌트(프로세서, 운영 체제, 서브시스템 및 통신 프로토콜)의 네트워크에서 서로 통신할 수 있습니다.

이 인터페이스를 사용하여 설계되고 작성된 애플리케이션은 메시징 및 큐잉 스타일을 사용하기 때문에 메시지 큐잉 애플리케이션이라고 합니다.

- 메시징은 프로그램이 서로 직접 호출하지 않고 메시지의 각 기타 데이터를 송신하여 통신하는 것을 의미합니다.
- 큐잉은 메시지가 스토리지의 큐에 배치되어 프로그램이 서로 독립적으로 다른 속도와 시간에 다른 위치에서 논리 연결 없이 실행할 수 있도록 하는 것을 의미합니다.

메시지 큐잉이 다년간 데이터 처리에 사용되어 왔습니다. 메시지 큐잉은 전자 메일에서 가장 많이 사용됩니다. 큐잉 없이 전자 메시지를 장거리 송신하는 경우 라우트의 모든 노드가 메시지 전달에 대해 사용 가능해야 하고 주소가 로그온되어야 하며 사용자가 이 주소에 메시지 송신을 시도한다는 사실에 대해 인식하고 있어야 합니다. 큐잉 시스템에서 시스템이 메시지를 전달할 준비가 될 때까지 메시지는 중간 노드에 저장됩니다. 최종 목적지에서 주소가 메시지를 읽을 준비가 될 때까지 메시지는 전자 편지함에 저장됩니다.

그렇다 하더라도 다수의 복잡한 비즈니스 트랜잭션이 오늘날 큐잉 없이 처리됩니다. 대형 네트워크에서 시스템은 사용 준비 상태인 수천의 연결을 유지보수하고 있을 수 있습니다. 시스템의 한 부분에 문제가 생기는 경우 시스템의 많은 부분이 사용 불가능하게 됩니다.

메시지 큐잉을 프로그램용 전자 메일로 생각할 수 있습니다. 메시지 큐잉 환경에서 애플리케이션 스위트를 구성하는 각 프로그램은 특정 요청에 대한 응답으로 명확하고 독립적인 기능을 수행합니다. 다른 프로그램과 통신하려면 프로그램이 사전정의된 큐에 메시지를 넣어야 합니다. 다른 프로그램은 큐에서 메시지를 검색하고 메시지에 포함된 요청 및 정보를 처리합니다. 따라서 메시지 큐잉은 프로그램 대 프로그램 통신의 스타일입니다.

큐잉은 애플리케이션이 메시지를 처리할 준비가 될 때까지 메시지가 보유되는 메커니즘입니다. 큐잉을 통해 다음을 수행할 수 있습니다.

- 통신 코드를 작성할 필요 없이 프로그램(각각 다른 환경에서 실행될 수 있음) 간에 통신합니다.
- 프로그램이 메시지를 처리하는 순서를 선택하십시오.
- 메시지 수가 임계값을 초과할 때 둘 이상의 프로그램이 큐를 서비스하도록 구성하여 시스템에 로드를 밸런스화합니다.
- 1차 시스템이 사용 가능하지 않은 경우 대체 시스템이 큐를 서비스하도록 구성하여 애플리케이션의 사용가능성을 증가시키십시오.

메시지 큐 개념

간단하게 큐라고 하는 메시지 큐는 메시지를 송신할 수 있는 이름 지정된 목적지입니다. 큐를 서비스하는 프로그램이 메시지를 검색할 때까지 메시지는 큐에 누적됩니다.

큐 관리자에서 큐가 상주하고 관리됩니다(8 페이지의 『메시지 큐잉 용어』 참조). 큐의 물리적 특성은 큐 관리자가 실행되는 운영 체제에 따라 다릅니다. 큐는 컴퓨터 메모리의 휘발성 버퍼이거나 영구 스토리지 디바이스(예: 디스크)에서 데이터 세트일 수 있습니다. 큐의 실제 관리는 큐 관리자가 담당하며 참여하는 애플리케이션 프로그램에 대해 분명하지 않게 작성됩니다.

프로그램은 큐 관리자의 외부 서비스를 통해서만 큐에 액세스할 수 있습니다. 프로그램은 큐를 열고, 큐로(부터) 메시지를 넣고, 메시지를 가져오고 큐를 닫을 수 있습니다. 또한 큐의 속성을 설정 및 조회할 수 있습니다.

메시지 큐잉의 다른 스타일

포인트-투-포인트

큐에는 메시지가 한 개만 배치되며 하나의 애플리케이션이 해당 메시지를 수신합니다.

포인트-투-포인트 메시징에서 송신 애플리케이션이 수신 애플리케이션에 대한 정보를 알아야 수신 애플리케이션으로 메시지를 송신할 수 있습니다. 예를 들어, 송신 애플리케이션은 정보를 송신할 큐 이름을 알아야 하며 큐 관리자 이름도 지정할 수 있습니다.

발행/구독

발행 애플리케이션이 발행하는 각 메시지의 사본은 모든 관련 애플리케이션으로 전달됩니다. 관련 애플리케이션이 하나 또는 여러 개일 수 있으며, 전혀 없을 수도 있습니다. 발행/구독에서 관심 애플리케이션은 구독자로 알려져 있으며 메시지는 구독으로 식별되는 큐에 삽입됩니다.

발행/구독 메시징을 사용하면 정보의 제공자를 해당 정보의 이용자와 분리할 수 있습니다. 송신 애플리케이션과 수신 애플리케이션은 송수신할 정보를 위해 서로에 대해 알지 않아도 됩니다. 자세한 정보는 55 페이지의 『발행/구독 메시징』의 내용을 참조하십시오.

애플리케이션 설계자 및 개발자에 대한 메시지 큐잉의 이점

IBM MQ를 통해 애플리케이션 프로그램은 메시지 큐잉을 사용하여 메시지 구동 처리에 참여할 수 있습니다. 애플리케이션 프로그램은 적절한 메시지 큐잉 소프트웨어 제품을 사용하여 여러 플랫폼 사이에서 통신할 수 있습니다. 예를 들어, z/OS® 애플리케이션은 IBM MQ for z/OS를 통해 통신할 수 있습니다. 애플리케이션은 근본적인 통신의 메커닉으로부터 보호됩니다. 메시지 큐잉의 그 밖의 이점은 다음과 같습니다.

- 여러 애플리케이션에서 공유할 수 있는 소형 프로그램을 사용하여 애플리케이션을 설계할 수 있습니다.
- 이러한 빌딩 블록을 다시 사용하여 새 애플리케이션을 빠르게 빌드할 수 있습니다.
- 메시지 큐잉 기술을 사용하도록 작성된 애플리케이션은 큐 관리자가 작업하는 도중에 발생한 변경사항에 영향을 받지 않습니다.
- 통신 프로토콜을 사용할 필요가 없습니다. 큐 관리자는 사용자를 위해 통신의 모든 측면을 다룹니다.
- 메시지가 프로그램에 송신될 때 메시지를 수신하는 프로그램이 실행 중일 필요는 없습니다. 메시지는 큐에 보유됩니다.

메시지 큐잉을 사용하지 않는 애플리케이션에 비해 프로그래밍 기술에 대한 요구가 낮아지고, 필요한 개발자의 수가 적어지고 개발이 빨라지기 때문에 설계자는 애플리케이션 비용을 줄일 수 있습니다.

IBM MQ는 애플리케이션이 실행되는 위치마다 메시지 큐 인터페이스(또는 MQI)라는 공용 애플리케이션 프로그래밍 인터페이스를 구현합니다. 이를 통해 하나의 플랫폼에서 다른 플랫폼으로 애플리케이션 프로그램을 쉽게 포트시킬 수 있습니다.

MQI에 대한 자세한 내용은 MQI(Message Queue Interface) 개요를 참조하십시오.

메시지 큐잉의 기본 기능 및 이점

이 정보는 메시지 큐잉의 일부 기능 및 이점을 강조합니다. 이 정보에서는 메시지 큐잉의 보안 및 데이터 무결성과 같은 기능을 설명합니다.

메시지 큐잉 기술을 사용하는 애플리케이션의 기본 기능은 다음과 같습니다.

- 7 페이지의 『프로그램 간의 직접 연결이 없음』
- 8 페이지의 『시간 독립 통신』
- 8 페이지의 『소형 프로그램』
- 8 페이지의 『메시지 구동 처리』
- 8 페이지의 『이벤트 중심 처리』
- 8 페이지의 『메시지 우선순위』
- 8 페이지의 『보안』
- 8 페이지의 『데이터 무결성』
- 8 페이지의 『복구 지원』

참고: IBM MQ 클라이언트 및 서버를 고려할 때 새 플랫폼에서의 추가 IBM MQ MQI clients 지원을 위해 서버 애플리케이션을 변경할 필요가 없습니다. 마찬가지로, IBM MQ MQI client는 변경 없이도 추가 유형의 서버와 함께 작동할 수 있습니다.

프로그램 간의 직접 연결이 없음

메시지 큐잉은 간접적 프로그램 대 프로그램 통신용 기술입니다. 프로그램이 서로 통신하는 모든 애플리케이션 내에서 이 메시지 큐잉을 사용할 수 있습니다. 메시지를 큐 관리자가 소유한 큐에 넣는 하나의 프로그램과 큐에서 메시지를 가져오는 다른 하나의 프로그램에 의해서 통신이 발생합니다.

프로그램은 다른 프로그램이 큐에 넣는 메시지를 가져올 수 있습니다. 다른 프로그램은 수신 프로그램과 동일한 큐 관리자 또는 다른 큐 관리자에 연결될 수 있습니다. 이러한 다른 큐 관리자는 다른 시스템, 다른 컴퓨터 시스템 상에 있거나 다른 비즈니스 또는 엔터프라이즈 내에 있을 수도 있습니다.

메시지 큐를 사용하여 통신하는 프로그램 간에는 물리적 접속이 없습니다. 프로그램이 큐 관리자가 소유하는 큐에 메시지를 송신하고 다른 프로그램은 해당 큐에서 메시지를 검색합니다(7 페이지의 그림 1 참조).

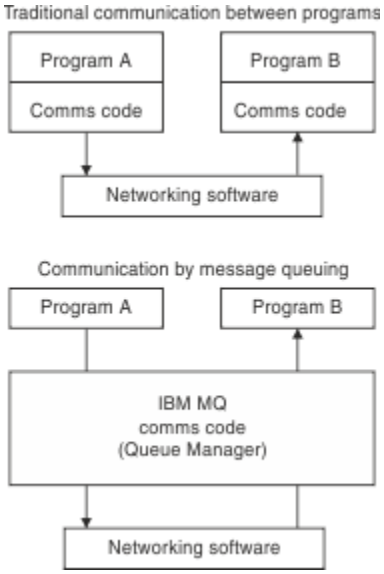


그림 *1.* 일반 통신과 메시지 큐잉 비교

전자 메일의 경우와 마찬가지로 트랜잭션의 일부인 개별 메시지는 저장 및 전달 네트워크를 통해 이동합니다.기초. 노드 간의 링크가 실패하면 링크가 복원되거나 연산자 또는 프로그램이 메시지를 다시 전송할 때까지 메시지가 보관됩니다.

메시지가 큐에서 큐로 이동하는 메커니즘은 프로그램으로부터 숨겨집니다. 따라서 프로그램이 좀 더 단순해집니다.

시간 독립 통신

다른 프로그램에 작업 수행을 요청하는 프로그램은 요청에 대한 응답을 기다릴 필요가 없습니다. 다른 작업을 수행하고 응답이 도착할 때 또는 나중에 응답을 처리할 수 있습니다. 메시징 애플리케이션을 작성할 때 프로그램이 언제 메시지를 송신하는지 또는 대상이 메시지를 받을 수 있는 상태인지에 대해 알거나 염려할 필요가 없습니다. 메시지는 손실되지 않으며 대상이 메시지를 처리할 준비가 될 때까지 큐 관리자가 보유합니다. 메시지는 프로그램이 제거할 때까지 큐에 남습니다. 이는 송신 및 수신 애플리케이션 프로그램이 연결 해제되었으며 송신자가 수신자의 메시지의 수신 확인을 기다릴 필요 없이 처리를 계속할 수 있음을 의미합니다. 대상 애플리케이션은 메시지가 송신될 때 실행 중일 필요조차 없습니다. 대상 애플리케이션은 시작된 후에 메시지를 검색할 수 있습니다.

소형 프로그램

메시지 큐잉을 사용하면 소형, 자급 프로그램 사용의 이점을 활용할 수 있습니다. 작업의 모든 파트를 순차적으로 수행하는 단일, 대형 프로그램 대신에 작업을 여러 개의 작고 독립적인 프로그램으로 분산시킬 수 있습니다. 요청 프로그램은 개별 프로그램 각각에 해당 기능을 수행하라고 요청하는 메시지를 송신합니다. 각 프로그램이 완료되면 그 결과를 하나 이상의 메시지로 되돌려 보냅니다.

메시지 구동 처리

메시지가 큐에 도착할 때 이 메시지는 트리거를 사용하여 애플리케이션을 자동으로 시작할 수 있습니다. 필요에 따라 하나의 메시지 또는 다수의 메시지가 처리될 때 애플리케이션을 중지할 수 있습니다.

이벤트 중심 처리

프로그램은 큐의 상태에 따라 제어할 수 있습니다. 예를 들어, 프로그램이 메시지가 큐에 도착하는 즉시 시작되도록 구성하거나 큐에 특정 우선순위보다 상위에 있는 10개의 메시지 또는 임의의 우선순위를 가지는 10개의 메시지가 있을 때까지 프로그램이 시작되지 않도록 지정할 수 있습니다.

메시지 우선순위

프로그램은 메시지를 큐에 넣을 때 메시지에 우선순위를 지정할 수 있습니다. 이는 큐에서 새 메시지가 추가되는 위치를 판별합니다.

프로그램은 메시지가 큐에 있는 순서 또는 특정 메시지를 가져오는 순서대로 큐에서 메시지를 가져올 수 있습니다. (프로그램이 이전에 송신한 요청에 대한 응답을 검색하고 있으면 프로그램은 특정 메시지를 가져오려고 할 수 있습니다.)

보안

큐 관리자를 사용하는 경우의 애플리케이션의 인증, 큐 관리자에서 큐와 같은 자원을 사용하는 경우의 권한 검사, 네트워크를 통해 이동하고 큐에 상주하는 메시지 데이터의 암호화를 포함하여 보안 기능이 제공됩니다. 보안에 대한 자세한 정보는 보안 개요를 참조하십시오.

데이터 무결성

데이터 무결성은 작업 단위로 제공됩니다. 작업 단위의 시작 및 종료 동기화가 MQGET 또는 MQPUT 각각에 대한 옵션으로 완전하게 지원되어 작업 단위의 결과가 커미트되거나 롤백되도록 허용합니다. 동기점 지원은 애플리케이션에 대해 선택된 동기점 조정의 형식에 따라 IBM MQ에 대해 내부적으로 또는 외부적으로 작동합니다.

복구 지원

가능한 복구의 경우, 모든 지속적 IBM MQ 업데이트가 로그됩니다. 복구가 필요한 경우에 모든 지속 메시지가 복원되고, 모든 인플라이트 트랜잭션이 롤백되며 모든 동기점 커미트 및 백아웃이 제어 중인 동기점 관리자에 의해 일반적인 방식으로 핸들링됩니다. 지속 메시지에 대한 자세한 정보는 메시지 지속성을 참조하십시오.

메시지 큐잉 용어

이 정보는 메시지 큐잉에 사용되는 일부 용어에 대한 자세한 설명을 제공합니다.

여기에는 다음이 포함됩니다.

- 채널
- 클러스터
- IBM MQ MQI client
- z/OS 그룹 내 큐잉
- 메시지
- 메시지 채널 에이전트
- 메시지 디스크립터
- 포인트-투-포인트
- 공개/등록
- 큐
- 큐 관리자
- z/OS 큐 공유 그룹
- z/OS 공유 큐
- 구독
- 주제

채널

채널은 한 큐 관리자에서 다른 큐 관리자로 메시지를 이동시키는 데 사용되며 기본 통신 프로토콜로부터 애플리케이션을 보호합니다. 큐 관리자들은 동일하거나 서로 다른 플랫폼에서 동일한 시스템 또는 다른 시스템에 존재할 수 있습니다. 전송되는 메시지는 여러 위치에서 생성할 수 있습니다.

- 하나의 노드에서 다른 노드로 데이터를 전송하는 사용자 작성 애플리케이션 프로그램.
- PCF 명령 또는 MQAI를 사용하는 사용자 작성 관리 애플리케이션.
- IBM MQ Explorer입니다.
- 도구 이벤트 메시지를 다른 큐 관리자에 전송하는 큐 관리자.
- 원격 관리 명령을 다른 큐 관리자에 전송하는 큐 관리자(예: MQSC 명령 또는 administrative REST API).

채널에 대한 자세한 정보는 29 페이지의 『채널 정의』의 내용을 참조하십시오.

클러스터

클러스터는 논리적으로 연관된 큐 관리자의 네트워크입니다.

클러스터링 없이 분산 큐잉을 사용하는 IBM MQ 네트워크에서 모든 큐 관리자는 독립적입니다. 한 큐 관리자가 다른 큐 관리자에게 메시지를 송신해야 하는 경우 리모트 큐 관리자에 대한 전송 큐 및 채널이 정의되어 있어야 합니다.

클러스터를 사용하는 두 가지 다른 이유는 시스템 관리를 줄이는 것과 사용가능성 및 워크로드 밸런싱을 높이기 위해서 입니다.

가장 작은 클러스터라도 설정이 끝나고 나면 단순화된 시스템 관리에 도움이 됩니다. 클러스터의 일부인 큐 관리자는 보다 소수의 정의만을 필요로 하기 때문에 정의에서 오류가 발생하는 위험을 줄일 수 있습니다.

클러스터링에 대한 자세한 정보는 클러스터를 참조하십시오.

IBM MQ MQI client

IBM MQ MQI 클라이언트는 IBM MQ의 독립적으로 설치 가능한 컴포넌트입니다. MQI 클라이언트를 통해 통신 프로토콜을 사용하는 IBM MQ 애플리케이션을 실행하고 다른 플랫폼에 있는 하나 이상의 MQI(Message Queue Interface)와 상호 작용하며 해당 큐 관리자에 접속할 수 있습니다.

IBM MQ MQI client 컴포넌트의 설치 및 사용 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

- AIX AIX®에 IBM MQ 클라이언트 설치
- Linux Linux®에 IBM MQ 클라이언트 설치
- Windows Windows에 IBM MQ 클라이언트 설치
- IBM i IBM i에 IBM MQ 클라이언트 설치

및 서버 및 클라이언트 간의 연결 구성.

그룹 내 큐잉

z/OS

큐 공유 그룹에 있는 큐 관리자는 일반 채널을 사용하여 통신할 수 있습니다. 또는 사용자가 그룹 내 큐잉(IGQ)이라는 기술을 사용하여 채널 정의 없이 빠른 메시지 전송을 수행할 수 있습니다. 이는 IBM MQ for z/OS에만 적용됩니다.

그룹 내 큐잉에 대한 자세한 정보는 190 페이지의 『그룹 내 큐잉』의 내용을 참조하십시오.

메시지

메시지 큐잉에서 메시지는 하나의 프로그램에서 송신한 데이터의 콜렉션이며 다른 프로그램에도 사용됩니다. IBM MQ 메시지를 참조하십시오.

메시지 유형에 대한 정보는 메시지 유형을 참조하십시오.

메시지 채널 에이전트

채널의 한쪽 끝에는 메시지 채널 에이전트이 있습니다. 메시지 채널 에이전트 쌍(송신 에이전트와 수신 에이전트)은 채널을 구성하고 한 큐 관리자에서 다른 관리자로 메시지를 이동합니다.

메시지 채널 에이전트 사용 방법에 대한 정보는 분산 큐 관리 소개를 참조하십시오.

메시지 디스크립터

IBM MQ 메시지는 제어 정보 및 애플리케이션 데이터로 구성됩니다.

제어 정보는 메시지 디스크립터 구조(MQMD)에 정의되고 다음과 같은 사항을 포함합니다.

- 메시지의 유형
- 메시지의 ID
- 메시지 전달의 우선순위

애플리케이션 데이터의 구조 및 컨텐츠는 IBM MQ가 아닌 참여 프로그램에 의해 판별됩니다.

자세한 정보는 MQMD를 참조하십시오.

포인트-투-포인트 메시징

지점간 메시징에서 각 메시지는 단일 생성 애플리케이션에서 단일 이용 애플리케이션으로 이동합니다. 메시지는 메시지를 큐에 넣는 생성 애플리케이션을 통해 전송되며 이용 애플리케이션은 큐에서 메시지를 받습니다.

발행/구독 메시징

발행/구독 메시징에서 발행 애플리케이션이 발행한 각 메시지의 사본은 모든 관심 애플리케이션으로 전달됩니다. 관심 애플리케이션의 개수는 하나 또는 다수이거나 전혀 없을 수도 있습니다. 발행/구독에서 관심 애플리케이션은 구독자로 알려져 있으며 메시지는 구독으로 식별되는 큐에 삽입됩니다.

자세한 정보는 55 페이지의 『발행/구독 메시징』의 내용을 참조하십시오.

큐

메시지를 전송할 수 있는 이름 지정된 대상입니다. 큐를 서비스하는 프로그램이 메시지를 검색할 때까지 메시지는 큐에 누적됩니다.

자세한 정보는 18 페이지의 『큐』의 내용을 참조하십시오.

큐 관리자

큐 관리자는 애플리케이션에 큐잉 서비스를 제공하는 시스템 프로그램입니다.

큐 관리자는 API(Application Programming Interface)를 제공하여 프로그램이 큐로(부터) 메시지를 넣고 가져올 수 있도록 합니다. 큐 관리자는 관리자가 새 큐를 작성하고, 기존 큐의 특성을 대체하고 큐 관리자의 조작을 제어할 수 있도록 추가 기능을 제공합니다.

IBM MQ 메시지 큐잉 서비스를 시스템에서 사용하려면 큐 관리자가 실행 중이어야 합니다. 단일 시스템에는 둘 이상의 실행 중인 큐 관리자가 있을 수 있습니다(예를 들어, 라이브 시스템에서 테스트 시스템을 분리하려는 경우). 애플리케이션에 대해 각 큐 관리자는 연결 핸들(*Hconn*)로 식별됩니다.

여러 다른 애플리케이션은 큐 관리자의 서비스를 동시에 사용할 수 있으며 이러한 애플리케이션은 전부 비관련일 수 있습니다. 프로그램이 큐 관리자의 서비스를 이용하게 하려면 해당 큐 관리자에 대한 연결을 설정해야 합니다.

애플리케이션이 다른 큐 관리자에 연결된 애플리케이션에 메시지를 송신하게 하려면 큐 관리자는 서로 통신할 수 있어야 합니다. IBM MQ에서는 저장 후 전달 프로토콜을 구현하여 이러한 애플리케이션 간의 안전한 메시지 전달을 보장합니다.

자세한 정보는 25 페이지의 『큐 관리자』의 내용을 참조하십시오.

큐 공유 그룹

동일한 세트의 공유 큐에 액세스할 수 있는 큐 관리자는 큐 공유 그룹(QSG)이라는 그룹을 구성합니다. 공유 큐를 저장하는 커플링 기능(CF)을 사용하여 서로 통신합니다. 이는 IBM MQ for z/OS에만 적용됩니다.

자세한 정보는 149 페이지의 『공유 큐 및 큐 공유 그룹』의 내용을 참조하십시오.

공유 큐

공유 큐는 메시지가 있는 로컬 큐의 유형으로 SYSPLEX에 있는 하나 이상의 큐 관리자가 액세스할 수 있습니다. 이 큐는 동일한 큐 관리자를 사용하여 둘 이상의 애플리케이션에서 공유하는 큐와 같지 않습니다. 이는 IBM MQ for z/OS에만 적용됩니다.

구독

발행/구독 애플리케이션은 특정 주제에 대한 메시지에 관심을 등록할 수 있습니다. 이를 수행하는 애플리케이션은 구독자로 알려져 있으며 기간 구독은 일치하는 메시지가 처리를 위해 큐에 대기되는 방식을 정의합니다.

구독에는 구독자의 ID 및 발행이 배치될 목적지 큐의 ID에 관한 정보가 들어있습니다. 또한 발행이 목적지 큐에 배치되는 방법에 대한 정보도 들어있습니다.

자세한 정보는 57 페이지의 『구독자 및 구독』의 내용을 참조하십시오.

주제

토픽은 발행/구독 메시지에서 발행된 정보의 제목을 설명하는 문자열입니다.

토픽은 발행/구독 시스템에서 메시지를 성공적으로 전달하기 위한 핵심 사항입니다. 각 메시지에 특정 목적지 주소를 포함하는 대신 발행자가 각 메시지에 대한 토픽을 지정합니다. 큐 관리자가 해당 토픽을 구독하는 구독자 목록과 토픽을 일치시킨 다음 각 해당 구독자에게 메시지를 전달합니다.

자세한 정보는 60 페이지의 『토픽』의 내용을 참조하십시오.

메시지 및 큐

메시지 및 큐는 메시지 큐잉 시스템의 기본 컴포넌트입니다.

메시지 개념

메시지는 바이트 문자열이며 문자열을 사용하는 애플리케이션에 의미가 있습니다. 메시지는 하나의 애플리케이션 프로그램에서 다른 애플리케이션 프로그램으로 (또는 동일한 애플리케이션의 다른 파트 사이에서) 정보를 전송하는 데 사용됩니다. 애플리케이션은 동일한 플랫폼 또는 다른 플랫폼에서 실행될 수 있습니다.

IBM MQ 메시지는 다음으로 구성되어 있습니다.

- 애플리케이션 데이터.애플리케이션 데이터의 컨텐츠 및 구조는 이를 사용하는 애플리케이션 프로그램에 의해 정의됩니다.
- 메시지 디스크립터.메시지 디스크립터는 메시지를 식별하고 메시지 유형 및 송신 애플리케이션에 의해 메시지에 지정된 우선순위와 같은 추가 제어 정보를 포함합니다.

  메시지 디스크립터의 형식은 IBM MQ에 의해 정의됩니다. 메시지 디스크립터에 대한 자세한 설명은 MQMD - 메시지 디스크립터를 참조하십시오.
- 메시지 특성. 메시지에 대한 메타데이터. 메시지 특성의 컨텐츠는 이를 사용하는 애플리케이션 프로그램에 의해 정의됩니다. 자세한 정보는 메시지 특성을 참조하십시오.

메시지 길이

메시지 길이를 100MB의 최대 길이로 증가시킬 수 있기는 하지만 (여기에서 1MB는 1 048 576바이트임) 기본 최대 메시지 길이는 4MB입니다. 실제로 메시지 길이는 다음으로 제한될 수 있습니다.

- 수신 큐에 대해 정의된 최대 메시지 길이
- 큐 관리자에 대해 정의된 최대 메시지 길이
- 큐에 의해 정의된 최대 메시지 길이
- 수신 또는 송신 애플리케이션에 의해 정의된 최대 메시지 길이
- 메시지에 사용 가능한 스토리지 용량

애플리케이션이 요구하는 모든 정보를 송신하는 데 여러 메시지가 필요할 수 있습니다.

애플리케이션이 메시지를 송신하고 수신하는 방법

애플리케이션 프로그램은 **MQI 호출**을 사용하여 메시지를 송신하고 수신합니다.

예를 들어, 큐에 메시지를 배치하기 위해 애플리케이션은 다음을 수행합니다.

1. MQI `MQOPEN` 호출을 발행하여 필요한 큐를 엽니다.
2. MQI `MQPUT` 호출을 발행하여 메시지를 큐에 배치합니다.

다른 애플리케이션이 MQI `MQGET` 호출을 발행하여 동일한 큐에서 메시지를 검색할 수 있습니다.

MQI 호출에 대한 자세한 정보는 MQI 호출을 참조하십시오.

큐의 개념

큐는 메시지를 저장하는 데 사용되는 데이터 구조입니다.

각 큐는 큐 관리자가 소유합니다. 큐 관리자는 소유한 큐를 유지보수하고 적절한 큐로 수신한 모든 메시지 저장을 담당합니다. 애플리케이션 프로그램 또는 큐 관리자의 일반적인 조작으로 큐에 메시지를 배치할 수 있습니다.

사전정의된 큐 및 동적 큐

큐는 작성되는 방식에 따라 특성화할 수 있습니다.

- **사전정의된 큐**는 적절한 MQSC 또는 PDF 명령을 사용하는 관리자에 의해 작성됩니다.사전정의된 큐는 영구적이며 이를 사용하는 애플리케이션과는 독립적으로 존재하고 IBM MQ를 다시 시작해도 없어지지 않습니다.

- **동적 큐**는 애플리케이션이 모델 큐의 이름을 지정하는 MQOPEN 요청을 발행할 때 작성됩니다. 작성된 큐는 모델 큐라고 하는 템플리트 큐 정의를 기본으로 합니다.MQSC 명령 DEFINE QMODEL을 사용하여 모델 큐를 작성할 수 있습니다. 모델 큐의 속성(예: 모델 큐에 저장될 수 있는 최대 메시지 수)은 모델 큐에서 작성되는 동적 큐로 상속됩니다.

  모델 큐에는 동적 큐가 영구적 또는 임시적인지 여부를 지정하는 속성이 있습니다. 영구적 큐는 애플리케이션과 큐 관리자가 재시작해도 남아 있지만 임시적 큐는 재시작 시에 손실됩니다.

큐에서 메시지 검색

적합하게 권한 부여된 애플리케이션은 다음 검색 알고리즘에 따라 큐에서 메시지를 검색할 수 있습니다.

- FIFO(First In, First Out)
- 메시지 디스크립터에 정의된 메시지 우선순위입니다. 동일한 우선순위를 가진 메시지는 FIFO(First In, First Out) 기준으로 검색됩니다.
- 특정 메시지에 대한 프로그램 요청

애플리케이션에서의 MQGET 요청은 사용되는 메소드를 판별합니다.

IBM MQ 오브젝트

큐 관리자는 IBM MQ 오브젝트에 대한 특성을 정의합니다. 이 특성 값은 IBM MQ에서 해당 오브젝트를 처리하는 방법에 영향을 줍니다. IBM MQ 명령 및 인터페이스를 사용하여 오브젝트를 작성하고 관리합니다. 애플리케이션에서 MQI(Message Queue Interface)를 사용하여 이들 오브젝트를 제어합니다. 오브젝트는 프로그램에서 취급할 때 MQOD(IBM MQ *object descriptor*)에서 식별합니다.

오브젝트 관리

오브젝트 관리에는 다음 태스크가 포함됩니다.

- 큐 관리자 시작 및 중지
- 애플리케이션용 오브젝트(특히 큐)를 작성
- 오브젝트의 속성 표시 또는 변경
- 오브젝트 삭제
- 다른(원격) 시스템에 있는 큐 관리자에 대한 통신 경로를 작성하기 위해 채널에 대한 작업
- 큐 관리자의 클러스터를 작성하여 전체 관리 프로세스를 단순화하고 워크로드를 밸런스화함

동적 큐의 경우를 제외하고 오브젝트에 대한 작업을 할 수 있으려면 먼저 큐 관리자에 대해 오브젝트를 정의해야 합니다.

IBM MQ 명령을 사용하여 오브젝트 관리 조작을 수행하는 경우 큐 관리자는 조작을 수행하는 데 필요한 권한 레벨을 보유했는지 확인합니다. 마찬가지로, 애플리케이션이 MQOPEN 호출을 사용하여 오브젝트를 열 때 큐 관리자는 애프리케이션이 필요한 권한 레벨을 가지고 있는지 확인한 후 해당 오브젝트에 대한 액세스를 허용합니다. 열려 있는 오브젝트의 이름에 대해 확인합니다.

다음 방법을 사용하여 오브젝트를 정의 및 관리할 수도 있습니다.

- 프로그래밍 가능 명령 형식 참조 및 관리 태스크 자동화에서 설명된 PCF 명령
- MQSC 명령에 설명된 MQSC 명령
- z/OS IBM MQ for z/OS 관리 에 설명된 IBM MQ for z/OS 조작 및 제어판
- Windows Linux IBM MQ Explorer (Windows 및 Intel 시스템 전용 Linux). 자세한 정보는 MQ 탐색기 소개를 참조하십시오.

또한 다음 방법을 사용하여 오브젝트를 관리할 수도 있습니다.

- 키보드로 입력하는 제어 명령. 제어 명령을 사용하여 IBM MQ for Multiplatforms 관리를 참조하십시오.
- 프로그램에서의 MQAI(IBM MQ Administration Interface) 호출. IBM MQ 관리 인터페이스(MQAI)를 참조하십시오.

ALW AIX, Linux, and Windows에서 IBM MQ 명령의 순서의 경우, MQSC 기능을 사용하여 파일에 보유된 일련의 명령을 실행할 수 있습니다. 자세한 정보는 MQSC 명령을 사용하여 IBM MQ 관리를 참조하십시오.

IBM i 정기적으로 사용하는 IBM MQ for IBM i 명령 시퀀스의 경우 CL 프로그램을 작성할 수 있습니다. 자세한 정보는 CL 명령을 사용하여 IBM MQ for IBM i 관리를 참조하십시오.

z/OS 정기적으로 사용하는 IBM MQ for z/OS 명령 순서의 경우 명령을 포함하는 메시지를 작성하고 이러한 메시지를 시스템 명령 입력 큐에 배치하는 관리 프로그램을 작성할 수 있습니다. 큐 관리자는 명령행 또는 조작 및 제어판에서 입력한 명령을 처리하는 방식과 동일한 방식으로 이러한 큐에 대한 메시지를 처리합니다. 이 기술은 IBM MQ 관리 프로그램 작성에 설명되어 있으며, IBM MQ for z/OS에서 제공하는 메일 관리자 샘플 애플리케이션에 예시되어 있습니다. 이 샘플에 대한 설명은 IBM MQ for z/OS용 샘플 프로그램을 참조하십시오.

오브젝트 속성

오브젝트의 특성은 그 속성으로 정의됩니다. 일부는 지정할 수 있고 그 외에는 보기만 가능합니다.

예를 들어, 큐가 수용할 수 있는 최대 메시지 길이는 큐의 **MaxMsgLength** 속성으로 정의할 수 있으며 큐를 작성할 때 이 속성을 지정할 수 있습니다. **DefinitionType** 속성은 큐가 작성된 방법을 지정하며 이 속성은 표시만 할 수 있습니다.

IBM MQ에는 다음 속성을 참조하는 두 가지 방식이 있습니다.

- 해당 PCF 이름 사용(예: **MaxMsgLength**)
- 해당 MQSC 명령 이름 사용(예: MAXMSGL)

큐 공유 그룹

z/OS

동일한 세트의 공유 큐에 액세스할 수 있는 큐 관리자는 큐 공유 그룹(QSG)이라는 그룹을 구성하고 공유 큐를 저장하는 커플링 기능(CF)을 사용하여 서로 통신합니다. QSG는 엄격히 오브젝트가 아닙니다.

공유 큐는 큐 공유 그룹에 있는 하나 이상의 큐 관리자가 액세스할 수 있는 메시지가 있는 로컬 큐의 유형입니다. 이 큐는 동일한 큐 관리자를 사용하여 둘 이상의 애플리케이션에서 공유하는 큐와 같지 않습니다.

큐 공유 그룹은 최대 4자의 이름을 가집니다. 이 이름은 네트워크에서 고유해야 하며 큐 관리자 이름과 달라야 합니다.

중요사항: 공유 큐 및 큐 공유 그룹은 IBM MQ for z/OS에서만 지원됩니다.

자세한 정보는 149 페이지의 『공유 큐 및 큐 공유 그룹』의 내용을 참조하십시오.

시스템 기본 오브젝트

시스템 기본 오브젝트는 큐 관리자가 작성될 때마다 자동으로 작성되는 오브젝트 정의 세트입니다. 설치 시 애플리케이션에서 사용하기 위해 이러한 오브젝트 정의를 복사하고 수정할 수 있습니다.

기본 오브젝트 이름에는 어간(SYSTEM)이 포함됩니다(예를 들어, 기본 로컬 큐는 SYSTEM.DEFAULT.LOCAL.QUEUE이고 기본 수신자 채널은 SYSTEM.DEF.RECEIVER임). 이러한 오브젝트의 이름을 바꿀 수 없으며 이러한 이름의 기본 오브젝트가 필요합니다.

오브젝트를 정의할 때 명시적으로 지정하지 않은 모든 속성은 적절한 기본 오브젝트에서 복사됩니다. 예를 들어, 로컬 큐를 정의하는 경우 지정하지 않은 속성은 기본 큐 SYSTEM.DEFAULT.LOCAL.QUEUE로부터 가져옵니다.

자세한 정보는 시스템 및 기본 오브젝트 를 참조하십시오.

오브젝트 유형

관리 태스크 중 다수가 다양한 IBM MQ 오브젝트 유형의 조작을 포함합니다.

IBM MQ 오브젝트 이름 지정에 대한 정보는 32 페이지의 『IBM MQ 오브젝트 이름 지정』의 내용을 참조하십시오.

큐 관리자에 작성된 기본 오브젝트에 대한 정보는 14 페이지의 『시스템 기본 오브젝트』의 내용을 참조하십시오.

다른 유형의 IBM MQ 오브젝트에 대한 정보는 다음을 참조하십시오.

인증 정보 오브젝트

인증 정보 오브젝트는 인증서 폐기 확인을 수행하는 데 필요한 정의를 제공합니다.

큐 관리자 인증 정보 오브젝트는 TLS(Transport Layer Security)에 대한 IBM MQ 지원의 일부입니다. 폐기된 인증서를 확인하는 데 필요한 정의를 제공합니다. 인증 기관은 더 이상 신뢰할 수 없는 인증서를 폐기합니다.

MQSC 명령 **DEFINE AUTHINFO**를 사용하여 인증 정보 오브젝트를 정의할 수 있습니다. 인증 정보 오브젝트의 속성에 대한 자세한 정보는 **DEFINE AUTHINFO**를 참조하십시오.

인증 정보 오브젝트와 함께 다음 IBM MQ 제어 명령을 사용할 수 있습니다.

- **setmqaut**(권한 부여 또는 취소)
- **dspmqaut**(오브젝트 권한 표시)
- **dmpmqaut**(덤프 권한)
- **rcrmqobj**(오브젝트 재작성)
- **rcdmqimg**(매체 이미지 기록)
- **dspmqfls**(파일 이름 표시)

TLS의 개요 및 인증 정보 오브젝트의 사용은 TLS (Transport Layer Security) 개념 및 IBM MQ 의 TLS 보안 프로토콜의 내용을 참조하십시오.

채널

채널은 한 큐 관리자에서 다른 큐 관리자로 통신 경로를 제공하는 오브젝트입니다.

자세한 정보는 26 페이지의 『채널』의 내용을 참조하십시오.

통신 정보 오브젝트

IBM MQ 멀티캐스트에서는 낮은 지연, 높은 팬아웃, 신뢰할 수 있는 멀티캐스트 메시징을 제공합니다. 통신 정보(COMMINFO) 오브젝트는 멀티캐스트 전송을 사용하는 데 필요합니다.

자세한 정보는 93 페이지의 『IBM MQ 멀티캐스트』의 내용을 참조하십시오.

COMMINFO 오브젝트는 멀티캐스트 전송과 연관된 속성을 포함하는 IBM MQ 오브젝트입니다. 이 속성에 대한 자세한 정보는 DEFINE COMMINFO를 참조하십시오. COMMINFO 오브젝트의 작성에 대한 자세한 정보는 멀티캐스트 시작하기를 참조하십시오.

리스너

리스너는 다른 큐 관리자 또는 클라이언트에서 네트워크 요청을 승인하는 프로세스이며 연관된 채널을 시작합니다.

리스너 프로세스 는 **runmqlsr** 제어 명령을 사용하여 시작할 수 있습니다.

리스너 오브젝트는 IBM MQ 오브젝트이며, 이를 통해 큐 관리자의 범위 내에서 리스너 프로세스의 시작 및 중단을 관리할 수 있습니다. 리스너 오브젝트의 속성을 정의하여 다음을 수행하십시오.

- 리스너 프로세스를 구성합니다.
- 큐 관리자가 시작 및 중지될 때 자동으로 리스너 프로세스를 시작 및 중지할지 여부를 지정합니다.

중요사항: z/OS 리스너 오브젝트는 IBM MQ for z/OS에서 지원되지 않습니다. 채널 시작기를 사용하여 IBM MQ for z/OS가 청취를 구현하는 방법에 대한 자세한 정보는 145 페이지의 『z/OS의 채널 시작기』의 내용을 참조하십시오.

이름 목록

이름 목록은 클러스터 이름, 큐 이름 또는 인증 정보 오브젝트 이름 목록이 포함된 IBM MQ 오브젝트입니다. 클러스터에서 이름 목록은 큐 관리자가 저장소에 보유하는 클러스터의 목록을 식별하는 데 사용될 수 있습니다.

이름 목록은 다른 IBM MQ 오브젝트 목록이 포함된 IBM MQ 오브젝트입니다. 일반적으로 이름 목록은 큐 그룹을 식별하는 데 사용되는 트리거 모니터와 같은 애플리케이션에 의해 사용됩니다. 이름 목록 사용의 장점은 애플리케이션과 독립적으로 유지보수된다는 점이며 이름 목록을 사용하는 애플리케이션을 중지하지 않고도 업데이트할 수 있다는 것입니다. 또한 하나의 애플리케이션이 실패하는 경우 이름 목록은 영향을 받지 않으며 다른 애플리케이션은 계속해서 이름 목록을 사용할 수 있습니다.

이름 목록은 둘 이상의 IBM MQ 오브젝트가 참조하는 클러스터 목록을 유지보수하기 위해 큐 관리자 클러스터와 함께 사용되기도 합니다.

MQSC 명령 DEFINE NAMELIST 및 ALTER NAMELIST 를 사용하여 이름 목록을 정의하고 수정할 수 있습니다.

참고: z/OS 또는 z/OS에서 IBM MQ for z/OS 조작 및 제어판을 사용할 수 있습니다.

프로그램은 MQI를 사용하여 이러한 이름 목록에 포함되는 큐를 알아낼 수 있습니다. 이름 목록의 조직은 애플리케이션 설계자 및 시스템 관리자가 담당합니다.

사용 가능한 이름 목록 속성 목록은 이름 목록의 속성을 참조하십시오.

프로세스 정의

프로세스 정의 오브젝트를 통해 큐 관리자 사용에 대한 애플리케이션의 속성을 정의하여 운영자가 개입할 필요 없이 애플리케이션을 시작할 수 있습니다.

프로세스 정의 오브젝트는 IBM MQ 큐 관리자에서 트리거 이벤트에 대한 응답으로 시작하는 애플리케이션을 정의합니다. 프로세스 정의 속성에는 애플리케이션 ID, 애플리케이션 유형 및 애플리케이션에 특정한 데이터가 포함됩니다. 자세한 정보는 24 페이지의 『IBM MQ에서 특정 목적에 사용되는 큐』에서 이니시에이션 큐를 참조하십시오.

트리거를 사용한 IBM MQ 애플리케이션 시작에 설명된 대로 운영자 개입이 없이도 애플리케이션이 시작될 수 있도록 허용하려면, 애플리케이션의 속성을 큐 관리자에 알려야 합니다. 이러한 속성은 프로세스 정의 오브젝트에 정의되어 있습니다.

오브젝트를 작성할 때 **ProcessName** 속성은 수정됩니다. 그러나 IBM MQ 명령을 사용하여 다른 속성을 변경할 수 있습니다.

참고: z/OS 또는 z/OS에서 IBM MQ for z/OS 조작 및 제어판을 사용할 수 있습니다.

MQINQ - 오브젝트 속성 조회를 사용하여 모든 속성의 값에 대해 조회할 수 있습니다.

사용 가능한 프로세스 정의 속성 목록은 프로세스 정의 속성을 참조하십시오.

큐

IBM MQ 큐는 애플리케이션이 메시지를 넣을 수 있고 애플리케이션이 메시지를 가져올 수 있는 이름 지정된 오브젝트입니다.

자세한 정보는 18 페이지의 『큐』의 내용을 참조하십시오.

큐 관리자

IBM MQ 큐 관리자는 애플리케이션에 큐잉 서비스를 제공하며 그에 포함된 큐를 관리합니다.

자세한 정보는 25 페이지의 『큐 관리자』의 내용을 참조하십시오.

서비스

서비스 오브젝트는 큐 관리자가 시작 또는 중지될 때 실행할 프로그램을 정의하는 방법입니다.

프로그램 유형은 다음 중 하나입니다.

서버

서버는 SERVER로 지정된 매개변수 SERVTYPE을 가지는 서비스 오브젝트입니다. 서버 서비스 오브젝트는 지정된 큐 관리자가 시작될 때 실행되는 프로그램의 정의입니다. 서버 프로세스의 한 인스턴스만을 동시에 실행할 수 있습니다. 실행하는 중에 MQSC 명령인 DISPLAY SVSTATUS를 사용하여 서버 프로세스의 상태를 모니터할 수 있습니다. 일반적으로 서버 서비스 오브젝트는 데드 레터 핸들러 또는 트리거 모니터와 같은 프로그램의 정의이지만 실행될 수 있는 프로그램은 IBM MQ와 함께 제공되는 프로그램으로 제한되지 않습니다. 추가적으로 서버 서비스 오브젝트는 지정된 큐 관리자가 종료되어 프로그램이 종료될 때 실행할 수 있는 명령을 포함하도록 정의할 수 있습니다.

명령

명령은 COMMAND로 지정된 매개변수 SERVTYPE을 가지는 서비스 오브젝트입니다. 명령 서비스 오브젝트는 지정된 큐 관리자가 시작되거나 중지될 때 실행되는 프로그램의 정의입니다. 명령 프로세스의 다중 인스턴스는 동시에 실행될 수 있습니다. 명령 서비스 오브젝트는 프로그램이 일단 실행되면 큐 관리자가 프로그램을 모니터하지 않는다는 점에서 서버 서비스 오브젝트와 다릅니다. 일반적으로 명령 서비스 오브젝트는 단기적이고 하나 또는 둘 이상의 다른 태스크를 시작하는 것과 같은 특정 태스크를 수행하는 프로그램의 정의입니다.

중요사항: z/OS 서비스 오브젝트는 IBM MQ for z/OS에서 지원되지 않습니다.

자세한 정보는 서비스에 대한 작업을 참조하십시오.

스토리지 클래스

z/OS

스토리지 클래스는 하나 이상의 큐를 페이지 세트에 맵핑합니다.

이는 해당 큐에 대한 메시지가 해당 페이지 세트에 저장됨(버퍼링을 조건으로)을 의미합니다.

스토리지 클래스는 IBM MQ for z/OS에서만 지원됩니다.

스토리지 클래스에 대한 자세한 정보는 z/OS에서 IBM MQ 환경 계획을 참조하십시오.

토픽 오브젝트

토픽 오브젝트는 기본이 아닌 특정 속성을 토픽에 지정할 수 있는 IBM MQ 오브젝트입니다.

토픽은 특정 토픽 문자열로 발행 또는 구독하는 애플리케이션에서 정의합니다. 토픽 문자열은 토픽을 슬래시(/)로 구분하여 토픽의 계층을 지정할 수 있습니다. 이러한 계층은 토픽 트리로 시각화할 수 있습니다. 예를 들어, 애플리케이션이 토픽 문자열인 `/Sport/American Football` 및 `/Sport/Soccer`를 발행하면 `American Football` 및 `Soccer`의 두 개 하위를 가진 상위 노드 `Sport`가 있는 토픽 트리가 작성됩니다.

토픽은 토픽 트리에 있는 첫 번째 상위 관리 노드에서 해당 속성을 상속합니다. 특정 토픽 트리에 관리 토픽 노드가 없는 경우, 모든 토픽은 기본 토픽 오브젝트 SYSTEM.BASE.TOPIC에서 해당 속성을 상속합니다.

해당 노드의 토픽 문자열을 토픽 오브젝트의 TOPICSTR 속성에 지정하여 토픽 트리의 모든 노드에서 토픽 오브젝트를 작성할 수 있습니다. 또한 관리 토픽 노드에 다른 속성을 정의할 수도 있습니다. 이 속성에 대한 자세한 정보는 MQSC 명령 또는 PCF 명령을 사용하여 관리 자동화를 참조하십시오. 기본적으로, 각 토픽 오브젝트는 가장 가까운 상위 관리 토픽 노드에서 해당 속성을 상속합니다.

토픽 오브젝트는 애플리케이션 개발자로부터 전체 토픽 트리를 숨기는 데 사용될 수 있습니다. 토픽 `/Sport/American Football`에 대해 이름이 `FOOTBALL.US` 인 토픽 오브젝트가 작성되는 경우 애플리케이션은 동일한 결과의 문자열 `/Sport/American Football` 대신 이름이 `FOOTBALL.US` 인 오브젝트를 발행하거나 구독할 수 있습니다.

토픽 오브젝트의 토픽 문자열에 #, +, / 또는 * 문자를 입력한 경우 이 문자는 문자열 내에서 정상적인 문자로 처리되며 토픽 오브젝트와 연관된 토픽 문자열의 일부로 간주됩니다.

토픽 오브젝트에 대한 자세한 정보는 55 페이지의 『발행/구독 메시징』의 내용을 참조하십시오.

관련 개념

5 페이지의 『메시지 큐잉 소개』
IBM MQ 제품을 사용하면 프로그램이 일관된 API(Application Programming Interface)를 사용하여 다른 컴포넌트(프로세서, 운영 체제, 서브시스템 및 통신 프로토콜)의 네트워크에서 서로 통신할 수 있습니다.

관련 참조
MQSC 명령

큐

IBM MQ 큐 및 큐 속성에 대한 소개입니다.

메시지가 큐에 저장되어 넣기 애플리케이션이 해당 메시지에 대한 응답을 기다리는 경우 이 응답을 기다리는 동안 다른 작업을 할 수 있습니다. 애플리케이션은 메시지 큐 인터페이스 개요에 설명된 MQI(Message Queue Interface)를 사용하여 큐에 액세스합니다.

메시지를 큐에 넣기 전에 먼저 큐가 작성되어 있어야만 합니다. 큐 관리자가 큐를 소유하며 해당 큐 관리자는 다수의 큐를 소유할 수 있습니다. 하지만 각 큐는 큐 관리자 내에서 고유한 이름을 가져야 합니다.

큐는 큐 관리자를 통해 유지보수됩니다. 대부분의 경우 각 큐는 해당 큐 관리자가 실제로 관리하지만 애플리케이션 프로그램에 대해서는 분명하지 않습니다. IBM MQ for z/OS 공유 큐는 큐 공유 그룹에 있는 모든 큐 관리자가 관리할 수 있습니다.

큐를 작성하는 데 IBM MQ 명령(MQSC), PCF 명령 또는 플랫폼별 인터페이스를 사용할 수 있습니다. 예를 들어, IBM MQ for z/OS 조작 및 제어판은 플랫폼에 따라 다릅니다.

애플리케이션에서 임시 작업용 로컬 큐를 동적으로 작성할 수 있습니다. 예를 들어, 응답 대상 큐를 작성할 수 있습니다(애플리케이션이 종료된 후에는 필요하지 않음). 자세한 정보는 22 페이지의 『동적 및 모델 큐』의 내용을 참조하십시오.

큐를 사용하기 전에 큐를 열어 큐에 대해 수행하려는 작업을 지정해야 합니다. 예를 들어, 다음의 작업을 위해 큐를 열 수 있습니다.

- 메시지 찾아보기 전용(메시지 검색은 아님)
- 메시지 검색(및 다른 프로그램과 액세스 공유 또는 독점 액세스 사용)
- 큐에 메시지 넣기
- 큐의 속성 조회
- 큐의 속성 설정

큐를 열 때 지정할 수 있는 전체 옵션 목록은 MQOPEN - 오브젝트 열기를 참조하십시오.

큐의 속성

큐의 일부 속성은 큐가 정의될 때 지정되며 이후에 변경할 수 없습니다(예: 큐의 유형). 다른 큐 속성은 변경할 수 있는 속성으로 그룹화할 수 있습니다.

- 큐 처리 중에 큐 관리자에 의해 지정(예: 큐의 현재 용량)
- 명령에 의해서만 지정(예: 큐의 텍스트 설명)
- MQSET 호출을 사용하는 애플리케이션에 의해 지정(예: Put 조작이 큐에서 허용되는지 여부)

MQING 호출을 사용하여 모든 속성의 값을 찾을 수 있습니다.

둘 이상의 큐 유형에 공용인 속성은 다음과 같습니다.

QName
큐의 이름

QType
큐의 유형

QDesc
큐에 대한 텍스트 설명

InhibitGet
프로그램이 큐에서 메시지를 가져오도록 허용하는지 여부. 하지만, 리모트 큐에서 메시지를 가져올 수는 없습니다.

InhibitPut
프로그램이 큐에 메시지를 넣도록 허용하는지 여부

DefPriority
큐에 넣은 메시지의 기본 우선순위

DefPersistence
큐에 넣은 메시지의 기본 지속성

범위
이 큐에 대한 항목이 이름 서비스에 존재하는지 여부 제어

z/OS **Scope** 속성은 z/OS에서 지원되지 않습니다.

이 속성에 대한 자세한 설명은 큐의 속성을 참조하십시오.

큐 정의 방법

MQSC DEFINE 명령 또는 PCF 큐 작성 명령을 사용하여 IBM MQ 에 큐를 정의할 수 있습니다. 이 명령은 큐의 유형 및 해당 속성을 지정합니다. 예를 들어, 로컬 큐 오브젝트에는 애플리케이션이 MQI 호출에서 해당 큐를 참조할 때 발생하는 일을 지정하는 속성이 있습니다. 속성의 예는 다음과 같습니다.

- 애플리케이션이 큐에서 메시지를 검색할 수 있는지의 여부(GET 사용)
- 애플리케이션이 메시지를 큐에 넣을 수 있는지의 여부(PUT 사용)
- 큐에 대한 액세스가 하나의 애플리케이션에만 독점적인지 또는 애플리케이션 사이에서 공유되는지 여부
- 큐에 동시에 저장될 수 있는 최대 메시지 수 (최대 큐 용량)
- 큐에 넣을 수 있는 메시지의 최대 길이

큐를 정의하는 데 사용할 수 있는 다양한 플랫폼별 인터페이스가 있습니다.

관련 개념
51 페이지의 『클러스터 큐』
클러스터 큐는 클러스터 큐 관리자에 의해 호스팅되며 클러스터의 다른 큐 관리자가 사용할 수 있는 큐입니다.

44 페이지의 『데드-레터 큐』
데드-레터 큐(또는 미전달 메시지)는 올바른 목적지로 라우트할 수 없는 경우 메시지를 송신하는 큐입니다. 일반적으로, 각 큐 관리자에는 데드-레터 큐가 있습니다.

PCF 명령을 사용하여 관리 자동화
IBM MQ Console에서 큐에 대한 작업

관련 태스크
MQSC 명령을 사용하여 IBM MQ 관리
MQ 탐색기로 큐 관리자 및 오브젝트 작성 및 구성
IBM i CL 명령을 사용하여 IBM MQ for IBM i 관리
z/OS IBM MQ for z/OS 에서 MQSC및 PCF 명령을 실행할 수 있는 소스

관련 참조
52 페이지의 『공유 큐와 클러스터 큐의 비교』
이 정보는 공유 큐와 클러스터 큐를 비교하고 사용자의 시스템에 더 적합한 것을 결정하도록 돕기 위해 설계되었습니다.

관련 정보
149 페이지의 『공유 큐의 개념』

로컬 큐

전송, 시작, 데드 레터, 명령, 기본값, 채널 및 이벤트 큐는 로컬 큐의 유형입니다.

프로그램이 연결된 큐 관리자가 큐를 소유하고 있는 경우 큐는 프로그램에 로컬로 표시됩니다. 메시지를 로컬 큐에서 가져오거나 로컬 큐에 넣을 수 있습니다.

큐 정의 오브젝트는 큐에 넣은 실제 메시지뿐만 아니라 큐 정의 정보도 포함하고 있습니다.

각 큐 관리자는 특수 용도로 사용하는 몇몇 로컬 큐를 가질 수 있습니다.

전송 큐

애플리케이션이 리모트 큐로 메시지를 송신할 때 로컬 큐 관리자는 전송 큐라는 특수 로컬 큐에 메시지를 저장합니다. 애플리케이션은 전송 큐에 직접 또는 리모트 큐 정의를 통해 간접적으로 메시지를 넣습니다.

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. 리모트 큐 관리자와 동일한 이름을 가지는 전송 큐. 이 값은 리모트 큐 로컬 정의의 XMITQ의 기본값입니다.
3. 로컬 큐 관리자의 DEFXMITQ 속성에 이름 지정된 전송 큐.

메시지 채널 에이전트는 전송 큐와 연관된 채널 프로그램으로서 메시지를 다음 목적지로 전달합니다. 다음 목적지란 메시지 채널이 연결된 큐 관리자를 말합니다. 메시지의 마지막 목적지와 큐 관리자는 다를 수 있습니다. 메시지가 다음 목적지로 전달되면, 전송 큐에서 삭제됩니다. 메시지는 최종 목적지에 도달하는 과정에서 많은 큐 관리자를 통과해야 할 수 있습니다. 라우트에 따라 각각 다음 목적지로 전송하기 위해 대기 중인 메시지를 보유하는 각 큐 관리자에 전송 큐를 정의해야 합니다. 일반적인 전송 큐는 메시지의 최종 목적지가 서로 달라도 다음 목적지에 대한 메시지를 보유합니다. 클러스터 전송 큐는 여러 목적지에 대한 메시지를 보유합니다. 각 메시지의 `correlID`는 다음 목적지로 전송할 메시지가 있는 채널을 식별합니다.

큐 관리자에 여러 전송 큐를 정의할 수 있습니다. 다른 클래스의 서비스에 사용 중인 각 전송 큐가 있는 동일한 목적지에 대해 여러 전송 큐를 정의할 수도 있습니다. 예를 들어, 동일한 목적지로 가는 소형 메시지와 대형 메시지에 각기 다른 전송 큐를 작성할 수 있습니다. 서로 다른 메시지 채널을 사용하여 메시지를 전송하면 대형 메시지로 인한 소형 메시지 지연이 발생하지 않습니다. 클러스터 토픽 또는 클러스터 큐에 대한 모든 메시지는 기본적으로 단일 클러스터 전송 큐 `SYSTEM.CLUSTER.TRANSMIT.QUEUE`에 위치합니다. 옵션으로서 기본값을 변경할 수 있으며, 서로 다른 클러스터 큐 관리자로 이동하는 메시지 트래픽을 서로 다른 클러스터 전송 큐로 분리할 수 있습니다. `DEFCLXQ`를 `CHANNEL`로 큐 관리자 속성을 설정하면 각 클러스터 송신자 채널은 별도의 클러스터 전송 큐를 작성합니다. 대안으로서, 사용할 클러스터-송신자 채널에 대해 클러스터 전송 큐를 수동으로 정의할 수 있습니다.

전송 큐는 메시지 채널 에이전트를 트리거하여 메시지를 전방으로 보낼 수 있습니다. 트리거를 사용한 IBM MQ 애플리케이션 시작을 참조하십시오.

z/OS IBM MQ for z/OS에서 그룹 내 큐잉을 사용하는 경우, 그룹 내 큐잉 에이전트가 전송 큐를 제공합니다. 공유 전송 큐는 IBM MQ for z/OS에서 그룹 내 큐잉을 사용할 때 사용됩니다.

이니시에이션 큐

이니시에이션 큐는 애플리케이션 큐에서 트리거 이벤트가 발생할 때 큐 관리자가 트리거 메시지를 넣는 로컬 큐입니다.

트리거 이벤트는 프로그램이 큐 처리를 시작하도록 하는 이벤트입니다. 예를 들어 이벤트는 10개가 넘는 수신 메시지가 될 수 있습니다. 트리거의 작동 방법에 대한 자세한 정보는 트리거를 사용한 IBM MQ 애플리케이션 시작을 참조하십시오.

데드-레터(미전달 메시지) 큐

데드-레터(미전달 메시지) 큐는 큐 관리자가 전달할 수 없는 메시지를 넣는 로컬 큐입니다.

큐 관리자가 메시지를 데드-레터 큐에 넣을 때 메시지에 헤더를 추가합니다. 헤더 정보에는 큐 관리자가 데드-레터 큐에 메시지를 넣는 이유가 포함됩니다. 또한 원래 메시지의 목적지, 큐 관리자가 데드-레터 큐에 메시지를 넣는 날짜 및 시간도 포함됩니다.

애플리케이션은 전달할 수 없는 메시지에 대한 큐를 사용할 수도 있습니다. 자세한 정보는 데드-레터(미전달 메시지) 큐 사용을 참조하십시오.

시스템 명령 큐

시스템 명령 큐는 적절하게 권한 부여된 애플리케이션이 IBM MQ 명령을 송신할 수 있는 큐입니다. 이러한 큐는 플랫폼의 지원에 따라 PCF, MQSC 및 CL 명령을 수신하며 이러한 명령에 대해 조치하는 큐 관리자에 대해 준비된 상태입니다.

z/OS IBM MQ for z/OS 에서는 큐를 SYSTEM.COMMAND.INPUT 라고 합니다. 다른 플랫폼에서는 SYSTEM.ADMIN.COMMAND.QUEUE라고 합니다. 승인된 명령은 플랫폼에 따라 다릅니다. 자세한 내용은 프로그래밍 가능한 명령 형식 참조의 내용을 참조하십시오.

시스템 기본 큐

시스템 기본 큐에는 시스템에 대한 큐의 초기 정의가 포함됩니다. 큐 정의를 작성할 때, 큐 관리자는 적절한 시스템 기본 큐에서 정의를 복사합니다. 큐 정의 작성은 동적 큐 작성과 다릅니다. 동적 큐 정의는 동적 큐 템플리트로 선택한 모델 큐를 기준으로 합니다.

이벤트 큐
이벤트 큐는 이벤트 메시지를 보유합니다. 이러한 메시지는 큐 관리자 또는 채널이 보고합니다.

리모트 큐

프로그램에 있어서 큐가 다른 큐 관리자에 의해 소유되는 경우 큐는 프로그램이 연결되는 큐 관리자에 대해 리모트입니다.

통신 링크가 설정된 위치에서 프로그램은 리모트 큐로 메시지를 송신할 수 있습니다. 프로그램은 리모트 큐에서 메시지를 가져올 수 없습니다.

리모트 큐를 정의할 때 작성된 큐 정의 오브젝트는 로컬 큐 관리자가 메시지를 송신하려는 큐를 찾는 데 필요한 정보만을 보유합니다. 이 오브젝트를 리모트 큐의 로컬 정의라고 합니다. 리모트 큐의 모든 속성은 리모트 큐를 소유하는 큐 관리자가 보유하며 이는 리모트 큐가 큐 관리자에게는 로컬 큐이기 때문입니다.

리모트 큐를 열 때 이 큐를 식별하려면 다음 중 하나를 지정해야 합니다.

- 리모트 큐를 정의하는 로컬 정의의 이름 애플리케이션의 관점에서 이는 로컬 큐를 여는 것과 같습니다. 애플리케이션은 큐가 로컬 또는 리모트인지를 알 필요가 없습니다.

  IBM i를 제외한 모든 플랫폼에서 리모트 큐의 로컬 정의를 작성하려면 DEFINE QREMOTE 명령을 사용하십시오.

  IBM i IBM i에서 CRTMQMQ 명령을 사용하십시오.
- 해당 리모트 큐 관리자에게 알려진 대로의 리모트 큐 관리자의 이름 및 큐의 이름

리모트 큐의 로컬 정의에는 18 페이지의 『큐의 속성』에 설명된 공용 속성에 추가하여 세 가지 속성이 더 있습니다. 세 가지 속성은 다음과 같습니다.

RemoteQName
큐의 소유 큐 관리자가 알고 있는 이름입니다.

RemoteQMgrName
소유하고 있는 큐 관리자의 이름

XmitQName
메시지를 다른 큐 관리자에게 전달할 때 사용되는 로컬 전송 큐의 이름

이 속성에 대한 자세한 정보는 큐의 속성을 참조하십시오.

리모트 큐의 로컬 정의에 대하여 MQINQ 호출을 사용하는 경우 큐 관리자는 원격 시스템에서 일치하는 로컬 큐의 속성이 아닌 로컬 정의의 속성(리모트 큐 이름, 리모트 큐 관리자 이름 및 전송 큐 이름)만을 리턴합니다.

전송 큐의 내용도 참조하십시오.

알리어스 큐

알리어스 큐는 다른 큐 또는 토픽에 액세스하는 데 사용할 수 있는 IBM MQ 오브젝트입니다. 이는 둘 이상의 프로그램이 다른 이름을 사용하여 액세스하여 동일한 큐에 대해 작업할 수 있음을 의미합니다.

기본 큐라고 하는 알리어스 이름의 해석 결과인 큐 는 플랫폼에서 지원되는 다음과 같은 큐 유형 중 하나가 될 수 있습니다.

- 로컬 큐
- 리모트 큐의 로컬 정의
- z/OS 공유 큐는 IBM MQ for z/OS에서만 사용 가능한 로컬 큐의 유형입니다.
- 사전정의된 큐
- 동적 큐

알리어스 이름은 토픽으로 해석할 수도 있습니다. 현재 애플리케이션이 메시지를 큐에 넣고 있으면 큐 이름을 토픽의 알리어스로 작성하여 토픽으로 발행하도록 애플리케이션을 작성할 수 있습니다. 애플리케이션 코드의 변경은 필요하지 않습니다.

참고: 알리어스는 동일한 큐 관리자의 다른 알리어스로 직접 해석될 수 없습니다.

알리어스 큐 사용에 대한 예는 시스템 관리자가 기본 큐 이름(알리어스가 해석의 대상 큐) 및 다른 큐 이름에 다른 액세스 권한을 부여하는 것입니다. 이는 프로그램 또는 사용자에게 알리어스 큐 사용에 대한 권한은 부여되지만 기본 큐에 대한 권한은 부여될 수 없음을 의미합니다.

그렇지 않은 경우 알리어스 이름에 대해 Put 조작을 금지하지만 기본 큐에 대해서는 Put 조작을 허용하도록 권한 부여를 설정할 수 있습니다.

일부 애플리케이션에서 알리어스 큐 사용은 시스템 관리자가 애플리케이션을 변경할 필요 없이 알리어스 큐 오브젝트의 정의를 쉽게 변경할 수 있음을 의미합니다.

IBM MQ는 프로그램이 알리어스 이름을 사용하려고 할 때 이 이름에 대해 권한 검사를 수행합니다. 권한 검사에서는 알리어스가 해석하는 이름에 액세스하기 위한 권한이 프로그램에 부여되었는지 확인하지 않습니다. 따라서 프로그램은 해석된 큐 이름이 아닌 알리어스 큐 이름에 액세스할 수 있는 권한을 부여받을 수 있습니다.

18 페이지의 『큐』에 설명된 일반 큐 속성에 추가하여 알리어스 큐는 **BaseQName** 속성을 가질 수 있습니다. 알리어스 이름이 해석되는 기본 큐의 이름입니다. 이 속성에 대한 자세한 설명은 BaseQName (MQCHAR48)을 참조하십시오.

알리어스 큐의 *InhibitGet* 및 **InhibitPut** 속성(18 페이지의 『큐』 참조)은 알리어스 이름에 속합니다. 예를 들어, 알리어스 큐 이름 ALIAS1이 기본 큐 이름 BASE로 해석하는 경우 ALIAS1에 대한 금지가 ALIAS1에만 영향을 주며 BASE는 금지되지 않습니다. 하지만 BASE에 대한 금지는 ALIAS1에 영향을 줍니다.

또한 *DefPriority* 및 **DefPersistence** 속성도 알리어스 이름에 속합니다. 예를 들어 동일한 기본 큐의 다른 알리어스에 다른 기본 우선순위를 지정할 수 있습니다. 또한 알리어스를 사용하는 애플리케이션에 대한 변경없이 이러한 우선순위를 변경할 수 있습니다.

동적 및 모델 큐

이 정보는 동적 큐, 임시 및 영구적 동적 큐의 특성, 동적 큐의 사용, 동적 큐를 사용할 때의 고려사항 및 모델 큐에 대한 통찰을 제공합니다.

애플리케이션 프로그램이 MQOPEN 호출을 발행하여 모델 큐를 열 때 큐 관리자는 모델 큐와 같은 속성을 가진 로컬 큐의 인스턴스를 동적으로 작성합니다. 모델 큐의 *DefinitionType* 필드 값에 따라, 큐 관리자는 임시 또는 영구적 동적 큐를 작성합니다(동적 큐 작성 참조).

임시 동적 큐의 특성

임시 동적 큐에는 다음 특성이 있습니다.

- z/OS 이는 공유 큐일 수 없으며 큐 공유 그룹의 큐 관리자에서 액세스할 수 없습니다.

  참고로 큐 공유 그룹은 IBM MQ for z/OS에서만 사용 가능합니다.
- 비지속 메시지만 보유합니다.
- 복구 가능하지 않습니다.
- 큐 관리자가 시작될 때 삭제됩니다.
- 큐를 작성한 MQOPEN 호출을 발행한 애플리케이션이 큐를 닫거나 종료할 때 삭제됩니다.
  - 큐에 커미트된 메시지가 있는 경우 삭제됩니다.

- 이 때 큐에 대해 미해결된 커미트되지 않은 MQGET, MQPUT 또는 MQPUT1 호출이 있을 경우 이러한 호출이 커미트된 후에 큐는 논리적으로 삭제된 것으로 표시되며 닫기 처리의 일부로 또는 애플리케이션이 종료될 때에만 실제로 삭제됩니다.
- 이때 큐가 작성 또는 다른 애플리케이션에 의해 사용 중이면 큐가 논리적으로 삭제된 것으로 표시되며 큐를 사용하여 마지막 애플리케이션을 종료할 때에만 큐가 실제로 삭제됩니다.
- 논리적으로 삭제된 큐(큐를 닫은 경우는 제외)에 대한 액세스 시도가 이유 코드 MQRC_Q_DELETED로 실패했습니다.
- 큐를 작성한 해당 MQOPEN 호출에 대해 MQCO_NONE, MQCO_DELETE 및 MQCO_DELETE_PURGE가 MQCLOSE 호출에 지정되었을 때 이 호출 모두 MQCO_NONE으로 처리됩니다.

영구적 동적 큐의 특성

영구적 동적 큐에는 다음의 특성이 있습니다.

- 지속적 또는 비지속 메시지를 보유합니다.
- 시스템 실패 이벤트에서 복구 가능합니다.
- 애플리케이션(큐를 작성한 MQOPEN 호출 발행 애플리케이션일 필요는 없음)이 MQCO_DELETE 또는 MQCO_DELETE_PURGE 옵션을 사용하여 큐를 성공적으로 닫을 때 삭제됩니다.
  - 큐에 여전히 메시지(커미트된 또는 커미트되지 않은 메시지)가 있는 경우 MQCO_DELETE 옵션을 사용한 닫기 요청이 실패합니다. 큐에 커미트된 메시지(닫기의 일부로 삭제되는 메시지)가 있는 경우에도 MQCO_DELETE_PURGE 옵션을 사용한 닫기 요청은 성공하지만 큐에 대해 미해결된 커미트되지 않은 MQGET, MQPUT 또는 MQPUT1 호출이 있을 경우에는 실패합니다.
  - 삭제 요청이 성공했지만 큐가 사용 중(작성 또는 다른 애플리케이션에 의해)이면 이 큐는 논리적으로 삭제된 것으로 표시되며 큐를 사용하여 마지막 애플리케이션이 큐를 닫았을 때에만 실제로 삭제됩니다.
- 닫기 애플리케이션이 큐를 작성한 MQOPEN 호출을 발행하지 않는 한 큐 삭제에 대한 권한이 부여되지 않은 애플리케이션이 큐를 닫는 경우 삭제되지 않습니다. 해당하는 MQOPEN 호출을 유효성 검증하는 데 사용되는 사용자 ID(또는 MQOO_ALTERNATE_USER_AUTHORITY가 지정된 경우 대체 사용자 ID)에 대해 권한 검사를 수행합니다.
- 정상 큐와 같은 방식으로 삭제될 수 있습니다.

동적 큐의 사용

다음에 대해서 동적 큐를 사용할 수 있습니다.

- 애플리케이션이 종료된 후 보유될 큐를 요구하지 않는 애플리케이션입니다.
- 다른 애플리케이션이 처리하는 메시지에 대한 응답을 요구하는 애플리케이션입니다. 이런 애플리케이션은 모델 큐를 열어서 응답 대상 큐를 동적으로 작성할 수 있습니다. 예를 들어, 클라이언트 애플리케이션은 다음을 수행할 수 있습니다.
  1. 동적 큐를 작성하십시오.
  2. 요청 메시지의 메시지 디스크립터 구조의 **ReplyToQ** 필드에 동적 큐의 이름을 제공하십시오.
  3. 서버에 의해 처리되고 있는 큐에 요청을 배치하십시오.

서버는 응답 대상 큐에 응답 메시지를 배치할 수 있습니다. 마지막으로 클라이언트는 응답을 처리하고 삭제 옵션을 사용하여 응답 대상 큐를 닫습니다.

동적 큐 사용 시 고려사항

동적 큐 사용 시 다음 사항을 고려하십시오.

- 클라이언트 서버 모델에서 각 클라이언트는 고유 동적 응답 대상 큐를 작성하고 사용해야 합니다. 동적 응답 대상 큐가 둘 이상의 클라이언트 사이에서 공유되는 경우 응답 대상 큐 삭제가 지연될 수 있습니다. 이는 큐에 대해 미해결된 커미트되지 않은 활동이 있거나 큐가 다른 클라이언트에 의해 사용 중이기 때문입니다. 추가적으로 큐는 논리적으로 삭제된 것으로 표시될 수 있으며 후속 API 요청(MQCLOSE 제외)에 대해 액세스 가능하지 않을 수 있습니다.
- 애플리케이션 환경이 동적 큐가 애플리케이션 사이에서 공유되어야 함을 요구하는 경우 큐에 대한 모든 활동이 커미트될 때에만 큐가 닫히는지(삭제 옵션 사용) 확인하십시오. 이는 마지막 사용자에 의해 수행되어

야 합니다. 이 큐의 삭제가 지연되지 않는지 확인하고 논리적으로 삭제된 것으로 표시되어 큐에 액세스가 불가능한 시간을 최소화합니다.

모델 큐

모델 큐는 사용자가 동적 큐를 작성할 때 사용하는 큐 정의의 템플리트입니다.

큐 속성의 템플리트로 사용하려는 모델 큐에 이름을 지정하여 IBM MQ 프로그램에서 동적으로 로컬 큐를 작성할 수 있습니다. 이때 새 큐의 일부 속성을 변경할 수 있습니다. 그러나 **DefinitionType**은 변경할 수 없습니다. 예를 들어 영구적 큐가 필요한 경우 정의 유형이 영구적으로 설정된 모델 큐를 선택하십시오. 일부 대화식 애플리케이션은 응답을 처리한 후 이러한 큐를 유지보수할 필요가 없기 때문에 조회에 대한 응답을 보유하는 데 동적 큐를 사용할 수 있습니다.

모델 큐의 이름을 MQOPEN 호출의 오브젝트 디스크립터(MQOD)에 지정합니다. 모델 큐의 속성을 사용하여 큐 관리자는 로컬 큐를 동적으로 작성합니다.

동적 큐에 대한 전체 이름 또는 이름의 어간(예: ABC)을 지정하고 큐 관리자가 여기에 고유한 파트를 추가하도록 하거나 큐 관리자가 고유한 전체 이름을 지정하도록 할 수 있습니다. 큐 관리자가 이름을 지정하는 경우 이 이름을 MQOD 구조에 삽입합니다.

MQPUT1 호출을 모델 큐에 직접 발행할 수 없지만 MQPUT1을 모델 큐를 열어 작성된 동적 큐에 MQPUT1을 발행할 수 있습니다.

MQSET 및 MQINQ는 모델 큐에서 실행될 수 없습니다. MQOO_INQUIRE 또는 MQOO_SET를 포함하는 모델 큐를 열면 동적으로 작성된 큐에서 후속 MQINQ 및 MQSET 호출이 작성됩니다.

모델 큐의 속성은 로컬 큐의 속성 서브세트입니다. 자세한 설명은 큐의 속성을 참조하십시오.

*IBM MQ*에서 특정 목적에 사용되는 큐

IBM MQ에서는 그 조작과 관련된 특정 목적의 로컬 큐를 사용합니다.

IBM MQ에서 이러한 큐를 사용할 수 있도록 하려면 먼저 큐를 정의해야 합니다.

이니시에이션 큐

이니시에이션 큐는 트리거에서 사용되는 큐입니다. 큐 관리자는 트리거 이벤트가 발생할 때 트리거 메시지를 이니시에이션 큐에 넣습니다. 트리거 이벤트는 큐 관리자가 감지하는 조건의 논리적 조합입니다. 예를 들어, 큐의 메시지 수가 사전정의된 용량에 도달하면 트리거 이벤트가 생성될 수 있습니다. 이러한 이벤트는 큐 관리자가 지정된 이니시에이션 큐에 트리거 메시지를 넣는 원인이 됩니다. 이 트리거 메시지는 이니시에이션 큐를 모니터하는 특수 애플리케이션인 트리거 모니터로 검색할 수 있습니다. 트리거 모니터는 트리거 메시지에 지정된 애플리케이션 프로그램을 시작합니다.

큐 관리자가 트리거를 사용하는 경우 최소 하나 이상의 이니시에이션 큐가 해당 큐 관리자에 대해 정의되어야 합니다. 트리거할 오브젝트 관리, runmqtrm, 트리거를 사용하여 IBM MQ 애플리케이션 시작을 참조하십시오.

전송 큐

전송 큐는 리모트 큐 관리자를 목적지로 하는 메시지를 임시로 저장하는 큐입니다. 로컬 큐 관리자가 메시지를 직접 송신하는 각 리모트 큐 관리자에 대해 최소 하나 이상의 전송 큐를 정의해야 합니다. 이 큐는 원격 관리에서도 사용됩니다. 로컬 큐 관리자에서 원격 관리를 참조하십시오. 분산 큐잉에서 전송 큐의 사용에 대한 정보는 IBM MQ 분산 큐잉 기술을 참조하십시오.

각 큐 관리자에는 기본 전송 큐가 있을 수 있습니다.

- 클러스터에 포함되지 않는 큐 관리자가 메시지를 리모트 큐에 넣는 경우, 기본 조치는 기본 전송 큐를 사용하는 것입니다.
- 목적지 큐 관리자와 이름이 동일한 전송 큐가 있는 경우, 메시지는 해당 전송 큐에 배치됩니다.
- **RQMNAME** 매개변수가 목적지 큐 관리자와 일치하며 **XMITQ** 매개변수가 정의되어 있는 큐 관리자 알리어스 정의가 있는 경우, 메시지는 **XMITQ**로 이름 지정된 전송 큐에 배치됩니다.
- **XMITQ** 매개변수가 없으면 메시지가 해당 메시지에 이름 지정된 로컬 큐에 배치됩니다.
- 원격 큐 관리자가 없고 메시지가 원격 큐 관리자를 대상으로 하는 경우 IBM MQ 는 기본 전송 큐인 **SYSTEM.DEFAULT.XMIT.QUEUE** 을 사용합니다.

클러스터 전송 큐

클러스터 내의 각 큐 관리자에는 클러스터 전송 큐 SYSTEM.CLUSTER.TRANSMIT.QUEUE와 모델 클러스터 전송 큐 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE가 있습니다. 이러한 큐의 정의는 큐 관리자를 정의할 때 기본적으로 작성됩니다. 큐 관리자 속성 **DEFCLXQ**를 CHANNEL로 설정한 경우 생성된 각 클러스터-송신자 채널에 대해 영구적인 동적 클러스터 전송 큐가 자동으로 생성됩니다. 큐를 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*라고 합니다. 또한 클러스터 전송 큐를 수동으로 정의할 수도 있습니다.

클러스터에 포함되는 큐 관리자는 이러한 큐 중 하나에서 동일한 클러스터에 있는 다른 큐 관리자로 메시지를 송신합니다.

이름 해석 중에, 클러스터 전송 큐는 기본 전송 큐보다 우선하고 특정 클러스터 전송 큐는 SYSTEM.CLUSTER.TRANSMIT.QUEUE보다 우선합니다.

데드-레터 큐

데드-레터(미전달 메시지) 큐는 올바른 목적지로 라우트되지 못한 메시지를 저장하는 큐입니다. 예를 들어, 목적지 큐가 가득 찬 경우에는 메시지를 라우트할 수 없습니다. 제공된 데드-레터 큐를 SYSTEM.DEAD.LETTER.QUEUE라고 합니다.

분산 큐잉의 경우 포함된 각 큐 관리자에서 데드-레터 큐를 정의하십시오.

명령 큐

명령 큐인 SYSTEM.ADMIN.COMMAND.QUEUE는 적합하게 권한 부여된 애플리케이션이 처리를 위한 MQSC 명령을 보낼 수 있는 로컬 큐입니다. 이러한 명령은 명령 서버라고 하는 IBM MQ 컴포넌트로 검색됩니다. 명령 서버는 명령을 유효성 검증하고 큐 관리자에 의한 처리를 위해 올바른 명령을 전달하며 적절한 응답 대상 큐로 응답을 리턴합니다.

명령 큐는 해당 큐 관리자가 작성될 때 각 큐 관리자에 대해 자동으로 작성됩니다.

응답 대상 큐

애플리케이션이 요청 메시지를 송신하면 메시지를 수신하는 애플리케이션은 응답 메시지를 송신 애플리케이션으로 돌려 보낼 수 있습니다. 이 메시지를 큐(응답 대상 큐라고 함)에 넣고 이 큐는 일반적으로 송신 애플리케이션에 대한 로컬 큐입니다. 응답 대상 큐의 이름은 메시지 디스크립터의 일부로 송신 애플리케이션에 의해 지정됩니다.

이벤트 큐

도구 이벤트는 MQI 애플리케이션과는 독립적으로 큐 관리자를 모니터하는 데 사용할 수 있습니다.

도구 이벤트가 발생하면 큐 관리자는 이벤트 메시지를 이벤트 큐에 넣습니다. 이 메시지는 모니터링 애플리케이션이 읽을 수 있으며 이벤트가 문제점을 표시할 때 관리자에게 알리거나 일부 보수 조치를 시작할 수 있습니다.

참고: 트리거 이벤트는 도구 이벤트와 다릅니다. 트리거 이벤트는 동일한 조건에 의해 발생하지 않으며 이벤트 메시지를 생성하지 않습니다.

도구 이벤트에 대한 자세한 정보는 도구 이벤트를 참조하십시오.

큐 관리자

애플리케이션에 제공하는 큐 관리자 및 큐잉 서비스 소개입니다.

프로그램이 해당 큐 관리자의 서비스를 사용할 수 있으려면 큐 관리자에 대한 연결이 있어야 합니다. 프로그램은 이 연결을 MQCONN 또는 MQCONNX 호출을 사용하여 명시적으로 수행할 수 있거나 암묵적으로 수행할 수도 있습니다 (이는 프로그램이 실행되는 플랫폼 및 환경에 따라 다름).

IBM MQ 큐 관리자는 다음 조치를 확인합니다.

- 오브젝트 속성이 수신된 명령에 따라 변경됩니다.
- 적절한 조건에 부합되는 경우 트리거 이벤트 또는 도구 이벤트와 같은 특수 이벤트가 생성됩니다.
- MQPUT 호출을 작성하는 애플리케이션이 요청한 대로 올바른 큐에 메시지를 넣습니다. 이러한 조치가 수행될 수 없는 경우에는 애플리케이션에 알려지며 적절한 이유 코드가 제공됩니다.

각 큐는 단일 큐 관리자에 속하며 해당 큐 관리자에 대한 로컬 큐라고 합니다. 애플리케이션이 연결되는 큐 관리자는 해당 애플리케이션에 대한 로컬 큐 관리자라고 합니다. 애플리케이션의 경우 해당 로컬 큐 관리자에 속하는 큐는 로컬 큐입니다.

리모트 큐는 다른 큐 관리자에 속하는 큐입니다. 리모트 큐 관리자는 로컬 큐 관리자를 제외한 다른 모든 큐 관리자입니다. 리모트 큐 관리자는 네트워크 전체의 원격 시스템에 존재하거나 로컬 큐 관리자와 동일한 시스템에 존재할 수 있습니다. IBM MQ는 동일한 시스템에서 다중 큐 관리자를 지원합니다.

큐 관리자 오브젝트는 일부 MQI 호출에서 사용할 수 있습니다. 예를 들어, MQI 호출 `MQINQ`를 사용하여 큐 관리자 오브젝트의 속성에 대해 조회할 수 있습니다.

큐 관리자의 속성

각 큐 관리자에 연관된 속성(또는 특성) 세트는 큐 관리자의 특성을 정의합니다.큐 관리자의 일부 속성은 작성될 때 수정되며 그 외의 속성은 IBM MQ 명령을 사용하여 변경할 수 있습니다. TLS(Transport Layer Security) 암호화에 사용된 속성을 제외한 모든 속성 값에 대해서는 MQINQ 호출을 사용하여 조회할 수 있습니다.

고정된 속성에는 다음이 포함됩니다.

- 큐 관리자의 이름
- 큐 관리자가 실행되는 플랫폼(예: Windows)
- 큐 관리자가 지원하는 시스템 제어 명령의 레벨
- 큐 관리자에 의해 처리되는 메시지에 지정할 수 있는 최고 우선순위
- 프로그램이 IBM MQ 명령을 송신할 수 있는 큐의 이름
- 큐 관리자가 처리할 수 있는 최대 메시지 길이입니다 z/OS (IBM MQ for z/OS에서만 수정됨).
- 프로그램이 메시지를 넣고 가져올 때 큐 관리자가 동기점을 지원하는지 여부

변경 가능한 속성에는 다음이 포함됩니다.

- 큐 관리자에 대한 텍스트 설명
- 큐 관리자가 MQI 호출을 처리할 때 문자열에 사용하는 문자 세트의 ID
- 큐 관리자가 트리거 메시지의 수를 제한하는 데 사용하는 시간 간격
- z/OS 큐 관리자가 만료된 메시지에 대해 큐를 스캔하는 빈도를 판별하는 데 사용하는 시간 간격(IBM MQ for z/OS에만 해당됨)
- 큐 관리자의 데드-레터(미전달 메시지) 큐 이름
- 큐 관리자의 기본 전송 큐 이름
- 하나의 연결에 대한 최대 열린 핸들 수
- 다양한 범주의 이벤트 보고 사용 및 사용 안함
- 작업 단위 내에서 커미트되지 않은 메시지의 최대 수

큐 관리자 및 워크로드 관리

동일한 큐에 대해 둘 이상의 정의를 가지고 있는 큐 관리자의 클러스터를 설정할 수 있습니다(예를 들어, 클러스터에 있는 큐 관리자는 서로의 복제본일 수 있음). 특정 큐에 대한 메시지는 큐의 인스턴스를 호스팅하는 큐 관리자로 핸들링할 수 있습니다. 워크로드 관리 알고리즘은 메시지를 핸들링할 큐 관리자를 결정하여 큐 관리자 간에 워크로드를 분배합니다. 자세한 정보는 클러스터 워크로드 관리 알고리즘을 참조하십시오.

채널

분산 큐 관리자에서 사용되는 채널은 IBM MQ MQI client와 IBM MQ 서버 간 또는 두 IBM MQ 서버 간의 논리 통신 링크입니다.

채널은 한 큐 관리자에서 다른 큐 관리자로 메시지를 이동시키는 데 사용되며 기본 통신 프로토콜로부터 애플리케이션을 보호합니다. 큐 관리자들은 동일하거나 서로 다른 플랫폼에서 동일한 시스템 또는 다른 시스템에 존재할 수 있습니다. 전송되는 메시지는 여러 위치에서 생성할 수 있습니다.

- 하나의 노드에서 다른 노드로 데이터를 전송하는 사용자 작성 애플리케이션 프로그램.
- PCF 명령 또는 MQAI를 사용하는 사용자 작성 관리 애플리케이션.
- IBM MQ Explorer입니다.
- 도구 이벤트 메시지를 다른 큐 관리자에 전송하는 큐 관리자.
- 원격 관리 명령을 다른 큐 관리자에 전송하는 큐 관리자(예: MQSC 명령 또는 administrative REST API).

채널에는 두 가지 정의가 있으며 연결의 양 끝에 하나씩 있습니다. 큐 관리자가 서로 통신하기 위해서는 다른 큐 관리자로 메시지를 보낼 큐 관리자에서 하나의 채널 오브젝트를 정의하고 메시지를 받을 큐 관리자에서 보충 채널 오브젝트를 정의해야 합니다. 동일한 채널 이름이 연결의 양 끝에서 사용되어야 하며 사용된 채널 유형은 호환 가능해야 합니다.

IBM MQ에는 세 개의 채널 범주가 있으며, 이러한 범주 내에는 다른 채널 유형이 포함됩니다.

- 단방향이며, 한 큐 관리자에서 다른 큐 관리자로 메시지를 전송하는 메시지 채널입니다.
- IBM MQ MQI client에서 큐 관리자로 MQI 호출을 전송하고 큐 관리자에서 IBM MQ 클라이언트로 응답하는 양방향 MQI 채널입니다.
- AMQP 채널. 이는 양방향이며, AMQP 클라이언트를 큐 관리자 또는 서버 시스템에 연결합니다. IBM MQ에서는 AMQP 채널을 사용하여 AMQP 애플리케이션 및 큐 관리자 사이에서 AMQP 호출 및 응답을 전송합니다.

메시지 채널

메시지 채널의 용도는 하나의 큐 관리자에서 다른 큐 관리자로 메시지를 전송하는 것입니다. 클라이언트 서버 환경에서는 메시지 채널이 필요하지 않습니다.

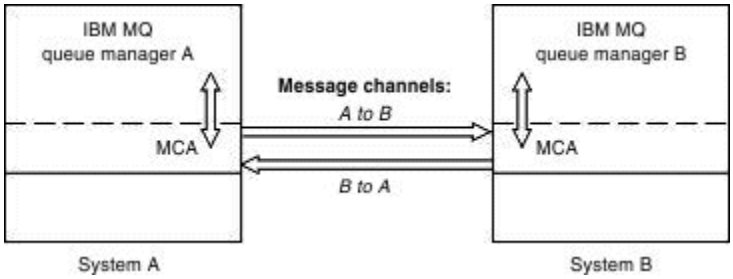


그림 *2.* 두 큐 관리자 간의 메시지 채널

메시지 채널은 단방향 링크입니다. 리모트 큐 관리자가 로컬 큐 관리자가 전송한 메시지에 응답하도록 하려는 경우 응답을 로컬 큐 관리자에 다시 전송하도록 두 번째 채널을 설정해야 합니다.

메시지 채널은 메시지 채널 에이전트(MCA)를 사용하여 두 큐 관리자에 연결됩니다. 채널의 각 끝에는 메시지 채널 에이전트가 있습니다. 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. 파이프라이닝에 대한 자세한 정보는 채널 속성을 참조하십시오.

채널에 대한 자세한 정보는 채널 엑시트 호출 및 데이터 구조 및 41 페이지의 『분산 큐잉 컴포넌트』의 내용을 참조하십시오.

MQI 채널

MQI (Message Queue Interface) 채널은 IBM MQ MQI client 를 서버 시스템의 큐 관리자에 연결하며 IBM MQ MQI client 애플리케이션에서 MQCONN 또는 MQCONNX 호출을 발행할 때 설정됩니다.

이는 양방향 링크이고 MQI 호출 및 응답의 전송에만 사용되며, 여기에는 메시지 데이터가 포함된 MQPUT 호출 및 메시지 데이터의 리턴으로 귀결되는 MQGET 호출이 포함됩니다. 채널 정의를 작성하고 사용하는 다양한 방법이 있습니다(MQI 채널 정의 참조).

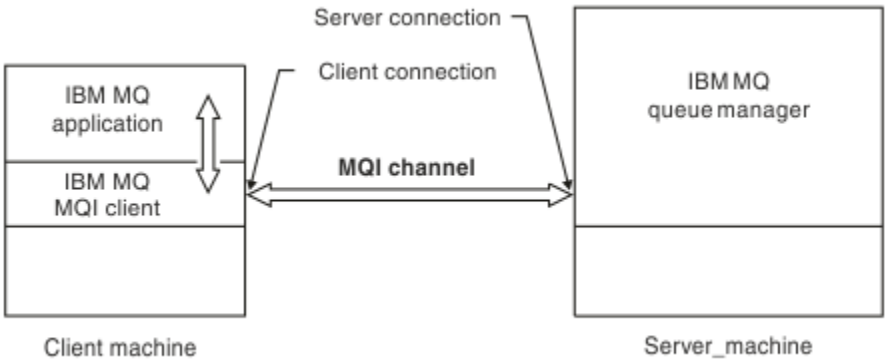


그림 *3. MQI* 채널에서의 클라이언트 연결 및 서버 연결

z/OS MQI 채널은 클라이언트를 단일 큐 관리자에 연결하거나 큐 공유 그룹의 일부인 큐 관리자에 연결하는 데 사용될 수 있습니다(큐 공유 그룹에 클라이언트 연결 참조).

MQI 채널 정의에는 두 가지 채널 유형이 있습니다. 양방향 MQI 채널을 정의합니다.

클라이언트 연결 채널
이 유형은 IBM MQ MQI client용입니다.

서버 연결 채널
이 유형은 IBM MQ MQI client 환경에서 실행 중인 IBM MQ 애플리케이션이 통신하는 큐 관리자를 실행하는 서버를 위한 것입니다.

AMQP 채널

Multi

AMQP 채널의 유형은 한 개뿐입니다.

이 채널을 사용하여 AMQP 메시징 애플리케이션을 큐 관리자와 연결하면 애플리케이션이 IBM MQ 애플리케이션과 메시지를 교환할 수 있습니다. AMQP 채널을 사용할 경우 MQ Light에서 애플리케이션을 개발한 다음 엔터프라이즈 애플리케이션으로 배치할 수 있으므로 IBM MQ에서 제공하는 엔터프라이즈 레벨 기능을 이용할 수 있습니다.

클라이언트 연결 채널

클라이언트 연결 채널은 IBM MQ MQI client에서 큐 관리자로의 통신 경로를 제공하는 오브젝트입니다.

클라이언트 연결 채널은 분산 큐잉에 사용되어 큐 관리자와 클라이언트 사이에 메시지를 이동시킵니다. 이 채널은 근본적인 통신 프로토콜로부터 애플리케이션을 숨깁니다. 클라이언트는 큐 관리자와 동일한 또는 다른 플랫폼에 존재할 수 있습니다.

채널 정의

각 채널 유형에 대한 설명은 29 페이지의 『채널 정의』의 내용을 참조하십시오.

관련 개념
38 페이지의 『분산 큐잉 및 클러스터』
분산 큐잉은 한 큐 관리자에서 다른 큐 관리자로 메시지를 송신하는 것을 의미합니다. 수신 큐 관리자는 동일 시스템 또는 다른 쪽 세계 또는 인접한 곳의 다른 시스템에 있을 수 있습니다. 로컬 큐 관리자와 동일한 플랫폼에서 실행 중이거나 IBM MQ에서 지원되는 플랫폼 중 하나에서 실행 중일 수 있습니다. 분산 큐잉 환경에서 모든 연결을 수동으로 정의하거나 클러스터를 작성하고 IBM MQ에서 자동으로 연결 세부사항 대부분을 정의하도록 설정할 수 있습니다.

MQI(Message Queue Interface) 개요
관련 태스크
원격 IBM MQ 오브젝트 관리

MQI 채널 중지
서버와 클라이언트 간의 연결 구성

관련 참조
채널 엑시트 호출 및 데이터 구조
31 페이지의 『통신』
IBM MQ MQI clients는 MQI 채널을 사용하여 서버와 통신합니다.

채널 정의

IBM MQ에서 사용하는 다양한 유형의 메시지 채널 및 MQI 채널을 설명하는 표입니다.

메시지 채널을 참조할 때 채널은 종종 채널 정의와 동의어로 사용됩니다. 일반적으로 문맥 상에서 양측이 있는 완전한 채널을 말하는지 한 측만 있는 채널 정의를 말하는지 알 수 있습니다.

메시지 채널

메시지 채널은 다음 유형 중 하나일 수 있습니다.

메시지 채널 정의 유형	설명
송신자	송신자 채널은 큐 관리자가 다른 큐 관리자에 메시지를 송신하는 데 사용하는 메시지 채널입니다. 이 송신자 채널을 사용하여 메시지를 송신하려면 다른 큐 관리자에 이 송신자 채널과 동일한 이름의 수신자 채널도 작성해야 합니다. "콜백" 메커니즘을 구현 중인 경우 또한 송신자 채널을 요청자 채널과 함께 사용할 수도 있습니다.
서버	서버 채널은 큐 관리자가 다른 큐 관리자에 메시지를 송신하는 데 사용하는 메시지 채널입니다. 이 서버 채널을 사용하여 메시지를 송신하려면 다른 큐 관리자에 이 서버 채널과 동일한 이름의 수신자 채널도 작성해야 합니다. 또한 요청자 채널과 함께 서버 채널을 사용할 수도 있습니다. 이 경우에 채널의 다른 쪽의 요청자 채널 정의는 서버 채널 정의를 시작하도록 요청합니다. 서버는 요청자에 메시지를 송신합니다. 서버가 파트너 채널의 연결 이름을 알고 있으면 통신을 시작할 수도 있습니다.
수신자	수신자 채널은 큐 관리자가 다른 큐 관리자로부터 메시지를 수신하는 데 사용하는 메시지 채널입니다. 수신자 채널을 사용하여 메시지를 수신하려면 다른 큐 관리자에 이 수신자 채널과 동일한 이름의 송신자 또는 서버 채널도 작성해야 합니다.
요청자	요청자 채널은 큐 관리자가 다른 큐 관리자로부터 메시지를 수신하는 데 사용하는 메시지 채널입니다. 요청자 채널은 리모트 측에 정의된 파트너 채널을 시작하도록 요청할 수 있습니다. 파트너 채널이 서버 채널인 경우 서버 채널은 시작 요청을 승인하고 서버 채널 정의에서 식별된 전송 큐에서 요청자 채널로 메시지를 전송하기 시작합니다. 파트너 채널이 송신자 채널인 경우 송신자 채널은 시작 요청을 승인하지만 요청자와의 연결을 닫습니다. 그런 다음, 송신자 채널이 시작되고 파트너 요청자 채널과의 세션을 협상하며, 송신자 채널 정의에서 식별된 전송 큐에서 메시지를 전송하기 시작합니다. 후자의 경우 특히 요청자 채널이 송신자 채널에 콜백을 요청하는 콜백 메커니즘에 대해 제공됩니다.

메시지 채널 정의 유형	설명
클러스터 송신자	클러스터 송신자(CLUSSDR) 채널 정의는 클러스터 큐 관리자가 전체 저장소 중 하나에 클러스터 정보를 송신할 수 있는 채널의 송신측을 정의합니다. 클러스터 송신자 채널은 큐 관리자의 상태에 대한 모든 변경사항을 저장소에 알리기 위해 사용됩니다(예: 큐의 추가 또는 제거). 메시지 전송에도 사용됩니다. 전체 저장소 큐 관리자 자체에는 서로 가리키는 클러스터-송신자 채널이 있습니다. 이를 사용하여 서로 클러스터 상태 변경사항을 통신할 수 있습니다. 큐 관리자의 CLUSSDR 채널 정의가 어떠한 전체 저장소를 가리키는지는 별로중요하지 않습니다. 초기 접속이 이루어진 후에는 추가적인 클러스터 큐 관리자 오브젝트가 필요에 따라 자동으로 정의됩니다. 따라서 큐 관리자가 클러스터 정보를 모든 전체 저장소로 송신하고 메시지를 모든 큐 관리자로 송신할 수 있습니다.
클러스터 수신자	클러스터 수신자(CLUSRCVR) 채널 정의는 클러스터 큐 관리자가 클러스터의 기타 큐 관리자로부터 메시지를 수신할 수 있는 채널의 수신측을 정의합니다. 클러스터 수신자 채널은 저장소를 목적지로 하는 클러스터-정보에 대한 정보를 전달할 수도 있습니다. 큐 관리자는 클러스터 수신자 채널을 정의하여 다른 클러스터 큐 관리자에게 자신이 메시지를 수신할 수 있음을 표시합니다. 클러스터 큐 관리자마다 최소 하나의 클러스터-수신자 채널이 필요합니다.

각 채널에 대해 채널의 각 측에 대한 채널 정의가 있도록 양측 모두를 정의해야 합니다. 채널의 양측은 호환 가능한 유형이어야 합니다.

다음과 같은 채널 정의의 결합이 있을 수 있습니다.

- 송신자-수신자
- 서버-수신자
- 요청자-서버
- 요청자-송신자(콜백)
- 클러스터-송신자-클러스터-수신자

메시지 채널 에이전트

작성한 각 채널 정의는 특정 큐 관리자에 속합니다. 큐 관리자는 같거나 다른 유형의 채널을 몇 개 가질 수 있습니다. 각 채널의 끝에는 메시지 채널 에이전트(MCA) 프로그램이 있습니다. 채널의 한쪽 끝에서 호출자 MCA는 트랜스미션 큐에서 메시지를 가져와 채널을 통해 송신합니다. 채널의 다른 쪽 끝에서 응답자 MCA는 메시지를 수신하여 리모트 큐 관리자에 전달합니다.

호출자 MCA는 송신자, 서버 또는 요청자 채널과 연관될 수 있습니다. 응답자 MCA는 메시지 채널의 모든 유형과 연관될 수 있습니다.

IBM MQ는 연결의 양측에서 다음 채널 유형 조합을 지원합니다.

호출자		메시지 플로우의 방향	응답자	
채널 유형	리스너가 필요합니까?		리스너가 필요합니까?	채널 유형
송신자	아니오	호출자에서 응답자로	예	수신자
서버	아니오	호출자에서 응답자로	예	수신자
서버	아니오	호출자에서 응답자로	예	요청자
요청자	아니오	응답자에서 호출자로	예	서버

호출자		메시지 플로우의 방향	응답자	
채널 유형	**리스너가 필요합니까?**		**리스너가 필요합니까?**	**채널 유형**
요청자	예	응답자에서 호출자로	예	송신자

MQI 채널

MQI 채널은 다음 중 하나의 유형일 수 있습니다.

MQI 채널 유형	설명
서버 연결	서버 연결 채널은 IBM MQ 서버에 IBM MQ 클라이언트를 연결하는 데 사용되는 양방향 MQI 채널입니다. 서버 연결 채널은 채널의 서버 측입니다.
클라이언트 연결	클라이언트 연결 채널은 IBM MQ 서버에 IBM MQ 클라이언트를 연결하는 데 사용되는 양방향 MQI 채널입니다. IBM MQ Explorer는 또한 클라이언트 연결을 사용하여 리모트 큐 관리자에 연결합니다. 클라이언트 연결 채널은 채널의 클라이언트 측입니다. 클라이언트 연결 채널을 작성하면 큐 관리자를 호스팅하는 컴퓨터에서 파일이 작성됩니다. IBM MQ 클라이언트 컴퓨터로 클라이언트-연결 파일을 복사해야 합니다.

Multi 다중 스레드 지원 - 파이프라이닝

선택적으로, 메시지 채널 에이전트(MCA)가 다중 스레드를 사용하여 메시지를 전송하도록 허용할 수 있습니다. 파이프라이닝이라고 하는 이 프로세스는 MCA가 보다 적은 대기 상태에서 보다 효율적으로 메시지를 전송하도록 함으로써 채널 성능을 향상시킵니다. 각 MCA는 최대 2개의 스레드로 제한됩니다.

qm.ini 파일에서 *PipeLineLength* 매개변수로 파이프라이닝을 제어합니다. 이 매개변수는 채널 스탠자에 추가됩니다.

참고: 파이프라이닝은 TCP/IP 채널에만 적용됩니다.

파이프라이닝을 사용할 때, 채널 양 측의 큐 관리자는 1보다 큰 *PipeLineLength*를 갖도록 구성되어야 합니다.

채널 엑시트 고려사항

파이프라이닝은 다음 이유로 일부 엑시트 프로그램이 실패하도록 할 수 있습니다.

- 엑시트가 직렬로 호출되지 않았을 수 있습니다.
- 엑시트를 다른 스레드에서 대신 호출했을 수 있습니다.

파이프라이닝을 사용하기 전에 엑시트 프로그램의 디자인을 검사하십시오.

- 엑시트는 실행의 모든 단계에서 다시 진입해야 합니다.
- MQI 호출 사용 시에는 다른 스레드로부터 엑시트가 호출될 때 동일한 MQI 핸들을 사용할 수 없음을 기억하십시오.

큐를 열고 엑시트의 모든 후속 호출에서 MQPUT 호출을 위해 이의 핸들을 사용하는 메시지 엑시트를 고려하십시오. 엑시트가 상이한 스레드에서 호출되므로 이는 파이프라이닝 모드에서 실패합니다. 이러한 실패를 피하려면, 각 스레드마다 큐 핸들을 유지하고 엑시트가 호출될 때마다 스레드 ID를 확인하십시오.

통신

IBM MQ MQI clients는 MQI 채널을 사용하여 서버와 통신합니다.

채널 정의는 IBM MQ MQI client 및 서버의 연결 종료 시에 둘 다 작성되어야 합니다. 채널 정의를 작성하는 방법은 MQI 채널 정의에 설명되어 있습니다.

가능한 전송 프로토콜은 다음 표에 표시됩니다.

표 *1. MQI* 채널용 전송 프로토콜				
클라이언트 플랫폼	**LU 6.2**	**TCP/IP**	**NetBIOS**	**SPX**
IBM i		예		
Linux AIX and Linux 시스템	예[1]	예		
Windows	예	예	예	예

참고:

1. Linux LU6.2는 다음 플랫폼에서 지원되지 않습니다.
   - Linux (POWER 플랫폼)
   - Linux (x86-64 플랫폼)
   - Linux (zSeries s390x 플랫폼)

전송 프로토콜 - IBM MQ MQI client 및 서버 플랫폼의 조합은 이러한 전송 프로토콜을 사용하는 IBM MQ MQI client 및 서버 플랫폼의 가능한 조합을 보여줍니다.

IBM MQ MQI client 의 IBM MQ 애플리케이션은 큐 관리자가 로컬인 경우와 동일한 방식으로 모든 MQI 호출을 사용할 수 있습니다. **MQCONN** 또는 **MQCONNX** 는 연결 핸들을 작성하여 IBM MQ 애플리케이션을 선택된 큐 관리자와 연관시킵니다. 해당 연결 핸들을 사용하는 다른 호출은 연결된 큐 관리자가 처리합니다. IBM MQ MQI client 통신은 연결 독립 및 시간 독립인 큐 관리자 간의 통신과는 반대로 클라이언트와 서버 간의 활성 연결을 요구합니다.

채널 정의를 사용하여 전송 프로토콜이 지정되며 애플리케이션에는 영향을 주지 않습니다. 예를 들어, Windows 애플리케이션은 TCP/IP로 하나의 큐 관리자에 연결되고 NetBIOS로 다른 큐 관리자에 연결할 수 있습니다.

성능 고려사항

사용하는 전송 프로토콜은 IBM MQ 클라이언트 및 서버 시스템에 영향을 미칠 수도 있습니다. 전송 속도가 느린 특정 상황에서는 IBM MQ 채널 압축을 사용할 수 있습니다.

IBM MQ 오브젝트 이름 지정

IBM MQ 오브젝트에 채택된 이름 지정 규칙은 오브젝트에 따라 달라집니다. IBM MQ에 대해 사용하는 시스템 및 사용자 ID의 이름 또한 일부 이름 지정 제한의 대상이 됩니다.

큐 관리자의 각 인스턴스는 그 이름으로 알려집니다. 한 큐 관리자가 제공된 메시지를 전송할 대상 큐 관리자를 분명하게 식별할 수 있도록 이 이름은 상호 연결된 큐 관리자의 네트워크 내에서 고유해야 합니다.

다른 유형 오브젝트의 경우 각 오브젝트는 오브젝트에 연관된 이름을 가지며 해당 이름으로 참조될 수 있습니다. 이러한 이름은 하나의 큐 관리자 및 오브젝트 유형에서 고유해야 합니다. 예를 들어, 동일한 이름을 가진 큐 및 프로세스를 가질 수 있지만 동일한 이름을 가진 두 개의 큐를 가질 수는 없습니다.

IBM MQ에서 이름은 최대 20자까지 가능한 채널의 경우를 제외하고 최대 48자까지 가질 수 있습니다. IBM MQ 오브젝트 이름 지정에 대한 자세한 정보는 33 페이지의 『IBM MQ 오브젝트 이름 지정 규칙』의 내용을 참조하십시오.

IBM MQ에 대해 사용하는 시스템 및 사용자 ID의 이름 또한 다음의 일부 이름 지정 제한의 대상이 됩니다.

- 시스템 이름에 공백이 포함되지 않도록 하십시오. IBM MQ는 공백을 포함하는 시스템 이름을 지원하지 않습니다. 이러한 시스템에서 IBM MQ를 설치할 경우 큐 관리자를 작성할 수 없습니다.
- IBM MQ 권한 부여의 경우, 사용자 ID 및 그룹의 이름은 20자를 초과할 수 없습니다(공백 사용 불가).
- Windows 클라이언트가 @ 문자가 포함된 사용자 ID(예: abc@d)로 실행 중인 경우 IBM MQ for Windows 서버는 IBM MQ MQI client의 연결을 지원하지 않습니다.

관련 개념

36 페이지의 『IBM MQ 파일 이름』

각 IBM MQ 큐 관리자, 큐, 프로세스 정의, 이름 목록, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 인증 정보 오브젝트는 파일별로 표시됩니다. 오브젝트 이름은 꼭 올바른 파일 이름일 필요는 없기 때문에 큐 관리자는 필요한 위치에서 오브젝트 이름을 올바른 파일 이름으로 변환합니다.

관련 참조

33 페이지의 『IBM MQ 오브젝트 이름 지정 규칙』

IBM MQ 오브젝트 이름은 최대 길이를 갖고 있으며 대소문자가 구분됩니다. 모든 문자가 모든 오브젝트 유형에 대해 지원되지는 않으며, 많은 오브젝트가 이름의 고유성에 관한 규칙을 갖고 있습니다.

IBM MQ 오브젝트 이름 지정 규칙

IBM MQ 오브젝트 이름은 최대 길이를 갖고 있으며 대소문자가 구분됩니다. 모든 문자가 모든 오브젝트 유형에 대해 지원되지는 않으며, 많은 오브젝트가 이름의 고유성에 관한 규칙을 갖고 있습니다.

여러 가지 유형의 IBM MQ 오브젝트가 있으며, 각 유형의 오브젝트는 별도의 오브젝트 네임스페이스에 존재하기 때문에 모두가 동일한 이름을 가질 수 있습니다. 예를 들어 로컬 큐와 송신자 채널은 둘 다 동일한 이름을 가질 수 있습니다. 그러나 동일한 네임스페이스에서는 한 오브젝트가 다른 오브젝트와 동일한 이름을 가질 수 없습니다. 예를 들어 로컬 큐는 모델 큐와 동일한 이름을 가질 수 없으며, 송신자 채널은 수신자 채널과 동일한 이름을 가질 수 없습니다.

다음 IBM MQ 오브젝트는 별도의 오브젝트 네임스페이스에 존재합니다.

- 인증 정보
- 채널
- 클라이언트 채널
- 리스너
- 이름 목록
- 프로세스
- 큐
- 서비스
- 스토리지 클래스
- 구독
- 주제

오브젝트 이름의 문자 길이

일반적으로 IBM MQ 오브젝트 이름의 길이는 최대 48자까지 가능합니다. 이 규칙은 다음 오브젝트에 적용됩니다.

- 인증 정보
- 클러스터
- 리스너
- 이름 목록
- 프로세스 정의
- 큐
- 큐 관리자
- 서비스
- 구독
- 주제

제한사항은 다음과 같습니다.

1. z/OS z/OS 시스템에서 큐 관리자는 최대 4자여야 하며 대문자와 숫자 문자만으로 구성되어야 합니다.
2. 채널 오브젝트 이름 및 클라이언트 연결 채널 이름의 최대 길이는 20문자입니다. 채널에 대한 자세한 정보는 채널 정의를 참조하십시오.
3. 토픽 문자열은 최대 10240바이트일 수 있습니다. 모든 IBM MQ 오브젝트 이름은 대소문자가 구분됩니다.
4. 구독 이름은 최대 10240바이트일 수 있으며 공백을 포함할 수 있습니다.
5. 스토리지 클래스 이름의 최대 길이는 8자입니다.
6. CF 구조 이름의 최대 길이는 12자입니다.

오브젝트 이름의 문자

IBM MQ 오브젝트 이름에 대해 올바른 문자는 다음과 같습니다.

문자	제한사항
대문자 A - Z	• 없음
소문자 a - z	• MQSC 스크립트에서 소문자가 포함된 이름은 작은따옴표로 묶어야 합니다. 이렇게 하면 소문자가 대문자로 변경되지 않습니다. • EBCDIC 카타카나를 사용하는 시스템은 오브젝트 이름에서 소문자 a- z 문자를 사용할 수 없습니다. • z/OS z/OS 시스템에서 소문자를 사용할 때 제한이 있을 수 있습니다. 예를 들어 큐 관리자 이름은 소문자를 포함할 수 없습니다. • IBM i IBM i 시스템에서 CL 명령을 사용할 때 소문자가 포함된 이름은 작은따옴표로 묶어야 합니다. 이렇게 하면 소문자가 대문자로 변경되지 않습니다.
숫자 0 - 9	• 없음
마침표(.)	• 없음
밑줄(_)	• Multi 없음 • z/OS 선두 또는 후미 문자 밑줄을 갖는 이름은 IBM MQ for z/OS 조작 및 제어판에 의해 처리될 수 없기 때문에 사용을 피하십시오.
정방향 슬래시(/)	• Windows Windows 시스템에서는 큐 관리자 이름의 첫 번째 문자가 슬래시일 수 없습니다. • IBM i IBM i 시스템에서 CL 명령을 사용할 때 슬래시가 포함된 이름은 작은따옴표로 묶어야 합니다. • z/OS 없음

문자	제한사항
퍼센트 부호(%)	• ALW 없음 • z/OS RACF®를 IBM MQ for z/OS에 대한 외부 보안 관리자로 사용 중인 경우, 오브젝트 이름에서 %를 사용하지 마십시오. 이런 이름은 RACF 일반 프로파일이 사용될 때 보안 검사에 포함되지 않기 때문입니다. • IBM i IBM i 시스템에서 CL 명령을 사용할 때 퍼센트 기호가 포함된 이름은 작은따옴표로 묶어야 합니다.

오브젝트 이름의 문자에 관한 몇 가지 규칙이 있습니다.

1. 선두 문자 또는 임베드된 공백은 허용되지 않습니다.
2. 자국어(NL) 문자는 허용되지 않습니다.
3. 전체 필드 길이보다 작은 모든 이름은 오른쪽에 공백을 채울 수 있습니다. 큐 관리자가 리턴하는 모든 짧은 이름은 항상 오른쪽이 공백으로 채워집니다.

큐 이름

큐 이름은 다음 두 파트를 갖습니다.

- 큐 관리자의 이름
- 해당 큐 관리자에게 알려진 큐의 로컬 이름

큐 이름의 각 파트는 길이가 48자입니다.

로컬 큐를 참조하기 위해(공백 문자로 바꾸거나 선행 널 문자를 사용하여) 큐 관리자의 이름을 생략할 수 있습니다. 그러나 IBM MQ에 의해 프로그램으로 리턴되는 모든 큐 이름에 큐 관리자의 이름이 들어있습니다.

z/OS 큐 공유 그룹의 모든 큐 관리자에게 액세스할 수 있는 공유 큐는 동일한 큐 공유 그룹에 있는 임의의 공유되지 않는 로컬 큐와 동일한 이름을 가질 수 없습니다. 이 제한은 애플리케이션이 로컬 큐를 열려고 할 때 실수로 공유 큐를 열거나 반대 상황이 발생할 가능성을 피하게 합니다. 공유 큐 및 큐 공유 그룹은 IBM MQ for z/OS에서만 사용할 수 있습니다.

리모트 큐를 참조하려면 프로그램이 전체 큐 이름에 큐 관리자의 이름을 포함시켜야 하거나 리모트 큐의 로컬 정의가 있어야 합니다.

애플리케이션이 큐 이름을 사용할 때 해당 이름은 로컬 큐의 이름(또는 로컬 큐에 대한 알리어스) 또는 리모트 큐의 로컬 정의의 이름일 수 있지만, 큐에서 메시지를 가져와야 하는 경우가 아니면(큐가 로컬이어야 함) 애플리케이션이 어느 것이든 알 필요는 없습니다. 애플리케이션이 큐 오브젝트를 열 때 MQOPEN 호출이 이름 해석 기능을 수행하여 후속 조작을 수행할 큐를 판별합니다. 이것의 중요성은 애플리케이션이 큐 관리자 네트워크의 특정 위치에서 정의되는 특정 큐에 대한 내장된 종속성을 갖지 않는다는 점입니다. 그러므로 시스템 관리자가 네트워크에서 큐의 위치를 변경하고 해당 정의를 변경하는 경우, 해당 큐를 사용하는 애플리케이션이 변경될 필요가 없습니다.

예약 오브젝트 이름

SYSTEM. 로 시작하는 오브젝트 이름은 큐 관리자가 정의한 오브젝트에 예약되어 있습니다. **Alter**, **Define**및 **Replace** 명령을 사용하여 설치에 맞게 이러한 오브젝트 정의를 변경할 수 있습니다. IBM MQ에 사용하도록 정의된 이름은 큐 이름에 모두 나열되어 있습니다.

z/OS IBM MQ for z/OS에서 커플링 기능 애플리케이션 구조 이름 CSQSYSAPPL은 예약되어 있습니다.

관련 개념

AIX, Linux, and Windows에서 설치 이름

IBM MQ 파일 이름

각 IBM MQ 큐 관리자, 큐, 프로세스 정의, 이름 목록, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 인증 정보 오브젝트는 파일별로 표시됩니다. 오브젝트 이름은 꼭 올바른 파일 이름일 필요는 없기 때문에 큐 관리자는 필요한 위치에서 오브젝트 이름을 올바른 파일 이름으로 변환합니다.

큐 관리자 디렉토리에 대한 기본 경로는 다음과 같습니다.

- 접두부(IBM MQ 구성 정보에 정의되어 있음)
  - Linux AIX AIX and Linux에서 기본 접두부는 `/var/mqm`입니다. 기본 접두부는 mqs.ini 구성 파일의 `DefaultPrefix` 스탠자에서 구성됩니다.
  - Windows Windows 32비트 시스템에서 기본 접두부는 `C:\Program Files (x86)\IBM\WebSphere MQ`입니다. Windows 64비트 시스템에서 기본 접두부는 `C:\Program Files\IBM\MQ`입니다. 32비트와 64비트 설치 모두 데이터 디렉토리는 `C:\ProgramData\ IBM \ MQ`에 설치됩니다. 기본 접두부는 mqs.ini 구성 파일의 `DefaultPrefix` 스탠자에서 구성됩니다.

  사용 가능한 경우 접두부는 IBM MQ 탐색기의 IBM MQ 특성 페이지를 사용하여 변경할 수 있으며, 그렇지 않은 경우 `mqs.ini` 구성 파일을 수동으로 편집하십시오.
- 큐 관리자 이름이 올바른 디렉토리 이름으로 변환됩니다. 예를 들어, 다음 큐 관리자의 경우

  ```
  queue.manager
  ```

  다음과 같이 표시됩니다.

  ```
  queue!manager
  ```

  이 프로세스를 이름 변환이라고 합니다.

IBM MQ에서 최대 48자를 포함하는 이름을 큐 관리자에 지정할 수 있습니다.

예를 들어, 큐 관리자의 이름을 다음과 같이 지정할 수 있습니다.

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

하지만 각 큐 관리자는 파일로 표시되며 파일 이름의 최대 길이 및 이름에 사용될 수 있는 문자 수에 대한 제한이 있습니다. 결과적으로 오브젝트를 나타내는 파일의 이름이 자동으로 변환되어 파일 시스템의 요구사항을 충족시킵니다.

큐 관리자 이름의 변환을 제어하는 규칙은 다음과 같습니다.

1. 개별 문자 변환:
   - .에서 !
   - /에서 &로
2. 이름이 여전히 올바르지 않은 경우에는 다음을 수행합니다.
   a. 여덟개 문자로 이름을 자릅니다.
   b. 세 숫자 접미어를 첨부합니다.

예를 들어, 기본 접두부 및 큐 관리자의 이름을 `queue.manager`로 가정합니다.

- Windows NTFS 또는 FAT32를 사용하는 Windows에서 큐 관리자 이름은 다음이 됩니다.

  ```
  C:\Program Files\IBM\MQ\qmgrs\queue!manager
  ```

- Windows FAT를 사용하는 Windows에서 큐 관리자 이름은 다음이 됩니다.

  ```
  C:\Program Files\IBM\MQ\qmgrs\queue!ma
  ```

- Linux AIX AIX and Linux에서 큐 관리자 이름은 다음이 됩니다.

```
/var/mqm/qmgrs/queue!manager
```

변환 알고리즘은 또한 대소문자를 구분하지 않는 파일 시스템에서만 달라지는 이름을 구별합니다.

오브젝트 이름 변환

오브젝트 이름이 올바른 파일 시스템 이름일 필요는 없습니다. 오브젝트 이름을 변환해야 합니다. 사용된 메소드는 큐 관리자 이름의 메소드와 다릅니다. 각 시스템에 소수의 큐 관리자 이름만 있더라도 각 큐 관리자에 대해 많은 수의 다른 오브젝트가 있을 수 있기 때문입니다. 큐, 프로세스 정의, 이름 목록, 채널, 클라이언트 연결 채널, 리스너, 서비스 및 인증 정보 오브젝트는 파일 시스템에 표시됩니다.

변환 프로세스가 새 이름을 생성할 때 원래 오브젝트 이름에 대한 단순 관계가 없습니다. **dspmqfls** 명령을 사용하여 실제와 변환된 오브젝트 이름 사이에서 변환할 수 있습니다.

관련 참조

dspmqfls(파일 이름 표시)

관련 정보

mqs.ini 파일의 AllQueueManagers 스탠자

IBM i IBM i의 오브젝트 이름

큐 관리자는 고유한 이름이 있는 연관된 큐 관리자 라이브러리를 가집니다. IBM i IFS(Integrated File System)의 요구사항을 충족하기 위해 큐 관리자 이름 및 오브젝트 이름을 변환해야 할 수도 있습니다.

큐 관리자를 작성할 때, IBM MQ는 큐 관리자 라이브러리를 큐 관리자와 연관시킵니다. 이 큐 관리자 라이브러리는 사용자 정의 큐 관리자 이름을 기반으로 길이가 10자 미만인 고유 이름이 제공됩니다. 큐 관리자 및 큐 관리자 라이브러리는 접두부가 /QIBM/UserData/mqm인 큐 관리자 이름을 기반으로 하는 디렉토리에 놓입니다. 큐 관리자, 큐 관리자 라이브러리 및 디렉토리의 예제는 다음과 같습니다.

큐 관리자 이름	ORANGE
큐 관리자 라이브러리 이름	QMORANGE
디렉토리	/QIBM/UserData/mqm/ORANGE

모든 큐 관리자 이름 및 큐 관리자 라이브러리 이름은 /QIBM/UserData/mqm/mqs.ini 파일의 스탠자에 작성됩니다.

IBM MQ IFS 디렉토리 및 파일

IBM i IFS(Integrated File System)는 데이터를 저장하기 위해 IBM MQ에서 광범위하게 사용됩니다. IFS에 관한 자세한 정보는 *IFS(Integrated File System)* 소개를 참조하십시오.

각 IBM MQ 오브젝트(예: 채널 또는 큐 관리자)는 파일로 표시됩니다. 오브젝트 이름은 꼭 올바른 파일 이름일 필요는 없기 때문에 큐 관리자는 필요한 위치에서 오브젝트 이름을 올바른 파일 이름으로 변환합니다.

큐 관리자 디렉토리 경로의 형식은 다음과 같습니다.

- 큐 관리자 구성 파일 qm.ini에 정의된 접두부. 기본 접두부는 /QIBM/UserData/mqm입니다.
- 리터럴, qmgrs.
- 코드화된 큐 관리자 이름, 올바른 디렉토리 이름으로 변환된 큐 관리자 이름입니다. 예를 들어, 큐 관리자 queue/manager는 queue&manager로 표시됩니다.

이 프로세스를 이름 변환이라고 합니다.

IFS 큐 관리자 이름 변환

IBM MQ에서 최대 48자를 포함하는 이름을 큐 관리자에 지정할 수 있습니다.

예를 들어, 큐 관리자 QUEUE/MANAGER/ACCOUNTING/SERVICES의 이름을 지정할 수 있습니다. 각 큐 관리자에 대해 라이브러리가 작성되는 것과 같은 방식으로, 각 큐 관리자도 파일로 표시됩니다. EBCDIC의 변형 코드 포인트 때문에, 이름에서 사용될 수 있는 문자에 제한사항이 있습니다. 결과적으로, 오브젝트를 표시하는 IFS 파일의 이름은 파일 시스템의 요구사항을 충족하도록 자동으로 변환됩니다.

이름이 queue/manager인 큐 관리자의 예제를 사용하고, / 문자를 &로 변환하고, 기본 접두부를 가정하여 IBM MQ for IBM i의 큐 관리자 이름은 /QIBM/UserData/mqm/qmgrs/queue&manager가 됩니다.

오브젝트 이름 변환

오브젝트 이름은 필수적으로 올바른 파일 시스템 이름이 아니므로 오브젝트 이름을 변환해야 할 수 있습니다. 각 시스템에 대해서는 큐 관리자 이름이 몇 개 없지만 각 큐 관리자에 대해서는 다른 오브젝트가 많을 수 있기 때문에 사용된 메소드는 큐 관리자 이름의 메소드와 다릅니다. 프로세스 정의, 큐 및 이름 목록만이 파일 시스템에 표시됩니다. 채널은 이 고려사항에 영향을 받지 않습니다.

변환 프로세스가 새 이름을 생성할 때 원래 오브젝트 이름에 대한 단순 관계가 없습니다. DSPMQMOBJN 명령을 사용하여 IBM MQ 오브젝트로 변환된 이름을 볼 수 있습니다.

분산 큐잉 및 클러스터

분산 큐잉은 한 큐 관리자에서 다른 큐 관리자로 메시지를 송신하는 것을 의미합니다. 수신 큐 관리자는 동일 시스템 또는 다른 쪽 세계 또는 인접한 곳의 다른 시스템에 있을 수 있습니다. 로컬 큐 관리자와 동일한 플랫폼에서 실행 중이거나 IBM MQ에서 지원되는 플랫폼 중 하나에서 실행 중일 수 있습니다. 분산 큐잉 환경에서 모든 연결을 수동으로 정의하거나 클러스터를 작성하고 IBM MQ에서 자동으로 연결 세부사항 대부분을 정의하도록 설정할 수 있습니다.

분산 큐잉

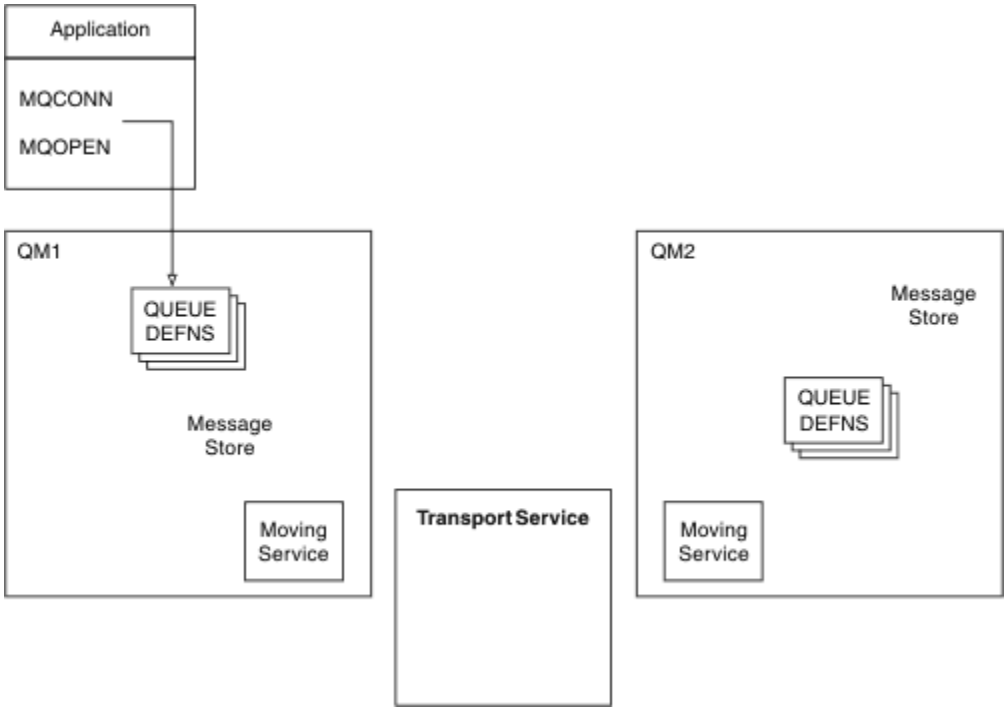


그림 *4.* 분산 큐잉의 컴포넌트 개요

이전 그림에서:

- 애플리케이션은 MQCONN 호출을 사용하여 큐 관리자에 연결합니다. 애플리케이션은 MQOPEN 호출을 사용하여 큐를 열어 메시지를 큐에 넣을 수 있습니다.
- 각 큐 관리자에는 각 큐에 대한 정의가 있습니다. 로컬 큐(이 큐 관리자에 의해 호스트됨)의 정의, 리모트 큐(다른 큐 관리자에 의해 호스트됨)의 정의가 있을 수 있습니다.

- 메시지가 리모트 큐를 대상으로 하면, 리모트 큐 관리자에 메시지를 전달할 준비가 될 때까지 로컬 큐 관리자를 메시지 저장소에서 지속되는 전송 큐에 보유합니다.
- 각 큐 관리자는 큐 관리자가 다른 큐 관리자와 통신하는 데 사용하는 이동 서비스라는 통신 소프트웨어를 포함합니다.
- 전송 서비스는 큐 관리자와 무관하며 다음 중 하나일 수 있습니다(플랫폼에 따라).
  - SNA APPC(Systems Network Architecture Advanced Program-to Program Communication)
  - TCP/IP(Transmission Control Protocol/Internet Protocol)
  - NetBIOS(Network Basic Input/Output System)
  - SPX(Sequenced Packet Exchange)

메시지를 송신해야 하는 컴포넌트

메시지를 리모트 큐 관리자에 전송하려는 경우, 로컬 큐 관리자는 전송 큐 및 채널에 대한 정의가 필요합니다. 채널은 두 큐 관리자 간의 단방향 통신 링크입니다. 리모트 큐 관리자의 임의의 수의 큐를 대상으로 하는 메시지를 전달할 수 있습니다.

각 채널 끝에는 송신 끝 또는 수신 끝과 같이 이를 정의하는 별도의 정의가 있습니다. 단순 채널은 로컬 큐 관리자의 송신자 채널 정의 및 리모트 큐 관리자의 수신자 채널 정의로 구성됩니다. 이 두 정의는 이름이 동일해야 하며 함께 하나의 채널을 구성해야 합니다.

메시지 송신 및 수신을 핸들링하는 소프트웨어는 메시지 채널 에이전트(MCA)라고 합니다. 채널의 각 끝에 메시지 채널 에이전트(MCA)가 있습니다.

각 큐 관리자에 데드-레터 큐(미전달 메시지 큐라고도 하는)가 있어야 합니다. 목적지에 메시지를 전달할 수 없는 경우 이 큐에 메시지를 넣습니다.

다음 그림은 큐 관리자, 전송 큐, 채널 및 MCA 간의 관계를 표시합니다.

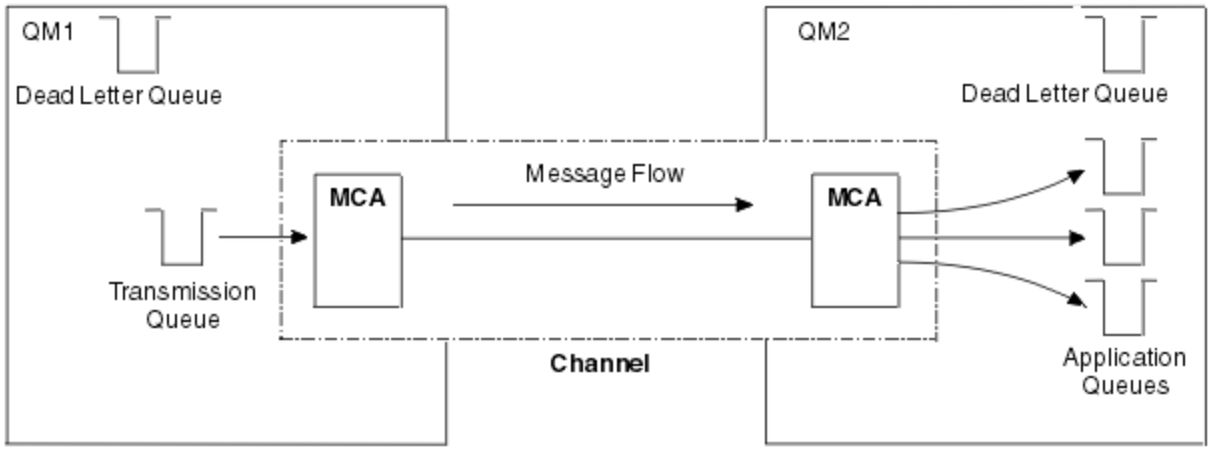


그림 *5.* 메시지 송신

메시지를 리턴해야 하는 컴포넌트

애플리케이션이 메시지를 리모트 큐 관리자에서 리턴해야 하는 경우, 다음 그림에 표시된 대로 큐 관리자 간에 반대 방향으로 실행하도록 다른 채널을 정의해야 합니다.

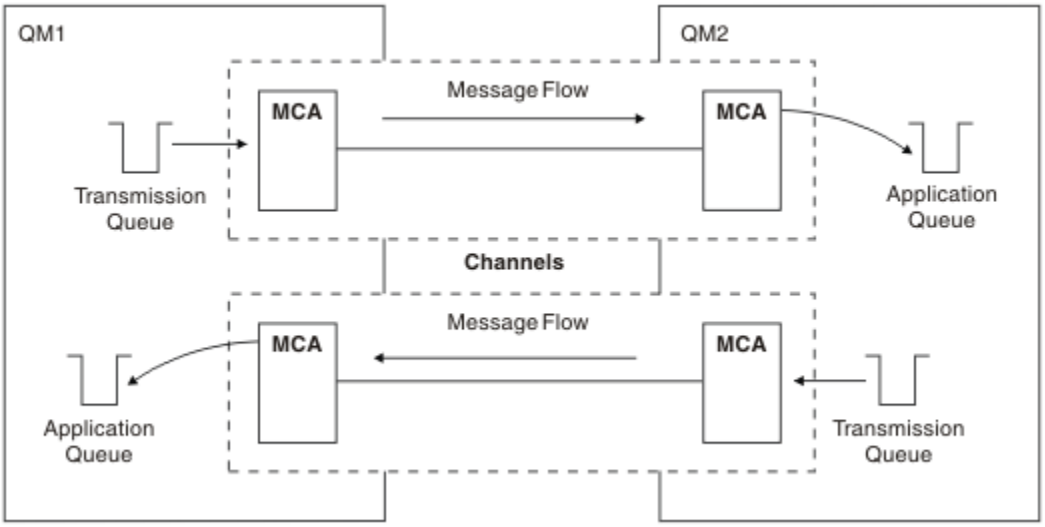


그림 6. 양방향으로 메시지 송신

클러스터

분산 큐잉 환경에서 모든 연결을 수동으로 정의하는 대신 클러스터의 큐 관리자 세트를 그룹화할 수 있습니다. 이를 수행하는 경우, 큐 관리자는 명확한 채널 정의, 리모트 큐 정의 또는 각 목적지에 대한 전송 큐가 필요없이 호스팅하는 큐를 클러스터의 다른 큐 관리자에서 사용할 수 있도록 설정할 수 있습니다. 클러스터의 모든 큐 관리자에는 클러스터의 다른 큐 관리자에 메시지를 전송하는 단일 전송 큐가 있습니다. 각 큐 관리자에 대해 하나의 클러스터-수신자 채널 및 하나의 클러스터-송신자 채널만 정의해야 합니다. 추가 채널은 클러스터가 자동으로 관리합니다.

IBM MQ 클라이언트는 임의의 다른 큐 관리자에게 연결할 수 있는 것처럼 클러스터의 일부인 큐 관리자에 연결할 수 있습니다. 수동으로 구성된 분산 큐잉과 마찬가지로 MQPUT 호출을 사용하여 메시지를 큐 관리자에 있는 큐에 넣습니다. MQGET 호출을 사용하여 로컬 큐에서 메시지를 검색합니다.

클러스터를 지원하는 플랫폼의 큐 관리자는 클러스터의 파트일 필요가 없습니다. 클러스터를 사용하는 대신 분산 큐잉도 수동으로 계속 구성할 수 있습니다.

클러스터 사용의 이점

클러스터링은 다음과 같은 두 가지 주요 이점을 제공합니다.

- 클러스터는 보통 구성할 채널, 전송 큐, 리모트 큐에 대한 많은 오브젝트 정의를 필요로 하는 IBM MQ 네트워크 관리를 단순화합니다. 이 상황은 특히 많은 큐 관리자를 상호 연결해야 하는, 잠재적으로 계속 변화하는 대규모 네트워크에서 적용됩니다. 이 아키텍처는 특히 구성 및 활동적인 유지보수가 어렵습니다.
- 클러스터는 큐 및 클러스터의 큐 관리자에서 메시지 트래픽의 워크로드를 분산하는 데 사용할 수 있습니다. 이러한 분산을 통해 단일 큐의 메시지 워크로드를 여러 큐 관리자에 있는 해당 큐의 동등한 인스턴스에 분산할 수 있습니다. 이러한 워크로드 분산을 사용하여 시스템에서 특히 활성 메시지 플로우의 확장 성능을 향상시키고 시스템 장애에 대해 더 많은 회복 기능을 달성할 수 있습니다. 이러한 환경에서 분산된 큐의 각 인스턴스는 메시지를 처리하는 이용 애플리케이션을 보유합니다. 자세한 정보는 워크로드 관리를 위한 클러스터 사용을 참조하십시오.

메시지가 클러스터에서 라우팅되는 방식

클러스터를 성실한 시스템 관리자가 유지보수하는 큐 관리자의 네트워크로 생각할 수 있습니다. 사용자가 클러스터 큐를 정의할 때마다 시스템 관리자가 자동으로 다른 큐 관리자에서 필요한 대로 대응하는 리모트 큐 정의를 작성합니다.

IBM MQ가 클러스터의 각 큐 관리자에 대한 전송 큐를 제공하기 때문에 사용자가 전송 큐 정의를 작성할 필요가 없습니다. 이 단일 전송 큐를 사용하여 클러스터의 다른 모든 큐 관리자로 메시지를 전달할 수 있습니다. 단일 전송 큐를 사용하도록 제한되지는 않습니다. 큐 관리자는 다중 전송 큐를 사용하여 클러스터의 각 큐 관리자로 전달되는 메시지를 구분할 수 있습니다. 일반적으로 큐 관리자는 단일 클러스터 전송 큐를 사용합니다. 큐 관리자에서

클러스터의 각 큐 관리자마다 각기 다른 클러스터 전송 큐를 사용할 수 있도록 큐 관리자 속성 DEFCLXQ를 변경할 수 있습니다. 또한 클러스터 전송 큐를 수동으로 정의할 수도 있습니다.

클러스터에 조인하는 모든 큐 관리자는 이 방식으로 작업하는 데 동의합니다. 자기 자신 및 자신이 호스팅하는 큐에 대한 정보를 송신하고, 클러스터의 다른 멤버에 관한 정보를 수신합니다.

큐 관리자를 사용할 수 없게 되면 정보를 잃게 되도록 하기 위해 클러스터에서 두 개의 큐 관리자를 전체 저장소의 역할을 하도록 지정합니다. 이러한 큐 관리자는 클러스터의 모든 큐 관리자 및 큐에 대한 전체 정보 세트를 저장합니다. 클러스터의 다른 모든 큐 관리자는 이러한 큐 관리자 및 메시지를 교환하는 큐에 대한 정보만 저장합니다. 이러한 큐 관리자를 부분 저장소라고 합니다. 자세한 정보는 50 페이지의 『클러스터 저장소』의 내용을 참조하십시오.

클러스터의 파트가 되기 위해서는 큐 관리자가 클러스터-송신자 채널과 클러스터-수신자 채널이라는 두 채널을 가져야 합니다.

- 클러스터-송신자 채널은 송신자 채널 같은 통신 채널입니다. 이미 클러스터의 멤버인 전체 저장소에 연결하기 위해 수동으로 큐 관리자에 하나의 클러스터-송신자 채널을 작성해야 합니다.
- 클러스터-수신자 채널은 수신자 채널 같은 통신 채널입니다. 하나의 클러스터-수신자 채널을 수동으로 작성해야 합니다. 클러스터 통신을 수신하기 위해 채널이 큐 관리자에 대한 메커니즘으로 작용합니다.

그러면 이 큐 관리자와 클러스터의 다른 멤버 사이에서 통신에 필요한 다른 모든 채널이 자동으로 작성됩니다.

다음 그림은 CLUSTER라는 클러스터의 컴포넌트를 표시합니다.

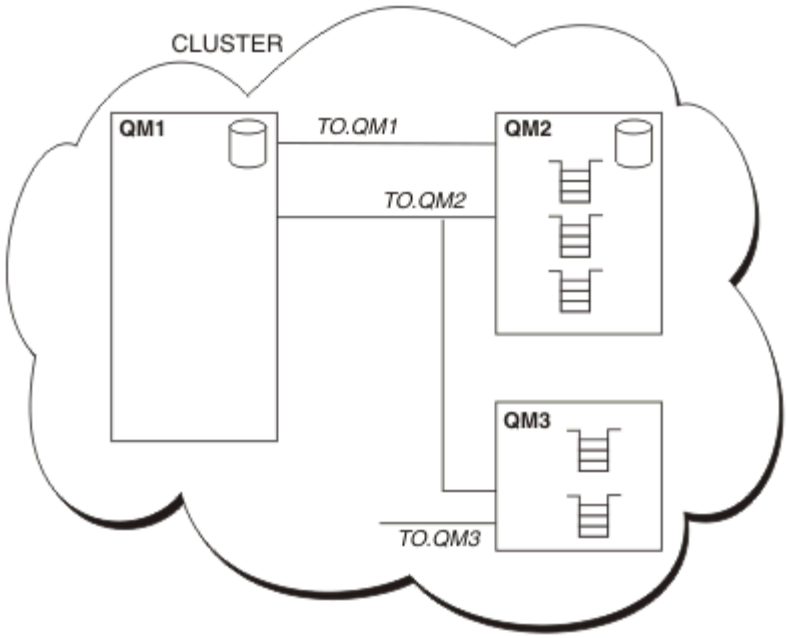


그림 7. 큐 관리자의 클러스터

- CLUSTER에는 세 개의 큐 관리자(QM1, QM2 및 QM3)가 포함되어 있습니다.
- QM1 및 QM2는 클러스터의 큐 및 큐 관리자에 관한 정보의 전체 저장소를 호스트합니다.
- QM2 및 QM3은 일부 클러스터 큐, 즉 클러스터의 다른 큐 관리자에 액세스 가능한 큐를 호스트합니다.
- 각 큐 관리자에는 메시지를 수신할 수 있는 TO.qmgr이라는 클러스터-수신자 채널이 있습니다.
- 각 큐 관리자에는 또한 정보를 저장소 큐 관리자 중의 하나로 송신할 수 있는 클러스터-송신자 채널이 있습니다.
- QM1 및 QM3은 QM2에 있는 저장소에 송신하며 QM2는 QM1에 있는 저장소에 송신합니다.

분산 큐잉 컴포넌트

분산 큐잉의 컴포넌트로는 메시지 채널, 메시지 채널 에이전트, 전송 큐, 채널 시작기 및 리스너, 채널 엑시트 프로그램이 있습니다. 메시지 채널의 각 끝에 대한 정의는 다음 유형 중 하나일 수 있습니다.

메시지 채널은 하나의 큐 관리자에서 다른 큐 관리자로 메시지를 이동시키는 채널입니다. MQI 채널과 메시지 채널을 혼동하지 마십시오. 두 가지 유형의 MQI 채널(서버 연결(SVRCONN) 및 클라이언트 연결(CLNTCONN))이 있습니다. 자세한 정보는 채널을 참조하십시오.

메시지 채널의 각 끝의 정의는 다음 유형 중 하나일 수 있습니다.

- 송신자(SDR)
- 수신자(RCVR)
- 서버(SVR)
- 요청자(RQSTR)
- 클러스터 송신자(CLUSSDR)
- 클러스터 수신자(CLUSRCVR)

메시지 채널은 한 끝에 정의된 다음 유형 중 하나 및 다른 끝의 호환 가능한 유형을 사용하여 정의됩니다. 가능은 결합은 다음과 같습니다.

- 송신자-수신자
- 요청자-서버
- 요청자-송신자(콜백)
- 서버-수신자
- 클러스터 송신자-클러스터 수신자

송신자-수신자 채널의 작성에 대한 자세한 지시사항은 채널 정의에 포함되어 있습니다. 송신자-수신자 채널을 설정하기 위해 필요한 매개변수의 예는 사용자 플랫폼에 적용 가능한 구성 정보 예를 참조하십시오. 모든 유형의 채널을 정의하는 데 필요한 매개변수는 DEFINE CHANNEL을 참조하십시오.

송신자-수신자 채널

한 시스템의 송신자는 채널을 시작하여 메시지를 다른 시스템에 송신할 수 있습니다 송신자는 채널의 다른 끝에 있는 수신자가 시작되기를 요청합니다. 송신자는 메시지를 전송 큐에서 수신자에게 송신합니다. 수신자는 메시지를 목적지 큐에 넣습니다. 42 페이지의 그림 8은 이 내용을 설명합니다.

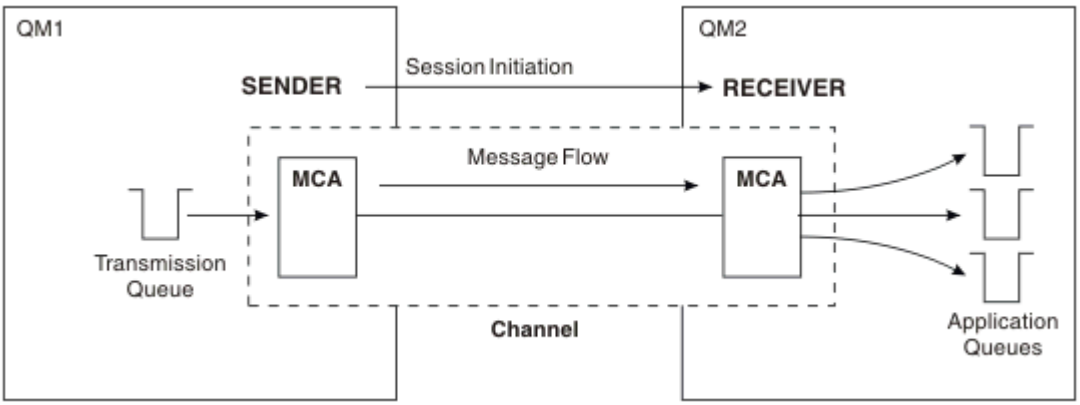


그림 *8.* 송신자-수신자 채널

요청자-서버 채널

한 시스템의 요청자는 채널을 시작하여 메시지를 다른 시스템에 수신할 수 있습니다 요청자는 채널의 다른 끝에 있는 서버가 시작되기를 요청합니다. 서버는 채널 정의에서 정의된 전송 큐에서 요청자에게 메시지를 송신합니다.

서버 채널은 통신을 시작하여 메시지를 요청자에게도 송신할 수 있습니다. 채널 정의에서 지정된 파트너의 연결 이름이 있는 서버 채널인 완전한 서버에만 적용됩니다. 완전한 서버는 요청자에서 시작되거나 요청자와의 통신을 시작할 수 있습니다.

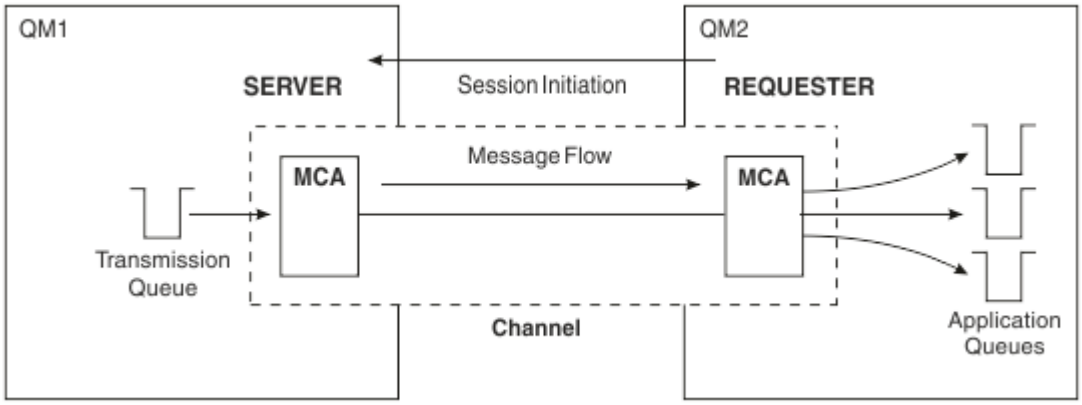


그림 *9.* 요청자-서버 채널

요청자-송신자 채널

요청자는 채널을 시작하고 송신자는 호출을 종료합니다. 그런 다음 송신자는 채널 정의(콜백이라는)의 정보에 따라 통신을 다시 시작합니다. 전송 큐에서 요청자에게 메시지를 송신합니다.

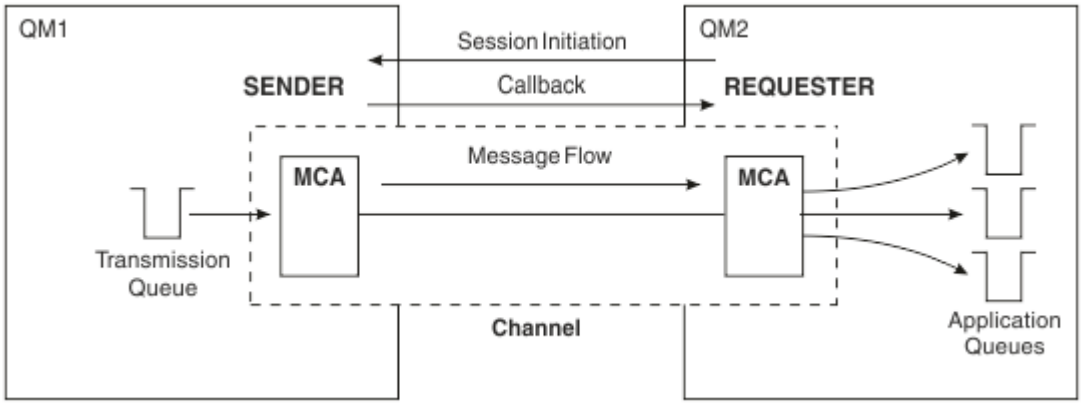


그림 *10.* 요청자-송신자 채널

서버-수신자 채널

송신자-수신자와 같지만 채널 정의에서 지정된 파트너의 연결 이름이 있는 서버 채널인 완전한 서버에만 적용됩니다. 채널 시동은 링크의 서버 끝에서 시작되어야 합니다. 이 그림은 42 페이지의 그림 8과 같습니다.

클러스터-송신자 채널

클러스터에서, 각 큐 관리자에는 클러스터 정보를 전체 저장소 큐 관리자 중 하나로 송신할 수 있는 클러스터-송신자 채널이 있습니다. 큐 관리자는 메시지를 클러스터-송신자 채널의 다른 큐 관리자에게도 송신할 수 있습니다.

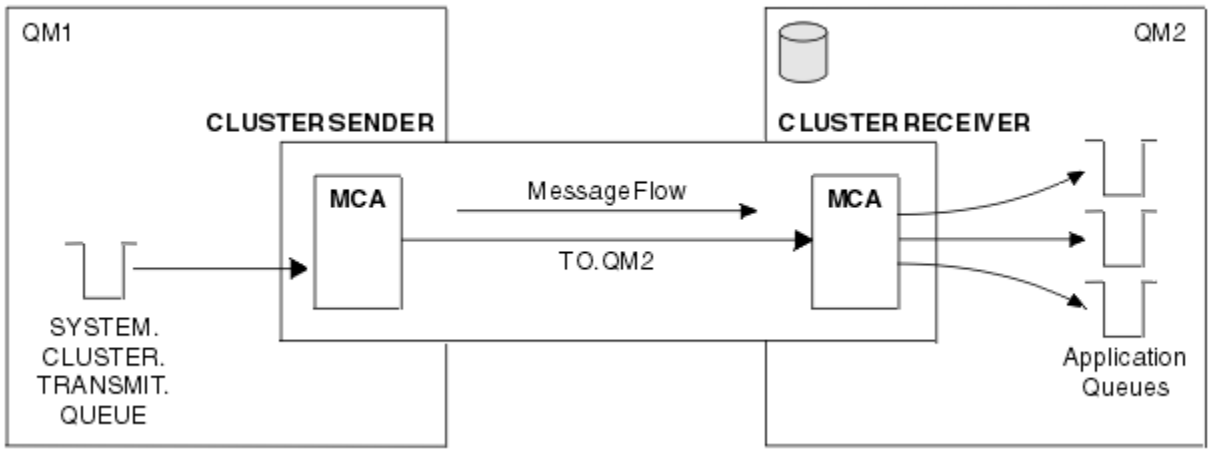


그림 *11.* 클러스터 송신자 채널

클러스터-수신자 채널

클러스터에는, 각 큐 관리자에 클러스터에 관한 정보 및 메시지를 수신할 수 있는 클러스터-수신자 채널이 있습니다. 이 그림은 44 페이지의 그림 11과 같습니다.

데드-레터 큐

데드-레터 큐(또는 미전달 메시지)는 올바른 목적지로 라우트할 수 없는 경우 메시지를 송신하는 큐입니다. 일반적으로, 각 큐 관리자에는 데드-레터 큐가 있습니다.

미전달 메시지 큐라고도 하는 데드-레터 큐(DLQ)는 예를 들어 큐가 존재하지 않거나 가득 차서 목적지 큐로 전달할 수 없는 메시지에 대한 보유 큐입니다. 데드-레터 큐가 데이터 변환 오류에 대한 채널 종료 송신에도 사용됩니다. 네트워크의 모든 큐 관리자는 일반적으로 데드-레터 큐로 사용할 로컬 큐를 가지므로 올바른 목적지로 전달할 수 없는 메시지를 추후 검색을 위해 저장할 수 있습니다.

메시지는 큐 관리자, 메시지 채널 에이전트(MCA) 및 애플리케이션에 의해 DLQ에 넣어질 수 있습니다. DLQ에 대한 모든 메시지는 데드 레터 헤더 구조, MQDLH를 접두부로 사용합니다. MQDLH 구조의 *Reason* 필드에는 메시지가 DLQ에 있는 이유를 식별하는 이유 코드가 포함됩니다.

일반적으로, 각 큐 관리자에 대해 데드-레터 큐를 정의해야 합니다. 이를 고려하지 않고 MCA가 메시지를 넣을 수 없으면, 전송 큐에 남게 되며 채널이 중지됩니다. 또한 빠른 비지속 메시지(빠른 비지속 메시지 참조)를 전달할 수 없으며 데드-레터 큐가 대상 시스템에 없으면 이 메시지는 제거됩니다.

그러나, 데드-레터 큐 사용은 메시지를 전달한 순서에 영향을 줄 수 있어 이를 사용하지 않도록 선택할 수 있습니다.

관련 태스크

데드-레터 큐에 대한 작업

미배달 메시지 문제점 해결

관련 참조

runmqdlq(데드-레터 큐 핸들러 실행)

리모트 큐 정의

리모트 큐 정의는 다른 큐 관리자에서 소유하는 큐에 대한 정의입니다.

애플리케이션이 로컬 큐에서만 메시지를 검색할 수 있는 반면, 로컬 큐나 리모트 큐에 메시지를 넣을 수 있습니다. 그러므로, 각 로컬 큐에 대한 정의뿐 아니라 큐 관리자에는 리모트 큐 정의가 있을 수 있습니다. 리모트 큐 정의의 장점은 리모트 큐나 리모트 큐 관리자의 이름, 또는 전송 큐의 이름을 지정하지 않고 애플리케이션을 사용하여 메시지를 리모트 큐에 넣을 수 있다는 것입니다. 리모트 큐 정의는 위치 독립성을 제공합니다.

리모트 큐 정의를 위한 다른 사용이 있으며, 이는 나중에 설명됩니다.

리모트 큐 관리자로 가져오는 방법

각 소스 및 대상 큐 관리자 간에 하나의 채널이 항상 없을 수 있습니다. 두 관리자 간의 링크 방법으로 멀티호핑, 채널 공유, 다른 채널 사용 및 클러스터링을 포함한 여러 다른 방법이 있습니다.

멀티홉

소스 큐 관리자와 대상 큐 관리자 간에 직접적인 통신 링크가 없는 경우, 대상 큐 관리자로 가는 중에 하나 이상의 중간 큐 관리자를 통해 전달할 수 있습니다. 이를 멀티홉이라고 합니다.

모든 큐 관리자와 중간 큐 관리자의 전송 큐 사이에 채널을 정의해야 합니다. 45 페이지의 그림 12에 설명되어 있습니다.

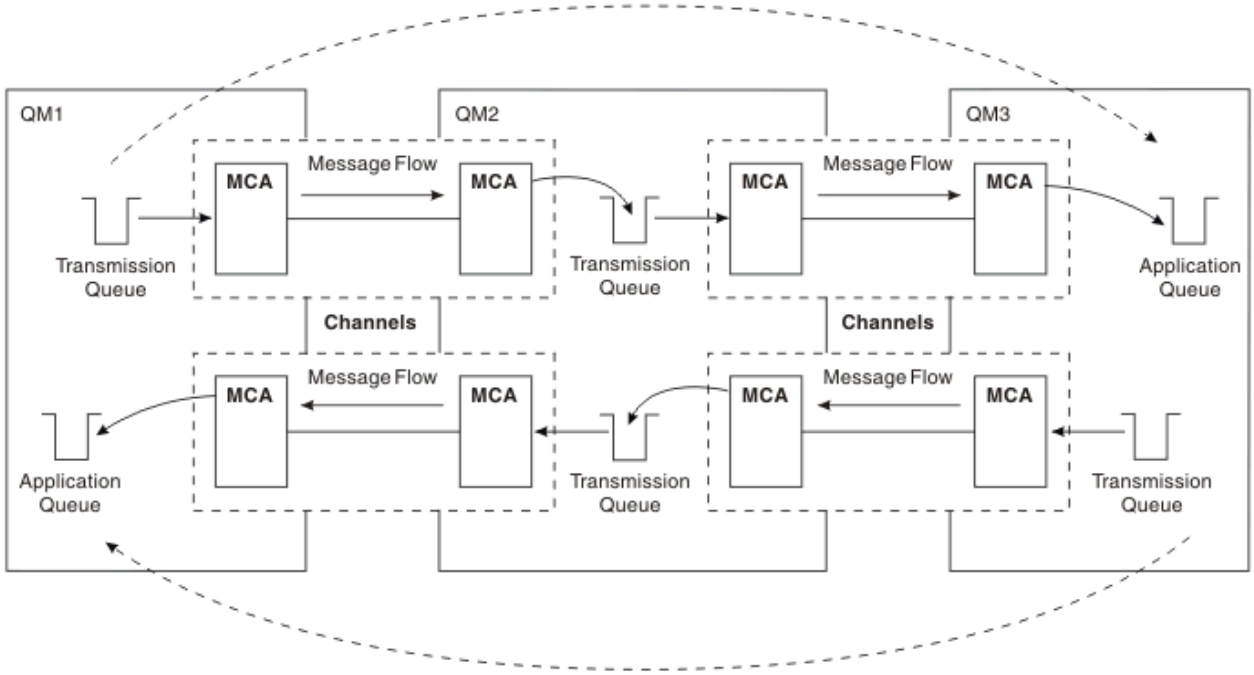


그림 *12.* 중간 큐 관리자를 통해 전달

채널 공유

애플리케이션 설계자로서 애플리케이션이 큐 이름과 함께 리모트 큐 관리자 이름을 지정하도록 강제 실행하거나 각 리모트 큐에 대해 리모트 큐 정의를 작성하도록 선택할 수 있습니다. 이 정의에는 리모트 큐 관리자 이름, 큐 이름 및 전송 큐의 이름이 있습니다. 이런 방식으로, 동일한 리모트 위치에 큐의 주소를 지정하는 모든 애플리케이션의 모든 메시지에는 동일한 전송 큐를 통해 전송된 메시지가 있습니다. 45 페이지의 그림 13에 설명되어 있습니다.

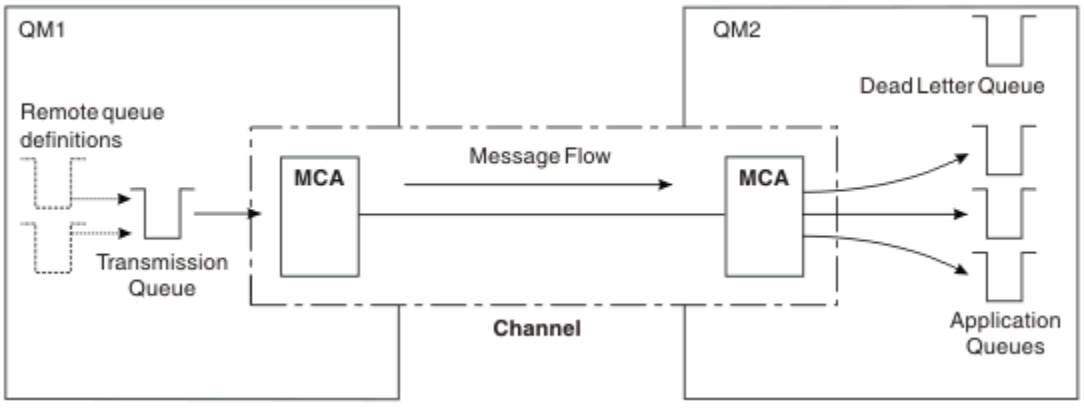


그림 *13.* 전송 큐 공유

45 페이지의 그림 13는 메시지가 다중 애플리케이션에서 다중 리모트 큐까지 동일 채널을 사용할 수 있음을 설명합니다.

다른 채널 사용

두 큐 관리자 간에 송신될 다른 유형의 메시지가 있는 경우 두 관리자 간에 둘 이상의 채널을 정의할 수 있습니다. 아마도 보안 용도로, 대량의 메시지 트래픽에 대한 전달 속도의 균형을 유지하기 위해 대체 채널이 필요한 경우가 있습니다.

다른 채널 및 다른 전송 큐를 정의하기 위해 필요한 두 번째 채널을 설정하려면 위치를 지정하는 리모트 큐 정의 및 전송 큐 이름을 작성하십시오. 그런 다음 애플리케이션은 채널을 사용할 수 있지만 메시지는 여전히 동일한 대상 큐에 전달됩니다. 46 페이지의 그림 14에 설명되어 있습니다.

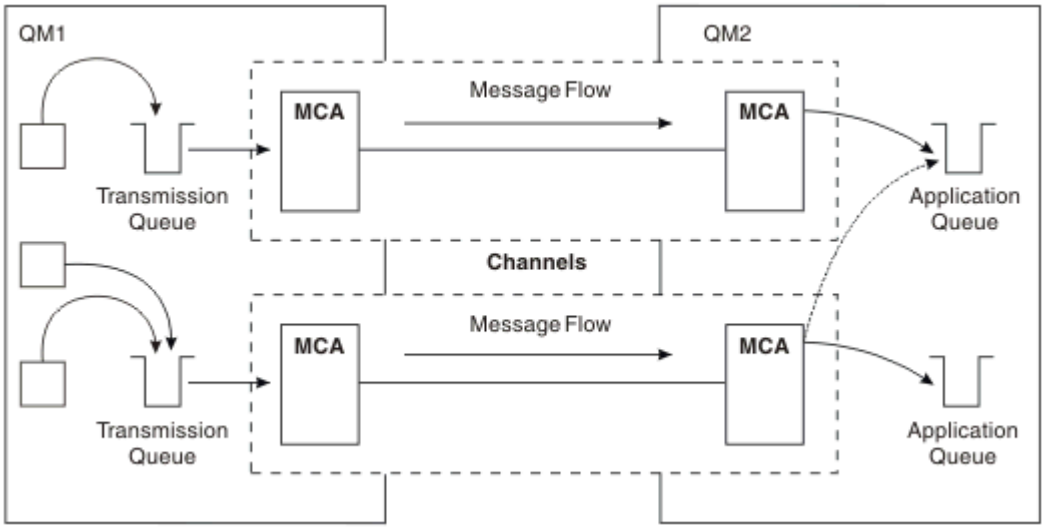


그림 *14.* 다중 채널 사용

리모트 큐 정의를 사용하여 전송 큐를 지정하는 경우 애플리케이션은 위치(즉, 목적지 큐 관리자) 자체를 지정하지 **않아야** 합니다. 이렇게 하는 경우, 큐 관리자는 리모트 큐 정의를 사용하지 않습니다. 리모트 큐 정의는 위치 독립성을 제공합니다. 애플리케이션은 큐가 있는 위치를 알지 않아도 메시지를 논리 큐에 넣을 수 있으며 애플리케이션을 변경하지 않고도 물리적 큐를 변경할 수 있습니다.

클러스터링 사용

클러스터 내의 모든 큐 관리자는 클러스터 수신자 채널을 정의합니다. 다른 큐 관리자가 메시지를 큐 관리자에 보내려는 경우, 해당 클러스터 송신자 채널을 자동으로 정의합니다. 예를 들어, 클러스터에 둘 이상의 큐 인스턴스가 있는 경우 클러스터 송신자 채널은 큐를 호스팅하는 큐 관리자에 정의될 수 있습니다. IBM MQ는 라운드 로빈 루틴을 사용하여 메시지를 라우트할 사용 가능한 큐 관리자를 선택하는 워크로드 관리 알고리즘을 사용합니다. 자세한 정보는 클러스터를 참조하십시오.

주소 지정 정보

애플리케이션이 리모트 큐 관리자에 대해 목적지인 메시지를 넣으면, 로컬 큐 관리자는 전송 큐에 넣기 전에 전송 헤더를 추가합니다. 이 헤더에는 목적지 큐와 큐 관리자의 이름, 즉 주소 지정 정보가 포함됩니다.

단일 큐 관리자 환경에서 애플리케이션이 메시지를 넣을 큐를 열 때 목적지 큐의 주소가 설정됩니다. 목적지 큐가 동일한 큐 관리자에 있기 때문에, 주소 정보에는 필요하지 않습니다.

분산 큐잉 환경에서 큐 관리자는 목적지 큐 이름뿐만 아니라 해당 큐의 위치(즉, 큐 관리자 이름) 및 원격 위치에 대한 라우트(즉, 전송 큐)를 알아야 합니다. 이 주소 지정 정보는 전송 헤더에 포함되어 있습니다. 수신 채널은 전송 헤더를 제거하며 전송 헤더의 정보를 사용하여 목적지 큐를 찾습니다.

리모트 큐 정의를 사용하는 경우 애플리케이션이 목적지 큐 관리자의 이름을 지정하지 않아도 됩니다. 이 정의는 리모트 큐의 이름, 메시지의 대상인 리모트 큐 관리자의 이름 및 메시지를 전송하는 데 사용되는 전송 큐의 이름을 지정합니다.

알리어스의 개념

알리어스는 메시지 서비스 품질을 제공하는 데 사용됩니다. 애플리케이션을 변경하지 않고도 큐 관리자 알리어스를 사용하여 시스템 관리자가 대상 큐 관리자의 이름을 변경할 수 있습니다. 또한 시스템 관리자가 라우트를 목적지 큐 관리자로 변경하거나 다른 여러 큐 관리자를 통한 전달(멀티호핑)과 관련된 라우트를 설정할 수 있습니다. 응답 대상 큐 알리어스는 응답을 위한 서비스 품질을 제공합니다.

큐 관리자 알리어스 및 응답 대상 큐 알리어스는 공백 RNAME이 있는 리모트 큐 정의를 사용하여 작성됩니다. 이 정의는 실제 큐를 정의하지 않습니다. 큐 관리자가 이 정의를 사용하여 물리적인 큐 이름, 큐 관리자 이름 및 전송 큐를 해석합니다.

알리어스 정의는 공백 RNAME이 있다는 특징을 가집니다.

큐 이름 해석

큐가 열릴 때마다 모든 큐 관리자에서 큐 이름 해석이 이루어집니다. 대상 큐, 대상 큐 관리자(로컬일 수 있음) 및 큐 관리자에 대한 라우트(널일 수 있음)를 식별하는 것이 목적입니다. 해석된 이름에는 큐 관리자 이름, 큐 이름 및 큐 관리자가 리모트인 경우 전송 큐의 세 부분이 있습니다.

리모트 큐 정의가 존재하면 알리어스 정의를 참조하지 않습니다. 애플리케이션에서 제공되는 큐 이름은 리모트 큐 정의에서 지정된 전송 큐, 리모트 큐 관리자 및 목적지 큐의 이름으로 해석됩니다. 큐 이름 해석에 대한 자세한 정보는 큐 이름 해석을 참조하십시오.

리모트 큐 정의가 없고 큐 관리자 이름이 지정되어 있거나 이름 서비스로 해석된 경우, 큐 관리자는 제공된 큐 관리자 이름과 일치하는 큐 관리자 알리어스 정의가 있는지 보고 확인합니다. 있는 경우, 정의 내의 정보를 사용하여 큐 관리자 이름을 목적지 큐 관리자의 이름으로 해석합니다. 큐 관리자 알리어스 정의는 목적지 큐 관리자에 대한 전송 큐를 판별하는 데도 사용될 수 있습니다.

해석된 큐 이름이 로컬 큐가 아닌 경우, 큐 관리자 이름 및 큐 이름 모두 애플리케이션이 전송 큐에 넣은 각 메시지의 전송 헤더에 포함됩니다.

리모트 큐 정의 또는 큐 관리자 알리어스 정의가 변경되지 않는 한 일반적으로 사용된 전송 큐의 이름은 해석된 큐 관리자와 동일합니다. 이러한 전송 큐를 정의하지 않았지만 기본 전송 큐를 정의한 경우 이 이름이 사용됩니다.

z/OS z/OS에서 실행 중인 큐 관리자의 이름은 4자로 제한됩니다.

큐 관리자 알리어스 정의

큐를 열어 메시지를 넣은 애플리케이션이 큐 이름 **및** 큐 관리자 이름을 지정하는 경우 큐 관리자 알리어스 정의가 적용됩니다.

큐 관리자 알리어스 정의는 다음과 같이 3가지로 사용됩니다.

- 메시지 송신 시, 큐 관리자 이름 다시 맵핑
- 메시지 송신 시, 전송 큐 변경 또는 지정
- 메시지 수신 시, 로컬 큐 관리자가 해당 메시지에 대해 의도된 목적지인지 여부 판별

아웃바운드 메시지 - 큐 관리자 이름 다시 맵핑

큐 관리자 알리어스 정의를 사용하여 MQOPEN 호출에서 지정된 큐 관리자 이름을 다시 맵핑할 수 있습니다. 예를 들어, MQOPEN 호출은 THISQ의 큐 이름 및 YOURQM의 큐 관리자 이름을 지정합니다. 로컬 큐 관리자에 다음 예제와 같은 큐 관리자 알리어스 정의가 있습니다.

```
DEFINE QREMOTE (YOURQM) RQMNAME(REALQM)
```

애플리케이션이 메시지를 큐 관리자 YOURQM에 넣을 때 사용할 실제 큐 관리자가 REALQM임을 표시합니다. 로컬 큐 관리자가 REALQM인 경우, 메시지를 로컬 큐인 THISQ 큐에 넣습니다. 로컬 큐 관리자가 REALQM이 아닌 경우, REALQM이라는 전송 큐로 메시지를 라우트합니다. 큐 관리자는 YOURQM 대신 REALQM을 알리도록 전송 헤더를 변경합니다.

아웃바운드 메시지 - 전송 큐 대체 또는 지정

48 페이지의 그림 15에서는 큐 관리자 QM3에 큐 이름을 표시하는 전송 헤더가 있는 큐 관리자 QM1에 메시지가 도달하는 시나리오를 보여줍니다. 이 시나리오에서 QM3는 QM2를 통해 멀티호핑하여 도달할 수 있습니다.

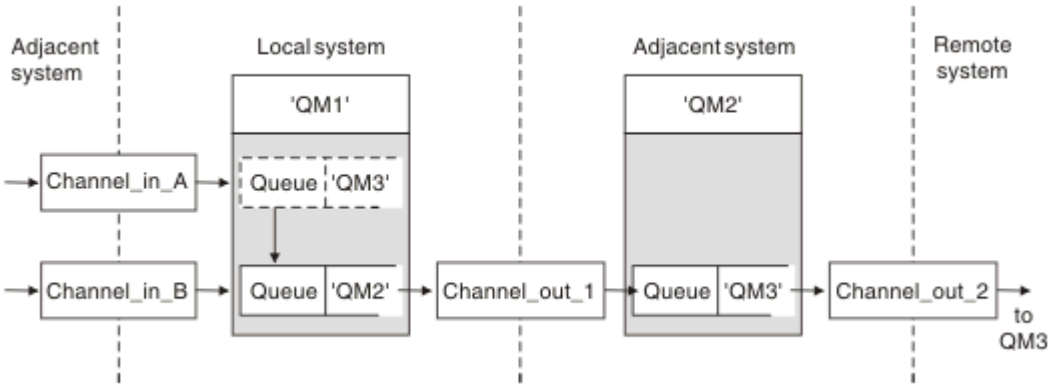


그림 *15.* 큐 관리자 알리어스

QM3의 모든 메시지는 큐 관리자 알리어스가 있는 QM1에 캡처됩니다. 큐 관리자 알리어스의 이름은 QM3로 지정되며 전송 큐 QM2를 통한 정의 QM3를 포함합니다. 정의는 다음 예제와 같습니다.

```
DEFINE QREMOTE (QM3) RNAME(' ') RQMNAME(QM3) XMITQ(QM2)
```

큐 관리자는 메시지를 전송 큐 QM2에 넣지만 목적지 큐 관리자 QM3의 이름이 변경되지 않아 전송 큐 헤더를 변경하지 않습니다.

QM1에 도달하여 QM2의 큐 이름을 포함한 전송 헤더를 표시하는 모든 메시지도 QM2 전송 큐에 넣습니다. 이런 방식으로 목적지가 다른 메시지는 적절한 인접 시스템에 대한 공용 전송 큐로 수집되어 계속 목적지로 전송됩니다.

인바운드 메시지 - 목적지 판별

수신 MCA는 전송 헤더에서 참조된 큐를 엽니다. 큐 관리자 알리어스 정의가 참조된 큐 관리자와 동일한 이름으로 존재하면 전송 헤더에서 수신된 큐 관리자 이름은 해당 목적지의 RQMNAME으로 바뀝니다.

이 프로세스는 다음과 같이 2가지로 사용됩니다.

- 메시지를 다른 큐 관리자에게 전달
- 큐 관리자 이름을 로컬 큐 관리자와 동일하도록 변경

응답 대상 큐 알리어스 정의

응답 대상 큐 알리어스 정의는 메시지 디스크립터에서 응답 정보에 대한 대체 이름을 지정합니다. 애플리케이션을 변경하지 않고 큐나 큐 관리자의 이름을 변경할 수 있다는 이점이 있습니다.

큐 이름 해석

애플리케이션이 메시지에 응답할 때 수신된 메시지의 메시지 디스크립터에 있는 데이터를 사용하여 응답할 큐의 이름을 찾습니다. 송신 애플리케이션은 응답이 송신되어 이 정보를 메시지에 첨부하는 위치를 표시합니다. 이 개념은 애플리케이션 디자인의 일부로 통합될 수 있습니다.

큐 이름 해석은 메시지를 큐에 넣기 전에 애플리케이션의 송신 끝에서 발생합니다. 따라서 큐 이름 해석은 원격 애플리케이션과 상호작용하기 전에 발생합니다. 이 상황에서만 큐가 열리지 않을 때 이름 해석이 이루어집니다.

큐 관리자 알리어스를 사용하여 큐 이름 해석

보통 애플리케이션은 응답 대상 큐를 지정하며 응답 대상 큐 관리자 이름을 공백으로 둡니다. 큐 관리자는 넣기 시 자신의 이름을 완성합니다. 예를 들어, 이 방법은 전송 큐 QM1을 사용하는 기본 리턴 채널 대신 전송 큐 QM1_relief를 사용하는 채널의 경우와 같이 응답에 사용할 대체 채널을 원하는 경우를 제외하고는 잘 작동합

니다. 이런 상황에서는 전송 큐 헤더에 지정된 큐 관리자 이름이 "실제" 큐 관리자 이름과 일치하지 않지만 큐 관리자 알리어스 정의를 사용하여 다시 지정됩니다. 대체 라우트와 함께 응답을 리턴하려면 응답 대상 큐 알리어스 정의를 사용하여 응답 대상 큐 데이터에도 맵핑해야 합니다.

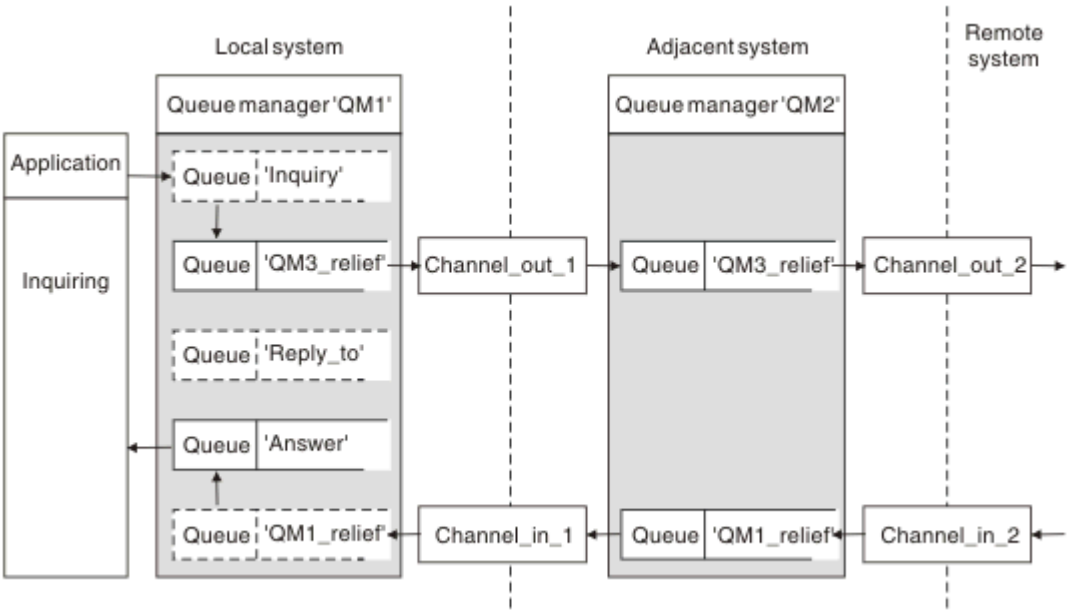


그림 *16.* 응답 위치 변경에 사용되는 응답 대상 큐 알리어스

49 페이지의 그림 16에서의 예제:

1. 애플리케이션은 MQPUT 호출을 사용하고 메시지 디스크립터에서 다음 정보를 지정하여 메시지를 넣습니다.

   ```
   ReplyToQ='Reply_to'
   ReplyToQMgr=' '
   ```

   응답 대상 큐 알리어스가 사용되려면 ReplyToQMgr는 공백이어야 합니다.
2. 이름 Answer 및 큐 관리자 이름 QM1_relief가 포함된 Reply_to라는 응답 대상 큐 알리어스 정의를 작성합니다.

   ```
   DEFINE QREMOTE ('Reply_to') RNAME ('Answer')
   RQMNAME ('QM1_relief')
   ```
3. 메시지는 ReplyToQ='Answer' 및 ReplyToQMgr='QM1_relief'를 표시하는 메시지 디스크립터와 함께 전송됩니다.
4. 애플리케이션 스펙에는 응답을 Reply_to가 아닌 큐 Answer에서 찾을 수 있는 정보를 포함해야 합니다.

다음을 정의하는 병렬 리턴 채널을 작성해야 하는 응답을 준비하려면:

- QM2에서, QM1_relief라는 전송 큐

  ```
  DEFINE QLOCAL ('QM1_relief') USAGE(XMITQ)
  ```
- QM1에서, 큐 관리자 알리어스 QM1_relief

  ```
  DEFINE QREMOTE ('QM1_relief') RNAME() RQMNAME(QM1)
  ```

  이 큐 관리자 알리어스는 병렬 리턴 채널의 체인을 종료하며 QM1의 메시지를 캡처합니다.

나중에 가끔 이를 수행하기를 원한다고 생각되면, 애플리케이션이 시작부터 알리어스 이름을 사용하는지 확인하십시오. 현재까지는 응답 대상 큐에 대한 정상 큐 알리어스이지만 나중에는 큐 관리자 알리어스로 변경될 수 있습니다.

응답 대상 큐 이름

응답 대상 큐 이름 지정에 주의가 필요합니다. 애플리케이션이 응답 대상 큐 이름을 메시지에 넣은 이유는 응답이 전송되는 큐를 지정할 수 있기 때문입니다. 이 이름으로 응답 대상 큐 알리어스 정의를 작성하면 실제 응답 대상 큐(즉, 로컬 큐 정의)를 동일한 이름으로 지정할 수 없습니다. 그러므로 응답 대상 큐 알리어스 정의가 큐 관리자 이름뿐만 아니라 새 큐 이름을 포함해야 하며 애플리케이션 스펙에는 다른 큐에서 응답을 찾을 수 있는 정보를 포함해야 합니다.

이제 애플리케이션은 원본 메시지를 넣을 때의 응답 대상 큐로 이름 지정된 큐와 다른 큐에서 메시지를 검색해야 합니다.

클러스터 컴포넌트

클러스터는 큐 관리자, 클러스터 저장소, 클러스터 채널 및 클러스터 큐로 구성됩니다.

각 클러스터 컴포넌트에 대한 정보는 다음 하위 주제를 참조하십시오.

관련 개념
클러스터링과 분산 큐잉의 비교
관련 태스크
큐 관리자 클러스터 구성
새 클러스터 설정

클러스터 저장소

저장소는 클러스터의 멤버인 큐 관리자에 대한 정보의 콜렉션입니다.

저장소 정보에는 큐 관리자 이름, 해당 위치, 해당 채널, 호스팅하고 있는 큐 및 기타 정보가 포함됩니다. 이 정보는 SYSTEM.CLUSTER.REPOSITORY.QUEUE라고 하는 큐에 메시지의 양식으로 저장됩니다. 이 큐는 기본 오브젝트 중 하나입니다. Multi 다중 플랫폼에서는 IBM MQ 큐 관리자를 작성할 때 정의됩니다. z/OS IBM MQ for z/OS에서 큐 관리자 사용자 정의의 일부로 정의됩니다.

전체 저장소와 부분 저장소

일반적으로, 클러스터에서 두 개의 큐 관리자가 전체 저장소를 보유합니다. 나머지 큐 관리자 모두 부분 저장소를 보유합니다.

클러스터에서 모든 큐 관리자에 대한 전체 정보 세트를 호스팅하는 큐 관리자에 전체 저장소가 있습니다. 클러스터의 다른 큐 관리자에는 전체 저장소에 있는 정보의 서브세트를 포함하는 부분 저장소가 있습니다.

부분 저장소에는 큐 관리자가 메시지를 교환해야 하는 큐 관리자에 대한 정보만 포함됩니다. 큐 관리자는 필요한 정보에 대한 업데이트를 요청합니다. 그러면 필요한 경우, 전체 저장소 큐 관리자가 큐 관리자에 새 정보를 보낼 수 있습니다. 많은 시간 동안, 부분 저장소에는 큐 관리자가 클러스터 내에서 수행하는 데 필요한 모든 정보가 포함됩니다. 큐 관리자에서 일부 추가 정보가 필요한 경우, 큐 관리자는 전체 저장소를 조회하여 자신의 부분 저장소를 업데이트합니다. 큐 관리자는 SYSTEM.CLUSTER.COMMAND.QUEUE라는 큐를 사용하여 저장소에 대한 업데이트를 요청 및 수신합니다.

클러스터의 멤버인 큐 관리자를 마이그레이션하는 경우 부분 저장소를 마이그레이션하기 전에 전체 저장소를 마이그레이션하십시오. 이는 이전 저장소가 신규 릴리스에서 도입된 새 속성을 저장하지 못하기 때문입니다. 이전 저장소는 이를 허용하지만 저장하지는 않습니다.

클러스터 큐 관리자

클러스터 큐 관리자는 클러스터의 멤버인 큐 관리자입니다.

큐 관리자는 둘 이상의 클러스터의 멤버가 될 수 있습니다. 각 클러스터 큐 관리자의 이름은 멤버인 모든 클러스터에서 고유해야 합니다.

클러스터 큐 관리자는 큐를 호스팅할 수 있습니다. 이 큐는 클러스터에서 다른 큐 관리자에 광고합니다. 그러나 이 작업을 수행할 필요는 없습니다. 이 큐 관리자는 클러스터의 다른 위치에서 호스트된 큐에 메시지를 대신 공급하고 해당 클러스터에 명시적으로 전달되는 응답만 수신할 수 있습니다.

z/OS IBM MQ for z/OS에서 클러스터 큐 관리자는 큐 공유 그룹의 멤버일 수 있습니다. 이러한 경우, 클러스터 큐 관리자는 샘플 큐 공유 그룹의 다른 큐 관리자와 해당되는 큐 정의를 공유합니다.

클러스터 큐 관리자는 자율적입니다. 클러스터 큐 관리자는 정의하는 채널 및 큐에 대해 전체 제어를 가지고 있습니다. 해당되는 정의는 다른 큐 관리자(동일한 큐 공유 그룹에 있는 큐 관리자가 아닌 다른)에서 수행할 수 없습니다. 저장소 큐 관리자는 클러스터의 다른 큐 관리자에서 정의를 제어하지 않습니다. 필요할 때 사용하기 위해 모든 전체 정의 세트를 보유합니다. 클러스터는 큐 관리자의 연합입니다.

클러스터 큐 관리자에서 정의를 작성하거나 변경한 후, 정보는 전체 저장소 큐 관리자로 보내집니다. 클러스터의 다른 저장소는 나중에 업데이트됩니다.

전체 저장소 큐 관리자

전체 저장소 큐 관리자는 클러스터의 자원에 대한 전체 표시를 보유하는 클러스터 큐 관리자입니다. 사용 가능성을 보증하기 위해, 각 클러스터에서 두 개 이상의 전체 저장소 큐 관리자를 설정하십시오. 전체 저장소 큐 관리자는 클러스터의 다른 큐 관리자가 보낸 정보를 수신하고 해당되는 저장소를 업데이트합니다. 둘 다 클러스터에 대한 새 정보로 최신 상태를 유지하도록 하기 위해 서로 메시지를 보냅니다.

큐 관리자 및 저장소

모든 클러스터에는 클러스터에 있는 큐 관리자, 큐 및 채널에 대한 정보의 전체 저장소를 보유하는 최소 하나의 (두 개가 선호됨) 큐 관리자가 있습니다. 이 저장소는 또한 클러스터의 다른 큐 관리자로부터의 정보에 대한 업데이트 요청을 포함합니다.

다른 큐 관리자는 각각 통신해야 하는 큐 관리자와 큐 서브세트에 대한 정보를 포함하는 부분 저장소를 보유합니다. 큐 관리자는 먼저 다른 큐 또는 큐 관리자에 액세스해야 하는 경우 조회를 작성하여 부분 저장소를 빌드합니다. 큐 관리자는 해당 큐 또는 큐 관리자에 관한 새 정보를 알리도록 요청합니다.

각 큐 관리자는 `SYSTEM.CLUSTER.REPOSITORY.QUEUE`라고 하는 큐에 메시지의 저장소 정보를 저장합니다. 큐 관리자는 `SYSTEM.CLUSTER.COMMAND.QUEUE`라고 하는 큐에서 메시지의 저장소 정보를 교환합니다.

클러스터에 조인하는 각 큐 관리자는 클러스터 송신자 `CLUSSDR` 채널을 저장소 중 하나에 대해 정의합니다. 클러스터의 다른 큐 관리자가 전체 저장소를 보유한다고 즉시 학습합니다. 그때부터, 큐 관리자는 저장소에서 정보를 요청할 수 있습니다. 큐 관리자가 정보를 선택된 저장소로 보낼 때, 하나의 다른 저장소(있는 경우)로도 정보를 보냅니다.

전체 저장소는 호스팅하는 큐 관리자가 링크되는 큐 관리자 중 하나에서 새 정보를 수신할 때 업데이트됩니다. 새 정보는 또한 저장소 큐 관리자가 서비스 외부에 있는 경우 지연되는 위험을 줄이기 위해 다른 저장소로 보냅니다. 모든 정보는 두 번 보내므로, 저장소는 중복을 버려야 합니다. 정보의 각 항목은 저장소가 중복을 식별하기 위해 사용하는 순서 번호를 전달합니다. 모든 저장소는 메시지를 교환하여 서로 맞춥니다.

클러스터 큐

클러스터 큐는 클러스터 큐 관리자에 의해 호스팅되며 클러스터의 다른 큐 관리자가 사용할 수 있는 큐입니다.

클러스터 큐 정의는 클러스터의 다른 큐 관리자에 통지됩니다. 다른 큐 관리자는 해당하는 리모트 큐 정의 없이도 클러스터 큐에 메시지를 넣을 수 있습니다. 클러스터 큐는 클러스터 이름 목록을 사용하여 둘 이상의 클러스터에 통지될 수 있습니다.

큐가 통지되면 클러스터의 큐 관리자가 해당 큐에 메시지를 넣을 수 있습니다. 메시지를 넣으려면 큐 관리자가 전체 저장소에서 큐가 호스팅되고 있는 위치를 찾아야 합니다. 그런 다음 메시지에 몇 가지 라우팅 정보를 추가하고 클러스터 전송 큐에 메시지를 넣습니다.

z/OS 클러스터 큐는 IBM MQ for z/OS에서 큐 공유 그룹의 멤버가 공유하는 큐일 수 있습니다.

관련 태스크

클러스터 큐 정의

공유 큐와 클러스터 큐의 비교

이 정보는 공유 큐와 클러스터 큐를 비교하고 사용자의 시스템에 더 적합한 것을 결정하도록 돕기 위해 설계되었습니다.

채널 시작기 비용

클러스터 큐에서 메시지는 채널에서 전송하므로 애플리케이션 비용에 채널 시작기 비용이 추가됩니다. 채널이 메시지를 보내고 받으므로 네트워크에 비용이 발생합니다. 공유 큐에서는 이러한 비용이 발생하지 않으므로 큐 공유 그룹의 큐 관리자 사이에 메시지를 이동하는 경우 클러스터 큐보다 더 적은 처리 능력을 사용합니다.

메시지의 사용가능성

큐에 넣는 경우 클러스터 큐는 사용자의 큐 관리자에 연결된 활성 채널이 있는 큐 관리자 중 하나로 메시지를 송신합니다. 리모트 큐 관리자에서 메시지를 처리하는 데 사용되는 애플리케이션이 작동하지 않는 경우 메시지는 처리되지 않으며 애플리케이션이 시작될 때까지 대기합니다. 마찬가지로, 큐 관리자가 종료된 경우 큐 관리자의 모든 메시지는 큐 관리자가 다시 시작될 때까지 사용할 수 없습니다. 이러한 인스턴스는 공유 큐를 사용하는 경우보다 더 낮은 메시지 가용성을 나타냅니다.

공유 큐를 사용하는 경우, 큐 공유 그룹의 모든 애플리케이션은 전송된 메시지를 받을 수 있습니다. 큐 공유 그룹에서 단일 큐 관리자를 종료하는 경우에는 다른 큐 관리자에 대해 메시지를 사용할 수 있으므로 클러스터 큐를 사용할 때보다 높은 메시지 가용성을 제공합니다.

용량

커플링 기능은 디스크보다 비쌉니다. 그러므로 로컬 큐에 1,000,000개의 메시지를 저장하는 것이 동일한 수의 메시지를 저장할 수 있는 충분한 용량이 있는 커플링 기능보다 비용이 덜 듭니다.

다른 큐 관리자로 송신

공유 큐 메시지는 큐 공유 그룹 내에서만 사용 가능합니다. 큐 공유 그룹 외부에서 큐 관리자를 사용하려면, 채널을 사용해야 합니다. 클러스터링을 사용하여 여러 원격 분산 큐 관리자 간에 작업을 분산할 수 있습니다.

워크로드 밸런싱

클러스터링을 사용하여 메시지 전송의 일부를 받을 채널 및 큐 관리자에 가중치를 부여할 수 있습니다. 예를 들어, 메시지의 60%를 하나의 큐 관리자로 송신하고 메시지의 40%를 다른 큐 관리자로 송신할 수 있습니다. 이 인스턴스는 작업을 처리할 리모트 큐 관리자의 능력에 따라 달라지지 않습니다. 첫 번째 큐 관리자가 있는 시스템이 과부하되고 두 번째 큐 관리자가 있는 시스템이 유휴 상태여도 대부분의 메시지는 여전히 첫 번째 큐 관리자로 이동합니다.

공유 큐를 사용하는 경우에는 두 CICS® 시스템이 메시지를 받을 수 있습니다. 한 시스템이 과부하되면 다른 시스템이 대부분의 과부하를 넘겨 받습니다.

클러스터 채널

모든 전체 저장소에서 클러스터-수신기 채널과 클러스터에 있는 기타 모든 전체 저장소에 연결하는 클러스터-송신기 채널 세트를 수동으로 정의합니다. 부분 저장소를 추가할 때 클러스터-수신기 채널과 전체 저장소 중 하나에 연결하는 단일 클러스터-송신기 채널을 수동으로 정의합니다. 추가 클러스터-송신기 채널은 필요할 때 클러스터에서 자동으로 정의합니다. 자동 정의된 클러스터-송신기 채널은 수신 큐 관리자에서 대응하는 클러스터-수신기 채널 정의의 속성을 사용합니다.

클러스터-수신자 채널: CLUSRCVR

CLUSRCVR 채널 정의는 클러스터 큐 관리자가 클러스터에서 다른 큐 관리자로부터 메시지를 수신할 수 있는 채널의 끝을 정의합니다.

클러스터 큐 관리자마다 하나 이상의 CLUSRCVR 채널을 정의해야 합니다. CLUSRCVR 채널을 정의하여, 큐 관리자는 메시지를 수신할 수 있음을 다른 클러스터 큐 관리자에 표시합니다.

CLUSRCVR 채널 정의를 사용하여 다른 큐 관리자가 해당 클러스터-송신자 채널 정의를 자동으로 정의할 수 있습니다. 이 문서의 53 페이지의 『자동 정의된 클러스터-송신자 채널』 절을 참조하십시오.

클러스터-송신자 채널: CLUSSDR

전체 저장소 큐 관리자 모두에서 클러스터에 있는 다른 모든 전체 저장소 큐 관리자까지 CLUSSDR 채널을 수동으로 정의합니다. 전체 저장소에 의해 교환되는 모든 업데이트는 이 채널에서 독점적으로 플로우됩니다. 이러한 채널을 수동으로 정의하여 전체 저장소의 네트워크를 명시적으로 제어합니다.

부분 저장소 큐 관리자를 클러스터에 추가할 때 전체 저장소 중 하나에 연결할 단일 CLUSSDR 채널을 수동으로 정의합니다. 처음에 접속한 후 CLUSSDR 채널을 포함하는 큐 관리자의 추가 클러스터 큐 관리자 오브젝트가 필요에 따라 자동으로 정의되므로 어떤 전체 저장소를 선택하느냐는 관계가 거의 없습니다. 따라서 큐 관리자가 모든 전체 저장소에 클러스터 정보를 보내고, 클러스터에 있는 모든 큐 관리자에 메시지를 보낼 수 있습니다.

이 문서의 절에 설명되어 있는 것처럼 자동 정의된 송신자 채널은 클러스터-수신자 채널의 구성을 기반으로 합니다. 따라서 클러스터 채널에 설정하는 채널 특성은 일치하는 CLUSSDR 및 클러스터-수신자 채널에서 동일하게 설정되거나 클러스터-수신자 채널에만 설정되어야 합니다.

이전에 설명한 이유에 한해서만 CLUSSDR 채널을 수동으로 정의해야 합니다. 즉, 부분 저장소를 전체 저장소에 처음 연결하거나 두 개의 전체 저장소를 함께 연결하려는 경우입니다. 부분 저장소에 연결하거나 클러스터에 없는 큐 관리자에 연결하는 CLUSSDR 채널을 수동으로 구성하면 AMQ9427 및 AMQ9428과 같은 오류 메시지가 발생됩니다. 예를 들어, 전체 저장소의 위치를 수정하는 경우와 같이 일시적으로 이러한 상황이 불가피할 수는 있지만 수동 정의는 가능한 빨리 삭제해야 합니다.

자동 정의된 클러스터-송신자 채널

일반적으로 부분 저장소 큐 관리자를 클러스터에 추가할 때 다음과 같이 큐 관리자에서 두 개의 클러스터 채널만 수동으로 정의합니다.

- 클러스터의 전체 저장소 큐 관리자에 대한 클러스터-송신자(CLUSSDR) 채널.
- 클러스터 수신자(CLUSRCVR) 채널.

사용자가 정의하는 CLUSSDR 채널을 사용하여 큐 관리자가 클러스터에 처음 접속할 수 있습니다. 처음 접속한 후 클러스터가 필요에 따라 추가 CLUSSDR 채널을 자동으로 정의합니다.

자동 정의된 CLUSSDR 채널은 수신 큐 관리자에 있는 대응하는 CLUSRCVR 채널 정의의 속성을 사용합니다. 수동으로 정의된 CLUSSDR 채널이 있어도 자동 정의된 CLUSSDR 채널의 속성이 사용됩니다. 예를 들어, **CONNAME** 매개변수에 포트 번호를 지정하지 않고 CLUSRCVR 채널을 정의하고 포트 번호를 지정하는 CLUSSDR 채널을 수동으로 정의한다고 가정하십시오. 자동 정의된 CLUSSDR 채널이 수동으로 정의된 채널을 대체하면 포트 번호(CLUSRCVR 채널에서 가져옴)가 비어 있게 됩니다. 기본 포트 번호를 사용하고 채널이 실패합니다.

수동으로 정의된 CLUSSDR 채널과 대응하는 CLUSRCVR 채널 정의의 구성이 서로 다른 경우, 일부 차이점은 바로 적용되며(예: 워크로드 밸런싱 매개변수) 다른 일부는 채널이 재시작될 때 적용됩니다(예: TLS 구성).

혼동을 피하기 위해 가능하면 다음 지침을 준수하십시오.

- 전체 저장소를 가리키는 CLUSSDR 채널만 수동으로 정의하십시오.
- 수동으로 정의한 CLUSSDR 채널이 있는 경우 수신 큐 관리자에 있는 대응하는 CLUSRCVR 채널 정의와 동일하게 구성하십시오.

자동 정의된 채널에 대한 작업도 참조하십시오.

관련 개념
자동 정의된 채널에 대한 작업
클러스터 전송 큐 및 클러스터 송신자 채널에 대한 작업
관련 태스크
새 클러스터 설정
클러스터에 큐 관리자 추가

클러스터 토픽

클러스터 토픽은 **cluster** 속성이 정의된 관리 토픽입니다. 클러스터 토픽에 대한 정보는 클러스터의 모든 멤버에게 푸시된 다음 로컬 토픽과 결합되어 여러 큐 관리자에 걸쳐 있는 토픽 공간의 일부분을 구성합니다. 따라서 한 큐 관리자의 토픽에 발행된 메시지를 클러스터에 있는 다른 큐 관리자의 구독으로 전달할 수 있습니다.

큐 관리자에 대한 클러스터 토픽을 정의하면 클러스터 토픽 정의가 전체 저장소 큐 관리자로 송신됩니다. 그러면 전체 저장소에서 클러스터 토픽 정의가 클러스터 내의 모든 큐 관리자로 전파되므로 클러스터의 모든 큐 관리자에 있는 발행자와 구독자가 동일한 클러스터 토픽을 사용할 수 있습니다. 클러스터 토픽이 작성된 큐 관리자를 클러스터 토픽 호스트라고 합니다. 클러스터 토픽은 클러스터의 모든 큐 관리자에서 사용할 수 있지만, 해당 토픽이 정의된 큐 관리자(호스트)에서 수정해야 합니다. 이 경우 수정사항이 전체 저장소를 통해 클러스터의 모든 멤버에게 전파됩니다.

직접 라우팅 또는 토픽 호스트 라우팅을 사용하도록 클러스터 토픽을 구성하는 데 대한 정보 및 클러스터된 토픽 상속 및 와일드카드 구독에 대한 정보는 클러스터 토픽 정의를 참조하십시오.

클러스터 토픽을 표시하는 데 사용할 명령에 대한 정보는 관련 정보를 참조하십시오.

관련 개념
관리 토픽에 대한 작업
구독에 대한 작업
관련 참조
표시 주제
표시 TP상태
표시 등록

기본 클러스터 오브젝트

Multi 멀티플랫폼에서 기본 클러스터 오브젝트는 큐 관리자를 정의할 때 자동으로 작성되는 기본 오브젝트 세트에 포함됩니다. z/OS z/OS의 사용자 정의 샘플에서 기본 클러스터 오브젝트 정의를 찾을 수 있습니다.

참고: 다른 채널 정의와 동일한 방법으로, MQSC 또는 PCF 명령을 실행하여 기본 채널 정의를 변경할 수 있습니다. SYSTEM.CLUSTER.HISTORY.QUEUE의 경우를 제외하고는 기본 큐 정의를 변경하지 마십시오.

SYSTEM.CLUSTER.COMMAND.QUEUE
클러스터의 각 큐 관리자는 메시지를 전체 저장소로 전송하는 데 사용되는 SYSTEM.CLUSTER.COMMAND.QUEUE라고 하는 로컬 큐입니다. 메시지에는 큐 관리자에 대한 새 정보나 변경된 정보, 또는 다른 큐 관리자에 관한 정보에 대한 요청이 포함됩니다.
SYSTEM.CLUSTER.COMMAND.QUEUE는 보통 비어 있습니다.

SYSTEM.CLUSTER.HISTORY.QUEUE
클러스터의 각 큐 관리자에는 SYSTEM.CLUSTER.HISTORY.QUEUE(이)라는 로컬 큐가 있습니다.
SYSTEM.CLUSTER.HISTORY.QUEUE은(는) 서비스 목적으로 클러스터 상태 정보의 히스토리를 저장하는 데 사용됩니다.

기본 오브젝트 설정에서 SYSTEM.CLUSTER.HISTORY.QUEUE은(는) PUT(ENABLED)으로 설정됩니다. 히스토리 콜렉션을 억제하려면 설정을 PUT(DISABLED)으로 변경하십시오.

SYSTEM.CLUSTER.REPOSITORY.QUEUE
클러스터의 각 큐 관리자에는 SYSTEM.CLUSTER.REPOSITORY.QUEUE(이)라는 로컬 큐가 있습니다. 이 큐는 모든 전체 저장소 정보를 저장하는 데 사용됩니다. 이 큐는 보통 비어있지 않습니다.

SYSTEM.CLUSTER.TRANSMIT.QUEUE
각 큐 관리자에는 로컬 큐 SYSTEM.CLUSTER.TRANSMIT.QUEUE에 대한 정의가 있습니다.
SYSTEM.CLUSTER.TRANSMIT.QUEUE은(는) 클러스터 내에 있는 모든 큐 및 큐 관리자에 대한 모든 메시지의 기본 전송 큐입니다. 큐 관리자 속성 DEFCLXQ 을 변경하여 각 클러스터-송신자 채널의 기본 전송 큐를 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*로 변경할 수 있습니다.
SYSTEM.CLUSTER.TRANSMIT.QUEUE를 삭제할 수 없습니다. 또한 사용되는 기본 전송 큐가

SYSTEM.CLUSTER.TRANSMIT.QUEUE 또는 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*인지 여부를 확인하는 권한 검사를 정의하는 데 사용됩니다.

SYSTEM.DEF.CLUSRCVR
각 클러스터에는 SYSTEM.DEF.CLUSRCVR이라고 하는 기본 CLUSRCVR 채널 정의가 있습니다. SYSTEM.DEF.CLUSRCVR은 클러스터의 큐 관리자에서 클러스터 수신자 채널을 작성할 때 지정하지 않는 속성에 대한 기본값을 제공하기 위해 사용됩니다.

SYSTEM.DEF.CLUSSDR
각 클러스터에는 SYSTEM.DEF.CLUSSDR이라고 하는 기본 CLUSSDR 채널 정의가 있습니다. SYSTEM.DEF.CLUSSDR은 클러스터의 큐 관리자에서 클러스터 송신자 채널을 작성할 때 지정하지 않는 속성에 대한 기본값을 제공하기 위해 사용됩니다.

관련 개념
기본 클러스터 오브젝트에 대한 작업

발행/구독 메시징

발행/구독 메시징을 사용하면 해당 정보의 이용자로부터 정보의 제공자를 분리시킬 수 있습니다. 정보가 송수신되기 위해 송신 애플리케이션과 수신 애플리케이션이 서로에 대해 어느 것도 알 필요가 없습니다.

포인트-투-포인트 IBM MQ 애플리케이션은 메시지를 다른 애플리케이션으로 송신하기 전에 해당 애플리케이션에 대한 일부 정보를 알아야 합니다. 예를 들어 정보를 송신할 큐의 이름을 알아야 하며 큐 관리자 이름을 지정할 수도 있습니다.

IBM MQ 발행/구독에서는 애플리케이션이 대상 애플리케이션을 알지 않아도 됩니다. 모든 송신 애플리케이션은 다음을 수행해야 합니다.

- 애플리케이션이 원하는 정보를 포함하는 IBM MQ 메시지를 넣으십시오 .
- 정보의 주제를 나타내는 메시지를 토픽에 지정합니다.
- IBM MQ에서 해당 정보 분배를 처리하도록 허용합니다.

유사하게 대상 애플리케이션은 수신한 정보의 소스에 대해 알 필요가 없습니다.

다음 그림은 가장 간단한 발행/구독 시스템을 표시합니다. 하나의 발행자, 하나의 큐 관리자 및 하나의 구독자가 있습니다. 큐 관리자의 구독자가 구독을 작성하고, 발행이 발행자로부터 큐 관리자로 송신된 후 발행이 큐 관리자에 의해 구독자에게 전달됩니다.

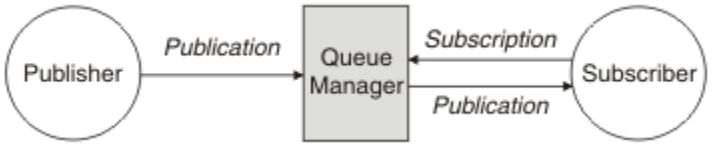


그림 *17.* 간단한 발행/구독 구성

전형적인 발행/구독 시스템에는 많은 다른 토픽에서 둘 이상의 발행자와 둘 이상의 구독자 및 종종 둘 이상의 큐 관리자가 있습니다. 한 애플리케이션이 발행자이면서 구독자일 수 있습니다.

발행/구독 메시징과 포인트-투-포인트 간의 다른 큰 차이점은 포인트-투-포인트 큐로 보내진 메시지는 단일 사용 애플리케이션에 의해서만 처리된다는 점입니다. 발행/구독 토픽에 발행된 메시지는(둘 이상의 구독자가 관심을 등록함) 모든 관심 있는 구독자에 의해 처리됩니다.

발행/구독 컴포넌트

발행/구독은 구독자가 발행자로부터 메시지 양식으로 정보를 수신할 수 있는 메커니즘입니다. 발행자와 구독자 사이의 상호작용은 표준 IBM MQ 기능을 사용하여 큐 관리자에 의해 제어됩니다.

전형적인 발행/구독 시스템에는 많은 다른 토픽에서 둘 이상의 발행자와 둘 이상의 구독자 및 종종 둘 이상의 큐 관리자가 있습니다. 한 애플리케이션이 발행자이면서 구독자일 수 있습니다.

정보 제공자는 발행자라고 합니다. 발행자는 해당 정보에 관심이 있는 애플리케이션에 대해 알지 않고도 주제에 대한 정보를 제공합니다. 발행자는 이 메시지의 토픽을 발행하고 정의하려는 발행이라고 하는 메시지 양식으로 이 정보를 생성합니다.

정보의 이용자는 구독자라고 합니다. 구독자는 구독자가 관심이 있는 토픽을 설명하는 구독을 작성합니다. 따라서 구독은 구독자에게 전달되는 발행을 판별합니다. 구독자는 여러 구독을 작성하고 많은 다양한 발행자로부터 정보를 수신할 수 있습니다.

발행된 정보는 IBM MQ 메시지에서 전송되며, 정보의 주제는 해당 토픽으로 식별됩니다. 발행자는 정보를 발행할 때 토픽을 지정하고 구독자는 발행을 수신하려는 토픽을 지정합니다. 구독자는 구독하는 토픽에 대한 정보만 받습니다.

토픽을 통해 포인트-투-포인트 메시징에 필요한 경우 각 메시지에서 특정 목적지를 포함해야 하는 필요를 없애 정보의 제공자와 이용자를 발행/구독 메시징에서 분리할 수 있습니다.

발행자와 구독자 사이의 상호작용은 큐 관리자에 의해 모두 제어됩니다. 큐 관리자는 발행자로부터 메시지를 수신하고 토픽 범위에 따라 구독자로부터 구독을 수신합니다. 큐 관리자의 작업은 메시지 토픽에서 관심사를 등록한 구독자에게 발행된 메시지를 라우팅하는 것입니다.

표준 IBM MQ 기능은 메시지 분산에 사용되므로 애플리케이션은 기존 IBM MQ 애플리케이션에서 사용 가능한 모든 기능을 사용할 수 있습니다. 즉, 지속 메시지를 사용하여 한 번만 보장된 전달을 받을 수 있으며, 발행자가 커미트한 경우에만 구독자에게 메시지를 전달하도록 메시지를 트랜잭션 작업 단위로 포함할 수 있음을 의미합니다.

발행자 및 발행물

IBM MQ 발행/구독에서 발행자는 발행물이라고 하는 표준 IBM MQ 메시지의 양식으로 큐 관리자가 사용할 수 있는 특정 토픽에 대한 정보를 작성하는 애플리케이션입니다. 발행자는 둘 이상의 토픽에 대한 정보를 발행할 수 있습니다.

발행자는 MQPUT 동사를 사용하여 이전에 열린 토픽에 메시지를 넣으며, 이 메시지는 발행물입니다. 그리고 로컬 큐 관리자는 발행물의 토픽을 구독하는 구독자에게 해당 발행물을 라우팅합니다. 발행된 메시지를 둘 이상의 구독자가 이용할 수 있습니다.

해당 구독을 보유하는 모든 로컬 구독자에게 발행물을 배포함은 물론, 큐 관리자는 이에 연결된 기타 큐 관리자에게 직접적으로 또는 토픽에 대한 구독자가 있는 큐 관리자의 네트워크를 통해 발행물을 배포할 수도 있습니다.

IBM MQ 발행/구독 네트워크에서 발행 애플리케이션은 구독자일 수도 있습니다.

동기점 하의 발행물

발행자는 작업 단위로 구독자에게 전달된 모든 메시지를 포함하도록 동기점에서 MQPUT 또는 MQPUT1 호출을 실행할 수 있습니다. MQPMO_RETAIN 옵션 또는 토픽 전달 옵션 NPMSGDLV 또는 PMSGDLV(ALL 또는 ALLDUR 값의)가 지정되면, 큐 관리자는 발행자 MQPUT 또는 MQPUT1 호출 범위 내의 동기점에서 내부 MQPUT 또는 MQPUT1 호출을 사용합니다.

상태 및 이벤트 정보

발행은 주식의 현재 가격과 같은 상태 발행이나, 해당 주식의 거래와 같은 이벤트 발행으로 분류할 수 있습니다.

상태 발행

상태 발행은 축구 경기의 현재 점수 또는 주가와 같이 어떤 항목의 현재 상태에 대한 정보를 포함합니다. 주가가 변동되거나 축구 경기의 점수가 변경되는 것과 같이 무슨 일이 발생하면 이전 상태 정보가 새 정보로 대체되므로 더 이상 필요하지 않게 됩니다.

구독자는 시작할 때 상태 정보의 최신 버전을 수신하려고 하며, 상태가 변경되면 새 정보가 전송되기를 원합니다.

발행에 상태 정보가 포함되는 경우, 흔히 보유된 발행으로 발행됩니다. 일반적으로 새 구독자는 즉시 현재 상태 정보를 원하며, 구독자는 정보가 다시 발행되는 이벤트를 기다리기를 원하지 않습니다. 구독자가 MQSO_PUBLICATIONS_ON_REQUEST 또는 MQSO_NEW_PUBLICATIONS_ONLY 옵션을 사용하지 않는 한, 구독할 때 구독자는 자동으로 토픽의 보유된 발행을 수신합니다.

이벤트 발행

이벤트 발행은 일부 주식의 거래나 특정 골의 득점과 같이 발생하는 개별 이벤트에 대한 정보를 포함합니다. 각 이벤트는 다른 이벤트와 서로 독립적입니다.

구독자는 이벤트가 발생하면 해당 이벤트에 대한 정보를 수신하려고 합니다.

보유된 발행물

기본적으로 발행은 관련된 모든 구독자에게 전송된 후에 제거됩니다. 그러나 발행자는 발행 사본을 보유하도록 지정할 수 있습니다. 그러면 토픽에 관심사를 등록한 향후 구독자에게 이를 전송할 수 있습니다.

이벤트 정보의 경우 관련된 모든 구독자에게 전송된 후에 발행을 삭제하는 것이 적절하지만, 상태 정보에서는 항상 적절하지 않을 수도 있습니다. 메시지를 보유하면 새 구독자가 초기 상태 정보를 수신하기 전에 정보 발행을 다시 기다리지 않아도 됩니다. 예를 들어 주가에 대해 구독하는 구독자는 주가가 변경되고 이후에 다시 발행되기를 기다리지 않고도 현재 가격을 직접 수신할 수 있습니다.

큐 관리자는 각 토픽에 대해 하나의 발행만 보유할 수 있으므로 새 보유된 발행이 큐 관리자에 도달하면 토픽의 기존 보유된 발행은 삭제됩니다. 그러나 기존 발행의 삭제가 새 보유된 발행의 도착과 동시에 수행되지 않을 수도 있습니다. 따라서 토픽에서 보유된 발행을 송신하는 발행자가 하나만 존재할 수 있습니다.

구독자는 MQSO_NEW_PUBLICATIONS_ONLY 구독 옵션을 사용하여 보유된 발행을 수신하지 않도록 지정할 수 있습니다. 기존 구독자는 보유된 발행의 중복 사본을 전송해줄 것을 요청할 수 있습니다.

상태 정보라도 발행을 보유하지 않으려는 경우가 있습니다.

- 토픽에 대한 모든 구독이 해당 토픽에 대한 발행을 수행하기 전에 이루어진 경우 새 구독을 예상하지 않았거나 이를 허용하지 않으면 처음 발행될 때 전체 구독자 세트에 전달되므로 발행을 보유하지 않아도 됩니다.
- 발행이 자주(가령 매초마다) 나타나면 새 구독자 또는 장애에서 복구한 구독자는 초기 구독 이후 거의 즉시 현재 상태를 수신하므로 이러한 발행을 보유하지 않아도 됩니다.
- 발행이 큰 경우 각 토픽에 대한 보유된 발행을 저장하는 데 상당한 스토리지 공간이 필요하게 될 수도 있습니다. 다중 큐 관리자 환경에서 보유된 발행은 일치하는 구독이 있는 네트워크의 모든 관리자에서 저장합니다.

보유된 발행의 사용 여부를 결정할 때 구독 애플리케이션이 실패에서 복구하는 방법을 고려하십시오. 발행자가 보유된 발행을 사용하지 않을 경우, 구독자 애플리케이션은 현재 상태를 로컬로 저장해야 할 수도 있습니다.

발행을 보유하려면 MQPMO_RETAIN 메시지 넣기 옵션을 사용하십시오. 이 옵션이 사용되고 발행물을 보유할 수 없는 경우에는 메시지가 발행되지 않고 호출은 MQRC_PUT_NOT_RETAINED가 발생하여 실패합니다.

메시지가 보유된 발행이면 MQIsRetained 메시지 특성으로 표시됩니다. 메시지의 지속성은 원래 발행될 때 속성 그대로입니다.

관련 개념

발행/구독 클러스터에서 보유된 발행에 대한 디자인 고려사항

동기점 하의 발행물

IBM MQ 발행/구독에서 동기점은 발행자가 사용하거나 큐 관리자가 내부적으로 사용할 수 있습니다.

발행자는 MQPMO_SYNCPOINT 옵션을 사용하여 MQPUT/MQPUT1 호출을 발행할 때 동기점을 사용합니다. 구독자에 전달된 모든 메시지는 작업 단위에서 커미트되지 않은 최대 메시지 수로 계산됩니다. MAXUMSGS 큐 관리자 속성이 이 한계를 지정합니다. 한계에 도달하면 발행자가 2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_REACHED 이유 코드를 수신합니다.

발행자가 MQPMO_RETAIN 옵션 또는 값이 ALL이나 ALLDUR인 토픽 전달 옵션 NPMSGDLV/PMSGDLV와 함께 MQPMO_NO_SYNCPOINT를 사용하여 MQPUT/MQPUT1 호출을 발행하는 경우 큐 관리자는 메시지가 요청된 대로 전달될 수 있도록 내부 동기점을 사용합니다. 한계가 발행자 MQPUT/MQPUT1 호출의 범위에 도달하면 발행자가 2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_REACHED 이유 코드를 수신할 수 있습니다.

구독자 및 구독

IBM MQ 발행/구독에서 구독자는 발행/구독 네트워크에 있는 큐 관리자로부터 특정 토픽에 대한 정보를 요청하는 애플리케이션입니다. 구독자는 둘 이상의 발행자로부터 동일하거나 다른 토픽에 대한 메시지를 수신할 수 있습니다.

구독은 애플리케이션 또는 MQSC 명령을 사용하여 수동으로 작성할 수 있습니다. 이러한 구독은 로컬 큐 관리자로 발행되고, 구독자가 수신하려는 발행에 대한 정보를 포함합니다.

- 구독자가 관심이 있는 토픽. 이는 와일드카드를 사용하면 여러 토픽으로 해석될 수 있습니다.
- 발행된 메시지에 적용할 선택적 선택 문자열.
- 선택한 발행을 배치해야 하는 큐(구독자 큐라고 함)에 대한 핸들 및 선택적 CorrelId.

로컬 큐 관리자는 구독 정보를 저장하고 발행을 수신하면 구독의 토픽 및 선택 문자열과 일치하는 구독이 있는지 여부를 판별하도록 정보를 스캔합니다. 일치하는 각 구독에 대해 큐 관리자는 구독자의 구독자 큐로 발행을 지정합니다. 큐 관리자가 구독에 대해 저장하는 정보는 DIS SUB 및 DIS SBSTATUS 명령을 사용하여 볼 수 있습니다.

다음과 같은 이벤트 중 하나가 발생하는 경우에만 구독이 삭제됩니다.

- 구독자는 MQCLOSE 호출을 사용하여 구독을 해제합니다(구독이 비지속으로 작성된 경우).
- 구독이 만료됩니다.
- 구독은 시스템 관리자가 DELETE SUB 명령을 사용하여 삭제합니다.
- 구독자 애플리케이션이 종료됩니다(구독이 비지속으로 작성된 경우).
- 큐 관리자가 중지되거나 재시작됩니다(구독이 비지속으로 작성된 경우).

메시지를 가져올 때 MQGET 호출에서 적절한 옵션을 사용하십시오. 애플리케이션이 한 구독의 메시지만 처리하는 경우에는 최소한 `get-by-correlid`를 C 샘플 프로그램 `amqssbxa.c` 및 비관리 MQ 구독자에 시연되어 있는 바와 같이 사용해야 합니다. 사용할 **CorrelId** 는 MQSD의 MQSUB에서 리턴됩니다.**SubCorrelId** 필드.

관련 개념

복제 및 공유된 구독

관련 참조

sharedSubscription 특성을 정의하는 방법의 예

관리 큐 및 발행/구독

구독을 작성하면 관리 큐잉을 사용하도록 선택할 수 있습니다. 관리 큐잉을 사용하는 경우 구독을 작성할 때 구독 큐는 자동으로 작성됩니다. 관리 큐는 구독의 지속성에 따라 자동으로 정리됩니다. 관리 큐의 사용은 발행을 수신하는 큐를 작성하지 않아도 되며, 지속 불가능 구독 연결이 닫히면 이용하지 않은 발행은 구독자 큐에서 자동으로 제거됨을 의미합니다.

애플리케이션이 구독자 큐로 특정 큐를 사용하지 않아도 되면 이를 수신하는 발행의 목적지는 MQSO_MANAGED 구독 옵션을 사용하여 관리 구독을 사용할 수 있습니다. 관리 구독을 작성하면 큐 관리자는 발행이 수신되는 큐 관리자가 작성한 구독자 큐의 구독자에게 오브젝트 핸들을 리턴합니다. 이는 관리 구독이 IBM MQ에서 구독을 처리하는 구독이기 때문입니다. 큐 오브젝트 핸들을 리턴하여 큐를 찾아보거나 가져오거나 조회할 수 있습니다(임시 동적 큐에 대한 액세스 권한을 명시적으로 부여받지 않는 한, 관리 큐의 속성을 설정하거나 여기에 배치할 수 없음).

구독의 지속성은 구독 애플리케이션에서 큐 관리자로의 연결이 끊어진 경우 관리 큐가 남아 있는지 여부를 판별합니다.

관리 구독은 애플리케이션 연결이 종료될 때 (그렇지 않은 경우) 이용되지 않은 메시지가 구독 큐에 계속 남아 있어서 큐 관리자에서 무한으로 공간을 차지하므로 지속 불가능 구독과 함께 사용할 때 특히 유용합니다. 관리 구독을 사용하는 경우 관리 큐는 임시 동적 큐이며, 다음과 같은 이유로 연결이 끊어질 때 이용되지 않은 메시지와 함께 삭제됩니다.

- MQCO_REMOVE_SUB를 포함하는 MQCLOSE가 사용되고 관리 Hobj가 닫힙니다.
- 지속 불가능 구독(MQSO_NON_DURABLE)을 사용하는 애플리케이션과의 연결이 끊어집니다.
- 구독이 만료되었고 관리 Hobj가 닫혔으므로 구독이 제거됩니다.

관리 구독을 지속 가능 구독과 함께 사용할 수도 있지만 연결을 다시 열 때 검색할 수 있도록 이용되지 않은 메시지를 구독자 큐에 남겨둘 수 있습니다. 이러한 이유로 지속 가능 구독에 대한 관리 큐는 영구적 동적 큐 양식을 취하고 구독 애플리케이션과 큐 관리자의 연결이 끊어진 경우에도 그대로 남아 있습니다.

연결이 끊어진 후에도 큐가 그대로 존재하도록 영구적 동적 관리 큐를 사용하려는 경우 구독에서 만료를 설정할 수 있으며, 무한으로 계속 존재하지 않습니다.

관리 큐를 삭제하면 오류 메시지가 수신됩니다.

작성된 관리 큐의 이름은 각각 고유하도록 끝에 숫자(시간 소인)가 추가됩니다.

구독 지속성

구독은 지속 가능 또는 지속 불가능으로 구성될 수 있습니다. 구독 지속성은 구독 애플리케이션과 큐 관리자의 연결이 끊어질 때 구독에서 나타나는 상황을 판별합니다.

지속 가능 구독

지속 가능 구독의 경우 구독 애플리케이션과 큐 관리자의 연결이 닫힌 경우에도 계속 존재합니다. 구독이 지속 가능하면 구독 애플리케이션의 연결이 끊어질 때 구독은 그대로 남아 있으며, 구독을 작성할 때 리턴된 **SubName**을 사용하여 다시 구독을 요청하며 다시 연결할 때 구독 애플리케이션에서 이를 사용할 수 있습니다.

지속적으로 구독하는 경우 구독 이름(**SubName**)이 필요합니다. 구독을 식별하는 데 사용할 수 있도록 구독 이름은 큐 관리자에서 고유해야 합니다. 이러한 식별의 방법은 큐 관리자에서 연결이 끊어졌거나 MQCO_KEEP_SUB 옵션을 사용하여 구독에 대한 연결을 고의적으로 닫은 경우 재개하려는 구독을 지정할 때 필요합니다. MQSO_RESUME 옵션과 함께 MQSUB 호출을 사용하여 기존 구독을 계속할 수 있습니다. SUBTYPE ALL 또는 ADMIN과 함께 DISPLAY SBSTATUS 명령을 사용하는 경우에도 subscription 이름이 표시됩니다.

애플리케이션에 지속 가능 구독이 더 이상 필요하지 않으면 MQCO_REMOVE_SUB 옵션과 함께 MQCLOSE 함수를 사용하여 제거할 수 있거나 수동으로 MQSC 명령 DELETE SUB를 사용하여 삭제할 수 있습니다.

DURSUB 토픽 속성을 사용하여 토픽에 대해 지속 가능 구독을 작성할 수 있는지 여부를 지정할 수 있습니다.

MQSO_RESUME 옵션을 사용하여 MQSUB 호출에서 리턴할 때 구독 만료는 남은 만료 시간이 아닌, 원래 구독의 만료로 설정됩니다.

큐 관리자는 계속해서 발행을 송신하여 구독자 애플리케이션이 연결되지 않아도 지속 가능 구독을 만족시킬 수 있습니다. 이로 인해 구독자 큐에 메시지가 누적될 수 있습니다. 이 문제점을 피하는 가장 쉬운 방법은 해당되는 경우에만 지속 불가능 구독을 사용하는 것입니다. 그러나 지속 가능 구독을 사용해야 하는 경우 구독자가 보유된 발행 옵션을 사용하여 구독하면 메시지 누적을 방지할 수 있습니다. 그러면 구독자는 MQSUBRQ 호출을 사용하여 발행을 수신하는 시기를 제어할 수 있습니다.

지속 불가능 구독

지속 불가능 구독은 구독 애플리케이션과 큐 관리자의 연결이 열려 있는 경우에만 존재합니다. 이 구독은 고의적으로 또는 연결 유실에 의해 구독 애플리케이션과 큐 관리자의 연결이 끊길 때 제거됩니다. 연결이 닫히면 구독에 대한 정보가 큐 관리자에서 제거되고 DISPLAY SBSTATUS 명령을 사용하여 구독을 표시하는 경우 더 이상 표시되지 않습니다. 더 이상 구독자 큐에 메시지를 넣지 않습니다.

지속 불가능 구독에 대한 구독자 큐에서 이용되지 않은 발행의 처리 방법은 다음과 같이 판별됩니다.

- 구독 애플리케이션이 관리 목적지를 사용하는 경우 이용되지 않은 발행물이 자동으로 제거됩니다.
- 구독 애플리케이션이 구독 시 고유한 구독자 큐로 핸들을 제공하는 경우 이용되지 않은 메시지는 자동으로 제거되지 않습니다. 적절한 경우 큐를 지우는 것은 애플리케이션의 몫입니다. 큐가 둘 이상의 구독자 또는 다른 포인트-투-포인트 애플리케이션에서 공유되는 경우 큐를 완전히 지우는 것은 적절하지 않을 수도 있습니다.

지속 불가능 구독에 필요하지는 않지만 제공되는 경우 구독 이름은 큐 관리자에서 사용됩니다. 구독을 식별하는 데 사용할 수 있도록 구독 이름은 큐 관리자에서 고유해야 합니다.

관련 개념

복제 및 공유된 구독

관련 태스크

JMS 2.0 공유 구독 사용

관련 참조

sharedSubscription 특성을 정의하는 방법의 예

선택 문자열

선택 문자열은 구독과 일치하는지 여부를 판별하기 위해 발행에 적용되는 표현식입니다. 선택 문자열은 와일드카드 문자를 포함할 수 있습니다.

구독하는 경우 토픽 지정 외에도 선택 문자열을 지정하여 메시지 특성에 따라 발행을 선택할 수 있습니다.

선택 문자열은 각 구독자에 전달하도록 수정하기 전에 발행자가 입력한 메시지에 대해 평가됩니다. 선택 문자열에서 발행 조작의 일부로 수정될 수 있는 필드를 사용할 때는 주의하십시오. 예를 들어 MQMD 필드 `UserIdentifier`, `MsgId`, `CorrelId`가 이에 해당합니다.

선택 문자열은 발행 조작의 일부로 큐 관리자가 추가한 메시지 특성 필드를 참조하지 않습니다(발행/구독 메시지 특성 참조). 단, 발행을 위해 토픽 문자열을 포함하는 메시지 특성 `MQTopicString`은 예외입니다.

관련 개념

선택 문자열 규칙 및 제한사항

토픽

토픽은 발행/구독 메시지에서 발행된 정보의 제목입니다.

포인트-투-포인트 시스템의 메시지는 특정 목적지 주소로 송신됩니다. 주제 기반 발행/구독 시스템의 메시지는 메시지의 컨텐츠에 대해 설명하는 주제에 따른 구독자로 송신됩니다. 컨텐츠 기반 시스템에서, 메시지는 메시지의 컨텐츠를 기반으로 한 구독자로 송신됩니다.

IBM MQ 발행/구독 시스템은 주제 기반 발행/구독 시스템입니다. 발행자는 메시지를 작성하고 발행 주제에 맞는 토픽 문자열을 사용하여 해당 메시지를 발행합니다. 구독자는 발행을 수신하기 위해 발행 토픽을 선택하도록 토픽 문자열과 일치하는 패턴의 구독을 작성합니다. 큐 관리자는 발행 토픽과 일치하는 구독을 가진 구독자에게 발행을 전달하며 이 큐 관리자에게 발행을 수신할 수 있는 권한이 부여됩니다. 61 페이지의 『토픽 문자열』 글에서는 발행 주제를 식별하는 토픽 문자열의 구문에 대해 설명합니다. 또한 구독자는 수신할 토픽을 선택하도록 토픽 문자열을 작성합니다. 구독자가 작성하는 토픽 문자열에는 발행에 있는 토픽 문자열의 패턴 일치에 대한 두 가지 대체 와일드카드 설계 중 하나가 있습니다. 패턴 일치는 61 페이지의 『와일드카드 설계』에 설명되어 있습니다.

주제 기반 발행/구독에서는, 발행자 또는 관리자가 주제를 토픽으로 분류해야 합니다. 일반적으로 주제는 '/' 문자로 토픽 문자열에서 하위 토픽을 작성하여 계층적으로 토픽 트리로 구성됩니다. 토픽 트리 예는 67 페이지의 『토픽 트리』의 내용을 참조하십시오. 토픽은 토픽 트리의 노드입니다. 토픽은 추가 하위 토픽이 없는 리프 노드 또는 하위 토픽이 있는 중간 노드입니다.

주제를 계층적 토픽 트리로 구성하면서, 토픽을 관리 토픽 오브젝트와 연관시킬 수 있습니다. 토픽을 관리 토픽 오브젝트와 연관시켜 토픽을 클러스터에 분배할지의 여부와 같은 속성을 토픽에 지정합니다. 관리 토픽 오브젝트의 `TOPICSTR` 속성을 사용하여 토픽의 이름을 지정하면 연관이 이루어집니다. 명시적으로 관리 토픽 오브젝트를 토픽에 연관시키지 않으면, 토픽이 관리 토픽 오브젝트와 연관시킨 토픽 트리의 가장 가까운 상위의 속성을 상속합니다. 어떤 상위 토픽도 정의하지 않은 경우, `SYSTEM.BASE.TOPIC`에서 상속합니다. 관리 토픽 오브젝트는 68 페이지의 『관리 토픽 오브젝트』에 설명되어 있습니다.

참고: `SYSTEM.BASE.TOPIC`에서 토픽의 속성을 모두 상속하는 경우에도, `SYSTEM.BASE.TOPIC`에서 직접 상속하는 토픽의 루트 토픽을 정의하십시오. 예를 들어, 토픽 공간인 미국의 주 `USA/Alabama`, `USA/Alaska` 등에서는 `USA`가 루트 토픽입니다. 루트 토픽은 주로 잘못된 구독과 발행이 일치하는 것을 피하기 위해 겹치지 않는 개별 토픽 공간을 작성하는 데 사용됩니다. 또한 이 루트 토픽을 사용하면 전체 토픽 공간에 영향을 미치도록 루트 토픽의 속성을 변경할 수 있습니다. 예를 들어, **CLUSTER** 속성에 이름을 설정할 수 있습니다.

발행자 또는 구독자로 토픽을 참조하는 경우, 토픽 문자열을 제공하거나 토픽 오브젝트를 참조할 수 있습니다. 또한 제공하는 토픽 문자열에서 토픽 오브젝트의 하위 토픽을 정의하는 경우에는 둘 다 수행할 수 있습니다. 큐 관리자는 토픽 오브젝트에서 이름 지정된 토픽 문자열 접두부에 토픽 문자열을 추가하고 두 토픽 문자열 사이에 '/'를 삽입하여 토픽을 식별합니다(예: *topic string/object string*). 65 페이지의 『토픽 문자열 결합』에서는 이에 대해 자세히 설명합니다. 이렇게 생성된 토픽 문자열은 토픽을 식별하고 이 토픽을 관리 토픽 오브젝트와 연관시키는 데 사용됩니다. 관리 토픽 오브젝트는 마스터 토픽에 해당하는 토픽 오브젝트와 다를 수도 있습니다.

컨텐츠 기반 발행/구독에서는, 모든 메시지의 컨텐츠를 검색하는 선택 문자열을 제공하여 수신할 메시지 내용을 정의합니다. IBM MQ에서는 메시지의 전체 컨텐츠와 다른 메시지 특성을 스캔하는 메시지 선택자를 사용하여 컨텐츠 기반 발행/구독의 중간 형식을 제공합니다. 선택기를 참조하십시오. 메시지 선택자는 전형적으로 토픽을 구독한 다음 숫자 특성에 대한 선택 사항을 규정하는 데 사용됩니다. 선택자를 사용하면 특정 범위의 값에만(문자 또는 토픽 기반 와일드카드를 사용하여 수행할 수 없는 내용) 관심이 있음을 지정할 수 있습니다. 메시지의 전체 컨텐츠를 기반으로 하여 필터링해야 하는 경우, IBM Integration Bus를 사용해야 합니다.

토픽 문자열

토픽 문자열을 사용하여 토픽으로 발행하는 레이블 정보입니다. 문자 또는 토픽 기반 와일드카드 토픽 문자열을 사용하여 토픽 그룹을 구독합니다.

토픽

토픽 문자열은 발행/구독 메시지의 토픽을 식별하는 문자열입니다. 토픽 문자열을 구성할 때 원하는 모든 문자를 사용할 수 있습니다.

토픽 문자열

3자는 IBM WebSphere® MQ 7 발행/구독에서 특별한 의미를 지닙니다. 이는 토픽 문자열에서 허용되지만, 주의해서 사용해야 합니다. 특수 문자의 사용은 62 페이지의 『토픽 기반 와일드카드 설계』에서 설명됩니다.

슬래시(/)

토픽 레벨 구분 기호입니다. '/' 문자를 사용하여 토픽 트리로 토픽을 구성합니다.

가능한 경우 빈 토픽 레벨('//')은 피하십시오. 이는 토픽 문자열이 없는 토픽 계층의 노드에 대응합니다. 토픽 문자열에서 선두 또는 후미 문자 '/'는 선두 또는 후미의 빈 노드에 대응하므로 이러한 사용도 피해야 합니다.

해시 기호(#)

'/'와 함께 사용하여 구독에서 다중 레벨 와일드카드를 구성합니다. 발행된 토픽의 이름을 지정하는 데 사용되는 토픽 문자열에서 '/' 가까이에서 '#'를 사용할 때 주의하십시오. 61 페이지의 『토픽 문자열 예제』에서는 '#'의 합리적인 사용법을 보여줍니다.

'.../#/...', '#/...', '.../#' 문자열에는 구독 토픽 문자열에서 특별한 의미를 지닙니다. 문자열은 토픽 계층에서 하나 이상의 레벨에 있는 모든 토픽과 일치합니다. 따라서 이러한 순서 중 하나로 토픽을 작성한 경우 토픽 계층의 다중 레벨에서 모든 토픽을 구독하지 않는 한, 이를 구독할 수 없습니다.

더하기 기호(+)

'/'와 함께 사용하여 구독에서 단일 레벨 와일드카드를 구성합니다. 발행된 토픽의 이름을 지정하는 데 사용되는 토픽 문자열에서 '/' 가까이에서 '+'를 사용할 때 주의하십시오.

'.../+/...', '+/...', '.../+' 문자열에는 구독 토픽 문자열에서 특별한 의미를 지닙니다. 문자열은 토픽 계층에서 한 레벨에 있는 모든 토픽과 일치합니다. 따라서 이러한 순서 중 하나로 토픽을 작성한 경우 토픽 계층의 한 레벨에서 모든 토픽을 구독하지 않는 한, 이를 구독할 수 없습니다.

토픽 문자열 예제

```
IBM/Business Area#/Results
IBM/Diversity/%African American
```

관련 참조

TOPIC

와일드카드 설계

여러 토픽을 구독하는 데 사용하는 2개의 와일드카드 설계가 있습니다. 설계 선택은 구독 옵션입니다.

MQSO_WILDCARD_TOPIC

토픽 기반 와일드카드 설계를 사용하여 구독할 토픽을 선택합니다.

이는 와일드카드 스키마를 명확하게 선택하지 않은 경우 기본값입니다.

MQSO_WILDCARD_CHAR

문자 기반 와일드카드 설계를 사용하여 구독할 토픽을 선택합니다.

DEFINE SUB 명령에서 **wschema** 매개변수를 지정하여 설계를 설정합니다. 자세한 정보는 DEFINE SUB를 참조하십시오.

참고: IBM WebSphere MQ 7.0 이전에 작성된 구독은 항상 문자 기반 와일드카드 설계를 사용합니다.

예

```
IBM/+/Results
#/Results
IBM/Software/Results
IBM/*ware/Results
```

토픽 기반 와일드카드 설계
토픽 기반 와일드카드를 사용하면 구독자가 한 번에 둘 이상의 토픽을 구독할 수 있습니다.

토픽 기반 와일드카드는 IBM MQ 발행/구독에서 토픽 시스템의 강력한 기능입니다. 다중 레벨 와일드카드와 단일 레벨 와일드카드는 구독에 사용될 수 있으나, 메시지 발행자가 토픽에서 사용할 수는 없습니다.

토픽 기반 와일드카드 설계에서는 토픽 레벨로 그룹화된 발행을 선택할 수 있습니다. 토픽 계층구조의 각 레벨마다 해당 토픽 레벨에 대한 구독의 문자열이 발행의 문자열과 정확히 일치해야 하는지 여부를 선택할 수 있습니다. 예를 들어, `IBM/+/Results` 구독은 모든 토픽을 선택합니다.

```
IBM/Software/Results
IBM/Services/Results
IBM/Hardware/Results
```

두 종류의 와일드카드가 있습니다.

다중 레벨 와일드카드

- 다중 레벨 와일드카드는 구독에서 사용됩니다. 발행에서 사용되면 리터럴로 처리됩니다.
- 다중 레벨 와일드카드 문자 `'#'`는 토픽 내 많은 레벨과 일치시키는 데 사용됩니다. 예를 들어 토픽 트리 예제를 사용하면 `'USA/Alaska/#'`를 구독할 때 토픽 `'USA/Alaska'` 및 `'USA/Alaska/Juneau'`에서 메시지를 수신합니다.
- 다중 레벨 와일드카드는 0개 이상의 레벨을 표시할 수 있습니다. 따라서 `'USA/#'`는 단일 `'USA'`와도 일치할 수 있습니다. 여기서 `'#'` 기호는 0 레벨을 나타냅니다. 토픽 레벨 구분 기호는 분리할 레벨이 없기 때문에 이 컨텍스트에서는 의미가 없습니다.
- 다중 레벨 와일드카드는 자체에 또는 토픽 레벨 분리 문자 옆에 지정된 경우에만 유효합니다. 따라서 `'#'` 및 `'USA/#'`은 `'#'` 문자가 와일드카드로 처리되는 유효한 토픽입니다. `'USA#'`도 유효한 토픽 문자열이지만 `'#'` 문자는 와일드카드로 간주되지 않으며, 특별한 의미를 지니지 않습니다. 64 페이지의 『토픽 기반 와일드카드가 와일드카드가 아닌 경우』의 내용을 참조하십시오.

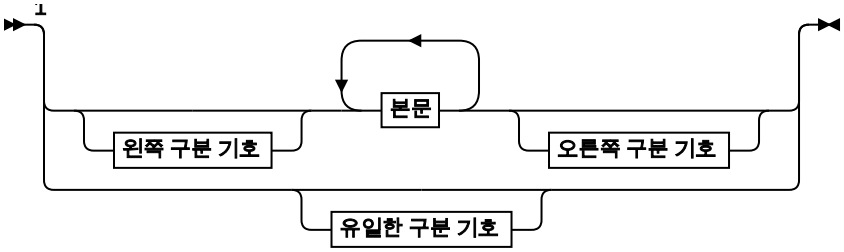
단일 레벨 와일드카드

- 단일 와일드카드는 구독에서 사용됩니다. 발행에서 사용되면 리터럴로 처리됩니다.
- 단일 레벨 와일드카드 문자 `'+'`는 하나의 유일한 토픽 레벨과 일치합니다. 예를 들어 `'USA/+'`는 `'USA/Alabama/Auburn'`이 아닌 `'USA/Alabama'`와 일치합니다. 단일 레벨 와일드카드는 단일 레벨과만 일치하므로 `'USA/+'`는 `'USA'`와 일치하지 않습니다.
- 단일 레벨 와일드카드는 토픽 트리의 레벨에서 그리고 다중 레벨 와일드카드와 연결하여 사용될 수 있습니다. 단일 레벨 와일드카드는 자체에 지정된 경우를 제외하고 토픽 레벨 분리 문자 옆에 지정해야 합니다. 따라서 `'+'` 및 `'USA/+'`은 `'+'` 문자가 와일드카드로 처리되는 유효한 토픽입니다. `'USA+'`도 유효한 토픽 문자열이지만 `'+'` 문자는 와일드카드로 간주되지 않으며, 특별한 의미를 지니지 않습니다. 64 페이지의 『토픽 기반 와일드카드가 와일드카드가 아닌 경우』의 내용을 참조하십시오.

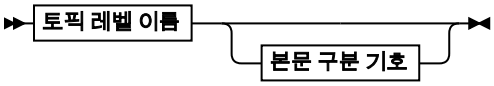
토픽 기반 와일드카드 설계의 구문에는 이스케이프 문자가 없습니다. `'#'` 및 `'+'`가 와일드카드로 처리되는지 여부는 컨텍스트에 따라 달라집니다. 자세한 정보는 64 페이지의 『토픽 기반 와일드카드가 와일드카드가 아닌 경우』의 내용을 참조하십시오.

참고: 토픽 문자열의 시작과 끝은 특별한 방식으로 처리됩니다. `''$''`을 사용하여 문자열의 끝을 나타내고, `''$#/...''`는 다단계 와일드카드, `''$/#/..''`은 . 루트의 빈 노드이고 다음에 다중 레벨 와일드카드가 나옵니다.

토픽 기반 와일드카드 문자열

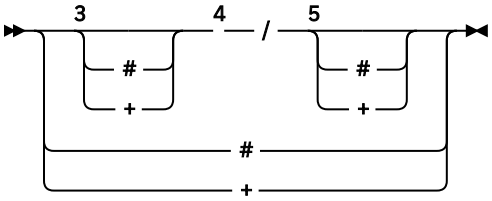


본문



토픽 레벨 이름

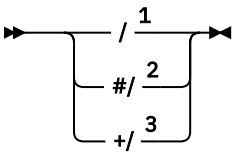
유일한 구분 기호



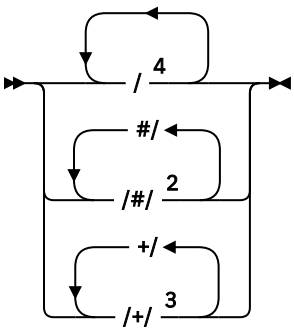
참고:

[1] 널 또는 길이가 0인 토픽 문자열은 유효하지 않음
[2] 문자 기반 와일드카드 체계와 주제 기반 와일드카드 체계 간의 호환성을 위해 레벨 이름 문자열에 '*, ?, %'을 사용하지 않는 것이 좋습니다.
[3] 이 경우는 왼쪽 구분 기호 패턴과 동일합니다.
[4] 와일드카드가 없는 /는 단일 빈 토픽과 일치합니다.
[5] 이 경우는 오른쪽 구분 기호 패턴과 동일합니다.
[6] 모든 토픽과 일치합니다.
[7] 1개 레벨만 존재하는 모든 토픽과 일치합니다.

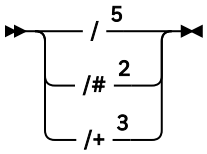
왼쪽 구분 기호



본문 구분 기호



오른쪽 구분 기호



참고:

[1] 토픽 문자열이 빈 토픽으로 시작함
[2] 0개 이상의 레벨과 일치합니다. 여러 개의 다중 레벨 일치 문자열은 하나의 다중 레벨 일치 문자열과 효과는 동일합니다.
[3] 정확히 하나의 레벨과 일치합니다.
[4] //는 빈 토픽(토픽 문자열이 없는 토픽 오브젝트)입니다.
[5] 토픽 문자열이 빈 토픽으로 끝남

토픽 기반 와일드카드가 와일드카드가 아닌 경우

와일드카드 문자 '+' 및 '#'은 토픽 레벨에서 해당 문자를 포함하여 다른 문자와 함께 혼합된 경우 특별한 의미는 없습니다.

즉 토픽 레벨에서 다른 문자와 함께 '+' 또는 '#'을 포함하는 토픽을 발행할 수 있음을 의미합니다.

예를 들어 다음 두 개의 토픽을 고려하십시오.

1. level0/level1/+/level4/#
2. level0/level1/#+/level4/level#

첫 번째 예제에서 '+' 및 '#' 문자는 와일드카드로 처리되므로 발행하려는 토픽 문자열에서 유효하지 않지만, 구독에서 유효합니다.

두 번째 예제에서 '+' 및 '#' 문자는 와일드카드로 처리되지 않으므로 토픽 문자열은 발행 및 구독 대상 모두일 수 있습니다.

예

```
IBM/+/Results
#/Results
IBM/Software/Results
```

문자 기반 와일드카드 설계
문자 기반 와일드카드 설계에서는 기존 문자 일치 방식에 기반하여 토픽을 선택할 수 있습니다.

문자열 '*'를 사용하여 토픽 계층에서 여러 레벨에 있는 모든 토픽을 선택할 수 있습니다. 문자 기반 와일드카드 설계에서 '*'를 사용하는 것은 토픽 기반 와일드카드 문자열 '#'를 사용하는 것과 같습니다.

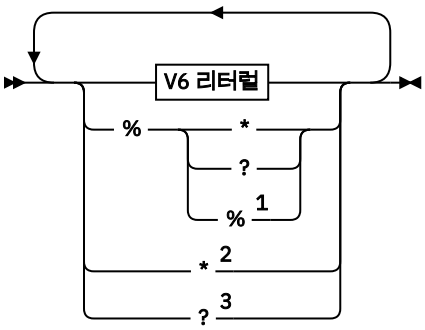
' $x/*/y$ ''은 토픽 기반 체계에서'' $x/\#/y$ ''에 해당하며,'' x'과' y ''레벨 사이의 토픽 계층 구조에서 모든 토픽을 선택합니다. 여기서'' x ''와'' y ''은 와일드카드가 반환한 레벨 집합에 없는 토픽 이름입니다.

문자 기반 설계에는 토픽 기반 설계의 '/+/'와 정확히 동일한 항목이 없습니다. 'IBM/*/Results'가 'IBM/Patents/Software/Results'를 선택할 수도 있습니다. 계층의 각 레벨에서 토픽 이름 세트가 고유한 경우에만 동일한 일치를 생성하는 2개 설계를 포함하는 쿼리를 항상 구성할 수 있습니다.

일반적으로 사용되는 문자 기반 설계에서 '*' 및 '?'는 토픽 기반 설계에서 동일한 기능이 없습니다. 토픽 기반 설계는 와일드카드를 사용하는 부분 일치를 수행하지 않습니다. 문자 기반 와일드카드 구독 'IBM/*ware/Results'에는 이와 동등한 토픽 기반 항목이 없습니다.

참고: 문자 와일드카드 구독을 사용하는 일치는 토픽 기반 구독을 사용하는 일치보다 느립니다.

문자 기반 와일드카드 문자열



V6 리터럴

**,? 및%를 제외한 모든 유니코드 문자*

참고:

[1] 즉, 다음 문자를 이스케이프 처리하므로 리터럴로 처리됩니다. '%''뒤에는''*','''?''또는''%''가 와야 합니다. 61 페이지의 『토픽 문자열 예제』를 참조하십시오.
[2] 즉, 구독에서 0개 이상의 문자와 일치합니다.
[3] 즉, 구독에서 1개 문자와 정확히 일치합니다.

예

```
IBM/*/Results
IBM/*ware/Results
```

토픽 문자열 결합

메시지를 구독할 수 있도록 구독을 작성하거나 토픽을 열 경우, 토픽 문자열이 두 개의 별도 하위 토픽 문자열 또는 "subtopics"를 결합하여 형식화될 수 있습니다. 한 하위 토픽은 애플리케이션 또는 관리 명령에 의해 토픽 문자열로 제공되며, 다른 하위 토픽은 토픽 오브젝트와 연관된 토픽 문자열입니다. 하위 토픽을 자체에서 토픽 문자열로 사용하거나 새 토픽 이름을 형성하도록 둘을 결합할 수 있습니다.

예를 들어, MQSC 명령 **DEFINE SUB**를 사용하여 구독을 정의할 경우 이 명령은 **TOPICSTR**(토픽 문자열) 또는 **TOPICOBJ**(토픽 오브젝트) 중 하나 또는 둘 다를 속성으로 사용할 수 있습니다. **TOPICOBJ**만 제공되면 해당 토픽 오브젝트와 연관된 토픽 문자열이 토픽 문자열로 사용됩니다. **TOPICSTR**만 제공되면 이것이 토픽 문자열로 사용됩니다. 둘 다 제공되면 **TOPICOBJ** / **TOPICSTR** 형식으로 단일 토픽 문자열을 형성하도록 병합됩니다. 여기서 **TOPICOBJ** 구성 토픽 문자열이 항상 첫 번째에 오며, 문자열의 두 부분은 항상 "/" 문자로 나뉩니다.

마찬가지로 MQI 프로그램에서 전체 토픽 이름은 MQOPEN에 의해 작성됩니다. 이는 다음에 나열된 순서대로 발행/구독 MQI 호출에서 사용되는 2개의 필드로 구성됩니다.

1. **ObjectName** 필드에 이름 지정된, 토픽 오브젝트의 **TOPICSTR** 속성.
2. 애플리케이션에서 제공하는 하위 토픽을 정의하는 **ObjectString** 매개변수.

결과로 생성되는 토픽 문자열은 **ResObjectString** 매개변수에서 리턴됩니다.

이러한 필드는 각 필드의 첫 번째 문자가 공백 또는 널 문자가 아니면 존재한다고 간주되며 필드 길이는 0보다 큽니다. 필드 중 하나만 존재하면 이는 토픽 이름으로 그대로 사용됩니다. 필드 모두에 값이 없는 경우 전체 토픽 이름이 유효하지 않으면 호출은 이유 코드 MQRC_UNKNOWN_OBJECT_NAME 또는 MQRC_TOPIC_STRING_ERROR로 실패합니다.

두 필드가 모두 있으면 "/" 문자는 결과로 생성되어 결합된 토픽 이름의 두 요소 사이에 삽입됩니다.

다음 표는 토픽 문자열 연결의 예를 표시합니다.

표 2. 토픽 문자열 연결 예제

토픽 오브젝트의 TOPICSTR	애플리케이션 또는 DEFINE SUB 명령에서 제공되는 토픽 문자열	전체 토픽 이름	주석
Football/Scores	' '	Football/Scores	토픽 오브젝트의 TOPICSTR은 단독으로 사용됩니다.
' '	Football/Scores	Football/Scores	ObjectString/TOPICSTR은 단독으로 사용됩니다.
Football	Scores	Football/Scores	"/" 문자가 연결 지점에 추가되었습니다.
Football	/Scores	Football//Scores	두 문자열 사이에 '빈 노드'가 생성됩니다. 이는 "Football/Scores"와 다릅니다.
/Football	Scores	/Football/Scores	토픽이 '빈 노드'로 시작합니다. 이는 "Football/Scores"와 다릅니다.

"/" 문자는 특수 문자로 간주되며 67 페이지의 『토픽 트리』에서 전체 토픽 이름에 대한 구조를 제공합니다. "/" 문자는 토픽 트리의 구조가 영향을 받기 때문에 다른 이유로 사용해서는 안 됩니다. 토픽 "/Football"은 토픽 "Football"과 동일하지 않습니다.

참고: 구독을 작성할 때 토픽 오브젝트를 사용하면, 토픽 오브젝트 토픽 문자열의 값이 정의 시에 구독에서 수정됩니다. 토픽 오브젝트의 후속 변경사항은 구독이 정의되는 토픽 문자열에 영향을 미치지 않습니다.

토픽 문자열의 와일드카드 문자

다음 와일드카드 문자는 특수 문자입니다.

- 더하기 부호(+)
- 숫자 부호(#)
- 별표(*)
- 물음표(?)

와일드카드 문자는 구독에서 사용될 경우에만 특별한 의미를 가집니다. 이러한 문자가 다른 곳에서 사용될 경우 적합하지 않다고 간주되지 않지만, 사용 방법을 이해하고 있어야 하며 토픽 오브젝트 발행 또는 정의 시에 토픽 문자열에 이러한 문자를 사용하지 않는 것이 좋습니다.

한 토픽 레벨 내의 다른 문자(해당 문자 포함)와 함께 # 또는 +가 결합된 토픽 문자열에서 발행할 경우, 토픽 문자열은 와일드카드 설계와 함께 구독될 수 있습니다.

두 / 문자 사이의 유일한 문자로 # 또는 +를 사용하여 토픽 문자열에서 발행할 경우, 토픽 문자열은 와일드카드 설계 MQSO_WILDCARD_TOPIC을 사용하여 애플리케이션에 의해 명시적으로 구독될 수 없습니다. 이 상황으로 인해 애플리케이션은 예상보다 많은 발행을 가져옵니다.

정의된 토픽 오브젝트의 토픽 문자열에서 와일드카드 문자를 사용해서는 안 됩니다. 이렇게 할 경우, 오브젝트가 발행자에 의해 사용될 때는 문자가 리터럴 문자로 처리되고 구독에서 사용될 경우 와일드카드 문자로 처리됩니다. 이는 혼동을 야기할 수 있습니다.

코드 스니펫 예제

예제 프로그램 예제 2: 가변 토픽에 대한 발행자에서 추출한 이 코드 스니펫은 가변 토픽 문자열에 토픽 오브젝트를 결합합니다.

```
MQOD   td = {MQOD_DEFAULT}; /* Object Descriptor      */
td.ObjectType = MQOT_TOPIC;  /* Object is a topic      */
td.Version = MQOD_VERSION_4;  /* Descriptor needs to be V4  */
strncpy(td.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
td.ObjectString.VSPtr = topicString;
td.ObjectString.VSLength = (MQLONG)strlen(topicString);
td.ResObjectString.VSPtr = resTopicStr;
td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
```

토픽 트리

정의한 각 토픽은 요소 또는 토픽 트리의 노드입니다. 토픽 트리는 비워 두거나 MQSC 또는 PCF 명령을 사용하여 이미 정의된 토픽으로 시작되거나 해당 토픽을 포함할 수 있습니다. 토픽 작성 명령을 사용하거나 발행 또는 구독에 토픽을 처음으로 지정하면 새 토픽을 정의할 수 있습니다.

아무 문자열이나 사용해도 토픽의 토픽 문자열을 정의할 수 있으나 계층 구조 트리에 적합한 토픽 문자열을 선택해야 합니다. 토픽 문자열 및 토픽 트리를 신중하게 디자인하면 다음과 같은 조작에 도움이 될 수 있습니다.

- 다중 토픽을 구독합니다.
- 보안 정책을 설정합니다.

토픽 트리를 평면적 선형 구조로 구성할 수도 있지만, 토픽 트리를 하나 이상의 루트 토픽이 있는 계층 구조로 빌드하는 것이 더 좋습니다. 보안 계획 및 토픽에 대한 자세한 정보는 발행/구독 보안을 참조하십시오.

67 페이지의 그림 18에서는 루트 토픽이 하나인 토픽 트리 예제를 보여줍니다.

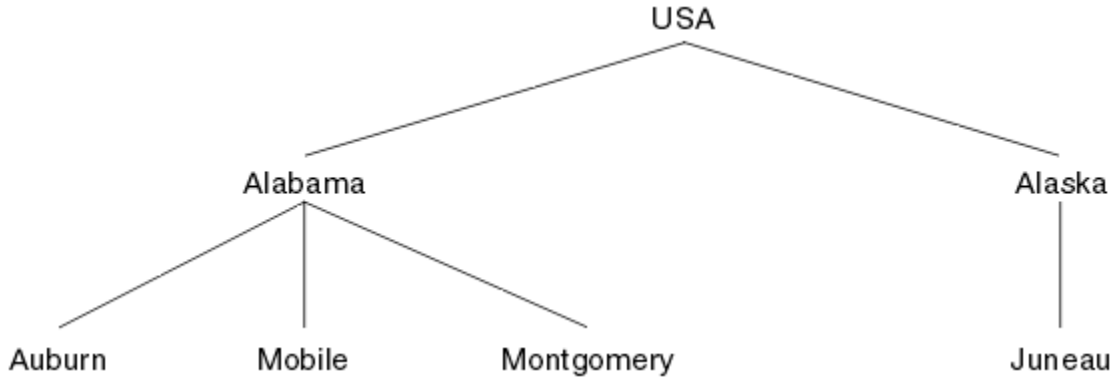


그림 *18.* 토픽 트리 예제

그림의 각 문자열은 토픽 트리의 노드를 나타냅니다. 전체 토픽 문자열은 토픽 트리에 있는 하나 이상의 레벨에서 노드를 집계하여 작성됩니다. 레벨은 "/" 문자로 구분됩니다. 지정된 전체 토픽 문자열의 형식은 "root/level2/level3"입니다.

67 페이지의 그림 18에 표시된 토픽 트리의 올바른 토픽은 다음과 같습니다.

"USA"
"USA/Alabama"
"USA/Alaska"
"USA/Alabama/Auburn"
"USA/Alabama/Mobile"
"USA/Alabama/Montgomery"
"USA/Alaska/Juneau"

토픽 문자열 및 토픽 트리를 디자인할 때, 큐 관리자는 토픽 문자열의 의미를 도출하도록 시도하거나 토픽 문자열 자체를 해석하지 않습니다. 단순히 선택한 메시지를 해당 토픽의 구독자에게 보내는 데 토픽 문자열을 사용합니다.

토픽 트리의 구성 및 컨텐츠에는 다음 원칙이 적용됩니다.

- 토픽 트리의 레벨 수에 제한이 없습니다.
- 토픽 트리의 레벨 이름 길이에 제한이 없습니다.
- "루트" 노드 수에 제한이 없습니다. 즉 토픽 트리 수에 제한이 없습니다.

관련 태스크

토픽 트리에서 불필요한 토픽 수 감소

관리 토픽 오브젝트

관리 토픽 오브젝트를 사용하여 기본이 아닌 특정 속성을 토픽에 지정할 수 있습니다.

68 페이지의 그림 19에서는 서로 다른 스포츠를 포함하는 별도의 토픽으로 구분된 Sport의 상위 레벨 토픽을 토픽 트리로 시각화하는 방법을 보여줍니다.

그림 *19.* 토픽 트리의 시각화

68 페이지의 그림 20에서는 각 스포츠에 대한 여러 유형의 정보를 구분하기 위해 추가로 토픽 트리를 구분하는 방법을 보여줍니다.

그림 *20.* 확장된 토픽 트리

표시된 토픽 트리를 작성하려는 경우 관리 토픽 오브젝트를 정의하지 않아도 됩니다. 이 트리의 각 노드는 발행 또는 구독 조작에서 작성된 토픽 문자열에 의해 정의됩니다. 트리에서 각 토픽은 해당 상위에서 속성을 상속합니다. 기본적으로 모든 속성은 ASPARENT로 설정되므로 속성은 상위 토픽 오브젝트에서 상속됩니다. 이 예제에서 모든 토픽에는 Sport 토픽과 동일한 속성이 있습니다. Sport 토픽에는 관리 토픽 오브젝트가 없으며 SYSTEM.BASE.TOPIC에서 해당 속성을 상속합니다.

토픽 트리의 루트 노드(즉, SYSTEM.BASE.TOPIC)에서 mqm이 아닌 사용자에게는 권한을 부여하지 않는 것이 좋습니다. 권한은 상속되지만 제한할 수 없기 때문입니다. 따라서 이 레벨에서 권한을 제공하면 전체 트리에 권한을 제공하게 됩니다. 따라서 계층의 더 낮은 토픽 레벨에서 권한을 제공해야 합니다.

관리 토픽 오브젝트는 토픽 트리에 있는 특정 노드에 대한 특정 속성을 정의하는 데 사용할 수 있습니다. 다음 예제에서 관리 토픽 오브젝트는 Soccer 토픽의 지속 가능 구독 특성 DURSUB를 NO 값으로 설정하도록 정의됩니다.

```
DEFINE TOPIC(FOOTBALL.EUROPEAN)
TOPICSTR('Sport/Soccer')
DURSUB(NO)
DESCR('Administrative topic object to disallow durable subscriptions')
```

이제 토픽 트리는 다음과 같이 시각화될 수 있습니다.

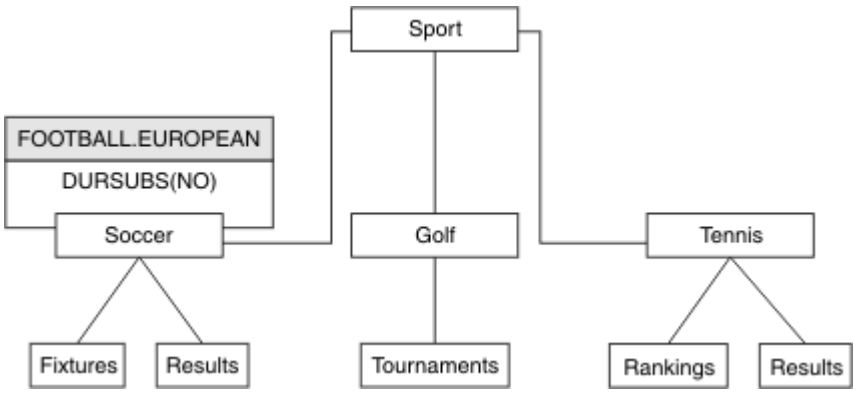


그림 *21. Sport/Soccer* 토픽과 연관된 관리 토픽 오브젝트의 시각화

트리에서 Soccer 아래 토픽을 구독하는 애플리케이션은 계속해서 관리 토픽 오브젝트를 추가하기 전에 사용하던 토픽 문자열을 사용할 수 있습니다. 그러나 이제 애플리케이션은 문자열 `/Sport/Soccer` 대신 오브젝트 이름 `FOOTBALL.EUROPEAN`을 사용하여 구독하도록 작성될 수 있습니다. 예를 들어 `/Sport/Soccer/Results`를 구독하려면 애플리케이션은 `MQSD.ObjectName`을 `FOOTBALL.EUROPEAN`으로, `MQSD.ObjectString`을 `Results`로 지정할 수 있습니다.

이 기능을 사용하면 애플리케이션 개발자로부터 토픽 트리의 일부를 숨길 수 있습니다. 토픽 트리의 특정 노드에서 관리 토픽 오브젝트를 정의하면 애플리케이션 개발자가 노드의 하위로 고유한 토픽을 정의할 수 있습니다. 개발자는 상위 토픽에 대해 알아야 하지만 상위 트리의 다른 노드에 대해서는 알지 않아도 됩니다.

속성 상속

토픽 트리에 관리 토픽 오브젝트가 많으면 기본적으로 각 관리 토픽 오브젝트는 가장 가까운 상위 관리 토픽에서 해당 속성을 상속합니다. 이전 예제는 69 페이지의 그림 22에서 확장되었습니다.

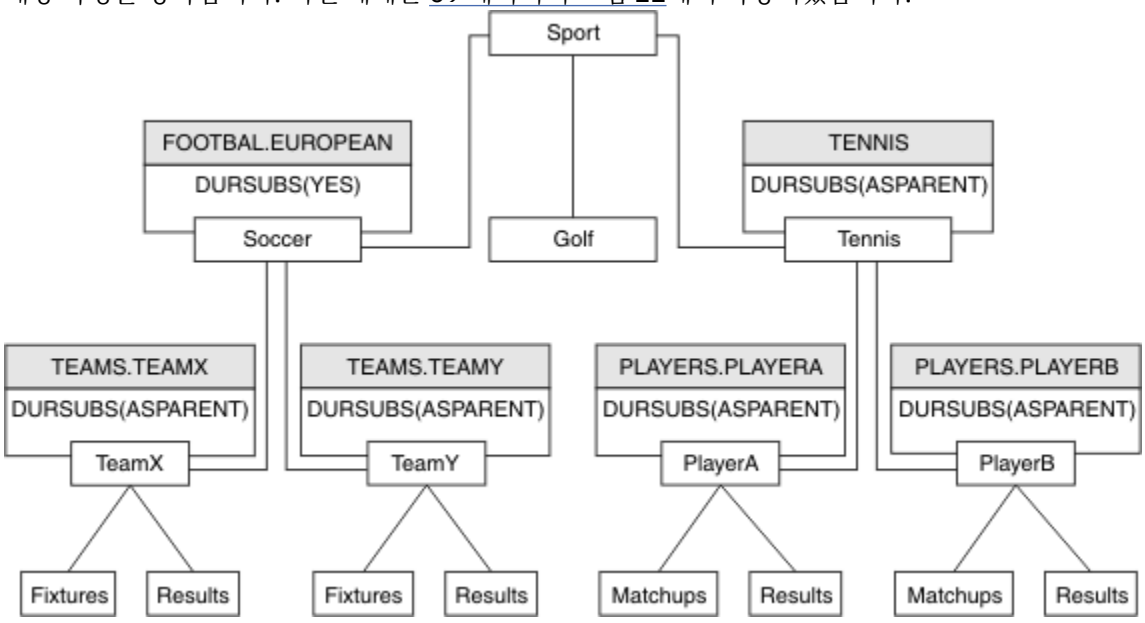


그림 *22.* 여러 관리 토픽 오브젝트를 포함하는 토픽 트리

예를 들어 상속을 사용하여 `/Sport/Soccer`의 모든 하위 토픽에 구독이 비지속인 특성을 제공합니다. `FOOTBALL.EUROPEAN` 의 `DURSUB` 속성을 `NO`로 변경하십시오.

이 속성은 다음과 같은 명령을 사용하여 설정할 수 있습니다.

```
ALTER TOPIC(FOOTBALL.EUROPEAN) DURSUB(NO)
```

Sport/Soccer의 하위 토픽에 대한 모든 관리 토픽 오브젝트에서는 DURSUB가 기본값, ASPARENT로 설정되어 있습니다. FOOTBALL.EUROPEAN의 DURSUB 특성 값을 NO로 변경하면 Sport/Soccer의 하위 토픽은 DURSUB 특성 값 NO를 상속합니다. Sport/Tennis의 모든 하위 토픽은 SYSTEM.BASE.TOPIC 오브젝트에서 DURSUB의 값을 상속합니다. SYSTEM.BASE.TOPIC의 값은 YES입니다.

이제 Sport/Soccer/TeamX/Results 토픽에서 지속 가능 구독을 작성하려고 하면 실패합니다. 그러나 Sport/Tennis/PlayerB/Results의 지속 가능 구독 작성에는 성공합니다.

WILDCARD 특성으로 와일드카드 사용 제어

MQSC **Topic** WILDCARD 특성 또는 동등한 PCF 토픽 WildcardOperation 특성을 사용하여 와일드카드 토픽 문자열 이름을 사용하는 구독자 애플리케이션으로 발행물의 전달을 제어할 수 있습니다. WILDCARD 특성은 다음과 같은 두 가지 가능한 값을 보유할 수 있습니다.

WILDCARD
이 토픽에 관한 와일드카드 구독의 동작입니다.

PASSTHRU
이 토픽 오브젝트의 토픽 문자열보다 덜 특정한 와일드카드 토픽에 대한 구독이 이 토픽 및 이 토픽보다 더욱 특정한 토픽 문자열에 대한 발행물을 수신합니다.

BLOCK
이 토픽 오브젝트의 토픽 문자열보다 덜 특정한 와일드카드 토픽에 대한 구독이 이 토픽 또는 이 토픽보다 더욱 특정한 토픽 문자열에 대한 발행물을 수신하지 않습니다.

이 속성의 값은 구독이 정의될 때 사용됩니다. 이 속성을 대체할 경우 기존 구독에 포함된 토픽 세트는 수정의 영향을 받지 않습니다. 이 시나리오는 토픽 오브젝트를 작성 또는 삭제할 때 토폴로지가 변경된 경우에도 적용됩니다. WILDCARD 속성을 수정한 후에 작성된 구독과 일치하는 토픽 세트가 수정된 토폴로지를 사용하여 작성됩니다. 일치하는 토픽 세트를 강제로 기존 구독에 대해 재평가하려는 경우 큐 관리자를 재시작해야 합니다.

예제, 73 페이지의 『예제: Sport 발행/구독 클러스터 작성』 에서는 70 페이지의 그림 23에 표시된 토픽 트리 구조를 작성하는 단계를 수행할 수 있습니다.

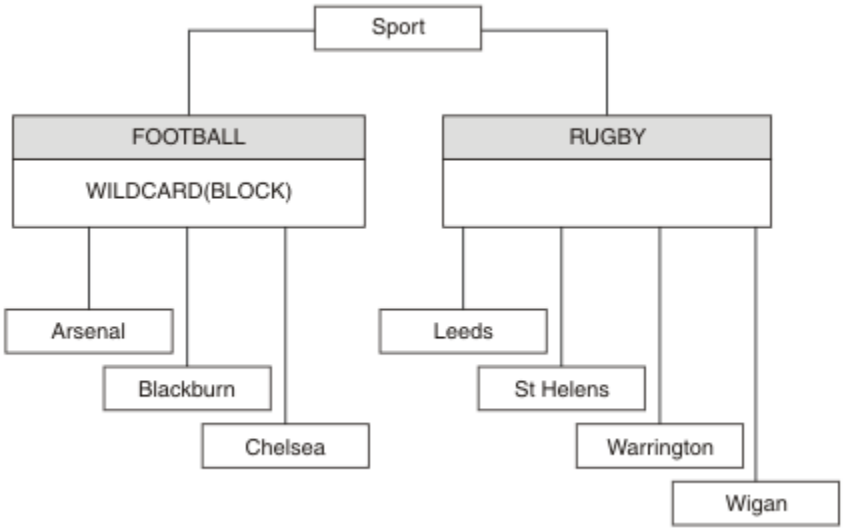


그림 *23. WILDCARD* 특성, *BLOCK*을 사용하는 토픽 트리

와일드카드 토픽 문자열 #을 사용하는 구독자는 Sport 토픽 및 Sport/Rugby 하위 트리에 대한 모든 발행을 수신합니다. 구독자는 Sport/Football 하위 트리에 대한 발행을 수신하지 않습니다. Sport/Football 토픽의 WILDCARD 특성 값은 BLOCK입니다.

PASSTHRU는 기본 설정입니다. Sport 트리의 노드에서 WILDCARD 특성 값 PASSTHRU를 설정할 수 있습니다. 노드에 WILDCARD 특성 값 BLOCK이 없으면 PASSTHRU를 설정해도 Sports 트리의 노드에서 구독자가 관찰하는 작동은 대체되지 않습니다.

예제에서 구독을 작성하여 와일드카드 설정이 전달되는 발행에 미치는 영향을 확인하십시오(75 페이지의 그림 27 참조). 일부 발행을 작성하려면 76 페이지의 그림 30에서 발행 명령을 실행하십시오.

```
pub QMA
```

그림 *24. QMA*에 발행

결과는 71 페이지의 표 3에 표시됩니다. WILDCARD 특성 값 BLOCK을 설정할 때 와일드카드를 포함하는 구독이 와일드카드 범위 내 토픽에 대한 발행을 수신하지 않도록 하는 방법에 주의하십시오.

표 *3. QMA*에서 수신된 발행

구독	토픽 문자열	수신된 발행	참고
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football에서 WILDCARD(BLOCK)으로 차단된 Football 하위 트리에 대한 모든 발행
SARSENAL	Sports/#/Arsenal	-	Sports/Football에서 WILDCARD(BLOCK)은 Arsenal에서의 와일드카드 구독을 금지함
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby의 기본 WILDCARD는 Leeds에서 와일드카드 구독을 금지하지 않습니다.

참고:

구독에 WILDCARD 특성 값 BLOCK이 있는 토픽 오브젝트와 일치하는 와일드카드가 있다고 가정합니다. 또한 구독이 일치하는 와일드카드 오른쪽에 토픽 문자열을 포함하면 구독은 발행을 수신하지 않습니다. 차단되지 않은 발행 세트는 차단된 와일드카드 상위에 해당하는 토픽에 대한 발행입니다. BLOCK 특성 값이 설정된 토픽의 하위에 해당하는 토픽에 대한 발행은 와일드카드로 차단됩니다. 따라서 와일드카드 오른쪽에 토픽을 포함하는 구독 토픽 문자열은 일치하는 발행을 수신하지 않습니다.

WILDCARD 특성 값을 BLOCK으로 설정해도 와일드카드를 포함하는 토픽 문자열을 사용하여 구독할 수 없음을 의미하지는 않습니다. 이러한 구독은 정상적입니다. 구독이 WILDCARD 특성 값 BLOCK이 설정된 토픽 오브젝트를 포함하는 토픽과 일치하는 명시적 토픽을 보유합니다. 이는 WILDCARD 특성 값 BLOCK이 설정된 토픽의 상위 또는 하위에 해당하는 토픽에 대한 와일드카드를 사용합니다. 70 페이지의 그림 23의 예제에서 Sports/Football/#과 같은 구독은 발행을 수신할 수 있습니다.

와일드카드 및 클러스터 토픽

클러스터 토픽 정의는 클러스터의 모든 큐 관리자로 전파됩니다. 클러스터의 한 큐 관리자에서 클러스터 토픽에 대한 구독으로 인해 큐 관리자가 프록시 구독을 작성할 수 있습니다. 프록시 구독은 클러스터의 다른 모든 큐 관리자에서 작성됩니다. 클러스터 토픽과 결합해 와일드카드를 포함하는 토픽 문자열을 사용하는 구독은 작동을 예상하기 어려울 수 있습니다. 작동은 다음 예제에서 설명됩니다.

예제, 73 페이지의 『예제: Sport 발행/구독 클러스터 작성』에 설정된 클러스터의 경우 QMB에는 QMA와 동일한 구독 세트가 있지만 QMB는 발행자가 QMA에 발행한 후 발행을 수신하지 않습니다(71 페이지의 그림 24 참조). Sports/Football 및 Sports/Rugby 토픽이 클러스터 토픽이어도 fullsubs.tst에 정의된 구독은 클러스터 토픽을 참조하지 않습니다. 프록시 구독은 QMB에서 QMA로 전파되지 않습니다. 프록시 구독이 없으면 QMA에 대한 발행은 QMB로 전달되지 않습니다.

Sports/#/Leeds와 같은 일부 구독은 이 경우 Sports/Rugby에 해당하는 클러스터 토픽을 참조할 수도 있습니다. Sports/#/Leeds 구독은 실제로 토픽 오브젝트 SYSTEM.BASE.TOPIC으로 해석됩니다.

Sports/#/Leeds와 같이 구독에서 참조하는 토픽 오브젝트를 해석하는 규칙은 다음과 같습니다. 토픽 문자열을 첫 번째 와일드카드까지 자릅니다. 토픽 문자열을 왼쪽부터 스캔하여 연관된 관리 토픽 오브젝트가 있는 첫 번째 토픽을 찾습니다. 이 토픽 오브젝트는 클러스터 이름을 지정하거나 로컬 토픽 오브젝트를 정의할 수 있습니다.

예 Sports/#/Leeds에서 잘림 이후의 토픽 문자열은 토픽 오브젝트가 없는 Sports이므로 Sports/#/Leeds 는 로컬 토픽 오브젝트인 SYSTEM.BASE.TOPIC에서 상속합니다.

클러스터 토픽을 구독하여 와일드카드 전파 방식을 변경하는 방법을 확인하려면 배치 스크립트, upsubs.bat를 실행하십시오. 스크립트는 구독 큐를 지우고 fullsubs.tst에서 클러스터 토픽 구독을 추가합니다. puba.bat를 다시 실행하여 발행 배치를 작성하십시오(71 페이지의 그림 24 참조).

72 페이지의 표 4에서는 발행이 발행된, 동일한 큐 관리자에 2개의 새 구독을 추가하는 결과를 보여줍니다. 결과는 예상한 대로입니다. 새 구독은 각각 하나의 발행을 수신하고 다른 구독에서 수신하는 발행 수는 변경되지 않습니다. 다른 클러스터 큐 관리자에서 예상치 못한 결과가 발생합니다(72 페이지의 표 5 참조).

표 *4. QMA*에서 수신된 발행

구독	토픽 문자열	수신된 발행	참고
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football에서 WILDCARD(BLOCK)으로 차단된 Football 하위 트리에 대한 모든 발행
SARSENAL	Sports/#/Arsenal	-	Sports/Football에서 WILDCARD(BLOCK)은 Arsenal에서의 와일드카드 구독을 금지함
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby의 기본 WILDCARD는 Leeds에서 와일드카드 구독을 금지하지 않습니다.
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal은 구독이 와일드카드를 포함하지 않으므로 발행을 수신합니다.
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds는 모든 이벤트에서 발행을 수신합니다.

72 페이지의 표 5에서는 QMB에서 2개의 새 구독을 추가하고 QMA에서 발행하는 결과를 보여줍니다. QMB가 이러한 2개의 새 구독 없이 발행을 수신하지 않는다는 점을 상기하십시오. 예상대로 Sports/FootBall 및 Sports/Rugby 가 둘 다 클러스터 토픽이므로 두 개의 새 구독이 발행물을 수신합니다. QMB 는 Sports/Football/Arsenal 및 Sports/Rugby/Leeds 에 대한 프록시 구독을 QMA에 전달하며, 이는 QMB에 발행물을 송신합니다.

예상치 못한 결과는, 이전에 발행을 수신하지 않던 2개의 구독 Sports/# 및 Sports/#/Leeds가 이제 발행을 수신한다는 점입니다. 이유는 바로, 다른 구독에 대해 QMB로 전달되는 Sports/Football/Arsenal 및 Sports/Rugby/Leeds 발행이 이제 QMB에 연결된 모든 구독자에게 사용 가능합니다. 결과적으로 로컬 토픽 Sports/# 및 Sports/#/Leeds에 대한 구독은 Sports/Rugby/Leeds 발행을 수신합니다. Sports/#/Arsenal은 계속해서 발행을 수신하지 않습니다. Sports/Football에서 해당 WILDCARD 특성 값이 BLOCK으로 설정되었기 때문입니다.

표 *5. QMB*에서 수신된 발행

구독	토픽 문자열	수신된 발행	참고
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*Sports/Football*에서 *WILDCARD(BLOCK)*에 의해 차단된 *Football* 하위 트리에 대한 모든 발행
SARSENAL	Sports/#/Arsenal	-	Sports/Football에서 WILDCARD(BLOCK)은 Arsenal에서의 와일드카드 구독을 금지함

표 5. *QMB*에서 수신된 발행 *(계속)*			
구독	**토픽 문자열**	**수신된 발행**	**참고**
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	*Sports/Rugby*의 기본 *WILDCARD*는 *Leeds*에서 와일드카드 구독을 금지하지 않습니다.
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal은 구독이 와일드카드를 포함하지 않으므로 발행을 수신합니다.
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds는 모든 이벤트에서 발행을 수신합니다.

대부분의 애플리케이션에서는 한 구독이 다른 구독의 작동에 영향을 주는 것은 바람직하지 않습니다. 값이 BLOCK인 WILDCARD 특성의 한 가지 중요한 사용법은 균일하게 작동하는 와일드카드를 포함하는 동일한 토픽 문자열에 대한 구독을 작성하는 것입니다. 발행자와 동일한 큐 관리자 또는 다른 큐 관리자의 구독인지에 상관없이 구독 결과는 동일합니다.

와일드카드 및 스트림

발행/구독 API에 기록된 새 애플리케이션의 경우 * 에 대한 구독이 발행물을 수신하지 않습니다. 모든 Sport 발행을 수신하려면 Sports/* 또는 Sports/#을 구독해야 하며,하고 Business 발행도 마찬가지입니다.

큐된 기존 발행/구독 애플리케이션의 동작은 발행/구독 브로커가 IBM MQ의 이후 버전으로 마이그레이션될 때 변경되지 않습니다. **Publish**, **Register Publisher**또는 **Subscriber** 명령의 **StreamName** 특성은 스트림이 마이그레이션된 토픽의 이름에 맵핑됩니다.

와일드카드 및 구독 지점

발행/구독 API에 기록된 새 애플리케이션의 경우 마이그레이션의 효과는 * 에 대한 구독이 발행물을 수신하지 않는다는 점입니다. 모든 Sport 발행을 수신하려면 Sports/* 또는 Sports/#을 구독해야 하며,하고 Business 발행도 마찬가지입니다.

큐된 기존 발행/구독 애플리케이션의 동작은 발행/구독 브로커가 IBM MQ의 이후 버전으로 마이그레이션될 때 변경되지 않습니다. **Publish**, **Register Publisher** 또는 **Subscriber** 명령에서 **SubPoint** 특성은 구독이 마이그레이션되는 토픽 이름에 맵핑됩니다.

예제: Sport 발행/구독 클러스터 작성

다음 단계는 네 개의 큐 관리자 (두 개의 전체 저장소, CL1A 및 CL1B, 두 개의 부분 저장소, QMA 및 QMB) 가 있는 CL1클러스터를 작성합니다. 전체 저장소는 클러스터 정의만 보유하는 데 사용됩니다. QMA는 클러스터 토픽 호스트를 지정합니다. 지속 가능 구독은 QMA 및 QMB 모두에서 정의됩니다.

참고: 예제는 Windows용으로 코딩되었습니다. 다른 플랫폼에서 예제를 구성하고 테스트하도록 qmgrs.bat 작성 및 pub.bat 작성을 다시 코딩해야 합니다.

1. 스크립트 파일을 작성하십시오.
   a. topics.tst 작성
   b. wildsubs.tst 작성
   c. fullsubs.tst 작성
   d. qmgrs.bat 작성
   e. pub.bat 작성
2. qmgrs.bat 작성을 실행하여 구성을 작성하십시오.

```
qmgrs
```

70 페이지의 그림 23에서 토픽을 작성하십시오. 그림 5에서 스크립트는 클러스터 토픽 Sports/Football 및 Sports/Rugby를 작성합니다.

참고: REPLACE 옵션은 주제의 TOPICSTR 특성을 바꾸지 않습니다. TOPICSTR은(는) 다른 토픽 트리를 테스트하는 예제에서 사용되는 특성입니다. 토픽을 변경하려면 먼저 토픽을 삭제하십시오.

```
DELETE TOPIC ('Sports')
DELETE TOPIC ('Football')
DELETE TOPIC ('Arsenal')
DELETE TOPIC ('Blackburn')
DELETE TOPIC ('Chelsea')
DELETE TOPIC ('Rugby')
DELETE TOPIC ('Leeds')
DELETE TOPIC ('Wigan')
DELETE TOPIC ('Warrington')
DELETE TOPIC ('St. Helens')

DEFINE TOPIC ('Sports')   TOPICSTR('Sports')
DEFINE TOPIC ('Football')  TOPICSTR('Sports/Football') CLUSTER(CL1) WILDCARD(BLOCK)
DEFINE TOPIC ('Arsenal')  TOPICSTR('Sports/Football/Arsenal')
DEFINE TOPIC ('Blackburn') TOPICSTR('Sports/Football/Blackburn')
DEFINE TOPIC ('Chelsea')  TOPICSTR('Sports/Football/Chelsea')
DEFINE TOPIC ('Rugby')   TOPICSTR('Sports/Rugby')  CLUSTER(CL1)
DEFINE TOPIC ('Leeds')   TOPICSTR('Sports/Rugby/Leeds')
DEFINE TOPIC ('Wigan')   TOPICSTR('Sports/Rugby/Wigan')
DEFINE TOPIC ('Warrington') TOPICSTR('Sports/Rugby/Warrington')
DEFINE TOPIC ('St. Helens') TOPICSTR('Sports/Rugby/St. Helens')
```

그림 *25.* 토픽 삭제 및 작성*: topics.tst*

참고: REPLACE가 토픽 문자열을 바꾸지 않으므로 토픽을 삭제합니다.

와일드카드를 사용하여 구독을 작성하십시오. 70 페이지의 그림 23에서 토픽 오브젝트를 포함하는 토픽에 대응하는 와일드카드. 각 구독에 대한 큐를 작성하십시오. 스크립트를 실행하거나 다시 실행하면 큐가 지워지고 구독이 삭제됩니다.

참고: REPLACE 옵션은 구독의 TOPICOBJ 또는 TOPICSTR 특성을 바꾸지 않습니다. TOPICOBJ 또는 TOPICSTR은 다른 구독을 테스트하기 위해 예제에서 다양하게 사용할 수 있는 특성입니다. 변경하려면 먼저 구독을 삭제하십시오.

```
DEFINE QLOCAL(QSPORTS)  REPLACE
DEFINE QLOCAL(QSARSENAL) REPLACE
DEFINE QLOCAL(QSLEEDS)  REPLACE
CLEAR QLOCAL(QSPORTS)
CLEAR QLOCAL(QSARSENAL)
CLEAR QLOCAL(QSLEEDS)

DELETE SUB  (SPORTS)
DELETE SUB  (SARSENAL)
DELETE SUB  (SLEEDS)
DEFINE SUB  (SPORTS)  TOPICSTR('Sports/#')     DEST(QSPORTS)
DEFINE SUB  (SARSENAL) TOPICSTR('Sports/+/Arsenal') DEST(QSARSENAL)
DEFINE SUB  (SLEEDS)  TOPICSTR('Sports/+/Leeds')  DEST(QSLEEDS)
```

그림 *26.* 와일드카드 구독 작성*: wildsubs.tst*

클러스터 토픽 오브젝트를 참조하는 구독을 작성하십시오.

참고:

구분 기호, /는 TOPICOBJ에서 참조하는 토픽 문자열과 TOPICSTR에서 정의하는 토픽 문자열 사이에 자동으로 삽입됩니다.

DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) 정의는 동일한 구독을 작성합니다. TOPICOBJ는 이미 정의한 토픽 문자열을 참조하는 빠른 방법으로 사용됩니다. 작성되면 구독은 더 이상 토픽 오브젝트를 참조하지 않습니다.

```
DEFINE QLOCAL(QFARSENAL) REPLACE
DEFINE QLOCAL(QRLEEDS)  REPLACE
CLEAR QLOCAL(QFARSENAL)
CLEAR QLOCAL(QRLEEDS)

DELETE SUB  (FARSENAL)
DELETE SUB  (RLEEDS)
DEFINE SUB  (FARSENAL) TOPICOBJ('Football') TOPICSTR('Arsenal') DEST(QFARSENAL)
DEFINE SUB  (RLEEDS)  TOPICOBJ('Rugby')  TOPICSTR('Leeds')  DEST(QRLEEDS)
```

그림 *27.* 구독 삭제 및 삭제*: fullsubs.tst*

2개의 저장소를 포함하는 클러스터를 작성하십시오. 발행 및 구독을 위해 2개의 부분 저장소를 작성하십시오. 모두 삭제하고 다시 시작하도록 스크립트를 다시 실행하십시오. 또한 스크립트는 토픽 계층 및 초기 와일드카드 구독을 작성합니다.

참고:

다른 플랫폼에 비슷한 스크립트를 작성하거나 모두 명령을 입력하십시오. 스크립트를 사용하면 모두 삭제하고 동일한 구성으로 다시 빠르게 시작할 수 있습니다.

```
@echo off
set port.CL1B=1421
set port.CLIA=1420
for %%A in (CL1A CL1B QMA QMB) do call :createQM %%A
call :configureQM CL1A CL1B %port.CL1B% full
call :configureQM CL1B CL1A %port.CL1A% full
for %%A in (QMA QMB) do call :configureQM %%A CL1A %port.CL1A% partial
for %%A in (topics.tst wildsubs.tst) do runmqsc QMA < %%A
for %%A in (wildsubs.tst) do runmqsc QMB < %%A
goto:eof

:createQM
echo Configure Queue manager %1
endmqm -p %1
for %%B in (dlt crt str) do %%Bmqm %1
goto:eof

:configureQM
if %1==CL1A set p=1420
if %1==CL1B set p=1421
if %1==QMA set p=1422
if %1==QMB set p=1423
echo configure %1 on port %p% connected to repository %2 on port %3 as %4 repository
echo DEFINE LISTENER(LST%1) TRPTYPE(TCP) PORT(%p%) CONTROL(QMGR) REPLACE | runmqsc %1
echo START LISTENER(LST%1) | runmqsc %1
if full==%4 echo ALTER QMGR REPOS(CL1) DEADQ(SYSTEM.DEAD.LETTER.QUEUE) | runmqsc %1
echo DEFINE CHANNEL(TO.%2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('LOCALHOST(%3)') CLUSTER(CL1)
REPLACE | runmqsc %1
echo DEFINE CHANNEL(TO.%1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('LOCALHOST(%p%)')
CLUSTER(CL1) REPLACE | runmqsc %1
goto:eof
```

그림 *28.* 큐 관리자 작성*: qmgrs.bat*

클러스터 토픽에 구독을 추가하여 구성을 업데이트하십시오.

```
@echo off
for %%A in (QMA QMB) do runmqsc %%A < wildsubs.tst
for %%A in (QMA QMB) do runmqsc %%A < upsubs.tst
```

그림 *29.* 구독 업데이트*: upsubs.bat*

큐 관리자에서 매개변수로 pub.bat를 실행하여 발행 토픽 문자열을 포함하는 메시지를 발행하십시오. Pub.bat 는 샘플 프로그램 **amqspub**를 사용합니다.

```
@echo off
@rem Provide queue manager name as a parameter
set S=Sports
set S=6 Sports/Football Sports/Football/Arsenal
set S=6 Sports/Rugby Sports/Rugby/Leeds
for %%B in (6) do echo %%B | amqspub %%B %1
```

그림 *30.* 발행*: pub.bat*

스트림 및 토픽

큐에 있는 발행/구독에는 통합된 발행/구독 모델에 존재하지 않는 발행 스트림의 개념이 내포되어 있습니다. 큐에 있는 발행/구독에서는 스트림이 여러 가지 토픽에 대한 정보 플로우를 분리하는 방법을 제공합니다. 스트림은 다른 토픽 ID에 관리적으로 맵핑될 수 있는 최상위 레벨 토픽으로 구현됩니다.

기본 스트림 SYSTEM.BROKER.DEFAULT.STREAM은 네트워크의 모든 브로커 및 큐 관리자에 대해 자동으로 설정되며 기본 스트림을 사용하기 위해 추가 구성이 필요하지 않습니다. 디폴트 스트림을 이름이 지정되지 않은 디폴트 토픽 공간으로 생각해보십시오. 기본 스트림에 발행된 토픽은 큐에 있는 발행/구독이 사용 가능한 상태에서 연결된 모든 큐 관리자가 즉시 사용할 수 있습니다. 이름 지정된 스트림은 별도의 이름 지정된 토픽 공간입니다. 이름 지정된 스트림을 사용되는 각 브로커에 정의해야 합니다.

발행자 및 구독자가 다른 큐 관리자에 있으면 브로커를 동일한 브로커 계층에 연결한 후에 발행 및 구독에서 이들 사이의 전달을 위해 추가 구성이 필요하지 않습니다. 동일한 상호 운용성이 역으로도 작용합니다.

이름 지정된 스트림

큐에 있는 발행/구독 프로그래밍 모델에서 작동하는 솔루션 설계자는 Sport로 이름 지정된 스트림에 모든 Sport 발행을 배치하도록 결정할 수 있습니다. 큐된 발행/구독이 사용 가능한 IBM MQ 에서 실행되는 큐 관리자가 스트림을 사용할 수 있도록 하려면 스트림을 수동으로 추가해야 합니다.

스트림 Sport의 Soccer/Results를 구독하는 큐에 있는 발행/구독 애플리케이션은 그대로 작동합니다. 또한 MQSUB를 사용하고 Soccer/Results 토픽 문자열을 제공하여 Sport 토픽을 구독하는 통합된 발행/구독 애플리케이션은 동일한 발행을 수신합니다.

스트림 추가 태스크는 스트림 추가 주제에서 설명됩니다. 스트림을 수동으로 추가해야 하는 두 가지 이유가 있습니다.

1. 통합된 발행/구독 MQI 인터페이스로 애플리케이션을 마이그레이션하는 대신, 이후 버전 큐 관리자에서 실행되는 큐에 있는 발행/구독 애플리케이션을 계속 개발합니다.
2. 스트림을 토픽에 디폴트 맵핑하면 토픽 공간에 "충돌"이 발생하고 스트림의 발행에 다른 곳의 발행과 동일한 토픽 문자열이 생깁니다.

권한

기본적으로 토픽 트리의 루트에는 여러 토픽 오브젝트 (SYSTEM.BASE.TOPIC, SYSTEM.BROKER.DEFAULT.STREAM및 SYSTEM.BROKER.DEFAULT.SUBPOINT) 가 있습니다. 권한(예: 발행 또는 구독용 권한)은 SYSTEM.BASE.TOPIC의 권한으로 판별됩니다.
SYSTEM.BROKER.DEFAULT.STREAM 또는 SYSTEM.BROKER.DEFAULT.SUBPOINT의 권한은 무시됩니다.
SYSTEM.BROKER.DEFAULT.STREAM 또는 SYSTEM.BROKER.DEFAULT.SUBPOINT가 삭제되고 비어 있지 않은 토픽 문자열로 다시 작성되면 해당 오브젝트에 정의된 권한은 보통 토픽 오브젝트와 같은 방식으로 사용됩니다.

스트림과 토픽 간의 맵핑

큐된 발행/구독 스트림은 큐를 작성하고 스트림과 동일한 이름을 제공하여 IBM MQ 에서 모방됩니다. 때로 큐를 스트림 큐라 부릅니다. 스트림 큐가 큐에 있는 발행/구독 애플리케이션에 이와 같이 표시되기 때문입니다.
SYSTEM.QPUBSUB.QUEUE.NAMELIST라는 특수 이름 목록에 큐를 추가해서 발행/구독 엔진에 큐가 식별됩니다. 추가적인 특수 큐를 이름 목록에 추가해서 필요한 만큼 스트림을 추가할 수 있습니다. 마지막으로 토픽에 발행 및 구독할 수 있도록 스트림과 동일한 이름의 토픽 및 스트림 이름과 동일한 토픽 문자열을 추가할 필요가 있습니다.

그러나 예외적인 상황에서 스트림에 대응하는 토픽에 토픽을 정의할 때 선택한 토픽 문자열을 제공할 수 있습니다. 토픽 문자열의 목적은 토픽 공간에서 토픽에 고유한 이름을 제공하는 것입니다. 일반적으로 스트림 이름은 이 목적을 완벽하게 수행합니다. 때때로 스트림 이름과 기존 토픽 이름이 충돌합니다. 문제점을 해결하려면 스트림과 연관된 토픽에 대해 다른 토픽 문자열을 선택하십시오. 토픽 문자열을 선택하고 고유한지 확인하십시오.

토픽 정의에 정의된 토픽 문자열에는 MQOPEN 또는 MQSUB MQI 호출을 사용하여 발행자 및 구독자가 제공한 토픽 문자열에 정상적인 방식으로 접두부가 붙습니다. 토픽 오브젝트를 사용하여 토픽을 참조하는 애플리케이션은 접두부 토픽 문자열의 선택에 의해 영향을 받지 않습니다. 따라서 토픽 공간에서 발행을 고유하게 만드는 임의의 토픽 문자열을 선택할 수 있습니다.

다른 스트림을 다른 토픽으로 다시 맵핑하는 작업은 한 토픽 세트를 다른 토픽 세트와 완전히 분리하기 위해 고유하도록 토픽 문자열에 사용되는 접두부에 따라 다릅니다. 맵핑이 작동하도록 엄격하게 지켜지는 보편적인 토픽 이름 지정 규칙을 정의해야 합니다.

IBM MQ에서 접두부 메커니즘을 사용하여 토픽 문자열을 토픽 영역의 다른 위치로 다시 맵핑합니다.

참고: 스트림을 삭제할 때에는 먼저 스트림의 모든 구독을 삭제하십시오. 이 조치는 브로커 계층의 기타 브로커에서 구독이 생성되는 경우 가장 중요합니다.

구독 지점 및 토픽

이름 지정된 구독 지점은 토픽 및 토픽 오브젝트에 의해 에뮬레이트됩니다.

수동으로 구독 지점을 추가하려면 구독 지점 추가를 참조하십시오.

IBM MQ의 구독 지점

IBM MQ는 IBM MQ 토픽 트리 내의 다른 토픽 공간에 구독 지점을 맵핑합니다. 구독 지점이 없는 명령 메시지의 토픽은 IBM MQ 토픽 트리의 루트에 변경되지 않은 상태로 맵핑되며 SYSTEM.BASE.TOPIC에서 특성을 상속합니다.

구독 지점을 포함하는 명령 메시지는 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST의 토픽 오브젝트 목록을 사용하여 처리됩니다. 명령 메시지의 구독 지점 이름은 목록에 있는 각 토픽 오브젝트에 대한 토픽 문자열과 비교하여 일치됩니다. 일치 항목을 찾으면 구독 지점 이름이 토픽 문자열에 토픽 노드로 추가됩니다. 토픽은 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST에 있는 연관된 토픽 오브젝트에서 해당 특성을 상속합니다.

구독 지점 사용 효과는 각 구독 지점에 대해 별도의 토픽 공간을 작성하는 것입니다. 토픽 공간은 구독 지점과 이름이 같은 토픽에 기반을 둡니다. 각 토픽 공간의 토픽은 구독 지점과 이름이 같은 토픽 오브젝트에서 해당 속성을 상속합니다.

일치하는 토픽 오브젝트에 설정되지 않은 특성은 일반적인 방식으로 SYSTEM.BASE.TOPIC에서 상속됩니다.

MQRFH2 메시지 헤더를 사용하는 기존의 큐된 발행/구독 애플리케이션은 Publish 또는 Register subscriber 명령 메시지에서 **SubPoint** 특성을 설정하여 계속 작동합니다. 구독 지점은 명령 메시지에서 토픽 문자열과 결합되며 결과로 생성되는 토픽은 다른 것과 마찬가지로 처리됩니다.

IBM MQ 애플리케이션은 구독 지점의 영향을 받지 않습니다. 애플리케이션이 일치하는 토픽 오브젝트 중 하나에서 상속되는 토픽을 사용하는 경우, 해당 애플리케이션은 일치하는 구독 지점을 사용하여 큐된 애플리케이션과 상호 운용됩니다.

예

기존 WebSphere Message Broker (현재는 IBM Integration Bus 로 알려짐) IBM MQ 로 마이그레이션된 발행/구독 애플리케이션이 해당 토픽 문자열 'GBP' 및 'USD'를 사용하여 두 개의 토픽 오브젝트 GBP 및 USD를 작성했습니다.

구독 지점 USD를 사용하며 IBM MQ에서 실행되도록 마이그레이션된 토픽 NYSE/IBM/SPOT에 대한 기존 발행자는 토픽 USD/NYSE/IBM/SPOT에 대한 발행물을 작성합니다. 마찬가지로 구독 지점 USD를 사용하는 NYSE/IBM/SPOT에 대한 기존 구독자는 USD/NYSE/IBM/SPOT에 대한 구독을 작성합니다.

MQSUB를 호출하여 IBM MQ 발행/구독 프로그램에서 달러 현물 가격을 구독하십시오. 'C' 코드 단편에서 보여준 대로 USD 토픽 오브젝트 및 토픽 문자열 'NYSE/IBM/SPOT'을 사용하여 구독을 작성하십시오.

```
strncpy(sd.ObjectName, "USD", MQ_TOPIC_NAME_LENGTH);
sd.ObjectString.VSPtr = "NYSE/IBM/SPOT";
sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
```

1. 클러스터 토픽 호스트에서 USD 및 GBP 토픽 오브젝트의 CLUSTER 속성을 설정하십시오.
2. 클러스터의 다른 큐 관리자에서 USD 및 GBP 토픽 오브젝트의 모든 사본을 삭제하십시오.
3. USD 및 GBP가 클러스터의 모든 큐 관리자에 있는 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST에 정의되었는지 확인하십시오.

단일 큐 관리자 발행/구독 구성 예제

78 페이지의 그림 31에서는 기본적인 단일 큐 관리자 발행/구독 구성을 보여줍니다. 예제에서는 뉴스 서비스에 대한 구성을 보여줍니다. 여기에서 발행자로부터 여러 토픽에 대한 정보를 사용할 수 있습니다.

- 발행자 1은 Sport 토픽을 사용하여 스포츠 결과에 대한 정보를 발행함
- 발행자 2는 Stock 토픽을 사용하여 주가에 대한 정보를 발행함
- 발행자 3은 Films 토픽을 사용하여 영화 비평에 대한 정보를 발행하고 TV 토픽을 사용하여 TV 프로그램에 대한 정보를 발행함

3명의 구독자는 서로 다른 토픽을 관심사로 등록했으므로 큐 관리자는 관심이 있는 정보를 해당 구독자에게 송신합니다.

- 구독자 1은 스포츠 결과와 주가를 수신함
- 구독자 2는 영화 비평을 수신함
- 구독자 3은 스포츠 결과를 수신함

TV 프로그램을 관심사로 등록한 구독자는 없으므로 이는 배포되지 않습니다.

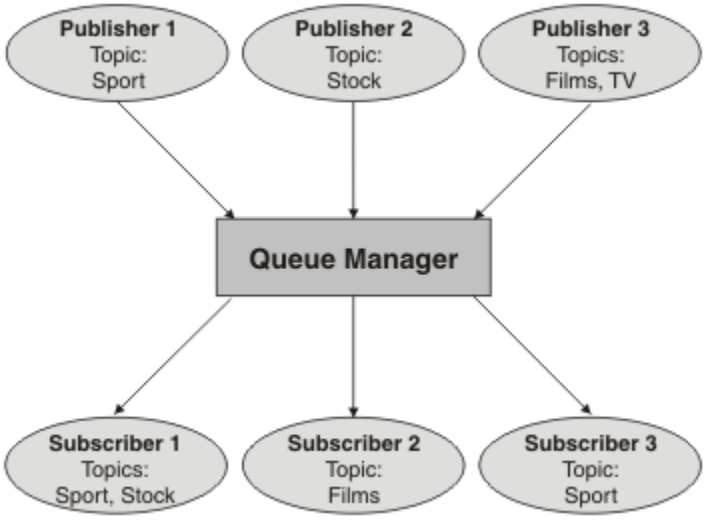


그림 *31.* 단일 큐 관리자 발행/구독 예제

분산 발행/구독 네트워크

각 큐 관리자는 해당 토픽에 가입한 로컬로 구성된 구독을 가진 토픽에 게시된 메시지와 일치합니다. 한 큐 관리자에 연결된 애플리케이션에 의해 발행된 메시지가 네트워크에서 다른 큐 관리자에 작성된 일치하는 구독으로 전달되도록 큐 관리자의 네트워크를 구성할 수 있습니다. 이를 위해서는 큐 관리자 간에 단순 채널에 대한 추가 구성이 필요합니다.

배포된 발행/구독 구성은 함께 연결된 일련의 큐 관리자입니다. 큐 관리자는 모두 동일한 물리적 시스템에 있거나 몇몇 물리적 시스템에 분산되어 있을 수 있습니다. 함께 큐 관리자를 연결하는 경우, 구독자는 하나의 큐 관리자를 구독하여 다른 큐 관리자에게 초기에 발행된 메시지를 수신할 수 있습니다. 이를 설명하기 위해 다음 그림에서는 두 번째 큐 관리자를 78 페이지의 『단일 큐 관리자 발행/구독 구성 예제』에 설명된 구성에 추가합니다.

- 발행자 4는 큐 관리자 2를 사용하여 일기 예보 정보(날씨 토픽 사용) 및 주요 도로의 트래픽 상황에 대한 정보(트래픽 토픽 사용)를 발행합니다.
- 구독자 4도 이 큐 관리자를 사용하며, 트래픽 토픽을 사용하여 트래픽 상황에 대한 정보를 구독합니다.
- 구독자 3은 발행자와 다른 큐 관리자를 사용하긴 하지만 역시 날씨 상황에 대한 정보를 구독합니다. 큐 관리자가 서로 링크되어 있기 때문에 가능합니다.

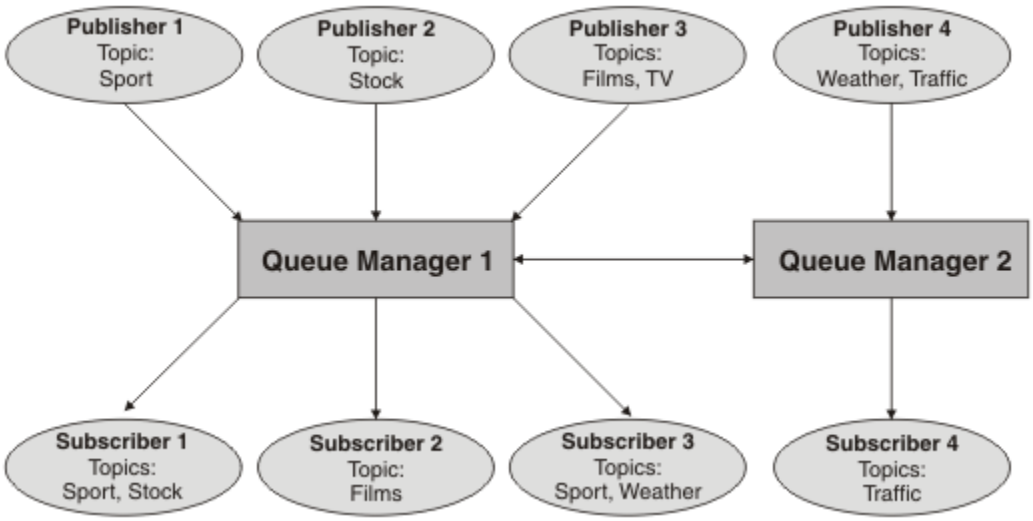


그림 *32.* 큐 관리자가 *2*개인 발행/구독 예제

상위 및 하위 계층에서 큐 관리자를 수동으로 연결하거나 발행/구독 클러스터를 작성하고 IBM MQ에서 대부분의 연결 세부사항을 정의하도록 할 수 있습니다. 또한 계층에서 여러 클러스터를 함께 조인하여 두 토폴로지를 함께 결합하여 사용할 수 있습니다.

발행/구독 클러스터 개요

발행/구독 클러스터는 클러스터에 추가된 하나 이상의 토픽 오브젝트가 있는 표준 클러스터입니다. 클러스터의 큐 관리자에 관리 토픽 오브젝트를 정의하고 클러스터 이름을 지정하여 클러스터된 해당 토픽 오브젝트를 작성할 경우, 해당 토픽에 대한 발행자 및 구독자는클러스터의 큐 관리자에 연결할 수 있으며 발행된 메시지는 큐 관리자 간의 클러스터 채널을 통해 구독자로 라우팅됩니다.

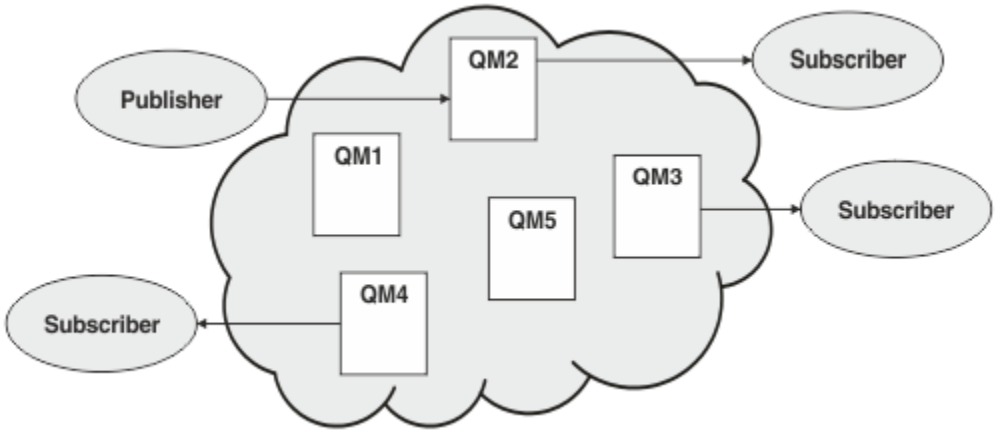


그림 *33.* 발행/구독 클러스터

발행/구독 메시지가 클러스터에서 라우팅되는 방법을 구성하는 데는 2가지 방법이 있습니다.

- 직접 라우팅(direct routing)
- 토픽 호스트 라우팅(topic host routing)

직접 라우팅된 클러스터 토픽을 구성할 경우, 한 큐 관리자에 발행된 메시지는 직접 해당 큐 관리자에서 클러스터의 다른 큐 관리자에 있는 모든 구독으로 보내집니다. 이는 구독에 대한 가장 직접적인 경로를 제공할 수 있지만 실제로는 클러스터의 모든 큐 관리자가 모든 다른 큐 관리자를 인식하게 되며 잠재적으로 이들 간에 설정된 클러스터 채널을 가지게 됩니다.

토픽 호스트 라우팅을 사용할 경우, 한 큐 관리자에 발행된 메시지는 관리 토픽 오브젝트의 정의를 호스팅하는 큐 관리자로 보내집니다. 토픽 호스트 큐 관리자는 클러스터의 다른 큐 관리자에 있는 모든 구독에 메시지를 라우팅합니다. 발행자 또는 구독자가 토픽 호스트 큐 관리자에 없는 경우, 발행에 대한 라우트가 더 길어집니다. 그러나 이점은 토픽 호스트 큐 관리자가 클러스터의 모든 다른 큐 관리자를 인식하게 되므로 잠재적으로 이와 함께 설정된 클러스터 채널을 갖게 된다는 점입니다.

자세한 정보는 81 페이지의 『발행/구독 클러스터』의 내용을 참조하십시오.

발행/구독 계층 개요

발행/구독 계층은 계층로 채널에 의해 연결된 일련의 큐 관리자입니다. 각 큐 관리자는 발행/구독 계층에 큐 관리자 연결에 설명된 대로 해당 상위 큐 관리자를 식별합니다.

토픽에 대한 발행자 및 구독자는 계층에서 큐 관리자에 연결할 수 있으며 메시지는 계층 큐 관리자 연결성을 사용하여 이들 간을 플로우됩니다.

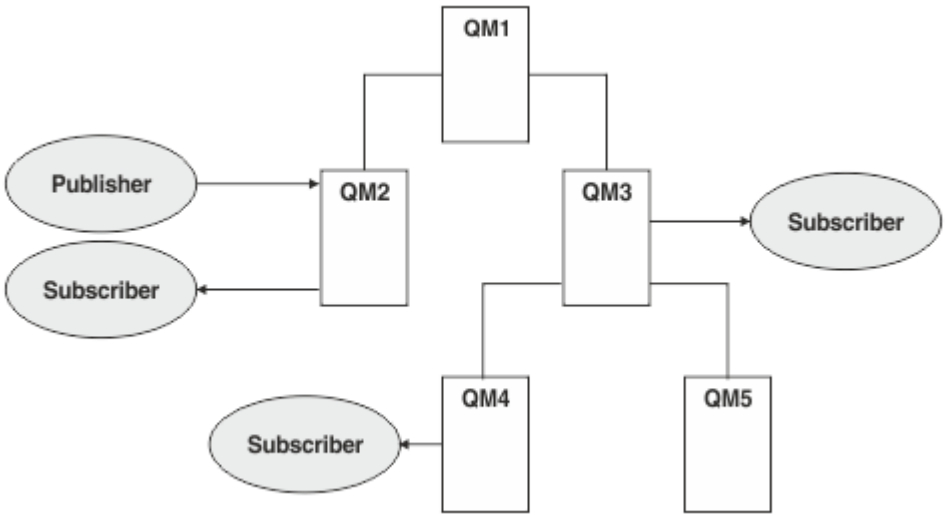


그림 *34.* 발행/구독 계층

앞의 그림에서 QM3 및 QM4의 구독자에 전달된 발행은 QM2에서 QM1로,그 다음은 QM3으로, 그리고 마지막으로 QM4로 라우팅되었습니다.

계층을 통해 사용자는 계층 내의 모든 큐 관리자 간의 관계를 직접 제어할 수 있습니다. 이는 발행자로부터 구독자로의 메시지 라우팅에 대한 보다 상세한 제어를 허용하며, 특히 제한된 연결성을 가진 큐 관리자 네트워크 간의 라우팅 시에 더 유용합니다. 메시지를 발행자에서 구독자로 라우팅하는 방법을 통해 모든 큐 관리자의 가용성과 기능을 신중하게 고려해야 합니다.

자세한 정보는 83 페이지의 『발행/구독 계층』의 내용을 참조하십시오.

큐 관리자 간의 발행 분배

라우팅 선택 외에도 큐 관리자의 네트워크를 통해 발행을 분배하기 위한 두 가지 접근법이 있습니다.

- 한 큐 관리자에서 해당 발행에 대한 구독을 현재 호스팅하는 큐 관리자에게만 발행을 송신합니다.

- 모든 큐 관리자에 각 발행을 송신하고 해당 구독에 대해 일치하도록 합니다.

전자는 발행 메시지가 필요한 곳으로만 보내지도록 하지만 큐 관리자 간에 구독 알림 레벨을 공유해야 합니다. 후자는 구독 알림을 공유할 필요는 없지만 불필요한 발행 메시지가 큐 관리자 간에 보내질 수 있습니다.

기본적으로 IBM MQ는 전자의 메소드를 사용하며, 여기서는 발행이 이에 대한 구독을 가진 큐 관리자로만 보내집니다. 구독 알림은 프록시 구독 양식으로 큐 관리자 간에 전파됩니다. 이는 배포된 발행/구독 토폴로지에서 사용하기에 가장 효율적이이므로 구독의 배포와 수명 주기, 발행의 빈도에 따라 달라집니다. 발행/구독 네트워크에서의 구독 성능을 참조하십시오.

관련 개념

67 페이지의 『토픽 트리』
정의한 각 토픽은 요소 또는 토픽 트리의 노드입니다. 토픽 트리는 비워 두거나 MQSC 또는 PCF 명령을 사용하여 이미 정의된 토픽으로 시작되거나 해당 토픽을 포함할 수 있습니다. 토픽 작성 명령을 사용하거나 발행 또는 구독에 토픽을 처음으로 지정하면 새 토픽을 정의할 수 있습니다.

발행/구독 계층 시나리오

관련 태스크

발행/구독 클러스터 디자인

발행/구독 클러스터

발행/구독 클러스터는 발행이 발행 애플리케이션에서 클러스터의 큐 관리자에 존재하는 구독으로 자동 이동되는 상호 연결된 큐 관리자의 표준 클러스터입니다. 발행/구독 클러스터에 대한 발행물 라우팅에는 직접 라우팅과 토픽 호스트 라우팅이라는 두 가지 옵션이 있습니다. 사용자가 선택하는 라우팅은 사용자 클러스터에 대한 크기와 예상 활동 패턴에 따라 달라집니다.

메시징 발행/구독에 사용되는 클러스터는 표준 IBM MQ 클러스터와 다르지 않습니다. 마찬가지로 발행/구독 클러스터 내 큐 관리자는 실제로 별도의 컴퓨터에 존재할 수 있으며, 큐 관리자의 각 쌍은 필요한 경우 클러스터 채널에 의해 자동으로 함께 연결됩니다. 자세한 정보는 클러스터를 참조하십시오.

발행/구독 메시징에 대한 큐 관리자의 표준 클러스터를 구성하기 위해 클러스터의 큐 관리자에 하나 이상의 관리 토픽 오브젝트를 정의합니다. 토픽을 클러스터 토픽으로 설정하려는 경우, 클러스터의 이름이 있는 **CLUSTER** 특성을 구성합니다. 이를 수행할 경우, 토픽 트리의 해당 지점 또는 그 아래에 있는 발행자 또는 구독자에서 사용된 토픽이 클러스터의 모든 큐 관리자 간에 공유되며 토픽 트리의 클러스터 분기에 발행된 메시지는 클러스터의 다른 큐 관리자에 대한 구독으로 자동 라우팅됩니다.

대상 큐 관리자에 있는 메시지에 대한 구독자 수에 관계없이 각 메시지의 사본 하나만 발행자 큐 관리자와 각각 다른 큐 관리자 간에 보내집니다. 하나 이상의 구독을 가진 큐 관리자에 도착할 경우 메시지가 모든 구독에 대해 중복됩니다.

클러스터에 조인하는 큐 관리자는 자동으로 클러스터된 토픽을 알게 되며 해당 큐 관리자의 발행자와 구독자는 자동으로 클러스터에 참여합니다.

클러스터된 토픽 오브젝트에 속하지 않은 토픽 문자열에 대한 작업을 수행하면 클러스터되지 않은 발행/구독 활동이 발행/구독 클러스터에서 수행될 수도 있습니다.

발행/구독 클러스터에 대한 발행물 라우팅에는 직접 라우팅과 토픽 호스트 라우팅이라는 두 가지 옵션이 있습니다. 클러스터에서 사용할 메시지 라우팅을 선택하려면 관리 토픽 오브젝트의 **CLROUTE** 특성을 다음 값 중 하나로 설정합니다.

- **DIRECT**
- **TOPICHOST**

기본적으로 토픽 라우팅은 **DIRECT**입니다. IBM MQ 8.0 이전에는 이 옵션이 유일한 옵션이었습니다. 큐 관리자에서 직접 라우트되는 클러스터 토픽을 구성하는 경우, 클러스터의 모든 큐 관리자는 클러스터의 다른 모든 큐 관리자를 인식하게 됩니다. 따라서 발행 및 구독 조작을 수행할 경우 큐 관리자가 각각 클러스터에 있는 다른 큐 관리자에 직접 연결될 수 있습니다.

IBM MQ 8.0부터는 대신 토픽 라우팅을 **TOPICHOST**로 구성할 수 있습니다. 토픽 호스트 라우팅을 사용할 경우, 클러스터의 모든 큐 관리자가 라우팅되는 토픽 정의를 호스팅하는 클러스터 큐 관리자(토픽 오브젝트를 정의한 큐 관리자)를 인식하게 됩니다. 발행 및 구독 조작을 수행할 경우, 클러스터의 큐 관리자는 서로 직접 연결되지 않

고 이러한 토픽 호스트 큐 관리자에만 연결됩니다. 토픽 호스트 큐 관리자는 구독이 일치하는 큐 관리자에 발행물을 발행하는 큐 관리자에서 발행물을 라우팅하는 작업을 담당합니다.

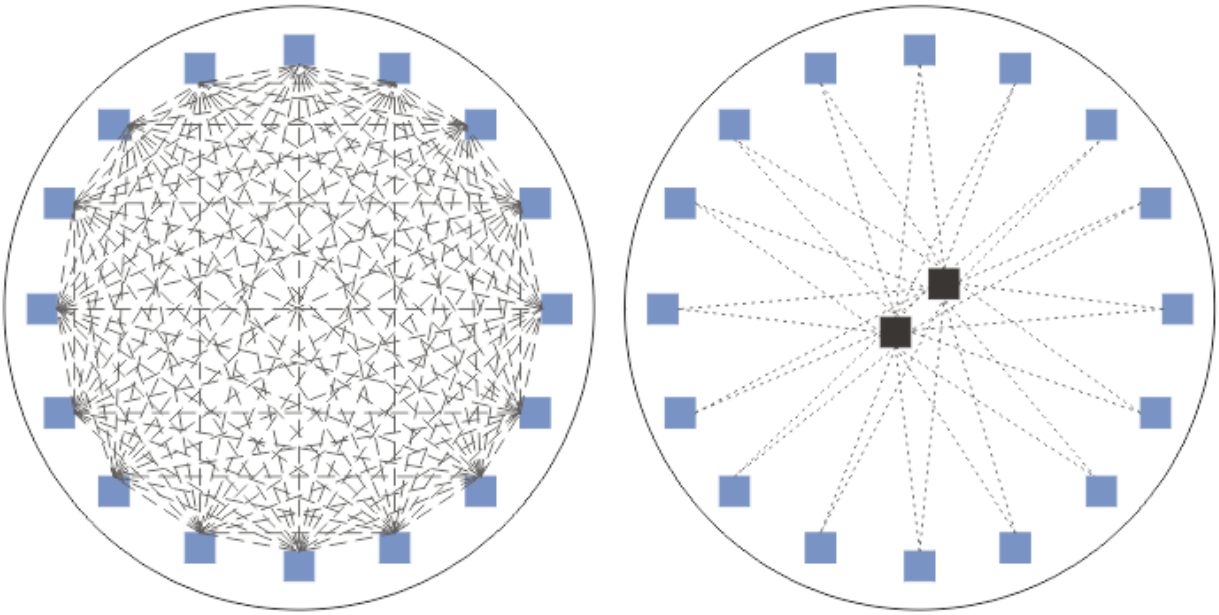


그림 *35.* 직접 라우팅 및 토픽 호스트 라우팅

직접 라우팅 개요

관리 토픽 오브젝트가 직접 라우팅용으로 구성된 경우 이에 대해 파악하려면 모든 큐 관리자에 대해 클러스터의 큐 관리자 중 하나에 토픽 오브젝트를 정의하기만 하면 됩니다. 토픽이 정의되는 큐 관리자 선택은 토픽에 대한 발행/구독 메시징의 작동에는 영향을 주지 않습니다.

각 메시지는 중간 큐 관리자를 통해 전달되는 것이 아니라 발행자 큐 관리자로부터 클러스터의 다른 큐 관리자에 있는 각 구독으로 직접 플로우됩니다.

기본적으로, 메시지는 하나 이상의 구독을 호스팅하는 클러스터의 다른 큐 관리자로만 송신됩니다.

- 이는 현재 하나 이상의 구독이 있는 모든 토픽의 클러스터에 있는 모든 다른 큐 관리자에게 직접 알리는 각 큐 관리자에 달려 있습니다. 결과적으로 클러스터의 모든 큐 관리자는 구독 중인 모든 토픽을 알게 되며 구독을 호스팅하는 큐 관리자는 다른 모든 큐 관리자에 채널을 설정하게 됩니다. 이는 각 큐 관리자가 발행자를 갖고 있는지와는 별개입니다.
- 모든 큐 관리자의 각 개별 구독 토픽에 대한 알림은 구독이 있는지에 상관없이 모든 발행을 클러스터의 모든 큐 관리자에 보내는 모델로 변경함으로써 제거할 수 있습니다. 이는 구독 알림 트래픽을 줄여주지만 발행 트래픽과 각 큐 관리자가 설정하는 채널 수는 늘어날 수 있습니다. 발행/구독 네트워크에서의 구독 성능을 참조하십시오.

직접 라우팅된 클러스터 토픽을 사용하는 발행/구독 메시지는 각 클러스터의 한 큐 관리자를 발행/구독 계층에 추가함으로써 다중 발행/구독 클러스터를 확장할 수 있습니다. 다중 클러스터의 토픽 공간 결합을 참조하십시오.

직접 라우팅에 대한 자세한 탐색을 위해서는 발행/구독 클러스터의 직접 라우팅을 참조하십시오.

토픽 호스트 라우팅 개요

관리 토픽 오브젝트가 토픽 호스트 라우팅용으로 구성된 경우, 클러스터의 큐 관리자로부터의 발행은 토픽 오브젝트가 구성된 큐 관리자("토픽 호스트")를 통해 구독이 존재하는 큐 관리자로 라우팅됩니다.

- 이는 현재 하나 이상의 구독이 있는 모든 토픽의 모든 토픽 호스트를 알리는 각 큐 관리자에 달려있습니다. 구독을 호스팅하는 큐 관리자는구독이 관련되어 있는 토픽에 대한 모든 토픽 호스트에 채널을 설정합니다.
- 비토픽 호스팅 큐 관리자는 발행/구독을 목적으로 클러스터에서 다른 비토픽 호스팅 큐 관리자를 인식하도록 되어 있지 않으며, 채널이 이 목적을 위해 이들 간에 설정되지 않습니다.

- 발행 애플리케이션이 해당 토픽을 호스팅하는 큐 관리자에 연결되어 있는 경우, 발행된 메시지는 추가 '홉(Hop)' 없이도 일치하는 구독이 작성된 큐 관리자에 직접 라우팅됩니다. 마찬가지로 일치하는 구독이 토픽을 호스팅하는 큐 관리자에서만 작성되면 해당 토픽을 구독한 메시지는 추가 홉 없이 큐 관리자에 직접 라우팅됩니다.
- 발행자와 동일한 큐 관리자에 있는 구독은 토픽 오브젝트의 호스트에 대한 첫 발행물 라우팅 없이도 충족됩니다.

클러스터된 큐의 경우 다중 큐 관리자가 동일한 관리 토픽 오브젝트를 구성할 수 있습니다. 이는 더 높은 메시지 라우팅 가용성과 워크로드 밸런스를 통한 수평적 확장을 제공합니다. 토픽 오브젝트를 라우팅한 토픽 호스트의 경우, 다중 큐 관리자가 토픽 트리의 동일한 분기에 대해 동일한 이름의 토픽을 구성할 경우, 각 토픽 호스트는 구독을 호스팅하는 모든 큐 관리자에 의해 구독되는 토픽을 파악할 수 있게 됩니다.

- 메시지가 발행되면 이는 큐 관리자를 호스팅하는 구독에 전달하기 위해 토픽 호스트 큐 관리자 중 하나로 보내집니다. 토픽 호스트 큐 관리자의 선택은 클러스터된 포인트-투-포인트 큐에 대해서와 동일한 기본 워크로드 밸런스 규칙을 따릅니다.
- 하나 이상의 토픽 호스트 큐 관리자를 발행 큐 관리자에 의해 연결하지 못할 경우 메시지가 남아 있는 사용 가능한 토픽 호스팅 큐 관리자에 라우팅됩니다.

토픽 트리의 라우팅된 분기에 있는 토픽에 대한 모든 발행은 클러스터에 해당 토픽에 대한 구독이 없는 경우에도 토픽 호스트 중 하나로 전달됩니다. 기본적으로, 메시지는 여기에서 하나 이상의 구독을 호스팅하는 클러스터의 다른 큐 관리자로만 송신됩니다.

- 이는 클러스터의 각 큐 관리자에서 모든 구독된 토픽 문자열에 대한 알림을 받는 각 토픽 호스트 큐 관리자에 따라 달라집니다.
- 각 개별 구독 토픽에 대한 알림은 구독이 있는지에 상관없이 토픽 호스트에 대해 라우팅된 모든 구독을 클러스터의 모든 큐 관리자에 보내는 모델로 변경함으로써 제거할 수 있습니다. 이는 구독 알림 트래픽을 줄여주지만 발행 트래픽과 잠재적으로 각 토픽 호스팅 큐 관리자와 설정된 채널 수는 늘어날 수 있습니다. 발행/구독 네트워크에서의 구독 성능을 참조하십시오.

클러스터된 토픽을 라우팅한 토픽 호스트를 사용한 발행/구독 메시지 플로우는 발행/구독 계층의 사용을 통해 다중 발행/구독 클러스터를 확장할 수 **없습니다**.

토픽 호스트 라우팅에 대한 자세한 탐색을 위해서는 발행/구독 클러스터의 토픽 호스트 라우팅을 참조하십시오.

발행/구독 계층

채널을 사용하여 큐 관리자를 함께 링크하고 큐 관리자 쌍 간의 하위-상위 관계를 정의하여 발행/구독 계층을 빌드합니다. 메시지는 계층에서 직접 관계를 통해 발행자에서 구독으로 플로우됩니다. 이는 다중 "홉"을 의미할 수 있습니다.

대상 큐 관리자의 메시지에 대한 구독자 수에 상관없이 큐 관리자의 한 쌍 간에는 하나의 메시지 사본만 보내집니다. 하나 이상의 구독을 가진 큐 관리자에 도착할 경우 메시지가 모든 구독에 대해 중복됩니다.

기본적으로, 메시지는 다른 큐 관리자의 구독에 대한 라우트에 있는 계층의 다른 큐 관리자로만 송신됩니다.

- 이는 이 큐 관리자 또는 다른 관계 중 하나에서 현재 하나 이상의 구독을 갖고 있는 모든 토픽의 각 직접 관계를 알리는 각 큐 관리자에 따라 달라집니다. 이렇게 되면 계층의 모든 큐 관리자가 모든 토픽이 구독 중임을 알게 됩니다.
- 이 작동은 존재하는 구독에 상관없이 계층의 모든 큐 관리자에게 항상 구독을 보내도록 변경될 수 있습니다. 이렇게 되면 계층에 걸쳐 구독 정보를 전파할 필요는 없어지지만 구독 트래픽이 늘어날 수 있습니다.

클러스터를 작성할 경우 메시지가 네트워크 내에서 영원히 순환하게 되는 루프를 작성하지 않도록 주의해야 합니다. 이러한 루프는 계층으로 작성할 수 없습니다.

모든 큐 관리자는 고유 큐 관리자 이름을 가져야 합니다.

발행/구독 메시지 플로우는 다중 발행/구독 클러스터로 확장될 수 있습니다. 이를 수행하려면 각 클러스터에서 하나의 큐 관리자를 발행/구독 계층에 추가하십시오.

보다 자세한 탐색을 위해 발행/구독 계층에서 라우팅을 참조하십시오.

발행/구독 네트워크에서의 프록시 구독

프록시 구독은 하나의 큐 관리자에 발행된 토픽에 대해 다른 큐 관리자가 작성하는 구독입니다. 프록시 구독은 구독에서 구독하는 각 개별 토픽 문자열에 대해 큐 관리자 사이에서 플로우됩니다. 사용자는 프록시 구독을 명확하게 작성하지 않지만 큐 관리자는 사용자를 위해 프록시 구독을 명확하게 작성합니다.

발행/구독 클러스터 또는 발행/구독 계층으로 함께 큐 관리자를 연결할 수 있습니다. 프록시 구독은 연결된 큐 관리자 사이에서 전달됩니다. 프록시 구독의 경우 한 큐 관리자에 연결된 발행자가 작성한 토픽에 대한 발행은 다른 큐 관리자에 연결된 해당 토픽에 대한 구독에서 수신됩니다. 78 페이지의 『분산 발행/구독 네트워크』의 내용을 참조하십시오.

개별 토픽 문자열에 대한 수천 개의 구독을 포함하는 발행/구독 토폴로지 또는 이러한 구독의 존재 여부가 빠르게 변화하는 환경에서 프록시 구독 전파의 오버헤드를 고려해야 합니다. 이 토픽의 나머지에서 설명된 자동 집계 외에도 연결된 큐 관리자 간에 프록시 구독 및 발행의 플로우를 추가로 제한하고 연결된 모든 큐 관리자에 프록시 구독을 전파하기 위해 대기하는 지연 시간을 줄이는 수동 구성 변경을 작성할 수 있습니다. 발행/구독 네트워크에서의 구독 성능을 참조하십시오.

프록시 구독에는 로컬 구독에 사용되는 선택자가 포함되지 않으며, 와일드카드가 포함된 구독 토픽 문자열이 간소화될 수 있습니다. 이는 큐 관리자 간의 추가 발행 플로우가 발생하여 실제 구독이 없는 프록시 구독과 일치하는 발행이 발생할 수 있습니다. 구독을 호스팅하는 큐 관리자는 추가 발행이 구독으로 리턴되지 않도록 불일치를 필터링합니다.

프록시 구독 집계

프록시 구독은 중복 제거 시스템을 사용하여 집계됩니다. 해석된 특정 토픽 문자열의 경우 프록시 구독은 수신된 프록시 구독 또는 첫 번째 로컬 구독에서 전송됩니다. 동일한 토픽 문자열에 대한 후속 구독은 이 기존 프록시 구독을 사용합니다.

마지막 로컬 구독 후에 프록시 구독이 취소되거나 수신된 프록시 구독이 취소됩니다.

발행 집계

큐 관리자에 동일한 토픽 문자열에 대한 구독이 둘 이상인 경우 해당 토픽 문자열과 일치하는 각 발행의 사본 하나만 발행/구독 토폴로지의 다른 큐 관리자에서 전송됩니다. 메시지가 도착하면 로컬 큐 관리자는 일치하는 각 구독에 메시지의 사본을 전달합니다.

프록시 구독이 와일드카드를 포함하는 경우 단일 발행의 토픽 문자열과 둘 이상의 구독이 일치할 수 있습니다. 메시지가 하나의 연결된 큐 관리자에서 작성한 둘 이상의 프록시 구독과 일치하는 큐 관리자에서 발행되는 경우 발행의 사본 하나만 리모트 큐 관리자로 전달되어 다중 프록시 구독을 만족시킵니다.

관련 개념

분산 발행/구독 네트워크의 루프 감지

프록시 구독의 와일드카드

구독은 발행에서 다중 토픽 문자열에 대해 일치하도록 토픽 문자열에서 와일드카드를 사용할 수 있습니다.

구독이 사용할 수 있는 와일드카드 스키마는 *topic-based*와 *character-based*의 두 가지입니다. 61 페이지의 『와일드카드 설계』의 내용을 참조하십시오.

IBM MQ 에서 와일드카드 구독에 대한 모든 프록시 구독은 토픽 기반 와일드카드를 사용하도록 변환됩니다. 문자 기반 와일드카드가 나오면, 가장 가까운 / 뒤가 # 문자로 바뀝니다. 예를 들어 `/aaa/bbb/c*d`는 `/aaa/bbb/#`으로 변환됩니다. 이러한 변환으로 인해 리모트 큐 관리자는 명시적으로 구독하는 것보다 약간 많은 발행을 송신합니다. 추가 발행은 로컬 구독자로 발행을 전달할 때 로컬 큐 관리자로 필터링됩니다.

WILDCARD 특성으로 와일드카드 사용 제어

MQSC **Topic** WILDCARD 특성 또는 동등한 PCF 토픽 `WildcardOperation` 특성을 사용하여 와일드카드 토픽 문자열 이름을 사용하는 구독자 애플리케이션으로 발행물의 전달을 제어할 수 있습니다. WILDCARD 특성은 다음과 같은 두 가지 가능한 값을 보유할 수 있습니다.

WILDCARD
이 토픽에 관한 와일드카드 구독의 동작입니다.

PASSTHRU
이 토픽 오브젝트의 토픽 문자열보다 덜 특정한 와일드카드 토픽에 대한 구독이 이 토픽 및 이 토픽보다 더욱 특정한 토픽 문자열에 대한 발행물을 수신합니다.

BLOCK
이 토픽 오브젝트의 토픽 문자열보다 덜 특정한 와일드카드 토픽에 대한 구독이 이 토픽 또는 이 토픽보다 더욱 특정한 토픽 문자열에 대한 발행물을 수신하지 않습니다.

이 속성의 값은 구독이 정의될 때 사용됩니다. 이 속성을 대체할 경우 기존 구독에 포함된 토픽 세트는 수정의 영향을 받지 않습니다. 이 시나리오는 토픽 오브젝트를 작성 또는 삭제할 때 토폴로지가 변경된 경우에도 적용됩니다. WILDCARD 속성을 수정한 후에 작성된 구독과 일치하는 토픽 세트가 수정된 토폴로지를 사용하여 작성됩니다. 일치하는 토픽 세트를 강제로 기존 구독에 대해 재평가하려는 경우 큐 관리자를 재시작해야 합니다.

예제, 73 페이지의 『예제: Sport 발행/구독 클러스터 작성』 에서는 70 페이지의 그림 23에 표시된 토픽 트리 구조를 작성하는 단계를 수행할 수 있습니다.

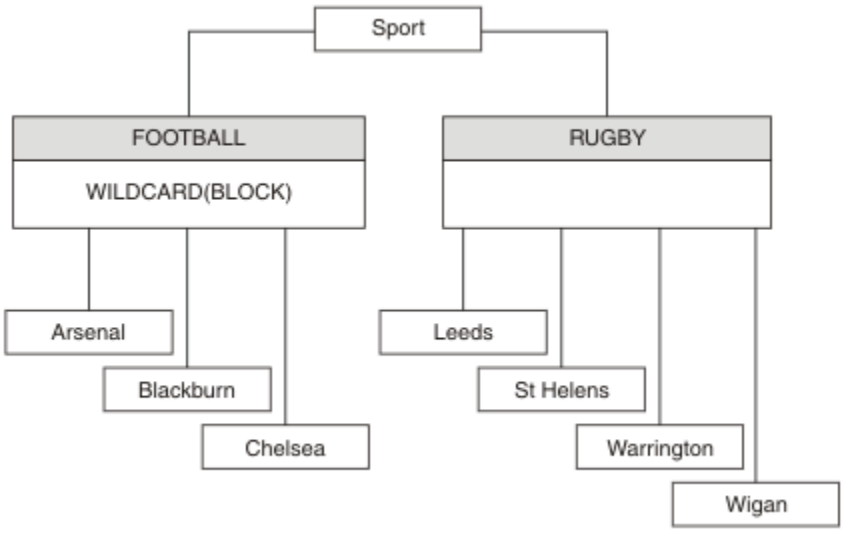


그림 *36. WILDCARD* 특성, *BLOCK*을 사용하는 토픽 트리

와일드카드 토픽 문자열 #을 사용하는 구독자는 Sport 토픽 및 Sport/Rugby 하위 트리에 대한 모든 발행을 수신합니다. 구독자는 Sport/Football 하위 트리에 대한 발행을 수신하지 않습니다. Sport/Football 토픽의 WILDCARD 특성 값은 BLOCK입니다.

PASSTHRU는 기본 설정입니다. Sport 트리의 노드에서 WILDCARD 특성 값 PASSTHRU를 설정할 수 있습니다. 노드에 WILDCARD 특성 값 BLOCK이 없으면 PASSTHRU를 설정해도 Sports 트리의 노드에서 구독자가 관찰하는 작동은 대체되지 않습니다.

예제에서 구독을 작성하여 와일드카드 설정이 전달되는 발행에 미치는 영향을 확인하십시오(75 페이지의 그림 27 참조). 일부 발행을 작성하려면 76 페이지의 그림 30에서 발행 명령을 실행하십시오.

```
pub QMA
```

그림 *37. QMA*에 발행

결과는 71 페이지의 표 3에 표시됩니다. WILDCARD 특성 값 BLOCK을 설정할 때 와일드카드를 포함하는 구독이 와일드카드 범위 내 토픽에 대한 발행을 수신하지 않도록 하는 방법에 주의하십시오.

표 6. QMA에서 수신된 발행			
구독	**토픽 문자열**	**수신된 발행**	**참고**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football에서 WILDCARD(BLOCK)으로 차단된 Football 하위 트리에 대한 모든 발행
SARSENAL	Sports/#/Arsenal	-	Sports/Football에서 WILDCARD(BLOCK)은 Arsenal에서의 와일드카드 구독을 금지함
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby의 기본 WILDCARD는 Leeds에서 와일드카드 구독을 금지하지 않습니다.

참고:

구독에 WILDCARD 특성 값 BLOCK이 있는 토픽 오브젝트와 일치하는 와일드카드가 있다고 가정합니다. 또한 구독이 일치하는 와일드카드 오른쪽에 토픽 문자열을 포함하면 구독은 발행을 수신하지 않습니다. 차단되지 않은 발행 세트는 차단된 와일드카드 상위에 해당하는 토픽에 대한 발행입니다. BLOCK 특성 값이 설정된 토픽의 하위에 해당하는 토픽에 대한 발행은 와일드카드로 차단됩니다. 따라서 와일드카드 오른쪽에 토픽을 포함하는 구독 토픽 문자열은 일치하는 발행을 수신하지 않습니다.

WILDCARD 특성 값을 BLOCK으로 설정해도 와일드카드를 포함하는 토픽 문자열을 사용하여 구독할 수 없음을 의미하지는 않습니다. 이러한 구독은 정상적입니다. 구독이 WILDCARD 특성 값 BLOCK이 설정된 토픽 오브젝트를 포함하는 토픽과 일치하는 명시적 토픽을 보유합니다. 이는 WILDCARD 특성 값 BLOCK이 설정된 토픽의 상위 또는 하위에 해당하는 토픽에 대한 와일드카드를 사용합니다. 70 페이지의 그림 23의 예제에서 Sports/Football/#과 같은 구독은 발행을 수신할 수 있습니다.

와일드카드 및 클러스터 토픽

클러스터 토픽 정의는 클러스터의 모든 큐 관리자로 전파됩니다. 클러스터의 한 큐 관리자에서 클러스터 토픽에 대한 구독으로 인해 큐 관리자가 프록시 구독을 작성할 수 있습니다. 프록시 구독은 클러스터의 다른 모든 큐 관리자에서 작성됩니다. 클러스터 토픽과 결합해 와일드카드를 포함하는 토픽 문자열을 사용하는 구독은 작동을 예상하기 어려울 수 있습니다. 작동은 다음 예제에서 설명됩니다.

예제, 73 페이지의 『예제: Sport 발행/구독 클러스터 작성』에 설정된 클러스터의 경우 QMB에는 QMA와 동일한 구독 세트가 있지만 QMB는 발행자가 QMA에 발행한 후 발행을 수신하지 않습니다(71 페이지의 그림 24 참조). Sports/Football 및 Sports/Rugby 토픽이 클러스터 토픽이어도 fullsubs.tst에 정의된 구독은 클러스터 토픽을 참조하지 않습니다. 프록시 구독은 QMB에서 QMA로 전파되지 않습니다. 프록시 구독이 없으면 QMA에 대한 발행은 QMB로 전달되지 않습니다.

Sports/#/Leeds와 같은 일부 구독은 이 경우 Sports/Rugby에 해당하는 클러스터 토픽을 참조할 수도 있습니다. Sports/#/Leeds 구독은 실제로 토픽 오브젝트 SYSTEM.BASE.TOPIC으로 해석됩니다.

Sports/#/Leeds와 같이 구독에서 참조하는 토픽 오브젝트를 해석하는 규칙은 다음과 같습니다. 토픽 문자열을 첫 번째 와일드카드까지 자릅니다. 토픽 문자열을 왼쪽부터 스캔하여 연관된 관리 토픽 오브젝트가 있는 첫 번째 토픽을 찾습니다. 이 토픽 오브젝트는 클러스터 이름을 지정하거나 로컬 토픽 오브젝트를 정의할 수 있습니다. 예 Sports/#/Leeds에서 잘림 이후의 토픽 문자열은 토픽 오브젝트가 없는 Sports이므로 Sports/#/Leeds 는 로컬 토픽 오브젝트인 SYSTEM.BASE.TOPIC에서 상속합니다.

클러스터 토픽을 구독하여 와일드카드 전파 방식을 변경하는 방법을 확인하려면 배치 스크립트, upsubs.bat를 실행하십시오. 스크립트는 구독 큐를 지우고 fullsubs.tst에서 클러스터 토픽 구독을 추가합니다. puba.bat를 다시 실행하여 발행 배치를 작성하십시오(71 페이지의 그림 24 참조).

72 페이지의 표 4에서는 발행이 발행된, 동일한 큐 관리자에 2개의 새 구독을 추가하는 결과를 보여줍니다. 결과는 예상한 대로입니다. 새 구독은 각각 하나의 발행을 수신하고 다른 구독에서 수신하는 발행 수는 변경되지 않습니다. 다른 클러스터 큐 관리자에서 예상치 못한 결과가 발생합니다(72 페이지의 표 5 참조).

표 *7. QMA*에서 수신된 발행

구독	토픽 문자열	수신된 발행	참고
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	Sports/Football에서 WILDCARD(BLOCK)으로 차단된 Football 하위 트리에 대한 모든 발행
SARSENAL	Sports/#/Arsenal	-	Sports/Football에서 WILDCARD(BLOCK)은 Arsenal에서의 와일드카드 구독을 금지함
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby의 기본 WILDCARD는 Leeds에서 와일드카드 구독을 금지하지 않습니다.
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal은 구독이 와일드카드를 포함하지 않으므로 발행을 수신합니다.
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds는 모든 이벤트에서 발행을 수신합니다.

72 페이지의 표 5에서는 QMB에서 2개의 새 구독을 추가하고 QMA에서 발행하는 결과를 보여줍니다. QMB가 이러한 2개의 새 구독 없이 발행을 수신하지 않는다는 점을 상기하십시오. 예상대로 Sports/FootBall 및 Sports/Rugby 가 둘 다 클러스터 토픽이므로 두 개의 새 구독이 발행물을 수신합니다. QMB 는 Sports/Football/Arsenal 및 Sports/Rugby/Leeds 에 대한 프록시 구독을 QMA에 전달하며, 이는 QMB에 발행물을 송신합니다.

예상치 못한 결과는, 이전에 발행을 수신하지 않던 2개의 구독 Sports/# 및 Sports/#/Leeds가 이제 발행을 수신한다는 점입니다. 이유는 바로, 다른 구독에 대해 QMB로 전달되는 Sports/Football/Arsenal 및 Sports/Rugby/Leeds 발행이 이제 QMB에 연결된 모든 구독자에게 사용 가능합니다. 결과적으로 로컬 토픽 Sports/# 및 Sports/#/Leeds에 대한 구독은 Sports/Rugby/Leeds 발행을 수신합니다. Sports/#/Arsenal은 계속해서 발행을 수신하지 않습니다. Sports/Football에서 해당 WILDCARD 특성 값이 BLOCK으로 설정되었기 때문입니다.

표 *8. QMB*에서 수신된 발행

구독	토픽 문자열	수신된 발행	참고
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*Sports/Football*에서 *WILDCARD(BLOCK)*에 의해 차단된 *Football* 하위 트리에 대한 모든 발행
SARSENAL	Sports/#/Arsenal	-	Sports/Football에서 WILDCARD(BLOCK)은 Arsenal에서의 와일드카드 구독을 금지함
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	*Sports/Rugby*의 기본 *WILDCARD*는 *Leeds*에서 와일드카드 구독을 금지하지 않습니다.
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal은 구독이 와일드카드를 포함하지 않으므로 발행을 수신합니다.
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds는 모든 이벤트에서 발행을 수신합니다.

대부분의 애플리케이션에서는 한 구독이 다른 구독의 작동에 영향을 주는 것은 바람직하지 않습니다. 값이 BLOCK인 WILDCARD 특성의 한 가지 중요한 사용법은 균일하게 작동하는 와일드카드를 포함하는 동일한 토픽 문자열에 대한 구독을 작성하는 것입니다. 발행자와 동일한 큐 관리자 또는 다른 큐 관리자의 구독인지에 상관없이 구독 결과는 동일합니다.

와일드카드 및 스트림

발행/구독 API에 기록된 새 애플리케이션의 경우 * 에 대한 구독이 발행물을 수신하지 않습니다. 모든 Sport 발행을 수신하려면 Sports/* 또는 Sports/#을 구독해야 하며,하고 Business 발행도 마찬가지입니다.

큐된 기존 발행/구독 애플리케이션의 동작은 발행/구독 브로커가 IBM MQ의 이후 버전으로 마이그레이션될 때 변경되지 않습니다. **Publish**, **Register Publisher**또는 **Subscriber** 명령의 **StreamName** 특성은 스트림이 마이그레이션된 토픽의 이름에 맵핑됩니다.

와일드카드 및 구독 지점

발행/구독 API에 기록된 새 애플리케이션의 경우 마이그레이션의 효과는 * 에 대한 구독이 발행물을 수신하지 않는다는 점입니다. 모든 Sport 발행을 수신하려면 Sports/* 또는 Sports/#을 구독해야 하며,하고 Business 발행도 마찬가지입니다.

큐된 기존 발행/구독 애플리케이션의 동작은 발행/구독 브로커가 IBM MQ의 이후 버전으로 마이그레이션될 때 변경되지 않습니다. **Publish**, **Register Publisher** 또는 **Subscriber** 명령에서 **SubPoint** 특성은 구독이 마이그레이션되는 토픽 이름에 맵핑됩니다.

예제: Sport 발행/구독 클러스터 작성

다음 단계는 네 개의 큐 관리자 (두 개의 전체 저장소, CL1A 및 CL1B, 두 개의 부분 저장소, QMA 및 QMB) 가 있는 CL1클러스터를 작성합니다. 전체 저장소는 클러스터 정의만 보유하는 데 사용됩니다. QMA는 클러스터 토픽 호스트를 지정합니다. 지속 가능 구독은 QMA 및 QMB 모두에서 정의됩니다.

참고: 예제는 Windows용으로 코딩되었습니다. 다른 플랫폼에서 예제를 구성하고 테스트하도록 qmgrs.bat 작성 및 pub.bat 작성을 다시 코딩해야 합니다.

1. 스크립트 파일을 작성하십시오.
   a. topics.tst 작성
   b. wildsubs.tst 작성
   c. fullsubs.tst 작성
   d. qmgrs.bat 작성
   e. pub.bat 작성
2. qmgrs.bat 작성을 실행하여 구성을 작성하십시오.

```
qmgrs
```

70 페이지의 그림 23에서 토픽을 작성하십시오. 그림 5에서 스크립트는 클러스터 토픽 Sports/Football 및 Sports/Rugby를 작성합니다.

참고: REPLACE 옵션은 주제의 TOPICSTR 특성을 바꾸지 않습니다. TOPICSTR은(는) 다른 토픽 트리를 테스트하는 예제에서 사용되는 특성입니다. 토픽을 변경하려면 먼저 토픽을 삭제하십시오.

```
DELETE TOPIC ('Sports')
DELETE TOPIC ('Football')
DELETE TOPIC ('Arsenal')
DELETE TOPIC ('Blackburn')
DELETE TOPIC ('Chelsea')
DELETE TOPIC ('Rugby')
DELETE TOPIC ('Leeds')
DELETE TOPIC ('Wigan')
DELETE TOPIC ('Warrington')
DELETE TOPIC ('St. Helens')

DEFINE TOPIC ('Sports')   TOPICSTR('Sports')
DEFINE TOPIC ('Football')  TOPICSTR('Sports/Football') CLUSTER(CL1) WILDCARD(BLOCK)
DEFINE TOPIC ('Arsenal')  TOPICSTR('Sports/Football/Arsenal')
DEFINE TOPIC ('Blackburn') TOPICSTR('Sports/Football/Blackburn')
DEFINE TOPIC ('Chelsea')  TOPICSTR('Sports/Football/Chelsea')
DEFINE TOPIC ('Rugby')   TOPICSTR('Sports/Rugby')  CLUSTER(CL1)
DEFINE TOPIC ('Leeds')   TOPICSTR('Sports/Rugby/Leeds')
DEFINE TOPIC ('Wigan')   TOPICSTR('Sports/Rugby/Wigan')
DEFINE TOPIC ('Warrington') TOPICSTR('Sports/Rugby/Warrington')
DEFINE TOPIC ('St. Helens') TOPICSTR('Sports/Rugby/St. Helens')
```

그림 *38.* 토픽 삭제 및 작성*: topics.tst*

참고: REPLACE가 토픽 문자열을 바꾸지 않으므로 토픽을 삭제합니다.

와일드카드를 사용하여 구독을 작성하십시오. 70 페이지의 그림 23에서 토픽 오브젝트를 포함하는 토픽에 대응하는 와일드카드. 각 구독에 대한 큐를 작성하십시오. 스크립트를 실행하거나 다시 실행하면 큐가 지워지고 구독이 삭제됩니다.

참고: REPLACE 옵션은 구독의 TOPICOBJ 또는 TOPICSTR 특성을 바꾸지 않습니다. TOPICOBJ 또는 TOPICSTR은 다른 구독을 테스트하기 위해 예제에서 다양하게 사용할 수 있는 특성입니다. 변경하려면 먼저 구독을 삭제하십시오.

```
DEFINE QLOCAL(QSPORTS)  REPLACE
DEFINE QLOCAL(QSARSENAL) REPLACE
DEFINE QLOCAL(QSLEEDS)  REPLACE
CLEAR QLOCAL(QSPORTS)
CLEAR QLOCAL(QSARSENAL)
CLEAR QLOCAL(QSLEEDS)

DELETE SUB  (SPORTS)
DELETE SUB  (SARSENAL)
DELETE SUB  (SLEEDS)
DEFINE SUB  (SPORTS)  TOPICSTR('Sports/#')     DEST(QSPORTS)
DEFINE SUB  (SARSENAL) TOPICSTR('Sports/+/Arsenal') DEST(QSARSENAL)
DEFINE SUB  (SLEEDS)  TOPICSTR('Sports/+/Leeds')  DEST(QSLEEDS)
```

그림 *39.* 와일드카드 구독 작성*: wildsubs.tst*

클러스터 토픽 오브젝트를 참조하는 구독을 작성하십시오.

참고:

구분 기호, /는 TOPICOBJ에서 참조하는 토픽 문자열과 TOPICSTR에서 정의하는 토픽 문자열 사이에 자동으로 삽입됩니다.

DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) 정의는 동일한 구독을 작성합니다. TOPICOBJ는 이미 정의한 토픽 문자열을 참조하는 빠른 방법으로 사용됩니다. 작성되면 구독은 더 이상 토픽 오브젝트를 참조하지 않습니다.

```
DEFINE QLOCAL(QFARSENAL) REPLACE
DEFINE QLOCAL(QRLEEDS)  REPLACE
CLEAR QLOCAL(QFARSENAL)
CLEAR QLOCAL(QRLEEDS)

DELETE SUB  (FARSENAL)
DELETE SUB  (RLEEDS)
DEFINE SUB  (FARSENAL) TOPICOBJ('Football') TOPICSTR('Arsenal') DEST(QFARSENAL)
DEFINE SUB  (RLEEDS)  TOPICOBJ('Rugby')  TOPICSTR('Leeds')  DEST(QRLEEDS)
```

그림 *40.* 구독 삭제 및 삭제*: fullsubs.tst*

2개의 저장소를 포함하는 클러스터를 작성하십시오. 발행 및 구독을 위해 2개의 부분 저장소를 작성하십시오. 모두 삭제하고 다시 시작하도록 스크립트를 다시 실행하십시오. 또한 스크립트는 토픽 계층 및 초기 와일드카드 구독을 작성합니다.

참고:

다른 플랫폼에 비슷한 스크립트를 작성하거나 모두 명령을 입력하십시오. 스크립트를 사용하면 모두 삭제하고 동일한 구성으로 다시 빠르게 시작할 수 있습니다.

```
@echo off
set port.CL1B=1421
set port.CLIA=1420
for %%A in (CL1A CL1B QMA QMB) do call :createQM %%A
call :configureQM CL1A CL1B %port.CL1B% full
call :configureQM CL1B CL1A %port.CL1A% full
for %%A in (QMA QMB) do call :configureQM %%A CL1A %port.CL1A% partial
for %%A in (topics.tst wildsubs.tst) do runmqsc QMA < %%A
for %%A in (wildsubs.tst) do runmqsc QMB < %%A
goto:eof

:createQM
echo Configure Queue manager %1
endmqm -p %1
for %%B in (dlt crt str) do %%Bmqm %1
goto:eof

:configureQM
if %1==CL1A set p=1420
if %1==CL1B set p=1421
if %1==QMA set p=1422
if %1==QMB set p=1423
echo configure %1 on port %p% connected to repository %2 on port %3 as %4 repository
echo DEFINE LISTENER(LST%1) TRPTYPE(TCP) PORT(%p%) CONTROL(QMGR) REPLACE | runmqsc %1
echo START LISTENER(LST%1) | runmqsc %1
if full==%4 echo ALTER QMGR REPOS(CL1) DEADQ(SYSTEM.DEAD.LETTER.QUEUE) | runmqsc %1
echo DEFINE CHANNEL(TO.%2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('LOCALHOST(%3)') CLUSTER(CL1)
REPLACE | runmqsc %1
echo DEFINE CHANNEL(TO.%1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('LOCALHOST(%p%)')
CLUSTER(CL1) REPLACE | runmqsc %1
goto:eof
```

그림 *41.* 큐 관리자 작성*: qmgrs.bat*

클러스터 토픽에 구독을 추가하여 구성을 업데이트하십시오.

```
@echo off
for %%A in (QMA QMB) do runmqsc %%A < wildsubs.tst
for %%A in (QMA QMB) do runmqsc %%A < upsubs.tst
```

그림 *42.* 구독 업데이트*: upsubs.bat*

큐 관리자에서 매개변수로 pub.bat를 실행하여 발행 토픽 문자열을 포함하는 메시지를 발행하십시오. Pub.bat 는 샘플 프로그램 **amqspub**를 사용합니다.

```
@echo off
@rem Provide queue manager name as a parameter
set S=Sports
set S=6 Sports/Football Sports/Football/Arsenal
set S=6 Sports/Rugby Sports/Rugby/Leeds
for %%B in (6) do echo %%B | amqspub %%B %1
```

그림 *43.* 발행*: pub.bat*

관련 개념

와일드카드 구독 및 보유된 발행

발행 범위

발행/구독 클러스터 또는 계층을 구성하는 경우, 발행 범위는 큐 관리자가 리모트 큐 관리자로 발행을 전달할지 여부를 추가로 제어합니다. **PUBSCOPE** 토픽 속성을 사용하여 발행 범위를 관리합니다.

발행이 리모트 큐 관리자에 전달되지 않으면 로컬 구독자만 발행을 수신합니다.

발행/구독 클러스터를 사용할 경우, 발행의 범위는 주로 토픽 트리의 특정 지점에 있는 클러스터 토픽 오브젝트의 정의에 의해 제어됩니다. 발행 범위는 클러스터의 다른 큐 관리자로 발행의 플로우를 허용하도록 설정되어야 합니다. 특정 큐 관리자에서 특정 토픽의 상세한 제어를 필요로 하는 경우 클러스터 토픽에 대한 발행 범위만 제한해야 합니다.

발행/구독 계층을 사용할 경우, 발행의 범위는 주로 구독 범위 속성과 결합된 이 속성에 의해 제어됩니다.

PUBSCOPE 속성은 특정 토픽에서 만들어진 발행 범위를 판별하는 데 사용됩니다. 속성을 다음 값 중 하나로 설정할 수 있습니다.

QMGR
: 발행은 로컬 구독자에게만 전달됩니다. 이러한 발행을 로컬 발행이라고 합니다. 로컬 발행은 리모트 큐 관리자로 전달되지 않으므로 리모트 큐 관리자에 연결된 구독자는 수신하지 않습니다.

모두
: 발행은 발행/구독 클러스터 또는 계층의 리모트 큐 관리자에 연결된 구독자 및 로컬 구독자에게 전달됩니다. 이러한 발행을 글로벌 발행이라고 합니다.

ASPARENT
: 토픽 트리에 있는 상위 토픽의 **PUBSCOPE** 설정을 사용합니다.

또한 발행자는 MQPMO_SCOPE_QMGR 입력 메시지 옵션을 사용하여 발행 종류(로컬 또는 글로벌)를 지정할 수 있습니다. 이 옵션을 사용하면 **PUBSCOPE** 토픽 속성을 사용하여 설정된 동작을 대체합니다.

관련 개념

68 페이지의 『관리 토픽 오브젝트』
관리 토픽 오브젝트를 사용하여 기본이 아닌 특정 속성을 토픽에 지정할 수 있습니다.

관련 태스크

분산 발행/구독 네트워크 구성

구독 범위

구독 범위는 한 큐 관리자의 구독이 발행/구독 클러스터 또는 계층에 있는 다른 큐 관리자에서 발행된 발행을 수신하는지, 아니면 로컬 발행자의 발행을 수신하는지 여부를 제어합니다.

큐 관리자로 구독 범위를 제한하면 프록시 구독이 발행/구독 토폴로지에 있는 다른 큐 관리자로 전달되는 작업이 중지됩니다. 이로 인해 큐 관리자 내 발행/구독 메시징 트래픽을 줄입니다.

발행/구독 클러스터를 사용할 경우, 구독의 범위는 주로 토픽 트리의 특정 지점에 있는 클러스터 토픽 오브젝트의 정의에 의해 제어됩니다. 구독 범위는 클러스터의 다른 큐 관리자로 구독의 플로우를 허용하도록 설정되어야 합니다. 특정 큐 관리자에서 특정 토픽의 상세한 제어를 필요로 하는 경우 클러스터 토픽에 대한 구독 범위만 제한해야 합니다.

발행/구독 계층을 사용할 경우, 구독의 범위는 주로 발행 범위 속성과 결합된 이 속성에 의해 제어됩니다.

SUBSCOPE 토픽 속성은 특정 토픽에 대한 구독의 범위를 판별하는 데 사용됩니다. 속성을 다음 값 중 하나로 설정할 수 있습니다.

QMGR
: 구독은 로컬 발행만 수신하며 프록시 구독은 리모트 큐 관리자로 전파되지 않습니다.

모두
: 프록시 구독은 발행/구독 클러스터 또는 계층의 리모트 큐 관리자로 전파되며 구독자는 로컬 및 원격 발행을 수신합니다.

ASPARENT
: 토픽 트리에 있는 상위 토픽의 **SUBSCOPE** 설정을 사용합니다.

토픽에 대한 구독 범위가 직접 또는 ASPARENT를 통해 분석된 ALL로 설정된 경우, 구독을 작성할 때 MQSO_SCOPE_QMGR을 지정하여 해당 토픽에 대한 개별 구독이 해당 범위를 QMGR로 제한할 수 있습니다. QMGR의 범위가 있는 토픽에 대한 구독은 ALL로 범위를 확장할 수 없습니다.

관련 개념
68 페이지의 『관리 토픽 오브젝트』
관리 토픽 오브젝트를 사용하여 기본이 아닌 특정 속성을 토픽에 지정할 수 있습니다.

관련 태스크
분산 발행/구독 네트워크 구성

토픽 공간

토픽 공간은 사용자가 구독 및 발행할 수 있는 토픽 세트입니다. 분산 발행/구독 토폴로지의 큐 관리자에는 해당 토폴로지의 연결된 큐 관리자에서 구독 및 발행된 토픽을 잠재적으로 포함하는 토픽 공간이 있습니다.

참고: 관리 토픽 오브젝트, 토픽 문자열 및 토픽 트리와 같은 큐 관리자 내의 토픽에 대한 개요는 60 페이지의 『토픽』의 내용을 참조하십시오. 현재 문서에서 토픽에 대한 추가 참조는 달리 지정하지 않는 한 토픽 문자열을 의미합니다.

토픽은 다음 방법 중 하나로 초기에 작성됩니다.

- 토픽 오브젝트 또는 지속 가능 구독을 정의하는 경우 관리 하에
- 애플리케이션이 새 토픽에 대해 동적으로 발행 또는 구독을 작성할 경우 동적으로

토픽은 프록시 구독 및 관리 클러스터 토픽 오브젝트를 작성하는 방식 모두를 통해 다른 큐 관리자로 전파됩니다. 프록시 구독의 경우 발행자가 연결된 큐 관리자에서 구독의 큐 관리자로 발행이 전달됩니다.

프록시 구독은 큐 관리자 계층에서 상위-하위 관계로 함께 연결되는 모든 큐 관리자 사이에 전파됩니다. 결과적으로 한 큐 관리자에서 계층의 다른 큐 관리자에 정의된 토픽을 구독할 수 있습니다. 큐 관리자 사이에 연결된 경로가 있는 한, 큐 관리자의 연결 방식은 중요하지 않습니다.

프록시 구독은 또한 구독을 위해 발행/구독 클러스터의 클러스터 토픽으로도 전파됩니다. 클러스터 토픽은 **CLUSTER** 속성이 있거나 상위로부터 속성을 상속하는 토픽 오브젝트에 연결된 토픽입니다. 클러스터 토픽이 아닌 토픽은 로컬 토픽이라고 하며 클러스터에 복제되지 않습니다. 클러스터에 대해서는 구독에서 로컬 토픽으로 프록시 구독이 전파되지 않습니다.

즉, 프록시 구독은 두 가지 상황에서 구독자에 대해 작성됩니다.

1. 큐 관리자가 계층의 멤버이고 프록시 구독이 큐 관리자의 상위와 하위로 전달됩니다.
2. 큐 관리자가 클러스터의 멤버이고 구독 토픽 문자열이 클러스터 토픽 오브젝트와 연관된 토픽으로 분석됩니다. 토픽이 직접 라우팅된 클러스터 토픽인 경우, 프록시 구독이 클러스터의 모든 멤버에게 전달됩니다. 토픽이 토픽 호스트 라우트 클러스터 토픽인 경우, 프록시 구독은 클러스터 토픽 오브젝트를 정의한 클러스터의 큐 관리자로만 전달됩니다. 자세한 정보는 81 페이지의 『발행/구독 클러스터』의 내용을 참조하십시오.

큐 관리자가 클러스터 및 계층의 멤버이면 구독자에게 중복 발행을 전달하지 않고도 두 메커니즘에서 프록시 구독이 전파됩니다.

3개 발행/구독 토폴로지의 토픽 공간은 다음 목록에서 설명됩니다.

- 92 페이지의 『케이스 1. 발행/구독 클러스터』.
- 93 페이지의 『케이스 2. 발행/구독 계층』.

별도의 토픽에서, 다음 구성 태스크는 토픽 공간을 결합하는 방법에 대해 설명합니다.

- 발행/구독 클러스터에서 단일 토픽 공간 작성
- 다중 클러스터의 토픽 공간 결합
- 다중 클러스터에서 토픽 공간 결합 및 격리
- 다중 클러스터에서 토픽 공간 발행 및 구독

케이스 1. 발행/구독 클러스터

예에서, 큐 관리자가 발행/구독 계층에 연결되지 않았다고 가정합니다.

큐 관리자가 발행/구독 클러스터의 멤버이면 해당 토픽 공간은 로컬 토픽 및 클러스터 토픽에서 구성됩니다. 로컬 토픽은 **CLUSTER** 속성 없이 토픽 오브젝트에 연관됩니다. 큐 관리자에 로컬 토픽 오브젝트 정의가 있으면 해당 토픽 공간은 로컬로 정의된 고유한 토픽 오브젝트도 포함하는 클러스터의 다른 큐 관리자와는 다릅니다.

구독한 토픽을 클러스터 토픽 오브젝트로 해석하지 않는 한, 발행/구독 클러스터에서 다른 큐 관리자에 정의된 토픽을 구독할 수 없습니다.

클러스터 토픽 오브젝트의 같은 이름을 가진 정의가 다중 큐 관리자에 필요한 경우, 예를 들어 토픽 호스트 라우팅을 사용 중인 경우, 필요할 때 모든 정의가 일치하는 것은 중요합니다. 자세한 정보는 발행/구독 클러스터에서 단일 토픽 공간 작성을 참조하십시오.

토픽 오브젝트의 로컬 정의는 클러스터 토픽에 대한 정의인지, 로컬 토픽에 대한 정의인지에 상관없이 클러스터의 다른 위치에서 정의된 동일한 토픽 오브젝트보다 우선합니다. 다른 곳에서 정의된 오브젝트가 보다 최근 항목이어도 로컬로 정의된 토픽이 사용됩니다.

클러스터 토픽 오브젝트는 클러스터의 모든 위치에서 동일한 토픽 문자열에 연관됩니다. 토픽 오브젝트가 연관된 토픽 문자열은 수정할 수 없습니다. 다른 토픽 문자열과 동일한 토픽 오브젝트를 연관하려면 토픽 오브젝트를 삭제하고 새 토픽 문자열로 다시 작성해야 합니다. 토픽이 클러스터된 경우 효과는 클러스터의 다른 멤버에 저장된 토픽 오브젝트의 사본을 삭제하고 클러스터의 모든 위치에서 새 토픽 오브젝트의 사본을 작성하는 것입니다. 토픽 오브젝트의 사본은 모두 동일한 토픽 문자열을 참조합니다.

클러스터의 서로 다른 큐 관리자에서 서로 다른 토픽 문자열을 사용하여 동일한 이름의 토픽 오브젝트 정의를 우발적으로 두 개 작성할 수 있습니다. 그러면 동작이 혼동될 수 있습니다. 토픽 문자열이 서로 다른, 동일한 토픽 오브젝트의 여러 정의로 인해 토픽을 참조하는 방법과 위치에 따라 다른 결과가 나올 수 있습니다. 이 중요한 시점에 대한 자세한 정보는 동일한 이름의 다중 클러스터 토픽 정의를 참조하십시오.

케이스 2. 발행/구독 계층

예에서 큐 관리자가 발행/구독 클러스터의 구성원이 아니라고 가정합니다.

IBM MQ에서 큐 관리자가 발행/구독 계층의 멤버인 경우 해당 토픽 공간은 로컬 및 연결된 큐 관리자에 정의된 모든 토픽으로 구성됩니다. 계층에서 모든 큐 관리자의 토픽 공간은 동일합니다. 로컬 토픽과 글로벌 토픽으로 토픽을 구분하지 않습니다.

PUBSCOPE 및 **SUBSCOPE** 옵션을 QMGR로 설정하여 토픽의 발행을 발행자로부터 계층의 다른 큐 관리자에 연결된 구독자로 전달되지 않도록 하십시오.

큐 관리자 QMA에서 토픽 문자열 USA/Alabama를 포함하는 토픽 오브젝트 Alabama를 정의한다고 가정하십시오. 결과는 다음과 같습니다.

1. 이제 QMA의 토픽 공간에 토픽 오브젝트 Alabama 및 토픽 문자열 USA/Alabama가 포함됩니다.
2. 애플리케이션 또는 관리자가 토픽 오브젝트 이름 Alabama를 사용하여 QMA에 구독을 작성할 수 있습니다.
3. 애플리케이션은 계층의 큐 관리자에서 USA/Alabama를 포함하여 토픽에 대한 구독을 작성할 수 있습니다. QMA가 로컬로 정의되지 않은 경우 토픽 USA/Alabama가 토픽 오브젝트 SYSTEM.BASE.TOPIC으로 해석됩니다.

관련 개념

90 페이지의 『발행 범위』
발행/구독 클러스터 또는 계층을 구성하는 경우, 발행 범위는 큐 관리자가 리모트 큐 관리자로 발행을 전달할지 여부를 추가로 제어합니다. **PUBSCOPE** 토픽 속성을 사용하여 발행 범위를 관리합니다.

91 페이지의 『구독 범위』
구독 범위는 한 큐 관리자의 구독이 발행/구독 클러스터 또는 계층에 있는 다른 큐 관리자에서 발행된 발행을 수신하는지, 아니면 로컬 발행자의 발행을 수신하는지 여부를 제어합니다.

관련 태스크

분산 발행/구독 네트워크 구성

IBM MQ 멀티캐스트

IBM MQ 멀티캐스트는 낮은 지연, 높은 팬아웃, 신뢰할 수 있는 멀티캐스트 메시징을 제공합니다.

멀티캐스트는 성능을 저하시키지 않고 다수의 구독자에 맞게 스케일링할 수 있기 때문에 발행/구독 메시징의 효율적인 형식입니다. IBM MQ는 수신확인, 부정적 수신확인 및 순서 번호를 사용하여 높은 팬아웃을 갖는 낮은 지연 메시징을 달성함으로써 신뢰할 수 있는 멀티캐스트 메시징을 가능하게 합니다.

IBM MQ 멀티캐스트의 공정한 전달은 거의 동시 전달을 가능하게 하므로, 어떤 수신자도 우위를 차지하지 않도록 보장합니다. IBM MQ 멀티캐스트가 네트워크를 사용하여 메시지를 전달하기 때문에 발행/구독 엔진은 데이터를 팬아웃하기 위해 필요하지 않습니다. 토픽이 그룹 주소에 맵핑된 후, 발행자와 구독자가 피어 투 피어 모드에서 동작할 수 있기 때문에 큐 관리자가 필요없습니다. 여기에서는 큐 관리자 서버에 대한 로드가 감소하므로, 큐 관리자 서버가 더 이상 잠재적인 실패 지점이 되지 않습니다.

초기 멀티캐스트 개념

IBM MQ 멀티캐스트는 통신 정보(COMMINFO) 오브젝트를 사용하여 기존 시스템과 애플리케이션에 쉽게 통합할 수 있습니다. 두 개의 TOPIC 오브젝트 필드를 사용하면 기존 TOPIC 오브젝트의 빠른 구성에서 멀티캐스트 트래픽을 지원하거나 무시할 수 있습니다.

멀티캐스트에 필요한 오브젝트

다음 정보는 IBM MQ 멀티캐스트에 필요한 두 개의 오브젝트에 대한 간단한 개요입니다.

COMMINFO 오브젝트
COMMINFO 오브젝트에는 멀티캐스트 전송과 연관된 속성이 포함됩니다. COMMINFO 오브젝트 매개변수에 대한 자세한 정보는 DEFINE COMMINFO를 참조하십시오.

설정해야 하는 COMMINFO 필드만 COMMINFO 오브젝트의 이름입니다. 이 이름은 토픽에 대해 COMMINFO 오브젝트를 식별하는 데 사용됩니다. 값이 올바른 멀티캐스트 그룹 주소인지 확인하려면 COMMINFO 오브젝트의 **GRPADDR** 필드를 선택해야 합니다.

토픽 오브젝트
토픽은 발행/구독 메시지로 발행된 정보의 주제이며, 이러한 토픽은 TOPIC 오브젝트를 작성하여 정의합니다. TOPIC 오브젝트 매개변수에 대한 자세한 정보는 DEFINE TOPIC을 참조하십시오.

TOPIC 오브젝트 매개변수의 값(**COMMINFO** 및 **MCAST**)을 변경하여 멀티캐스트에서 기존 토픽을 사용할 수 있습니다.

- **COMMINFO** 이 매개변수는 멀티캐스트 통신 정보 오브젝트의 이름을 지정합니다.
- **MCAST** 이 매개변수는 토픽 트리의 이 지점에서 멀티캐스트를 허용할 수 있는지의 여부를 지정합니다. 기본적으로 **MCAST**는 토픽의 멀티캐스트 속성이 상위에서 상속됨을 의미하는 ASPARENT로 설정됩니다. **MCAST**를 ENABLED로 설정하면 이 노드에서 멀티캐스트 트래픽이 가능합니다.

멀티캐스트 네트워크 및 토픽

다음 정보는 여러 유형의 구독 및 토픽 정의가 있는 구독에 발생하는 상황에 대한 개요입니다. 이러한 예에서는 모두 TOPIC 오브젝트 **COMMINFO** 매개변수가 올바른 COMMINFO 오브젝트의 이름으로 설정되어 있다고 가정합니다.

사용 가능한 멀티캐스트로 설정된 토픽
토픽 문자열 **MCAST** 매개변수를 ENABLED로 설정하면, 다음과 같은 경우를 제외하고는 멀티캐스트 가능 클라이언트의 구독이 허용되며 멀티캐스트 구독이 이루어집니다.

- 멀티캐스트 가능 클라이언트의 지속 가능 구독인 경우
- 멀티캐스트 가능 클라이언트의 비관리 구독인 경우
- 멀티캐스트 불가능 클라이언트의 구독인 경우

이러한 경우 멀티캐스트가 아닌 구독이 이루어지며 구독이 일반 발행/구독으로 다운그레이드됩니다.

사용 불가능한 멀티캐스트로 설정된 토픽
토픽 문자열 **MCAST** 매개변수가 DISABLED로 설정되면, 항상 멀티캐스트가 아닌 구독이 이루어지고 구독이 일반 발행/구독으로 다운그레이드됩니다.

멀티캐스트로만 설정된 토픽
토픽 문자열 **MCAST** 매개변수를 ONLY로 설정하면, 다음과 같은 경우를 제외하고는 멀티캐스트 가능 클라이언트의 구독이 허용되며 멀티캐스트 구독이 이루어집니다.

- 지속 가능 구독임: 지속 가능 구독은 이유 코드 2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED로 거부됨

- 비관리 구독임: 비관리 구독은 이유 코드 2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR로 거부됨
- 멀티캐스트가 불가능한 클라이언트의 구독임: 이러한 구독은 이유 코드 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY로 거부됨
- 로컬로 바인드된 애플리케이션의 구독임: 이러한 구독은 이유 코드 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY로 거부됨

Windows Linux AIX MQ Telemetry 개요

MQ Telemetry는 큐 관리자의 일부인 텔레메트리(MQXR) 서비스, 직접 작성하거나 무료로 다운로드할 수 있는 텔레메트리 클라이언트와 명령행 및 탐색기 관리 인터페이스로 구성됩니다. 텔레메트리는 다양한 원격 장치에서 데이터를 수집하고 이런 장치를 관리하는 것을 말합니다. MQ Telemetry를 사용하면 데이터 수집과 장치 제어를 웹 애플리케이션으로 통합할 수 있습니다.

MQ Telemetry은 IBM MQ의 구성요소입니다. 이러한 버전으로 업그레이드하려면 IBM MQ의 이후 버전을 설치해야 합니다.

샘플 애플리케이션은 Eclipse Paho및 MQTT.org에서 무료로 계속 사용할 수 있습니다. IBM MQ Telemetry Transport 샘플 프로그램을 참조하십시오.

MQ Telemetry 는 IBM MQ의 구성요소이므로 기본 제품과 함께 또는 기본 제품이 설치된 후에 MQ Telemetry 를 설치할 수 있습니다. 마이그레이션 정보는 Windows 에서 MQ Telemetry 마이그레이션 및 Linux 에서 MQ Telemetry 마이그레이션의 내용을 참조하십시오.

MQ Telemetry에는 다음 컴포넌트가 포함되어 있습니다.

텔레메트리 채널
MQTT 클라이언트와 IBM MQ 사이의 연결을 관리하려면 텔레메트리 채널을 사용하십시오. 텔레메트리 채널은 새 IBM MQ 오브젝트 (예: SYSTEM.MQTT.TRANSMIT.QUEUE) 를 사용하여 IBM MQ와 상호작용합니다.

텔레메트리(MQXR) 서비스
MQTT 클라이언트는 SYSTEM.MQXR.SERVICE 텔레메트리 서비스를 사용하여 텔레메트리 채널에 연결합니다.

IBM MQ Explorer 지원 MQ Telemetry
MQ Telemetry는 IBM MQ Explorer를 사용하여 관리할 수 있습니다.

문서
MQ Telemetry 문서는 표준 IBM MQ 제품 문서에 포함되어 있습니다. Java와 C 클라이언트를 위한 SDK 문서는 Javadoc 및 HTML 형식으로 제품 문서에서 제공됩니다.

Telemetry 개념

우리는 주변 환경에서 정보를 수집하여 무엇을 할 지 결정합니다. 소비자로서 음식을 구입하기 전에 가게에 어떤 물건이 있는지 확인합니다. 환승 예약을 하기 전에 지금 떠나면 얼마나 오래 걸릴 지 알고 싶어합니다. 의사를 방문할까 결정하기 전에 자신의 증상을 확인합니다. 좀 더 기다릴까 정하기 전에 버스가 언제 도착할지 확인합니다. 이러한 결정을 내리게 되는 근거가 되는 정보는 측정기와 디바이스, 문서와 화면, 그리고 우리에게서 나옵니다. 어디에 있든, 필요할 때면 언제나 우리는 정보를 수집하고 종합하여 이를 토대로 행동합니다.

정보의 소스가 넓게 분산되어 있거나 액세스할 수 없으면 가장 정확한 정보를 모으기 어려우며 비용이 많이 들게 됩니다. 큰 변화를 가져오려 하거나 변화를 만들기 어려울 경우 변화시키지 못한 것이 생기거나 덜 효과적일 때 변화가 일어나게 됩니다.

넓게 분산되어 있는 디바이스를 디지털 기술로 인터넷에 연결시켜 이를 제어하고 여기서 정보를 수집하는 비용이 획기적으로 줄어들 수 있다면 어떠십니까? 이 정보는 인터넷과 기업의 자원을 사용하여 분석할 수 있습니다. 정보에 기반하여 결정을 내리고 행동을 취할 수 있는 기회가 더 많아집니다.

기술 트렌드, 환경 및 경제적 요구로 인해 다음과 같은 변화가 일어나고 있습니다.

1. 표준화와 저비용 디지털 프로세서와의 연결을 통해 센서와 작동기를 연결하고 제어하는 비용이 줄어들고 있습니다.
2. 디바이스를 연결하는 데 인터넷과 인터넷 기술이 점점 더 많이 사용되고 있습니다. 몇몇 나라에서는 인터넷 애플리케이션에 대한 연결 수에서 휴대전화가 개인용 컴퓨터를 앞서고 있습니다. 다른 디바이스들도 곧 이 뒤를 따르게 될 것입니다.

3. 인터넷과 인터넷 기술이 애플리케이션이 훨씬 쉽게 데이터를 얻게 해 줍니다. 쉽게 데이터에 액세스할 수 있게 되면서 센서에서 얻을 데이터를 많은 솔루션에 유용한 정보로 바꿔 주는 데이터 분석의 사용이 크게 늘고 있습니다.
4. 자원을 지능적으로 사용하는 것이 탄소 배출량과 비용을 절감하는 더 빠르고 저렴한 방법입니다. 새 자원을 찾거나 기존 자원을 사용하는 새 기술을 개발하는 것과 같은 대안 솔루션은 오랜 기간이 걸릴 수 있습니다. 단기간에 새 기술을 개발하거나 새 자원을 찾는 것은 기존 솔루션을 개선하는 것보다 종종 더 위험하고, 느리며 비용이 많이 듭니다.

예

다음 예는 어떻게 이러한 트렌드가 환경과 지능적으로 상호작용할 수 있는 새 기회를 제공하는지 보여줍니다.

국제해상안전협약(International Convention for the Safety of Life at Sea)은 많은 배에 자동 식별 시스템(AIS, automatic identification system)을 배치하도록 요구합니다. 300톤이 넘는 상선과 여객선에는 필수 사항입니다. AIS는 기본적으로 연안 해운의 충돌 회피 시스템입니다. 해군 당국에서 연안을 모니터하고 제어하는 데에도 사용됩니다.

전 세계의 열정적인 사람들이 저비용 AIS 추적 기지를 배치하고 연안 해운 정보를 인터넷에 올립니다. AIS의 정보를 인터넷의 다른 정보와 결합시키는 애플리케이션을 작성하는 사람도 있습니다. 결과는 웹 사이트에 게제되며 Twitter와 SMS를 통해 발행됩니다.

한 애플리케이션에서 사우스햄턴 가까이 있는 AIS 기지의 정보가 배 소유주 및 지리적 정보와 결합됩니다. 이 애플리케이션은 선박 도착 및 출발에 대한 실시간 정보를 Twitter에 제공합니다. 사우스햄턴과 아일오브와이트 사이에서 연락선을 사용하는 정기 통근자들은 Twitter 또는 SMS를 사용하여 새 피드를 구독합니다. 피드에 연락선이 느리게 운행 중이라고 표시되면 통근자들은 출발을 늦추고 연락선이 예정된 도착 시간보다 늦게 정박하면 연락선을 탈 수 있습니다.

다른 예는 97 페이지의 『텔레메트리 유스 케이스』의 내용을 참조하십시오.

관련 태스크

설치 MQ Telemetry

MQ Telemetry 관리

Windows 에서 MQ Telemetry 마이그레이션

Linux 에서 MQ Telemetry 마이그레이션

MQ Telemetry용 애플리케이션 개발

MQ Telemetry 문제점 해결

관련 참조

MQ Telemetry 참조

Windows Linux AIX

MQ Telemetry 소개

사람과 비즈니스, 정부는 우리가 살며 일하는 환경과 좀 더 스마트한 상호작용을 하기 위해 MQ Telemetry를 점점 더 많이 활용하고자 합니다. MQ Telemetry는 모든 종류의 디바이스를 인터넷 및 기업과 연결하며 스마트 디바이스를 위한 애플리케이션을 빌드하는 비용을 감소시킵니다.

MQ Telemetry의 개념

- IBM MQ에 제공된 범용 메시징 백본을 광범위한 원격 센서, 작동기 및 텔레메트리 디바이스로 확장하는 IBM MQ 기능입니다. MQ Telemetry는 지능형 엔터프라이즈 애플리케이션, 서비스 및 의사결정자를 도구화된 디바이스 네트워크와 상호 연결할 수 있도록 IBM MQ를 확장합니다.
- MQ Telemetry의 핵심 파트는 다음과 같습니다.

  MQ Telemetry(MQXR) 서비스
  이 서비스는 IBM MQ 서버 내에서 실행되며 IBM MQ Telemetry Transport (MQTT) 프로토콜을 사용하여 텔레메트리 디바이스와 통신합니다.

  사용자가 기록하는 MQTT 애플리케이션
  이러한 애플리케이션은 텔레메트리 디바이스와 IBM MQ 큐 관리자 간에 다뤄지는 정보와 해당 정보에 대한 응답으로 수행되는 조치를 제어합니다. 이러한 애플리케이션을 작성하는 데 도움이 되도록 MQTT 클라이언트 라이브러리를 사용합니다.

사용 목적

- MQTT는 다양한 디바이스에 사용할 MQTT 구현을 작성할 수 있게 해주는 개방형 메시징 전송입니다.
- MQTT 클라이언트는 자원이 제한된 작은 풋프린트 디바이스에서 실행할 수 있습니다.
- MQTT는 대역폭이 낮거나, 송신 데이터 비용이 많이 소비되거나, 불안정한 네트워크에서 효과적으로 작동합니다.
- 메시지 전달은 보장되고, 애플리케이션과 구분됩니다.
- 애플리케이션 프로그래머에게 통신 프로그래밍 지식이 요구될 필요가 없습니다.
- 메시지는 다른 메시징 애플리케이션과 교환될 수 있습니다. 이는 다른 텔레메트리 애플리케이션, MQI, JMS 또는 엔터프라이즈 메시징 애플리케이션일 수 있습니다.

사용 방법

- 샘플 IBM MQ Telemetry Transport v3 클라이언트 애플리케이션(`mqttv3app.jar`)에서 작동하는 샘플 스크립트가 제공됩니다. IBM MQ Telemetry Transport 샘플 프로그램을 참조하십시오.
- IBM MQ Explorer 및 연관된 도구를 사용하여 IBM MQ의 텔레메트리 기능을 관리하십시오.
- 큐 관리자에 연결하고 발행/구독 메시징을 사용하는 MQTT 애플리케이션을 작성하는 데 도움이 되도록 클라이언트 라이브러리를 사용하십시오.
- 애플리케이션 및 클라이언트 라이브러리를 애플리케이션을 실행할 디바이스에 분배하십시오.

동작 방법

- MQTT는 발행/구독 프로토콜입니다. MQTT 클라이언트 애플리케이션은 MQTT 서버에 메시지를 발행하거나 MQTT 서버에 연결하는 애플리케이션이 송신하는 메시지를 구독할 수 있습니다.
- MQTT 클라이언트 애플리케이션은 MQTT 메시지 전송을 구현하는 클라이언트 라이브러리를 사용합니다.
- 기본 MQTT 클라이언트 애플리케이션은 표준 MQ 클라이언트처럼 작동하지만 더 광범위한 플랫폼 및 네트워크에서 실행할 수 있습니다.
- MQ Telemetry(MQXR) 서비스는 IBM MQ 큐 관리자를 MQTT 서버로 전환합니다.
- IBM MQ 큐 관리자가 MQTT 서버 역할을 수행하는 경우, 큐 관리자에 연결하는 다른 애플리케이션은 MQTT 클라이언트를 대상으로 메시지를 구독하고 수신할 수 있습니다.
- 큐 관리자는 발행 애플리케이션에서 구독 애플리케이션으로 메시지를 분배하는 라우터 역할을 합니다.
- 메시지 서로 다른 유형의 클라이언트 애플리케이션 간에 분배될 수 있습니다. 예를 들어 텔레메트리 클라이언트와 JMS 클라이언트 간에 분배될 수 있습니다.

참고: MQ Telemetry는 WebSphere Message Broker(지금은 IBM Integration Bus라고 함)의 버전 7에서 제거된 SCADA 노드를 대체하며 Windows, Linux 및 AIX에서 실행됩니다.

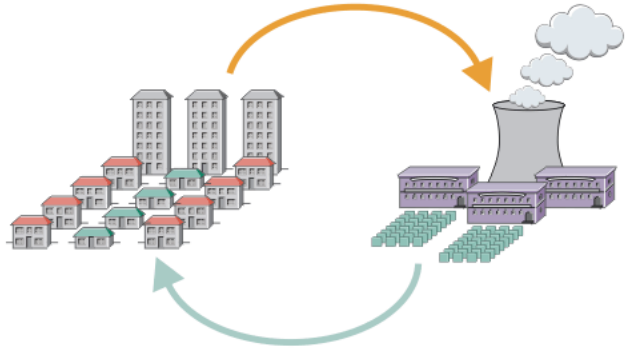
Windows Linux AIX 텔레메트리 유스 케이스

텔레메트리는 자동화된 감지, 데이터 측정 및 원격 디바이스의 제어를 가리킵니다. 주안점은 디바이스에서 중앙 제어 지점으로의 데이터 전송입니다. 또한 텔레메트리는 디바이스로 구성 및 제어 정보를 송신하는 것을 포함합니다.

MQ Telemetry는 MQTT protocol을 사용하는 소형 디바이스를 연결하고 해당 디바이스를 IBM MQ를 사용하는 다른 애플리케이션으로 연결합니다. MQ Telemetry는 디바이스와 인터넷 사이의 틈새를 중계하여 "스마트 솔루션"을 더 쉽게 구축할 수 있게 해 줍니다. 스마트 솔루션은 디바이스를 모니터하고 제어하는 애플리케이션을 위해 인터넷과 엔터프라이즈 애플리케이션으로부터 사용 가능한 풍부한 정보를 제공합니다.

다음 다이어그램은 MQ Telemetry의 일반적인 사용을 보여줍니다.

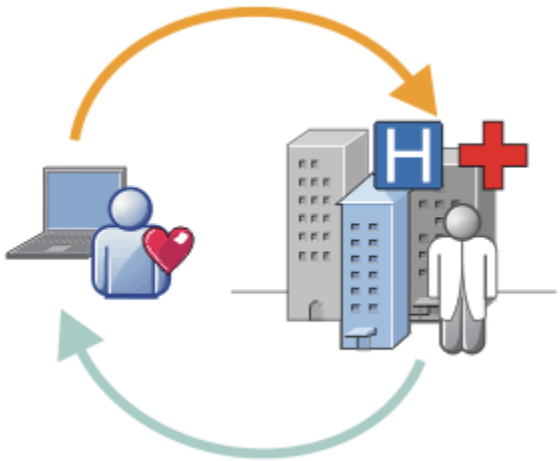
Telemetry: Smart Electricity



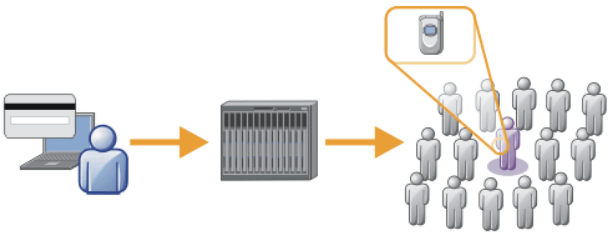
- 서비스 제공자에 전송된 에너지 사용 데이터를 포함하는 MQTT 메시지입니다.
- MQ Telemetry는 에너지 사용 데이터의 분석에 기반하여 CONTROL COMMANDS를 보냅니다.
- 자세한 정보는 다음 유스 케이스를 참조하십시오. 100 페이지의 『텔레메트리 유스 케이스: 가정 에너지 모니터링 및 제어』

Telemetry: Smart Health Services

- MQ Telemetry는 의료 데이터를 병원과 의사에게 송신합니다.
- MQTT 메시지 경보 또는 피드백은 의료 데이터의 분석을 기초로 송신될 수 있습니다.
- 자세한 정보는 다음 유스 케이스를 참조하십시오. 98 페이지의 『텔레메트리 유스 케이스: 가정 환자 모니터링』



Telemetry: 수천 명의 사람들 중 한 사람 식별



- 단순 카드 트랜잭션이 은행의 서버에 전송됩니다.
- MQ Telemetry는 수천 명의 사람들 중에서 한 사람을 식별하여 카드를 사용한 고객을 알려줍니다.
- MQ Telemetry는 가장 간단한 정보 입력을 사용하여 해당 개인을 찾습니다.

하위 주제에서 설명하는 유스 케이스는 실제 예제에서 유래합니다. 여기에서는 Telemetry를 사용하는 몇 가지 방법과 Telemetry 기술에서 해결해야 하는 몇 가지 공용된 문제점을 보여줍니다.

Windows Linux AIX 텔레메트리 유스 케이스: 가정 환자 모니터링

심장병 환자 간호 시스템에서의 IBM과 보건 의료 서비스 제공자와의 협업에서는 이식된 제세동기가 병원과 통신합니다. 환자와 이식된 디바이스에 대한 데이터는 RF 텔레메트리를 이용해 환자의 가정에 있는 MQTT 디바이스로 전송됩니다.

일반적으로 전송은 병상에 설치된 송신기에서 밤에 진행됩니다. 송신기는 전화 시스템을 이용해 데이터를 안전하게 병원으로 전송하며, 병원에서 이 데이터를 분석합니다.

이 시스템은 환자가 의사를 만나야 하는 횟수를 줄여 줍니다. 환자나 디바이스에 주의가 필요할 때를 감지하며 응급시에는 통화 가능한 의사에게 알립니다.

IBM과 보건 의료 서비스 제공자 사이의 협업에는 다수의 텔레메트리 유스 케이스에 공통되는 다음과 같은 몇 가지 특징이 있습니다.

투명성

디바이스는 전원, 전화 회선을 공급하는 것과 디바이스의 범위 안에 있을 것을 제외하고 사용자 개입을 필요로 하지 않습니다. 조작은 안정적이며 사용하기 간단합니다.

환자가 디바이스를 설정해야 할 필요를 없애기 위해 디바이스 공급자는 디바이스를 사전 구성합니다. 환자는 전원을 연결하기만 하면 됩니다. 환자가 직접 구성하지 않도록 함으로써 조작이 간편해지며 디바이스가 잘못 구성될 확률이 줄어듭니다.

MQTT 클라이언트가 디바이스의 한 부분으로 임베드되어 있습니다. 디바이스 개발자는 MQTT 클라이언트 구현을 디바이스에 임베드하며 개발자 또는 공급자가 사전 구성의 일부로서 MQTT 클라이언트를 구성합니다.

MQTT 클라이언트는 Java SE JAR 파일로 제공되며 개발자는 이를 자신의 Java 애플리케이션에 포함시킵니다. 이 디바이스와 같은 비Java 환경에서 디바이스 개발자는 발행된 MQTT 포맷 및 프로토콜을 이용하여 다른 언어로 클라이언트를 구현할 수 있습니다. 또는, 개발자는 Windows, Linux 및 ARM 플랫폼을 위한 공유 라이브러리로 제공된 C 클라이언트 중 하나를 사용할 수 있습니다.

불규칙한 연결

제세동기와 병원 사이의 통신에는 불규칙한 네트워크 특성이 있습니다. 환자에게서 데이터를 수집하고, 데이터를 병원으로 송신하는 서로 다른 문제를 해결하기 위해 두 개의 서로 다른 네트워크가 사용됩니다. 환자와 MQTT 디바이스 사이에는 단거리 저전력 RF 네트워크가 사용됩니다. 송신기는 VPN TCP/IP 연결을 사용하여 낮은 대역폭 전화 회선으로 병원과 연결합니다.

모든 디바이스를 인터넷 프로토콜 네트워크와 직접 연결하는 방법을 찾는 것은 보통 실용적이지 않습니다. 허브로 연결된 두 네트워크를 사용하는 것이 보통 솔루션입니다. MQTT 디바이스는 간단한 허브로, 환자에게서 정보를 저장하고 이를 병원으로 전달합니다.

보안

의사는 환자 데이터의 신뢰성을 믿을 수 있어야 하며 환자는 개인정보인 자신의 데이터가 보호되기를 바랍니다.

몇몇 상황에서는 VPN 또는 TLS를 사용하여 연결을 암호화하는 것으로 충분합니다. 데이터가 저장된 후에도 이를 보호하는 것이 더 좋은 상황도 있습니다.

텔레메트리 디바이스가 안전하지 않은 경우도 있습니다. 예를 들면, 공동 주택에 있거나 하는 경우입니다. 데이터가 올바른 환자의 것임을 보장할 수 있도록 디바이스 사용자는 반드시 인증을 받아야 합니다. 디바이스 자체도 TLS를 사용하여 서버의 인증을 받을 수 있고, 서버 또한 디바이스의 인증을 받을 수 있습니다.

디바이스와 큐 관리자 사이의 텔레메트리 채널은 사용자 인증을 위해 JAAS를 지원하며 통신 암호화와 디바이스 인증을 위해 TLS를 지원합니다. 발행에 대한 액세스는 IBM MQ의 오브젝트 권한 관리자에 의해 제어됩니다.

사용자 인증에 사용된 ID는 공용 환자 ID와 같은 다른 ID에 맵핑될 수 있습니다. 공용 ID는 IBM MQ의 발행 토픽에 대한 권한 구성을 단순화합니다.

연결성

MQTT 디바이스와 병원 사이의 연결은 전화 접속을 사용하며 1초에 300보오 정도의 낮은 대역폭으로 작동합니다.

300보오에서 효율적으로 작업하기 위해 MQTT protocol은 TCP/IP 헤더 외에 추가 바이트를 메시지에 조금 추가합니다.

MQTT protocol은 단일 전송 전송 후 삭제 메시징을 제공하며 이는 대기 시간을 낮게 유지합니다. 또한 보장된 전달이 응답 시간보다 더 중요한 경우에는 적어도 한 번 및 정확히 한 번*e* 전달을 보장하기 위해 다중 전송을 사용할 수 있습니다. 전달을 보장하기 위해 메시지는 전달이 완료될 때까지 디바이스에 저장됩니다. 디바이스가 무선으로 연결되어 있을 경우 보장된 전달은 특히 유용합니다.

확장성

텔레메트리 디바이스는 보통 수만에서 수백만에 이르는 규모로 많은 수가 배치됩니다.

시스템에 많은 디바이스를 연결할 경우 많은 솔루션이 필요하게 됩니다. 디바이스와 그 소프트웨어에 대한 비용과 같은 비즈니스적 요구, 라이센스, 디바이스와 사용자를 관리하는 관리 요구가 이에 해당합니다. 기술적 요구는 네트워크나 서버의 부하를 포함합니다.

연결을 여는 것은 열린 연결을 유지하는 것보다 많은 서버 자원을 사용합니다. 하지만 전화 회선을 사용하는 이와 같은 유스 케이스에서 연결 비용은 필요할 때를 제외하고 연결이 열려 있지 않다는 것을 의미합니다. 데이터 전송은 대부분 일괄 처리로 이뤄집니다. 연결은 취침 시간에 갑자기 증가하는 연결을 피하기 위해 심야에 스케줄될 수 있습니다.

클라이언트 측면에서 클라이언트의 확장성은 필요한 클라이언트 구성이 적을수록 향상됩니다. MQTT 클라이언트는 디바이스에 임베드되어 있습니다. 디바이스를 환자에 배치하는 데 있어서 구성이나 MQTT 클라이언트 라이센스 수락 단계와 같은 요구사항은 없습니다.

서버 측면에서 MQ Telemetry에는 큐 관리자당 50,000개의 열린 연결이라는 초기 목표가 있습니다.

연결은 IBM MQ Explorer를 사용하여 관리됩니다. IBM MQ Explorer는 관리 가능한 숫자만을 표시하도록 연결을 필터합니다. 적절하게 선택된, 클라이언트에 대한 ID 할당 설계를 사용하면 지정학적으로, 또는 환자 이름의 영문자 순서대로 연결을 필터할 수도 있습니다.

Windows Linux AIX 텔레메트리 유스 케이스: 가정 에너지 모니터링 및 제어

스마트 측정기는 기존 측정기보다 에너지 소비에 대한 세부 사항을 더 많이 수집합니다.

스마트 측정기는 가정의 개별 어플라이언스를 모니터 및 제어하기 위해 보통 로컬 텔레메트리 네트워크와 결합됩니다. 몇몇은 원거리에서 모니터링과 제어를 할 수 있도록 원격으로도 연결됩니다.

원격 연결은 개별 기기, 전원 유틸리티 및 중앙 제어 지점에 의해 설정될 수 있습니다. 원격 제어 지점은 전력 사용을 읽고 사용 데이터를 제공할 수 있습니다. 이는 연속적인 가격 설정이나 날씨 정보에 영향을 줄 수 있는 데이터를 제공할 수 있습니다. 전체적인 전력 생산 효율을 높이기 위해 부하를 제한할 수도 있습니다.

스마트 측정기는 점점 더 널리 보급되고 있습니다. 예를 들면, 영국 정부는 2020년까지 모든 국내 가정에 스마트 측정기를 보급하는 것에 대해 협의 중에 있습니다.

가정 측정 유스 케이스에는 다음과 같은 몇 가지 공용되는 특성이 있습니다.

투명성

사용자가 측정기를 사용하여 에너지를 절약하고자 하지 않는 이상 측정기는 사용자 개입을 필요로 하지 않습니다. 이는 개별 어플라이언스에 대한 에너지 공급 안정성을 저하시키지 않아야 합니다.

MQTT 클라이언트는 측정기와 함께 배치된 소프트웨어에 임베드될 수 있으며 별도의 설치나 구성이 필요하지 않습니다.

불규칙한 연결

어플라이언스와 스마트 측정기는 측정기와 원격 연결 지점 사이의 연결과는 다른 연결성 기준을 요구합니다.

스마트 측정기에서 어플라이언스로의 연결은 가용성이 높아야 하며 홈 네트워크의 네트워크 기준에 부합해야 합니다.

원격 네트워크는 다양한 실제 접속을 사용할 가능성이 높습니다. 이 중 무선과 같은 몇몇은 전송 비용이 높으며 단속적일 수 있습니다. MQTT v3 스펙은 원격 연결 및 로컬 어댑터와 스마트 측정기 사이의 연결을 주요 대상으로 하고 있습니다.

전원 콘센트와 어플라이언스, 그리고 측정기 사이의 연결은 Zigbee와 같은 홈 네트워크를 사용합니다. 센서 네트워크용 MQTT(MQTT-S)는 Zigbee 및 기타 낮은 대역폭 네트워크 프로토콜과 작업하도록 설계되었습니다. MQ Telemetry는 MQTT-S를 직접 지원하지 않습니다. MQTT-S를 MQTT v3에 연결하려면 게이트웨이가 필요합니다.

가정 환자 모니터링과 마찬가지로 가정 에너지 모니터링 및 제어에 대한 솔루션은 다중 네트워크를 필요로 하며 이는 스마트 측정기를 허브로 하여 연결되어 있어야 합니다.

보안

스마트 측정기와 연관된 보안 문제에는 몇 가지 있습니다. 이 문제에는 트랜잭션 부인 방지, 시작된 모든 제어 조치의 권한 부여와 전력 소비 데이터의 개인정보 보호가 포함되어 있습니다.

확실한 개인정보 보호를 위해 측정기와 원격 제어 지점 사이에 MQTT로 전송된 데이터는 TLS를 사용하여 암호화될 수 있습니다. 제어 조치에 대한 확실한 권한 부여를 위해 측정기와 원격 제어 지점 사이의 MQTT 연결은 TLS를 사용하여 상호 인증하도록 할 수 있습니다.

연결성

원격 네트워크의 실제 환경은 매우 다양할 수 있습니다. 기존 광대역 연결을 사용하거나 높은 비용을 지불하며 모바일 네트워크를 사용할 수도 있고, 가용성은 단속적일 수 있습니다. 고비용이며 간헐적인 연결에 대해 MQTT는 효율적이며 안정적인 프로토콜입니다. 98 페이지의 『텔레메트리 유스 케이스: 가정 환자 모니터링』의 내용을 참조하십시오.

확장성

전력 회사나 중앙 제어 지점은 언젠가 수천만 개의 스마트 측정기를 설치하려고 계획하고 있습니다. 처음 배치 당 측정기의 수는 수만에서 수십만 단위입니다. 이 숫자는 초기 MQTT 목표였던 큐 관리자당 50,000개의 열린 클라이언트 연결과 비슷합니다.

가정 에너지 모니터링 및 제어의 아키텍처가 가진 중요한 면은 스마트 측정기를 네트워크 집선기로서 사용한다는 점입니다. 각 어플라이언트 어댑터는 별도의 센서입니다. 이들을 MQTT를 사용하여 로컬 허브에 연결함으로써 허브는 중앙 제어 지점과의 단일 TCP/IP 세션으로 데이터 플로우를 집중시킬 수 있으며 세션 정전 시를 대비해 짧은 시간 동안 메시지를 저장할 수도 있습니다.

가정 에너지 유스 케이스에서 원격 연결은 두 가지 이유로 인해 열려 있어야 합니다. 첫째, 연결을 여는 것은 요청을 송신하는 것과 비교해 시간이 오래 걸리기 때문입니다. 짧은 간격 안에 "부하 제한" 요청을 보내기 위해 많은 연결을 여는 데는 시간이 너무 오래 걸립니다. 둘째, 전력 회사로부터 부하 제한 요청을 수신하려면 클라이언트로부터 먼저 연결이 열려야 합니다. MQTT를 사용하면 연결은 항상 클라이언트에 의해 시작되며 전력 회사로부터 부하 제한 요청을 수신하기 위해서 연결은 열린 채로 남겨져 있어야 합니다.

연결 열기 등급이 중요이거나 서버가 시간에 쫓기는 요청을 시작할 경우 보통 솔루션은 많은 열린 연결을 유지하는 것입니다.

Windows Linux AIX 텔레메트리 유스 케이스: RFID(Radio Frequency Identification)

RFID는 오브젝트를 무선으로 식별하고 추적하는 임베드된 RFID 태그 사용법입니다. RFID 태그는 수 미터 범위 안에서, 태그와 PFID 리더 사이에 물체가 있더라도 읽을 수 있습니다. 수동 태그는 RFID 리더에 의해 활성화됩니다. 활성 태그는 외부에서 작동시키지 않더라도 전송됩니다. 활성 태그에는 전원이 있어야 합니다. 수동 태그는 그 범위를 늘리기 위해 전원이 있을 수 있습니다.

RFID는 많은 애플리케이션과 수많은 유형의 유스 케이스에서 사용됩니다. RFID 유스 케이스에는 가정 환자 모니터링, 가정 에너지 모니터링, 제어 유스 케이스와 몇 가지 유사한 점과 다른 점이 있습니다.

투명성

많은 유스 케이스에서 RFID 리더는 다수 배치되며 사용자 개입 없이 작동해야 합니다. 리더는 중앙 제어 지점과의 통신을 위한 임베드된 MQTT 클라이언트를 포함하고 있습니다.

예를 들면, 유통 창고에서 리더는 팰릿을 감지하는 데 동작 센서를 사용합니다. 이는 팰릿 위에 놓인 물건의 RFID 태그를 활성화시키며 데이터와 요청을 중앙 애플리케이션으로 송신합니다. 이 데이터는 재고의 위치를 업데이트하는 데 사용됩니다. 요청은 팰릿에 수행될 다음 작업(예: 특정 공간으로 이를 이동시키는 등)을 제어합니다. 항공사나 공항 수화물 시스템은 이런 방식으로 RFID를 사용합니다.

일부 RFID 유스 케이스에서는 리더에 표준 컴퓨팅 환경(예: Java ME(Java Platform, Micro Edition))이 있습니다. 이러한 경우 MQTT 클라이언트는 제조 후 고유한 구성 단계를 거쳐 배치될 수 있습니다.

불규칙한 연결

RFID 리더는 MQTT 클라이언트를 포함하고 있는 로컬 제어 디바이스와 분리될 수 있으며, 각 리더가 MQTT 클라이언트를 임베드할 수도 있습니다. 일반적으로 지형적 또는 통신상의 요인이 토폴로지를 결정하게 됩니다.

보안

RFID 태그를 사용에 있어서 개인정보 보호 및 신뢰성은 주요 보안 대상입니다. RFID 태그는 눈에 잘 띄지 않으며 비밀리에 모니터, 위조나 도용의 대상이 될 수 있습니다.

RFID 보안 문제에 대한 솔루션은 새 RFID 솔루션이 배치될 기회를 증가시킵니다. 그러나 보안 취약점은 주로 RFID 태그에 있으며, 중앙 정보 처리를 사용하는 로컬 리더는 다양한 위협을 제거할 방법을 제시해 줍니다. 예를 들면, 태그 도용은 입고와 출고에 대해 동적으로 상호 연관된 재고량을 통해 발견될 수 있습니다.

연결성

RFID 애플리케이션은 보통 RFID 리더에서 수집한 일괄 처리되는 저장 후 전달 정보, 그리고 즉각적 조회를 모두 포함하고 있습니다. 유통 창고 유스 케이스에서 RFID 리더는 항상 연결되어 있습니다. 태그가 읽히면 이는 리더에 관한 정보와 함께 발행됩니다. 창고 애플리케이션은 리더에게 응답을 발행합니다.

창고 애플리케이션에서 네트워크는 일반적으로 안정적이며 즉각적 요청은 낮은 지연시간 성능을 위해 전송 후 삭제 메시지를 사용할 수 있습니다. 일괄처리되는 저장 후 전달 데이터는 데이터 삭제와 연관된 관리 비용을 최소화하기 위해 정확히 한 번 메시징을 사용할 수 있습니다.

확장성

RFID 애플리케이션이 1 - 2초 정도 소요되는 즉각적 응답을 필요로 할 경우 RFID 리더는 계속 연결되어 있어야 합니다.

Windows Linux AIX 텔레메트리 유스 케이스: 환경 감지

환경 감지는 강물의 깊이와 수질, 대기 오염원 및 기타 환경 데이터를 수집하는 데 텔레메트리를 사용합니다.

센서는 보통 외진 곳에, 유선 통신에 대한 액세스 없이 설치됩니다. 무선 대역폭은 비경제적이며 안정성은 낮을 수 있습니다. 작은 공간에 있는 다수의 환경 센서가 안전한 장소의 로컬 모니터링 디바이스에 연결되어 있는 것이 전형입니다. 로컬 연결은 유선 또는 무선입니다.

투명성

센서 디바이스는 보통 중앙 모니터링 디바이스보다 액세스하기 어렵고, 전력을 덜 사용하며 다수 배치됩니다. 센서는 때때로 "조용하며" 로컬 모니터링 디바이스는 센서 데이터를 변환하고 저장하는 어댑터를 포함하고 있습니다. 모니터링 디바이스에는 Java SE(Java Platform, Standard Edition) 또는 Java ME(Java Platform, Micro Edition)를 지원하는 범용 컴퓨터가 포함되어 있을 가능성이 큽니다. 투명성은 MQTT 클라이언트를 구성할 때 주요 요구사항은 아닙니다.

불규칙한 연결

센서의 기능, 원격 연결의 비용과 대역폭 때문에 보통 로컬 모니터링 허브는 중앙 서버에 연결됩니다.

보안

솔루션이 군사 또는 방어 관련 유스 케이스에서 사용되는 것이 아닐 경우 보안은 주요 요구사항이 아닙니다.

연결성

대부분의 경우 연속적 모니터링이나 데이터의 즉시 가용성이 필요하지 않습니다. 홍수 레벨 경보와 같은 예외 데이터는 즉시 전달되어야 할 필요가 있습니다. 센서 데이터는 연결 및 통신 비용 절감을 위해 로컬 모니터에서 집계되어 스케줄된 연결을 사용하여 전송됩니다. 예외 데이터는 모니터에서 감지되는 즉시 전달됩니다.

확장성

센서는 로컬 허브에 집중되어 있으며 센서 데이터는 스케줄에 따라 전송되는 패킷에 집계됩니다. 이 두 요소가 직접 연결된 센서를 사용할 때 발생할 수 있는 중앙 서버의 부담을 덜어줍니다.

Windows Linux AIX 텔레메트리 유스 케이스: 모바일 애플리케이션

모바일 애플리케이션은 무선 디바이스에서 실행되는 애플리케이션입니다. 이 디바이스는 일반 애플리케이션 플랫폼이거나 특수 디바이스입니다.

일반 플랫폼에는 휴대전화나 PDA와 같은 소형 디바이스나 노트북 컴퓨터와 같은 휴대용 디바이스가 포함됩니다. 특수 디바이스는 특정 애플리케이션에 맞춰진 특수 목적 하드웨어를 사용합니다. 특수 모바일 디바이스의 한 예로는 "전달 확인" 소포 전달을 기록하는 디바이스가 있습니다. 특수 모바일 디바이스의 애플리케이션은 보통 일반 소프트웨어 플랫폼에서 빌드됩니다.

투명성

사용자 정의 모바일 애플리케이션의 배치가 관리되며 MQTT 클라이언트 애플리케이션 구성을 포함할 수 있습니다. 투명성은 MQTT 클라이언트를 구성하는 데 주된 요구사항은 아닙니다.

불규칙한 연결

앞에 나온 유스 케이스의 로컬 허브 토폴로지와 달리 모바일 클라이언트는 원격으로 연결합니다. 클라이언트 애플리케이션 계층은 중앙 허브의 애플리케이션으로 직접 연결합니다.

보안

물리적 보안성이 높지 않기 때문에 모바일 디바이스와 모바일 사용자는 인증을 받아야 합니다. 디바이스 ID를 확인하는 데는 TLS가, 사용자를 인증하는 데는 JAAS가 사용됩니다.

연결성

모바일 애플리케이션이 무선 범위에 종속적일 경우 이는 오프라인으로 조작이 가능하며 인터럽트된 연결을 효율적으로 다룰 수 있어야 합니다. 이와 같은 환경에서 목표는 연결을 유지하는 동시에 애플리케이션에서 메시지를 저장하고 전달할 수 있게 하는 것입니다. 메시지는 종종 명령이나 전달 확인인 경우가 있으며 중요한 비즈니스 가치가 있습니다. 이들은 안정적으로 저장 및 전달되어야 합니다.

확장성

확장성은 주요 문제는 아닙니다. 사용자 정의 모바일 애플리케이션 유스 케이스에서 애플리케이션 클라이언트의 수는 수천 또는 수만을 넘어가지 않을 가능성이 큽니다.

Windows Linux AIX 텔레메트리 디바이스를 큐 관리자에 연결

텔레메트리 디바이스는 MQTT v3 클라이언트를 사용하여 큐 관리자에 연결합니다. MQTT v3 클라이언트는 TCP/IP를 사용하여 텔레메트리(MQXR) 서비스라고 하는 TCP/IP 리스너에 연결합니다.

텔레메트리 디바이스를 큐 관리자에 연결하는 경우, MQTT 클라이언트는 `MqttClient.connect` 메소드를 사용하여 TCP/IP 연결을 시작합니다. IBM MQ 클라이언트처럼 MQTT 클라이언트는 메시지를 송신하고 수신하려면 큐 관리자에 연결되어 있어야 합니다. 연결은 텔레메트리(MQXR) 서비스라고 하는 MQ Telemetry와 함께 설치된 TCP/IP 리스너를 사용하여 서버에서 이루어집니다. 각각의 큐 관리자는 최대로 하나의 텔레메트리(MQXR) 서비스를 실행합니다.

텔레메트리(MQXR) 서비스는 텔레메트리 채널에 대한 연결을 할당하기 위해 `MqttClient.connect` 메소드에서 각 클라이언트가 설정한 원격 소켓 주소를 사용합니다. 소켓 주소는 TCP/IP 호스트 이름과 포트 번호의 조합입니다. 동일한 원격 소켓 주소를 사용하는 다수의 클라이언트는 텔레메트리(MQXR) 서비스에 의해 동일한 텔레메트리 채널에 연결됩니다.

서버에 다수의 큐 관리자가 있을 경우 큐 관리자 사이에 텔레메트리 채널을 분할하십시오. 큐 관리자 사이에 원격 소켓 주소를 할당하십시오. 각 텔레메트리 채널을 고유한 원격 소켓 주소로 정의하십시오. 두 텔레메트리 채널은 같은 소켓 주소를 사용해서는 안됩니다.

다수의 큐 관리자에 텔레메트리 채널에 대해 같은 소켓 주소가 구성되어 있을 경우 처음 연결하는 텔레메트리 채널이 남게 됩니다. 동일한 주소에 연결하는 후속 채널은 실패합니다.

서버에 다수의 네트워크 어댑터가 있을 경우 텔레메트리 채널 사이에 원격 소켓 주소를 분할하십시오. 하나의 특정 소켓 주소라도 유일한 텔레메트리 채널에 구성되어 있는 한 소켓 주소 할당은 완전히 임의로 이뤄집니다.

IBM MQ Explorer용 MQ Telemetry 부록에 제공된 마법사를 사용하여 MQTT 클라이언트를 연결하도록 IBM MQ 를 구성하십시오. 또는 수동으로 텔레메트리를 구성하려면 Linux 및 AIX에서 텔레메트리에 대해 큐 관리자 구성 및 Windows에서 텔레메트리에 대해 큐 관리자 구성의 지시사항을 따르십시오.

관련 참조

MQXR 특성

Windows Linux AIX 텔레메트리 연결 프로토콜

MQ Telemetry는 TCP/IP IPv4 및 IPv6과 TLS를 지원합니다.

Windows Linux AIX 텔레메트리(MQXR) 서비스

텔레메트리(MQXR) 서비스는 TCP/IP 리스너이며, IBM MQ 서비스로 관리됩니다. IBM MQ Explorer 마법사나 **runmqsc** 명령을 사용하여 서비스를 작성하십시오.

MQ Telemetry (MQXR) 서비스를 SYSTEM.MQXR.SERVICE .

IBM MQ Explorer용 MQ Telemetry 함수에서 제공되는 텔레메트리 샘플 구성 마법사는 텔레메트리 서비스 및 샘플 텔레메트리 채널을 작성합니다. IBM MQ Explorer 를 사용하여 MQ Telemetry 설치 확인 을 참조하십시오.

명령행에서 샘플 구성을 작성하십시오. 명령행을 사용한 MQ Telemetry 설치 확인을 참조하십시오.

텔레메트리(MQXR) 서비스는 큐 관리자에서 자동으로 시작되고 중지됩니다. IBM MQ Explorer의 서비스 폴더를 사용하여 서비스를 제어하십시오. 서비스를 보려면 아이콘을 클릭하여 IBM MQ Explorer 표시에서 SYSTEM 오브젝트 필터링을 중지해야 합니다.

서비스를 수동으로 작성하는 방법에 대한 예제는 다음을 참조하십시오.

- Linux AIX Linux에서 SYSTEM.MQXR.SERVICE 작성.
- Windows Windows에서 SYSTEM.MQXR.SERVICE 작성.

V 9.3.0 IBM MQ 9.3.0 부터 Linux에서 SYSTEM.MQXR.SERVICE 작성및 Windows에서 SYSTEM.MQXR.SERVICE 작성 이 MQTT TLS 채널의 비밀번호 문구를 암호화해야 하는 기본 키를 지정하도록 업데이트됩니다. 자세한 정보는 MQTTTLS 채널에 대한 비밀번호 문구의 암호화를 참조하십시오.

Windows Linux AIX 텔레메트리 채널

Java 인증 및 권한 부여 서비스(JAAS)나 TLS 인증과 같은 다양한 특성과 연결을 작성하기 위해서나 클라이언트 그룹을 관리하려면 텔레메트리 채널을 작성하십시오.

IBM MQ Explorer용 MQ Telemetry 함수에 제공된 **New Telemetry Channel** 마법사를 사용하여 텔레메트리 채널을 작성하십시오. 특정 TCP/IP의 MQTT 클라이언트로부터 연결을 허용하려면 마법사를 사용하여 채널을 구성하십시오. IBM WebSphere MQ 7.1부터 명령행 프로그램 **runmqsc**를 사용하여 MQ Telemetry 를 구성할 수 있습니다.

클라이언트를 그룹으로 나눠 대규모 클라이언트 연결을 관리하기 쉽도록 서로 다른 포트에 다중 텔레메트리 채널을 작성하십시오. 각 텔레메트리 채널은 서로 다른 이름을 가지고 있습니다.

다양한 유형의 연결을 작성하기 위해 다양한 보안 속성으로 텔레메트리 채널을 구성할 수 있습니다. 다양한 TCP/IP 주소로부터 클라이언트 연결을 수락하려면 다중 채널을 작성하십시오. TLS를 사용하여 메시지를 암호화하고 텔레메트리 채널 및 클라이언트를 인증하십시오. MQTT 클라이언트 및 텔레메트리 채널의 TLS 구성을 참조하십시오. IBM MQ 오브젝트에 대한 액세스 권한을 부여하는 작업을 단순화하려면 사용자 ID를 지정하십시오. JAAS로 MQTT 사용자를 인증할 수 있도록 JAAS 구성을 지정하십시오. MQTT 클라이언트 식별, 권한 부여 및 인증을 참조하십시오.

Windows Linux AIX IBM MQ Telemetry Transport 프로토콜

IBM MQ Telemetry Transport (MQTT) v3 프로토콜은 낮은 대역폭 또는 비용이 많이 드는 연결에서 소형 디바이스 간에 메시지를 교환하고 메시지를 안정적으로 전송하도록 설계되었습니다. TCP/IP를 사용합니다.

MQTT protocol은 발행됩니다. IBM MQ Telemetry Transport 형식 및 프로토콜을 참조하십시오. 프로토콜 버전 3은 발행/구독을 사용하며 다음과 같은 세 가지 서비스 품질을 제공합니다. 전송 후 삭제, 적어도 한 번 및 정확히 한 번

프로토콜 헤더의 작은 크기와 바이트 배열 메시지 페이로드로 인해 메시지 용량은 작습니다. 헤더는 2바이트의 고정된 헤더와 12바이트까지의 추가 변수 헤더를 포함합니다. 프로토콜은 구독 및 연결을 하는 데 12바이트의 변수 헤더를 사용하며 대부분의 발행에 대한 변수 헤더에는 2바이트만을 사용합니다.

세 가지 서비스 품질(QoS)을 사용하여 낮은 지연 시간과 안정성 사이에서 적절하게 균형을 유지할 수 있습니다. MQTT 클라이언트가 제공하는 서비스 품질(QoS)을 참조하십시오. 전송 후 삭제는 지속적 디바이스 스토리지를 사용하지 않으며 발행물 송신 및 수신에 하나의 전송만을 사용합니다. 적어도 한 번 및 정확히 한 번에는 인식될 때까지 프로토콜 상태를 유지하고 메시지를 저장할 지속적 스토리지가 디바이스에 필요합니다.

Windows Linux AIX MQTT 클라이언트

MQTT 클라이언트 애플리케이션은 Telemetry 디바이스에서 정보를 수집하고 서버에 연결하며 정보를 서버에 발행하는 역할을 합니다. 또한 토픽을 구독하고 서적을 수신하며 텔레메트리 디바이스를 제어할 수도 있습니다.

IBM MQ 클라이언트 애플리케이션과 달리 MQTT 클라이언트 애플리케이션은 IBM MQ 애플리케이션이 아닙니다. 이 클라이언트는 연결할 큐 관리자를 지정하지 않습니다. 특정 IBM MQ 프로그래밍 인터페이스를 사용해야 하는 제약도 받지 않습니다. 대신에, MQTT 클라이언트는 MQTT 3 프로토콜을 구현합니다. 프로그래밍 언어를 사용하여 선택한 플랫폼에서 MQTT protocol에 대한 클라이언트 라이브러리의 인터페이스를 작성할 수 있습니다. IBM MQ Telemetry Transport 형식 및 프로토콜을 참조하십시오.

MQTT 클라이언트 앱 작성을 단순화하려면 여러 플랫폼에 대해 MQTT protocol 를 캡슐화하는 C, Java및 JavaScript 클라이언트 라이브러리를 사용하십시오. 이러한 라이브러리를 MQTT 앱에 통합하면 완전한 기능을 갖춘 MQTT 클라이언트가 15행의 코드로 짧을 수 있습니다. MQTT 클라이언트 라이브러리는 Eclipse Paho 및 MQTT.org에서 무료로 제공됩니다. IBM MQ Telemetry Transport 샘플 프로그램을 참조하십시오.

MQTT 클라이언트 애플리케이션은 항상 텔레메트리 채널로 연결을 시작합니다. 연결된 후 MQTT 클라이언트 애플리케이션이나 IBM MQ 애플리케이션 모두에서 메시지 교환을 시작할 수 있습니다.

MQTT 클라이언트 애플리케이션과 IBM MQ 애플리케이션은 동일한 토픽 세트에 발행 및 구독합니다. IBM MQ 애플리케이션은 클라이언트 애플리케이션이 구독을 먼저 작성하지 않고 MQTT 클라이언트 애플리케이션에 직접 메시지를 송신할 수 있습니다. MQTT 클라이언트에 메시지를 송신하도록 분산 큐잉 구성을 참조하십시오.

MQTT 클라이언트 애플리케이션은 텔레메트리 채널을 사용하여 IBM MQ에 연결됩니다. 텔레메트리 채널은 MQTT 및 IBM MQ가 사용하는 서로 다른 유형의 메시지 사이에서 브릿지 역할을 수행합니다. 이는 MQTT 클라이언트 애플리케이션을 대신하여 큐 관리자에서 발행 및 구독을 작성합니다. 텔레메트리 채널은 MQTT 클라이언트 애플리케이션의 구독과 일치하는 발행을 큐 관리자에서 MQTT 클라이언트 애플리케이션으로 송신합니다.

Windows Linux AIX MQTT 클라이언트에 메시지 송신

IBM MQ 애플리케이션은 클라이언트가 작성한 구독을 발행하거나 직접 메시지를 보내서 MQTT v3 클라이언트 메시지를 보낼 수 있습니다. MQTT 클라이언트는 다른 클라이언트가 구독한 토픽을 발행하는 방식으로 메시지를 다른 클라이언트에게 보낼 수 있습니다.

MQTT 클라이언트가 IBM MQ에서 수신한 발행을 구독함

IBM MQ 에서 MQTT 클라이언트로 발행물을 보내려면 107 페이지의 『IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행』 태스크를 수행하십시오.

MQTT v3 클라이언트가 메시지를 수신하는 일반적인 방법은 토픽 또는 토픽 세트에 구독을 작성하는 것입니다. 이 예제 코드 스니펫 106 페이지의 그림 44에서는 MQTT 클라이언트가 토픽 문자열 `"MQTT Examples"`을 사용하여 구독합니다. IBM MQ C 애플리케이션, 106 페이지의 그림 45는 토픽 문자열 `"MQTT Examples"`를 사용하여 토픽을 발행합니다. 이 코드 스니펫 106 페이지의 그림 46에서는 MQTT 클라이언트가 콜백 메소드 `messageArrived`로 발행을 수신합니다.

MQTT 클라이언트의 구독에 대한 응답으로 발행물을 송신하도록 IBM MQ 를 구성하는 방법에 대한 자세한 정보는 MQTT 클라이언트 구독에 대한 응답으로 메시지 발행을 참조하십시오.

IBM MQ 애플리케이션이 MQTT 클라이언트로 직접 메시지를 보냄

IBM MQ 에서 MQTT 클라이언트로 직접 메시지를 보내려면 111 페이지의 『IBM MQ Explorer 를 사용하여 MQTT 클라이언트에 메시지 송신』 태스크를 수행하십시오.

이러한 방식으로 MQTT 클라이언트로 보낸 메시지를 요청하지 않은 메시지라고 합니다. MQTT v3 클라이언트는 토픽 이름 세트가 포함된 발행으로 요청하지 않은 메시지를 수신합니다. 텔레메트리(MQXR) 서비스는 토픽 이름을 리모트 큐 이름으로 설정합니다.

MQTT 클라이언트에 직접 메시지를 보내도록 IBM MQ 를 구성하는 방법에 대한 자세한 정보는 클라이언트에 직접 메시지 보내기를 참조하십시오.

MQTT 클라이언트가 메시지를 발행함

MQTT v3 클라이언트는 다른 MQTT v3 클라이언트가 수신한 메시지를 발행할 수 있지만 요청하지 않은 메시지는 보낼 수 없습니다. 코드 스니펫 107 페이지의 그림 47 에서는 Java로 작성된 MQTT v3 클라이언트가 메시지를 발행하는 방법을 보여줍니다.

특정 MQTT v3 클라이언트로 메시지를 보내는 일반적인 방법은 각 클라이언트가 고유의 `ClientIdentifier`에 구독을 작성하는 것입니다. `ClientIdentifier`를 토픽 문자열로 사용하여 한 MQTT 클라이언트에서 다른 MQTT 클라이언트로 메시지를 발행하려면 112 페이지의 『특정 MQTT v3 클라이언트로 메시지 발행』 태스크를 수행하십시오.

예제 코드 스니펫

106 페이지의 그림 44 의 코드 스니펫은 Java 로 작성된 MQTT 클라이언트가 구독을 작성하는 방법을 보여줍니다. 구독에 대한 발행을 수신하려면 `messageArrived`라는 콜백 메소드도 필요합니다.

```
String    clientId = String.format("%-23.23s",
                     System.getProperty("user.name") + "_" +
                     (UUID.randomUUID().toString())).trim()).replace('-', '_');
MqttClient  client = new MqttClient("localhost", clientId);
String topicString = "MQTT Examples";
int            QoS = 1;
client.subscribe(topicString, QoS);
```

그림 *44. MQTT v3* 클라이언트 구독자

106 페이지의 그림 45의 코드 스니펫에서는 C로 작성된 IBM MQ 애플리케이션이 발행을 보내는 방법을 보여줍니다. 코드 스니펫은 변수 토픽에 발행자 작성 태스크를 통해 추출됩니다.

```
/* Define and set variables to.defaults */
/* Omitted lines declaring variables    */
char * topicName   = ""
char * topicString = "MQTT Examples"
char * publication = "Hello world!";
do {
  MQCONN(qMgrName, &Hconn, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  td.ObjectType = MQOT_TOPIC;    /* Object is a topic             */
  td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4     */
  strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
  td.ObjectString.VSPtr = topicString;
  td.ObjectString.VSLength = (MQLONG)strlen(topicString);
  MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
  MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  MQDISC(&Hconn, &CompCode, &Reason);
} while (0);
```

그림 *45. IBM MQ* 발행자

발행물이 도착하면 MQTT 클라이언트는 MQTT 애플리케이션 클라이언트 `MqttCallback` 클래스의 `messageArrived` 메소드를 호출합니다.

```
public class CallBack implements MqttCallback {
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
// ... Other callback methods
}
```

그림 *46. `messageArrived`* 메소드

107 페이지의 그림 47에서는 MQTT v3가 106 페이지의 그림 44에서 작성된 구독에 메시지를 발행하는 방법을 보여줍니다.

```
String          address = "localhost";
String         clientId = String.format("%-23.23s",
                          System.getProperty("user.name") + "_" +
                          (UUID.randomUUID().toString())).trim()).replace('-', '_');
MqttClient       client = new MqttClient(address, clientId);
String      topicString = "MQTT Examples";
MqttTopic         topic = client.getTopic(Example.topicString);
String     publication  = "Hello world";
MqttMessage     message = new MqttMessage(publication.getBytes());
MqttDeliveryToken token = topic.publish(message);
```

그림 *47. MQTT v3* 클라이언트 발행자

Windows Linux AIX IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행

IBM MQ Explorer를 사용하여 메시지를 발행하고 MQTT 클라이언트 유틸리티를 사용하여 해당 메시지를 구독하려면 다음 태스크의 단계를 수행하십시오. 다음 태스크에서는 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하지 않고 큐 관리자 알리어스를 구성하는 방법을 보여줍니다.

시작하기 전에

이 태스크에서는 사용자가 IBM MQ 및 IBM MQ Explorer에 익숙하고 IBM MQ 및 MQ Telemetry 기능이 설치되어 있다고 가정합니다.

이 태스크에서 큐 관리자 자원을 작성하는 사용자는 이에 대한 충분한 권한을 가지고 있어야 합니다. 데모를 위해 IBM MQ Explorer 사용자 ID가 mqm 그룹의 구성원이라고 가정합니다.

이 태스크 정보

이 태스크에서는 IBM MQ에 토픽을 작성하고 MQTT 클라이언트 유틸리티를 사용하여 해당 토픽을 구독합니다. IBM MQ Explorer를 사용하여 토픽을 발행하면 MQTT 클라이언트가 발행을 수신합니다.

프로시저

다음 태스크 중 하나를 수행하십시오.

- MQ Telemetry를 설치했지만 아직 시작하지 않았습니다. 108 페이지의 『텔레메트리(MQXR) 서비스가 아직 정의되지 않은 태스크 시작』 태스크를 수행하십시오.
- 이전에 IBM MQ Telemetry를 실행했지만 데모를 위해 새 큐 관리자를 사용하려고 합니다. 108 페이지의 『텔레메트리(MQXR) 서비스가 아직 정의되지 않은 태스크 시작』 태스크를 수행하십시오.
- 텔레메트리 자원이 정의되지 않은 기존의 큐 관리자를 사용하는 태스크를 수행하려고 합니다. **샘플 구성 정의** 마법사를 실행하지 않으려고 합니다.

  a. 텔레메트리를 설정하려면 다음 태스크 중 하나를 수행하십시오.

  - Linux 및 AIX에서 텔레메트리에 대해 큐 관리자 구성
  - Windows에서 텔레메트리에 대해 큐 관리자 구성

  b. 108 페이지의 『실행 중인 텔레메트리(MQXR) 서비스가 있는 태스크 시작』 태스크를 수행하십시오.
- 텔레메트리 자원이 이미 정의되어 있는 기존의 큐 관리자를 사용하는 태스크를 수행하려는 경우 108 페이지의 『실행 중인 텔레메트리(MQXR) 서비스가 있는 태스크 시작』 태스크를 수행하십시오.

다음에 수행할 작업

클라이언트 유틸리티로 직접 메시지를 보내려면 111 페이지의 『IBM MQ Explorer 를 사용하여 MQTT 클라이언트에 메시지 송신』 태스크를 수행하십시오.

텔레메트리(*MQXR*) 서비스가 아직 정의되지 않은 태스크 시작

큐 관리자를 작성하고 **샘플 구성 정의**를 실행하여 큐 관리자에 대한 샘플 텔레메트리 자원을 정의합니다. IBM MQ Explorer를 사용하여 메시지를 발행하고 MQTT 클라이언트 유틸리티로 해당 메시지를 구독합니다.

이 태스크 정보

샘플 구성 정의 마법사를 사용하여 샘플 텔레메트리 자원을 설정할 때 해당 마법사가 게스트 사용자 ID 권한을 설정합니다. 이러한 방식으로 게스트 사용자 ID에 권한을 부여할지 신중하게 고려하십시오. Windows의 `guest` 및 Linux의 `nobody` 에는 토픽 트리의 루트를 발행하고 구독하며 `SYSTEM.MQTT.TRANSMIT.QUEUE`에 메시지를 넣을 수 있는 권한이 부여됩니다.

또한 마법사는 기본 전송 큐를 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 설정하는데, 이는 기존의 큐 관리자에서 실행 중인 애플리케이션에 지장을 줄 수 있습니다. 기본 전송 큐를 사용하지 않고 텔레메트리를 구성하는 것은 가능하지만 번거로운 작업입니다. 110 페이지의 『큐 관리자 알리어스 사용』 태스크에 따라 작업을 수행하십시오. 이 태스크에서는 큐 관리자를 작성하여 기존의 기본 전송 큐에 지장을 주지 않도록 합니다.

프로시저

1. IBM MQ Explorer를 사용하여 새 큐 관리자를 작성하고 시작하십시오.
   a) `Queue Managers` 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **큐 관리자...**를 클릭하십시오. 큐 관리자 이름을 입력하고 **마침**을 클릭하십시오.

      큐 관리자 이름을 작성하십시오(예: `MQTTQMGR`).
2. 텔레메트리(MQXR) 서비스를 작성 및 시작하고 샘플 텔레메트리 채널을 작성하십시오.
   a) `Queue Managers\`*`QmgrName`*`\Telemetry` 폴더를 여십시오.
   b) **샘플 구성 정의...** > **마침**을 클릭하십시오.

      MQTT 클라이언트 유틸리티 시작 선택란을 선택하십시오.
3. MQTT 클라이언트 유틸리티를 사용하여 `MQTT Example` 에 대한 구독을 작성하십시오.
   a) **연결**을 클릭하십시오.

      클라이언트 실행 기록은 `Connected` 이벤트를 기록합니다.
   b) `MQTT Example`을 **Subscription\Topic** 필드에 입력하고 **구독**을 입력하십시오.

   클라이언트 실행 기록은 `Subscribed` 이벤트를 기록합니다.
4. IBM MQ에서 `MQTTExampleTopic`을 작성하십시오.
   a) **MQ 탐색기**에서 `Queue Managers\`*`QmgrName`*`\Topics` 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **토픽**을 클릭하십시오.
   b) `MQTTExampleTopic`을 **이름**으로 입력하고 **다음**을 클릭하십시오.
   c) `MQTT Example`을 **토픽 문자열**로 입력하고 **마침**을 클릭하십시오.
   d) 수신확인 창을 닫으려면 **확인**을 클릭하십시오.
5. IBM MQ Explorer를 사용하여 `Hello World!` 를 `MQTT Example` 토픽에 발행하십시오.
   a) IBM MQ Explorer에서 `Queue Managers\`*`QmgrName`*`\Topics` 폴더를 클릭하십시오.
   b) `MQTTExampleTopic`을 마우스 오른쪽 단추로 클릭하고 **발행물 테스트...**를 클릭하십시오.
   c) **메시지 데이터** 필드에 `Hello World!` 를 입력하고 > **메시지 발행** > MQTT 클라이언트 유틸리티 창으로 전환하십시오.

   클라이언트 실행 기록은 `Received` 이벤트를 기록합니다.

실행 중인 텔레메트리(*MQXR*) 서비스가 있는 태스크 시작

텔레메트리 채널 및 토픽을 작성합니다. 사용자에게 토픽 및 텔레메트리 전송 큐를 사용할 수 있는 권한을 부여합니다. IBM MQ Explorer를 사용하여 메시지를 발행하고 MQTT 클라이언트 유틸리티로 해당 메시지를 구독합니다.

시작하기 전에

이 태스크 버전에서는 큐 관리자 *QmgrName*이 정의되어 실행되고 있습니다. 텔레메트리(MQXR) 서비스가 정의되어 실행 중입니다. 텔레메트리(MQXR) 서비스는 수동으로 또는 **샘플 구성 정의** 마법사를 실행하여 작성되었을 수 있습니다.

이 태스크 정보

이 태스크에서는 기존 큐 관리자를 구성하여 발행을 MQTT 클라이언트 유틸리티로 보냅니다.

태스크의 109 페이지의 『1』 단계에서는 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하는데, 이는 기존의 큐 관리자에서 실행 중인 애플리케이션에 지장을 줄 수 있습니다. 기본 전송 큐를 사용하지 않고 텔레메트리를 구성하는 것은 가능하지만 번거로운 작업입니다. 110 페이지의 『큐 관리자 알리어스 사용』 태스크에 따라 작업을 수행하십시오.

프로시저

1. SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 설정하십시오.
   a) Queue Managers*QmgrName* folder를 마우스 오른쪽 단추로 클릭하고 **특성...**을 클릭하십시오.
   b) 네비게이터에서 **통신**을 클릭하십시오.
   c) **선택...**을 클릭하고 SYSTEM.MQTT.TRANSMIT.QUEUE > **확인** > **확인**을 선택하십시오.
2. 텔레메트리 채널 MQTTExampleChannel을 작성하여 MQTT 클라이언트 유틸리티를 IBM MQ에 연결하고 MQTT 클라이언트 유틸리티를 시작하십시오.
   a) **MQ 탐색기**에서 Queue Managers*QmgrName* \Telemetry\Channels 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **텔레메트리 채널...**을 클릭하십시오.
   b) MQTTExampleChannel을 **채널 이름** 필드에 입력하고 > **다음** > **다음**을 클릭하십시오.
   c) 클라이언트 권한 패널에서 **고정 사용자 ID**를 MQTTExample를 발행하고 구독하려는 사용자 ID로 변경한 후 **다음**을 클릭하십시오.
   d) **클라이언트 유틸리티 시작**을 선택하고 **마침**을 클릭하십시오.
3. MQTT 클라이언트 유틸리티를 사용하여 MQTT Example 에 대한 구독을 작성하십시오.
   a) **연결**을 클릭하십시오.

      클라이언트 실행 기록은 Connected 이벤트를 기록합니다.
   b) MQTT Example을 **Subscription\Topic** 필드에 입력하고 **구독**을 입력하십시오.

   클라이언트 실행 기록은 Subscribed 이벤트를 기록합니다.
4. IBM MQ에서 MQTTExampleTopic을 작성하십시오.
   a) **MQ 탐색기**에서 Queue Managers*QmgrName*\Topics 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **토픽**을 클릭하십시오.
   b) MQTTExampleTopic을 **이름**으로 입력하고 **다음**을 클릭하십시오.
   c) MQTT Example을 **토픽 문자열**로 입력하고 **마침**을 클릭하십시오.
   d) 수신확인 창을 닫으려면 **확인**을 클릭하십시오.
5. mqm 그룹에 속하지 않은 사용자가 MQTTExample 토픽을 발행하고 구독하도록 하려면 다음을 수행하십시오.
   a) 사용자에게 MQTTExampleTopic 토픽을 발행 및 구독할 수 있는 권한을 부여하십시오.

   ```
   setmqaut -m qMgrName -t topic -n MQTTExampleTopic -p User ID -all +pub +sub
   ```

   b) 사용자에게 SYSTEM.MQTT.TRANSMIT.QUEUE에 메시지를 넣을 수 있는 권한을 부여하십시오.

   ```
   setmqaut -m qMgrName -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p User ID -all +put
   ```
6. IBM MQ Explorer를 사용하여 Hello World! 를 MQTT Example 토픽에 발행하십시오.

a) IBM MQ Explorer에서 Queue Managers*QmgrName*\Topics 폴더를 클릭하십시오.

b) MQTTExampleTopic을 마우스 오른쪽 단추로 클릭하고 **발행물 테스트...**를 클릭하십시오.

c) **메시지 데이터** 필드에 Hello World! 를 입력하고 > **메시지 발행** > MQTT 클라이언트 유틸리티 창으로 전환하십시오.

클라이언트 실행 기록은 Received 이벤트를 기록합니다.

큐 관리자 알리어스 사용

기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하지 않고 IBM MQ Explorer 를 사용하여 MQTT 클라이언트 유틸리티에 메시지를 발행하십시오.

이 태스크는 이전 태스크와 연결되는 것으로, 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하지 않도록 하기 위해 큐 관리자 알리어스를 사용합니다.

시작하기 전에

108 페이지의 『텔레메트리(MQXR) 서비스가 아직 정의되지 않은 태스크 시작』 태스크 또는 108 페이지의 『실행 중인 텔레메트리(MQXR) 서비스가 있는 태스크 시작』 태스크를 완료하십시오.

이 태스크 정보

MQTT 클라이언트가 구독을 작성하면 IBM MQ가 ClientIdentifier를 리모트 큐 관리자 이름으로 사용하여 해당 응답을 보냅니다. 이 태스크에서는 ClientIdentifier, MyClient를 사용합니다.

MyClient라는 큐 관리자 알리어스나 전송 큐가 없으면 응답이 기본 전송 큐에 배치됩니다. 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하면 MQTT 클라이언트가 응답을 받습니다.

큐 관리자 알리어스를 사용하면 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하지 않아도 됩니다. 각 ClientIdentifier에 대해 큐 관리자 알리어스를 설정해야 합니다. 일반적으로 클라이언트가 지나치게 많아 실제로 큐 관리자 알리어스를 사용할 수 없습니다. 대개 ClientIdentifier는 예측이 불가능하므로 이러한 방식으로 텔레메트리를 구성하는 것은 불가능합니다.

그럼에도 불구하고 특정 환경에서는 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE이외의 다른 설정으로 구성해야 합니다. 프로시저 의 단계에서는 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정하는 대신 큐 관리자 알리어스를 구성합니다.

프로시저

1. SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 제거하십시오.

   a) Queue Managers*QmgrName* folder를 마우스 오른쪽 단추로 클릭하고 **특성...**을 클릭하십시오.

   b) 네비게이터에서 **통신**을 클릭하십시오.

   c) **기본 전송 큐** 필드에서 SYSTEM.MQTT.TRANSMIT.QUEUE를 제거하고 **확인**을 클릭하십시오.

2. MQTT 클라이언트 유틸리티에서 더 이상 구독을 작성할 수 없는지 확인하십시오.

   a) **연결**을 클릭하십시오.

   클라이언트 실행 기록은 Connected 이벤트를 기록합니다.

   b) MQTT Example을 **Subscription\Topic** 필드에 입력하고 **구독**을 입력하십시오.

   클라이언트 실행 기록은 Subscribe failed 및 Connection lost 이벤트를 기록합니다.

3. ClientIdentifier, MyClient에 대한 큐 관리자 알리어스를 작성하십시오.

   a) Queue Managers*QmgrName*\Queues 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **리모트 큐 정의**를 클릭하십시오.

   b) 정의 이름을 MyClient로 지정하고 **다음**을 클릭하십시오.

   c) **리모트 큐 관리자** 필드에 MyClient를 입력하십시오.

   d) **전송 큐** 필드에 SYSTEM.MQTT.TRANSMIT.QUEUE를 입력하고 **마침**을 클릭하십시오.

4. MQTT 클라이언트 유틸리티를 다시 연결하십시오.
   a) **클라이언트 ID**가 MyClient로 설정되어 있는지 확인하십시오.
   b) **연결**

   클라이언트 실행 기록은 Connected 이벤트를 기록합니다.
5. MQTT 클라이언트 유틸리티를 사용하여 MQTT Example 에 대한 구독을 작성하십시오.
   a) **연결**을 클릭하십시오.

      클라이언트 실행 기록은 Connected 이벤트를 기록합니다.
   b) MQTT Example을 **Subscription\Topic** 필드에 입력하고 **구독**을 입력하십시오.

   클라이언트 실행 기록은 Subscribed 이벤트를 기록합니다.
6. IBM MQ Explorer를 사용하여 Hello World! 를 MQTT Example 토픽에 발행하십시오.
   a) IBM MQ Explorer에서 Queue Managers*QmgrName*\Topics 폴더를 클릭하십시오.
   b) MQTTExampleTopic을 마우스 오른쪽 단추로 클릭하고 **발행물 테스트...**를 클릭하십시오.
   c) **메시지 데이터** 필드에 Hello World! 를 입력하고 > **메시지 발행** > MQTT 클라이언트 유틸리티 창으로 전환하십시오.

   클라이언트 실행 기록은 Received 이벤트를 기록합니다.

Windows Linux AIX IBM MQ Explorer 를 사용하여 MQTT 클라이언트에 메시지 송신

IBM MQ Explorer를 사용하여 IBM MQ 큐에 메시지를 넣어 MQTT 클라이언트 유틸리티에 메시지를 송신하십시오. 다음 태스크에서는 MQTT 클라이언트로 직접 메시지를 보내도록 리모트 큐 정의를 구성하는 방법을 보여줍니다.

시작하기 전에

107 페이지의 『IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행』 태스크를 수행하십시오. MQTT 클라이언트 유틸리티를 연결된 상태로 두십시오.

이 태스크 정보

다음 태스크에서는 토픽을 발행하지 않고 큐를 사용하여 MQTT 클라이언트로 메시지를 보내는 방법을 보여줍니다. 클라이언트에서 구독을 작성하지 않습니다. 이 태스크의 111 페이지의 『2』 단계에서는 삭제된 이전 구독에 대해 설명합니다.

프로시저

1. MQTT 클라이언트 유틸리티의 연결을 끊었다가 다시 연결하여 기존의 구독을 모두 제거하십시오.

   기본값을 변경하지 않으면 MQTT 클라이언트 유틸리티가 정리 세션과 연결되므로 구독이 제거됩니다. 정리 세션을 참조하십시오.

   태스크를 더 쉽게 수행하려면 MQTT 클라이언트 유틸리티에서 작성한 생성된 ClientIdentifier 를 사용하지 말고 사용자 자신의 ClientIdentifier를 입력하십시오.

   a) **연결 끊기**를 클릭하여 텔레메트리 채널에서 MQTT 클라이언트 유틸리티의 연결을 끊으십시오.

      클라이언트 실행 기록은 Disconnected 이벤트를 기록합니다.
   b) **클라이언트 ID**를 MyClient로 변경하십시오.
   c) **연결**을 클릭하십시오.

   클라이언트 실행 기록은 Connected 이벤트를 기록합니다.
2. MQTT 클라이언트 유틸리티가 MQTTExampleTopic에 대한 발행물을 더 이상 수신하지 않는지 확인하십시오.

a) IBM MQ Explorer에서 Queue Managers*QmgrName*\Topics 폴더를 클릭하십시오.

b) MQTTExampleTopic을 마우스 오른쪽 단추로 클릭하고 **발행물 테스트...**를 클릭하십시오.

c) **메시지 데이터** 필드에 Hello World! 를 입력하고 > **메시지 발행** > MQTT 클라이언트 유틸리티 창으로 전환하십시오.

이 경우 **클라이언트 실행 기록**에 이벤트가 기록되지 않습니다.

3. 클라이언트에 대한 리모트 큐 정의를 작성하십시오.

   ClientIdentifier, MyClient를 리모트 큐 정의의 리모트 큐 관리자 이름으로 설정하십시오. 원하는 이름을 리모트 큐 이름으로 사용하십시오. 리모트 큐 이름이 MQTT 클라이언트에 토픽 이름으로 전달됩니다.

   a) Queue Managers*QmgrName*\Queues 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **리모트 큐 정의**를 클릭하십시오.

   b) 정의 이름을 MyClientRemoteQueue로 지정하고 **다음**을 클릭하십시오.

   c) **리모트 큐** 필드에 MQTTExampleQueue를 입력하십시오.

   d) **리모트 큐 관리자** 필드에 MyClient를 입력하십시오.

   e) **전송 큐** 필드에 SYSTEM.MQTT.TRANSMIT.QUEUE를 입력하고 **마침**을 클릭하십시오.

4. 테스트 메시지를 MyClientRemoteQueue에 넣으십시오.

   a) **MyClientRemoteQueue**를 마우스의 오른쪽 단추로 클릭하고 **테스트 메시지 넣기...**를 클릭하십시오.

   b) 메시지 데이터 필드에 Hello queue!를 입력하고 **메시지 넣기** > **닫기**를 클릭하십시오.

   클라이언트 실행 기록은 Received 이벤트를 기록합니다.

5. SYSTEM.MQTT.TRANSMIT.QUEUE를 기본 전송 큐로 제거하십시오.

   a) Queue Managers*QmgrName* folder를 마우스 오른쪽 단추로 클릭하고 **특성...**을 클릭하십시오.

   b) 네비게이터에서 **통신**을 클릭하십시오.

   c) **기본 전송 큐** 필드에서 SYSTEM.MQTT.TRANSMIT.QUEUE를 제거하고 **확인**을 클릭하십시오.

6. 112 페이지의 『4』 단계를 다시 실행하십시오.

   MyClientRemoteQueue는 전송 큐에 명시적으로 이름 지정된 리모트 큐 정의입니다. MyClient로 메시지를 보내기 위해 기본 전송 큐를 정의할 필요가 없습니다.

다음에 수행할 작업

기본 전송 큐가 더 이상 SYSTEM.MQTT.TRANSMIT.QUEUE로 설정되지 않으면 MQTT 클라이언트 유틸리티는 큐 관리자 알리어스가 ClientIdentifier, MyClient에 대해 정의되지 않는 한 새 구독을 작성할 수 없습니다. 기본 전송 큐를 SYSTEM.MQTT.TRANSMIT.QUEUE로 복원하십시오.

Windows Linux AIX 특정 MQTT v3 클라이언트로 메시지 발행

ClientIdentifier 를 토픽 이름으로 사용하고 IBM MQ 를 발행/구독 브로커로 사용하여 하나의 MQTT v3 클라이언트에서 다른 클라이언트로 메시지를 발행합니다.

시작하기 전에

107 페이지의 『IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행』 태스크를 수행하십시오. MQTT 클라이언트 유틸리티를 연결된 상태로 두십시오.

이 태스크 정보

이 태스크에서는 다음 두 가지 사항에 대해 설명합니다.

1. 한 MQTT 클라이언트의 토픽을 구독하고 다른 MQTT 클라이언트에서 발행을 수신합니다.
2. ClientIdentifier를 토픽 문자열로 사용하여 "포인트-투-포인트" 구독을 설정합니다.

프로시저

1. MQTT 클라이언트 유틸리티의 연결을 끊었다가 다시 연결하여 기존의 구독을 모두 제거하십시오.

   기본값을 변경하지 않으면 MQTT 클라이언트 유틸리티가 정리 세션과 연결되므로 구독이 제거됩니다. 정리 세션을 참조하십시오.

   태스크를 더 쉽게 수행하려면 MQTT 클라이언트 유틸리티에서 작성한 생성된 `ClientIdentifier` 를 사용하지 말고 사용자 자신의 `ClientIdentifier`를 입력하십시오.

   a) **연결 끊기**를 클릭하여 텔레메트리 채널에서 MQTT 클라이언트 유틸리티의 연결을 끊으십시오.

   클라이언트 실행 기록은 `Disconnected` 이벤트를 기록합니다.

   b) **클라이언트 ID**를 `MyClient`로 변경하십시오.

   c) **연결**을 클릭하십시오.

   클라이언트 실행 기록은 `Connected` 이벤트를 기록합니다.
2. 토픽, `MyClient`에 대한 구독을 작성하십시오.

   `MyClient`는 이 클라이언트의 `ClientIdentifier`입니다.

   a) `MyClient`를 **Subscription\Topic** 필드>에 입력하고 **구독**을 클릭하십시오.

   클라이언트 실행 기록은 `Subscribed` 이벤트를 기록합니다.
3. 다른 MQTT 클라이언트 유틸리티를 시작하십시오.

   a) `Queue Managers\`*`QmgrName`*`\Telemetry\channels` 폴더를 여십시오.

   b) **일반 텍스트** 채널을 마우스의 오른쪽 단추로 클릭하고 **MQTT 클라이언트 유틸리티 실행...**을 클릭하십시오.

   c) **연결**을 클릭하십시오.

   클라이언트 실행 기록은 `Connected` 이벤트를 기록합니다.
4. `Hello MyClient!`를 `MyClient` 토픽에 발행하십시오.

   a) `ClientIdentifier`, `MyClient`로 실행 중인 MQTT 클라이언트 유틸리티에서 구독 토픽 `MyClient`를 복사하십시오.

   b) `MyClient`를 각 MQTT 유틸리티 인스턴스의 **Publication\Topic** 필드에 붙여넣으십시오.

   c) **Publication\message** 필드에 `Hello MyClient!` 를 입력하십시오.

   d) 두 인스턴스 모두에서 **발행**을 클릭하십시오.

결과

`ClientIdentifier`가 `MyClient`인 MQTT 클라이언트 유틸리티의 **클라이언트 실행 기록**에 두 개의 **Received** 이벤트와 하나의 **Published** 이벤트가 기록됩니다. 다른 MQTT 클라이언트 유틸리티 인스턴스는 한 개의 **발행됨** 이벤트를 기록합니다.

수신됨 이벤트가 하나만 표시되는 경우 다음이 원인일 수 있습니다.

1. 큐 관리자에 대한 기본 전송 큐가 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 설정되었습니까?
2. 다른 실습에서 `MyClient`를 참조하는 리모트 큐 정의나 큐 관리자 알리어스를 작성했습니까? 구성 문제점이 있는 경우 큐 관리자 알리어스 또는 전송 큐와 같은 `MyClient`를 참조하는 자원을 모두 삭제하십시오. 클라이언트 유틸리티의 연결을 끊고 텔레메트리(MQXR) 서비스를 중지한 후에 다시 시작하십시오.

Windows Linux AIX MQTT 클라이언트에서 IBM MQ 애플리케이션으로 메시지 전송

IBM MQ 애플리케이션은 토픽을 구독함으로써 MQTT v3 클라이언트에서 메시지를 수신할 수 있습니다. MQTT 클라이언트는 텔레메트리 채널을 사용하여 IBM MQ에 연결하며 같은 토픽에 발행함으로써 IBM MQ 애플리케이션에 메시지를 송신합니다.

114 페이지의 『' MQTT 클라이언트에서 ' IBM MQ ' 클라이언트로 메시지 게시하기』 태스크를 수행하여 MQTT 클라이언트에서 IBM MQ에 정의된 구독으로 발행을 송신하는 방법을 배우십시오.

토픽이 클러스터되어 있거나 발행/구독 계층을 사용하여 분배되어 있을 경우 구독은 다른 큐 관리자에서 MQTT 클라이언트가 연결되어 있는 큐 관리자로 될 수 있습니다.

Windows Linux AIX ' MQTT 클라이언트에서 ' IBM MQ ' 클라이언트로 메시지 게시하기

IBM MQ Explorer를 사용하여 토픽의 구독을 작성하고 MQTT 클라이언트 유틸리티를 사용하여 토픽에 발행하십시오.

시작하기 전에

107 페이지의 『IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행』 태스크를 수행하십시오. MQTT 클라이언트 유틸리티를 연결된 상태로 두십시오.

이 태스크 정보

이 태스크는 MQTT 클라이언트로 메시지를 발행하는 것과 IBM MQ Explorer로 작성된 관리되지 않은 지속 가능 구독을 사용하여 발행을 수신하는 것을 보여줍니다.

프로시저

1. 토픽 문자열 `MQTT Example`에 대한 지속 가능 구독을 작성하십시오.

   IBM MQ Explorer를 사용하여 큐와 구독을 작성하려면 다음 단계를 수행하십시오.

   a) IBM MQ Explorer에서 `Queue Managers\`*`QmgrName`*`\Queues` 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **로컬 큐...**를 클릭하십시오.

   b) `MQTTExampleQueue`를 큐 이름으로 입력하고 **마침**을 클릭하십시오.

   c) IBM MQ Explorer에서 `Queue Managers\`*`QmgrName`*`\Subscriptions` 폴더를 마우스 오른쪽 단추로 클릭하고 **새로 작성** > **구독...**을 클릭하십시오.

   d) `MQTTExampleSubscription`을 큐 이름으로 입력하고 **다음**을 클릭하십시오.

   e) **선택...** > `MQTTExampleTopic` > **확인**을 클릭하십시오.

   107 페이지의 『IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행』의 108 페이지의 『4』 단계에서 `MQTTExampleTopic` 토픽을 이미 작성했습니다.

   f) `MQTTExampleQueue`를 대상 이름으로 입력하고 **마침**을 클릭하십시오.

2. 선택적 단계로, `mqm` 권한이 없는 다른 사용자가 사용하도록 큐를 설정하십시오.

   `mqm`보다 낮은 권한을 가진 사용자에 대한 구성을 설정할 경우 `MQTTExampleQueue`에 `put` 및 `get` 권한을 부여해야 합니다. 토픽과 전송 큐에 대한 액세스는 107 페이지의 『IBM MQ Explorer 에서 MQTT 클라이언트 유틸리티에 메시지 발행』에서 구성되었습니다.

   a) 사용자에게 `MQTTExampleQueue` 큐에 대한 넣기 및 가져오기 권한을 부여하십시오.

   ```
   setmqaut -m qMgrName -t queue -n MQTTExampleQueue -p User ID -all +put +get
   ```

3. ' MQTT 클라이언트 유틸리티를 사용하여 ' `Hello IBM MQ!` '을 ' `MQTT Example` ' 주제에 게시합니다.

   MQTT 클라이언트 유틸리티가 연결되어 있지 않은 경우 **일반 텍스트** 채널을 마우스의 오른쪽 단추로 클릭하고 **MQTT 클라이언트 유틸리티 실행...** > **연결**을 클릭하십시오.

   a) **Publication\Topic** 필드에 `MQTT Example`을 입력하십시오.

   b) **게시\메시지** 필드에 ' `Hello IBM MQ!` '을 입력하고 **게시합니다**.

4. `Queue Managers\`*`QmgrName`*`\Queues` 폴더를 열고 `MQTTExampleQueue`를 찾으십시오.

   현재 큐 용량 필드는 1입니다.

5. Right-click `MQTTExampleQueue` > **메시지 찾아보기...** 를 클릭하고 게시물을 검토합니다.

Windows Linux AIX MQTT 발행/구독 애플리케이션

MQTT 애플리케이션 쓰기에 토픽 기반 발행/구독을 사용하십시오.

MQTT 클라이언트가 연결되면 발행은 클라이언트와 서버 사이에서 오고 가게 됩니다. 발행은 클라이언트에서 정보가 발행되면 클라이언트에서 전송됩니다. 메시지가 클라이언트에서 작성한 구독과 일치하는 토픽에 발행되었을 경우에는 클라이언트에서 발행을 수신합니다.

IBM MQ 발행/구독 브로커는 MQTT 클라이언트에서 작성한 토픽과 구독을 관리합니다. MQTT 클라이언트에서 작성된 토픽은 IBM MQ 애플리케이션에서 작성된 토픽과 동일한 토픽 공간을 공유합니다.

MQTT 클라이언트 구독의 토픽 문자열과 일치하는 발행물은 리모트 큐 관리자 이름이 클라이언트의 `ClientIdentifier` 로 설정된 `SYSTEM.MQTT.TRANSMIT.QUEUE` 에 배치됩니다. 텔레메트리(MQXR) 서비스는 구독을 작성한 클라이언트에 발행물을 전달합니다. 클라이언트를 식별하기 위해 리모트 큐 관리자 이름으로 설정된 `ClientIdentifier`를 사용합니다.

일반적으로 `SYSTEM.MQTT.TRANSMIT.QUEUE`는 기본 전송 큐로 정의되어야 합니다. 기본 전송 큐를 사용하지 않도록 MQTT를 구성하는 것은 가능하지만 번거로운 작업입니다. 메시지를 MQTT 클라이언트에 송신하도록 분산 큐잉 구성을 참조하십시오.

MQTT 클라이언트는 지속 세션을 작성할 수 있습니다. 118 페이지의 『MQTT 상태 비저장 및 상태 저장 세션』의 내용을 참조하십시오. 지속 세션에 작성된 구독은 지속 가능합니다. 지속 세션과 함께 클라이언트에 도착하는 발행은 `SYSTEM.MQTT.TRANSMIT.QUEUE`에 저장되며, 다시 연결될 때 클라이언트로 전달됩니다.

MQTT 클라이언트는 보유된 발행물에 발행하고 구독할 수도 있습니다. 보유된 발행물 및 MQTT 클라이언트를 참조하십시오. 보유된 발행물 토픽에 대한 구독자는 토픽의 최근 발행을 수신합니다. 구독자는 보유된 발행물이 구독을 작성하거나 이전 세션에 다시 연결할 때 보유된 발행물을 수신합니다.

Windows Linux AIX 텔레메트리 애플리케이션

IBM MQ 또는 IBM Integration Bus 메시지 플로우를 사용하여 텔레메트리 애플리케이션을 씁니다.

JMS, MQI 또는 기타 IBM MQ 프로그래밍 인터페이스를 사용하여 IBM MQ에서 텔레메트리 애플리케이션을 프로그래밍하십시오.

텔레메트리(MQXR) 서비스가 MQTT v3 메시지와 IBM MQ 메시지를 상호 변환합니다. MQTT 클라이언트를 대신해 구독과 발행을 작성하며 발행을 MQTT 클라이언트에 전달합니다. 발행은 MQTT v3 메시지의 페이로드입니다. 페이로드는 메시지 헤더와 `jms-bytes` 형식의 바이트 배열을 포함합니다. 텔레메트리 서버는 MQTT v3 메시지와 IBM MQ 메시지 간의 헤더를 상호 맵핑합니다. 115 페이지의 『큐 관리자와 MQ Telemetry의 통합』의 내용을 참조하십시오.

Publication, MQInput및 JMSInput 노드를 사용하여 IBM Integration Bus 및 MQTT 클라이언트 간에 publication을 송신 및 수신하십시오.

메시지 플로우를 사용하여 텔레메트리와 HTTP를 사용하는 웹 사이트를, 텔레메트리와 IBM MQ 및 WebSphere Adapters를 사용하는 다른 애플리케이션을 통합할 수 있습니다.

Windows Linux AIX 큐 관리자와 MQ Telemetry의 통합

MQTT 클라이언트는 발행/구독 애플리케이션으로 IBM MQ와 통합되어 있습니다. IBM MQ의 토픽을 발행 또는 구독하고 새 토픽을 작성하며 기존 토픽을 사용할 수 있습니다. 자체를 포함한 MQTT 클라이언트 또는 해당 구독의 토픽에 발행하는 다른 IBM MQ 애플리케이션의 결과로 IBM MQ 에서 발행물을 수신합니다. 발행의 속성을 결정하는 규칙이 적용됩니다.

IBM MQ에서 제공되는 토픽, 발행, 구독 및 메시지와 연관된 많은 속성은 지원되지 않습니다. 116 페이지의 『MQTT 클라이언트에서 IBM MQ 발행/구독 브로커로』와 117 페이지의 『IBM MQ에서 MQTT 클라이언트로 메시지 전송』에서 발행의 속성을 설정하는 방법에 대해 설명합니다. 설정값은 발행물이 IBM MQ 발행/구독 브로커로 송신되는지 또는 이 브로커로부터 수신되는지에 따라 다릅니다.

IBM MQ에서 발행/구독 토픽은 관리 토픽 오브젝트와 연관됩니다. MQTT 클라이언트에서 작성되는 토픽도 이와 마찬가지입니다. MQTT 클라이언트가 발행의 토픽 문자열을 작성하면 IBM MQ 발행/구독 브로커는 이를 관리 토픽 오브젝트와 연관시킵니다. 브로커는 발행의 토픽 문자열을 제일 가까운 관리 토픽 오브젝트 상위에 맵핑합

니다. 맵핑은 IBM MQ 애플리케이션의 경우와 동일합니다. 사용자가 작성한 토픽이 없을 경우 발행 토픽은 SYSTEM.BASE.TOPIC에 맵핑됩니다. 발행에 적용되는 토픽은 토픽 오브젝트에서 도출된 것입니다.

IBM MQ 애플리케이션 또는 관리자가 구독을 작성하는 경우 구독에 이름이 지정됩니다. IBM MQ Explorer 를 사용하거나 또는 PCF 명령을 사용하여 구독을 나열합니다. **runmqsc** 모든 MQTT 클라이언트 구독이 이름 지정됩니다. 양식의 이름이 지정됩니다: *ClientIdentifier*:*Topic name*

MQTT 클라이언트에서 IBM MQ 발행/구독 브로커로

MQTT 클라이언트는 발행물을 IBM MQ로 송신했습니다. 텔레메트리(MQXR) 서비스는 IBM MQ 메시지로 발행물을 변환합니다. IBM MQ 메시지에는 다음 세 가지 부분이 포함됩니다.

1. MQMD
2. RFH2
3. 메시지

MQMD 특성은 116 페이지의 표 9에 기록되어 있지 않은 이상 각각 기본값으로 설정됩니다.

표 *9. MQMD* 특성의 설정

MQMD 필드	**유형**	**값**
Format	MQCHAR8	MQFMT_RF_HEADER_2
UserIdentifier	MQCHAR12	다음 중 하나로 설정하십시오. MqttClient.ClientIdentifier MqttConnectOptions.UserName 텔레메트리 채널용으로 IBM MQ 관리자가 설정한 사용자 ID
Priority	MQLONG	MQPRI_PRIORITY_AS_Q_DEF(IBM MQ의 경우 기본값이며, 기본값이 4인 JMS와 다릅니다.)
Persistence	MQLONG	QoS=0→MQPER_NOT_PERSISTENT QoS=1→MQPER_PERSISTENT QoS=2→MQPER_PERSISTENT

RFH2 헤더는 JMS 메시지의 유형을 정의하는 <msd> 폴더를 포함하지 않습니다. 텔레메트리(MQXR) 서비스는 IBM MQ 메시지를 기본 JMS 메시지로 작성합니다. 기본 JMS 메시지-유형은 jms-bytes 메시지입니다. 애플리케이션은 메시지 특성으로서의 추가 헤더 정보에 액세스할 수 있습니다. 메시지 특성을 참조하십시오.

RFH2 값은 116 페이지의 표 10에 표시된 것과 같이 설정됩니다. 형식 특성은 RFH2 고정된 헤더에 설정되며 다른 값은 RFH2 폴더 안에 설정됩니다.

표 *10. RFH2* 특성의 설정

RFH2 특성	**유형/폴더**	**헤더**
Format	MQCHAR8	MQFMT_NONE
ClientIdentifier	mqtt/clientId	MqttClient.ClientIdentifier를 1 - 23바이트 길이로 복사하십시오.
QoS	mqtt/qos	수신 MQTT 메시지에서 QoS 를 복사하십시오.
메시지 ID	mqtt/msgid	QoS 가 1 또는 2인 경우 수신 MQTT 메시지에서 메시지 ID 를 복사하십시오.

표 *10. RFH2* 특성의 설정 *(계속)*		
RFH2 특성	**유형/폴더**	**헤더**
MQIsRetained	mqps/Ret	기존 MQTT 발행이 RETAIN 특성이 설정되어 전송되었으며 메시지가 보유된 발행물로 수신된 경우 설정하십시오.
MQTopicString	mqps/Top	MQTT 메시지 발행 대상이 된 토픽입니다.

MQTT 발행의 페이로드는 IBM MQ 메시지의 컨텐츠에 맵핑됩니다.

표 *11. MQTT* 발행의 페이로드가 *IBM MQ* 메시지 컨텐츠에 맵핑되는 방법		
메시지 컨텐츠	**유형**	**IBM MQ 메시지의 컨텐츠**
Buffer	MQBYTE *n*	수신되는 MQTT 메시지에서 바이트를 복사하십시오. 길이는 0이 될 수도 있습니다.

IBM MQ에서 MQTT 클라이언트로 메시지 전송

클라이언트가 발행 토픽을 구독했습니다. IBM MQ 애플리케이션이 토픽에 발행되어 IBM MQ 발행/구독 브로커가 MQTT 구독자에게 발행을 송신합니다. 또는 IBM MQ 애플리케이션이 요청되지 않은 메시지를 MQTT 클라이언트로 직접 전송했습니다. 117 페이지의 표 12에서는 고정된 메시지 헤더를 MQTT 클라이언트로 전송된 메시지에 설정하는 방법에 대해 설명합니다. IBM MQ 메시지 헤더의 다른 모든 데이터 또는 다른 모든 헤더는 제거됩니다. IBM MQ 메시지의 메시지 데이터는 변경 없이 MQTT 메시지 안의 메시지 페이로드로 전송됩니다. MQTT 메시지는 텔레메트리(MQXR) 서비스에 의해 MQTT 클라이언트로 전송됩니다.

표 *12. MQTT* 클라이언트에 전송된 *IBM MQ* 메시지에서 고정 메시지 헤더를 설정하는 방법		
MQTT 필드	**유형**	**값**
DUP	부울	QoS = 1 또는 2이며 이전 전송에서 메시지가 이 클라이언트로 전송되었고 이 메시지가 시간이 지난 뒤에도 수신확인되지 않았을 경우 설정하십시오.
QoS	int	IBM MQ 의 발행/구독 브로커에서 발신되는 발행물의 QoS 값이 설정되는 방법은 수신되는 발행물에 따라 다릅니다. QoS는 수신되는 발행물을 MQTT 클라이언트에서 송신했는지 또는 IBM MQ 애플리케이션에서 송신했는지에 따라 다릅니다. **MQTT** 수신되는 발행과 구독자에 의해 요청된 QoS의 QoS 값을 낮추십시오. **IBM MQ** 수신되는 발행에서 도출되는 QoS의 값을 낮추십시오. MQPER_NOT_PERSISTENT→QoS=0 MQPER_PERSISTENT→QoS=2 구독자에 의해 요청된 QoS의 QoS 값도 낮추십시오. 구독 없이 메시지가 클라이언트에 전송된 경우 QoS는 기본값인 2로 설정됩니다. 클라이언트는 다른 QoS로 DEFAULT.QoS를 구독하여 이 값을 변경할 수 있습니다.
RETAIN	부울	수신되는 발행에 보유된 특성이 설정되어 있을 경우 설정하십시오.

118 페이지의 표 13에서는 MQTT 클라이언트로 전송된 MQTT 메시지에 변수 메시지 헤더가 설정되는 방법에 대해 설명합니다.

표 13. MQTT 클라이언트로 전송된 MQTT 메시지에 MQTT 변수 헤더 특성을 설정하는 방법		
MQTT 필드	**유형**	**값**
Topic name	문자열	메시지와 같이 발행된 토픽 문자열입니다.
Message ID	문자열	발행을 SYSTEM.MQTT.TRANSMIT.QUEUE에 넣었을 때 그 발행에 있는 MQMD.MsgId 특성의 마지막 2바이트입니다.
Payload	byte[]	수신되는 발행의 바이트 사본을 발행/구독 브로커에 전달하십시오. 길이는 0이 될 수도 있습니다.

Windows Linux AIX MQTT 상태 비저장 및 상태 저장 세션

MQTT는 큐 관리자를 사용하여 상태 저장 세션을 작성할 수 있습니다. 상태 저장 MQTT 클라이언트의 연결이 끊어지면 큐 관리자가 클라이언트에 의해 작성된 구독 및 인플라이트 메시지를 유지합니다. 클라이언트가 다시 연결되면 인플라이트 메시지를 해석합니다. 전달을 위해 큐에 대기된 모든 메시지를 보내고 연결이 끊어진 동안 구독에 대해 발행한 모든 메시지를 수신합니다.

MQTT 클라이언트가 텔레메트리 채널에 연결되면 새 세션을 시작하거나 기존 세션을 재개합니다. 새 세션에는 수신확인되지 않은 미해결 메시지, 구독 및 전달되기를 기다리는 발행이 포함되어 있지 않습니다. 클라이언트가 연결되면 정리 세션으로 시작하는지 또는 기존 세션을 재개하는지 여부를 지정합니다. 정리 세션을 참조하십시오.

클라이언트가 기존 세션을 재개하면 연결이 끊어지지 않았던 것처럼 계속 진행됩니다. 전달을 기다리는 발행은 클라이언트로 전송되고 커미트되지 않은 메시지 전송이 완료됩니다. 지속 세션의 클라이언트와 텔레메트리(MQXR) 서비스와의 연결이 끊어질 때 클라이언트가 작성한 모든 구독은 그대로 남아 있습니다. 클라이언트가 다시 연결되면 해당 클라이언트에게 구독에 대한 발행이 전송됩니다. 이전 세션을 재개하지 않고 다시 연결하면 텔레메트리(MQXR) 서비스가 발행물을 제거합니다.

큐 관리자가 SYSTEM.MQTT.PERSISTENT.STATE 큐에 세션 상태 정보를 저장합니다.

IBM MQ 관리자는 세션의 연결을 끊고 세션을 제거할 수 있습니다.

Windows Linux AIX MQTT 클라이언트가 연결되지 않았을 때

클라이언트가 연결되지 않았을 때 큐 관리자는 이를 대신해 발행을 계속 수신할 수 있습니다. 발행은 클라이언트가 다시 연결할 때 여기로 전달됩니다. 클라이언트는 예상치 못하게 연결이 끊어질 경우 큐 관리자가 자신을 대신해 발행하는 "이상 종료 시 메시지"를 작성할 수 있습니다.

클라이언트가 예상치 못하게 연결이 끊겼을 때 알림을 받고 싶으면 이상 종료 시 메시지 발행을 등록할 수 있습니다. 이상 종료 시 메시지 발행을 참조하십시오. 클라이언트에 대한 연결이 클라이언트의 요청 없이 중단되었음을 감지한 경우, 이는 텔레메트리(MQXR) 서비스에 의해 전송됩니다.

클라이언트는 언제든지 보유된 발행물을 발행할 수 있습니다. 보유된 발행물 및 MQTT 클라이언트를 참조하십시오. 토픽에 대한 새 구독은 토픽과 연관된 어떤 보유된 발행물도 송신해 달라고 요청할 수 있습니다. 이상 종료 시 메시지를 보유된 발행물로서 작성할 경우 이를 클라이언트 상태를 모니터하는 데 사용할 수 있습니다.

예를 들면, 한 클라이언트는 연결할 때 자신의 가용성을 알리며 보유된 발행물을 발행합니다. 동시에, 비가용성을 알리는 보유된 이상 종료 시 메시지 발행을 작성합니다. 또한 계획된 연결 종료를 하기 전에 보유된 발행물로서 자신의 비가용성을 발행합니다. 클라이언트가 사용 가능한지 알아보려면 이 보유된 발행의 토픽을 구독합니다. 사용자는 항상 세 가지 발행 중 하나를 수신합니다.

클라이언트가 연결이 끊어져 있을 때 발행된 메시지를 수신하고자 하는 경우 클라이언트를 이전 세션에 다시 연결하십시오. 118 페이지의 『MQTT 상태 비저장 및 상태 저장 세션』의 내용을 참조하십시오. 그 구독은 삭제될 때까지 또는 클라이언트가 정리 세션을 작성할 때까지 활성 상태입니다.

Windows Linux AIX MQTT 클라이언트와 IBM MQ 애플리케이션 사이의 느슨한 결합

MQTT 클라이언트와 IBM MQ 애플리케이션 사이의 발행 플로우는 느슨히 결합되어 있습니다. 발행은 MQTT 클라이언트 또는 IBM MQ 애플리케이션에서 설정된 순서 없이 생성될 수 있습니다. 발행자와 구독자는 느슨히 결합되어 있습니다. 이들은 발행과 구독을 통해 서로 상호작용합니다. 또한 IBM MQ 애플리케이션에서 MQTT 클라이언트로 직접 메시지를 보낼 수도 있습니다.

MQTT 클라이언트와 IBM MQ 애플리케이션은 다음과 같은 두 가지 측면에서 느슨히 결합되어 있습니다.

1. 발행자와 구독자는 발행, 구독과 토픽의 연관을 통해 느슨히 결합되어 있습니다. 발행자와 구독자는 보통 다른 발행 또는 구독 소스의 주소나 ID를 알고 있습니다.
2. MQTT 클라이언트는 발행을 발행, 구독, 수신하며 분리된 스레드에서 전달 수신확인을 처리합니다.

MQTT 클라이언트 애플리케이션은 발행이 전달될 때까지 대기하지 않습니다. 애플리케이션은 MQTT 클라이언트로 메시지를 전달하고 그 후 애플리케이션은 자신의 스레드에서 계속됩니다. 발행물의 전달과 애플리케이션을 동기화하기 위해 전달 토큰이 사용됩니다. 전달 토큰을 참조하십시오.

MQTT 클라이언트에 메시지를 전달한 후 애플리케이션은 전달 토큰을 기다릴지 선택할 수 있습니다. 기다리는 대신 클라이언트는 IBM MQ에 발행이 전달되었을 때 호출되는 콜백 메소드를 제공할 수 있습니다. 이는 또한 전달 토큰을 무시할 수 있습니다.

메시지의 서비스 품질에 따라 전달 토큰은 콜백 메소드로 즉시 리턴되거나 상당한 시간이 지난 뒤 리턴될 수 있습니다. 전달 토큰은 클라이언트 연결이 끊어진 후 다시 연결되었을 때 리턴될 수도 있습니다. 서비스 품질이 전송 후 삭제인 경우 전달 토큰은 즉시 리턴됩니다. 다른 두 경우 전달 토큰은 클라이언트가 발행이 구독자로 송신되었다는 수신확인을 수신했을 때만 리턴됩니다.

클라이언트 구독의 결과로 MQTT 클라이언트로 송신된 발행은 `messageArrived` 콜백 메소드로 전달됩니다. `messageArrived`는 기본 애플리케이션과 다른 스레드에서 실행됩니다.

MQTT 클라이언트로 직접 메시지 송신

둘 중 한 가지 방법으로 특정 MQTT 클라이언트에 메시지를 송신할 수 있습니다.

1. IBM MQ 애플리케이션은 구독 없이 MQTT 클라이언트에 직접 메시지를 보낼 수 있습니다. 클라이언트에 직접 메시지 보내기를 참조하십시오.
2. 또 다른 방법은 `ClientIdentifier` 이름 지정 규칙을 사용하는 것입니다. 모든 MQTT 구독자가 자신의 고유한 `ClientIdentifier`를 토픽으로 사용하여 구독을 작성하도록 하십시오. *ClientIdentifier*에 발행하십시오. 발행은 토픽 *ClientIdentifier*를 구독한 클라이언트에게 송신됩니다. 이 기술을 사용하면 발행을 특정 MQTT 구독자에게 송신할 수 있습니다.

Windows Linux AIX MQ Telemetry 보안

텔레메트리 디바이스는 대부분 휴대용이며 잘 제어되지 않은 환경에서 사용하게 될 경우가 많으므로 디바이스를 보호하는 것은 중요합니다. VPN을 사용하여 MQTT 디바이스에서 텔레메트리(MQXR) 서비스로의 연결을 보호할 수 있습니다. MQ Telemetry는 TLS와 JAAS라는 다른 두 가지 보안 메커니즘을 제공합니다.

TLS는 기본적으로 디바이스 및 텔레메트리 채널 간의 통신을 암호화하며 디바이스가 올바른 서버에 연결 중인지 인증하기 위해 사용됩니다. TLS를 사용하여 텔레메트리 채널 인증을 참조하십시오. TLS를 사용하여 클라이언트 디바이스가 서버에 연결할 수 있는지 확인할 수도 있습니다. TLS를 사용하여 MQTT 클라이언트 인증을 참조하십시오.

JAAS는 기본적으로 디바이스의 사용자가 서버 애플리케이션을 사용할 수 있는지 확인하는 데 사용됩니다. 비밀번호를 사용하여 MQTT 클라이언트 인증을 참조하십시오. JAAS는 싱글 사인온 디렉토리를 사용하는 비밀번호를 확인하기 위해 LDAP와 함께 사용할 수 있습니다.

TLS 및 JAAS는 두 인수 인증을 제공하기 위해 결합해 사용할 수 있습니다. TLS에서 사용하는 암호를 FIPS 표준을 만족시키는 암호로 제한할 수 있습니다.

최소 수만 명의 사용자가 있을 경우 개별 보안 프로파일을 제공하는 것은 실용적이지 않을 수 있습니다. IBM MQ 오브젝트에 액세스하려는 개별 사용자를 인증하는 데 프로파일을 사용하는 것 또한 마찬가지입니다. 대신 토픽에 대한 발행 및 구독 인증과, 클라이언트로의 발행물 송신을 위해 사용자를 클래스로 그룹 지으십시오.

클라이언트를 공용 클라이언트 사용자 ID로 맵핑하도록 각 텔레메트리 채널을 구성하십시오. 특정 채널에서 연결하는 모든 클라이언트에 대해 공용 사용자 ID를 사용하십시오. MQTT 클라이언트 ID 및 권한을 참조하십시오.

사용자 그룹에 권한을 부여하는 것이 각 개인 인증에 문제를 일으키지는 않습니다. 각 개별 사용자는 클라이언트와 서버에서 자신의 Username과 Password로 인증을 받은 후 공용 사용자 ID를 사용하는 서버에서 권한 부여를 받을 수 있습니다.

Windows Linux AIX MQ Telemetry 다국어 지원

MQTT v3 프로토콜의 메시지 페이로드는 바이트 배열로 인코드됩니다. 일반적으로 텍스트를 처리하는 애플리케이션은 UTF-8로 메시지 페이로드를 작성합니다. 텔레메트리 채널은 메시지 페이로드를 UTF-8로 설명하지만 어떤 코드 페이지 변환도 수행하지 않습니다. 발행 토픽 문자열은 UTF-8이어야 합니다.

영문자 데이터를 올바른 코드 페이지로, 숫자 데이터를 올바른 숫자 인코딩으로 변환하는 것은 애플리케이션의 역할입니다.

MQTT Java 클라이언트에는 간편한 MqttMessage.toString 메소드가 있습니다. 이 메소드는 메시지 페이로드를 로컬 플랫폼의 기본 문자 세트(일반적으로 UTF-8)로 인코드된 것으로 취급합니다. 이는 페이로드를 Java String으로 변환합니다. Java에는 로컬 플랫폼의 기본 문자 세트를 사용하여 문자열을 인코드된 바이트 배열로 변환하는 String 메소드인 getBytes가 있습니다. 같은 기본 문자 세트를 가진 플랫폼 사이에서 메시지 페이로드로 텍스트를 교환하는 두 MQTT Java 프로그램은 UTF-8로 이를 쉽고 효율적으로 수행합니다.

한 플랫폼의 기본 문자 세트가 UTF-8이 아닐 경우에 애플리케이션은 메시지를 교환하는 데 변환을 설정해야 합니다. 예를 들면, 발행자가 getBytes("UTF8") 메소드를 사용하여 문자열에서 UTF-8로의 변환을 지정하는 경우가 있습니다. 메시지의 텍스트를 수신하기 위해 구독자는 메시지가 UTF-8 문자 세트로 인코드되어 있다고 추정합니다.

텔레메트리(MQXR) 서비스는 MQTT 클라이언트 메시지로부터 수신되는 모든 발행물의 인코딩이 UTF-8임을 설명합니다. MQMD.CodedCharSetId 를 UTF-8로, RFH2.CodedCharSetId 를 MQCCSI_INHERIT 로 115 페이지의 『큐 관리자와 MQ Telemetry의 통합』 참조. 발행의 형식은 MQFMT_NONE으로 설정되어 채널이나 MQGET에 의해 변환이 수행되지 않도록 합니다.

Windows Linux AIX MQ Telemetry의 성능 및 확장성

다수의 클라이언트를 관리하고 MQ Telemetry의 확장성을 향상시킬 때 다음과 같은 요인을 고려하십시오.

용량 계획

MQ Telemetry의 성능 보고서에 대한 정보를 보려면 MQ 성능 문서를 참조하십시오.

연결

연결과 관련된 비용은 다음을 포함하고 있습니다.

- 프로세서 사용 및 시간의 관점에서 본 연결 자체를 설정하는 비용.
- 네트워크 비용.
- 연결을 열어놓고 사용하지 않는 동안 사용되는 메모리.

클라이언트가 연결된 채일 때 발생하는 추가적인 부하가 있습니다. 연결이 열려 있는 경우 TCP/IP에는 플로우가 계속 발생하며 MQTT 메시지는 연결이 계속 있는지 확인하기 위해 네트워크를 사용합니다. 또한 서버에서 열려 있는 채인 각 클라이언트 연결에 대해 메모리가 사용됩니다.

1분에 하나 이상 메시지를 송신할 경우 새 연결을 시작하는 비용을 피하기 위해 연결을 계속 열어두십시오. 10 - 15분 동안 하나 이하의 메시지를 송신할 경우 연결을 열어두는 비용을 피하기 위해 연결을 끊는 것을 고려하십시오. TLS 연결은 설정하는 데 비용이 많이 소요되기 때문에 열어둔 채로 두는 편이 좋을 수 있습니다.

추가적으로 클라이언트의 기능을 고려하십시오. 클라이언트에 저장 후 전달 기능이 있는 경우 메시지를 배치해 두고 배치를 송신하는 사이에 연결을 끊을 수 있습니다. 그러나 클라이언트의 연결이 끊긴 경우에는 클라이언트가 서버에서 메시지를 수신할 수 없습니다. 따라서 애플리케이션의 목적이 의사결정에 영향을 미칩니다.

파일 전송과 같이 시스템에 많은 메시지를 송신하는 클라이언트가 있을 경우 메시지마다 서버 응답을 기다리지 마십시오. 대신 모든 메시지를 송신하고 마지막에 메시지가 모두 수신되었는지 확인하십시오. 또는 QoS(Quality of Service)를 사용하십시오.

중요하지 않은 메시지는 QoS 0을 사용하여 전달하고 중요한 메시지는 QoS 2를 사용하여 전달하는 것처럼 메시지마다 QoS를 달리 할 수 있습니다. 메시지 처리량은 QoS가 2일 때보다 QoS가 0일 때 약 두 배 더 높을 수 있습니다.

이름 지정 규칙

다수의 클라이언트를 대상으로 애플리케이션을 설계할 경우 효율적인 이름 지정 규칙을 구현하십시오. 각 클라이언트를 올바르게 `ClientIdentifier`에 맵핑하려면 `ClientIdentifier`를 의미있게 작성하십시오. 좋은 이름 지정 규칙은 어느 클라이언트가 실행 중인지 관리자가 쉽게 알 수 있게 해 줍니다. 이름 지정 규칙은 관리자가 IBM MQ 탐색기에서 긴 클라이언트 목록을 필터링하고 문제점을 판별하는 데 도움이 됩니다. 클라이언트 ID를 참조하십시오.

처리량

토픽 이름의 길이는 네트워크를 흘러가는 바이트 수에 영향을 줍니다. 발행하거나 구독할 때 메시지의 바이트 수는 중요할 수 있습니다. 그러므로 토픽 이름의 문자 개수를 제한하십시오. MQTT 클라이언트가 토픽을 구독할 때 IBM MQ는 여기에 다음과 같은 형식의 이름을 부여합니다.

```
ClientIdentifier: TopicName
```

MQTT 클라이언트의 모든 구독을 보려면 IBM MQ MQSC **DISPLAY** 명령을 사용할 수 있습니다.

```
DISPLAY SUB(' ClientID1:*')
```

IBM MQ에 MQTT 클라이언트에서 사용하기 위한 자원 정의

MQTT 클라이언트가 IBM MQ에 리모트 큐 관리자를 연결합니다. IBM MQ 애플리케이션이 MQTT 클라이언트에 메시지를 송신하는 기본 방법이 두 가지 있습니다. 기본 전송 큐를 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 설정하거나 큐 관리자 알리어스를 사용하는 것입니다. 다수의 MQTT 클라이언트가 있을 경우 큐 관리자의 기본 전송 큐를 정의하십시오. 기본 전송 큐 설정을 사용하면 관리 작업이 간소화됩니다. MQTT 클라이언트에 메시지를 송신하도록 분산 큐잉 구성을 참조하십시오.

구독하지 않음으로서 오는 확장성 향상

MQTT V3 클라이언트가 토픽을 구독하는 경우, 구독은 IBM MQ의 텔레메트리(MQXR) 서비스에 의해 작성됩니다. 구독은 클라이언트가 구독한 발행을 `SYSTEM.MQTT.TRANSMIT.QUEUE`로 라우팅합니다. 각 발행물의 전송 헤더에 있는 리모트 큐 관리자 이름은 구독을 작성한 MQTT 클라이언트의 `ClientIdentifier` 로 설정됩니다. 각각 자신의 구독을 작성하고 있는 다수의 클라이언트가 있을 경우 IBM MQ 발행/구독 클러스터 또는 계층 전반에 많은 프록시 구독이 유지되는 결과가 발생합니다. 발행/구독을 사용하는 대신 포인트-투-포인트 기반 솔루션 사용에 대한 정보는 클라이언트에 직접 메시지 보내기를 참조하십시오.

다수의 클라이언트 관리

다수의 동시에 연결된 클라이언트를 지원하려면 JVM 매개변수 **-Xms** 및 **-Xmx**를 설정하여 텔레메트리(MQXR) 서비스에 사용할 수 있는 메모리를 늘리십시오. 다음 단계를 수행하십시오.

1. 텔레메트리 서비스 구성 디렉토리에서 `java.properties` 파일을 찾으십시오. Windows의 텔레메트리(MQXR) 서비스 구성 디렉토리 또는 Linux의 텔레메트리 서비스 구성 디렉토리를 참조하십시오.
2. 파일의 지시사항을 따르십시오. 50,000개의 동시 연결된 클라이언트를 유지하는 데는 1GB 힙 정도면 충분합니다.

```
# Heap sizing options - uncomment the following lines to set the heap to 1G
#-Xmx1024m
#-Xms1024m
```

3. `java.properties` 파일에 텔레메트리(MQXR) 서비스를 실행 중인 JVM에 전달할 다른 명령행 인수를 추가하십시오. 텔레메트리(MQXR) 서비스에 JVM 매개변수 전달을 참조하십시오.

Linux에서 열린 파일 디스크립터의 수를 늘리려면 `/etc/security/limits.conf/`에 다음 행을 추가하고 다시 로그인하십시오.

```
@mqm soft nofile 65000
@mqm hard nofile 65000
```

각 소켓은 하나의 파일 디스크립터를 필요로 합니다. 텔레메트리 서비스는 몇몇 추가 파일 디스크립터를 필요로 하며 따라서 이 숫자는 요구되는 열린 소켓의 개수보다 커야 합니다.

큐 관리자는 각 지속 불가능 구독에 오브젝트 핸들을 사용합니다. 많은 활성 지속 불가능 구독을 지원하려면 큐 관리자에서 활성 핸들의 최대 개수를 증가시키십시오. 예:

```
echo ALTER QMGR MAXHANDS(999999999) | runmqsc qMgrName
```

그림 *48. Windows*에서 최대 핸들 수 대체

```
echo "ALTER QMGR MAXHANDS(999999999)" | runmqsc qMgrName
```

그림 *49. Linux*에서 최대 핸들 수 대체

기타 고려사항

시스템 요구사항을 계획할 때 다시 시작하는 데 드는 시간의 길이를 고려하십시오. 계획된 중단 시간은 처리될 때까지 큐에 들어가 대기하는 메시지의 개수에 영향을 줄 수 있습니다. 메시지가 허용 가능한 범위의 시간 안에 처리 완료될 수 있도록 시스템을 구성하십시오. 디스크 스토리지, 메모리 및 처리 능력을 검토하십시오. 몇몇 클라이언트 애플리케이션의 경우 클라이언트가 다시 연결할 때 메시지를 버릴 수도 있습니다. 메시지를 제거하려면 클라이언트 연결 매개변수의 `CleanSession`을 설정하십시오. 정리 세션을 참조하십시오. 또는 MQTT 클라이언트에서 최상의 서비스 품질 (QoS) `0`을 사용하여 발행 및 구독하십시오. 서비스 품질을 참조하십시오. IBM MQ에서 메시지를 보낼 때 비지속 메시지를 사용하십시오. 이 서비스 품질의 메시지는 시스템이나 연결이 다시 시작할 때 복구되지 않습니다.

Windows Linux AIX MQ Telemetry에서 지원되는 디바이스

MQTT 클라이언트는 센서 및 작동기부터 휴대용 디바이스 및 차량 시스템까지 다양한 범위의 디바이스에서 실행할 수 있습니다.

MQTT 클라이언트는 크기가 작으며, 적은 메모리를 갖고 처리 능력이 낮은 디바이스에서 실행됩니다. MQTT protocol은 안정적이며 헤더가 작고, 이는 대역폭이 낮고 사용 비용이 높으며 단속적인 네트워크에 적합합니다.

MQ Telemetry는 MQTT 클라이언트 애플리케이션을 통해 텔레메트리 디바이스와 통신합니다. 이러한 애플리케이션은 모두 MQTT v3 프로토콜을 구현하는 다음 자원을 사용합니다.

- 다음 클라이언트 라이브러리:
  - *MQTT client for Java*, 이는 예를 들어 Android, OS X, Linux 또는 Windows 디바이스에 대한 고유 애플리케이션을 빌드하는 데 사용됩니다. 이 클라이언트 라이브러리를 사용하는 애플리케이션은 가장 작은 CLDC(Connected Limited Device Configuration)/MIDP(Mobile Information Device Profile)에서부터 CDC(Connected Device Configuration)/Foundation, J2SE(Java Platform, Standard Edition), 그리고 J2EE(Java Platform, Enterprise Edition)까지 Java의 모든 변형 형태에서 실행할 수 있습니다. IBM jclRM 사용자 정의된 클래스 라이브러리 또한 지원됩니다. Java ME 플랫폼은 작동 장치, 센서, 휴대전화 및 기타 임베드된 디바이스와 같은 소형 디바이스에 보통 사용됩니다. Java SE 플랫폼은 데스크탑 컴퓨터 및 서버와 같은 더 고사양의 임베드된 디바이스에 보통 설치됩니다.
  - *MQTT client for C*, 이는 예를 들어 iOS, OS X, Linux 또는 Windows 디바이스에 대한 고유 애플리케이션을 빌드하는 데 사용됩니다. 이 클라이언트 라이브러리는 Windows 및 Linux 시스템에 대해 사전 빌드된 고유 클라이언트와 함께 C 참조 구현을 제공합니다. C 참조 구현은 MQTT가 다양한 디바이스와 플랫폼으로 포트될 수 있도록 해 줍니다. Windows 7을 포함한 Intel의 일부 Windows 시스템, RedHat, Ubuntu 및 ARM 플랫폼(예: Eurotech Viper)의 일부 Linux 시스템은 C 클라이언트를 실행하는 Linux 버전을 구현하지만 IBM

에서는 이러한 플랫폼에 대한 서비스 지원을 제공하지 않습니다. 사용자의 IBM 지원 센터를 호출하려면 지원되는 플랫폼에서 클라이언트로 이 문제점을 다시 발생시켜야 합니다.

- *MQTT client for Java*, 이는 브라우저 기반 웹 애플리케이션을 빌드하는 데 사용됩니다.

MQTT 클라이언트 라이브러리는 Eclipse Paho 및 MQTT.org에서 무료로 제공됩니다. IBM MQ Telemetry Transport 샘플 프로그램을 참조하십시오.

IBM MQ의 보안

IBM MQ에는 권한 서비스 인터페이스, 사용자 작성 또는 써드파티, 채널 엑시트, TLS(Transport Layer Security)를 사용하는 채널 보안, 채널 인증 레코드 및 메시지 보안 등과 같이 보안을 제공하는 여러 가지 방법이 있습니다.

권한 서비스 인터페이스

MQI 호출에 대한 인증, 명령 및 오브젝트에 대한 액세스는 기본적으로 사용 설정된 **오브젝트 권한 관리자**(OAM)에 의해 제공됩니다. IBM MQ 엔티티에 대한 액세스는 IBM MQ 사용자 그룹 및 OAM을 통해 제어됩니다. 관리자는 명령행 인터페이스를 사용하여 필요에 따라 권한을 부여하거나 폐기할 수 있습니다.

권한 서비스 컴포넌트 작성에 대한 자세한 정보는 AIX, Linux, and Windows 시스템에서 보안 설정을 참조하십시오.

사용자 작성 또는 써드파티 채널 엑시트

채널은 사용자 작성 또는 써드파티 채널 엑시트를 사용할 수 있습니다. 자세한 정보는 메시지 채널의 채널-엑시트 프로그램을 참조하십시오.

TLS를 사용하는 채널 보안

TLS(Transport Layer Security) 프로토콜은 도청, 도용 및 위장에 대한 보호와 산업 표준 채널 보안을 제공합니다.

TLS는 메시지 기밀성 및 무결성을 제공하는 공용 키 및 대칭 기술과 상호 인증을 사용합니다.

TLS에 대한 자세한 정보를 포함하여 IBM MQ의 보안에 대한 포괄적인 검토는 보안을 참조하십시오. 이 절에 설명된 명령에 대한 포인터를 포함하여 TLS의 개요는 암호화 보안 프로토콜: TLS를 참조하십시오.

채널 인증 레코드

채널 인증 레코드를 사용하여 채널 레벨에서 시스템 연결에 부여된 액세스 권한에 대해 정확한 제어를 실행합니다. 자세한 정보는 채널 인증 레코드를 참조하십시오.

메시지 보안

IBM MQ의 별도로 설치되고 라이센스가 부여된 구성요소인 Advanced Message Security를 사용하여 IBM MQ를 사용하여 보내고 받은 메시지에 암호화 보호를 제공하십시오. Advanced Message Security의 내용을 참조하십시오.

관련 태스크

보안 설정

보안 요구사항 계획

IBM MQ.NET 관리 대상 클라이언트 TLS 지원

IBM MQ.NET 완전 관리 클라이언트는 Microsoft.NET SSLStreams 킷을 기반으로 하는 TLS(Transport Layer Security) 지원을 제공합니다. 이는 IBM Global Security Kit (GSKit)를 기반으로 하는 다른 IBM MQ 클라이언트와 다릅니다.

관리 모드 또는 비관리 모드로 실행되는 IBM MQ.NET 애플리케이션을 개발할 수 있습니다.

- 관리 모드에서 .NET 애플리케이션은 C MQI 호출 등의 교차 플랫폼 호출을 수행하지 않고 .NET CLR(Common Language Runtime) 내에서 작동합니다.

- 비관리 모드에서 C MQI는 기본 MQI 조작을 위해 호출됩니다. 기본적으로 비관리 모드 인터페이스는 C MQI 외에 .NET 랩퍼 클래스로 구성됩니다.

관리 IBM MQ.NET 클라이언트는 Microsoft.NET Framework 라이브러리를 사용하여 TLS 보안 소켓 프로토콜을 구현합니다. Microsoft 의 System.NET.Security.SSLStream 클래스는 IBM MQ.NET에서 보안 (TLS) 을 구현하는 데 사용됩니다.

비관리 IBM MQ.NET 클라이언트 모드는 C MQI (및 GSKit) 를 기반으로 하는 TLS 기능을 이미 지원합니다. 즉, TLS 조작은 C MQI에서 처리합니다. 이 경우 GSKit 는 TLS 보안 소켓 프로토콜을 구현합니다.

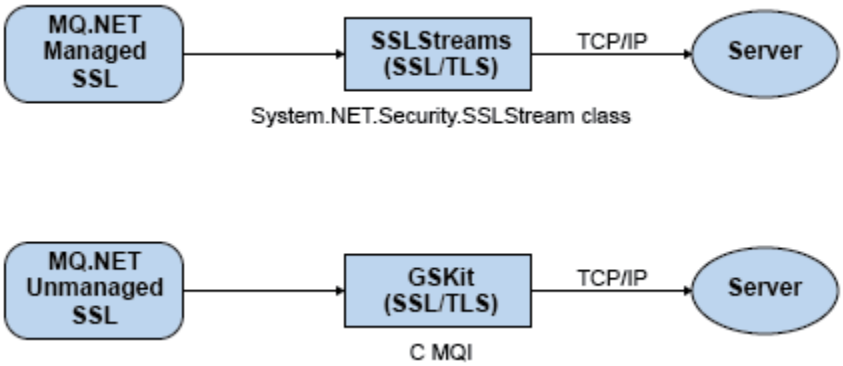


그림 *50. IBM MQ.NET* 관리 및 비관리 *TLS* 비교

다음 표는 관리와 비관리 구현 사이의 차이점을 요약합니다.

표 *14.* 관리와 비관리 구현 사이의 차이점

모드	프로토콜	구현	주석
IBM MQ.NET 관리 SSL	TLS	System.NET.Security.SSLStream 클래스 SSLStream 클래스는 연결된 TCP 소켓을 통한 스트림으로 작동	TLS 1.0 TLS 1.2(Microsoft.NET Framework v4.5 전용)
IBM MQ.NET 비관리 SSL	TLS	GSKIT 및 C-MQI	TLS 보안 소켓 프로토콜

관련 개념

.NET용 SSL(Secure Sockets Layer) 및 TLS(Transport Layer Security) 지원

IBM MQ MQI clients

IBM MQ MQI client는 큐 관리자가 실행되지 않는 시스템에 설치할 수 있는 IBM MQ 제품의 컴포넌트입니다.

IBM MQ MQI 클라이언트는 시스템에서 실행 중인 애플리케이션이 다른 시스템에서 실행 중인 큐 관리자에 대한 MQI 호출을 실행할 수 있도록 허용하는 컴포넌트입니다. 호출의 출력은 클라이언트로 되돌아오고 이는 애플리케이션으로 다시 전달됩니다.

IBM MQ MQI client를 사용하여 클라이언트와 동일한 시스템에서 실행되는 애플리케이션을 다른 시스템에서 실행 중인 큐 관리자에 연결할 수 있습니다. 애플리케이션은 해당 큐 관리자에게 MQI 호출을 발행할 수 있습니다. 이러한 애플리케이션을 IBM MQ MQI client 애플리케이션이라 하며 큐 관리자는 서버 큐 관리자라 합니다.

IBM MQ 서버는 하나 이상의 클라이언트에게 큐잉 서비스를 제공하는 큐 관리자입니다. 모든 IBM MQ 오브젝트 (예: 큐)는 큐 관리자 시스템(IBM MQ 서버 시스템)에만 존재하며 클라이언트에는 존재하지 않습니다. IBM MQ 서버는 로컬 IBM MQ 애플리케이션도 지원할 수 있습니다.

IBM MQ 서버와 일반 큐 관리자 간의 차이점은 서버가 각 클라이언트에 대한 전용 통신 링크를 가진다는 점입니다. 클라이언트 및 서버의 채널 작성에 대한 자세한 정보는 분산 큐잉 구성을 참조하십시오.

IBM MQ MQI client 애플리케이션과 서버 큐 관리자는 *MQI* 채널을 사용하여 서로 통신합니다. MQI 채널은 클라이언트 애플리케이션이 **MQCONN** 또는 **MQCONNX** 호출을 발행하여 큐 관리자에 연결할 때 시작하고 클라이언트 애플리케이션이 **MQDISC** 호출을 발행하여 큐 관리자와의 연결을 끊을 때 종료됩니다. MQI 호출의 입력 매개변수는 MQI 채널에서 한 방향으로 플로우하고 출력 매개변수는 반대 방향으로 플로우합니다.

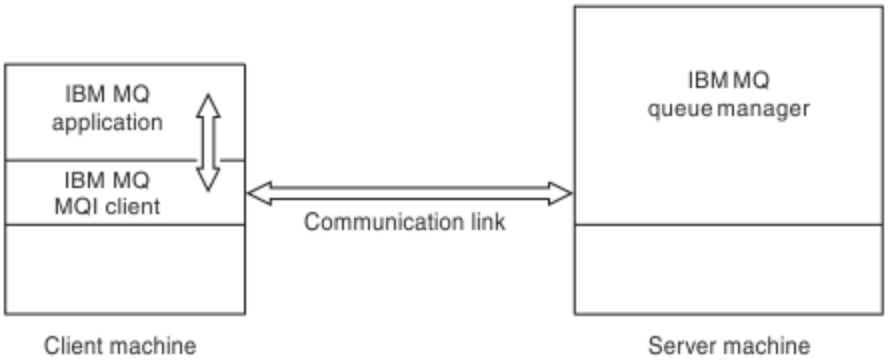


그림 *51.* 클라이언트와 서버 사이의 링크

다음 플랫폼을 사용할 수 있습니다. 이 결합은 사용하고 있는 IBM MQ 제품에 따라 다르며 126 페이지의 『IBM MQ 클라이언트의 플랫폼 지원』 에 설명되어 있습니다.

IBM MQ MQI client	**IBM MQ 서버**
AIX and Linux Windows IBM i	AIX and Linux Windows IBM i z/OS

MQI는 클라이언트 플랫폼에서 실행되고 있는 애플리케이션에 사용 가능하며 큐 및 기타 IBM MQ 오브젝트는 서버에 설치한 큐 관리자에 보유됩니다.

IBM MQ MQI client 환경에서 실행하려는 애플리케이션은 관련 클라이언트 라이브러리와 먼저 링크되어야 합니다. 애플리케이션이 MQI 호출을 발행하는 경우 IBM MQ MQI client가 이 요청을 큐 관리자에게 전달하면 큐 관리자에서 이 요청이 처리되며 응답이 IBM MQ MQI client로 다시 전송됩니다.

애플리케이션과 IBM MQ MQI client 사이의 링크는 런타임 시에 동적으로 설정됩니다.

또한 IBM MQ classes for .NET, IBM MQ classes for Java 또는 IBM MQ classes for Java Message Service(JMS)을(를) 사용하여 클라이언트 애플리케이션을 개발할 수 있습니다. 다음 플랫폼에서 Java 및 JMS 클라이언트를 사용할 수 있습니다.

- IBM i
- AIX
- Linux
- Windows

Java 및 JMS의 사용은 여기에 설명되어 있지 않습니다. IBM MQ classes for Java 및 IBM MQ classes for JMS의 설치, 구성 및 사용에 대한 자세한 내용은 IBM MQ classes for Java 사용 및 IBM MQ classes for JMS 사용을 참조하십시오.

클라이언트 서버 환경의 IBM MQ 애플리케이션

서버에 링크되면 클라이언트 IBM MQ 애플리케이션은 로컬 애플리케이션과 같은 방식으로 MQI 호출의 대부분을 발행할 수 있습니다. 클라이언트 애플리케이션은 지정된 큐 관리자에 연결하기 위한 MQCONN 호출을 발행합니다. 연결 요청에서 리턴된 연결 핸들을 지정하는 추가 MQI 호출은 이 큐 관리자가 처리할 수 있습니다.

애플리케이션을 적절한 클라이언트 라이브러리로 링크해야 합니다. IBM MQ MQI clients의 애플리케이션 빌드를 참조하십시오.

관련 개념

126 페이지의 『IBM MQ 클라이언트를 사용하는 이유』
IBM MQ 클라이언트 사용은 IBM MQ 메시징 및 큐잉을 구현하는 효율적인 방법입니다.

127 페이지의 『확장된 트랜잭션 클라이언트 개념』
IBM MQ 확장된 트랜잭션 클라이언트는 외부 트랜잭션 관리자의 제어 하에 다른 자원 관리자가 관리하는 자원을 업데이트할 수 있습니다.

128 페이지의 『클라이언트를 서버에 연결하는 방법』
클라이언트는 MQCONN 또는 MQCONNX를 사용하여 서버에 연결되고 채널을 통해서 통신합니다.

130 페이지의 『트랜잭션 관리 및 지원』
트랜잭션 관리 및 IBM MQ가 트랜잭션을 지원하는 방법을 소개합니다.

131 페이지의 『큐 관리자 기능 확장』
사용자 엑시트, API 엑시트 또는 설치 가능 서비스를 사용하여 큐 관리자 기능을 확장할 수 있습니다.

관련 정보

IBM MQ MQI client를 설정하는 방법

IBM MQ 클라이언트를 사용하는 이유

IBM MQ 클라이언트 사용은 IBM MQ 메시징 및 큐잉을 구현하는 효율적인 방법입니다.

한 시스템에서 실행 중인 MQI 및 다른 시스템에서 실행 중인 큐 관리자를 사용하는 애플리케이션을 가질 수 있습니다(실제 또는 가상). 이를 실행할 경우의 장점은 다음과 같습니다.

- 클라이언트 시스템에 전체 IBM MQ 구현이 필요하지 않습니다.
- 클라이언트 시스템에 대한 하드웨어 요구사항이 감소됩니다.
- 시스템 관리 요구사항이 감소됩니다.
- 클라이언트에서 실행되는 IBM MQ 애플리케이션을 다른 시스템의 다중 큐 관리자에 연결할 수 있습니다.
- 다른 전송 프로토콜을 사용하는 대체 채널을 사용할 수 있습니다.

IBM MQ 클라이언트의 플랫폼 지원

지원되는 모든 서버 플랫폼의 IBM MQ는 플랫폼의 IBM MQ MQI clients에서의 클라이언트 연결을 승인합니다.

지원되는 모든 서버 플랫폼에 기본 제품 및 서버 로 설치된 IBM MQ 는 다음 플랫폼의 IBM MQ MQI clients 에서 연결을 승인할 수 있습니다.

- IBM i
- AIX
- Linux
- Windows

클라이언트 연결은 CCSID(Coded Character Set ID)와 통신 프로토콜과의 차이에 따라 달라집니다.

IBM MQ MQI client에서 실행하는 애플리케이션

클라이언트 환경에서 전체 MQI가 지원됩니다. 이를 통해 IBM MQ MQI client 의 애플리케이션을 MQI 라이브러리가 아닌 MQIC 라이브러리에 링크하여 거의 모든 IBM MQ 애플리케이션을 IBM MQ MQI client 시스템에서 실행하도록 구성할 수 있습니다. 예외사항은 다음과 같습니다.

- 신호가 있는 MQGET
- 확장된 트랜잭션 클라이언트를 사용해야 하며 다른 자원 관리자와의 동기점 조정이 필요한 애플리케이션

미리 읽기를 사용하는 경우 비지속 메시징 성능을 향상시키는 데 모든 MQGET 옵션이 사용 가능한 것은 아닙니다. 테이블은 허용되는 옵션 및 이러한 옵션이 MOGET 호출 사이에서 대체될 수 있는지 여부를 표시합니다.

표 *15.* 미리 읽기가 사용 가능한 경우 허용되는 *MQGET* 옵션

값	**미리 읽기가 사용 가능한 경우 허용되며 MQGET 호출 간에 대체 가능**	**미리 읽기를 사용하는 경우 허용되지만 MQGET 호출 간에 대체될 수 없음[1]**	**미리 읽기가 사용 가능한 경우 허용되지 않는 MQGET 옵션[2]**
MQGET MD 값	MsgId [3] CorrelId [3]	Encoding CodedCharSetId	
MQGET MQGMO 옵션	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST[4] MQGMO_BROWSE_NEXT[4] MQGMO_BROWSE_MESSAGE_UNDER_CURSOR [4]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP_BACKOUT MQGMO_MSG_UNDER_CURSOR[4] MQGMO_LOCK MQGMO_UNLOCK
MQGMO 값		MsgHandle	

1. 이 옵션이 MQGET 호출 간에 대체되면 MQRC_OPTIONS_CHANGED 이유 코드가 리턴됩니다.
2. 이 옵션을 첫 번째 MQGET 호출에 지정할 경우 미리 읽기를 사용할 수 없습니다. 이 옵션을 후속 MQGET 호출에 지정할 경우 이유 코드 MQRC_OPTIONS_ERROR가 리턴됩니다.
3. 클라이언트 애플리케이션은 MsgId 및 CorrelId 값이 MQGET 호출 간에 대체된 경우, 이전 값을 가진 메시지가 이미 클라이언트에 전송되어 이용(또는 자동으로 제거)될 때까지 클라이언트 미리 읽기 버퍼에 남아있는지 알아야 합니다.
4. 첫 번째 MQGET 호출은 메시지를 찾아볼지 아니면 미리 읽기가 가능할 때 큐에서 가져올지 여부를 판별합니다. 애플리케이션이 찾아보기와 가져오기의 결합을 사용하려고 할 경우 MQRC_OPTIONS_CHANGED 이유 코드가 리턴됩니다.
5. MQGMO_MSG_UNDER_CURSOR는 미리 읽기에 사용할 수 없습니다. 메시지를 찾아보거나 미리 읽기가 가능할 때 가져올 수 있지만, 이 둘을 결합할 수는 없습니다.

IBM MQ MQI client에서 실행 중인 애플리케이션은 동시에 둘 이상의 큐 관리자에 연결되거나 MQCONN 또는 MQCONNX 호출에서 별표(*)가 있는 큐 관리자 이름을 사용할 수 있습니다(큐 관리자에 IBM MQ MQI client MQI 클라이언트 애플리케이션 연결의 예제 참조).

확장된 트랜잭션 클라이언트 개념

IBM MQ 확장된 트랜잭션 클라이언트는 외부 트랜잭션 관리자의 제어 하에 다른 자원 관리자가 관리하는 자원을 업데이트할 수 있습니다.

트랜잭션 관리의 개념에 익숙하지 않은 경우 130 페이지의 『트랜잭션 관리 및 지원』의 내용을 참조하십시오.

현재 XA 트랜잭션 클라이언트가 IBM MQ의 일부로 제공되고 있습니다.

클라이언트 애플리케이션은 연결된 큐 관리자가 관리하는 작업 단위에 참여할 수 있습니다. 작업 단위 내에서 클라이언트 애플리케이션은 해당 큐 관리자가 소유하는 큐에 메시지를 넣고 큐에서 메시지를 가져올 수 있습니다. 클라이언트 애플리케이션은 **MQCMIT** 호출을 사용하여 작업 단위를 커미트하거나 **MQBACK** 호출을 사용하여 작

업 단위를 백아웃할 수 있습니다. 그러나 동일한 작업 단위 내에서 클라이언트 애플리케이션은 다른 자원 관리자의 자원(예: Db2® 데이터베이스의 테이블)을 업데이트할 수 없습니다. IBM MQ 확장된 트랜잭션 클라이언트를 사용하면 이 제한사항이 제거됩니다.

IBM MQ 확장된 트랜잭션 클라이언트는 일부 추가 기능이 있는 IBM MQ MQI client입니다. 이 기능을 사용하면 동일한 작업 단위 내에 있는 클라이언트 애플리케이션에서 다음 태스크를 수행할 수 있습니다.

- 연결된 큐 관리자가 소유하는 큐에 메시지 넣기 및 큐에서 메시지 가져오기
- IBM MQ 큐 관리자가 아닌 자원 관리자의 자원 업데이트

이 작업 단위는 클라이언트 애플리케이션과 동일한 시스템에서 실행 중인 외부 트랜잭션 관리자에 의해 관리되어야 합니다. 작업 단위는 클라이언트 애플리케이션이 연결된 큐 관리자에 의해 관리될 수 없습니다. 이는 큐 관리자가 트랜잭션 관리자가 아닌 자원 관리자로서만 작업을 수행할 수 있음을 의미합니다. 또한 클라이언트 애플리케이션이 외부 트랜잭션 관리자가 제공한 API(Application Programming Interface)만을 사용하여 작업 단위를 커미트하거나 백아웃할 수 있음을 의미합니다. 클라이언트 애플리케이션은 따라서 MQI 호출, **MQBEGIN**, **MQCMIT** 및 **MQBACK**을 사용할 수 없습니다.

외부 트랜잭션 관리자는 큐 관리자에 연결된 클라이언트 애플리케이션이 사용하는 것과 동일한 MQI 채널을 사용하는 자원 관리자로서 큐 관리자와 통신합니다. 하지만 장애를 복구하는 상황에서, 실행되는 애플리케이션이 없을 때 트랜잭션 관리자는 전용 MQI 채널을 사용하여 큐 관리자가 실패 시에 참여하고 있었던 불완전한 작업 단위를 복구할 수 있습니다.

이 절에서 확장 트랜잭션 기능이 없는 IBM MQ MQI client는 IBM MQ 기본 클라이언트로 참조됩니다. 따라서 IBM MQ 확장된 트랜잭션 클라이언트가 확장 트랜잭션 기능이 추가된 IBM MQ 기본 클라이언트로 구성되는 것으로 간주할 수 있습니다.

참고: IBM i IBM i 의 IBM MQ MQI client 는 IBM MQ 확장 트랜잭션 기능을 지원하지 않습니다.

확장된 트랜잭션 클라이언트의 플랫폼 지원

Multi

확장된 트랜잭션 클라이언트는 기본 클라이언트를 지원하는 모든 멀티플랫폼에 대해 사용 가능합니다. z/OS에 대해 클라이언트가 사용 불가능합니다.

확장 트랜잭션 클라이언트를 사용하는 클라이언트 애플리케이션은 다음 IBM MQ 9.0이상 제품의 큐 관리자에만 연결할 수 있습니다.

- AIX IBM MQ for AIX
- IBM i IBM MQ for IBM i
- Linux IBM MQ for Linux
- Windows IBM MQ for Windows

z/OS z/OS에서 실행되는 확장된 트랜잭션 클라이언트가 없는 경우에도 확장된 트랜잭션 클라이언트를 사용 중인 클라이언트 애플리케이션을 z/OS에서 실행되는 큐 관리자에 연결할 수 있습니다.

각 플랫폼의 경우, 확장된 트랜잭션 클라이언트에 대한 하드웨어 및 소프트웨어 요구사항은 IBM MQ 기본 클라이언트에 대한 요구사항과 동일합니다. 프로그래밍 언어는 사용 중인 IBM MQ 기본 클라이언트 및 트랜잭션 관리자가 지원하는 경우에는 확장된 트랜잭션 클라이언트에 의해 지원됩니다.

모든 플랫폼의 외부 트랜잭션 관리자에 대한 정보는 시스템 요구 사항IBM MQ의 내용을 참조하십시오.

클라이언트를 서버에 연결하는 방법

클라이언트는 MQCONN 또는 MQCONNX를 사용하여 서버에 연결되고 채널을 통해서 통신합니다.

IBM MQ 클라이언트 환경에서 실행 중인 애플리케이션은 클라이언트와 서버 시스템 사이의 활성 연결을 유지해야 합니다.

MQCONN 또는 MQCONNX를 발행하는 애플리케이션을 통해 연결이 설정되었습니다. 클라이언트 및 서버는 *MQI* 채널을 통해 통신하거나 공유 대화를 사용하는 경우에는 각각의 대화가 MQI 채널 인스턴스를 공유합니다.

호출에 성공한 경우 MQI 채널 인스턴스 또는 대화는 애플리케이션이 MQDISC 호출을 발행할 때까지 연결된 상태로 유지됩니다. 이는 애플리케이션이 연결해야 하는 모든 큐 관리자에 해당됩니다.

동일한 시스템의 클라이언트 및 큐 관리자

시스템에 큐 관리자가 설치되어 있는 경우 IBM MQ MQI client 환경에서 애플리케이션을 실행할 수도 있습니다.

이 상황에서 큐 관리자 라이브러리에 링크할지 또는 클라이언트 라이브러리에 링크할지에 대한 선택을 할 수 있지만 클라이언트 라이브러리에 링크하는 경우 여전히 채널 연결을 정의할 필요가 있다는 점에 유의하십시오. 이는 애플리케이션의 개발 단계에서 유용할 수 있습니다. 자체 시스템에서 다른 시스템에 대한 종속성 없이 프로그램을 테스트할 수 있으며 프로그램을 독립적 IBM MQ MQI client 환경으로 이동시켜도 이 프로그램이 작동하는지에 대해 확인할 수 있습니다.

다른 플랫폼의 클라이언트

이 예에서는 서버 시스템이 다른 플랫폼에 있는 세 개의 IBM MQ MQI clients와 통신합니다.

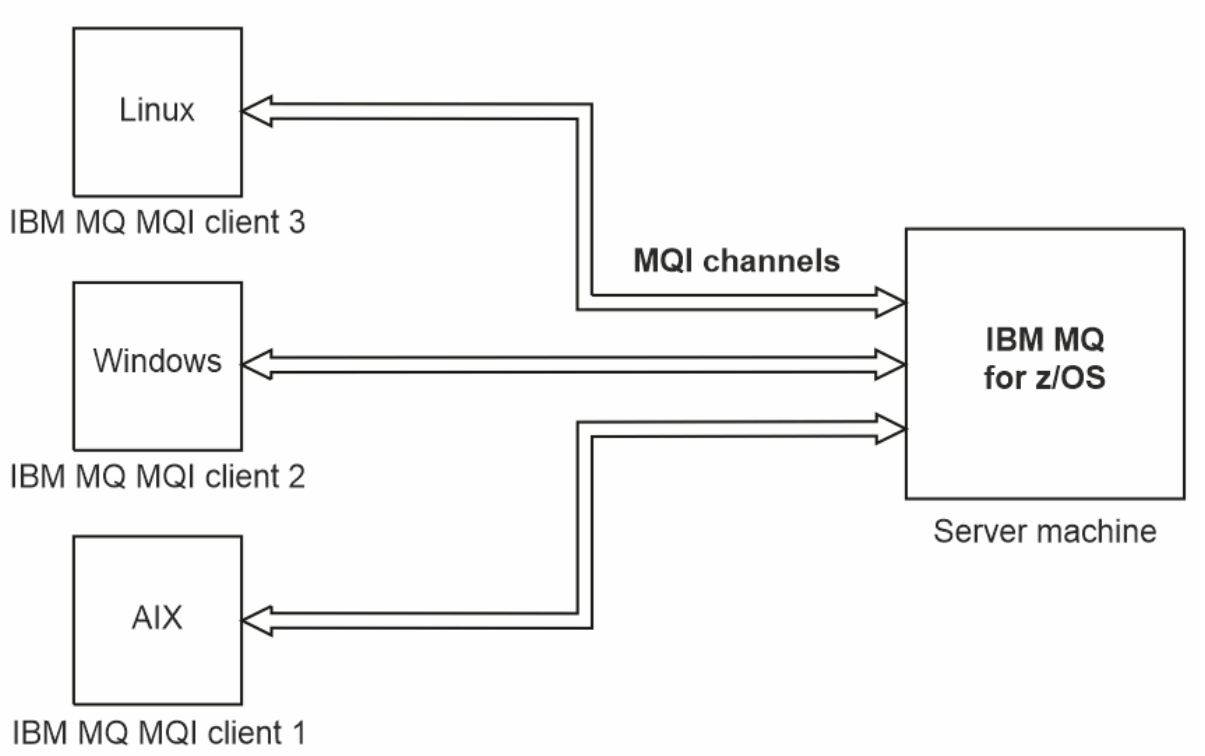


그림 *52.* 다른 플랫폼의 클라이언트에 연결된 *IBM MQ* 서버

다른 좀 더 복잡한 환경도 가능합니다. 예를 들어, IBM MQ 클라이언트는 둘 이상의 큐 관리자 또는 큐 공유 그룹의 일부로서 연결된 모든 큐 관리자에 연결할 수 있습니다.

다른 버전의 클라이언트 및 서버 소프트웨어 사용

이전 버전의 IBM MQ 제품을 사용하는 경우 클라이언트의 CCSID에서 코드 변환을 서버가 지원하는지 확인하십시오.

IBM MQ 클라이언트는 지원되는 모든 버전의 큐 관리자에 연결할 수 있습니다. 이전 버전 큐 관리자에 연결하는 경우, 클라이언트의 IBM MQ 애플리케이션에서 제품의 이후 버전 기능 및 구조를 사용할 수 없습니다.

IBM MQ 큐 관리자는 상호 지원되는 최상위 프로토콜 레벨로 협상하여 서로 다른 버전의 클라이언트와 자체적으로 통신할 수 있습니다. 이는 이전 클라이언트가 이후 큐 관리자 레벨에서 사용될 수 있음을 의미합니다. 문제점 진단을 용이하게 하고 IBM의 지원을 사용 가능하게 하기 위해 클라이언트 및 서버가 모두 현재 지원되는 IBM MQ 버전에 있는 것이 좋습니다.

자세한 정보는 애플리케이션 개발에서 지원되는 프로그래밍 언어를 참조하십시오.

트랜잭션 관리 및 지원

트랜잭션 관리 및 IBM MQ가 트랜잭션을 지원하는 방법을 소개합니다.

자원 관리자는 애플리케이션이 액세스하고 업데이트할 수 있는 자원을 소유하고 관리하는 컴퓨터 서브시스템입니다. 다음은 자원 관리자의 예입니다.

- 큐 자원이 있는 IBM MQ 큐 관리자
- 테이블 자원이 있는 Db2 데이터베이스

애플리케이션이 하나 이상의 자원 관리자의 자원을 업데이트할 때, 특정 업데이트가 모두 그룹으로 완료되거나 어떠한 업데이트도 완료되지 않도록 하는 비즈니스 요구사항이 있을 수 있습니다. 이 유형의 요구사항에 대한 이유는 이러한 업데이트 중 일부가 성공적으로 완료되었지만 그 외의 업데이트는 그렇지 못한 경우에 비즈니스 데이터가 일관되지 않은 상태로 남겨지기 때문입니다.

이러한 방식으로 관리되는 자원에 대한 업데이트는 작업 단위 또는 트랜잭션 내에서 발생합니다. 애플리케이션 프로그램은 한 세트의 업데이트를 하나의 작업 단위로 그룹화할 수 있습니다.

작업 단위 중에 애플리케이션은 자원 관리자에게 요청을 발행하여 자원 관리자의 자원을 업데이트합니다. 작업 단위는 애플리케이션이 모든 업데이트 커미트 요청을 발행할 때 종료됩니다. 업데이트가 커미트될 때까지 동일한 자원에 액세스 중인 다른 애플리케이션에 아무 업데이트도 표시되지 않습니다. 그렇지 않은 경우 애플리케이션이 어떤 이유로 작업 단위를 완료할 수 없다고 결정하면 애플리케이션은 요청했던 모든 업데이트를 해당 지점까지 백아웃하도록 하는 요청을 발행할 수 있습니다. 이러한 경우에 모든 업데이트는 다른 애플리케이션에 표시되지 않습니다. 이러한 업데이트는 보통 논리적으로 연관되어 있으며 보존되어야 하는 데이터 무결성에 대해 모두 성공적이어야 합니다. 다른 업데이트가 실패할 때 한 업데이트가 성공하면 데이터 무결성을 잃게 됩니다.

작업 단위가 성공적으로 완료되면 커미트라고 표시됩니다. 일단 커미트되면 해당 작업 단위 내에서 수행된 모든 업데이트는 영구적이 되며 되돌릴 수 없습니다. 하지만 작업 단위가 실패하면 모든 업데이트는 대신에 백아웃됩니다. 작업 단위가 커미트되거나 무결성으로 백아웃되는 이 프로세스를 동기점 조정이라고 합니다.

작업 단위 내 모든 업데이트가 커미트되거나 백아웃될 때의 지점을 동기점이라고 합니다. 작업 단위 내 업데이트는 동기점 제어 내에서 발생한다고 합니다. 애플리케이션이 동기점 제어 범위 외에 있는 업데이트를 요청하는 경우 진행 중인 작업 단위가 있고 업데이트를 나중에 백아웃할 수 없더라도 자원 관리자는 즉시 업데이트를 커미트합니다.

작업 단위를 관리하는 컴퓨터 서브시스템을 트랜잭션 관리자 또는 지점 통합기라고 합니다.

로컬 작업 단위에서는 업데이트된 유일한 자원이 IBM MQ 큐 관리자의 자원입니다. 여기에 1단계 커미트 프로세스를 사용하는 큐 관리자 스스로 동기점 조정을 제공합니다.

글로벌 작업 단위에서는 XA 준수 데이터베이스와 같이 다른 자원 관리자에 속하는 자원 또한 업데이트됩니다. 여기에, 2단계 커미트 프로시저가 사용되어야 하며 작업 단위는 큐 관리자 자체에서 조정할 수 있거나 IBM TXSeries® 또는 BEA Tuxedo와 같이 외부에서 다른 XA 준수 트랜잭션 관리자에 의해 조정할 수 있습니다.

트랜잭션 관리자는 작업 단위 내에서의 자원에 대한 모든 업데이트가 성공적으로 완료되는지 또는 업데이트가 전혀 완료되지 않는지 확인해야 합니다. 애플리케이션은 작업 단위의 커미트 또는 백아웃 요청을 트랜잭션 관리자에게 발행합니다. 트랜잭션 관리자의 예는 CICS 및 WebSphere Application Server이며, 이 두 트랜잭션 관리자는 모두 다른 기능도 가지고 있습니다.

일부 자원 관리자는 자체 트랜잭션 관리 기능을 제공합니다. 예를 들어, IBM MQ 큐 관리자는 자체 자원으로의 업데이트와 Db2 테이블로의 업데이트를 포함하여 작업 단위를 관리할 수 있습니다. 큐 관리자는 사용자가 요구하는 경우 트랜잭션 관리자를 사용할 수 있기는 하지만 이 기능을 수행할 별도의 트랜잭션 관리자를 필요로 하지 않습니다. 별도의 트랜잭션 관리자가 사용되는 경우 외부 트랜잭션 관리자로 참조됩니다.

작업 단위를 관리하는 외부 트랜잭션 관리자의 경우 트랜잭션 관리자와 작업 단위에 참여하는 모든 자원 관리자 사이에는 표준 인터페이스가 있어야 합니다. 이 인터페이스를 통해 트랜잭션 관리자와 자원 관리자가 서로 통신할 수 있습니다. 이러한 인터페이스 중 하나는 *XA* 인터페이스이며 이 인터페이스는 다수의 트랜잭션 관리자 및 자원 관리자가 지원하는 표준 인터페이스입니다. XA 인터페이스는 분배된 트랜잭션 처리*: XA* 스펙에서 The Open Group에 의해 발행됩니다.

둘 이상의 자원 관리자가 작업 단위에 참여한 경우 트랜잭션 관리자는 *2*단계 커미트 프로토콜을 사용하여 시스템 장애가 있다 하더라도 작업 단위 내의 모든 업데이트가 성공적으로 완료되었는지 또는 업데이트가 전혀 완료되지 않았는지 확인해야 합니다. 애플리케이션이 트랜잭션 관리자에게 작업 단위 커미트 요청을 발행하면 트랜잭션 관리자는 다음을 수행합니다.

단계 1(커미트 준비)
트랜잭션 관리자가 작업 단위에 참여하고 있는 각 자원 관리자에게 자원에 대해 예정된 업데이트에 대한 모든 정보가 복구 가능한 상태에 있는지 확인할 것을 요청합니다. 자원 관리자는 일반적으로 로그에 정보를 쓰고 정보가 하드 디스크를 통해 쓰여졌는지 확인하여 이러한 작업을 수행합니다. 단계 1은 트랜잭션 관리자가 각 자원 관리자로부터 자원에 대해 예정된 업데이트에 대한 정보가 복구 가능한 상태에 있다는 알림을 수신하면 완료됩니다.

단계 2(커미트)
단계 1이 완료되면 트랜잭션 관리자는 취소 불가능한 작업 단위 커미트 결정을 내립니다. 트랜잭션 관리자는 작업 단위에 참여하는 각 자원 관리자에게 자원에 대한 업데이트 커미트를 요청합니다. 자원 관리자가 이 요청을 수신하면 업데이트를 커미트해야 합니다. 자원 관리자에게는 이 단계에서 업데이트를 백아웃하는 옵션이 없습니다. 단계 2는 트랜잭션 관리자가 각 자원 관리자로부터 자원에 대한 업데이트를 커미트했다는 알림을 수신하면 완료됩니다.

XA 인터페이스는 2단계 커미트 프로토콜을 사용합니다.

자세한 정보는 트랜잭션 지원 시나리오를 참조하십시오.

또한 IBM MQ는 Microsoft Transaction Server(COM+)에 대한 지원을 제공합니다. Microsoft Transaction Server(COM+) 사용에서 COM+ 지원을 활용하도록 IBM MQ를 설정하는 방법에 대한 정보를 제공합니다.

큐 관리자 기능 확장

사용자 엑시트, API 엑시트 또는 설치 가능 서비스를 사용하여 큐 관리자 기능을 확장할 수 있습니다.

사용자 엑시트

사용자 엑시트는 사용자가 고유 코드를 큐 관리자 기능에 삽입할 수 있도록 하는 메커니즘을 제공합니다. 지원되는 사용자 엑시트에는 다음이 포함됩니다.

채널 엑시트
이 엑시트는 채널이 작동하는 방식을 변경합니다. 채널 엑시트는 메시지 채널에 대한 채널 엑시트 프로그램에 설명되어 있습니다.

데이터 변환 엑시트
이러한 엑시트는 데이터 형식을 하나의 형식에서 다른 형식으로 변환하는 애플리케이션 프로그램에 넣을 수 있는 소스 코드 단편을 작성합니다. 데이터 변환 엑시트는 데이터 변환 엑시트 작성에 설명되어 있습니다.

클러스터 워크로드 엑시트
이 엑시트가 수행하는 기능은 엑시트의 제공자에 의해 정의됩니다. 호출 정의 정보는 MQ_CLUSTER_WORKLOAD_EXIT - 호출 설명에 제공되어 있습니다.

API 엑시트

API 엑시트를 통해 사용자는 MQPUT 및 MQGET과 같이 IBM MQ API 호출의 작동을 변경하는 코드를 작성하고 해당 코드를 이러한 호출 바로 앞이나 뒤에 삽입할 수 있습니다. 삽입은 자동이며 큐 관리자는 등록된 지점에서 엑시트 코드를 드라이브합니다. API 엑시트에 대한 자세한 정보는 API 엑시트 사용 및 작성을 참조하십시오.

설치 가능 서비스

설치 가능 서비스가 다중 시작점이 있는 인터페이스(API)를 형식화했습니다.

설치 가능 서비스의 구현을 서비스 컴포넌트라고 합니다. IBM MQ에서 제공된 컴포넌트를 사용하거나 고유 컴포넌트를 작성하여 필요한 기능을 수행할 수 있습니다.

현재 다음 설치 가능 서비스가 제공됩니다.

권한 서비스
권한 서비스를 통해 고유 보안 기능을 빌드할 수 있습니다.

서비스를 구현하는 기본 서비스 컴포넌트는 오브젝트 권한 관리자(OAM)입니다. 기본적으로 OAM은 활성이며 OAM 구성을 위해 어떤 조치도 실행할 필요가 없습니다. 권한 서비스 인터페이스를 사용하여 OAM을 바꾸거나 보강시킬 다른 컴포넌트를 작성하십시오. OAM에 대한 자세한 정보는 AIX, Linux, and Windows 시스템에서 보안 설정을 참조하십시오.

이름 서비스

이름 서비스를 사용하면 애플리케이션이 리모트 큐를 로컬 큐인 것처럼 식별하여 큐를 공유할 수 있습니다.

고유의 이름 서비스 컴포넌트를 작성할 수 있습니다. 예를 들어 IBM MQ에 대해 이름 서비스를 사용하려는 경우 이러한 작업을 수행하려고 할 수 있습니다. 이름 서비스를 사용하려면 사용자 작성 또는 다른 소프트웨어 벤더가 공급한 컴포넌트가 있어야 합니다. 기본적으로, 이름 서비스는 비활성 상태입니다.

관련 개념

사용자 엑시트, API 엑시트 및 IBM MQ 설치 가능 서비스

IBM MQ Java 언어 인터페이스

IBM MQ 는 Java 애플리케이션에서 사용할 세 개의 API (Application Programming Interface) (IBM MQ classes for Jakarta Messaging, IBM MQ classes for JMS및 IBM MQ classes for Java) 를 제공합니다.

IBM 는 개방형 표준을 지원하며 개방형 표준의 활성 참여자입니다.

- IBM MQ 8.0부터 제품은 공유 구독과 같은 기능과 함께 단순화된 새 API를 도입한 JMS 2.0 표준을 구현합니다.
- V 9.3.0 V 9.3.0 IBM MQ 9.3.0부터 Jakarta Messaging 3.0 도 지원됩니다.
- 또한 WebSphere Liberty 는 IBM MQ와 함께 JMS 2.0 및 Jakarta Messaging 3.0 를 지원합니다.

IBM MQ 내에는 Java 애플리케이션에서 사용할 세 개의 API가 있습니다.

V 9.3.0 V 9.3.0 JM 3.0 **IBM MQ classes for Jakarta Messaging**

IBM MQ classes for Jakarta Messaging 는 IBM MQ 용 Jakarta Messaging 인터페이스를 메시징 시스템으로 구현하는 Jakarta Messaging 제공자입니다. Jakarta Connectors Architecture 는 Jakarta EE 환경에서 실행 중인 애플리케이션을 IBM MQ 또는 Db2와 같은 EIS (Enterprise Information System) 에 연결하는 표준 방법을 제공합니다.

JMS 2.0 **IBM MQ classes for JMS**

IBM MQ classes for JMS 는 IBM MQ 용 JMS 인터페이스를 메시징 시스템으로 구현하는 JMS 제공자입니다. Java Platform, Enterprise Edition Connector Architecture(JCA)는 Java EE 환경에서 실행 중인 애플리케이션을 EIS(Enterprise Information System)(예: IBM MQ 또는 Db2)에 연결하는 표준 방법을 제공합니다.

IBM MQ classes for Java

IBM MQ classes for Java 를 사용하면 Java 환경에서 IBM MQ 를 사용할 수 있습니다. IBM MQ classes for Java를 사용하면 Java 애플리케이션을 IBM MQ 클라이언트로 IBM MQ에 연결하거나 IBM MQ 큐 관리자에 직접 연결할 수 있습니다.

참고:

- V 9.3.0 V 9.3.0 JMS 2.0 은 Jakarta Messaging로 대체되었습니다. IBM MQ classes for JMS 는 JMS 2.0 표준을 계속 지원하지만 Java 메시징에 대한 향후 개선사항은 Jakarta Messaging에서만 나타나므로 IBM MQ classes for Jakarta Messaging에서도 나타납니다. IBM MQ classes for JMS 는 기존 JMS 2.0 애플리케이션을 유지보수하고 확장하는 경우에만 권장됩니다. IBM MQ classes for Jakarta Messaging 는 새 개발에 선호되는 기술이어야 합니다.
- Stabilized IBM MQ classes for Java는 IBM MQ 8.0에서 제공된 레벨에서 기능을 안정적으로 사용할 수 있습니다. IBM MQ classes for Java를 사용하는 기존 애플리케이션이 계속해서 완전히 지원되지만 이 API가 안정되었으므로 새 기능이 추가되지 않으며 개선 요청도 수락되지 않습니다. 완전히 지원이란 IBM MQ 시스템 요구사항의 변경에 따라 필요한 모든 변경사항과 함께 결함이 수정되는 것을 의미합니다.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3부터 IBM MQ classes for Java, IBM MQ classes for JMS및 IBM MQ classes for Jakarta Messaging 는 Java 8를 사용하여 빌드됩니다. 이러한 레벨 이상의 Java 런타임 환경은 이러한 인터페이스를 사용하여 애플리케이션을 실행하는 데 사용해야 합니다.

관련 개념

Java 에서 IBM MQ 에 액세스-API 선택

V 9.3.0 V 9.3.0 Jakarta Messaging에서 IBM MQ 클래스를 사용해야 하는 이유는 무엇입니까?

IBM MQ classes for JMS를 사용해야 하는 이유가 무엇입니까?
IBM MQ classes for Java를 사용해야 하는 이유가 무엇입니까?

IBM MQ classes for JMS/Jakarta Messaging

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 는 IBM MQ와 함께 제공되는 메시징 제공자입니다. 이러한 각 제공자는 메시징 API에 대한 두 개의 확장 세트도 제공합니다. Java Platform, Standard Edition (Java SE) 및 Java Platform, Enterprise Edition (Java EE) 애플리케이션 모두 이러한 메시징 제공자를 사용할 수 있습니다.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 에 대한 지원을 소개합니다 Jakarta Messaging 3.0. JMS 2.0 는 여전히 완벽하게 지원됩니다.

JMS 및 Jakarta Messaging 스펙은 애플리케이션이 메시징 조작을 수행하는 데 사용할 수 있는 인터페이스 세트를 정의합니다. IBM MQ 8.0부터 제품은 JMS 표준의 JMS 2.0 버전을 지원합니다. 이 구현은 클래식 API의 모든 기능을 제공하지만 필요한 인터페이스가 더 적고 사용이 더 단순합니다. 자세한 정보는 136 페이지의 『JMS 및 Jakarta Messaging 모델』 및 JMS 2.0 스펙 (Java.net) 을 참조하십시오. JM 3.0 IBM MQ 9.3.0부터 Jakarta Messaging 도 지원됩니다.

jakarta.jms (Jakarta Messaging 3.0) 또는 javax.jms (JMS 2.0) 패키지는 메시징 인터페이스의 세부사항을 지정하며 메시징 제공자는 특정 메시징 제품에 대해 이러한 인터페이스를 구현합니다. 예를 들면, 다음과 같습니다.

- IBM MQ classes for JMS 는 IBM MQ 용 JMS 인터페이스를 구현하는 JMS 제공자이며 JMS API에 대한 다음 두 개의 확장 세트도 제공합니다.
  - IBM MQ JMS 확장
  - IBM JMS 확장
- javax.dims또는 jakarta.jms인터페이스 또는 JMS 확장 세트를 사용하여 작성된 연결 팩토리, 큐 또는 토픽 오브젝트는 이러한 API중 하나를 사용하여 주소 지정할 수 있습니다. 즉, 임의의 인터페이스로 캐스트할 수 있습니다. 애플리케이션 이식 가능성을 최고 수준으로 유지하려면 사용자의 요구사항에 맞는 가장 일반적인 API를 사용하십시오.

JMS 와 은 공통점이 많으므로 이 주제에서 을 참조하는 것은 두 가지를 모두 참조하는 것으로 간주할 수 있습니다. Jakarta Messaging JMS 필요에 따라 차이점이 강조 표시됩니다.

IBM MQ JMS 확장

IBM MQ classes for JMS는 또한 JMS API에 대한 확장 기능을 제공합니다. IBM MQ classes for JMS 에는 MQConnectionFactory, MQQueue및 MQTopic 오브젝트에서 구현되는 확장이 포함되어 있습니다. 이러한 오브젝트에는 IBM MQ에 특정한 특성 및 메소드가 있습니다. 오브젝트는 관리 대상 오브젝트일 수 있거나 애플리케이션은 런타임에 오브젝트를 동적으로 작성할 수 있습니다. 이러한 확장을 IBM MQ JMS 확장이라고 합니다. 이 문서에서 런타임 시 애플리케이션에 의해 동적으로 작성되는 오브젝트는 관리 오브젝트로 간주되지 않습니다.

IBM JMS 확장

IBM MQ JMS 확장 외에도 IBM MQ classes for JMS 는 사용되는 프로그래밍 언어로 JMS API 또는 Java 에 보다 일반적인 확장 세트를 제공합니다. 이러한 확장을 IBM JMS 확장이라고 하며 다음과 같은 광범위한 목표가 있습니다.

- IBM JMS 제공자 간에 더 높은 레벨의 일관성을 제공합니다.
- 두 IBM 메시징 시스템 간에 브릿지 애플리케이션을 더 쉽게 작성할 수 있습니다.
- 한 IBM JMS 제공자에서 다른 제공자로 애플리케이션을 더 쉽게 이식할 수 있습니다.

이러한 확장의 주요 초점은 런타임에 동적으로 연결 팩토리 및 목적지를 작성 및 구성하는 점에 맞춰져 있지만, 확장에서는 문제점 판별 기능과 같이 메시징과 직접적으로 관련되지 않은 기능도 제공합니다.

관련 태스크

JMS/Jakarta Messaging에 IBM MQ 클래스 사용
JMS및 Jakarta Messaging 자원 구성

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ classes for Jakarta Messaging: 개요

IBM MQ 9.3.0 에서는 Jakarta Messaging에 대한 지원을 소개합니다. Jakarta Messaging 3.0의 경우 JMS 스펙의 제어가 Oracle 에서 Java 커뮤니티 프로세스로 이동되었습니다. 그러나, Oracle 는 다른 Java 기술에 사용되는 "javax" 이름에 대한 통제권을 유지합니다. Jakarta Messaging 3.0 는 기능적으로 JMS 2.0 와 동일하지만, 이름에 약간의 차이가 있습니다. 3.0 의 공식 명칭은 Jakarta Messaging 이고, Java Message Service 가 아닙니다. 패키지와 상수 이름은 `javax` 가 아니라 `jakarta` 로 시작합니다.

백그라운드

수년 동안 Java 플랫폼은 두 가지 양식 (Standard Edition 및 Enterprise Edition) 으로 제공되었습니다.

Java Platform, Standard Edition (때로는 Java SE로 축약됨) 는 독립형 컨텍스트에서 실행할 수 있는 코어 언어 및 클래스 라이브러리입니다. Java SE 의 대부분의 Java 패키지 이름은 "java." 로 시작합니다.

Java Platform, Enterprise Edition (Java EE) 는 메시징, 다양한 Bean, 트랜잭션성 등과 같은 기능을 추가하여 이를 확장합니다. 이러한 기술 중 일부는 Java SE 컨텍스트에서도 사용할 수 있습니다. Java EE 의 대부분의 Java 패키지에는 "javax" 로 시작하는 이름이 있습니다. 그러나 일부 교차가 있으므로 일부 Java SE 패키지에는 "javax" 가 있습니다. 이름에 대한 접두부로 사용됩니다.

Java Message Service (JMS) 는 Java Platform, Enterprise Edition의 일부입니다. Java EE 7 는 JMS 2.0를 통합합니다.

Java EE 7까지 이 기술은 Oracle의 관리 하에 있었습니다.

Java EE 기술은 최근에 Oracle 의 스튜워드십에서 Eclipse Foundation이 감독하는 커뮤니티 프로세스로 이동했습니다.

"javax." 이름을 새 프로젝트로 이동할 수 없습니다. 새 이름 지정이 채택되었습니다. 모든 패키지 및 특성 이름에 "jak자카르타" 라는 접두부가 추가되었습니다. Java Platform, Enterprise Edition 은 나중에 "Jakarta EE" 로 이름 지정됩니다. 버전 번호 지정이 계속되었습니다. 버전 8은 대부분 무시할 수 있는 임시 버전이며 Jakarta EE 9 는 "jak자카르타" 가 있는 지점입니다. 접두부가 적용됩니다.

IBM MQ 컨텍스트에 적용되는 기본 Jakarta EE 기술은 Jakarta Messaging 3.0 - Java Message Service (JMS) 2.0의 후속 기술입니다. 따라서 Jakarta EE 9 는 Jakarta Messaging 3.0를 통합합니다.

Jakarta EE 9 및 Jakarta Messaging 3.0에 대한 지원을 도입하는 동안 IBM MQ 는 계속해서 Java EE 7 및 JMS 2.0를 지원합니다.

제공되는 내용: Java SE

Java Platform, Standard Edition의 경우 IBM MQ classes for JMS (IBM MQ에서 JMS 2.0 조작을 지원함) IBM MQ 9.3.0 외에도 IBM MQ classes for Jakarta Messaging를 제공합니다. 이러한 클래스는 IBM MQ와 통합되는 Jakarta Messaging 3.0 제공자를 제공하여 Jakarta Messaging 조작을 용이하게 하기 위해 IBM MQ 큐 관리자를 사용할 수 있도록 합니다.

이 파일은 IBM MQ 설치의 `java/lib` 서브디렉토리에 표준 JAR 파일 `com.ibm.mq.jakarta.client.jar` 로 제공됩니다.

OSGi 컨테이너 (예: Apache Felix 또는 Eclipse Equinox IBM MQ) 에서 사용하기 위해 다음과 같은 OSGi번들 쌍도 제공합니다.

- `com.ibm.mq.osgi.jms30.clientprereqs_`*`V.R.M.F`*`.jar`
- `com.ibm.mq.osgi.jms30.client_`*`V.R.M.F`*`.jar`

여기서 *V.R.M.F* 는 IBM MQ의 버전을 나타냅니다 (예: 9.3.0.0). 이러한 번들은 IBM MQ 설치의 `java/lib/OSGi` 서브디렉토리에서 찾을 수 있습니다.

제공되는 것: Jakarta EE 9 와 Jakarta EE 10

Jakarta EE 9 과 그 이후 버전에서 호환되는 애플리케이션 서버에서 IBM MQ 기반 메시징을 지원하기 위해, IBM MQ 은 Jakarta Messaging 을 위한 리소스 어댑터를 제공합니다: `wmq.jakarta.jmsra.rar`. 이 파일은 IBM MQ 설치의 `java/lib/jca` 하위 디렉토리에서 찾을 수 있습니다.

IBM MQ Java EE 7 `wmq.jmsra.rar` 와 호환되는 리소스 어댑터()를 IBM MQ 설치의 `java/lib/jca` 하위 디렉토리에 계속 제공합니다.

이러한 아티팩트가 전달되는 방법

이 JAR 파일과 리소스 어댑터용 RAR 파일은 일반적인 IBM MQ 설치 미디어에 포함된 기존 아티팩트와 함께 패키징됩니다. 여기에는 " `.rpm` " 파일과 같은 플랫폼별 설치 미디어와 자동 압축 풀기 방식의 재배포 가능 클라이언트 JAR 파일과 같은 재배포 가능 미디어가 모두 포함됩니다.

JMS 2.0 및 Jakarta Messaging 3.0 간에 변경된 사항

Jakarta EE 9 및 Jakarta Messaging 3.0 에서는 새 기능을 도입하지 않습니다. 모든 변경사항은 이름입니다. 예를 들어, JMS 2.0에서 "javax.jms.Connection" 을 사용하는 경우 Jakarta Messaging 3.0에서 "jakarta.jms.Connection" 을 사용합니다.

Eclipse Foundation은 Jakarta EE 플랫폼을 앞으로 사용하므로 이 기반에서 빌드되며 이 이름 지정 규칙은 나중에 도입되는 새 기능에 사용됩니다.

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 간에 변경된 사항

요약

JMS 2.0에 대한 지원을 제공하는 IBM MQ classes for JMS는 계속 사용 가능하며 기본적으로 기존 애플리케이션을 유지보수하고 확장하는 데 권장됩니다. 완전히 지원됩니다.

Jakarta Messaging 3.0에 대한 지원을 제공하는 IBM MQ classes for Jakarta Messaging는 새 개발에 권장됩니다.

IBM MQ 9.3.0에서 이러한 두 오퍼링은 기능적으로 동등합니다. 이름만 다릅니다. 그러나 새 메시징 기능은 IBM MQ classes for JMS에서보다 IBM MQ classes for Jakarta Messaging 에서 나타날 가능성이 높습니다.

두 오퍼링은 상호 운용 가능합니다. IBM MQ classes for JMS 에서 생성된 메시지는 IBM MQ classes for Jakarta Messaging에서 이용할 수 있으며 그 반대의 경우도 가능합니다. 그러나 두 오퍼링은 단일 애플리케이션에서 공존할 수 없습니다.

이름 지정 변경사항

표 *16.* 패키지 이름에 대한 변경사항

IBM MQ classes for JMS 패키지 이름	**IBM MQ classes for Jakarta Messaging 패키지 이름**
com.ibm.mq.jms[.*]	com.ibm.mq.jakarta.jms[.*]
com.ibm.jms	com.ibm.jakarta.jms
com.ibm.msg.client.jms.*	com.ibm.msg.client.jakarta.jms.*
com.ibm.msg.client.wmq.*	com.ibm.msg.client.jakarta.wmq.*

공통 서비스 (추적, 로깅, 자국어 지원 등) 및 JMQI 구현 (로컬 및 원격) 과 관련된 패키지는 IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging모두에 공통되므로 이러한 영역에서 변경이 필요하지 않습니다.

특성 이름도 변경되었습니다. 예를 들어, IBM MQ classes for Jakarta Messaging 에서 IBM MQ 확장을 사용으로 설정하는 특성은 **`com.ibm.mq.jakarta.jms.SupportMQExtensions`**입니다.

IBM MQ classes for JMS 또는 IBM MQ classes for Jakarta Messaging에 독립적인 특성 이름 (예: 다양한 **`com.ibm.msg.client.commonservices.trace.*`** 특성) 은 두 오퍼링 모두에 동일하게 적용됩니다.

관리 유틸리티

crtmqenv 및 **setmqenv** 유틸리티는 이제 클래스 경로가 IBM MQ classes for JMS (`-j 2.0`) 또는 IBM MQ classes for Jakarta Messaging (`-j 3.0`) 에 대해 구성되어야 하는지 여부를 지정하는 옵션을 승인하며, **runjms30** 및 유사한 이름이라고 하는 **runjms** 유틸리티의 IBM MQ classes for Jakarta Messaging 변형이 있습니다.

dspmqver 유틸리티는 Java 구성요소에 대해 보고하도록 요청될 때 출력에 IBM MQ classes for Jakarta Messaging 를 포함합니다.

JNDI를 통해 IBM MQ classes for Jakarta Messaging 오브젝트를 검색하도록 구성하기 위해 새 **JMS30Admin** 유틸리티는 IBM MQ classes for JMS용 **JMSAdmin** 유틸리티와 동일합니다.

기본 오브젝트는 서로 다른 패키지에서 가져온 것입니다. **JMSAdmin** 에 의해 작성된 JNDI 정의는 IBM MQ classes for Jakarta Messaging에 의해 사용될 수 없으며, **JMS30Admin** 에 의해 작성된 JNDI 정의는 IBM MQ classes for JMS에 의해 사용될 수 없습니다.

참고: IBM MQ Explorer에서 제공하는 IBM MQ classes for Jakarta Messaging 오브젝트에 대한 지원은 없습니다. 해당 JNDI 통합은 IBM MQ classes for JMS 전용입니다.

관련 개념

Jakarta Messaging에서 IBM MQ 클래스를 사용해야 하는 이유는 무엇입니까?

JMS 및 Jakarta Messaging 모델

JMS 및 Jakarta Messaging 모델은 Java 애플리케이션이 메시징 조작을 수행하는 데 사용할 수 있는 인터페이스 세트를 정의합니다. IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 는 Java 메시징 오브젝트가 IBM MQ 개념과 관련되는 방법을 정의하는 메시징 제공자입니다. JMS 및 Jakarta Messaging 스펙은 특정 메시징 오브젝트가 관리 오브젝트가 될 것으로 예상합니다.

IBM MQ 8.0부터 제품은 JMS 1.1의 클래식 API를 유지하면서 단순화된 API를 도입한 JMS 표준의 JMS 2.0 버전을 지원합니다.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 에 대한 지원을 소개합니다 Jakarta Messaging 3.0. JMS 2.0 는 여전히 완벽하게 지원됩니다. JMS 와 은 공통점이 많으므로 이 주제에서 을 참조하는 것은 두 가지를 모두 참조하는 것으로 간주할 수 있습니다. Jakarta Messaging JMS 필요에 따라 차이점이 강조 표시됩니다.

간소화된 API

JMS 2.0 에서는 JMS 1.1의 도메인 특정 및 도메인 독립 인터페이스를 유지하면서 단순화된 API를 도입했습니다. 간소화된 API에서는 메시지를 송신하고 수신하는 데 필요한 오브젝트 수가 줄어들며 다음 인터페이스로 구성됩니다.

ConnectionFactory
ConnectionFactory는 JMS 클라이언트에서 Connection을 작성하는 데 사용하는 관리 대상 오브젝트입니다. 이 인터페이스는 클래식 API에서도 사용합니다.

JMS컨텍스트
이 오브젝트는 클래식 API의 Connection 및 Session 오브젝트를 결합합니다. JMSContext 오브젝트는 다른 JMSContext 오브젝트에서 작성할 수 있으며 기본 연결은 복제됩니다.

JMS 생성자
JMSProducer는 JMSContext를 통해 작성되며 큐나 토픽에 메시지를 송신하는 데 사용합니다. JMSProducer 오브젝트를 사용하면 메시지를 송신하는 데 필요한 오브젝트가 작성됩니다.

JMS 이용자
JMSConsumer는 JMSContext를 통해 작성되며 토픽이나 큐에서 메시지를 송신하는 데 사용합니다.

간소화된 API의 효과는 다음과 같이 여러 가지가 있습니다.

- JMSContext 오브젝트에서 항상 자동으로 기본 연결을 시작합니다.
- JMSProducers와 JMSConsumers는 이제 Message의 getBody 메소드를 사용하여 전체 Message 오브젝트를 가져올 필요 없이 메시지 본문과 직접 작업할 수 있습니다.

- Message 특성은 메시지 컨텐츠인 'body'를 송신하기 전에 메소드 체인을 사용하여 JMSProducer 오브젝트에 설정할 수 있습니다. JMSProducer는 메시지를 송신하는 데 필요한 모든 오브젝트의 작성을 핸들링합니다. JMS 2.0을 사용하여 특성을 설정할 수 있으며 메시지가 다음과 같이 송신됩니다.

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

JMS 2.0 에서는 여러 이용자 간에 메시지를 공유할 수 있는 공유 구독도 도입했습니다. 모든 JMS 1.1 구독은 공유하지 않는 구독으로 처리됩니다.

클래식 API

다음 목록은 클래식 API의 기본 JMS 인터페이스를 요약합니다.

목적지
목적지는 애플리케이션이 메시지를 보내는 위치이거나 애플리케이션이 메시지를 수신하는 소스입니다.

ConnectionFactory
ConnectionFactory 오브젝트는 연결을 위한 구성 특성 세트를 캡슐화합니다. 애플리케이션에서 연결 팩토리를 사용하여 연결을 작성합니다.

연결
Connection 오브젝트는 메시징 서버에 대한 애플리케이션의 활성화된 연결을 캡슐화합니다. 애플리케이션은 연결을 사용하여 세션을 작성합니다.

세션
세션은 메시지 송신 및 수신을 위한 단일 스레드 컨텍스트입니다. 애플리케이션은 세션을 사용하여 메시지, 메시지 생성자 및 메시지 이용자를 작성합니다. 세션은 트랜잭션되거나 트랜잭션되지 않습니다.

메시지
Message 오브젝트는 애플리케이션이 송신하거나 수신하는 메시지를 캡슐화합니다.

MessageProducer
애플리케이션은 메시지 생성자를 사용하여 메시지를 목적지에 송신합니다.

MessageConsumer
애플리케이션은 메시지 이용자를 사용하여 목적지로 송신된 메시지를 수신합니다.

137 페이지의 그림 53에서는 이러한 오브젝트와 해당 관계를 표시합니다.

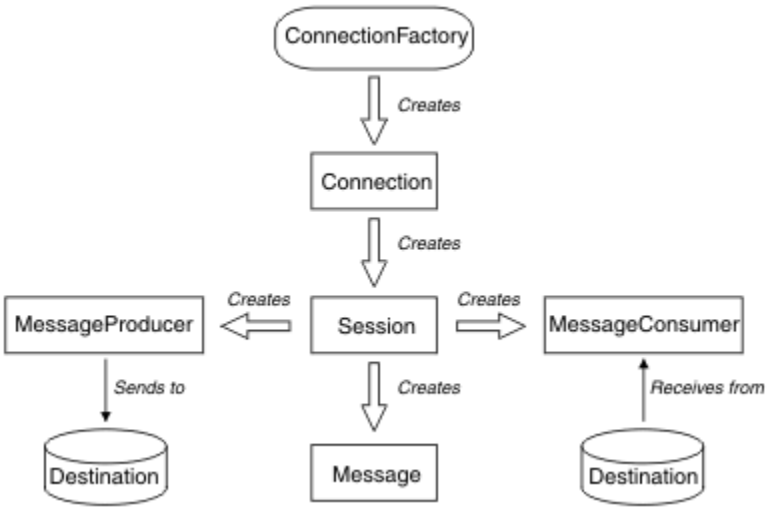


그림 *53. JMS* 오브젝트 및 해당 관계

다이어그램은 기본 인터페이스인 연결 팩토리, 연결, 세션, 메시지 생성자, 메시지 이용자, 메시지, 목적지를 보여줍니다. 애플리케이션은 연결 팩토리를 사용하여 연결을 작성하고 연결을 사용하여 세션을 작성합니다. 그런 다음 애플리케이션은 세션을 사용하여 메시지, 메시지 생성자 및 메시지 이용자를 작성할 수 있습니다. 애플리케이

션은 메시지 생성자를 사용하여 메시지를 목적지로 송신하고 메시지 이용자를 사용하여 목적지로 송신된 메시지를 수신합니다.

Destination, ConnectionFactory 또는 Connection 오브젝트는 멀티스레드 애플리케이션의 여러 다른 스레드에서 동시에 사용할 수 있지만, Session, MessageProducer 또는 MessageConsumer 오브젝트는 여러 다른 스레드에서 동시에 사용할 수 없습니다. Session, MessageProducer 또는 MessageConsumer 오브젝트를 동시에 사용하지 않는 가장 간단한 방법은 각 스레드의 개별 Session 오브젝트를 작성하는 것입니다.

JMS에서는 다음 두 스타일의 메시징을 지원합니다.

- 포인트-투-포인트 메시징
- 발행/구독 메시징

이러한 스타일의 메시징은 메시징 도메인이라고도 하며 한 애플리케이션에 두 스타일의 메시징을 모두 결합할 수 있습니다. 포인트-투-포인트 도메인에서는 목적지가 큐이며 발행/구독 도메인에서는 목적지가 토픽입니다.

JMS 1.1이전의 JMS 버전에서는 지점간 도메인에 대한 프로그래밍이 한 세트의 인터페이스 및 메소드를 사용하고 발행/구독 도메인에 대한 프로그래밍이 다른 세트를 사용합니다. 두 세트는 비슷하지만 분리되어 있습니다. JMS 1.1에서 두 메시징 도메인을 모두 지원하는 공통 인터페이스와 메소드 세트를 사용할 수 있습니다. 공통 인터페이스에서는 각 메시징 도메인의 도메인 독립적 보기를 제공합니다. 138 페이지의 표 17에서는 JMS 도메인 독립적 인터페이스와 대응하는 도메인 특정 인터페이스를 나열합니다.

표 *17. JMS* 도메인 독립적 인터페이스와 해당하는 도메인 특정 인터페이스

도메인 독립적 인터페이스	**포인트-투-포인트 도메인의 도메인 특정 인터페이스**	**발행/구독 도메인의 도메인 특정 인터페이스**
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
연결	QueueConnection	TopicConnection
목적지	큐	토픽
세션	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 2.0 IBM MQ classes for JMS 2.0 는 이전 JMS 1.1 도메인 특정 인터페이스와 JMS 2.0의 단순화된 API를 모두 지원합니다. 따라서 IBM MQ classes for JMS 2.0 는 기존 애플리케이션에서 새 기능 개발을 포함하여 기존 애플리케이션을 유지보수하는 데 사용할 수 있습니다.

JM 3.0 IBM MQ classes for Jakarta Messaging 3.0 는 동일한 인터페이스의 Jakarta Messaging 버전을 지원하며 새 애플리케이션 개발에 권장됩니다.

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging에서 JMS 오브젝트는 다음과 같은 방식으로 IBM MQ 개념과 관련됩니다.

- Connection 오브젝트에는 연결을 작성하는 데 사용한 연결 팩토리의 특성에서 파생된 특성이 있습니다. 이러한 특성은 애플리케이션이 큐 관리자에 연결하는 방식을 제어합니다. 이러한 특성의 예로는 큐 관리자의 이름이 있으며, 클라이언트 모드에서 큐 관리자에 연결하는 애플리케이션의 경우에는 큐 관리자가 실행 중인 시스템의 호스트 이름 또는 IP 주소입니다.
- Session 오브젝트가 IBM MQ 연결 핸들을 캡슐화하므로, 세션의 트랜잭션 범위를 정의합니다.
- MessageProducer 오브젝트와 MessageConsumer 오브젝트는 각각 IBM MQ 오브젝트 핸들을 캡슐화합니다.

IBM MQ classes for JMS 또는 IBM MQ classes for Jakarta Messaging를 사용하는 경우 IBM MQ 의 모든 일반 규칙이 적용됩니다. 특히, 애플리케이션에서 리모트 큐에 메시지를 송신할 수 있지만, 애플리케이션이 연결된 큐 관리자가 소유한 큐에서만 메시지를 수신할 수 있다는 점에 유의하십시오.

JMS 스펙에서는 ConnectionFactory 및 Destination 오브젝트가 관리 대상 오브젝트여야 합니다. 관리자가 중앙 저장소에서 관리 대상 오브젝트를 작성 및 유지보수하며 JMS 애플리케이션은 JNDI(Java Naming and Directory Interface)를 사용하여 이러한 오브젝트를 검색합니다.

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging에서 Destination 인터페이스의 구현은 Queue및 Topic의 추상 수퍼클래스이므로 Destination의 인스턴스는 Queue 오브젝트 또는 Topic 오브젝트입니다. 도메인 독립적 인터페이스에서는 큐나 토픽을 목적지로 처리합니다. MessageProducer 또는 MessageConsumer 오브젝트의 메시징 도메인은 목적지가 큐인지 아니면 토픽인지에 따라 결정됩니다.

따라서 IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 에서 다음 유형의 오브젝트는 관리 오브젝트일 수 있습니다.

- ConnectionFactory
- QueueConnectionFactory
- TopicConnectionFactory
- 큐
- 토픽
- XAConnectionFactory
- XAQueueConnectionFactory
- XATopicConnectionFactory

IBM MQ classes for JMS/Jakarta Messaging 아키텍처

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 에는 계층화된 아키텍처가 있습니다. 코드의 맨 위 계층은 IBM Java 메시징 제공자가 사용할 수 있는 공통 계층입니다.

IBM MQ 9.3.0 에 대한 지원을 소개합니다 Jakarta Messaging 3.0. JMS 2.0 는 여전히 완벽하게 지원됩니다.

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 에는 140 페이지의 그림 54다이어그램에 표시된 대로 계층화된 아키텍처가 있습니다. 코드의 맨 위 계층은 IBM JMS 또는 Jakarta Messaging제공자가 사용할 수 있는 공통 계층입니다. 애플리케이션이 JMS 또는 Jakarta Messaging메소드를 호출할 때 메시징 시스템에 특정하지 않은 호출의 처리는 공통 계층에 의해 수행되며, 이는 호출에 대한 일관된 응답도 제공합니다. 메시징 시스템에 특정적인 호출의 모든 처리는 하위 계층에 위임됩니다. 다음 다이어그램에서 두 개의 추가 메시징 제공자(메시징 제공자 A 및 메시징 제공자 B)와 함께 IBM MQ 메시징 제공자가 하위 계층에 표시됩니다.

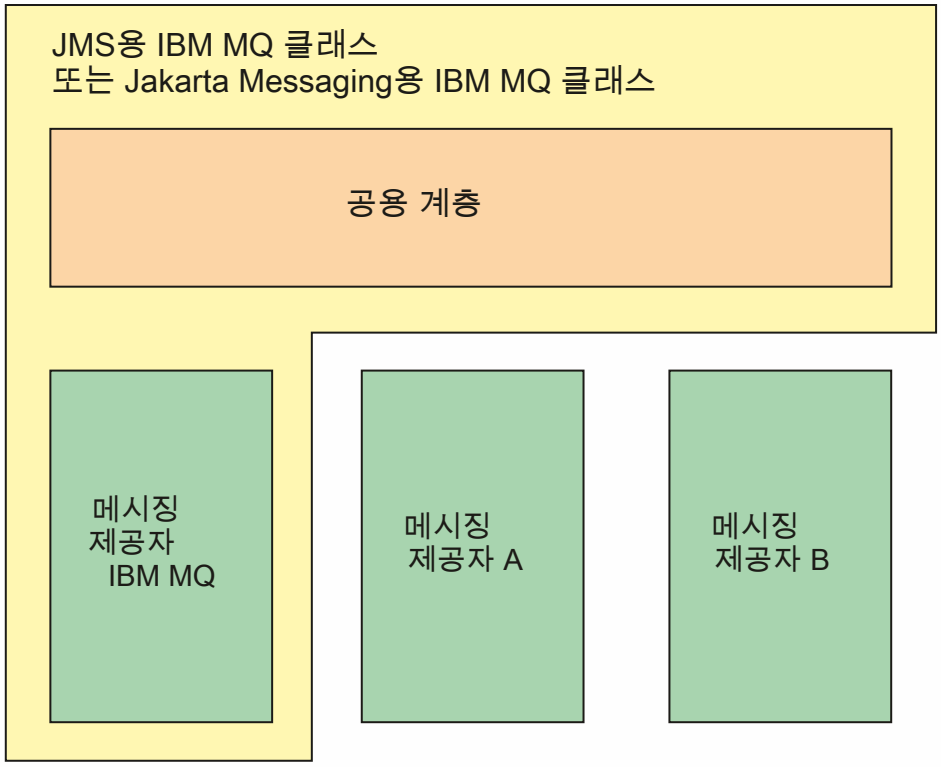


그림 *54. IBM JMS* 및 *Jakarta Messaging* 제공자의 계층화된 아키텍처

계층 아키텍처는 다음 목표를 달성합니다.

- 다양한 IBM JMS 및 Jakarta Messaging 제공자의 동작 일관성을 개선하기 위해
- 두 IBM 메시징 시스템 사이에서 브릿지 애플리케이션을 더 쉽게 작성
- 하나의 IBM JMS 또는 Jakarta Messaging 제공자에서 다른 제공자로 애플리케이션을 더 쉽게 이식하기 위해

관련 태스크

JMS/Jakarta Messaging에 IBM MQ 클래스 사용

관리 오브젝트 지원

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 는 관리 오브젝트의 사용을 지원합니다.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0, 에서 새로운 애플리케이션을 개발할 수 있습니다. Jakarta Messaging 3.0 IBM MQ 9.3.0 기존 애플리케이션에 대한 JMS 2.0 지원을 계속합니다. 동일한 애플리케이션에서 Jakarta Messaging 3.0 API와 JMS 2.0 API를 모두 사용하는 것은 지원되지 않습니다. 자세한 내용은 JMS/자카르타 메시징에 IBM MQ 클래스 사용을 참조하세요.

JMS 또는 IBM MQ classes for Jakarta Messaging 애플리케이션 내의 로직 플로우는 ConnectionFactory 및 Destination 오브젝트로 시작합니다. 애플리케이션은 Connection 오브젝트를 작성하기 위해 ConnectionFactory 오브젝트를 사용하며, 이는 애플리케이션에서 메시징 서버로의 활성 연결을 나타냅니다. 애플리케이션은 Session 오브젝트를 작성하기 위해 Connection 오브젝트를 사용하며, 이는 메시지를 생성하고 사용하기 위한 단일 스레드 컨텍스트입니다. 그런 다음 지정된 목적지로 메시지를 송신하는 데 사용하는 이 애플리케이션은 Session 오브젝트와 Destination 오브젝트를 사용하여 MessageProducer 오브젝트를 작성할 수 있습니다. 목적지는 메시징 시스템의 토픽 또는 큐 중 하나이며, Destination 오브젝트에 의해 캡슐화됩니다. 이 애플리케이션은 또한 Session 오브젝트와 Destination 오브젝트를 사용하여 MessageConsumer 오브젝트를 작성하며, 이 애플리케이션은 지정된 목적지로 송신된 메시지를 수신하기 위해 사용됩니다.

JMS 및 Jakarta Messaging 스펙은 ConnectionFactory 및 Destination 오브젝트가 관리 오브젝트가 될 것으로 예상합니다. 관리자는 중앙 저장소에서 관리 오브젝트를 작성하고 유지보수하며 JMS 또는 Jakarta Messaging 애플리케이션은 Java Naming Directory Interface (JNDI) 를 사용하여 이러한 오브젝트를 검색합니다. 관리 오

브젝트의 저장소는 단순 파일에서 LDAP (Lightweight Directory Access Protocol) 디렉토리까지 다양할 수 있습니다.

IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 는 관리 오브젝트의 사용을 지원합니다. 애플리케이션은 IBM MQ특정 정보를 애플리케이션 자체에 하드 코딩하지 않고 IBM MQ 를 통해 노출되는 IBM MQ classes for JMS 또는 IBM MQ classes for Jakarta Messaging 의 모든 기능을 사용할 수 있습니다. 이러한 배열은 애플리케이션이 기반의 IBM MQ 구성에서 상당히 독립적이게 합니다.

이러한 독립성을 얻기 위해 애플리케이션은 JNDI를 사용하여 관리 오브젝트로 저장된 연결 팩토리 및 대상을 검색하고 `javax.jms` (JMS 2.0) 또는 `jakarta.jms` (Jakarta Messaging 3.0) 패키지에 정의된 인터페이스만 사용하여 메시징 조작을 수행할 수 있습니다.

JMS 2.0 ' JMS 2.0'의 경우 관리자는 ' IBM MQ ' JMS ' 관리 도구 ' **JMSAdmin**' 또는 ' IBM MQ Explorer'를 사용하여 중앙 리포지토리에서 관리되는 개체를 생성하고 유지 관리할 수 있습니다.

JM 3.0 Jakarta Messaging 3.0 의 경우 을 사용하여 JNDI를 관리할 수 없습니다. IBM MQ Explorer JNDI 관리는 **JMSAdmin** 의 Jakarta Messaging 3.0 변형인 **JMS30Admin** 에서 지원됩니다.

Application Server는 일반적으로 관리 오브젝트에 대한 자체 저장소 및 오브젝트를 작성하고 유지보수하기 위한 자체 도구를 제공합니다. 따라서 Java EE JM 3.0 또는 Jakarta EE 애플리케이션은 JNDI 를 사용하여 애플리케이션 서버 저장소 또는 중앙 저장소에서 관리 오브젝트를 검색할 수 있습니다.

관련 태스크

JMS및 Jakarta Messaging 자원 구성

Java EE 및 Jakarta EE 플랫폼에서 지원되는 통신 유형

Java EE 및 Jakarta EE 플랫폼에서 IBM MQ classes for JMS 및 IBM MQ classes for Jakarta Messaging 는 애플리케이션의 컴포넌트와 IBM MQ 큐 관리자 간에 두 가지 유형의 통신을 지원합니다.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 에 대한 지원을 소개합니다 Jakarta Messaging 3.0. JMS 2.0 는 여전히 완벽하게 지원됩니다. JMS 와 은 공통점이 많으므로 이 주제에서 을 참조하는 것은 두 가지를 모두 참조하는 것으로 간주할 수 있습니다. Jakarta Messaging JMS 필요에 따라 차이점이 강조 표시됩니다.

다음 두 가지 유형의 통신이 애플리케이션의 컴포넌트와 IBM MQ 큐 관리자 사이에서 지원됩니다.

- 아웃바운드 통신
- 인바운드 통신

아웃바운드 통신

애플리케이션 컴포넌트는 JMS 또는 Jakarta Messaging API를 직접 사용하여 큐 관리자에 대한 연결을 작성한 후 메시지를 송수신합니다.

예를 들어, 애플리케이션 컴포넌트는 애플리케이션 클라이언트, 서블릿, JSP(JavaServer Page), Enterprise Java Bean(EJB) 또는 MDB(Message Driven Bean)일 수 있습니다. 이러한 유형의 통신에서 애플리케이션 서버 컨테이너는 메시징 조작(연결 풀링 및 스레드 관리 등)의 지원에서 낮은 레벨의 기능을 제공합니다.

인바운드 통신

인바운드 통신의 경우 목적지에 도착하는 메시지는 MDB로 전달되고 MDB가 메시지를 처리합니다.

Java EE JM 3.0 및 Jakarta EE 애플리케이션은 MDB를 사용하여 메시지를 비동기식으로 처리합니다. MDB는 JMS 메시지 리스너로서 작동하고 onMessage() 메소드에 의해 구현됩니다. 이 메소드는 메시지가 처리되는 방법을 정의합니다. MDB는 어플리케이션 서버의 EJB 컨테이너에 배치됩니다. MDB가 구성되는 정확한 방법은 사용 중인 애플리케이션 서버에 따라 결정되지만 구성 정보는 연결할 큐 관리자, 큐 관리자에 연결하는 방법, 메시지를 모니터할 목적지, MDB의 트랜잭션 작동을 지정해야 합니다. 그러면 이 정보가 EJB 컨테이너에 의해 사용됩니다. MDB의 선택 기준을 충족하는 메시지가 지정된 대상에 도착하면 EJB 컨테이너는 IBM MQ classes for JMS 또는 IBM MQ classes for Jakarta Messaging 를 사용하여 큐 관리자에서 메시지를 검색한 후 onMessage() 메소드를 호출하여 메시지를 MDB에 전달합니다.

IBM MQ classes for Java와의 관계

IBM MQ classes for Java, IBM MQ classes for Jakarta Messaging 및 IBM MQ classes for JMS 는 MQI에 대한 공용 Java 인터페이스를 사용하는 피어입니다.

142 페이지의 그림 55 에서는 IBM MQ classes for JMS, IBM MQ classes for Jakarta Messaging 및 IBM MQ classes for Java간의 관계를 보여줍니다.

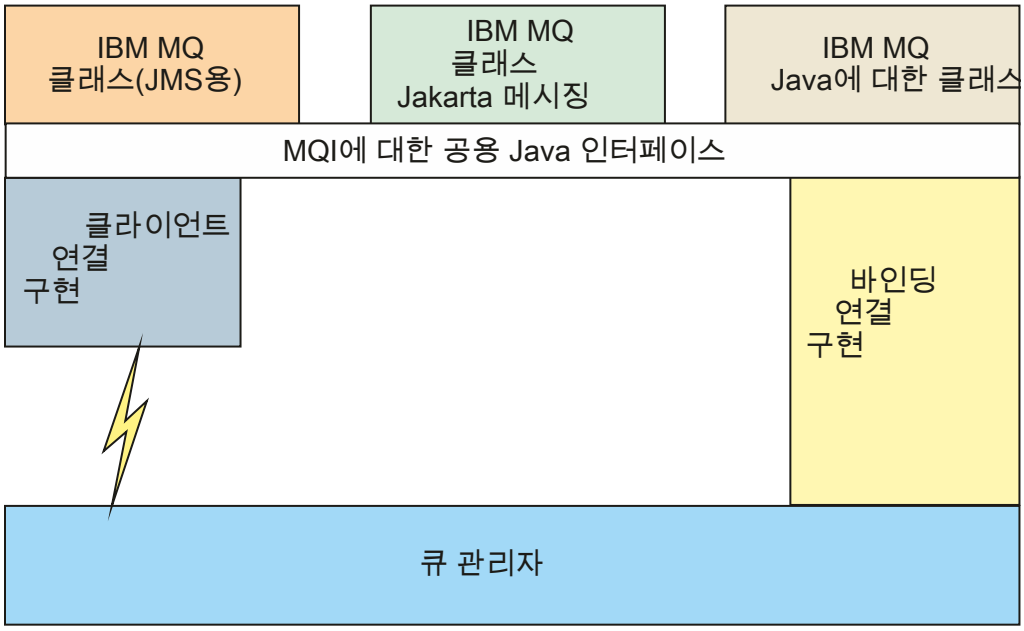


그림 *55. IBM MQ classes for JMS, IBM MQ classes for Jakarta Messaging* 및 *IBM MQ classes for Java* 간의 관계

일반적으로 Java 프로그램은 IBM MQ - IBM MQ classes for Java, IBM MQ classes for Jakarta Messaging 또는 IBM MQ classes for JMS와 인터페이스하기 위해 하나의 인터페이스만 사용해야 합니다. 인터페이스 혼합은 한 가지 예외를 제외하고 지원되지 않습니다. IBM WebSphere MQ 7.0이전 릴리스와의 호환성을 유지하기 위해, 채널 엑시트 클래스가 IBM MQ classes for JMS에서 호출되는 경우에도 Java 로 작성된 채널 엑시트 클래스는 여전히 IBM MQ classes for Java 인터페이스를 사용할 수 있습니다. 그러나 IBM MQ classes for Java 인터페이스를 사용하는 것은 애플리케이션이 여전히 다음 중 하나에 종속됨을 의미합니다.

- JMS 2.0 IBM MQ classes for Java JAR 파일, `com.ibm.mq.jar`. 사용자 클래스 경로에서 `com.ibm.mq.jar`를 원하지 않을 경우, 대신 `com.ibm.mq.exits` 패키지의 인터페이스 세트를 사용할 수 있습니다.
- V 9.3.0 V 9.3.0 JM 3.0 IBM MQ classes for Jakarta Messaging와 상호 운용할 때 `com.ibm.mq.jakarta.client.jar`를 사용합니다.

관련 개념

V 9.3.0 V 9.3.0 Jakarta Messaging에서 IBM MQ 클래스를 사용해야 하는 이유는 무엇입니까?
JMS용 IBM MQ 클래스를 사용해야 하는 이유는 무엇입니까?
Java용 IBM MQ 클래스를 사용해야 하는 이유는 무엇입니까?

IBM MQ 메시징 제공자

IBM MQ 메시징 제공자에는 3개의 조작 모드 즉, 정상 모드, 제한적 정상 모드, 마이그레이션 모드가 있습니다.

IBM MQ 메시징 제공자에는 3개의 조작 모드가 있습니다.

- IBM MQ 메시징 제공자 정상 모드
- 제한이 있는 IBM MQ 메시징 제공자 정상 모드
- IBM MQ 메시징 제공자 마이그레이션 모드

IBM MQ 메시징 제공자 정상 모드는 IBM MQ 큐 관리자의 모든 기능을 사용하여 JMS를 구현합니다. 이 모드는 JMS 2.0 V 9.3.0 V 9.3.0 JM 3.0 또는 Jakarta Messaging 3.0 API및 기능을 사용하도록 최적화되어 있습니다.

다음과 같은 경우

- 클라이언트는 **ConnectionFactory**에서 제공자 버전 6을 지정하며, 클라이언트는 IBM WebSphere MQ 6.0에서 제공되는 클라이언트와 호환 가능한 방식으로 작동합니다. JMS 1.1 및 JMS 2인터페이스만 지원되지만 일부 JMS 2기능 (예: 공유 구독, 전달 지연 및 비동기 송신) 은 사용 불가능합니다. 연결 공유가 없습니다.
- 클라이언트는 **ConnectionFactory**에서 제공자 버전 7을 지정합니다. JMS 1.1 및 JMS 2인터페이스 모두 완전히 지원됩니다.
- 제공자 버전이 지정되지 않았습니다. 제공자 버전 7과 연결하려고 시도합니다. 이에 실패하면 제공자 버전 6으로 추가 시도가 이루어집니다.

IBM MQ Enterprise Transport를 사용하여 IBM Integration Bus 에 연결하려면 마이그레이션 모드를 사용하십시오. IBM MQ Real-Time Transport를 사용하는 경우에는 연결 팩토리 오브젝트에서 명시적으로 특성을 선택했기 때문에 마이그레이션 모드가 자동으로 선택됩니다. IBM MQ Enterprise Transport를 사용하는 IBM Integration Bus 에 대한 연결은 JMS **PROVIDERVERSION** 특성 구성에 설명된 모드 선택에 대한 일반 규칙을 따릅니다.

관련 태스크

JMS 자원 구성

z/OS IBM MQ for z/OS 개념

IBM MQ for z/OS에서 사용하는 일부 개념은 z/OS 플랫폼에 고유합니다. 예를 들어, 로깅 메커니즘, 스토리지 관리 기술, 복구 단위 처리 및 큐 공유 그룹은 IBM MQ for z/OS에서만 제공됩니다. 이 주제를 이러한 개념에 대한 추가 정보에 대한 소개로 사용하십시오.

개념에는 다음을 포함하여 IBM MQ for z/OS에서 사용하는 오브젝트의 개요가 포함되어 있습니다.

- 큐 관리자
- 채널 시작기
- 공유 큐 및 큐 공유 그룹
- 그룹 내 큐잉

다음 주제에서는 다음을 포함하여 필요한 다양한 프로시저에 대해서도 다룹니다.

- z/OS 의 시스템 정의
- 스토리지 관리
- 복구 및 재시작
- IBM MQ for z/OS의 보안 개념

관련 개념

144 페이지의 『z/OS의 큐 관리자』
애플리케이션 프로그램이 z/OS 시스템에서 IBM MQ 를 사용하도록 하려면 IBM MQ for z/OS 제품을 설치하고 큐 관리자를 시작해야 합니다. 큐 관리자는 IBM MQ에서 사용하는 자원의 세트를 소유 및 관리합니다.

145 페이지의 『z/OS의 채널 시작기』
채널 시작기는 IBM MQ 분산 큐잉을 사용 가능하게 하는 자원을 제공하고 관리합니다. IBM MQ는 *MCA(Message Channel Agents)*를 사용하여 한 큐 관리자에서 다른 큐 관리자로 송신합니다.

147 페이지의 『IBM MQ for z/OS 관리에 대한 용어 및 태스크』
이 토픽을 IBM MQ for z/OS에 특정한 용어 및 태스크에 대한 소개로 사용하십시오.

149 페이지의 『공유 큐 및 큐 공유 그룹』
고가용성의 IBM MQ 자원을 구현하기 위해 공유 큐 및 큐 공유 그룹을 사용할 수 있습니다. 공유 큐 및 큐 공유 그룹은 z/OS 플랫폼의 IBM MQ for z/OS에 고유한 기능입니다.

190 페이지의 『그룹 내 큐잉』

이 절에서는 z/OS 플랫폼에 고유한 IBM MQ for z/OS 함수인 그룹 내 큐잉에 대해 설명합니다. 이 기능은 큐 공유 그룹에 정의된 큐 관리자만이 사용할 수 있습니다.

201 페이지의 『z/OS의 스토리지 관리』
IBM MQ for z/OS에서는 영구 및 임시 데이터 구조가 필요하며 이 데이터를 저장하기 위해 페이지 세트 및 메모리 버퍼를 사용합니다. 이 주제에서는 IBM MQ에서 이러한 페이지 세트 및 버퍼를 사용하는 방법에 대한 자세한 정보를 제공합니다.

206 페이지의 『IBM MQ for z/OS에서 로깅』
IBM MQ는 발생하는 데이터 변경사항 및 중요한 이벤트의 로그를 유지보수합니다. 이러한 로그는 필요한 경우 데이터를 이전 상태로 복구하는 데 사용할 수 있습니다.

225 페이지의 『z/OS에서의 복구 및 재시작』
이 토픽의 링크를 사용하여 재시작 및 복구를 위해 IBM MQ for z/OS의 기능에 대해 확인하십시오.

240 페이지의 『IBM MQ for z/OS의 보안 개념』
IBM MQ에서 보안의 중요성과 시스템에서 적절한 보안 설정이 되지 않은 경우 의미를 이해하는 데 이 토픽을 사용하십시오.

245 페이지의 『z/OS의 가용성』
IBM MQ for z/OS에는 고가용성을 위한 많은 기능이 있습니다. 이 주제에서는 가용성에 대한 몇 가지 고려사항을 설명합니다.

249 페이지의 『z/OS에서 복구 단위 지정』
특정 트랜잭션 애플리케이션은 큐 관리자 이름 대신 연결 시 QSG 이름을 지정하여 큐 공유 그룹(QSG)에서 큐 관리자에 연결할 때 GROUP 복구 단위 속성 지정(QMGR이 아님)을 사용할 수 있습니다. 이를 통해 트랜잭션 복구는 QSG의 동일한 큐 관리자에 다시 연결하는 데 필요한 요구사항을 제거하여 보다 유연하고 강력해질 수 있습니다.

관련 참조

215 페이지의 『z/OS 의 시스템 정의』
IBM MQ for z/OS는 많은 기본 오브젝트 정의를 사용하고 해당 기본 오브젝트를 작성하도록 샘플 JCL을 제공합니다. 이 토픽을 사용하여 이러한 기본 오브젝트 및 샘플 JCL을 이해하십시오.

248 페이지의 『IBM MQ for z/OS에 대한 모니터링 및 통계』
IBM MQ for z/OS에는 큐 관리자를 모니터링하고 통계를 수집하는 기능 세트가 있습니다.

z/OS z/OS의 큐 관리자

애플리케이션 프로그램이 z/OS 시스템에서 IBM MQ 를 사용하도록 하려면 IBM MQ for z/OS 제품을 설치하고 큐 관리자를 시작해야 합니다. 큐 관리자는 IBM MQ에서 사용하는 자원의 세트를 소유 및 관리합니다.

큐 관리자

큐 관리자는 애플리케이션에 메시징 서비스를 제공하는 프로그램입니다. MQI(Message Queue Interface)를 사용하는 애플리케이션은 큐에 메시지를 넣고 큐에서 메시지를 가져올 수 있습니다. 큐 관리자는 메시지가 올바른 큐에 송신되었는지 또는 다른 큐 관리자에 라우트되는지 확인합니다. 큐 관리자는 이에 발행된 MQI 호출과 이에 제출된 명령(소스에 상관없음) 모두를 처리합니다. 큐 관리자는 각 호출 또는 명령에 대해 적절한 완료 코드를 생성합니다.

큐 관리자에서 관리하는 자원은 다음을 포함합니다.

- IBM MQ 오브젝트 정의 및 메시지 데이터를 보유하는 페이지 세트
- 큐 관리자 실패 시 메시지 및 오브젝트를 복구하는 데 사용되는 로그
- 프로세서 스토리지
- 다른 애플리케이션 환경(CICS, IMS 및 배치)에서 IBM MQ API에 액세스할 수 있는 연결
- z/OS 시스템의 IBM MQ 와 기타 시스템 간의 통신을 허용하는 IBM MQ 채널 시작기

큐 관리자에는 이름이 있고 애플리케이션은 이 이름을 사용하여 이에 연결할 수 있습니다.

145 페이지의 그림 56에서는 다른 애플리케이션 환경 및 채널 시작기에 대한 연결을 표시하는 큐 관리자를 설명합니다.

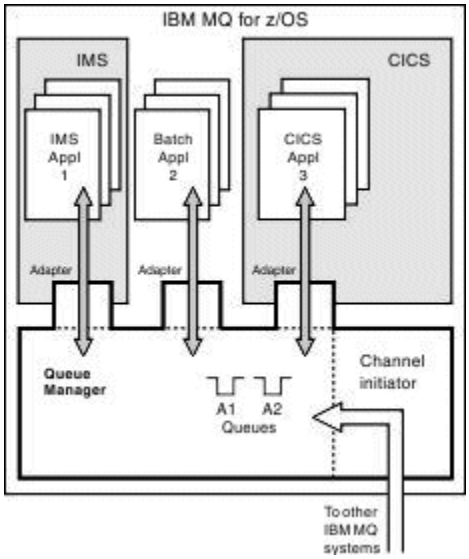


그림 *56. IBM MQ for z/OS* 개요

z/OS의 큐 관리자 서브시스템

z/OS에서 IBM MQ는 IPL 시점에 시작된 z/OS 서브시스템으로 실행됩니다. 서브시스템에서 큐 관리자는 로그에 대한 정보를 포함하고 오브젝트 정의 및 메시지 데이터(페이지 세트)를 보유하는 z/OS 데이터 세트를 지정하는 JCL 프로시저를 실행하여 시작됩니다. 서브시스템 및 큐 관리자는 최대 4자의 동일한 이름을 보유합니다. 다른 시스템, sysplex 또는 플랫폼에 있어도 네트워크의 모든 큐 관리자에서 이름은 고유해야 합니다.

z/OS의 채널 시작기

채널 시작기는 IBM MQ 분산 큐잉을 사용 가능하게 하는 자원을 제공하고 관리합니다. IBM MQ는 *MCA(Message Channel Agents)*를 사용하여 한 큐 관리자에서 다른 큐 관리자로 송신합니다.

큐 관리자 A에서 큐 관리자 B로 메시지를 송신하려면 큐 관리자 A의 송신 MCA에서 큐 관리자 B에 대한 통신 링크를 설정해야 합니다. 통신 링크에서 메시지를 수신하려면 수신 MCA가 큐 관리자 B에서 시작되어야 합니다. 송신 MCA, 통신 링크 및 수신 MCA로 구성된 이 단방향 경로를 채널이라고 합니다. 송신 MCA는 전송 큐로부터 메시지를 가져와 채널을 통해 수신 MCA로 송신합니다. 수신 MCA는 메시지를 수신하고 목적지 큐에 넣습니다.

IBM MQ for z/OS에서 송신 및 수신 MCA는 모두 채널 시작기 안에서 실행됩니다(채널 시작기는 이동기라고도 함). 채널 시작기는 큐 관리자 제어 아래에서 z/OS 주소 공간으로 실행됩니다. 큐 관리자에 연결된 채널 시작기는 하나만 존재할 수 있으며 이는 동일한 z/OS 이미지에서 큐 관리자로 실행됩니다. 채널 시작기 내부에서 수천 개의 MCA 프로세스가 동시에 실행 중일 수 있습니다.

146 페이지의 그림 57에서는 SYSPLEX에서 2개의 큐 관리자를 표시합니다. 각 큐 관리자에는 채널 시작기와 로컬 큐가 있습니다. AIX 및 Windows에서 큐 관리자가 보낸 메시지는 로컬 큐에 배치되며, 여기서 애플리케이션이 메시지를 검색합니다. 응답 메시지는 유사한 라우트로 리턴됩니다.

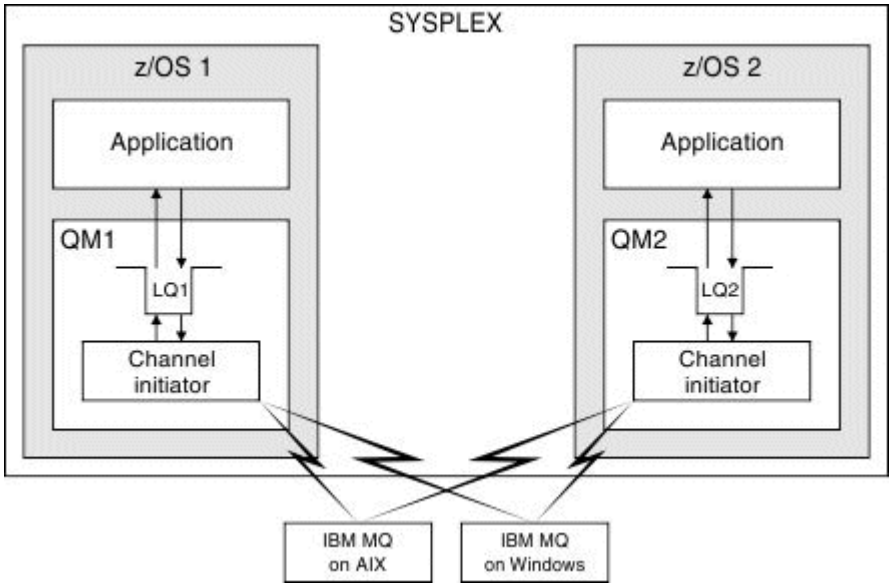


그림 57. 큐 관리자 간 통신

채널 시작기는 채널 관리와 관련된 다른 프로세스도 포함합니다. 다음 프로세스가 포함됩니다.

리스너
이러한 프로세스는 TCP와 같은 통신 서브시스템에서 인바운드 채널 요청을 대기하며, 인바운드 요청이 수신되면 이름 지정된 MCA를 시작합니다.

감독자
여기에서는 채널 시작기 주소 공간을 관리합니다. 예를 들어 실패 후 채널 시작을 담당합니다.

이름 서버
이는 TCP 이름을 주소로 해석하는 데 사용됩니다.

TLS 태스크
이는 암호화 및 복호화를 수행하고 인증서 폐기 목록(CRL)을 확인하는 데 사용됩니다.

z/OS 채널 시작기의 SMF 레코드

채널 시작기(CHINIT)는 태스크와 채널에 대한 정보가 있는 SMF 통계 레코드와 회계 레코드를 생성할 수 있습니다.

CHINIT는 다음 유형의 정보가 있는 SMF 통계 레코드와 회계 레코드를 생성할 수 있습니다.

- 태스크는 디스패처, 어댑터, 도메인 이름 서버(DNS) 및 SSL입니다. 이러한 태스크는 CHINIT라는 통계를 구성합니다.
- 채널은 DIS CHSTATUS 명령으로 사용 가능한 정보와 유사한 회계 정보를 제공합니다. 이 정보는 채널 회계라고 합니다.

IBM MQ for Multiplatforms 는 PCF 메시지를 SYSTEM.ADMIN.STATISTICS.QUEUE입니다. 통계 정보가 IBM MQ for Multiplatforms에 기록되는 방법에 대한 자세한 정보는 채널 통계 메시지 데이터 를 참조하십시오.

통계 데이터

이 정보를 사용하여 다음 정보를 찾을 수 있습니다.

- SSL TCB의 수 및 이러한 태스크에서 사용하는 CPU 크기 등의 추가 CHINIT 태스크가 필요한지 여부.
- 이러한 태스크에 대한 요청의 평균 시간.
- DNS 및 SSL 태스크에 대해 기간이 가장 긴 요청 및 이 요청이 발생한 시간. 이 시간과 채널에 발행한 문제를 서로 연관시킬 수 있습니다.

계정 데이터

이 정보를 사용하여 채널 사용을 모니터링하고 다음 정보를 알아낼 수 있습니다.

- 처리량이 가장 높은 채널.
- 메시지 송신율 및 송신된 데이터 비율(MB/초).
- 아카이브된 배치 크기. 아카이브된 배치 크기가 채널에 지정된 배치 크기에 가까운 경우 채널이 메시지 송신 한계에 근접했을 수 있습니다.

START TRACE 및 STOP TRACE 명령을 사용하여 회계 추적 및 통계 추적의 콜렉션을 제어합니다. 채널과 큐 관리자에서 STATCHL 및 STATACLS 옵션을 사용하여 채널의 SMF 데이터 생성 여부를 제어할 수 있습니다.

z/OS IBM MQ for z/OS 관리에 대한 용어 및 태스크

이 토픽을 IBM MQ for z/OS에 특정한 용어 및 태스크에 대한 소개로 사용하십시오.

IBM MQ for z/OS 관리에 필요한 몇몇 용어 및 태스크는 z/OS 플랫폼에 특정적입니다. 다음 목록에는 이러한 몇몇 용어 및 태스크가 들어 있습니다.

- 공유 큐
- 페이지 세트 및 버퍼 풀
- 로깅
- 큐 관리자 환경 조정
- 재시작 및 복구
- 보안
- 가용성
- 오브젝트 조작
- 모니터링 및 통계
- 애플리케이션 환경

공유 큐

큐는 단 하나의 큐 관리자만이 소유하고 액세스할 수 있는 비공유이거나 큐 공유 그룹이 소유하는 공유일 수 있습니다. 큐 공유 그룹은 동일한 IBM MQ 오브젝트 정의 및 메시지 데이터에 동시에 액세스할 수 있는 단일 z/OS sysplex내에서 실행되는 여러 큐 관리자로 구성됩니다. 큐 공유 그룹 내에서 공유 가능한 오브젝트 정의는 공유 Db2 데이터베이스에 저장되어 있습니다. 공유 큐 메시지는 하나 이상의 커플링 기능 구조(CF 구조) 내부에 보관됩니다. 구조 안에 직접 저장하기에는 메시지 데이터가 너무 크거나(크기가 63KB 초과) 설치 정의된 규칙이 오프로딩을 위해 선택할 만큼 메시지가 큰 경우, 메시지 제어 정보는 계속 커플링 기능 항목에 저장되지만 메시지 데이터는 공유 메시지 데이터 세트(SMDS) 또는 공유 Db2 데이터베이스로 오프로드됩니다. 공유 메시지 데이터 세트, 공유 Db2 데이터베이스 및 커플링 기능 구조는 그룹의 모든 큐 관리자가 합동 관리하는 자원입니다.

페이지 세트 및 버퍼 풀

메시지를 비공유 큐에 넣으면 큐 관리자는 후속 조작이 동일한 큐에서 메시지를 가져오는 경우 검색할 수 있도록 페이지 세트에 데이터를 저장합니다. 메시지가 큐에서 제거되면 데이터를 보유하는 페이지 세트의 공간이 나중에 재사용을 위해 비워집니다. 큐에 보유된 메시지 수가 늘어나면 페이지 세트에 사용되는 공간 크기가 늘어나고, 큐의 메시지 수가 줄어들면 페이지 세트에 사용된 공간이 줄어듭니다.

페이지 세트에 데이터를 쓰고, 페이지 세트에서 데이터를 읽는 경우 성능 비용을 줄이기 위해 큐 관리자는 프로세서 스토리지로 업데이트를 버퍼링합니다. 페이지 세트 액세스를 버퍼링하는 데 사용되는 스토리지의 양은 버퍼 풀이라는 IBM MQ 오브젝트를 통해 제어됩니다.

페이지 세트 및 버퍼 풀에 대한 자세한 정보는 스토리지 관리를 참조하십시오.

로그 기록

페이지 세트에 보유된 오브젝트에 대한 모든 변경과 지속 메시지에 대한 조작은 로그 레코드로 기록됩니다. 이러한 로그 레코드는 활성 로그라고 하는 로그 데이터 세트에 기록됩니다. 활성 로그 데이터 세트의 이름과 크기는 *BSDS(Bootstrap Data Set)*라고 하는 데이터 세트에 보유됩니다.

활성 로그 데이터 세트가 채워지면 로깅을 계속할 수 있도록 큐 관리자는 다른 로그 데이터 세트로 전환되며 전체 활성 로그 데이터 세트의 컨텐츠를 아카이브 로그 데이터 세트에 복사합니다. 아카이브 로그 데이터 세트 이름을 포함하여 이러한 조치에 대한 정보는 BSDS에 보유됩니다. 개념적으로 큐 관리자가 순환하는 활성 로그 데이터 세트의 고리가 있습니다. 활성 로그가 채워지면 로그 데이터는 아카이브 로그로 오프로드되고 활성 로그 데이터 세트를 재사용할 수 있습니다.

로그 및 BSDS에 대한 자세한 정보는 206 페이지의 『IBM MQ for z/OS에서 로깅』의 내용을 참조하십시오.

큐 관리자 환경 조정

큐 관리자가 시작되면 큐 관리자의 작동 방식을 제어하는 초기화 매개변수 세트를 읽습니다. 또한 IBM MQ 명령을 포함하는 데이터 세트를 읽고 이들이 포함하는 명령이 실행됩니다. 일반적으로 이 데이터 세트는 IBM MQ를 실행하는 데 필요한 시스템 오브젝트의 정의를 포함하며 이를 조정하여 운영 환경에 필요한 IBM MQ 오브젝트를 정의 또는 초기화할 수 있습니다. 이러한 데이터 세트를 읽으면 이들에서 정의하는 오브젝트가 Db2 또는 페이지 세트에 저장됩니다.

초기화 매개변수 및 시스템 오브젝트에 대한 자세한 정보는 215 페이지의 『z/OS 의 시스템 정의』의 내용을 참조하십시오.

복구 및 재시작

IBM MQ 조작 중 언제라도 아직 페이지 세트에 쓰지 않은 프로세서 스토리지에 보유된 변경이 있을 수 있습니다. 이러한 변경은 큐 관리자 내 백그라운드 태스크에서 최근 사용된 페이지 세트에 기록됩니다.

큐 관리자가 비정상적으로 종료되면 지속 메시지 데이터가 로그 레코드에 기록되므로 큐 관리자 재시작의 복구 단계는 유실된 페이지 세트 변경을 복구할 수 있습니다. 즉, IBM MQ는 실패 지점에서 바로 지속 메시지 데이터 및 오브젝트 변경을 복구할 수 있습니다.

큐 공유 그룹의 멤버인 큐 관리자에 커플링 기능 장애가 발생한 경우, 해당 큐의 지속 메시지는 커플링 기능 구조가 백업된 경우에만 복구될 수 있습니다.

복구 및 재시작에 대한 자세한 정보는 225 페이지의 『z/OS에서의 복구 및 재시작』의 내용을 참조하십시오.

보안

보안 서버(이전에는 RACF라 함)와 같은 외부 보안 관리자를 사용하여 IBM MQ가 소유하고 관리하는 자원을 비인가 사용자의 액세스로부터 보호할 수 있습니다. 또한 채널 보안을 위해 TLS(Transport Layer Security)를 사용할 수도 있습니다. TLS는 IBM MQ 제품의 일부로 포함됩니다.

IBM MQ 보안에 대한 자세한 정보는 240 페이지의 『IBM MQ for z/OS의 보안 개념』의 내용을 참조하십시오.

가용성

큐 관리자 또는 통신 서브시스템 실패 시 시스템 가용성을 늘리도록 고안된 IBM MQ의 여러 기능이 있습니다. 이 기능에 대한 자세한 정보는 245 페이지의 『z/OS의 가용성』의 내용을 참조하십시오.

오브젝트 조작

큐 관리자가 실행 중인 경우 z/OS 콘솔 인터페이스 또는 TSO에서 ISPF 서비스를 사용하는 관리 유틸리티를 통해 IBM MQ 오브젝트를 조작할 수 있습니다. 두 메커니즘 모두 IBM MQ 오브젝트를 정의, 대체 또는 삭제할 수 있습니다. 또한 다양한 IBM MQ 및 큐 관리자 기능 상태를 제어 및 표시할 수 있습니다.

이러한 기능에 대한 자세한 정보는 IBM MQ for z/OS에서 MQSC및 PCF 명령을 실행할 수 있는 소스를 참조하십시오.

또한 큐, 큐 관리자 및 기타 오브젝트 관련 작업에 대해 비주얼 방식을 제공하는 그래픽 사용자 인터페이스인 IBM MQ 탐색기를 사용하여 IBM MQ 오브젝트를 조작할 수도 있습니다.

모니터링 및 통계

큐 관리자와 채널 시작기를 모니터하는 데 사용할 수 있는 여러 기능이 있습니다. 또한 성능 평가 및 회계 용도로 통계를 수집할 수도 있습니다.

이 기능에 대한 자세한 정보는 248 페이지의 『IBM MQ for z/OS에 대한 모니터링 및 통계』의 내용을 참조하십시오.

애플리케이션 환경

큐 관리자가 시작되면 애플리케이션은 이에 연결하고 IBM MQ API를 사용하여 시작할 수 있습니다. 이는 CICS, IMS, 배치 또는 WebSphere Application Server 애플리케이션일 수 있습니다. IBM MQ 애플리케이션은 CICS 및 IMS 브릿지를 사용하여 IBM MQ를 인식하지 않는 CICS 및 IMS 시스템의 애플리케이션에 액세스할 수도 있습니다.

이 기능에 대한 자세한 정보는 251 페이지의 『IBM MQ 및 기타 z/OS 제품』의 내용을 참조하십시오.

IBM MQ 애플리케이션 작성에 대한 정보는 다음 문서를 참조하십시오.

- 애플리케이션 개발
- C++ 사용
- IBM MQ classes for Java 사용

z/OS 공유 큐 및 큐 공유 그룹

고가용성의 IBM MQ 자원을 구현하기 위해 공유 큐 및 큐 공유 그룹을 사용할 수 있습니다. 공유 큐 및 큐 공유 그룹은 z/OS 플랫폼의 IBM MQ for z/OS에 고유한 기능입니다.

이 절에서는 속성 및 혜택을 설명하며, 여러 큐 관리자가 동일한 큐 및 해당 큐의 메시지를 공유할 수 있는 방법에 대한 정보를 제공합니다.

공유 큐의 개념

공유 큐는 로컬 큐의 유형입니다. 해당 큐의 메시지를 SYSPLEX에 있는 하나 이상의 큐 관리자가 액세스할 수 있습니다.

큐 공유 그룹

동일한 공유 큐 세트에 액세스할 수 있는 큐 관리자는 큐 공유 그룹이라고 하는 그룹을 형성합니다.

큐 관리자의 메시지 액세스 가능

큐 공유 그룹의 큐 관리자는 공유 큐에 액세스할 수 있습니다. 이는 메시지를 하나의 큐 관리자의 공유 큐에 넣을 수 있으며 동일한 메시지를 다른 큐 관리자의 큐에서 가져올 수 있음을 의미합니다. 이는 큐 관리자 간의 채널 활성화가 필요하지 않는 큐 공유 그룹 내의 빠른 통신 메커니즘을 제공합니다.

IBM MQ 는 Db2 또는 공유 메시지 데이터 세트 (SMDS) 로의 메시지 오프로드를 지원합니다. 어떤 크기의 메시지의 오프-로딩도 구성할 수 있습니다.

150 페이지의 그림 58은 큐 공유 그룹을 구성하는 세 개의 큐 관리자와 커플링 기능을 보여줍니다. 세 개의 큐 관리자는 모두 커플링 기능의 공유 큐에 액세스할 수 있습니다.

애플리케이션은 큐 공유 그룹 내의 임의의 큐 관리자에 연결할 수 있습니다. 큐 공유 그룹의 모든 큐 관리자가 모든 공유 큐에 액세스할 수 있으므로, 애플리케이션은 특정 큐 관리자의 가용성에 의존하지 않으며 큐 공유 그룹의 큐 관리자는 큐를 서비스할 수 있습니다.

큐 관리자 중 하나에 문제점이 있으면 큐 공유 그룹의 다른 모든 큐 관리자가 계속해서 큐를 처리할 수 있으므로, 이는 더 많은 가용성을 제공합니다.

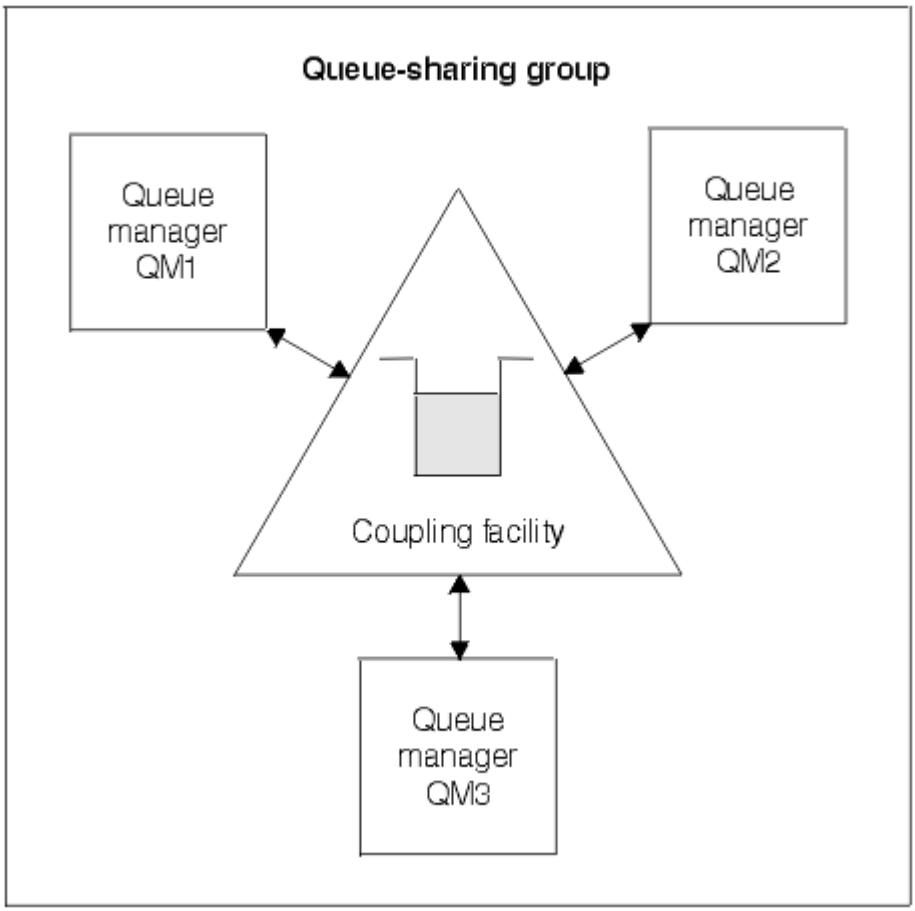


그림 *58.* 큐 공유 그룹

모든 큐 관리자가 공유하는 큐 정의

공유 큐 정의는 Db2 데이터베이스 테이블 OBJ_B_QUEUE에 저장되어 있습니다. 이로 인해 큐를 한 번만 정의해야 하며, 이후에 이는 큐 공유 그룹의 모든 큐 관리자에 의해 액세스가 가능합니다. 이는 정의할 항목이 더 적어짐을 의미합니다.

이와는 대조적으로, 비공유 큐의 정의는 큐를 소유하는 큐 관리자의 페이지 세트 영(0)에 저장됩니다(페이지 세트의 설명 참조).

해당 이름의 큐가 정의하는 큐 관리자의 페이지 세트에 이미 정의되어 있으면 공유 큐를 정의할 수 없습니다. 이와 마찬가지로, 동일한 이름의 공유 큐가 존재하면 큐 관리자 페이지 세트에 큐의 로컬 버전을 정의할 수 없습니다.

큐 공유 그룹의 개념

동일한 공유 큐에 액세스할 수 있는 큐 관리자의 그룹을 큐 공유 그룹이라고 합니다. 큐 공유 그룹의 각 멤버에게는 동일한 세트의 공유 큐에 대한 액세스 권한이 있습니다.

큐 공유 그룹은 최대 4자의 이름을 가집니다. 이 이름은 네트워크에서 고유해야 하며 큐 관리자 이름과 달라야 합니다.

151 페이지의 그림 59은 두 개의 큐 관리자가 포함된 큐 공유 그룹을 보여줍니다. 각각의 큐 관리자에는 채널 시작기 및 자체 로컬 페이지 세트와 로그 데이터 세트가 있습니다.

큐 공유 그룹의 각 멤버는 Db2 시스템에도 연결해야 합니다. 공유 오브젝트 정의를 보관하기 위해 사용되는 Db2 공유 저장소에 큐 관리자가 액세스할 수 있도록 Db2 시스템은 모두 동일한 Db2 데이터 공유 그룹에 있어야 합니다. 이는 한 번만 정의한 후에 그룹의 큐 관리자가 사용할 수 있는 IBM MQ 오브젝트의 유형에 대한 정의입니다 (예: 큐 및 채널). 이를 글로벌 정의라고 하며 개인용 및 글로벌 정의에 설명되어 있습니다.

둘 이상의 큐 공유 그룹이 특정 데이터 공유 그룹을 참조할 수 있습니다. Db2 서브시스템의 이름 및 시동 시에 IBM MQ 시스템 매개변수에서 큐 관리자가 사용하는 데이터 공유 그룹을 지정합니다.

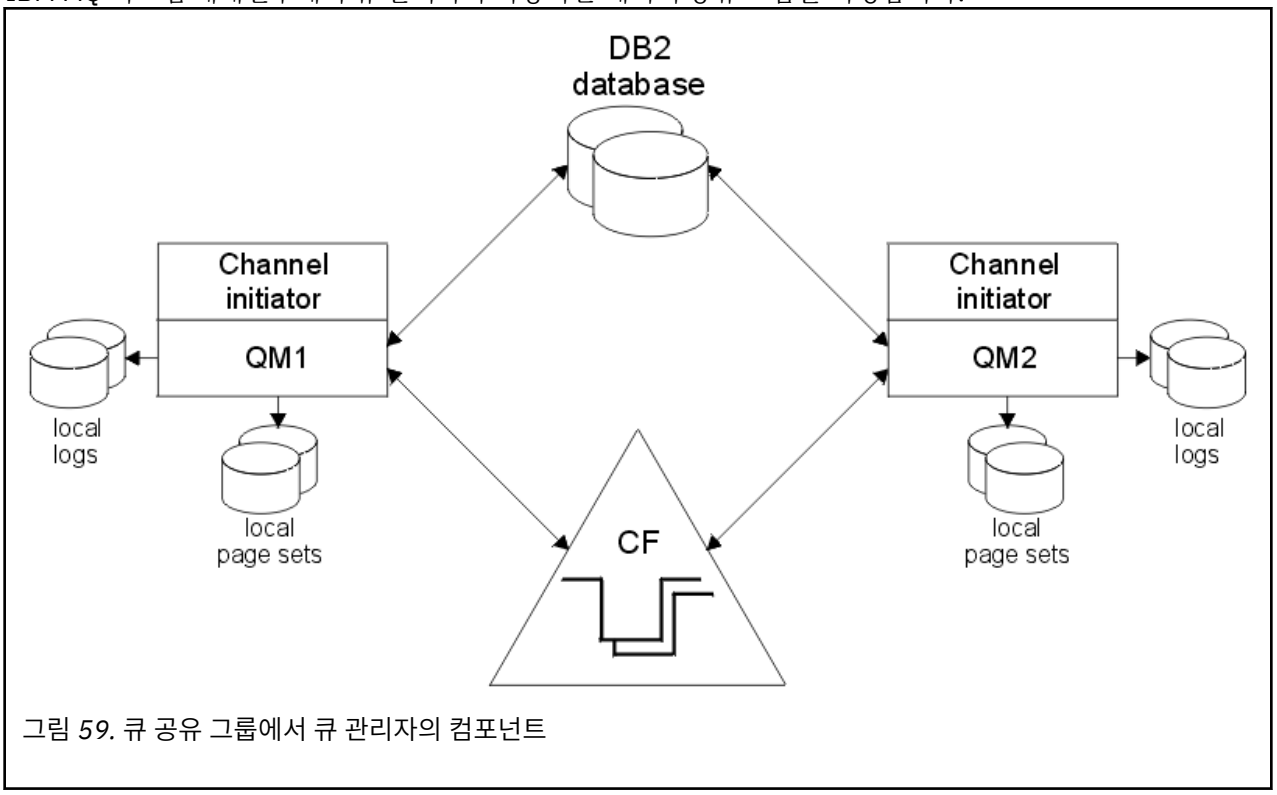


그림 *59.* 큐 공유 그룹에서 큐 관리자의 컴포넌트

큐 관리자가 큐 공유 그룹에 조인한 경우, 이는 해당 그룹에 대해 정의된 공유 오브젝트에 액세스할 수 있으며 사용자는 이 큐 관리자를 사용하여 그룹 내에 새 공유 오브젝트를 정의할 수 있습니다. 공유 큐가 그룹 내에 정의되어 있으면, 이 큐 관리자를 사용하여 해당 공유 큐에 메시지를 넣거나 여기에서 메시지를 가져올 수 있습니다. 이 그룹의 모든 큐 관리자는 공유 큐에 있는 메시지를 검색할 수 있습니다.

MQSC 명령을 한 번만 입력하고, 각 큐 관리자에서 개별적으로 입력된 것처럼 큐 공유 그룹 내의 모든 큐 관리자에서 이 명령이 실행되도록 할 수 있습니다. 이를 위해 명령 범위 속성이 사용됩니다. 이 속성은 서로 다른 큐 관리자에 명령 전달에 설명되어 있습니다.

큐 관리자가 큐 공유 그룹의 멤버로서 실행되는 경우에는 해당 큐 관리자에 개별적으로 정의된 IBM MQ 오브젝트와 큐 공유 그룹의 모든 큐 관리자가 사용할 수 있는 글로벌하게 정의된 IBM MQ 오브젝트 간에 구별이 가능해야 합니다. 이를 위해 큐 공유 그룹 배치 속성이 사용됩니다. 이 속성은 개인용 및 글로벌 정의에 설명되어 있습니다.

그룹 내의 모든 위치에서 IBM MQ 오브젝트에 대한 액세스를 제어하는 단일 세트의 보안 프로파일을 정의할 수 있습니다. 이는 정의해야 하는 프로파일의 수가 상당히 감소됨을 의미합니다.

큐 관리자는 단지 하나의 큐 공유 그룹에 속할 수 있으며, 그룹의 모든 큐 관리자는 동일한 SYSPLEX에 있어야 합니다. 사용자는 시동 시에 시스템 매개변수에 큐 관리자가 속하는 큐 공유 그룹을 지정합니다.

관련 개념

152 페이지의 『공유되는 큐 메시지가 보유되는 위치』
공유 큐의 각 메시지는 z/OS 커플링 기능 목록 구조의 항목으로 나타납니다. 메시지 데이터가 같은 항목에 넣기에 너무 큰 경우 이는 공유 메시지 데이터 세트(SMDS) 또는 Db2로 오프로드됩니다.

166 페이지의 『공유 큐 사용 이점』
공유 큐를 사용하면 IBM MQ 애플리케이션에서 확장 가능성, 고가용성을 허용하고 워크로드 밸런싱을 구현할 수 있습니다.

185 페이지의 『분산 큐잉 및 큐 공유 그룹』

분산 큐잉 및 큐 공유 그룹은 애플리케이션 시스템의 가용성을 높이는 데 사용할 수 있는 두 가지 기술입니다. 이러한 기술에 대한 추가 정보를 찾는 데 이 토픽을 사용하십시오.

188 페이지의 『공유 큐에 영향을 주는 워크로드 분배』
큐 공유 그룹의 공유 큐에서 워크로드 분배에 영향을 주는 요인을 이해하는 데 이 토픽을 사용하십시오.

관련 참조

189 페이지의 『공유 큐 및 큐 공유 그룹에 대한 자세한 정보를 제공하는 위치』
이 주제의 표를 사용하여 IBM MQ for z/OS에서 공유 큐 및 큐 공유 그룹을 사용하는 방법에 대한 자세한 정보를 찾으십시오.

z/OS 공유되는 큐 메시지가 보유되는 위치

공유 큐의 각 메시지는 z/OS 커플링 기능 목록 구조의 항목으로 나타납니다. 메시지 데이터가 같은 항목에 넣기에 너무 큰 경우 이는 공유 메시지 데이터 세트(SMDS) 또는 Db2로 오프로드됩니다.

CF 구조가 System Class Memory(SCM)를 사용하도록 구성된 경우 IBM MQ는 추가 구성 없이 이를 사용할 수 있습니다.

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

공유 큐 메시지 스토리지

공유 큐에 넣은 메시지는 페이지 세트에 저장되지 않으며 버퍼 풀을 사용하지 않습니다.

공유 큐의 메시지가 z/OS 커플링 기능(CF) 안에 있는 목록 구조 상에 항목을 갖고 있습니다. 동일한 SYSPLEX의 많은 큐 관리자가 CF 목록 구조를 사용하여 해당 메시지에 액세스할 수 있습니다.

작은 공유 큐 메시지의 메시지 데이터는 보통 커플링 기능 항목에 포함되어 있습니다. 큰 메시지의 경우 메시지 데이터는 공유 메시지 데이터 세트(SMDS)에 저장되거나 Db2 데이터 공유 그룹에 의해 공유되는 Db2 테이블 안에 하나 이상의 2진 대용량 오브젝트(BLOB)로 저장될 수 있습니다. 63KB를 초과하는 메시지 데이터는 항상 SMDS 또는 Db2로 오프로드됩니다. 커플링 기능 구조의 용량을 절약하기 위해 작은 메시지 또한 같은 방법을 통해 선택적으로 오프로드될 수 있습니다. 자세한 내용은 154 페이지의 『공유 메시지의 오프로드 옵션 지정』의 내용을 참조하십시오.

공유 큐에 들어간 메시지는 MQGET이 이를 검색할 때까지 커플링 기능 구조에서 참조됩니다. 커플링 기능 조작은 다음과 같은 목적에 사용됩니다.

- 다음 검색 가능한 메시지 찾기
- 공유 큐에서 커미트되지 않은 메시지 잠금
- 큐 관리자에게 커미트된 메시지 도착을 알림

지속 메시지에서 MQPUT 및 MQGET 조작은 해당 조작을 수행하는 큐 관리자의 로그에 기록됩니다. 이는 커플링 기능 실패와 같은 사태에서 데이터 유실 위험을 최소화시킵니다.

커플링 기능

공유 큐에 들어있는 메시지는 커플링 기능 안에서 참조됩니다. 커플링 기능은 sysplex 안의 모든 z/OS 이미지 바깥에 위치하며 일반적으로 다른 전원 공급 장치에서 실행되도록 구성됩니다. 따라서 커플링 기능은 소프트웨어 실패에 복원력이 있으며 또한 하드웨어 실패나 정전에 대한 복원력이 있도록 구성할 수 있습니다. 이는 커플링 기능에 저장된 메시지의 가용성이 높다는 것을 의미합니다.

IBM MQ가 사용하는 각 커플링 기능 목록 구조는 특정 큐 공유 그룹 전용이지만 커플링 기능은 둘 이상의 큐 공유 그룹에 대한 구조를 보유할 수 있습니다. 다른 큐 공유 그룹의 큐 관리자는 데이터를 공유할 수 없습니다. 한 커플링 기능 목록 구조에는 큐 공유 그룹 내 큐 관리자가 32개까지 동시에 연결할 수 있습니다.

하나의 커플링 기능 목록 구조는 512개의 공유 큐를 포함할 수 있습니다. 구조에 저장된 총 메시지 데이터 크기는 구조 용량으로 제한됩니다. 그러나 **CFLEVEL(5)** 에서는 오프로드 매개변수를 사용하여 63KB미만의 메시지에 대한 데이터를 오프로드하여 구조에 저장할 수 있는 메시지 수를 늘릴 수 있습니다. 그러나 각 메시지에는 여전히 최소한 커플링 기능 항목과 최소한 768바이트의 데이터가 필요하며, 이는 항목에 대해 256바이트로 구성되고 헤더 및 디스크립터의 두 요소에 대해 512바이트로 구성됩니다.

목록 구조의 크기는 다음 요소로 제한됩니다.

- 하나의 커플링 기능 안에 있어야 합니다.
- 이는 IBM MQ 및 기타 제품의 다른 구조와 사용 가능한 커플링 기능 스토리지를 공유할 수 있습니다.

커플링 기능 목록 구조는 스토리지 클래스 메모리가 연관되어 있을 수 있습니다. 특정 상황에서 이 스토리지 클래스 메모리는 공유 큐와 함께 사용할 때 유용할 수 있습니다. 자세한 정보는 168 페이지의 『공유 큐에서 스토리지 클래스 메모리의 사용』의 내용을 참조하십시오.

CF 구조 크기 계획

CF 구조의 크기 조정에 대한 지침이 필요한 경우 MP16: IBM MQ for z/OS 용량 계획 및 성능 조정 supportpac를 사용할 수 있습니다. CF 크기를 지원하기 위해 IBM 에서 제공하는 웹 기반 도구 CFSizer를 사용할 수도 있습니다.

CF 구조 오브젝트

큐 관리자의 커플링 기능 구조 사용은 CF 구조(CFSTRUCT) IBM MQ 오브젝트에 지정되어 있습니다.

이러한 구조 오브젝트는 Db2에 저장됩니다.

z/OS 명령이나 커플링 기능 구조와 관련된 정의를 사용할 때 큐 공유 그룹 이름의 첫 4자가 필요합니다. 그러나 IBM MQ CFSTRUCT 오브젝트는 단일 큐 공유 그룹 내 항상 존재하므로, 해당 이름은 큐 공유 그룹 이름의 첫 4자는 포함하지 않습니다. 예를 들어, SQ03으로 시작하는 큐 공유 그룹에 정의된 CFSTRUCT(MYDATA)는 커플링 기능 목록 구조 SQ03MYDATA를 사용할 수 있습니다.

CF 구조에는 해당 기능 용량을 판별하는 CFLEVEL 속성이 있습니다.

- 1, 2 - 63KB 미만의 비지속 메시지에 사용 가능
- 3 - 63KB 미만의 지속 및 비지속 메시지에 사용 가능
- 4 - 최대 100MB의 지속 및 비지속 메시지에 사용 가능
- 5 - 최대 100MB의 지속 및 비지속 메시지에 사용 가능하며, 공유 메시지 데이터 세트(SMDS) 또는 Db2에 선택적으로 오프로드됨

참고: IBM MQ 를 사용할 때 커플링 기능 구조를 암호화할 수 있습니다. 자세한 정보는 결합 기능 구조 데이터 암호화 를 참조하십시오.

커플링 기능의 백업 및 복구

IBM MQ 명령 BACKUP CFSTRUCT를 사용하여 커플링 기능 목록 구조를 백업할 수 있습니다. 그러면 현재 CF 구조에 있는 지속 메시지의 사본을 백업을 수행하는 큐 관리자의 활성 로그 데이터 세트에 넣고 Db2에 백업의 레코드를 기록합니다.

커플링 기능에 실패할 경우 IBM MQ 명령 RECOVER CFSTRUCT를 사용할 수 있습니다. 여기에서는 Db2의 백업 레코드를 사용하며 CF 구조의 백업에서 지속 메시지를 찾아 복원합니다. 마지막 백업 이후 모든 활동은 큐 공유 그룹의 모든 큐 관리자 로그를 사용하여 재실행되며 CF 구조는 실패 전 지점으로 복원됩니다.

자세한 내용은 BACKUP CFSTRUCT 및 RECOVER CFSTRUCT 명령을 참조하십시오.

관련 개념

154 페이지의 『공유 메시지의 오프로드 옵션 지정』
Db2 테이블 또는 공유 메시지 데이터 세트(SMDS) 중 공유 큐 메시지에 대한 메시지 데이터를 저장하는 위치를 선택할 수 있습니다. 또한 커플링 기능(CF) 구조의 현재 사용량 및 메시지 크기를 기반으로 하는 오프로드되는 메시지를 선택할 수도 있습니다.

156 페이지의 『공유 메시지 데이터 세트(SMDS, shared message data set) 환경 관리』
큰 메시지를 오프로드하기 위해 공유 메시지 데이터 세트를 선택할 경우 IBM MQ가 이 데이터 세트를 관리하는 데 사용하는 정보와 이 정보를 갖고 작업하는 데 사용되는 명령을 알아야 합니다. 이 주제를 사용하여 공유 메시지 데이터 세트를 관리하는 방법을 이해하십시오.

z/OS 공유 메시지의 오프로드 옵션 지정

Db2 테이블 또는 공유 메시지 데이터 세트(SMDS) 중 공유 큐 메시지에 대한 메시지 데이터를 저장하는 위치를 선택할 수 있습니다. 또한 커플링 기능(CF) 구조의 현재 사용량 및 메시지 크기를 기반으로 하는 오프로드되는 메시지를 선택할 수도 있습니다.

공유 큐에 대한 메시지 데이터는 커플링 기능에서 오프로드되어 Db2 테이블 또는 공유 메시지 데이터 세트(SMDS) 라는 IBM MQ 관리 데이터 세트에 저장될 수 있습니다.

63KB의 커플링 기능 입력 항목 크기보다 큰 메시지의 경우 SMDS로 메시지 데이터를 오프-로딩하면 Db2로 오프로딩할 때보다 성능이 더 크게 향상될 수 있습니다.

모든 공유 큐 메시지는 커플링 기능 구조의 목록 입력 항목을 사용하여 계속 관리되지만, 메시지 데이터가 SMDS로 오프로드되면 커플링 기능 입력 항목은 일부 제어 정보와 메시지가 저장되는 관련 디스크 블록에 대한 참조 목록만 포함합니다. 이 메커니즘을 사용하면 각 메시지에 필요한 커플링 기능 요소 스토리지의 크기는 메시지의 실제 크기 중 일부에만 해당됨을 의미합니다.

공유 큐 메시지가 저장되는 위치 선택

SMDS 또는 Db2 공유 메시지 스토리지의 선택은 **CFSTRUCT** 정의의 **OFFLOAD(SMDS|DB2)** 매개변수로 제어됩니다. **OFFLOAD(SMDS)**는 기본값입니다.

또한 이 매개변수에서는 **CFSTRUCT**가 **CFLEVEL(5)** 이상을 사용해야 합니다.

OFFLOAD 매개변수는 **CFLEVEL(5)**에서만 유효합니다. 자세한 정보는 DEFINE CFSTRUCT를 참조하십시오.

OFFLOAD(DB2)는 주로 마이그레이션 목적으로 지원됩니다.

오프로드되는 공유 큐 메시지 선택

메시지 데이터의 크기, 커플링 기능 구조의 현재 사용량을 기반으로 하는 Db2 또는 SMDS로 메시지 데이터가 오프로드됩니다. 3개의 규칙이 있으며 각 규칙은 일치하는 매개변수 쌍을 지정합니다. 이러한 매개변수는 해당되는 커플링 기능 구조 사용 임계값 백분율(**OFFLDnTH**) 및 메시지 크기 한계(**OFFLDnSZ**)입니다.

현재 3개 규칙의 구현은 다음 키워드 쌍을 사용하여 지정됩니다.

- OFFLD1TH 및 OFFLD1SZ
- OFFLD2TH 및 OFFLD2SZ
- OFFLD3TH 및 OFFLD3SZ

규칙 쌍	기본값	설명
규칙 쌍 1	OFFLD1TH(70) 및 OFFLD1SZ(32K)	커플링 기능 구조가 32KB를 초과하는 메시지에 대한 전체 오프로드 데이터 70%를 초과하는 경우
규칙 쌍 2	OFFLD2TH(80) 및 OFFLD2SZ(4K)	커플링 기능 구조가 4KB를 초과하는 메시지에 대한 전체 오프로드 데이터 80%를 초과하는 경우
규칙 쌍 3	OFFLD3TH(90) 및 OFFLD3SZ(0K)	커플링 기능 구조가 0KB를 초과하는 메시지(모든 메시지)에 대한 전체 오프로드 데이터 90%를 초과하는 경우

오프로드 규칙의 OFFLD *x* SZ 값이 64K인 경우 이는 규칙이 적용되지 않음을 표시합니다. 이 경우에는 다른 오프로드 규칙이 적용되거나 메시지가 63.75KB보다 커서 구조에 저장할 수 없는 경우에만 메시지가 오프로드됩니다.

오프로드되는 각 메시지는 커플링 기능에서 여전히 0.75KB의 스토리지를 요구합니다.

각 구조에 대해 지정할 수 있는 세 가지 오프로드 규칙은 다음과 같이 사용됩니다.

- 성능
  - 애플리케이션 구조의 공간이 많은 경우 메시지가 너무 커서 구조에 저장할 수 없거나 일부 더 낮은 메시지 크기 임계값을 초과하는 경우에만 메시지 데이터를 오프로드하여 메시지를 구조에 저장할 때의 성능이 필요한 구조 공간의 크기에 영향을 미치지 못하도록 해야 합니다.
  - 특정 메시지 크기 임계값이 필요한 경우에는 대개 첫 번째 오프로드 규칙을 사용하여 지정합니다.
- 용량
  - 애플리케이션 구조의 공간이 너무 작은 경우에는 남은 공간을 최적으로 사용할 수 있도록 최대 크기의 메시지 데이터가 오프로드되어야 합니다.
  - 일반적으로 세 번째 오프로드 규칙은 구조가 거의 찼을 경우 대부분의 메시지가 오프로드되어 애플리케이션 구조의 입력 항목이 최소 크기(약 0.75K바이트)가 된다는 것을 나타내는 데 사용됩니다.
  - 사용 임계값 매개변수는 애플리케이션 크기 구조 및 예상되는 최대 백로그에 기반하여 선택해야 합니다. 예를 들어, 예상되는 최대 백로그가 1M의 메시지인 경우 이 메시지 수에 필요한 구조 스토리지의 크기는 0.75G바이트 정도입니다. 즉, 구조가 약 10G바이트인 경우 모든 메시지를 오프-로딩하는 데 필요한 사용 임계값은 92% 이하로 설정되어야 합니다.
  - 구조 공간은 여러 개의 요소와 입력 항목으로 나뉘며, 전체 공간이 충분하더라도 이들 요소와 입력 항목 중 하나가 다른 것보다 먼저 소모될 수 있습니다. 시스템에서는 AUTOALTER 기능을 제공하여 필요할 때 비율을 조정할 수 있도록 하지만, 이 기능은 아주 세밀한 것이 아니므로 실제로 사용할 수 있는 공간의 양은 다소 적을 수 있습니다. 따라서 최대 구조 공간의 90%를 초과하여 사용하지 않도록 권장되므로, 이전 예제에서 모든 메시지의 오프-로딩을 위한 사용량 임계값은 약 80% 정도로 설정하는 것이 좋습니다.
- 쿠션을 통한 전이:
  - 커플링 기능 구조에 남아 있는 공간의 크기를 줄일 때 성능에 갑작스러운 큰 변화가 생기는 것은 좋지 않을 수 있습니다. 또한 커플링 기능 관리에서 사용 중인 요소에 대한 입력 항목의 일반적인 비율에 갑작스러운 임계값 변화가 생기는 것도 좋지 않습니다.
  - 일반적으로 두 번째 오프로드 규칙은 성능 및 용량 중심의 오프로드 규칙 사이의 매개 수단인 쿠션을 제공하는 데 사용됩니다. 커플링 기능 구조에 사용된 공간이 중간 임계값을 초과하는 경우, 오프로드 활동이 크게 증가하도록 설정할 수 있습니다. 즉, 남은 공간이 더 느리게 소모되고 커플링 기능의 자동 대체 처리에 더 많은 시간을 할애함으로써 높은 사용 레벨에 적응할 수 있도록 합니다.

커플링 기능 구조를 확장할 수 없고 사전 판별된 최소 메시지 수 이상을 저장해야 하는 경우 사전 판별된 해당 메시지 수에 대해 공간을 예약하기 위해 모든 메시지의 오프-로딩이 적절한 임계값에 시작되도록 필요하면 세 번째 규칙을 수정할 수 있습니다.

예를 들어 커플링 기능 구조 크기가 4GB이고 메시지의 사전 판별된 수가 1백만 개인 경우 1,000,000 * 0.75KB가 필요합니다. 이는 4GB의 18.75%인 768MB입니다. 이때 모든 메시지를 오프-로딩하는 경우 임계값은 90%보다는 80% 정도로 설정해야 합니다. 여기에서는 매개변수 OFFLD3TH(80) 및 OFFLD3SZ(0K)를 제공합니다. 다른 오프로드 매개변수도 조정해야 합니다.

매우 작은 메시지의 오프-로딩은 성능에 대한 영향이 크지만, 상대적인 영향은 더 큰 메시지보다는 적음을 확인한 경우 다른 규칙의 사용량 임계값을 줄여 더 먼저 더 큰 메시지를 오프로드할 수 있습니다. 그러면 오프로드해야 하기 전에 작은 메시지에 대해 구조에 더 많은 공간이 남습니다.

예를 들어 32KB를 초과하는 메시지가 자주 발생하지만 이들을 오프-로딩하는 경과된 시간 성능(RMF 통계 또는 애플리케이션 성능에서 판별됨)이 이들을 커플링 기능에 보관하는 경우와 매우 유사한 경우 첫 번째 규칙의 임계값을 0%로 설정하여 이러한 모든 메시지를 오프로드할 수 있습니다. 여기에서는 매개변수 OFFLD1TH(0) 및 OFFLD1SZ(32K)를 제공합니다. 다른 오프로드 매개변수를 다시 조정해야 합니다.

특정 중간 크기(예: 16KB 및 6KB)에 가까운 메시지가 많으면 두 번째 규칙의 메시지 크기 옵션을 변경하는 것이 유용합니다. 그러면 더 큰 메시지는 비교적 낮은 사용량 임계값에서 오프로드되어 상당한 공간을 절약하는 반면, 더 작은 메시지는 커플링 기능에만 계속 저장됩니다.

z/OS 공유 메시지 데이터 세트*(SMDS, shared message data set)* 환경 관리

큰 메시지를 오프로드하기 위해 공유 메시지 데이터 세트를 선택할 경우 IBM MQ가 이 데이터 세트를 관리하는 데 사용하는 정보와 이 정보를 갖고 작업하는 데 사용되는 명령을 알아야 합니다. 이 주제를 사용하여 공유 메시지 데이터 세트를 관리하는 방법을 이해하십시오.

SMDS 오브젝트

특성과 각 공유 메시지 데이터 세트의 상태는 큐 공유 그룹에서 큐 관리자를 통하여도 업데이트될 수 있는 공유된 SMDS 오브젝트에서 추적됩니다.

각 큐 관리자마다 각 커플링 기능 애플리케이션 구조에 액세스할 수 있는 하나의 공유 메시지 데이터 세트가 있습니다. 공유 메시지 데이터 세트는 SMDS 키워드를 사용하여 지정된 소유하고 있는 큐 관리자 이름 및 CFSTRUCT 키워드를 사용하여 지정된 애플리케이션 구조 이름으로 식별됩니다.

참고: 구조에 대한 SMDS 데이터 세트를 정의 할 때 각 큐 관리자에 SMDS 데이터 세트가 있어야 합니다.

SMDS 오브젝트는 Db2에 저장된 해당 CFSTRUCT 오브젝트의 확장을 구성하는 배열(그룹의 큐 관리자당 하나의 항목으로)에 저장됩니다.

CFSTRUCT 오브젝트의 일부로 작성 또는 삭제되므로 SMDS 오브젝트를 정의 또는 삭제하는 명령은 없지만 소유하고 있는 큐 관리자에 대한 설정을 변경하도록 이를 대체하는 명령이 있습니다.

SMDS 명령에 대한 추가 정보는 165 페이지의 『SMDS 관련 명령』의 내용을 참조하십시오.

SMDSCONN 정보

공유 메시지 데이터 세트가 정상 상태일 수 있지만, 이에 연결할 수 없는 하나 이상의 큐 관리자에 대해 가능합니다. 예를 들어 직접 액세스 디바이스 연결성 또는 보안 정의에 문제점이 있을 수 있기 때문입니다. 따라서 각 큐 관리자가 현재 연결할 수 있는지 여부 및 연결할 수 없는 경우 이유를 표시하는 연결 상태 및 각 공유 메시지 데이터 세트의 가용성 정보를 추적해야 합니다.

SMDSCONN 정보는 공유 메시지 데이터 세트에 대한 큐 관리자 연결을 표시합니다. 공유 메시지 데이터 세트와 마찬가지로, 이는 CFSTRUCT 이름과 결합된 공유 메시지 데이터 세트(공유 오브젝트 자체에 대한 SMDS 키워드에 지정됨)를 소유하는 큐 관리자에 의해 식별됩니다.

특정 큐 관리자로 주소 지정된 명령은 동일한 해당 큐 관리자에 대한 SMDSCONN 정보만 참조할 수 있으므로 연결하는 큐 관리자를 식별할 매개변수가 없습니다.

SMDSCONN 정보 입력 항목은 소유하고 있는 큐 관리자의 주 기억장치에서 유지보수되며 큐 관리자가 재시작될 때 다시 작성됩니다. 그러나 개별 큐 관리자의 연결이 명시적으로 중지된 경우 이 정보는 해당 CFSTRUCT 또는 SMDS 오브젝트의 연결 배열에서 플래그로도 저장되므로 큐 관리자 재시작에서 지속됩니다.

상태 및 사용가능성 정보

상태 정보는 자원 또는 연결 상태(예: 아직 사용 중이 아닌지, 정상 사용 중인, 또는 복구가 필요한지 여부)를 표시합니다. 일반적으로 이는 STATUS 키워드를 사용하여 설명됩니다. 가능한 값은 오브젝트의 유형에 따라 달라집니다.

예를 들어 자원 또는 연결을 사용하는 중에 오류가 감지되면 상태 정보는 일반적으로 자동 업데이트됩니다. 그러나 일부 경우에 큐 관리자가 올바른 상태를 자동으로 판별할 수 없는 경우에 허용하도록 명령을 사용하여 상태를 업데이트할 수 있습니다.

가용성 정보는 자원 또는 연결을 사용할 수 있는지 여부를 나타내며 상태 정보에 의해 기본적으로 판별됩니다. 공유 메시지 데이터 세트 지원에 사용되는 자원 또는 연결 유형의 경우 다음과 같은 세 가지 레벨의 가용성이 구현됩니다.

사용 가능
이는 일반적으로 자원이 사용 가능함을 의미합니다. 이는 현재 사용 중임을 반드시 의미하지는 않습니다(STATUS 값에서 대신 판별할 수 있음). 데이터 세트의 경우 재시작 처리가 필요하면 소유하고 있는 큐 관리자에서는 이를 열 수 있지만, 데이터 세트가 ACTIVE 상태로 다시 돌아갈 때까지 다른 큐 관리자는 기다려야 합니다.

오류로 인해 사용 불가능
이는 오류로 인해 자원이 자동으로 사용 불가능해졌으며, 일부 형식의 수리 또는 복구 처리가 수행될 때까지 사용할 수 없다고 예상됩니다. 그러나 운영자 개입 없이 다시 사용 가능하게 하려는 시도가 허용됩니다. 또한 이러한 시도는 복구 처리가 완료되었음을 알리는 것과 같은 방식으로 상태를 변경하는 명령 또는 자원을 사용 가능한 것으로 표시하는 명령에 의해 트리거될 수 있습니다.

자원이 사용 불가능해진 이유는 보통 관련 STATUS 값에서 명백하지만, 일부 경우에 자원이 사용 불가능해진 다른 이유가 있을 수 있습니다. 이때에는 이유를 표시하기 위해 별도의 REASON 값이 제공됩니다.

연산자 명령으로 인해 사용 불가능
이는 자원에 대한 액세스가 명령에 의해 명시적으로 사용 불가능함을 의미합니다. 이를 다시 사용 가능하게 하려면 명령을 사용하여 사용 가능하게 하는 방법뿐입니다.

SMDS 사용가능성

공유 SMDS 오브젝트의 경우 가용성은 가능한 값이 ENABLED, SUSPENED 및 DISABLED인 ACCESS 키워드에서 설명됩니다.

ACCESS (ENABLED) 또는 ACCESS (DISABLED) 를 설정하기 위해 그룹의 큐 관리자에서 관련 공유 오브젝트에 대한 **RESET SMDS** 명령을 사용하여 가용성을 업데이트할 수 있습니다.

이전에 가용성이 ACCESS(SUSPENDED)인 경우 ACCESS(ENABLED)로 변경하면 공유 메시지 데이터 세트를 사용하려는 새 시도가 트리거되지만, 이전 오류가 계속 존재하면, 가용성은 ACCESS(SUSPENDED)로 다시 재설정됩니다.

SMDSCONN 사용가능성

로컬 SMDSCONN 정보 입력 항목의 경우 가용성은 가능한 값이 NORMAL, ERROR 또는 STOPPED인 AVAIL 키워드로 설명됩니다. 해당 연결을 사용 또는 사용 안함으로 설정하기 위해 특정 큐 관리자로 주소 지정된 **START SMDSCONN** 또는 **STOP SMDSCONN** 명령을 사용하여 가용성을 업데이트할 수 있습니다.

이전에 가용성이 ACCESS(ERROR)인 경우 ACCESS(NORMAL)로 변경하면 공유 메시지 데이터 세트를 사용하려는 새 시도가 트리거되지만, 이전 오류가 계속 존재하면, 가용성은 ACCESS(ERROR)로 다시 재설정됩니다.

공유 메시지 데이터 세트 공유 상태 및 가용성

각 공유 메시지 데이터 세트의 가용성은 공유 상태 정보를 사용하여 그룹 내에서 관리되며, TYPE (SMDS) 과 함께 **DISPLAY CFSTATUS** 명령을 사용하여 표시할 수 있습니다. 여기에서는 각 구조의 데이터 세트를 활성화하는 각 큐 관리자에 대한 상태 정보를 표시합니다. 각 데이터 세트는 다음 상태 중 하나입니다.

NOTFOUND
이는 해당 데이터 세트가 아직 활성화되지 않았음을 의미합니다. 이 상태는 모든 큐 관리자가 선택된 경우 활성화되지 않은 데이터 세트는 건너뛰므로 특정 큐 관리자가 지정된 경우에만 나타납니다.

새로 작성
데이터 세트가 활성화할 준비로, 처음 열리고 초기화됩니다.

활성
이는 데이터 세트가 모두 사용 가능하고 구조에 대한 모든 활성 큐 관리자에 의해 할당되고 열려야 함을 의미합니다.

FAILED
이는 데이터 세트를 전혀 사용할 수 없으며(복구 처리에 대해서는 예외임) 모든 큐 관리자가 닫고 할당 취소해야 함을 의미합니다.

INRECOVER
이는 매체 복구(RECOVER CFSTRUCT)가 이 데이터 세트에 대해 진행 중임을 의미합니다.

RECOVERED
이는 실패한 데이터 세트를 다시 활성 상태로 전환하도록 명령이 실행되지만, 아직 완료되지 않은 추가 재시작 처리가 필요함을 의미합니다. 따라서 재시작 처리를 위해 소유하는 큐 관리자에서만 데이터 세트를 열 수 있습니다.

EMPTY
데이터 세트가 메시지를 포함하지 않습니다. 메시지를 포함하지 않은 경우에 소유하고 있는 큐 관리자에 의해 정상적으로 닫히면 데이터 세트가 이 상태로 설정됩니다. TYPE PURGE와 함께 **RECOVER CFSTRUCT**를 사용하거나 복구 불가능한 구조에 한해 구조의 이전 인스턴스를 삭제하여 애플리케이션 구조를 비웠기 때문에 이전 데이터 세트 컨텐츠를 제거해야 할 경우 데이터 세트가 EMPTY 상태에 배치될 수도 있습니다. 소유하고 있는 큐 관리자가 다음에 데이터 세트를 열면, 공간 맵이 비어 있음으로 재설정되고 상태가 ACTIVE로 변경됩니다. 이전 데이터 세트 컨텐츠가 더 이상 필요하지 않으므로 이 상태의 데이터 세트를 새로 할당된 데이터 세트로 바꿀 수 있습니다(예: 공간 할당을 변경하거나 다른 볼륨으로 이동).

명령 출력에는 복구 로깅이 사용 가능한 날짜 및 시간과 현재 활성 상태가 아닌 경우 데이터 세트가 실패한 날짜 및 시간이 포함됩니다.

공유 메시지 데이터 세트는 **RESET SMDS** 명령에 의해 또는 다음 유형의 오류가 발견될 때 자동으로 FAILED 상태가 될 수 있습니다.

- 소유하고 있는 큐 관리자에 의해 데이터 세트를 할당하거나 열 수 없습니다.
- 큐 관리자에 의해 성공적으로 열린 후에 데이터 세트 헤더의 유효성 검증에 실패합니다.
- 소유하고 있는 큐 관리자에서 데이터를 읽거나 쓸 때 영구적인 I/O 오류가 발생합니다.
- 큐 관리자가 열기 처리 및 유효성 검증을 성공적으로 완료한 데이터 세트에서 데이터를 읽을 때 영구적인 I/O 오류가 발생합니다.

데이터 세트가 FAILED 또는 INRECOVER 상태인 경우 정상 사용을 위해 사용할 수 없으므로, 가용성 상태가 ACCESS(ENABLED)인 경우 ACCESS(SUSPENDED)로 변경됩니다.

데이터 세트가 FAILED 상태로 설정되었지만 매체 복구가 필요하지 않은 경우(예: 데이터가 계속 유효하지만, 스토리지 디바이스가 일시적으로 오프라인이기 때문), **RESET SMDS** 명령을 사용하여 상태를 직접 RECOVERED 상태로 변경하도록 요청할 수 있습니다.

데이터 세트가 RECOVERED 상태로 설정되면 복구 처리 완료 시 또는 **RESET SMDS** 명령 결과로, 재시작 처리가 완료되면 다시 사용할 수 있습니다. ACCESS(SUSPENDED) 상태인 경우 자동으로 ACCESS(ENABLED) 상태로 전환되며, 이를 통해 소유하고 있는 큐 관리자가 재시작 처리를 수행할 수 있습니다. 재시작 처리를 완료하면 상태는 ACTIVE로 변경되고, 다른 모든 큐 관리자가 다시 데이터 세트에 연결할 수 있습니다.

공유 메시지 데이터 세트 연결 상태 및 가용성

각 큐 관리자는 자체적으로 소유한 것과 그룹의 다른 큐 관리자가 소유한 각 공유 메시지 데이터 세트로의 연결에 대한 로컬 상태 및 가용성 정보를 유지보수합니다. 이 정보는 **DISPLAY SMDSCONN** 명령을 사용하여 표시할 수 있습니다.

다른 큐 관리자에 속한 ACTIVE 상태의 공유 메시지 데이터 세트에 액세스할 수 없는 경우 고유한 시점에서 연결을 사용 불가능하다는 플래그를 지정합니다.

오류가 명확히 데이터 세트에 대한 문제점을 표시하는 경우 큐 관리자는 데이터 세트가 현재 FAILED 상태임을 표시하도록 자동으로 공유 큐 상태도 변경합니다. 그러나 오류가 환경 문제점으로 인해 발생한 경우(예: 데이터 세트를 열 권한이 없음) 큐 관리자는 오류 메시지를 발행하고 데이터 세트를 사용 불가능한 것으로 처리하지만, 공유 데이터 세트 상태는 수정하지 않습니다. 환경 오류가 데이터 세트 문제점인 것으로 판명되면(예: 일부 큐 관리자가 액세스할 수 없는 디바이스에 할당됨) 운영자는 STATUS(FAILED)를 지정하는 RESET SMDS 명령을 사용하여 필요하면 데이터 세트를 수리 또는 복구할 수 있습니다.

공유 메시지 데이터 세트에 대한 연결을 설정할 수 없지만, 데이터 세트가 올바른 것으로 나타나면 소유하는 큐 관리자에 대해 **START SMDSCONN** 명령을 실행하여 이를 사용하려는 새로운 시도가 트리거될 수 있습니다.

운영상 특정 큐 관리자와 데이터 세트 사이의 연결을 임시로 종료해야 하지만, 데이터 자체는 손상되지 않은 경우 **STOP SMDSCONN** 명령을 사용하여 데이터 세트를 닫고 할당 취소할 수 있습니다. 데이터 세트가 사용 중이면 해당 데이터 세트의 데이터에 대한 요청이 리턴 코드와 함께 거부되어도 큐 관리자는 이를 정상적으로 닫습니다. 소유한 데이터 세트인 경우 큐 관리자는 재시작 처리를 수행하지 않아도 되도록 CLOSE 처리 중 공간 맵을 저장합니다.

모든 큐 관리자에서 임시로 데이터 세트 지원을 중단해야 하지만(예: 이동 시) 손상되지 않은 경우 관련 데이터 세트에서 옵션 CMDSCOPE(*)와 함께 **STOP SMDSCONN**을 사용하여 먼저 이를 사용하는 큐 관리자를 중지하는 것이 가장 좋습니다. 그러면 데이터 세트를 다시 지원하게 될 때 재시작 처리를 수행하지 않아도 됩니다. 반대로 데

이터 세트가 FAILED로 표시되면 이는 큐 관리자에게 사용을 즉시 중지해야 함을 알립니다. 이는 공간 맵이 저장되지 않으며 재시작 처리로 다시 빌드해야 함을 의미합니다.

이전에 ACCESS(SUSPENDED) 상태인 공유 메시지 데이터 세트에 대한 액세스는 큐 관리자가 재시작되면 재시도됩니다.

공유 메시지 데이터 세트 복구 로깅

매체 복구를 위해 지속 공유 메시지가 로그됩니다. 즉, 복구 로그가 손상되지 않은 경우 공유 메시지 데이터 세트 또는 커플링 기능 구조의 실패 후 메시지를 복구할 수 있습니다. 또한 지속 메시지는 장애 복구 목적을 위해 다른 사이트에서 복구 로그를 다시 작성할 수 있습니다.

메시지 데이터가 공유 메시지 데이터 세트에 기록되면 데이터 세트에 기록된 각 블록은 커플링 기능에 쓰여진 대로, 메시지 입력 항목 앞에서(데이터 맵 포함) 별도로 로그됩니다. 복구 프로세스는 항상 커플링 기능 구조를 복구하지만, 데이터 세트 상태가 FAILED가 아니거나 데이터 세트가 다시 만들어졌음을 의미하는, 상태가 ACTIVE이나 데이터 세트 헤더 레코드가 올바르지 않은 경우가 아니면 복구 프로세스가 개별 공유 메시지 데이터를 복구할 필요는 없습니다. 상태가 ACTIVE이며 데이터 세트 헤더가 아직 올바르거나 실패 당시에 저장된 메시지가 없었음을 의미하는, 상태가 EMPTY인 경우가 아니면 데이터 세트는 복구를 위해 선택되지 않습니다.

공유 메시지 데이터 세트 백업

BACKUP CFSTRUCT가 애플리케이션 구조의 공유 메시지 백업을 작성하는 데 사용되는 경우 DB에 이전에 저장된 지속 공유 메시지에 대해서 공유 메시지 데이터 세트에 저장된 지속 메시지의 데이터도 동시에 백업됩니다.

공유 메시지 데이터 세트 복구

공유 메시지 데이터 세트가 파손되거나 유실된 경우 FAILED 상태로 설정하여 수리될 때까지 큐 관리자가 이를 사용하지 않도록 합니다. 이는 정상적으로 자동 수행되지만, STATUS(FAILED)를 지정하는 **RESET SMDS** 명령을 사용하여 수행할 수도 있습니다.

공유 메시지 데이터 세트가 지속 메시지를 포함하는 경우 RECOVER CFSTRUCT 명령을 사용하여 복구할 수 있습니다. 이 명령은 먼저 최근 BACKUP CFSTRUCT 명령의 해당 공유 메시지 데이터 세트에 대한 지속 메시지 데이터를 복원한 다음, 이후에 모든 로그된 변경을 적용합니다. 데이터 세트가 처음 활성화된 시간 이후에 **BACKUP CFSTRUCT** 명령이 수행되지 않은 경우 비어 있음으로 재설정되고 활성화 이후의 모든 변경사항이 적용됩니다.

CFSTRUCT 컨텐츠 및 모든 공유 메시지 데이터 세트를 사용할 수 없는 경우 (예: 재해 복구 상황에서) 단일 **RECOVER CFSTRUCT** 명령으로 모두 복구할 수 있습니다.

공유 메시지 데이터 세트가 손상되었지만 CFSTRUCT에 대해 복구가 활성 상태가 아니거나 마지막 BACKUP CFSTRUCT를 포함하는 로그가 사용 불가능하면 해당 데이터 세트로 오프로드된 메시지를 복구할 수 없습니다. 이 경우 매개변수 TYPE(PURTE)을 포함하는 **RECOVER CFSTRUCT** 명령을 사용하여 빈 상태로 공유 메시지 데이터 세트를 표시하고 해당 데이터 세트에 데이터를 저장하는 구조에서 메시지를 삭제할 수 있습니다.

RECOVER CFSTRUCT 명령이 발행되면 공유 메시지 데이터 세트 상태는 FAILED에서 INRECOVER로 변경됩니다. 복구가 성공적으로 완료되면 상태는 자동으로 RECOVERED로 변경됩니다. 그렇지 않으면 FAILED로 다시 변경됩니다.

데이터 세트가 RECOVERED 상태로 변경되면 이는 이제 데이터 세트를 열고 재시작 처리를 수행할 수 있음을 소유하고 있는 큐 관리자에 알립니다.

공유 메시지 데이터 세트 복구 및 동기점

공유 메시지 데이터 세트 복구 프로세스는 동기점에 상관없이 로그 끝에 모든 전체 로그 레코드의 변경을 다시 적용합니다.

동기점 안에서 변경이 이루어진 경우 CFSTRUCT에 대한 재시작 또는 복구 처리로 인해 커미트되지 않은 요청이 다시 백아웃되므로 복구된 일부 변경은 실제로 사용되지 않을 수 있지만, 이들을 복구해도 유해하지는 않습니다.

또한 커미트되지 않은 MQPUT 메시지를 구조에 기록할 수도 있지만, 해당 데이터는 로그 또는 데이터 세트에 기록되지 않을 수도 있습니다(동기점 처리 시작 시 I/O 완료가 강제 실행으로만 수행됨). 재시작 처리가 구조에서 메시지 입력 항목을 백아웃하므로 이 작업은 무해합니다. 따라서 복구되지 않은 데이터를 참조하는 사실은 문제가 없습니다.

공유 메시지 데이터 세트 재시작 처리

CFSTRUCT에 대한 큐 관리자 연결이 정상적으로 종료되면 큐 관리자는 데이터 세트를 닫기 바로 전에 각 공유 메시지 데이터 세트에 대한 여유 공간 맵을 데이터 세트 내 체크포인트 영역에 기록합니다. 그러면 CFSTRUCT 및 공유 메시지 데이터 세트에서 다음 재시작 전에 복구 처리가 필요하지 않은 경우 공간 맵은 연결 재시작 시 다시 읽을 수 있습니다.

그러나 큐 관리자가 비정상적으로 종료되거나 구조 또는 데이터 세트에서 복구 처리가 필요한 경우 구조에 대한 큐 관리자 연결을 재시작할 때 공간 맵을 동적으로 다시 빌드하도록 추가 처리가 필요합니다.

데이터 세트 자체를 복구하지 않아도 되는 경우 큐 관리자 재시작 시 현재 구조 컨텐츠를 단순히 스캔하여 현재 큐 관리자가 소유한 메시지 데이터 세트에 대한 참조를 찾고 공간 맵에서 소유한 것으로 관련 데이터 블록을 표시합니다. 다른 큐 관리자는 계속해서 구조를 사용하고 큐 관리자를 재시작하여 소유한 데이터를 읽으면서, 공간 맵을 다시 빌드합니다.

공유 메시지 데이터 세트 복구 후 재시작

공유 메시지 데이터 세트를 백업에서 복구해야 하는 경우 데이터 세트에 저장된 모든 비지속 메시지는 유실되며, 데이터 세트가 TYPE(PURGE)을 사용하여 복구된 경우 데이터 세트에 저장된 모든 메시지가 유실됩니다. 복구를 완료할 때까지 데이터 세트는 FAILED 또는 INRECOVER로 표시되므로, 다른 큐 관리자에서 관련된 메시지 중 하나를 읽으려고 하면, 데이터 세트가 일시적으로 사용 불가능함을 표시하는 오류 코드가 리턴됩니다.

데이터 세트가 복구되면 상태는 RECOVERED로 변경되고, 소유하고 있는 큐 관리자가 재시작 처리를 위해 이를 열 수 있습니다. 그러나 다른 큐 관리자에 대해서 데이터 세트는 사용 불가능 상태로 남아 있습니다. 큐 관리자 재시작 시 구조를 스캔하여 남은 메시지를 위해 공간 맵을 다시 빌드합니다. 또한 스캔 시 데이터가 유실된 메시지가 있는지 확인하고 구조에서 이들을 삭제합니다. 또는 필요한 경우 나중에 삭제하도록 이들을 유실된 것으로 플래그 지정합니다.

이 재시작 스캔이 완료되면 데이터 세트 상태는 자동으로 RECOVERED에서 ACTIVE로 변경됩니다. 이때 다른 큐 관리자를 이를 다시 사용하여 시작할 수 있습니다.

공유 메시지 데이터 세트 사용량 정보

이제 DISPLAY USAGE 명령은 현재 열린 공유 메시지 데이터 세트에 대한 공유 메시지 데이터 세트 공간 및 버퍼 풀 사용량에 대한 정보도 표시합니다. 새 옵션 TYPE(SMDS) 또는 기존 옵션 TYPE(ALL)이 지정된 경우 이 정보가 표시됩니다.

공유 메시지 데이터 성능 및 용량 고려사항

모니터링 데이터 세트 사용법

각 보유 공유 메시지 데이터 세트의 현재 용량 백분율은 옵션 **TYPE(SMDS)**을 사용한 **DISPLAY USAGE** 명령으로 표시할 수 있습니다.

SMDS 정의에 대해 **DSEXPAND(YES)** 옵션이 적용되는 경우 큐 관리자는 공유 메시지 데이터 세트 사용이 90%에 도달하면 자동으로 이를 확장합니다. 이는 SMDS 옵션이 **DSEXPAND(YES)**로 설정되어 있거나 SMDS 옵션이 **DSEXPAND(DEFAULT)**로 설정되고 CFSTRUCT 기본 옵션이 **DSEXPAND(YES)**로 설정되어 있는 경우에 적용됩니다.

데이터 세트가 작성되었을 때 2차 할당 크기가 지정되지 않아 확장 시도가 실패할 경우(이유 코드 203과 함께 메시지 IEC070I 표시) 큐 관리자는 현재 크기에서 약 20% 정도의 오버라이드 2차 할당을 사용하여 확장 요청을 반복합니다.

데이터 세트가 확장되면 확장 프로세스의 한 부분으로서 새 데이터 세트 익스텐트가 포맷되며 보통 수십 초가 걸리는데, 매우 큰 익스텐트의 경우 수십 분이 걸릴 수도 있습니다. 포맷이 완료된 후에는 새 공간이 사용 가능해지며 새로 지정된 자주 쓰이는 제어 간격을 표시하기 위해 카탈로그가 업데이트됩니다.

새 메시지가 매우 빠르게 작성되는 경우 확장 프로세스가 끝나기 전에 기존 데이터 세트이 모두 찰 가능성이 있습니다. 이런 경우 공간을 할당할 수 없는 모든 요청은 확장 시도가 완료되어 새 공간이 사용 가능하게 될 때까지 일시 중단됩니다. 확장이 완료되면 요청은 자동으로 재시도됩니다.

사용 가능한 공간이 부족하거나 최대 익스텐트에 이미 도달하여 확장 시도가 실패할 경우 실패 이유를 제공하는 메시지가 발행된 후 추가 확장 시도를 방지하기 위해 영향을 받는 SMDS에 대한 오버라이드 옵션이 자

동으로 **DSEXPAND(NO)**로 변경됩니다. 이 경우 데이터 세트가 모두 찰 위험이 있으며 이 때에는 데이터 세트 용량 초과에 설명된 조치가 추가로 필요합니다.

모니터링 애플리케이션 구조 사용법

애플리케이션 구조의 전체 이름 (큐 공유 그룹 접두부 포함) 을 지정하는 MVS **DISPLAY XCF,STRUCTURE** 명령을 사용하여 애플리케이션 구조의 사용 레벨을 표시할 수 있습니다. IXC360I 응답 메시지가 요소와 항목의 현재 사용을 표시합니다.

구조 사용이 CFRM 정책에 지정되어 있는 **FULLTHRESHOLD** 값을 초과할 때 시스템은 메시지 IXC585E를 발행하며, 지정되었을 경우 항목 대 요소 비율을 변경하거나 구조 크기를 늘리는 자동 **ALTER** 조치를 수행할 수 있습니다.

버퍼 풀 크기 최적화

공유 버퍼 풀의 각 버퍼는 한 메시지에 대해 논리 블록 크기까지 페이지의 인접하는 범위를 읽거나 쓰는 데 사용됩니다. 만약 메시지가 바깥 블록으로 넘칠 경우 떨어진 블록의 각 페이지 범위는 분리된 버퍼를 필요로 합니다.

쓰기 또는 읽기 조작 후 메시지를 포함하고 있는 버퍼는 잠시 후 같은 데이터에 대한 다시 읽기 요청이 디스크로 갈 필요가 없도록 스토리지에 보유되어 최소 사용된 최근 사용(LRU, least-recently-used) 캐시 설계에서 재사용됩니다. 이는 같은 시스템에서 실행되고 있는 애플리케이션에서 공유 메시지를 기록하고 이를 곧 읽는 데 상당한 최적화를 제공합니다. 만약 다른 큐 관리자에서 보유하고 있는 메시지를 선택을 위해 찾아보았다가 검색할 경우에도 디스크에서 메시지를 다시 읽을 필요를 없애줍니다.

이는 각 애플리케이션 구조에 필요한 버퍼의 수는 애플리케이션 구조에 많은 양의 메시지를 기록하거나 읽는 각 동시 API 요청마다 하나, 여기에 후속 읽기 액세스를 최적화하기 위해 최근 액세스한 데이터를 저장하는 데 사용될 추가 버퍼가 몇 개 필요하다는 의미입니다.

공유 버퍼 풀의 경우 버퍼 수가 충분하지 않을 때 API 요청은 버퍼가 바로 사용 가능하지 않으면 대기합니다. 하지만 이 상황은 성능을 대폭 하락시킬 수 있기 때문에 피해야만 합니다.

공유 버퍼 풀에 대한 **DISPLAY USAGE** 명령의 통계는 현재 통계 간격 내에 버퍼 대기가 있는지 여부를 표시하고 사용 가능한 버퍼의 가장 낮은 수 (또는 임의의 시간에 버퍼를 대기한 최대 스레드 수를 나타내는 음수 값), 데이터를 저장한 버퍼 수 및 LRU 체인에서 버퍼 요청이 저장된 데이터를 성공적으로 찾은 횟수의 백분율 ("LRU 히트") 을 표시합니다. 읽지 않아도 됨 ("LRU 누락")[1].

- 대기가 발생한 경우에 버퍼의 개수는 증가되어야 합니다.
- 사용되지 않은 버퍼가 많은 경우 버퍼의 개수를 줄여 다른 목적을 위해 리전에서 사용 가능한 스토리지를 더 많이 확보할 수 있습니다.
- 저장된 데이터를 포함한 버퍼가 많으나 이 저장된 데이터에 대해 히트였던 읽기의 비율의 매우 적을 경우 스토리지를 더 유용한 다른 목적을 위해 사용하려면 버퍼의 수를 줄일 수도 있습니다. 그러나 버퍼의 개수가 빈 버퍼의 최소 개수보다 적어지면 대기를 발생시킬 수 있어 이렇게 해서는 안되며, 빈 버퍼 개수가 보통 0보다 훨씬 크도록 적당히 높아야만 합니다.

공유 메시지 데이터 세트 삭제

DELETE CFSTRUCT 명령 (구조의 모든 공유 큐가 비어 있고 닫혀 있는 경우에만 허용됨) 은 공유 메시지 데이터 세트 자체를 삭제하지 않지만 이 명령이 완료된 후 일반적인 방법으로 삭제할 수 있습니다. 동일한 데이터 세트를 공유 메시지 데이터 세트로 재사용하려는 경우 먼저 재형식화하여 빈 상태로 재설정해야 합니다.

공유 메시지 데이터 세트의 예외 상황

소프트웨어 및 하드웨어 오류가 발생하지 않은 상황에서도 정상 사용하는 중에 몇 가지 예외 상황이 발생할 수 있습니다.

[1] (Hits / (Hits+Misses))* 100

데이터 세트 용량 초과

데이터 세트가 모두 차고 확장될 수 없을 경우나 확장 시도가 실패하는 경우 해당 큐 관리자를 사용하여 해당 애플리케이션 구조에 큰 메시지를 기록하는 애플리케이션들은 오류 2192, MQRC_STORAGE_MEDIUM_FULL(MQRC_PAGESET_FULL으로도 알려짐)을 수신합니다.

데이터 세트는 데이터를 처리해야하는 애플리케이션이 실패하여 많은 양의 메시지 백로그가 쌓이게 됨으로 인해 가득 찰 수 있습니다. 이런 경우 데이터 세트를 이 이상 확장하는 것은 임시 해결책에 불과하며, 애플리케이션 처리가 최대한 빨리 다시 이뤄지도록 하는 것이 중요합니다.

더 많은 공간을 사용 가능하게 만들 수 있는 경우 **DSEXPAND(YES)** 또는 **DSEXPAND(DEFAULT)**(YES가 설정되었거나 CFSTRUCT 정의에 대해 **DSEXPAND**가 기본값으로 설정되었다고 가정하고)를 설정하는 데 **ALTER SMDS** 명령을 사용할 수 있습니다. 그러나 실패의 이유가 최대 익스텐트에 도달한 것이면 새 확장 시도가 메시지와 함께 거부되며 **DSEXPAND(NO)**가 다시 설정됩니다. 이 경우에는 재할당하는 것만이 이를 추가로 확장하는 유일한 방법이며, 여기에는 다음에 설명하는 대로 이를 임시로 사용 불가능하게 하는 것이 포함됩니다.

이동 또는 다시 할당되어야 하는 데이터 세트

데이터 세트가 이동 또는 확장되어야 하는데 정상 사용 중인 경우 이동 또는 할당을 위해 임시로 사용을 중단할 수 있습니다. 사용 불가능 할 때 데이터 세트를 사용하려고 시도하는 모든 API 요청은 이유 코드 MQRC_DATA_SET_NOT_AVAILABLE을 수신합니다.

1. 데이터 세트를 **ACCESS(DISABLED)**로 표시하려면 **RESET SMDS** 명령을 사용하십시오. 이는 연결되어 있는 모든 큐 관리자가 데이터 세트를 정상적으로 닫고 할당 해제하도록 합니다.
2. 액세스 메소드 서비스(AMS, access method services) **REPRO** 명령을 사용하는 등의 방법으로 이전 컨텐츠를 새로 할당한 데이터 세트로 복사하여 데이터 세트를 목적에 맞게 이동 또는 다시 할당하십시오.

   복사된 데이터가 포맷된 데이터 세트 끝에 추가되는 결과가 일어날 수 있으니 이전 데이터를 복사하기 전에 새 데이터 세트를 사전 포맷하려 시도하지 마십시오.
3. **RESET SMDS** 명령을 사용해 데이터 세트를 **ACCESS(ENABLED)**로 다시 표시하고 사용하십시오.

   이전 컨텐츠가 새 데이터 세트의 크기보다 작을 경우 나머지 공간에서는 새 데이터 세트가 열릴 때 자동으로 사전 포맷됩니다.

   이전 컨텐츠가 새 데이터 세트의 크기보다 클 경우 큐 관리자는 활성 데이터 손실이 없도록 커플링 기능 구조에서 메시지를 스캔하여 공간 맵을 다시 빌드해야 합니다. 새 익스텐트의 바깥에 있는 데이터 블록에 대한 참조가 발견될 경우 데이터 세트는 **STATUS(FAILED)**로 표시되며, 올바른 사이즈의 데이터 세트로 이를 교체하고 이전 데이터 세트를 안에 다시 복사하거나 **RECOVER CFSTRUCT**를 사용하여 지속 메시지를 복구하는 방법으로 이를 고쳐야만 합니다.

커플링 기능 구조 공간 부족

커플링 기능 구조에 공간이 부족할 경우 메시지 IXC585E가 발행되며 최대 용량의 데이터가 오프로드되도록 오프로드 규칙이 설정되었는지 확인할 필요가 있습니다. 설정되지 않은 경우 오프로드 규칙은 **ALTER CFSTRUCT** 명령으로 수정할 수 있습니다.

공유 메시지 데이터 세트의 오류 상황

정상 조작 상황에서는 일어나지 않으며 오류에 의해 발생하는 유념해야 할 몇 가지 문제점이 있습니다.

보유한 데이터 세트 열기 불가능

공유 메시지 데이터 세트를 소유하는 큐 관리자가 이를 할당하거나 열 수 없거나 데이터 세트 속성이 지원되지 않는 경우, 큐 관리자는 **ALLOCFAIL** 또는 **OPENFAIL** 의 적절한 **SMDSCONN** 상태 값을 설정하며 **SMDSCONN** 가용성을 **AVAIL(ERROR)**로 설정합니다. 또한 SMDS 가용성을 **ACCESS(SUSPENDED)**로 설정합니다. 오류가 정정되면 **RESET SMDS** 명령을 사용하여 재시도를 트리거하도록 **ACCESS(ENABLED)** 를 설정하거나 소유하는 큐 관리자에 대해 **START SMDSCONN** 명령을 실행하십시오.

읽기 전용 데이터 세트 열기 불가능

다른 큐 관리자가 소유하고 **STATUS(ACTIVE)**로 표시된 공유 메시지 데이터 세트를 큐 관리자가 할당하거나 열 수 없는 경우, 이는 데이터 세트 자체의 문제점이 아니라 데이터 세트(**SMDSCONN** 오브젝트로 표시됨)에 대한 연결의 특정 문제점 때문일 수 있다고 가정합니다.

SMDSCONN을 **STATUS(ALLOCFAIL)** 또는 **STATUS(OPENFAIL)**로 적절하게 표시하고 **SMDSCONN** 가용성을 AVAIL(ERROR)로 표시하여 더 이상 사용하려고 시도하지 못하도록 합니다.

문제점을 데이터 세트 자체에 영향을 주지 않고 정정할 수 있는 경우 재시도를 트리거시키는 데 **START SMDSCONN** 명령을 사용하십시오.

데이터 세트 자체에 문제가 있는 것으로 판명되면 **RESET SMDS** 명령을 사용하여 복구될 때까지 데이터 세트를 **STATUS(FAILED)** 로 표시할 수 있습니다. 데이터 세트가 복구되면 상태를 다시 **STATUS(ACTIVE)**로 변경하는 조치로 인해 다른 큐 관리자가 알림을 받습니다. **SMDSCONN**이 **AVAIL(ERROR)**로 표시되면 자동으로 **AVAIL(NORMAL)**로 다시 변경되어 데이터 세트를 열기 위한 새로운 시도를 트리거합니다.

데이터 세트 헤더 손상

데이터 세트가 성공적으로 열렸지만 헤더 정보의 형식이 올바르지 않은 경우 큐 관리자는 데이터 세트를 닫고 할당 해제하며 상태 세트를 **STATUS(FAILED)**로 설정하고 가용성을 **ACCESS(SUSPENDED)**로 설정합니다. 이는 컨텐츠를 복구하는 데 **RECOVER CFSTRUCT** 사용을 허용합니다.

데이터 세트에 다른 사용에서의 잔여 데이터가 포함되어 있으며 후속으로 사전 포맷되지 않아서 오류가 발생한 경우, 데이터 세트를 사전 포맷하고 **RESET SMDS** 명령을 사용하여 상태를 **STATUS(RECOVERED)**로 변경하십시오.

그렇지 않으면 데이터 세트를 복구해야 합니다.

예상치 못한 공 데이터 세트

큐 관리자가 **STATUS(ACTIVE)**로 표시된 데이터 세트를 열었지만 해당 데이터 세트가 초기화되지 않았거나 새로 사전 포맷되었지만 그 외에는 유효함을 발견하는 경우 큐 관리자는 공유 메시지 데이터 세트를 닫고 할당 해제한 후 상태를 **STATUS(FAILED)**로 설정하고 가용성을 **ACCESS(SUSPENDED)**로 설정합니다.

데이터 세트에 영구적 I/O 오류 발생

OPEN 처리 완료 후 데이터 세트에 영구적 I/O 오류가 있을 경우 이는 복구되어야 할 가능성이 높습니다. 큐 관리자는 현재 연결된 모든 큐 관리자가 닫고 할당 해제할 수 있도록 데이터 세트를 **STATUS(FAILED)**로 표시합니다.

데이터 세트에 복구 가능한 I/O 오류 발생

데이터 세트에 하드웨어 문제점이 있을 경우 이는 복구 가능한 I/O 오류를 일으킬 가능성이 높으며, 이는 큐 관리자에까지 영향을 미치지는 않으나 심각한 성능 저하를 일으키고 머지 않아 영구적 I/O 오류가 발생할 위험이 있음을 나타냅니다.

이 경우 **RESET SMDS** 명령을 사용하여 데이터 세트를 **STATUS(FAILED)**로 표시함으로써 복구를 위해 데이터 세트를 오프라인으로 설정할 수 있습니다. 이는 모든 큐 관리자가 이를 닫고 할당 해제하도록 하며 따라서 이는 다시 사용 가능하도록 되기 전에 새 볼륨으로 이동되는 등과 같은 조치의 대상이 될 수 있습니다.

데이터 세트가 이런 방식으로 사용 불가능이 될 경우 데이터 세트가 다시 사용 가능해지기 전에 데이터 세트 안에서 메시지를 찾고 공간 맵을 다시 빌드하기 위해 큐 관리자 연결 다시 시작 처리가 커플링 기능 구조를 스캔할 필요가 있도록 공간 맵이 저장되지 않습니다. 대안으로서 공유 메시지 데이터 세트가 아직 사용 가능한 경우 데이터 세트가 다시 사용 가능하도록 준비될 때까지 **RESET SMDS** 명령을 사용하여 이를 **ACCESS(DISABLED)**로 표시하여 좀 더 완만히 사용 불가능으로 만들 수 있습니다.

올바르지 않은 데이터 세트 컨텐츠

큐 관리자는 데이터 세트를 포함하는 볼륨이 백업에서 복구된 것이라는 등의 이유로 데이터 세트에 올바르지 않은 데이터가 포함되어 있거나 이 데이터가 최신이 아니라는 것을 직접 알아낼 수는 없습니다. 그러나 무결성 검사를 수행하여 이와 같은 오류가 애플리케이션 프로그램이 보는 올바르지 않은 메시지 데이터로 이어지는 경우가 거의 없도록 합니다.

무결성 검사 목적으로 데이터 세트의 각 메시지 블록의 접두부는 고유 시간소인을 포함한 해당 커플링 기능 항목 ID 의 사본으로 되어 있으며 이는 메시지 블록이 읽혀질 때마다, 메시지 데이터가 사용자 프로그램으로 전달되기 전에 검사됩니다. 메시지 블록 접두부와 항목 ID(그리고 이 동안 삭제되지 않은 커플링 기능 항목)가 일치하지 않는 경우, 메시지 블록은 손상되었으며 사용 불가능한 것으로 추측됩니다.

손상된 메시지가 지속적인 경우 데이터 세트는 **STATUS(FAILED)** 로 표시되며 구조 컨텐츠는 **RECOVER CFSTRUCT** 명령을 사용하여 복구해야 합니다. 비지속 메시지가 손상된 경우 이를 복구할 방법은 없으므로 진단 메시지가 발행되며 해당 커플링 기능 메시지 항목은 삭제됩니다.

데이터 세트가 열렸을 때 사용 가능한 저장된 공간 맵이 없는 경우 이는 데이터 세트의 데이터에 대한 참조를 커플링 기능 구조에서 스캔함으로서 다시 빌드됩니다. 스캔 작업 동안 큐 관리자는 다음과 같은 여러 조치를 수행합니다.

1. 큐 관리자는 현재 데이터 세트에 남아있는 가장 최근 메시지(있는 경우)의 위치를 판별합니다.
2. 큐 관리자는 블록 접두부가 메시지 항목 ID와 일치하는지 확인하기 위해 이 메시지를 데이터 세트에서 읽습니다.

이 조치는 큐 관리자가 데이터 세트가 다운 레벨인 경우를 감지하고 데이터 세트가 FAILED로 표시될 수 있게 해 줍니다. 그러나 이 검사는 데이터 세트가 이전 사본에서 복원되어 이후 새 메시지가 추가되지 않은 경우나 복원 후 추가된 모든 메시지가 이후 읽힌 뒤 삭제된 경우를 허용합니다.

데이터 세트가 정상적으로 닫힌 경우 다운 레벨 데이터로부터의 보호를 위해 큐 관리자는 다음과 같은 여러 조치를 수행합니다.

1. 큐 관리자는 데이터 세트가 정상적으로 닫힐 때 Db2 내에 있는 SMDS 오브젝트의 공간 맵 시간소인의 사본을 저장합니다.
2. 큐 관리자는 데이터 세트가 다시 열렸을 때 공간 맵 시간소인이 동일한지 확인합니다.

시간소인이 일치하지 않는 경우 이는 데이터 세트의 다운 레벨 사본이 사용되었음을 의미하며, 따라서 큐 관리자가 기존 공간 맵을 무시하며 이를 다시 빌드할 것이고, 이는 실제 손실된 메시지가 없을 때만 성공합니다.

참고: 이 무결성 검사는 이론상 가능한 모든 경우에서 다운 레벨 또는 손상된 데이터 세트를 감지할 수 있다고 보장하지는 않습니다. 예를 들어, 메시지 블록의 시작은 올바르나 나머지 데이터가 부분적으로 덮어 씌인 경우는 감지하지 못합니다.

공유 메시지 데이터 세트의 복구 시나리오

이 절은 공유 메시지 데이터 세트 복구 시나리오를 설명합니다.

손실된 데이터가 없을 경우의 데이터 세트 복구

몇몇 경우에는 실제 복구를 할 필요 없이 실패한 데이터 세트의 올바른 컨텐츠를 복구할 수 있습니다. 데이터 세트가 이전 사용으로부터 잔여 데이터를 포함하고 있으며 사전 포맷되지 않았을 경우를 예로 들 수 있으며 이 경우 이를 사전 포맷함으로서 해결할 수 있습니다. 또 다른 경우는 데이터 세트가 이동되었으나 데이터를 복사하는 중에 오류가 발생한 경우로 이 때는 데이터를 다시 올바르게 복사하는 것으로 해결할 수 있습니다.

이 경우 **RESET SMDS** 명령을 사용하여 **STATUS(RECOVERED)** 를 설정하면 정정된 데이터 세트를 다시 사용할 수 있습니다. 현재 가용성이 **ACCESS(SUSPENDED)** 인 경우에는 자동으로 다시 **ACCESS(ENABLED)** 로 설정됩니다.

소유하는 큐 관리자는 데이터 세트가 복구되었다는 알림을 받으면 구조 컨텐츠를 스캔하여 공간 맵을 재구성한 후 상태를 **STATUS(ACTIVE)** 로 변경합니다. 그리고 나면 다른 큐 관리자도 이 데이터 세트를 다시 읽기 시작할 수 있습니다.

TYPE(NORMAL)의 데이터 세트 복구

데이터 세트의 컨텐츠가 손실되었으나 애플리케이션 구조가 **RECOVER(YES)** 로 정의되어 있고 적절한 복구 로그가 사용 가능한 경우 **RECOVER CFSTRUCT** 명령을 사용하여 공유 메시지 데이터 세트로 오프로드된 지속 메시지를 포함한 구조에 저장된 모든 지속 메시지를 복구할 수 있습니다. 이 명령은 **BACKUP CFSTRUCT** 명령으로 로그된 정보와 백업 때부터 로그된 지속 메시지의 모든 변경사항을 사용하여 현재 상태를 복구합니다.

RECOVER CFSTRUCT 명령은 항상 커플링 기능 구조의 모든 지속 메시지와 Db2에 저장된 모든 오프로드된 메시지 데이터를 복구합니다. 공유 메시지 데이터 세트에 저장된 오프로드된 데이터의 경우 각 데이터 세트

는 이미 **STATUS(FAILED)**로 표시되어 있는 경우 또는 복구 프로세스에서 열었을 때 예상치 못하게 비어 있거나 유효하지 않은 것으로 확인된 경우에만 복구 처리 대상으로 선택됩니다. 활성으로 표시되어 있으며 유효성 검증을 통과한 모든 공유 메시지 데이터 세트는 기존 메시지 데이터 세트가 이미 올바르기 때문에 복구될 필요가 없지만, 저장된 모든 공간 맵은 복구 후 모두 다시 빌드되어야 할 필요가 있기 때문에 헤더는 업데이트됩니다.

복구 처리는 구조의 모든 컨텐츠가 복구 처리에 의해 다시 구성될 필요가 있어 구조가 실패로 표시되어 있을 경우에만 가능합니다. 그러나 적어도 하나의 공유 메시지 데이터 세트가 실패로 표시되어 있을 경우 **RECOVER CFSTRUCT** 명령은 복구 처리가 진행되도록 허용하기 위해 필요하면 자동으로 구조를 실패로 표시합니다.

복구는 관련 데이터 세트에 대한 쓰기 액세스 권한이 주어진 경우 큐 공유 그룹의 어떤 큐 관리자에 의해서도 수행될 수 있습니다.

지속 메시지만이 백업 및 로그되기 때문에 보통 복구 처리는 모든 지속 메시지를 복구하지만 구조 안의 모든 비지속 메시지를 손실시킵니다.

복구가 완료되면 복구 대상으로 선택된 모든 데이터 세트가 자동으로 **STATUS(RECOVERED)**로 변경되며 가용성이 **ACCESS(SUSPENDED)**였던 경우에는 **ACCESS(ENABLED)**로 변경됩니다. 큐 관리자는 커플링 기능의 메시지를 스캔하여 각 데이터 세트의 공간 맵을 다시 빌드하고 데이터 세트를 다시 사용될 수 있도록 **STATUS(ACTIVE)**로 표시합니다.

TYPE(PURGE)의 데이터 세트 복구

복구 가능한 구조에서 데이터 세트 컨텐츠가 손실되었으나 복구 로그를 사용할 수 없거나 복구가 너무 오래 걸리는 등과 같이 어떤 이유로 인해 복구가 불가능한 경우 구조를 사용 가능한 상태로 만들기 위해 **TYPE(PURGE)**으로 **RECOVER CFSTRUCT** 명령을 사용할 수 있습니다. 그러면 구조가 비어 있는 상태로 재설정되고 연관된 모든 데이터 세트가 **STATUS(EMPTY)**로 표시됩니다.

애플리케이션 구조 삭제

MVS **SETXCF FORCE** 명령을 사용하여 또는 구조 실패의 결과로 복구 불가능한 애플리케이션 구조가 삭제되면 다음에 구조가 연결될 때 구조가 재설정되고 모든 기존 메시지가 제거되었음을 알리는 CSQE028I 메시지가 발행되고 기존 데이터 세트도 자동으로 **STATUS(EMPTY)** 로 재설정됩니다. 이 조치는 구조 또는 이와 연관된 데이터 세트에서 데이터가 손실된 후 복구 불가능한 구조를 사용 가능하게 만듭니다.

복구 가능한 애플리케이션 구조가 삭제된 경우에는 구조가 실패한 경우와 같은 취급을 받습니다.

데이터 세트 복구 실패

로그 데이터 세트가 더 이상 사용 가능하지 않거나 큐 관리자가 복구가 진행되는 도중 종료된 경우와 같이 **RECOVER CFSTRUCT**가 어떤 이유에서 완료될 수 없는 경우 복구가 적어도 시작되기는 했었던 모든 데이터 세트에는 부분적 복구 시도가 있었음이 헤더에 표시되며 데이터 세트는 **STATUS(FAILED)** 상태로 남겨지게 됩니다.

이 경우 옵션은 원래 복구 요청을 반복하거나 **TYPE(PURGE)**로 복구하여 기존 데이터를 버리는 것입니다.

데이터 세트를 실제로 복구하지 않고 **STATUS(RECOVERED)**로 표시하려고 시도하면 다음에 열릴 때 큐 관리자는 헤더에 불완전한 복구가 표시되었음을 확인하고 해당 데이터 세트를 다시 **STATUS(FAILED)**로 표시합니다.

오프 사이트 재해 복구

오프 사이트 재해 복구의 경우 지속 공유 메시지는 로그와 CFSTRUCT 정의를 포함하고 있는 Db2 공유 메시지 그리고 연관된 SMDS 상태 정보를 사용해서만 다시 작성될 수 있습니다.

정의를 포함하는 Db2 테이블을 설정한 후 애플리케이션 구조 및 공유 메시지 데이터 세트는 빈 것으로 설정될 수 있습니다. 큐 관리자가 여기 연결하여 예상치 못하게 빈 것을 발견할 경우 관리자는 이를 실패로 표시하며, 여기에 해당되는 모든 구조의 모든 지속 메시지는 나중에 단일 **RECOVER CFSTRUCT** 명령을 사용해 복원할 수 있습니다.

z/OS *SMDS* 관련 명령

이 주제에서는 공유 메시지 데이터 세트와 관련된 명령을 설명하고 이에 대한 액세스를 제공합니다.

Display and alter the **CFSTRUCT** options relating to large message offload (**OFFLOAD** and offload rules) and shared message data sets (**DSGROUP, DSBLOCK, DSBUFS, DSEXPAND)**:

- DISPLAY CFSTRUCT
- DEFINE CFSTRUCT
- ALTER CFSTRUCT
- DELETE CFSTRUCT

큰 메시지 오프로드**(OFFLDUSE)**에 관한 **CFSTRUCT** 상태를 표시하십시오.

- DISPLAY CFSTATUS

개별 큐 관리자에 대한 오버라이드 데이터 세트 옵션(**DSEXPAND**와 **DSBUFS**)을 표시하고 변경하십시오.

- DISPLAY SMDS
- ALTER SMDS

큐 공유 그룹 내에 있는 데이터 세트의 상태와 가용성을 표시 또는 수정하십시오.

- CFSTATUS 유형(SMDS) 표시
- RESET SMDS

큐 관리자에 대한 SMDS 데이터 세트 공간 사용 및 버퍼 사용 정보를 표시하십시오.

- 사용 유형(SMDS) 표시

개별 큐 관리자에서 데이터 세트에 대한 연결(**SMDSCONN**)의 상태 및 가용성을 표시 또는 수정하십시오.

- DISPLAY SMDSCONN
- START SMDSCONN
- STOP SMDSCONN

필요할 경우 SMDS의 큰 메시지 데이터를 포함한 공유 메시지를 백업하고 복구하십시오.

- BACKUP CFSTRUCT
- RECOVER CFSTRUCT

z/OS 공유 큐 사용 이점

공유 큐를 사용하면 IBM MQ 애플리케이션에서 확장 가능성, 고가용성을 허용하고 워크로드 밸런싱을 구현할 수 있습니다.

공유 큐의 이점

복제된 서버가 단일 공유 큐에서 작업을 풀링하는 공유 큐 아키텍처는 다음과 같이 유용한 몇 가지 특성을 지닙니다.

- 서버 애플리케이션의 인스턴스를 추가하거나 애플리케이션 사본 및 큐 공유 그룹에 있는 큐 관리자를 포함하는 새 z/OS 이미지를 추가하여 확장 가능합니다.
- 고가용성을 보장합니다.
- 이는 자연적으로 큐 공유 그룹에 있는 각 큐 관리자의 사용 가능한 처리 용량에 기반하여 풀 워크로드 밸런싱을 수행합니다.

고가용성을 위해 공유 큐 사용

다음 예에서는 공유 큐를 사용하여 애플리케이션 가용성을 늘릴 수 있는 방법을 보여줍니다.

네트워크에서 실행 중인 클라이언트 애플리케이션이 z/OS에서 실행 중인 서버 애플리케이션의 요청을 작성하려는 IBM MQ 시나리오를 고려하십시오. 클라이언트 애플리케이션은 요청 메시지를 구성하고 요청 큐에 배치합니다. 그러면 클라이언트는 요청 메시지의 메시지 디스크립터에서 이름 지정된 응답 대상 큐로 보낸 서버의 응답을 기다립니다.

IBM MQ는 클라이언트 시스템에서 z/OS의 서버 입력 큐로의 요청 메시지 전송과 서버에서 다시 클라이언트로의 응답 전송을 관리합니다. 서버의 입력 큐를 공유 큐로 정의하면 큐에 넣은 모든 메시지는 큐 공유 그룹의 모든 큐 관리자에서 검색할 수 있습니다. 즉, SYSPLEX의 각 z/OS 이미지에서 큐 관리자를 구성할 수 있으며 동일한 큐 공유 그룹에 모두를 연결하여 이들 중 하나가 서버의 입력 큐에서 메시지에 액세스할 수 있습니다.

관리상의 이유로 이를 중지해야 하거나 큐 관리자 중 하나가 비정상적으로 종료되어도 서버 입력 큐에 있는 메시지는 계속 사용 가능합니다. 전체 z/OS 이미지를 오프라인으로 설정할 수 있으며, 메시지는 계속 사용 가능합니다.

공유 큐에서 이러한 메시지 가용성을 이용하려면 167 페이지의 그림 60에 표시된 대로 Sysplex의 각 z/OS 이미지에서 서버 애플리케이션의 인스턴스를 실행하여 더 높은 서버 애플리케이션 용량 및 가용성을 제공하십시오.

서버 애플리케이션의 한 인스턴스가 공유 큐에서 요청 메시지를 검색하고 컨텐츠에 기반하여 해당 처리를 수행하면 IBM MQ 메시지로 클라이언트에 다시 보내는 결과를 생성합니다. 응답 메시지는 요청 메시지의 메시지 디스크립터에 이름 지정된 응답 대상 큐 및 응답 대상 큐의 목적지로 전달됩니다.

리턴 경로를 구성하는 데 사용할 수 있는 여러 옵션이 있습니다. 해당 옵션에 대한 자세한 정보는 185 페이지의 『분산 큐잉 및 큐 공유 그룹』의 내용을 참조하십시오.

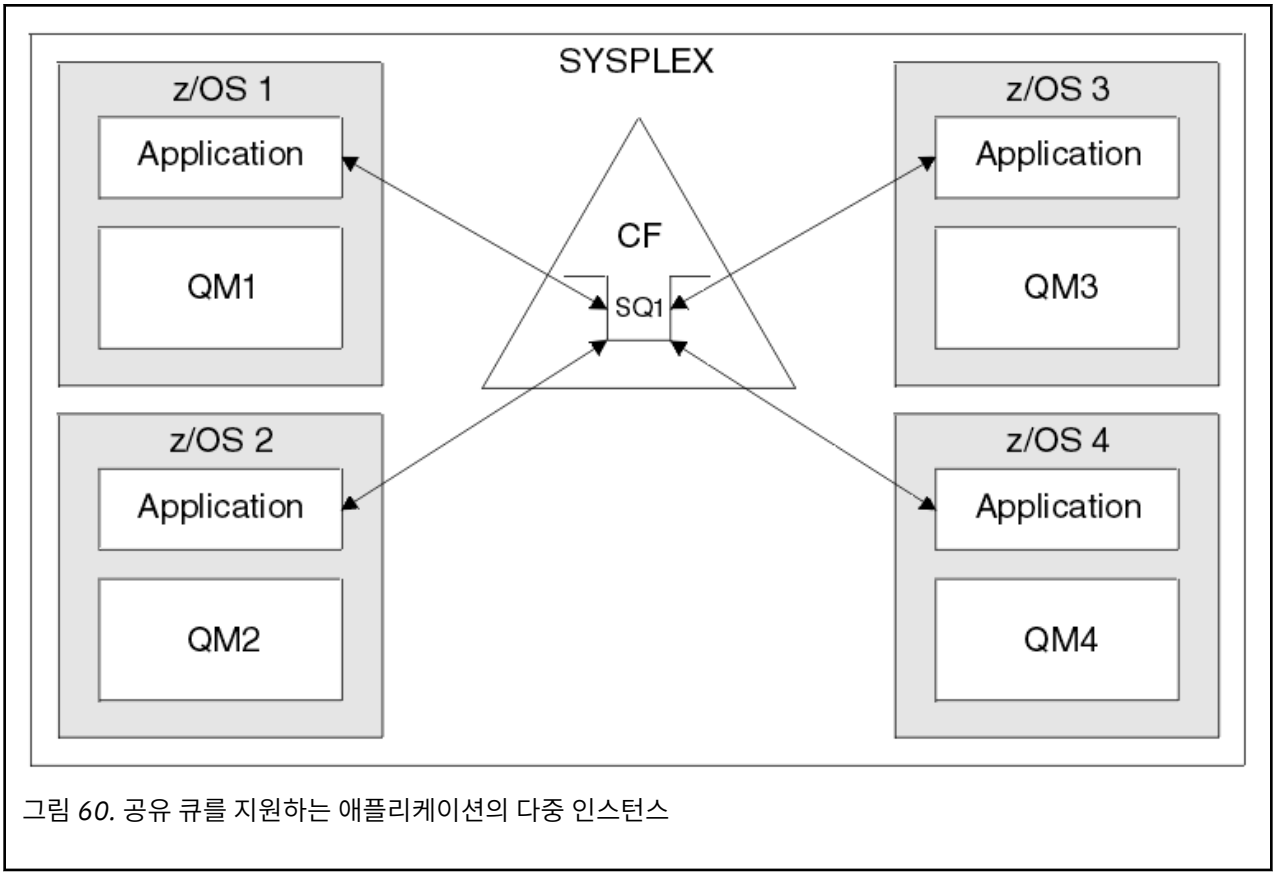


그림 *60.* 공유 큐를 지원하는 애플리케이션의 다중 인스턴스

피어 복구

큐 공유 그룹 내에서 메시지의 가용성을 더욱 강화시키기 위해 IBM MQ는 커플링 기능과 그룹 내의 다른 큐 관리자와의 비정상적 연결 끊김을 발견하고 가능한 경우 아직 보류 상태인 해당 큐 관리자의 작업 단위를 대신 완료합니다. 이 기능은 피어 복구라고도 합니다.

애플리케이션이 동기점의 큐에서 요청 메시지를 검색하는 시점에 큐 관리자가 비정상적으로 종료되었지만, 아직 응답 메시지를 넣거나 작업 단위를 커미트하지 않은 경우를 가정합니다. 큐 공유 그룹의 다른 큐 관리자가 실패를 감지하고 실패한 큐 관리자에서 수행 중인 불완전한 작업 단위를 백아웃합니다. 즉, 요청 메시지가 요청 큐로 다시 넣어지고 실패한 큐 관리자 재시작을 기다리지 않고도 처리할 다른 서버 인스턴스 중 하나에 대해 사용 가능해집니다.

IBM MQ에서 자동으로 작업 단위를 해석할 수 없는 경우, 수동으로 공유된 부분을 해석하여 큐 공유 그룹에 있는 다른 큐 관리자가 해당 작업을 계속 처리하도록 할 수 있습니다.

z/OS 공유 큐에서 스토리지 클래스 메모리의 사용

스토리지 클래스 메모리(SCM)의 사용은 IBM MQ for z/OS 공유 큐와 함께 사용하면 유리할 수 있습니다.

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

z13, zEC12 및 zBC12 시스템에서 Flash Express 카드를 설치할 수 있습니다. 이러한 카드에는 플래시 SSD(Solid-State Drive)가 포함되어 있습니다. 설치 후, 일반적으로 SCM으로 알려져 있는 하나 이상의 LPAR에 카드의 플래시 스토리지를 할당할 수 있습니다.

SCM은 I/O 대기 시간 및 비용 두 가지 측면에서 실제 스토리지와 DASD(Direct Access Storage Device) 사이에 위치합니다. SCM은 이동 부분이 없기 때문에 DASD보다 훨씬 더 낮은 I/O 대기 시간을 나타냅니다.

또한 SCM은 실제 스토리지보다 비용이 훨씬 더 적게 듭니다. 그 결과 상대적으로 낮은 비용으로 대용량 스토리지를 설치할 수 있습니다. 예를 들어, Flash Express 카드 쌍에는 1424GB의 사용 가능한 스토리지가 포함되어 있습니다.

이러한 특성은 단기간에 실제 스토리지에서 대용량 데이터를 가져와야 하는 경우에 SCM이 유용하다는 것을 의미합니다. 해당 데이터가 DASD에 기록될 때보다 SCM에 기록될 때의 속도가 훨씬 더 빠르기 때문입니다. IBM MQ 공유 큐를 포함하는 커플링 기능(CF) 목록 구조를 사용하는 경우 이 특정 지점이 매우 유용할 수 있습니다.

목록 구조가 가득 차는 이유

CF 구조는 정의될 때 구조의 최대 크기에 대해 설명하는 SIZE 속성으로 구성됩니다. CF 구조는 항상 실제 스토리지에 영구적으로 상주하기 때문에 CF에 정의된 구조의 SIZE 속성 합계는 CF에 할당된 실제 스토리지의 크기보다 작아야 합니다.

결과적으로, 추가 구조가 CF에 잘 맞도록 하기 위해 지정된 구조의 SIZE 값을 가능한 최소값으로 유지하기 위한 지속적 압력이 있습니다. 하지만 구조를 너무 작게 만들면 구조가 가득 차서 해당 구조를 사용하는 애플리케이션 또는 서브시스템이 손상될 수 있기 때문에 구조가 해당 목적을 달성할 수 있을 만큼 크면 충돌 압력이 발생합니다.

예상 사용량을 기반으로 구조를 정확하게 크기 조정하는 것이 꼭 필요합니다. 하지만 이 태스크는 시간이 지나면서 워크로드가 변경될 수 있기 때문에 수행하기가 어렵고 워크로드의 변동을 고려하기가 쉽지 않습니다.

IBM MQ 공유 큐는 CF 목록 구조를 사용하여 메시지를 저장합니다. IBM MQ는 메시지 및 애플리케이션 구조가 포함된 CF 구조를 호출합니다.

애플리케이션 구조는 IBM MQ CFSTRUCT 오브젝트에 저장된 정보를 사용하여 참조됩니다. 63KB보다 작은 메시지를 공유 큐에 넣으면, 메시지가 완전히 단일 목록 입력 항목 및 0개 이상의 목록 요소로 애플리케이션 구조에 저장됩니다.

IBM MQ 공유 큐가 목록 구조를 사용하기 때문에, 설명된 압력이 공유 큐에도 영향을 줍니다. 이 경우 공유 큐에 저장될 수 있는 최대 메시지 수는 다음의 함수입니다.

- 큐의 메시지 크기
- 구조의 최대 크기
- 구조에서 사용 가능한 입력 항목 및 요소의 수

최대 512개의 공유 큐가 동일한 구조를 사용하고 입력 항목과 요소에 대해 효과적으로 경쟁할 수 있기 때문에, 이것은 문제를 훨씬 더 복잡하게 만듭니다.

IBM MQ 큐는 애플리케이션 간 데이터 전송에 사용되며, 따라서 일반적인 상황은 해당 메시지를 가져와야 할 파트너 애플리케이션이 실행 중이 아닐 때 큐에 메시지를 넣는 애플리케이션입니다.

이 상황이 발생하면, 다음 상황 중 하나 이상이 발생할 때까지 시간이 지나면서 큐의 메시지 수가 늘어납니다.

- 넣기 애플리케이션이 메시지 넣기를 중지합니다.

- 가져오기 애플리케이션이 메시지 가져오기를 시작합니다.
- 큐의 기존 메시지가 만료되기 시작하고 큐에서 제거됩니다.
- MQRC_Q_FULL 이유 코드가 넣기 애플리케이션으로 리턴되는 경우에 큐가 해당 최대 용량에 도달합니다.
- 공유 큐를 포함하는 구조가 해당 최대 크기에 도달하거나 구조를 포함하는 CF에서 사용 가능한 스토리지가 부족합니다. 어떠한 경우에도 MQRC_STORAGE_MEDIUM_FULL 이유 코드가 넣기 애플리케이션으로 리턴됩니다.

마지막 세 가지 경우에는 큐가 가득 찹니다. 이 시점에서는 메시지가 이동할 데가 없기 때문에 넣기 애플리케이션에 문제점이 발생합니다. 넣기 애플리케이션은 일반적으로 다음 솔루션 중 하나 이상 사용하여 이 문제점을 해결합니다.

- 선택적으로 재시도 사이에 지연을 두고 메시지 넣기를 반복적으로 재시도하십시오.
- 메시지를 다른 위치에 넣으십시오(예: 데이터베이스 또는 파일). 메시지는 나중에 액세스하여 정상적으로 큐에 넣을 수 있습니다.
- 메시지가 비지속적이면 메시지를 제거하십시오.

그러나 일부 애플리케이션 클래스(예: 많은 양의 수신 메시지가 있는 애플리케이션 또는 파일 시스템에 대한 액세스 권한이 없는 애플리케이션)의 경우에는 이러한 솔루션이 실현 가능하지 않습니다. 큐가 첫 번째 위치에서 가득 차지 않거나 가득 찰 가능성이 거의 없도록 보장할 필요가 있으며, 이는 특히 공유 큐에 관련됩니다.

SMDS 및 오프로드 규칙

IBM WebSphere MQ 7.1에 도입된 오프로드 규칙은 애플리케이션 구조를 가득 채울 가능성을 줄이는 방법을 제공합니다.

각 애플리케이션 구조는 세 쌍의 키워드를 사용하여 지정되는 세 가지 규칙이 연관되어 있습니다.

- OFFLD1SZ 및 OFFLD1TH
- OFFLD2SZ 및 OFFLD2TH
- OFFLD3SZ 및 OFFLD3TH

각 규칙은 애플리케이션 구조와 연관된 스토리지 메커니즘에 메시지 데이터를 오프로드하기 위해 충족해야 할 조건을 지정합니다. 현재 두 가지 유형의 스토리지 메커니즘이 사용 가능합니다.

- Db2
- VSAM(Virtual Storage Access Method) 선형 데이터 세트 그룹(IBM MQ에서는 공유 메시지 데이터 세트(SMDS)라고 함)

다음 예는 DEFINE CFSTRUCT 명령을 사용하여 LIST1라는 애플리케이션 구조를 작성하는 MQSC 명령을 보여줍니다.

이 구조는 기본 오프로드 규칙이 준비되어 있고 SMDS를 오프로드 메커니즘으로 사용합니다. 이는 구조가 70% 채워지면(OFFLD1TH) 32KB 이상인 모든 메시지(OFFLD1SZ)가 SMDS로 오프로드됨을 의미합니다.

마찬가지로, 구조가 80% 채워지면(OFFLD2TH) 4KB 이상인 모든 메시지(OFFLD2SZ)가 오프로드됩니다. 구조가 90% 채워지면(OFFLD3TH) 모든 메시지(OFFLD3SZ)가 오프로드됩니다.

```
DEFINE CFSTRUCT(LIST1)
CFLEVEL(5)
OFFLOAD(SMDS)
OFFLD1SZ(32K) OFFLD1TH(70)
OFFLD2SZ(4K) OFFLD2TH(80)
OFFLD3SZ(0K) OFFLD3TH(90)
```

오프로드된 메시지는 오프로드 매체에 저장되고, 메시지 포인터는 구조에 저장됩니다. 오프로드 규칙이 스토리지가 부족할 때 구조에 메시지 데이터를 더 적게 넣는 방법으로 구조가 가득 찰 가능성을 낮추는 동안에도 일부 데이터(즉, 오프로드된 메시지에 대한 포인터)는 각 메시지의 구조에 기록됩니다.

또한 성능 비용에는 오프로드 규칙이 따라옵니다. 구조에 메시지 쓰기는 상대적으로 속도가 빠르며 CF에 쓰도록 요청을 송신하는 데 소요되는 시간에 크게 좌우됩니다. 구조에 실제 쓰기는 실제 스토리지 속도로 빠르게 수행됩니다.

SMDS에 메시지 쓰기는 구조에 메시지 포인터 쓰기와 SMDS에 메시지 데이터 쓰기를 포함하므로 훨씬 더 느립니다. 이 두 번째 쓰기 조작은 DASD 속도로 수행되고 대기 시간을 추가할 가능성이 있습니다. Db2가 오프로드 메커니즘으로 사용되는 경우 성능 비용은 훨씬 더 큽니다.

z/OS *IBM MQ for z/OS*에서 스토리지 클래스 메모리의 작동 방법

IBM MQ for z/OS 공유 큐에서 스토리지 클래스 메모리(SCM) 사용의 개요

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

CFLEVEL 19 이상의 커플링 기능(CF)에는 SCM이 할당되어 있을 수 있습니다. 이 경우 해당 CF에 정의된 구조는 SCM을 이용하여 구조가 가득 찰 가능성(구조 가득 참 조건이라 함)을 줄이도록 구성할 수 있습니다. SCM을 이용하도록 구성된 구조가 시스템 판별 지점을 넘어서서 가득 차면, CF는 구조에서 SCM으로 데이터 이동을 시작하여 새 데이터의 구조에서 공간을 비웁니다.

참고: SCM 자체가 가득 찰 수 있기 때문에 구조에 SCM을 할당하면 구조 가득 참 조건의 가능성은 줄어들지만 이 조건 발생의 가능성은 완전히 제거되지 않습니다.

구조는 해당 구조의 정의를 포함하는 커플링 기능 자원 관리자(CFRM) 정책에 **SCMALGORITHM** 및 **SCMMAXSIZE** 키워드를 둘 다 지정하여 SCM을 사용하도록 구성됩니다.

이러한 키워드가 지정되고 CFRM 정책이 적용된 후에는 구조를 다시 빌드하거나 해당 키워드가 적용될 수 있도록 할당 취소해야 합니다.

SCMALGORITHM 키워드

SCM의 입출력(I/O) 속도는 실제 스토리지의 입출력(I/O) 속도보다 더 느리기 때문에 CF는 SCM에 쓰기 또는 SCM에서 읽기의 영향을 줄이기 위해 구조의 예상 사용에 맞게 조정되는 알고리즘을 사용합니다.

알고리즘은 구조의 CFRM 정책에서 **SCMALGORITHM** 키워드로 *KEYPRIORITY1* 값을 사용하여 구성됩니다. IBM MQ 공유 큐에서 사용되는 목록 구조에만 *KEYPRIORITY1* 값을 사용해야 합니다.

KEYPRIORITY1 알고리즘은 대부분의 애플리케이션이 공유 큐에서 우선순위대로 메시지를 가져온다는 가정 하에 작동합니다. 즉, 애플리케이션이 메시지를 가져올 때 우선순위가 가장 높은 가장 오래된 메시지를 가져옵니다.

구조가 시스템에 의해 정의된 90%의 임계값 이상으로 채워지기 시작하면, CF는 다음에 가져올 가능성이 가장 적은 메시지를 비동기적으로 마이그레이션하기 시작합니다. 이러한 메시지는 최근에 큐에 넣어서 우선순위가 낮은 메시지입니다.

이와 같이 구조에서 SCM으로 메시지를 비동기 마이그레이션하는 작업은 "사전 스테이징"으로 알려져 있습니다.

사전 스테이징은 SCM에 대한 동기 입출력(I/O) 발생 중에 애플리케이션이 차단될 가능성을 줄여서 SCM 사용의 성능 비용을 절감합니다.

사전 스테이징 외에도 *KEYPRIORITY1* 알고리즘은 충분한 여유 공간이 사용 가능할 때 SCM에서 메시지를 다시 가져오고 구조로 메시지를 다시 보내는 작업을 비동기적으로 수행합니다. *KEYPRIORITY1* 알고리즘의 경우, 이는 구조가 70% 이하로 채워져 있음을 의미합니다.

SCM에서 구조로 메시지를 가져오는 행위는 "프리페칭"으로 알려져 있습니다.

프리페칭은 애플리케이션이 SCM에 사전 준비된 메시지를 가져오려고 할 때 CF가 동기적으로 해당 메시지를 구조로 다시 돌려주는 동안 대기해야 할 가능성을 줄입니다.

SCMMAXSIZE 키워드

SCMMAXSIZE 키워드는 구조에 사용될 수 있는 SCM의 최대 크기를 정의합니다. SCM은 필요할 때 CF로 구조에 할당되기 때문에, 사용 가능한 SCM의 총 크기보다 큰 **SCMMAXSIZE**를 지정할 수 있습니다. 이는 "초과 커미트"로 알려져 있습니다.

중요사항: SCM을 과도하게 커미트하지 마십시오. 과도하게 커미트하면, 이에 영향을 받는 애플리케이션이 예상한 대로 작동하지 않습니다. 예를 들어, 공유 큐를 사용하는 IBM MQ 애플리케이션은 예상치 못한 MQRC_STORAGE_MEDIUM_FULL 이유 코드를 가져올 수 있습니다.

CF는 다양한 데이터 구조를 사용하여 SCM 사용을 추적합니다. 이러한 데이터 구조는 CF에 할당되는 실제 스토리지에 상주하며, 결과적으로 구조에 사용될 수 있는 실제 스토리지의 크기를 줄입니다. 이러한 데이터 구조에 사용되는 스토리지는 "기능 보강된 공간"으로 알려져 있습니다.

구조가 SCM으로 구성된 경우, CF에서 적은 양의 실제 스토리지가 고정 기능 보강된 공간으로 알려진 구조로 할당됩니다. 이는 구조가 SCM을 사용하지 않는 경우에만 할당됩니다. 구조의 데이터가 SCM에 저장될 때, 동적 기능 보강된 공간이 CF의 예비 실제 스토리지에서 추가로 할당됩니다.

SCM에서 데이터가 제거되면, 동적 기능 보강된 공간이 CF로 리턴됩니다. 기능 보강된 공간(고정 또는 동적)은 구조에 할당되는 실제 스토리지에서 가져오지 못합니다.

기능 보강된 스토리지 외에도, 구조가 SCM을 사용하도록 구성된 경우 해당 구조에 사용되는 제어 스토리지의 크기가 증가합니다. 이는 SCM이 구성되지 않은 동일 크기의 구조보다 SCM으로 구성된 목록 구조에 입력 항목 및 요소가 더 적게 포함되어 있을 수 있음을 의미합니다.

SCM이 새 구조 또는 기존 구조에 미치는 영향을 이해하려면 CFSizer 도구를 사용하십시오.

마지막으로 주목해야 할 중요한 사항으로서, 구조에서 SCM으로 데이터가 이동되고 동적 기능 보강된 공간이 사용된 후에는 수동이든 자동이든 구조를 변경할 수 없습니다.

즉, 구조에 할당된 스토리지 크기를 늘리거나 줄일 수 없고, 구조에 사용되는 입력 항목 대 요소 비율 등도 변경할 수 없습니다. 구조를 다시 변경 가능하게 하려면, SCM에 저장된 데이터가 구조에 없어야 하고 구조가 동적 기능 보강된 스토리지를 이용 중이 아니어야 합니다.

z/OS *SCM 사용 이유*

응급 스토리지 및 향상된 성능은 IBM MQ for z/OS에서 SCM을 사용하기 위한 두 가지 유스 케이스입니다.

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

이 절에서는 두 가지 가능한 시나리오의 배경이 되는 이론을 소개합니다. 시나리오 설정 방법에 대한 추가적인 세부사항은 다음을 참조하십시오.

- 174 페이지의 『응급 스토리지 - 기본 구성』
- 180 페이지의 『향상된 성능 - 기본 구성』

중요사항: CF 구조를 가진 SCF의 사용은 IBM MQ의 특정 버전에 의존하지 않습니다. 하지만 응급 스토리지 시나리오에서는 SMDS 및 오프로드 규칙이 필요하므로 IBM WebSphere MQ 7.1 이상에서만 작동합니다.

응급 스토리지

SMDS 및 메시지 오프-로딩을 SCM과 함께 사용하면 오랜 가동 중단 동안 MQRC_STORAGE_MEDIUM_FULL 이유 코드가 IBM MQ 애플리케이션으로 리턴될 가능성을 줄일 수 있습니다.

개요

단일 공유 큐가 애플리케이션 구조에 구성됩니다. 넣기 애플리케이션은 공유 큐에 메시지를 넣고, 가져오기 애플리케이션은 공유 큐에서 메시지를 가져옵니다.

정상 실행 중에 큐 용량은 거의 0일 것으로 예상되지만, 비즈니스 요구사항에 따라 시스템은 두 시간 동안 가져오기 애플리케이션의 가동 중단을 허용할 수 있어야 합니다. 이는 공유 큐에 두 시간 동안 넣기 애플리케이션에서 생성되는 메시지가 포함될 수 있어야 함을 의미합니다.

현재 이 프로세스는 기본 오프로드 규칙 및 SMDS를 사용하여 오프-로딩과 연관되는 성능 비용을 줄이면서 구조의 크기가 최소화되도록 수행됩니다.

공유 큐에 송신되는 메시지의 비율은 단기에서 중기까지 두 배가 될 것으로 예상됩니다. 시스템은 여전히 두 시간의 가동 중단을 허용할 수 있어야 하지만, CF에서 구조의 크기를 두 배로 만들기 위해 사용 가능한 실제 스토리지가 충분하지 않습니다.

애플리케이션 구조를 포함하는 CF가 zEC12 시스템에 상주하기 때문에, 충분한 SCM을 구조와 연관시켜 두 시간의 가동 중단이 허용될 수 있도록 충분한 메시지를 저장할 수 있습니다.

일정 기간 동안 발생하는 상황을 고려하십시오.

1. 처음에는 시스템이 안정 상태에 있습니다. 넣기 및 가져오기 애플리케이션은 모두 정상적으로 실행 중이고 큐 용량은 0이거나 0에 가깝습니다. 그 결과 애플리케이션 구조는 주로 비어 있습니다.
2. 특정 시간에 가져오기 애플리케이션은 예상치 못한 실패가 발생하여 중지됩니다. 넣기 애플리케이션은 계속 메시지를 큐에 넣고, 애플리케이션 구조가 채워지기 시작합니다.
3. 구조가 전체의 70%에 도달하면, 첫 번째 오프로드 규칙의 조건이 충족되어 크기가 32KB 이상인 메시지가 모두 SMDS로 오프로드됩니다.

   오프로드 규칙의 개요는 169 페이지의 『SMDS 및 오프로드 규칙』의 내용을 참조하십시오.
4. 메시지를 계속 공유 큐에 넣는 동안은 (구조에 저장되는 메시지 데이터 때문에 또는 구조에 저장되는 오프로드된 메시지를 가리키는 포인터 때문에) 구조가 계속 채워집니다.

   구조가 전체의 80%까지 차면, 두 번째 오프로드 규칙이 적용되기 시작하여 4KB 이상인 메시지가 SMDS로 오프로드됩니다.
5. 구조가 전체의 90% 이상 차면, 모든 메시지가 SMDS로 오프로드되고 구조에는 메시지 포인터만 놓이게 됩니다.

   이 때쯤이면 사전 스테이징 알고리즘이 실행되기 시작되고 구조에서 SCM으로 데이터 이동을 시작합니다. 큐에 있는 모든 메시지의 우선순위가 동일하다고 가정할 때, 최신 메시지가 사전 준비됩니다.

   모든 메시지가 지금 SMDS로 오프로드되기 때문에, SCM으로 이동 중인 데이터는 실제 메시지 데이터가 아니고 SMDS의 메시지에 대한 포인터입니다.

   결과적으로 구조 및 구조와 연관된 SCM 및 SMDS의 조합에 저장할 수 있는 메시지 수는 매우 큽니다.

   성능: 이 가동 중단 스테이지에서는 넣기 애플리케이션이 SMDS에 써야 하기 때문에 어느 정도 성능 저하가 있을 수 있습니다. 이 경우 SCM 사용은 성능의 관점에서 넣기 애플리케이션의 제한 요인이 아니어야 합니다. SCM은 구조를 가득 채우지 못하도록 추가 공간을 제공합니다.
6. 결국 가져오기 애플리케이션은 다시 사용 가능해져서 가동 중단이 끝납니다.

   하지만 SCM은 구조에서 계속 사용 중입니다. 가져오기 애플리케이션은 큐에서 메시지를 읽기 시작하여 가장 오래되고 우선순위가 가장 높은 메시지를 맨 먼저 가져옵니다.

   이러한 메시지는 기록된 후에 구조가 채워지기 시작했으므로 전적으로 구조의 실제 스토리지 부분에서 나옵니다.
7. 구조가 비워지기 시작하면서 사전 스테이징이 활성화되는 임계값 아래로 내려가면 사전 스테이징이 중지됩니다.
8. 구조 사용량이 오프로드 규칙이 적용되는 지점 미만으로 줄어들면, 메시지가 63KB를 초과하지 않는 한 더 이상 SMDS로 오프로드되지 않습니다.

   이 때쯤에 프리페치 알고리즘은 SCM에서 구조로 데이터를 이동하기 시작합니다. 가져오기 애플리케이션은 SCM 알고리즘에 따라 예상되는 순서로 큐에서 메시지를 가져오기 때문에 가져오기 애플리케이션에서 필요로 하기 전에 메시지가 준비됩니다.

   이 결과로 가져오기 애플리케이션은 SCM에서 메시지를 동기적으로 가져올 때까지 대기할 필요가 없습니다.
9. 가져오기 애플리케이션은 큐 아래로 계속 이동하면서 SMDS로 오프로드된 메시지를 검색하기 시작합니다.
10. 최종적으로 시스템은 다시 안정 상태가 됩니다. SCM 또는 SMDS에는 메시지가 저장되지 않고 큐 용량은 거의 0입니다.

개선된 성능

이 시나리오에서는 SMDS 사용에 따른 성능 비용을 발생시키지 않고 공유 큐에 저장할 수 있는 메시지의 수를 늘리도록 SCM을 사용하는 방법에 대해 설명합니다.

설명

이 시나리오의 경우, 넣기 및 가져오기 애플리케이션은 애플리케이션 구조에 저장되는 공유 큐를 통해 통신합니다.

넣기 애플리케이션은 단시간 내에 많은 수의 메시지를 넣을 때 버스트로 실행되는 경향이 있습니다. 그런 다음 장시간 동안 메시지를 생성하지 않습니다.

가져오기 애플리케이션은 각 메시지를 순차적으로 처리하고 각 메시지에 대해 복잡한 처리를 수행합니다. 결과적으로 대부분의 시간에서 큐 용량이 0이지만, 넣기 애플리케이션이 실행되기 시작할 때에는 예외입니다. 이 경우, 메시지를 가져올 때보다 넣을 때 속도가 더 빠르기 때문에 큐 용량이 증가하기 시작합니다.

큐 용량은 넣기 애플리케이션이 중지될 때까지 증가하고, 가져오기 애플리케이션은 큐의 모든 메시지를 처리하기에 충분한 시간이 있습니다.

참고:

1. 이 시나리오에서 핵심 요인은 성능입니다. 큐로 송신되는 메시지는 항상 63KB 미만이므로 SMDS로 오프로드할 필요가 없습니다.
2. 애플리케이션 구조는 단일 "버스트" 에서 넣기 애플리케이션에 의해 배치될 모든 메시지를 포함하기에 충분히 크도록 크기가 조정되었습니다.
3. 구조가 채워지기 시작할 때에도 메시지가 SMDS로 오프로드되지 않도록 하기 위해 오프로드 규칙을 모두 사용 안함으로 설정해야 합니다. 이는 SMDS에 메시지를 쓰고 SMDS에서 메시지를 읽는 작업과 연관되는 성능 비용이 허용 불가능하다고 간주되기 때문입니다.

시간이 지나면서 넣기 애플리케이션이 버스트로 송신하는 메시지 수는 몇 배로 증가해야 합니다. 가져오기 애플리케이션은 각 메시지를 순차적으로 처리해야 하므로 구조가 가득 차는 지점까지 큐의 메시지 수가 증가합니다.

이 시점에서 넣기 애플리케이션은 메시지를 넣을 때 이유 코드(MQRC_STORAGE_MEDIUM_FULL)를 수신하고, Put 조작이 실패합니다. 넣기 애플리케이션은 메시지를 큐에 넣을 수 없는 기간을 잠시 동안만 허용할 수 있습니다. 이 기간이 너무 길면 애플리케이션이 종료됩니다.

넣기 애플리케이션 또는 가져오기 애플리케이션을 다시 작성할 수 있는 시간 또는 스킬이 없다고 가정할 때 이 문제점을 해결할 수 있는 세 가지 솔루션이 있습니다.

1. 애플리케이션 구조의 크기를 증가시킵니다.
2. 큐가 채워지기 시작할 때 메시지가 SMDS로 오프로드되도록 애플리케이션 구조에 오프로드 규칙을 추가합니다.
3. 구조와 SCM을 연관시킵니다.

첫 번째 솔루션은 구현이 빠르지만 CF에서 사용 가능한 실제 스토리지가 충분하지 않습니다.

또한 두 번째 솔루션도 구현은 빠르지만 SMDS로 오프-로딩 시 성능 영향이 이 옵션을 사용하기에 너무 큰 것으로 간주됩니다.

구조와 SCM을 연관시키는 세 번째 솔루션은 허용 가능한 비용과 성능의 밸런스를 제공합니다.

SCM을 구조와 연관시키면 가져오기 조작에서 사용한 기능 보강된 스토리지로 인해 CF에서 실제 스토리지 사용이 더 늘어납니다. 그러나 실제 스토리지의 실제 크기는 첫 번째 옵션에서 사용된 크기 미만이 됩니다.

다른 고려사항은 SCM의 비용입니다. 그러나 이 비용은 실제 스토리지보다 훨씬 더 저렴합니다. 이러한 요인을 결합하여 세 번째 옵션을 첫 번째 옵션보다 저렴하게 만들 수 있습니다.

첫 번째 옵션과 마찬가지로 세 번째 옵션도 수행되지 않을 가능성이 있지만 성능의 차이가 허용 가능하거나 경우에 따라 무시해도 될 정도가 되도록 CF에서 사용되는 프리페치 및 사전 스테이징 알고리즘을 결합할 수 있습니다.

확실히 SMDS를 사용하여 메시지를 오프로드하는 것보다는 성능이 훨씬 더 향상될 수 있습니다.

일정 기간 동안 발생하는 상황을 고려하십시오.

1. 처음에는 가져오기 애플리케이션이 활성 상태이고 메시지가 공유 큐로 전달될 때까지 대기합니다. 넣기 애플리케이션은 활성 상태가 아니고 공유 큐는 비어 있습니다.
2. 특정한 시간에 넣기 애플리케이션이 활성화되고 많은 수의 메시지를 공유 큐에 넣기 시작합니다. 가져오기 애플리케이션은 메시지를 가져오기 시작하지만, 가져오기 애플리케이션이 넣기 애플리케이션보다 더 느리기 때문에 큐 용량이 급격히 증가하기 시작합니다.

   결과적으로 애플리케이션 구조가 채워지기 시작합니다.
3. 시간이 증가하는 동안에도 넣기 애플리케이션은 활성 상태입니다. 애플리케이션 구조는 약 90%까지 채워집니다.

   이 때에 SCM 사전 스테이징 알고리즘이 구조에서 SCM으로 메시지를 이동하기 시작하여 구조의 공간을 비웁니다.

   가져오기 애플리케이션은 가장 오래되고 우선순위가 가장 높은 메시지를 큐에서 먼저 가져오기 때문에 항상 구조에서 메시지를 가져오며 SCM에서 구조로 메시지를 동기적으로 가져올 때까지 대기할 필요가 없습니다. i
4. 넣기 애플리케이션은 계속 활성 상태이고 메시지를 공유 큐에 넣습니다. 그러나 구조와 잘 맞지 않는 메시지를 모두 저장할 수 있는 충분한 공간이 SCM에 있기 때문에 애플리케이션은 MQRC_STORAGE_MEDIUM_FULL 이유 코드를 수신하지 않습니다.
5. 결국 넣을 메시지가 더 이상 없기 때문에 넣기 애플리케이션이 중지됩니다.

   구조가 사용 중에는 90% 밑으로 떨어지기 때문에 사전 스테이징 알고리즘이 중지되며, 가져오기 애플리케이션은 큐에서 메시지를 계속 처리합니다.
6. 가져오기 애플리케이션이 구조에서 공간을 비우기 시작할 때, 프리페치 알고리즘은 SCM에서 구조로 메시지를 되돌려 놓기 시작합니다.

   가져오기 애플리케이션은 프리페치 알고리즘에 따라 예상되는 순서로 메시지를 처리하기 때문에, 가져오기 애플리케이션이 차단되지 않고 SCM에서 구조로 메시지 데이터를 동기적으로 가져올 때까지 대기합니다.
7. 마침내 가져오기 애플리케이션은 공유 큐의 메시지를 모두 처리하고 다음 메시지가 사용 가능하게 될 때까지 대기합니다. 구조 및 SCM에는 메시지가 없습니다.

z/OS 응급 스토리지 - 기본 구성

IBM MQ에서 응급 스토리지에 대한 기본 시나리오를 설정하는 방법입니다.

이 태스크 정보

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

SMDS 및 메시지 오프-로딩을 SCM과 함께 사용하면 오랜 가동 중단 동안 MQRC_STORAGE_MEDIUM_FULL 이유 코드가 IBM MQ 애플리케이션으로 리턴될 가능성을 줄일 수 있습니다.

예를 들어, 엔터프라이즈에는 큐에 메시지를 넣는 애플리케이션과 큐에서 메시지를 가져오는 애플리케이션이 있습니다. 정상 실행 중에 큐 용량이 거의 0일 것으로 예상되지만, 비즈니스 요구사항에 따라 시스템은 두 시간 동안 메시지를 가져오는 애플리케이션의 가동 중단을 허용할 수 있습니다.

이는 사용 중인 공유 큐가 두 시간 동안 넣기 애플리케이션에서 생성되는 메시지를 포함할 수 있어야 함을 의미합니다. 현재 기본 오프로드 규칙 및 SMDS를 사용하여 이러한 목적을 달성합니다.

공유 큐에 송신되는 메시지의 비율은 단기에서 중기까지 두 배가 될 것으로 예상됩니다. 시스템은 여전히 두 시간의 가동 중단을 허용할 수 있어야 하지만, CF에서 구조의 크기를 두 배로 만들기 위해 사용 가능한 실제 스토리지가 충분하지 않습니다. 애플리케이션 구조를 포함하는 CF가 zEC12 시스템에 상주하기 때문에, 충분한 SCM을 구조와 연관시켜 두 시간의 가동 중단이 허용될 수 있도록 충분한 메시지를 저장할 수 있습니다.

이 초기 시나리오는 다음을 사용합니다.

- 단일 큐 관리자인 CSQ3이 포함되어 있는 큐 공유 그룹, IBM1. 큐 공유 그룹에는 관리 구조 외에도 단일 애플리케이션 구조인 SCEN1이 정의되어 있습니다.

- 커플링 기능(CF) CF01. 이 기능에는 SCEN1 애플리케이션 구조가 IBM1SCEN1 구조로 저장되어 있습니다. 이 구조의 최대 크기는 2GB입니다.
- 애플리케이션 구조가 사용하는 한 개의 공유 큐, SCEN1.Q

이 구성은 175 페이지의 그림 61에 설명되어 있습니다.

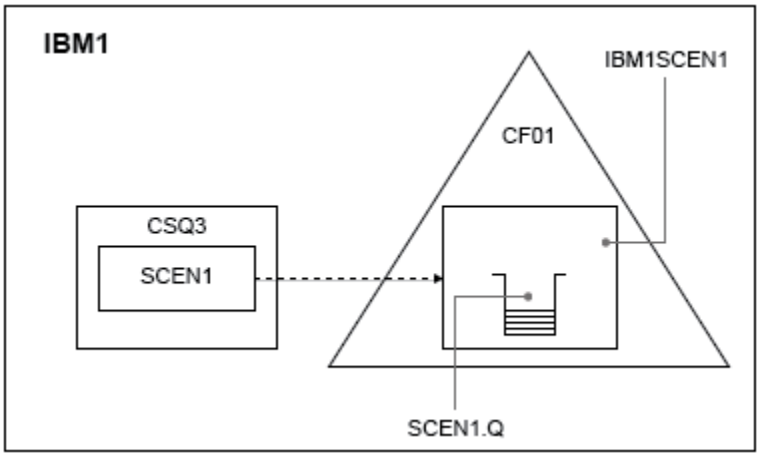


그림 *61.* 기본 구성

그리고 큐 관리자 CSQ3이 이미 큐 공유 그룹 IBM1의 유일한 멤버라고 가정하십시오.

구조 IBM1SCEN1의 정의를 커플링 기능 자원 관리자(CFRM) 정책에 추가해야 합니다. 편의상 PREFLIST(CF01)를 지정하여 하나의 커플링 기능(CF01)에서만 작성할 수 있도록 구조를 정의합니다.

주의: 프로덕션 시스템에서 고가용성을 허용하려면 IBM MQ에서 사용하는 모든 구조에 대해 PREFLIST에 두 개 이상의 CF를 포함해야 합니다.

프로시저

1. 다음 명령을 사용하여 CFRM 정책을 새로 고치십시오.

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN1
   ```

   구조 IBM1SCEN1에 대한 샘플 CFRM 정책

   ```
   STRUCTURE
   NAME(IBM1SCEN1)
   SIZE(1024M)
   INITSIZE(512M)
   ALLOWAUTOALT(YES)
   FULLTHRESHOLD(85)
   PREFLIST(CF01)
   ALLOWREALLOCATE(YES)
   DUPLEX(DISABLED)
   ENFORCEORDER(NO)
   ```

2. 다음 명령을 사용하여 구조가 올바르게 작성되었는지 확인하십시오.

   ```
   D XCF,STR,STRNAME=IBM1SCEN1
   ```

   이 시점에는 STATUS 행에 표시된 구조가 큐 공유 그룹에 할당되지 않았습니다.

3. CFRM 정책에 정의된 구조를 이용하도록 IBM MQ를 구성하십시오.

   a. 구조 이름이 SCEN1 인 DEFINE CFSTRUCT 명령을 사용하여 IBM MQ CFSTRUCT 오브젝트를 작성하십시오.

   ```
   DEFINE CFSTRUCT(SCEN1)
   CFCONLOS(TOLERATE)
   CFLEVEL(5)
   DESCR('Structure for SCM scenario 1')
   RECOVER(NO)
   ```

```
RECAUTO(YES)
OFFLOAD(DB2)
OFFLD1SZ(64K) OFFLD1TH(70)
OFFLD2SZ(64K) OFFLD2TH(80)
OFFLD3SZ(64K) OFFLD3TH(90)
```

.

b. DISPLAY CFSTRUCT 명령을 사용하여 구조의 유효성을 검증하십시오.

c. SCEN1 구조를 사용하려면 다음 MQSC 명령을 사용하여 SCEN1.Q 공유 큐를 정의하십시오.

```
DEFINE QLOCAL(SCEN1.Q) QSGDISP(SHARED) CFSTRUCT(SCEN1) MAXDEPTH(999999999)
```

4. IBM MQ Explorer를 사용하여 단일 메시지를 SCEN1.Q 큐에 넣었다가 다시 빼십시오.
5. 다음 명령을 실행하여 구조가 지금 할당되었는지 확인하십시오.

```
D XCF,STR,STRNAME=IBM1SCEN1
```

명령의 출력에서 STATUS 행에 ALLOCATED가 나타나는지 확인하십시오.

결과

기본 구성을 작성했습니다. 이제는 선택한 메소드를 사용하여 구성의 기준선 성능에 대한 아이디어를 얻을 수 있습니다.

다음에 수행할 작업

초기 구조에 SMDS 및 SCM 추가

관련 개념

168 페이지의 『공유 큐에서 스토리지 클래스 메모리의 사용』

스토리지 클래스 메모리(SCM)의 사용은 IBM MQ for z/OS 공유 큐와 함께 사용하면 유리할 수 있습니다.

z/OS 초기 구조에 *SMDS* 및 *SCM* 추가

IBM MQ에서 응급 스토리지용 SMDS 및 SCM을 추가하는 방법입니다.

이 태스크 정보

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

태스크의 이 부분은 174 페이지의 『응급 스토리지 - 기본 구성』에 설명된 기본 구성을 사용합니다. 이 시나리오는 초기 구조에 공유 메시지 데이터 세트(SMDS)를 추가한 후 SCM을 추가하는 방법에 대해 설명합니다.

이 최종 구성은 177 페이지의 그림 62에 설명되어 있습니다.

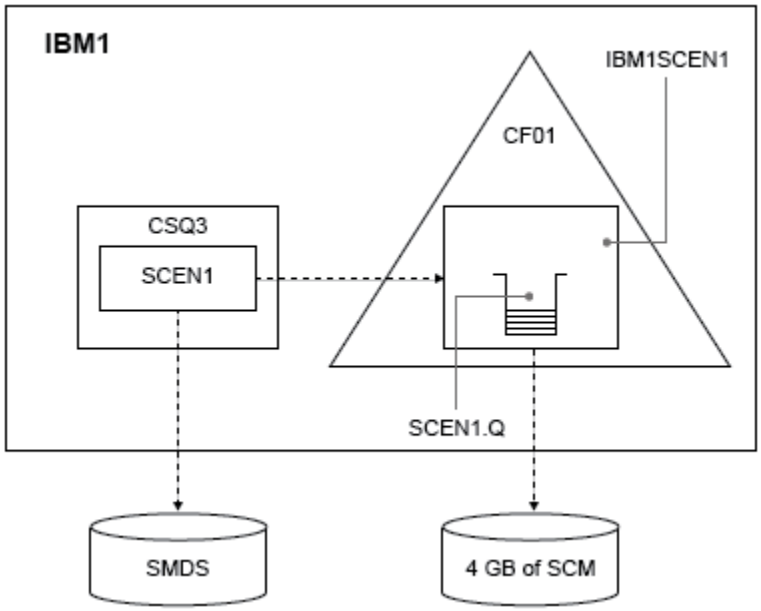


그림 *62.* 응급 스토리지용 *SMDS* 및 *SCM*을 추가하는 구성

프로시저

1. **CSQ4SMDS** 샘플 JCL을 다음과 같이 편집하여 SCEN1 애플리케이션 구조가 사용하는 SMDS 데이터 세트를 작성하십시오.

```
//CSQ4SMDS JOB NOTIFY=&SYSUID
//*
//* Allocate SMDS
//*
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER           -
(NAME(CSQSMDS.SCEN1.CSQ3.SMDS) -
MEGABYTES(5000 3000)     -
LINEAR                   -
SHAREOPTIONS(2 3) )      -
DATA                  -
(NAME(CSQSMDS.SCEN1.CSQ3.SMDS.DATA) )
/*
//*
//* Format the SMDS
//*
//FORM   EXEC PGM=CSQJUFMT,COND=(0,NE),REGION=0M
//STEPLIB DD DSN=MQ800.SCSQANLE,DISP=SHR
//     DD DSN=MQ800.SCSQAUTH,DISP=SHR
//SYSUT1  DD DISP=OLD,DSN=CSQSMDS.SCEN1.CSQ3.SMDS
//SYSPRINT DD SYSOUT=*
```

2. ALTER CFSTRUCT 명령을 실행하여, 오프로딩 및 기본 오프로드 규칙의 구현에 SMDS를 사용하도록 SCEN1 애플리케이션 구조를 변경하십시오.

```
ALTER CFSTRUCT(SCEN1) OFFLOAD(SMDS) OFFLD1SZ(32K) OFFLD2SZ(4K) OFFLD3SZ(0K)
DSGROUP('CSQSMDS.SCEN1.*.SMDS') DSBLOCK(1M)
```

   다음에 유의하십시오.

   - SCEN1.Q는 SCEN1 애플리케이션 구조에서 유일한 공유 큐이므로, **DSBLOCK** 값이 가능한 가장 큰 값인 1M으로 설정되었습니다. 이 값은 시나리오에 가장 효율적인 설정이어야 합니다.
   - 넣기 애플리케이션에서 송신된 메시지는 30KB이므로, 구조가 80% 차서 두 번째 오프로드 규칙이 충족될 때까지 SMDS로 오프-로딩이 시작되지 않습니다.

3. 테스트 애플리케이션을 다시 실행하십시오.
   큐에 있는 메시지의 증가된 스토리지를 주목하십시오.

4. 다음 프로시저를 수행하여 구조 IBM1SCEN1에 4GB의 SCM을 추가하십시오.
   a) 다음 명령을 실행하여 SCM이 설치되고 CF01에 할당된 양을 검사하십시오.

```
D CF,CFNAME=CF01
```

   b) 표시된 출력의 STORAGE CONFIGURATION 섹션에 있는 STORAGE-CLASS MEMORY 그림을 확인하여 사용 가능한 스토리지를 확인하십시오.
   c) 다음과 같은 SCMMAXSIZE 및 SCMALGORITHM 키워드로 CFRM 정책을 업데이트하십시오.

```
STRUCTURE
NAME(IBM1SCEN1)
SIZE(1024M)
INITSIZE(512M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
SCMMAXSIZE(4G)
SCMALGORITHM(KEYPRIORITY1)
```

5. 다음 명령을 실행하여 CFRM 정책을 활성화하십시오.

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=polname
```

6. IBM1SCEN1 구조를 다시 빌드하십시오.
   구조가 이전 변경을 수행했을 때 할당되었으므로 이 프로시저를 수행해야 합니다.
   다음 명령을 실행하여 구조를 다시 빌드하십시오.

```
SETXCF START,REBUILD,STRNM=IBM1SCEN1
```

결과

구성에 SCM 추가가 완료되었습니다.

다음에 수행할 작업

시스템의 성능을 최적화하십시오. 자세한 정보는 178 페이지의 『스토리지 클래스 메모리 사용 최적화』의 내용을 참조하십시오.

z/OS 스토리지 클래스 메모리 사용 최적화

스토리지 클래스 메모리(SCM)의 사용을 개선하는 방법입니다.

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

다음 명령을 실행하십시오.

```
D XCF,STR,STRNAME=IBM1SCEN1
```

이전 테스트로 인해 구조에 이미 메시지 데이터가 가득 찼기 때문에 구조에서 SCM으로 일부 메시지를 사전 스테이징하는 단계가 다시 빌드하는 과정에 포함되었습니다. 이 프로세스는 이전 명령을 사용하여 시작되었습니다.

이 명령의 출력이 생성됩니다. 예를 들면 다음과 같습니다.

```
ACTIVE STRUCTURE
----------------
ALLOCATION TIME: 06/17/2014 09:28:50
CFNAME : CF01
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
PARTITION: 3B CPCID: 00
STORAGE CONFIGURATION ALLOCATED MAXIMUM %
```

```
ACTUAL SIZE: 1024 M 1024 M 100
AUGMENTED SPACE: 3 M 142 M 2
STORAGE-CLASS MEMORY: 88 M 4096 M 2
ENTRIES: 120120 1089536 11
ELEMENTS: 240240 15664556 1
SPACE USAGE IN-USE TOTAL %
ENTRIES: 84921 219439 38
ELEMENTS: 2707678 3149050 85
EMCS: 2 282044 0
LOCKS: 1024
SCMHIGHTHRESHOLD : 90
SCMLOWTHRESHOLD : 70
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CD5186A0 2BD8B85C
LOGICAL VERSION: CD515C50 CE2ED258
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME : IXCLO053
DISPOSITION : KEEP
ACCESS TIME : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS : 1
CONNECTION NAME ID VERSION  SYSNAME JOBNAME  ASID STATE
--------------- -- -------  ------- -------  ---- ----------------
CSQEIBM1CSQ301  01 00010059 SC61    CSQ3MSTR 0091 ACTIVE
```

명령의 출력에서 다음에 유의하십시오.

- STORAGE_CLASS MEMORY는 4096MB(**MAXIMUM**)의 SCM이 구조에 추가되었음을 확인합니다.
- 사전 스테이징에 사용된 STORAGE-CLASS MEMORY의 크기에 대한 ALLOCATED 수치. SCM이 추가되기 전에 아무것도 없었던 구조에 이제는 여유 공간이 있습니다.
- SCM 사용량을 추적하는 데 사용된 AUGMENTED SPACE의 크기
- 사전 스테이징 알고리즘이 구조에서 SCM으로 데이터를 이동하기 시작하는 시점은 구조가 90% 찼을 때입니다. 이는 구성 불가능한 **SCMHIGHTHRESHOLD** 특성으로 표시됩니다.
- 프리페칭 알고리즘이 SCM에서 구조로 데이터를 이동하기 시작하는 시점은 구조가 70% 찼을 때입니다. 이는 구성 불가능한 **SCMLOWTHRESHOLD** 특성으로 표시됩니다.

이제 SCM의 사용을 최적화하기 위한 다양한 방법을 테스트할 수 있습니다. 다음에 유의하십시오.

- SCM을 사용하여 메시지를 저장한 후에는 SCM에서 모든 데이터를 제거할 때까지 구조를 변경할 수 없습니다.

  이 경우 입력 항목 대 요소 비율은 SCM이 처음 사용되었을 때 준비된 값에서 고정됩니다. 사전 스테이징 알고리즘이 SCM으로 데이터 이동을 시작하기 전에 구조가 원하는 상태에 있는지 주의 깊게 확인해야 합니다.
- SCM을 사용하기 전에 현재 구조 크기가 올바릅니까?

  예를 들어, 512MB에서 1GB의 SIZE로 **INITSIZE**를 늘렸습니까?

  이 작업을 수행하지 않으면, 구조의 자동 변경이 가능하도록 설정했더라도 사전 스테이징 알고리즘은 변경이 시작되기 전에 미리 데이터를 SCM으로 이동하기 시작합니다. 그 결과, 구조는 512MB의 실제 스토리지를 사용하여 고정됩니다.
- SCM을 사용하기 전에 입력 항목 대 요소 비율이 올바릅니까?

  이 시나리오의 목표는 가능한 한 많은 메시지를 전부 구조 스토리지에 보관하고, 전체적으로 구조 및 SCM에 저장할 수 있는 오프로드된 메시지 포인터의 수를 늘리는 것입니다. 이러한 메시지에 액세스하는 것은 SMDS의 메시지에 액세스하는 것보다 더 빠릅니다.

  따라서 메시지를 저장하기에 좋은 입력 항목 대 요소 비율로 시작되고 나서 프리스테이지 알고리즘이 처음 시작되기 전에 메시지 포인터를 저장하기에 좋은 비율로 상태 전이되는 구조를 가지고 있어야 합니다. 이 상태 전이는 IBM MQ 오프로드 규칙을 이용하여 어느 정도 달성할 수 있습니다.

다음 명령을 실행하여 오프로드 규칙을 변경하십시오.

```
ALTER CFSTRUCT(SCEN1) OFFLD1SZ(0K)
```

입력 항목 대 요소 비율을 최적화하기 위해 여러 가지 실행을 수행해야 할 수도 있습니다.

다음 표에서는 응급 스토리지 시나리오의 각기 다른 단계 동안 큐에 넣은 메시지 수가 개선될 수 있음을 보여 줍니다.

표 *18.* 응급 스토리지 시나리오의 결과 비교		
테스트 설명	**메시지 수**	**큐를 채우는 시간(초)**
기본 구성	27,850	3.2
기본 오프로드 규칙을 사용한 SMDS	205,000	158
기본 오프로드 규칙을 사용한 SCM	828,610	469
조정된 오프로드 규칙을 사용한 SCM	1,135,775	679

표의 마지막 행은 오프로드 규칙을 조정하여 필요한 효과를 얻었음을 보여 줍니다.

시스템을 조사하여 어떤 방법으로든 이러한 수치를 개선할 수 있는지 확인해야 합니다. 예를 들어, 사용 가능한 SMDS 스토리지가 부족할 수 있습니다. 추가 SMDS 스토리지를 할당할 수 있는 경우 큐의 메시지 수를 매우 크게 늘릴 수 있어야 합니다.

z/OS 향상된 성능 - 기본 구성
IBM MQ에서 공유 큐를 사용하여 향상된 성능에 대한 기본 시나리오를 설정하는 방법입니다.

이 태스크 정보

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

이 시나리오에서는 SMDS 사용에 따른 성능 비용을 발생시키지 않고 공유 큐에 저장할 수 있는 메시지의 수를 늘리도록 SCM을 사용하는 방법에 대해 설명합니다.

이 초기 시나리오는 응급 스토리지에 사용된 시나리오와 유사하며 다음을 사용합니다.

- 단일 큐 관리자인 CSQ3이 포함되어 있는 큐 공유 그룹, IBM1. 큐 공유 그룹에는 관리 구조 외에도 단일 애플리케이션 구조인 SCEN2가 정의되어 있습니다.
- 커플링 기능(CF) CF01. 이 기능에는 SCEN2 애플리케이션 구조가 IBM1SCEN2 구조로 저장되어 있습니다. 이 구조의 최대 크기는 2GB입니다.
- 단일 공유 큐, SCEN2.Q. 이는 애플리케이션 구조를 사용하도록 구성되어 있습니다.

이 구성은 180 페이지의 그림 63에 설명되어 있습니다.

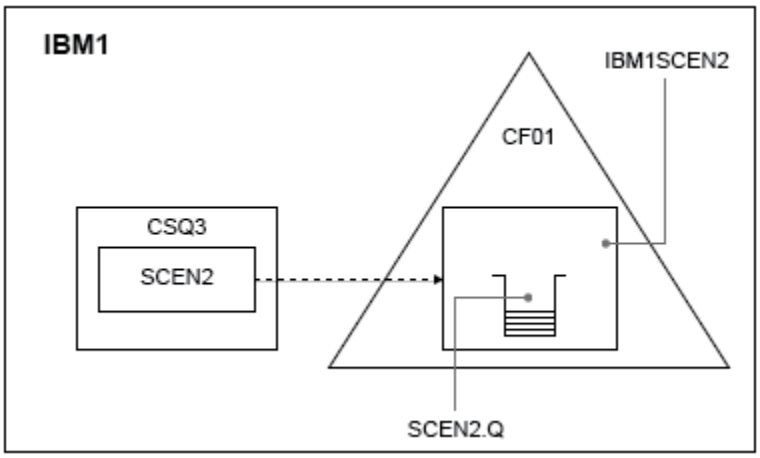


그림 *63.* 기본 구성

그리고 큐 관리자 CSQ3이 이미 큐 공유 그룹 IBM1의 유일한 멤버라고 가정하십시오.

구조 IBM1SCEN2의 정의를 커플링 기능 자원 관리자(CFRM) 정책에 추가해야 합니다. 편의상 PREFLIST(CF01)를 지정하여 하나의 커플링 기능(CF01)에서만 작성할 수 있도록 구조를 정의합니다.

구조 IBM1SCEN2에 대한 샘플 CFRM 정책

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

구조를 크기 조정할 수 없도록 하기 위해 **INITSIZE** 및 **SIZE** 키워드 모두 값 2048M을 가지고 있습니다.

프로시저

1. 다음 명령을 사용하여 CFRM 정책을 새로 고치십시오.

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
   ```

2. 다음 명령을 사용하여 구조가 올바르게 작성되었는지 확인하십시오.

   ```
   D XCF,STR,STRNAME=IBM1SCEN2
   ```

   앞의 명령을 실행하면 다음과 같은 출력이 표시됩니다.

   ```
   RESPONSE=SC61
   IXC360I 07.58.51 DISPLAY XCF 581
   STRNAME: IBM1SCEN2
   STATUS: NOT ALLOCATED
   POLICY INFORMATION:
   POLICY SIZE : 2048 M
   POLICY INITSIZE: 2048 M
   POLICY MINSIZE : 1536 M
   FULLTHRESHOLD : 85
   ALLOWAUTOALT : YES
   REBUILD PERCENT: N/A
   DUPLEX : DISABLED
   ALLOWREALLOCATE: YES
   PREFERENCE LIST: CF01
   ENFORCEORDER : NO
   EXCLUSION LIST IS EMPTY

   EVENT MANAGEMENT: MESSAGE-BASED   MANAGER SYSTEM NAME: SC53
   MANAGEMENT LEVEL : 01050107
   ```

   이 시점에는 STATUS 행에 표시된 구조가 큐 공유 그룹에 할당되지 않았습니다.

3. CFRM 정책에 정의된 구조를 이용하도록 IBM MQ를 구성하십시오.

   a. DEFINE CFSTRUCT 명령을 SCEN2 의 구조 이름과 함께 사용하여 IBM MQ CFSTRUCT 오브젝트를 작성하십시오.

      ```
      DEFINE CFSTRUCT(SCEN2)
      CFCONLOS(TOLERATE)
      CFLEVEL(5)
      DESCR('Structure for SCM scenario 2')
      RECOVER(NO)
      RECAUTO(YES)
      OFFLOAD(DB2)
      OFFLD1SZ(64K) OFFLD1TH(70)
      OFFLD2SZ(64K) OFFLD2TH(80)
      OFFLD3SZ(64K) OFFLD3TH(90)
      ```

   b. DISPLAY CFSTRUCT 명령을 사용하여 구조를 검사하십시오.

c. SCEN2 구조를 사용하려면 다음 MQSC 명령을 사용하여 SCEN2.Q 공유 큐를 정의하십시오.

```
DEFINE QLOCAL(SCEN2.Q) QSGDISP(SHARED) CFSTRUCT(SCEN2) MAXDEPTH(999999999)
```

4. IBM MQ 탐색기를 사용하여 단일 메시지를 SCEN2.Q 큐에 넣었다가 다시 빼십시오.
5. 다음 명령을 실행하여 구조가 지금 할당되었는지 확인하십시오.

```
D XCF,STR,STRNAME=IBM1SCEN2
```

명령의 출력을 검토하고, 이 출력의 일부가 표시되면 STATUS 행에 ALLOCATED가 나타나는지 확인하십시오.

```
RESPONSE=SC61
IXC360I 08.31.27 DISPLAY XCF 703
STRNAME: IBM1SCEN2
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
POLICY SIZE : 2048 M
POLICY INITSIZE: 2048 M
POLICY MINSIZE : 1536 M
FULLTHRESHOLD : 85
ALLOWAUTOALT : YES
REBUILD PERCENT: N/A
DUPLEX : DISABLED
ALLOWREALLOCATE: YES
PREFERENCE LIST: CF01
ENFORCEORDER : NO
EXCLUSION LIST IS EMPTY
```

또한 SPACE USAGE 섹션의 필드 값을 주목하십시오.

- ENTRIES
- ELEMENTS
- EMCS
- LOCKS

값의 예는 다음과 같습니다.

```
SPACE USAGE   IN-USE      TOTAL    %
ENTRIES:      344686     345242   99
ELEMENTS:    6548455    6548467   99
EMCS:              2     780318    0
LOCKS:          1024
```

결과

기본 구성을 작성했습니다. 이제는 선택한 메소드를 사용하여 구성의 기준선 성능에 대한 아이디어를 얻을 수 있습니다.

다음에 수행할 작업

기본 시나리오를 테스트해야 합니다. 예를 들어, 다음 세 가지 애플리케이션을 사용하여 표시되는 순서대로 애플리케이션을 시작하고 동시에 이를 실행할 수 있습니다.

1. PCF 애플리케이션을 사용하여 5초마다 SCEN2.Q의 현재 용량(**CURDEPTH**) 값을 요청하십시오. 출력을 사용하면 시간 경과에 따른 큐의 용량을 구성할 수 있습니다.
2. 단일 스레드 가져오기 애플리케이션은 무한 대기 후 가져오기를 사용하여 SCEN2.Q에서 메시지를 반복적으로 가져옵니다. 제거된 메시지의 처리를 시뮬레이션하기 위해 가져오기 애플리케이션은 10개의 메시지가 제거될 때마다 4밀리초 동안 일시정지합니다.
3. 단일 스레드 넣기 애플리케이션은 총 1백만 개의 4KB 비지속 메시지를 SCEN2.Q에 넣습니다. 이 애플리케이션은 각 메시지를 넣는 사이에 일시정지하지 않기 때문에 가져오기 애플리케이션이 메시지를 가져올 때보다 SCEN2.Q에 메시지를 넣을 때의 속도가 더 빠릅니다.

   결과적으로 넣기 애플리케이션이 실행 중일 때 SCEN2.Q의 용량이 증가합니다.

구조 IBM1SCEN2가 채워지고 넣기 애플리케이션이 MQRC_STORAGE_MEDIUM_FULL 이유 코드를 수신하면, 넣기 애플리케이션은 다음 메시지를 큐에 넣으려고 시도하기 전에 5초 동안 휴면합니다.

일정 기간 동안 CURDEPTH 애플리케이션의 결과를 구성할 수 있습니다. 넣기 애플리케이션이 큐를 부분적으로 비울 수 있도록 일시정지하는 동안은 톱니 파형 출력을 얻습니다.

183 페이지의 『초기 구조에 SCM 추가』(으)로 이동하십시오.

관련 개념

168 페이지의 『공유 큐에서 스토리지 클래스 메모리의 사용』

스토리지 클래스 메모리(SCM)의 사용은 IBM MQ for z/OS 공유 큐와 함께 사용하면 유리할 수 있습니다.

z/OS 초기 구조에 *SCM* 추가

IBM MQ에 대한 성능 향상을 위해 SCM을 추가하는 방법입니다.

이 태스크 정보

중요사항: IBM z16 은 결합 기능 이미지에 대한 가상 플래시 메모리 (스토리지 클래스 메모리 또는 SCM이라고도 함) 의 사용을 지원하기 위해 IBM Z ® 의 마지막 세대가 될 예정입니다. 자세한 정보는 IBM Z 및 IBM LinuxONE 4Q 2023 Statement of Direction을 참조하십시오.

또는 더 큰 구조를 사용하거나 메시지를 SMDS로 오프로드해야 합니다.

태스크의 이 부분은 180 페이지의 『향상된 성능 - 기본 구성』에 설명된 기본 구성을 사용합니다. 이 시나리오는 초기 구조에 대한 SCM 추가에 대해 설명합니다.

이 최종 구성은 183 페이지의 그림 64에 설명되어 있습니다.

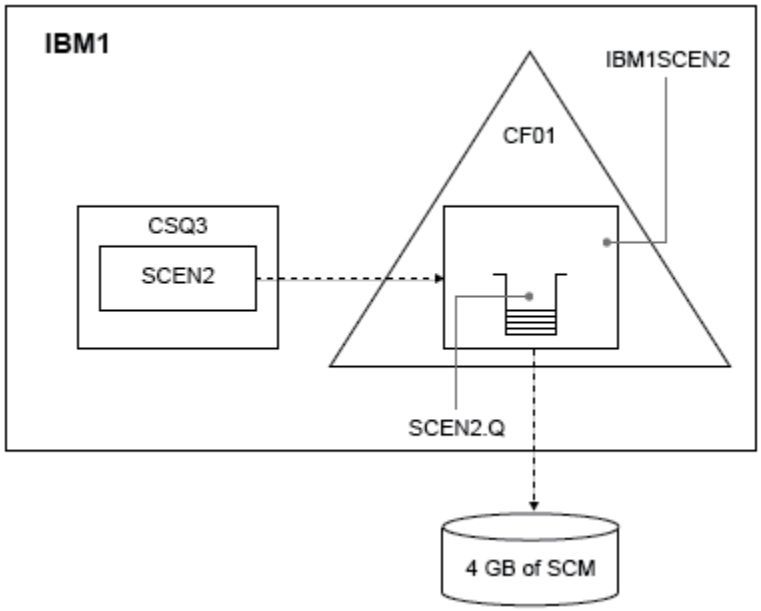


그림 *64.* 성능 향상을 위해 *SCM*을 추가하는 구성

프로시저

1. 다음 프로시저를 수행하여 구조 IBM1SCEN2에 4GB의 SCM을 추가하십시오.
   a) 다음 명령을 실행하여 SCM이 설치되고 CF01에 할당된 양을 검사하십시오.

```
D CF,CFNAME=CF01
```

   b) 표시된 출력의 STORAGE CONFIGURATION 섹션에 있는 STORAGE-CLASS MEMORY 그림을 확인하여 사용 가능한 스토리지를 확인하십시오.
   c) 다음과 같은 SCMMAXSIZE 및 SCMALGORITHM 키워드로 CFRM 정책을 업데이트하십시오.

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
SCMMAXSIZE(4G)
SCMALGORITHM(KEYPRIORITY1)
```

2. 다음 명령을 실행하여 CFRM 정책을 활성화하십시오.

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
```

3. IBM1SCEN2 구조를 다시 빌드하십시오.

   구조가 이전 변경을 수행했을 때 할당되었으므로 이 프로시저를 수행해야 합니다.

   다음 명령을 실행하여 구조를 다시 빌드하십시오.

```
SETXCF START,REBUILD,STRNM=IBM1SCEN2
```

4. 다음 명령을 실행하여 구조의 새 구성을 확인하십시오.

```
D XCF,STR,STRNAME=IBM1SCEN2
```

   명령의 출력을 검토하십시오. 이 출력의 일부는 다음과 같습니다.

```
SPACE USAGE     IN-USE       TOTAL      %
ENTRIES:          33         342684     0
ELEMENTS:         48        6503697     0
EMCS:              2         575600     0
LOCKS:                         1024
```

결과

SCM을 사용하는 데 필요한 제어 스토리지의 증가에 따른 실제 스토리지 사용의 변경을 계산하십시오.

- SCM이 구조에 추가되기 전, 구조에는 180 페이지의 『향상된 성능 - 기본 구성』에 표시된 대로 다음과 같은 합계가 있습니다.
  - 345,242개의 입력 항목
  - 6,548,467개의 요소
  - 780,318개의 EMCS
- SCM이 구조에 추가된 후, 구조에는 다음과 같은 합계가 있습니다.
  - 342,684개의 입력 항목
  - 6,503,697개의 요소
  - 575,600개의 EMCS

이러한 수치를 사용하여 SCM을 추가한 후에는 구조의 크기가 다음 항목의 수량만큼 감소됩니다.

- 2558개의 입력 항목
- 44,770개의 요소
- 204,718개의 EMCS

SCM을 관리하는 데 사용되는 구조 스토리지의 크기는 4GB의 SCM이 할당된 2GB 구조의 경우 다음과 같습니다.

```
(2558 + 44,770 + 204,718) * 256 = 61.5 MB
```

SCM 추적에 사용된 제어 스토리지의 양과 할당된 SCM의 양이 모두 스토리지 크기로서 증가하기 때문에, SCM을 더 추가하면 구조의 크기 감소는 단지 미미한 정도일 수 있다는 점에 유의하십시오.

다음에 수행할 작업

180 페이지의 『향상된 성능 - 기본 구성』의 최종 섹션에 설명된 테스트를 반복하십시오.

일정 기간 동안 수정된 애플리케이션의 결과를 구성할 수 있습니다. 이제는 큐가 부분적으로 비워질 때까지 넣기 애플리케이션이 더 이상 대기할 필요가 없기 때문에, 구성을 이전에 확보한 구성과 비교하여 톱니 파형이 없는 출력을 확보합니다.

자세한 정보는 MP16: WebSphere MQ for z/OS -용량 계획 및 조정을 참조하십시오.

z/OS 분산 큐잉 및 큐 공유 그룹

분산 큐잉 및 큐 공유 그룹은 애플리케이션 시스템의 가용성을 높이는 데 사용할 수 있는 두 가지 기술입니다. 이러한 기술에 대한 추가 정보를 찾는 데 이 토픽을 사용하십시오.

공유 큐에서 메시지의 고가용성을 보완하도록 IBM MQ의 분산 큐잉 컴포넌트는 다음을 제공하는 추가 기능을 포함합니다.

- 네트워크에 대한 고가용성.
- 큐 공유 그룹에 대한 인바운드 네트워크 연결의 증가된 용량.

185 페이지의 그림 65에서는 분산 큐잉 및 큐 공유 그룹을 설명합니다. 여기에서는 모두 동일한 큐 공유 그룹에 속하는 SYSPLEX 내 두 개의 큐 관리자를 표시합니다. 모두 공유 큐 SQ1에 액세스할 수 있습니다. 네트워크의 큐 관리자(예: AIX 및 Windows)는 두 큐 관리자의 채널 시작기를 통해 이 큐에 메시지를 넣을 수 있습니다. 두 큐 관리자의 복제된 애플리케이션은 큐 역할을 합니다.

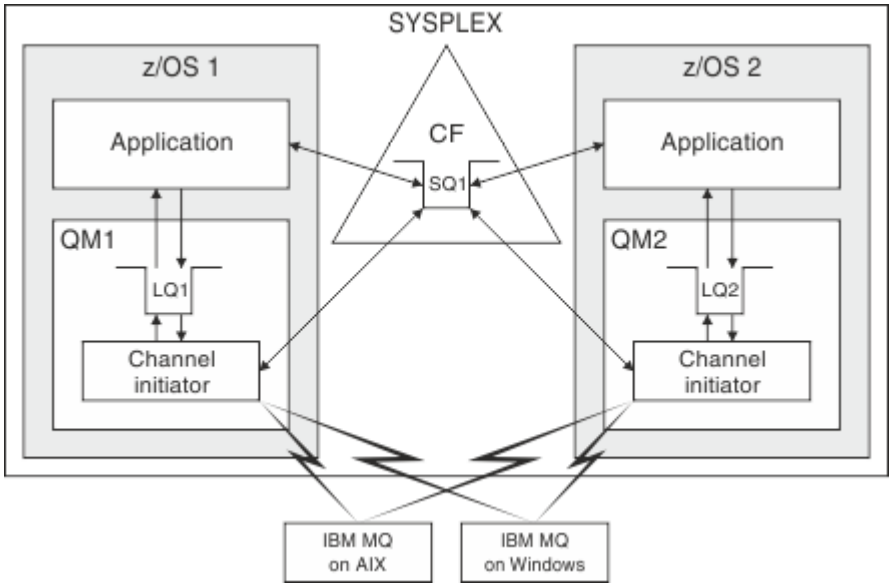


그림 *65.* 분산 큐잉 및 큐 공유 그룹

관련 개념

185 페이지의 『공유 채널』
이 주제를 사용하여 공유 채널의 개념과 IBM MQ for z/OS에서의 사용을 이해하십시오.

190 페이지의 『그룹 내 큐잉』
이 절에서는 z/OS 플랫폼에 고유한 IBM MQ for z/OS 함수인 그룹 내 큐잉에 대해 설명합니다. 이 기능은 큐 공유 그룹에 정의된 큐 관리자만이 사용할 수 있습니다.

187 페이지의 『클러스터 및 큐 공유 그룹』
클러스터에서 큐 공유 그룹을 사용하는 방법을 이해하는 데 이 토픽을 사용하십시오.

z/OS 공유 채널

이 주제를 사용하여 공유 채널의 개념과 IBM MQ for z/OS에서의 사용을 이해하십시오.

다양한 네트워킹 제품에서 네트워크로부터 서버 장애를 숨기거나 적합한 서버 세트에서 인바운드 네트워크 요청의 밸런스를 조정하는 메커니즘을 제공합니다. 네트워크는 인바운드 네트워크 연결 요청에 대해 일반 포트를 사용 가능하게 하고 적합한 서버 중 하나에 연결하여 인바운드 요청을 만족할 수 있습니다.

이러한 네트워킹 제품으로는 다음이 포함됩니다.

- VTAM 일반 자원
- SYSPLEX 분산기

채널 시작기는 공유 큐 기능을 사용하기 위해 이러한 제품을 활용합니다.

공유 채널에는 공유 인바운드 채널 및 공유 아웃바운드 채널과 같은 두 종류가 있습니다.

- 공유 인바운드 채널
- 공유 아웃바운드 채널

채널에 대한 자세한 정보는 다음을 참조하십시오.

- 공유 채널 요약
- 공유 채널 상태

공유 인바운드 채널

큐 공유 그룹의 각 채널 시작기는 추가 리스너 태스크를 시작하여 일반 포트에서 대기합니다. 이 일반 포트는 지원 기술(VTAM, TCP/IP) 중 하나에 의해 네트워크에서 사용할 수 있습니다. 일반 포트에 대한 인바운드 네트워크 첨부 요청은 네트워크 기술에 의해 일반 포트에서 대기하는 큐 공유 그룹(QSG)에 있는 리스너 중 하나로 디스패치됩니다.

채널 시작기에 해당 이름의 채널에 대한 채널 정의에 대한 액세스 권한이 있는 경우 인바운드 첨부가 전달되는 채널 시작기에서 채널을 시작할 수 있습니다. 채널 정의는 큐 관리자에 대해 개인용으로 정의할 수도 있고, 공유 저장소에 저장되어 모든 위치에서 사용 가능하도록 정의할 수도 있습니다(글로벌 정의). 즉, 이를 글로벌 정의로 정의하여 큐 공유 그룹의 채널 시작기에서 채널 정의를 사용 가능하게 할 수 있음을 의미합니다.

일반 포트를 통해 채널을 시작하는 경우 또 다른 차이점이 있습니다. 채널 동기화는 개별 큐 관리자가 아닌 큐 공유 그룹과 이루어집니다. 예를 들어 리모트 큐 관리자에서 일반 포트를 통해 채널을 시작하는 방법을 고려하십시오. 먼저 채널이 시작되면 큐 관리자 QM1 및 메시지 플로우에서 시작될 수 있습니다. 채널이 중지되고 큐 관리자 QM2에서 재시작되면 큐 공유 그룹과 동기화되므로 플로우되는 메시지 수에 대한 정보는 여전히 올바릅니다.

일반 포트를 통해 시작된 인바운드 채널을 사용하여 큐에 메시지를 넣을 수 있습니다. 리모트 큐 관리자는 대상 큐가 공유인지 여부를 알지 못합니다. 대상 큐가 공유 큐인 경우 리모트 큐 관리자는 로드 밸런스 조정된 방식으로 사용 가능한 채널 시작기를 통해 연결되며 메시지는 공유 큐에 넣어집니다.

대상 큐가 개인 큐인 경우 메시지는 채널의 현재 인스턴스가 연결된 큐 관리자가 소유한 개인 큐에 넣습니다. 복제된 로컬 큐라고 하는 이 환경에서 각 큐 관리자는 개인 큐의 동일한 세트를 정의해야 합니다.

큐 공유 그룹에 대한 SVRCONN 채널 구성

큐 공유 그룹에서 SVRCONN 채널에 대한 최적의 구성은 포인트-투-포인트 채널에서 서로 다른 포트 번호를 사용하는 개인용 리스너를 각 CHINIT에서 설정하는 것입니다. 그러면 이러한 리스너 포트는 가상 IP 주소(VIPA)를 사용하는 Sysplex 분산기와 같은 새 워크로드 분산 메커니즘에 대한 '백엔드' 자원으로 사용됩니다. 그런 다음 외부 VIPA 주소를 네트워크에서 CLNTCONN 정의에 대한 대상 주소로 사용합니다. SVRCONN 채널은 QSG의 모든 큐 관리자에 대해 동일한 정의를 사용할 수 있도록 QSGDISP(GROUP)로 정의될 수 있습니다. 이 구성은 공유 리스너를 사용하지 않으므로, 클라이언트/서버 채널에 필요하지 않은 공유 채널 상태를 유지하는 큐 공유 그룹의 성능 영향을 줄입니다.

공유 아웃바운드 채널

아웃바운드 채널은 공유 전송 큐에서 메시지를 가져오는 경우 공유 채널로 간주됩니다. 공유되는 경우 큐 공유 그룹 레벨에서 동기화 정보를 보유합니다. 이는 통신 서브시스템, 채널 시작기 또는 큐 관리자가 실패하면 큐 공유

그룹 내 서로 다른 큐 관리자 및 채널 시작기 인스턴스에서 채널을 재시작할 수 있음을 의미합니다. 이러한 방식으로 실패한 채널을 재시작하는 작업은 피어 채널 복구라고 하는 공유 채널의 기능입니다.

공유 아웃바운드 채널에 대한 워크로드 밸런싱

특정 채널 시작기에서 시작하도록 지정하지 않는 경우 아웃바운드 공유 채널이 큐 공유 그룹 내 채널 시작기에서 시작하는 데 적합합니다. IBM MQ에서 선택한 채널 시작기는 다음 기준을 사용하여 판별됩니다.

- 현재 필요한 통신 서브시스템이 채널 시작기에 대해 사용 가능합니까?
- Db2 연결이 채널 시작기에 대해 사용 가능합니까?
- 어떤 채널 시작기가 현재 워크로드가 가장 낮습니까? 워크로드에는 활성 및 재시도 중인 채널이 포함됩니다.

공유 채널 요약

공유 채널은 다음과 같은 점에서 개인용 채널과 다릅니다.

개인용 채널
단일 채널 시작기에 연결됩니다.

- 아웃바운드 채널은 로컬 전송 큐를 사용합니다.
- 인바운드 채널은 로컬 포트를 통해 시작됩니다.
- 동기화 정보는 SYSTEM.CHANNEL.SYNCQ 큐에 보유됩니다.

공유 채널
고가용성으로 밸런스 조정된 워크로드

- 아웃바운드 채널은 공유 전송 큐를 사용합니다.
- 인바운드 채널은 일반 포트를 통해 시작됩니다.
- 동기화 정보는 SYSTEM.QSG.CHANNEL.SYNCQ 큐에 보유됩니다.

CHLDISP 옵션을 START CHANNEL 명령과 함께 사용하여 채널을 시작할 때 채널이 개인용인지 또는 공유인지 여부를 지정합니다. 공유 채널은 개인용 채널과 같은 방법으로 트리거하여 시작될 수 있습니다. 그러나 공유 채널이 시작되면 IBM MQ는 워크로드 밸런싱을 수행하고 큐 공유 그룹 내 가장 적합한 채널 시작기에서 채널을 시작합니다. (필요한 경우 공유 채널을 특정 채널 시작기에서 시작하도록 지정할 수 있습니다.)

공유 채널 상태

큐 공유 그룹의 채널 시작기는 Db2에서 공유 채널 상태 테이블을 유지보수합니다. 이는 특정 채널 시작기에서 활성 상태인 채널을 기록합니다. 채널 시작기 또는 통신 시스템 실패가 발생한 경우 공유 채널 상태 테이블이 사용됩니다. 이는 큐 공유 그룹의 다른 채널 시작기에서 채널을 재시작해야 함을 표시합니다.

z/OS 클러스터 및 큐 공유 그룹

클러스터에서 큐 공유 그룹을 사용하는 방법을 이해하는 데 이 토픽을 사용하십시오.

단일 정의에서 공유 큐를 클러스터에 대해 사용 가능하도록 설정할 수 있습니다. 이를 수행하려면 공유 큐를 정의할 때 클러스터 이름을 지정합니다.

네트워크의 사용자는 큐 공유 그룹 내 각 큐 관리자가 호스팅할 공유 큐를 확인합니다(공유 큐는 큐 공유 그룹에서 호스팅하는 것으로 표시되지 않음). 클라이언트는 큐 공유 그룹의 멤버를 포함한 세션을 시작하여 동일한 공유 큐에 메시지를 넣을 수 있습니다.

188 페이지의 그림 66에서는 큐 공유 그룹의 멤버를 통해 클러스터의 멤버가 공유 큐에 액세스할 수 있는 방법을 보여줍니다.

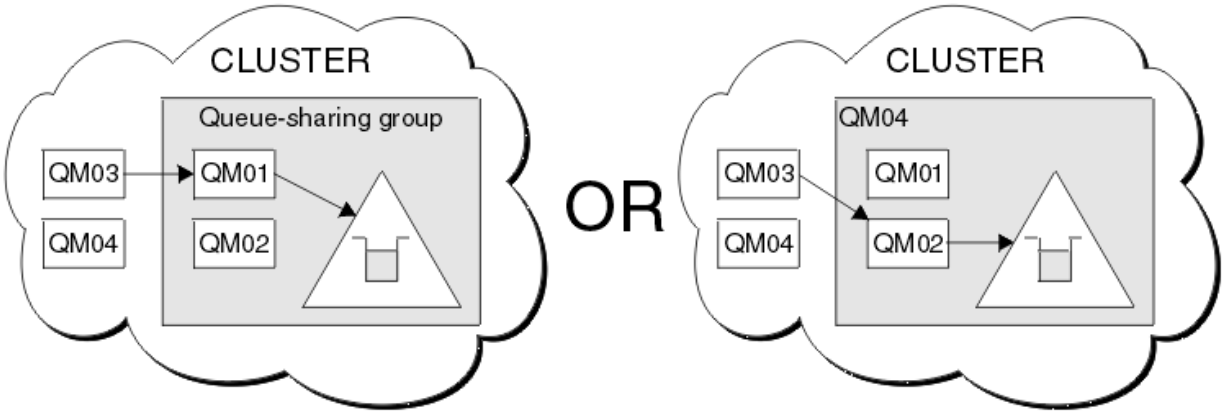


그림 66. 클러스터의 일부인 큐 공유 그룹

z/OS 공유 큐에 영향을 주는 워크로드 분배

큐 공유 그룹의 공유 큐에서 워크로드 분배에 영향을 주는 요인을 이해하는 데 이 토픽을 사용하십시오.

IBM MQ는 공유 큐에 대한 워크로드 밸런싱을 지원하지 않습니다. 그러나 큐 공유 그룹(QSG)에서 워크로드 분배는 풀 기반 방식에 영향을 줄 수 있습니다. 큐 관리자가 큐를 지원하는 방법에 대한 선택사항(공유 큐에 기록된 메시지를 수신함)은 큐 공유 그룹에 있는 각 큐 관리자의 사용 가능한 처리 용량과 sysplex에 정의된 워크로드 관리 목표에 의해 영향을 받습니다.

그러나 메시지의 MQPUT를 수행하는 큐 관리자는 메시지를 가져오는 큐 관리자를 결정하는 데 큰 영향을 줄 수 있다는 점을 명심해야 합니다.

로컬 큐 관리자가 MQGET을 수행할 수 있음

MQPUT를 수행하는 애플리케이션의 경우 로컬 큐 관리자는 애플리케이션이 연결되는 큐 관리지가 되도록 지시됩니다.

애플리케이션을 가져오는 대신, MQGET을 수행하여 메시지의 MQPUT를 지원하는 큐 관리자만 다음 고려사항의 영향을 받습니다.

메시지를 빈 공유 큐에 넣으면 큐 공유 그룹의 다른 큐 관리자에게 알리기 전에 일반적으로 로컬 큐 관리자에게 게시됩니다. 로컬 큐 관리자가 메시지를 처리하는 위치에 있으면 QSG의 다른 큐 관리자보다 먼저 커플링 기능(CF)에서 목록 전이 알림을 수신합니다. (목록 전이 알림은 빈 상태에서 비어 있지 않은 상태로 공유 큐 상태가 변경됨을 표시하는 알림입니다.)

이 경우 가능한 시나리오는 다음과 같습니다.

1. 동기점을 벗어난 비지속 메시지의 MQPUT 및 대기 중인 *Getter*에 빠른 넣기.

   큐에 대한 로컬 큐 관리자에 대기 상태의 *MQGET*을 포함하는 애플리케이션이 있는 경우 메시지의 MQPUT는 큐에 기록되지 않고 가져오는 애플리케이션의 버퍼에 직접 전달됩니다. 이는 공유 및 비공유 큐에 대해 참입니다. 이 기능은 종종 대기 중인 *Getter*에 빠른 넣기 메커니즘이라고 합니다. 공유 큐의 경우 큐가 빈 상태에서 비어 있지 않은 상태로 전이되지 않으면 이를 QSG의 다른 큐 관리자에 알립니다. 이는 예를 들어, 이 큐 관리자가 이 애플리케이션에서 모든 입력을 서비스하는 경우 큐에 메시지를 넣는 다른 애플리케이션이 없다고 가정하면 큐 공유 그룹의 다른 큐 관리자는 이 큐의 정리 작업을 지원하지 않음을 의미합니다. 그러나 로컬 큐 관리자에 대기 상태인 MQGET이 없고 메시지를 공유 큐에 넣으면 CF는 목록 전이 알림에 대한 규칙에 따라 큐 공유 그룹의 다른 큐 관리자에 알립니다.
2. 동기점 내 메시지 또는 지속 메시지의 MQPUT.

   이 경우 로컬 큐 관리자에 대기 상태의 *MQGET*을 포함하는 애플리케이션이 있는 경우 메시지를 공유 큐에 넣고 CF는 목록 상태 전이 알림의 규칙에 따라 큐 공유 그룹의 기타 큐 관리자에 알립니다. 그러나 로컬 큐 관리자는 CF에서 상태 전이 알림을 기다리지 않지만 먼저 로컬 대기 상태의 *MQGET*을 유지하고 일반적으로 큐 공유 그룹의 다른 큐 관리자가 CF 알림에 응답할 수 있기 전에 애플리케이션 대신 이 메시지의 가져오기를 수행합니다. 이는 로컬 큐 관리자의 사용 정도에 따라 달라집니다. 그렇지 않으면 빈 큐에 메시지의 도착으로 인해

CF에서 알리는 큐 관리자는 먼저 가져오기를 지원하려고 합니다. 응답할 첫 번째 큐 관리자가 새 메시지를 처리합니다.

3. 마지막으로 큐에서 메시지가 정리되지 않는 경우, CF가 빈 상태에서 비어 있지 않은 상태로 상태 변경을 표시하는 알림을 전송하는 경우 연결된 모든 큐 관리자는 큐의 처리를 지원할 기회를 갖게 됩니다. 이 경우 워크로드는 풀 기반이라고 합니다.

이 디자인에서는 순수하게 풀 기반 워크로드 분배에서 향상된 성능을 제공할 수 있습니다. 이 경우 목표는 CF에 보유된 큐에서 제공하는 고가용성을 활용하는 동시에, 가능한 경우 CF를 참조하지 않고도 큐 관리자가 MQGET을 수행하고 이를 통해 메시지 워크로드를 가능한 한 효율적으로 처리할 수 있습니다.

이전에 설명한 성능 개선보다 워크로드의 밸런스에 대한 강조가 보다 중요한 경우에는 대체 접근 방법을 채택할 수 있습니다. 예를 들어 넣기 애플리케이션이 연결된 동일한 큐 관리자에 연결된 가져오는 애플리케이션이 없도록 보장합니다. 이 설계를 사용하면 모든 메시지를 큐에 넣고 이러한 전이를 처리하는 CF 알고리즘에 따라 빈 상태에서 비어 있지 않은 상태로 큐가 이동되면 이를 QSG의 모든 큐 관리자에게 알립니다. 또한 대기 중인 *Getter*에 빠른 넣기 메커니즘을 적용할 수 없습니다.

z/OS 공유 큐 및 큐 공유 그룹에 대한 자세한 정보를 제공하는 위치

이 주제의 표를 사용하여 IBM MQ for z/OS에서 공유 큐 및 큐 공유 그룹을 사용하는 방법에 대한 자세한 정보를 찾으십시오.

표 *19.* 공유 큐 및 큐 공유 그룹에 대한 자세한 정보를 제공하는 위치

주제	**찾을 위치**
큐 공유 그룹 복구	225 페이지의 『z/OS에서의 복구 및 재시작』
큐 공유 그룹 보안	240 페이지의 『IBM MQ for z/OS의 보안 개념』
다른 큐 관리자에 대한 명령을 지시하는 개인용 및 글로벌 오브젝트 정의	z/OS
커플링 기능 계획 환경	커플링 기능 자원 정의
SMDS 환경 계획	공유 메시지 데이터 세트(SMDS) 환경 계획
DB2 환경 Db2 환경	Db2 환경 계획
공유 큐 설정 시스템 매개변수	149 페이지의 『공유 큐 및 큐 공유 그룹』
유틸리티 프로그램 큐 마이그레이션	z/OS 의 IBM MQ 유틸리티 참조
콘솔 메시지	IBM MQ for z/OS에 대한 메시지
MQSC 명령	MQSC 명령
IBM MQ 클러스터	큐 관리자 클러스터 구성
IBM MQ 분산 큐잉 채널 이름	분산 큐 관리 소개

표 19. 공유 큐 및 큐 공유 그룹에 대한 자세한 정보를 제공하는 위치 (계속)	
주제	**찾을 위치**
애플리케이션 작성	애플리케이션 설계 개요
MQCONNX 호출	MQCONNX

z/OS 그룹 내 큐잉

이 절에서는 z/OS 플랫폼에 고유한 IBM MQ for z/OS 함수인 그룹 내 큐잉에 대해 설명합니다. 이 기능은 큐 공유 그룹에 정의된 큐 관리자만이 사용할 수 있습니다.

큐 공유 그룹에 대한 정보는 149 페이지의 『공유 큐 및 큐 공유 그룹』의 내용을 참조하십시오.

그룹 내 큐잉 개념

채널을 정의하지 않고 큐 공유 그룹의 큐 관리자 사이에서 빠른 메시지 전송을 수행할 수 있습니다. 여기에서는 공유 전송 큐인 SYSTEM.QSG.TRANSMIT.QUEUE라고 하는 시스템 큐를 사용합니다. 큐 공유 그룹의 각 큐 관리자는 그룹 내 큐잉 에이전트를 호출하는 태스크를 시작하여 해당 큐 관리자에 대한 목적지로 지정된 메시지가 이 큐에 도달하기를 기다립니다. 해당 메시지가 감지되면 큐에서 제거되고 올바른 목적지 큐에 배치됩니다.

표준 이름 분석 규칙이 사용되지만, 그룹 내 큐잉(IGQ)이 사용 가능하고 대상 큐 관리자가 큐 공유 그룹 내에 있으면 SYSTEM.QSG.TRANSMIT.QUEUE를 사용하여 전송 큐 및 채널을 사용하는 대신 올바른 대상 큐 관리자로 메시지를 전송합니다.

큐 관리자 속성을 통해 그룹 내 큐잉을 사용 가능하게 합니다. 그룹 내 큐잉은 비지속 메시지는 동기점 외부로, 지속 메시지는 동기점 내부로 이동합니다. 대상 큐로 메시지를 전달하는 중 문제점을 발견하면, 그룹 내 큐잉은 데드-레터 큐로 메시지를 넣으려고 합니다. 데드-레터 큐가 가득 찼거나 정의되지 않은 경우 비지속 메시지는 제거되지만 지속 메시지는 백아웃되어 SYSTEM.QSG.TRANSMIT.QUEUE로 리턴되고 IGQ 에이전트는 성공할 때까지 메시지 전달을 시도합니다.

큐 공유 그룹 내 다른 큐 관리자에 있는 큐가 목적지로 지정된 메시지를 수신하는 인바운드 공유 채널은 올바른 목적지로 메시지를 홉*(Hop)*하기 위해 그룹 내 큐잉을 사용할 수 있습니다.

대상 큐 관리자에 처음 메시지를 전송하는 대신, 대상 큐가 공유 큐인 경우 로컬 큐 관리자에서 대상 큐에 직접 메시지를 넣으려는 경우 해당될 수 있습니다. 큐 관리자 속성 SQQMNAME을 사용하여 이를 제어할 수 있습니다. SQQMNAME의 값을 USE로 설정하면 **ObjectQMgrName**에서 지정한 큐 관리자에서 MQOPEN 명령이 수행됩니다.

그러나 대상 큐가 공유 큐이고 SQQMNAME의 값을 IGNORE로 설정하고 **ObjectQMgrName** 가 큐 공유 그룹에 있는 다른 큐 관리자의 값인 경우, 공유 큐가 로컬 큐 관리자에서 열립니다. 로컬 큐 관리자가 대상 큐를 열 수 없거나 큐에 메시지를 넣을 수 없는 경우, 메시지는 IGQ 또는 IBM MQ 채널을 통해 지정된 **ObjectQMgrName** 로 전송됩니다.

그룹 내 큐잉을 사용하여 큐 공유 그룹 내 리모트 큐 관리자에 있는 큐에 작은 메시지를 보다 효율적으로 전달할 수 있습니다. 그룹 내 큐잉도 대형 메시지를 지원합니다. 가장 큰 메시지는 전송 큐 헤더의 길이를 뺀 100MB입니다.

참고: 이 기능을 사용하는 경우 사용자는 큐 공유 그룹의 각 큐 관리자에 있는 큐에 대한 동일한 액세스 권한을 보유해야 합니다.

다음 다이어그램은 그룹 내 큐잉의 일반적인 예를 표시합니다.

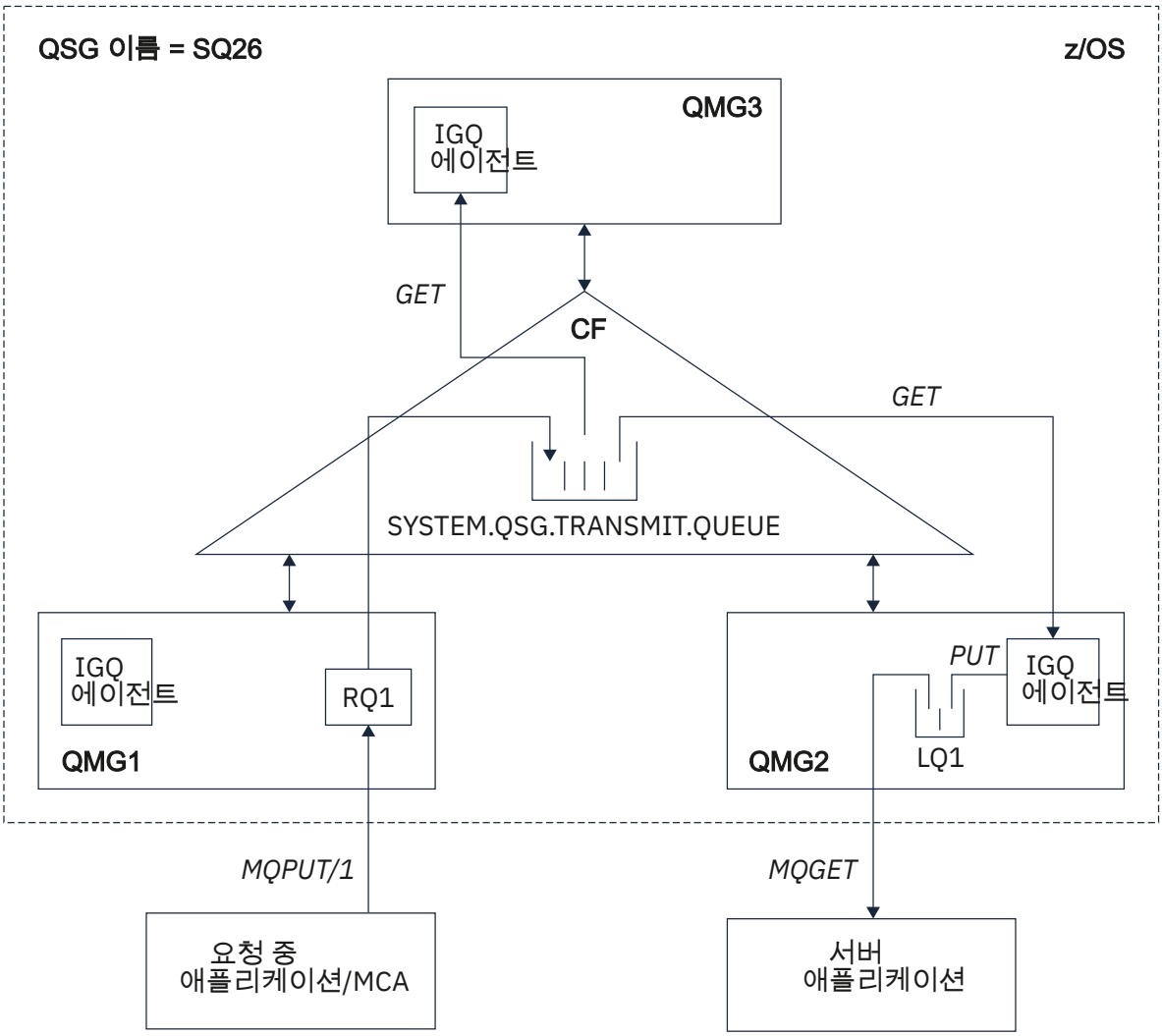


그림 67. 그룹 내 큐잉 예

다이어그램은 다음을 표시합니다.

- SQ26이라는 큐 공유 그룹에 정의된 3개의 큐 관리자(QMG1, QMG2, 및 QMG3)에서 실행 중인 IGQ 에이전트.
- 커플링 기능(CF)에 정의된 공유 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE.
- 큐 관리자 QMG1에 정의된 리모트 큐 정의.
- 큐 관리자 QMG2에 정의된 로컬 큐.
- 큐 관리자 QMG1에 연결된 애플리케이션 요청(이 애플리케이션은 MCA(Message Channel Agent)일 수 있음).
- 큐 관리자 QMG2에 연결된 서버 애플리케이션.
- SYSTEM.QSG.TRANSMIT.QUEUE에 놓인 요청 메시지.

그룹 내 큐잉 및 그룹 내 큐잉 에이전트

큐 관리자 초기화 중 IGQ 에이전트가 시작됩니다. 애플리케이션이 열리고 메시지를 리모트 큐에 넣을 때 로컬 큐 관리자는 그룹 내 큐잉이 메시지 전송에 사용되는지 여부를 판별합니다. 그룹 내 큐잉이 사용되면 로컬 큐 관리자는 메시지를 SYSTEM.QSG.TRANSMIT.QUEUE에 놓습니다. 대상 리모트 큐 관리자의 IGQ 에이전트는 메시지를 검색하여 목적지 큐에 놓습니다.

그룹 내 큐잉 용어

용어 설명: 그룹 내 큐잉, 그룹 내 큐잉을 사용하기 위한 공유된 전송 큐 및 그룹 내 큐잉 에이전트.

그룹 내 큐잉

그룹 내 큐잉은 채널 정의 필요 없이 큐 공유 그룹의 큐 관리자 간에 빠르고 저렴한 메시지 전송에 잠재적으로 영향을 줄 수 있습니다.

그룹 내 큐잉에서 사용하기 위한 공유된 전송 큐

각 큐 공유 그룹에는 그룹 내 큐잉에서 사용하기 위한 SYSTEM.QSG.TRANSMIT.QUEUE라는 공유된 전송 큐가 있습니다. 그룹 내 큐잉이 사용 가능한 경우, 리모트 큐를 연 경우 SYSTEM.QSG.TRANSMIT.QUEUE는 이름 해석 경로에 표시됩니다. 애플리케이션(MCA(Message Channel Agents) 포함)이 메시지를 리모트 큐에 넣으면, 로컬 큐 관리자는 SYSTEM.QSG.TRANSMIT.QUEUE에 빨리 전송하고 놓기 위한 메시지의 적합성을 판별합니다.

그룹 내 큐잉 에이전트

IGQ 에이전트는 큐 관리자 초기화에서 시작된 태스크로 적합한 메시지가 SYSTEM.QSG.TRANSMIT.QUEUE에 도달하길 대기합니다. IGQ 에이전트는 이 큐에서 적합한 메시지를 검색하여 목적지 큐에 전달합니다.

그룹 내 큐잉이 자체 내부 처리를 위해 큐 관리자 자체에서 사용되기 때문에 각 큐 관리자의 IGQ 에이전트가 항상 시작됩니다.

z/OS 그룹 내 큐잉의 이점

그룹 내 큐잉의 이점으로 축소된 시스템 정의, 축소된 시스템 관리, 개선된 성능, 마이그레이션 지원 및 큐 공유 그룹의 큐 관리자 간 멀티호핑 시 메시지 전달이 포함됩니다.

그룹 내 큐잉 이점은 다음과 같습니다.

축소된 시스템 정의
그룹 내 큐잉은 큐 공유 그룹 내 큐 관리자 간의 채널 정의의 필요성을 제거합니다.

축소된 시스템 관리
큐 공유 그룹의 큐 관리자 간에 정의된 채널이 없으므로 채널 관리에 대한 요구사항이 없습니다.

개선된 성능
메시지를 대상 큐에 전달하는 데 필요한 IGQ 에이전트가 하나이기 때문에(두 개의 중간 송신자 및 수신자 에이전트 대신), 그룹 내 큐잉을 사용하는 메시지 전달은 채널을 사용한 메시지 전달보다 저렴할 수 있습니다. 그룹 내 큐잉에서, 송신 컴포넌트의 필요성이 제거되었기 때문에 하나의 수신 컴포넌트만 있습니다. 이러한 공간 절약은 로컬 큐 관리자에 넣기 조작이 완료되면, 동기점 범위에 넣은 메시지의 경우 커미트되면, 대상 큐에 전달하기 위해 메시지가 대상 큐 관리자의 IGQ 에이전트에서 사용 가능하기 때문입니다.

마이그레이션 지원

큐 공유 그룹의 특정 큐 관리자에만 연결 중인 동안 큐 공유 그룹에 대한 외부적인 애플리케이션은 메시지를 큐 공유 그룹의 큐 관리자에 있는 큐에 전달합니다. 리모트 큐 관리자의 큐에 대한 목적지인 수신자 채널에 도착한 메시지가 그룹 내 큐잉을 사용하여 목적지 큐로 투명하게 송신될 수 있기 때문입니다. 이 기능을 사용하여 애플리케이션은 큐 공유 그룹에 외부적인 시스템을 변경하지 않고 큐 공유 그룹 간에 배치될 수 있습니다.

일반 구성은 다음 다이어그램에서 설명됩니다.

- 큐 관리자 QMG1에 연결된 요청 애플리케이션은 큐 관리자 QMG3의 로컬 큐로 메시지를 송신해야 합니다.
- 큐 관리자 QMG1은 큐 관리자 QMG2에만 연결됩니다.
- 채널을 사용하여 이전에 연결된 큐 관리자 QMG2 및 QMG3은 이제 큐 공유 그룹 SQ26의 멤버입니다.

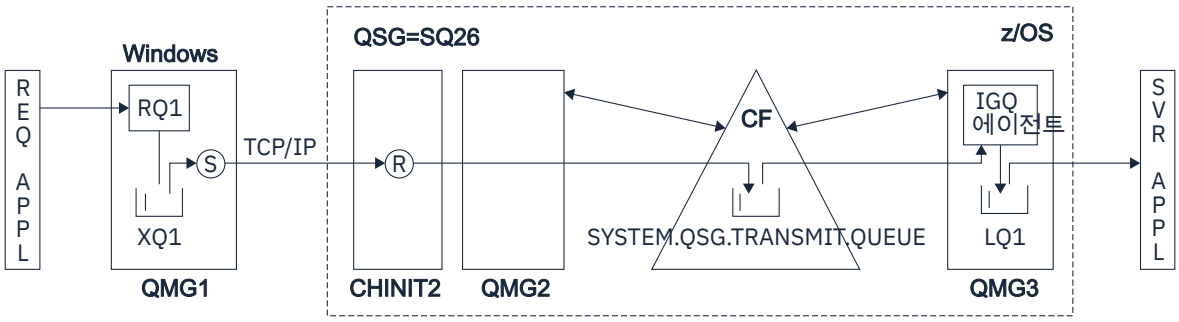


그림 *68.* 마이그레이션 지원 예

조작 플로우는 다음과 같습니다.

1. 요청 애플리케이션은 리모트 큐 관리자 QMG3의 로컬 큐 LQ1에 대한 대상인 메시지를 리모트 큐 정의 RQ1에 넣습니다.
2. Windows NT 워크스테이션에서 실행 중인 큐 관리자 QMG1은 메시지를 전송 큐 XQ1에 놓습니다.
3. QM1의 송신자 MCA(S)는 TCP/IP를 사용하여 메시지를 채널 시작기 CHINIT2의 수신자 MCA(R)로 전송합니다.
4. 채널 시작기 CHINIT2의 수신자 MCA(R)는 메시지를 공유 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE에 놓습니다.
5. 큐 관리자 QMG3의 IGQ 에이전트는 SYSTEM.QSG.TRANSMIT.QUEUE의 메시지를 검색하여 대상 로컬 큐 LQ1에 놓습니다.
6. 서버 애플리케이션은 대상 로컬 큐에서 메시지를 검색하여 처리합니다.

큐 공유 그룹의 큐 관리자 간에 멀티호핑 시 메시지 전달
마이그레이션 지원의 이전 다이어그램에서 큐 공유 그룹의 큐 관리자 간 멀티호핑 시 메시지의 전달도 설명합니다. 큐 공유 그룹의 다른 큐 관리자의 큐를 대상으로 하지만 큐 공유 그룹 내의 큐 관리자에 도달한 메시지는 그룹 내 큐잉을 사용하여 목적지 큐 관리자의 목적지 큐로 쉽게 전송될 수 있습니다.

z/OS 그룹 내 큐잉의 제한사항

그룹 내 큐잉의 제한사항은 그룹 내 큐잉을 사용하여 전송 가능한 메시지, 각 큐 관리자에 대한 그룹 내 큐잉 에이전트 수 및 그룹 내 큐잉 에이전트 시작 및 중지입니다.

이 주제에서는 그룹 내 큐잉의 제한사항에 대해 설명합니다.

그룹 내 큐잉을 사용하여 전송 가능한 메시지
그룹 내 큐잉은 커플링 기능(CF)에서 정의된 공유 전송 큐를 사용하기 때문에, 그룹 내 큐잉은 공유 큐에 대해 지원되는 최대 메시지 길이에서 전송 큐 헤더 길이(MQXQH)를 뺀 길이의 메시지를 전달하도록 제한됩니다.

큐 관리자당 그룹 내 큐잉 에이전트 수
하나의 IGQ 에이전트만 큐 공유 그룹의 각 큐 관리자에 대해 시작됩니다.

그룹 내 큐잉 에이전트 시작 및 중지
큐 관리자 초기화 중 IGQ 에이전트가 시작되며 큐 관리자 시스템 종료 중 종료됩니다. 길게 실행되는 자체 복구(비정상 종료 이벤트에), 태스크가 되도록 고안됩니다. SYSTEM.QSG.TRANSMIT.QUEUE 정의에 오류가 있는 경우 (예를 들어 이 큐가 Get(가져오기) 금지된 경우) IGQ 에이전트는 계속 재시도합니다. 큐 관리자가 여전히 활성인 동안 에이전트가 정상 종료되는 오류가 IGQ 에이전트에서 발생한 경우, ALTER QMGR IGQ(ENABLED) 명령을 발행하여 재시작될 수 있습니다. 이 명령은 큐 관리자를 재사용하지 않아도 됩니다.

큐 관리자 속성 IGQ를 ENABLED 또는 DISABLED로 설정
큐 관리자 속성 IGQ가 ENABLED 또는 DISABLED로 설정된 경우에는 기존 오브젝트 핸들은 이유 코드 MQRC_OBJECT_CHANGED로 무효화될 수도 있습니다. 자세한 정보는 그룹 내부 큐잉 시작하기를 참조하십시오.

z/OS 그룹 내 큐잉 시작하기

그룹 내 큐잉을 이 주제에서 설명된 것으로 사용 가능, 사용 불가능으로 설정하고 사용할 수 있습니다.

그룹 내 큐잉 사용 가능

큐 관리자에서 그룹 내 큐잉을 사용 가능으로 설정하려면 다음을 수행해야 합니다.

- SYSTEM.QSG.TRANSMIT.QUEUE라는 공유 전송 큐를 정의하십시오. 이 큐의 정의는 큐 공유 그룹용 SYSTEM 오브젝트에 대한 CSQINP2 샘플인 thlqual.SCSQPROC(CSQ4INSS)에서 찾을 수 있습니다. 이 큐는 그룹 내 큐잉이 적절하게 동작하도록 thlqual.SCSQPROC(CSQ4INSS)에 나타낸 대로 올바른 속성으로 정의되어야 합니다.
- IGQ 에이전트가 항상 큐 관리자 초기화에서 시작되기 때문에, 그룹 내 큐잉은 항상 인바운드 메시지 처리에서 사용 가능합니다. IGQ 에이전트는 SYSTEM.QSG.TRANSMIT.QUEUE에 놓여진 메시지를 처리합니다. 그러나 아웃바운드 처리를 위해 그룹 내 큐잉을 사용하려면 큐 관리자 속성 IGQ가 사용 가능으로 설정되어야 합니다.

  중요사항: 큐 관리자 속성 IGQ가 ENABLED로 설정된 경우에는 기존 오브젝트 핸들은 이유 코드 MQRC_OBJECT_CHANGED로 무효화될 수도 있습니다. 자세한 정보는 200 페이지의 『그룹 내 큐잉의 특정 특성』의 내용을 참조하십시오. 이 이유 코드에 대한 '프로그래머 응답' 절에 설명된 대로 이 상황을 처리하려면 애플리케이션을 코딩해야 합니다(세부사항은 2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED 참조).

또한 IGQ가 장기 실행 및 자체 복구 태스크로서 설계되었으므로 초기화 동안 시작되고 시스템 종료와 함께 종료됩니다. 자세한 정보는 193 페이지의 『그룹 내 큐잉의 제한사항』의 내용을 참조하십시오.

그룹 내 큐잉 사용 불가능

아웃바운드 메시지 전송을 위한 그룹 내 큐잉을 사용 불가능으로 설정하려면, 큐 관리자 속성 IGQ를 사용 불가능으로 설정하십시오. 그룹 내 큐잉이 특정 큐 관리자에 대해 사용 불가능한 경우, 아웃바운드 전송을 위해 그룹 내 큐잉이 사용 가능한 큐 관리자에 의해 SYSTEM.QSG.TRANSMIT.QUEUE에 놓여진 인바운드 메시지를 해당 큐 관리자의 IGQ 에이전트가 처리할 수 있습니다.

중요사항: 큐 관리자 속성 IGQ가 ENABLED로 설정된 경우에는 기존 오브젝트 핸들은 이유 코드 MQRC_OBJECT_CHANGED로 무효화될 수도 있습니다. 자세한 정보는 200 페이지의 『그룹 내 큐잉의 특정 특성』의 내용을 참조하십시오. 이 이유 코드에 대한 '프로그래머 응답' 절에 설명된 대로 이 상황을 처리하려면 애플리케이션을 코딩해야 합니다(세부사항은 2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED 참조).

또한 IGQ가 장기 실행 및 자체 복구 태스크로서 설계되었으므로 초기화 동안 시작되고 시스템 종료와 함께 종료됩니다. 자세한 정보는 193 페이지의 『그룹 내 큐잉의 제한사항』의 내용을 참조하십시오.

그룹 내 큐잉 사용

일단 그룹 내 큐잉을 사용 가능으로 설정하면, 사용할 수 있으므로 가능할 때마다 큐 관리자가 사용합니다. 즉, 애플리케이션이 메시지를 리모트 큐 정의, 완전한 리모트 큐 또는 클러스터 큐에 넣으면, 큐 관리자는 그룹 내 큐잉을 사용하여 메시지를 전달할 수 있는 경우 및 메시지가 SYSTEM.QSG.TRANSMIT.QUEUE에 놓여진 경우를 판별합니다. 적합한 메시지에 대해 큐 관리자가 SYSTEM.QSG.TRANSMIT.QUEUE를 다른 전송 큐보다 우선하여 사용하기 때문에 사용자 애플리케이션 또는 애플리케이션 큐를 변경할 필요가 없습니다.

z/OS 그룹 내 큐잉 구성

일반 그룹 내 큐잉 구성 외에 다른 구성이 가능합니다.

190 페이지의 『그룹 내 큐잉 개념』에서는 일반 구성에 대해 설명합니다.

관련 개념

195 페이지의 『그룹간 큐잉을 사용한 분산 큐잉(복수 전달 경로)』
짧은 메시지를 처리하는 애플리케이션의 경우, 큐 공유 그룹의 큐 관리자 간에 메시지를 전달하기 위해서만 그룹 내 큐잉을 구성하는 것이 가능할 수 있습니다.

196 페이지의 『그룹 내 큐잉을 통한 클러스터링(다중 전달 경로)』
큐 공유 그룹뿐 아니라 클러스터에 있도록 큐 관리자를 구성할 수 있습니다.

198 페이지의 『클러스터링, 그룹 내 큐잉 및 분산 큐잉』

큐 공유 그룹 외에 클러스터 멤버인 큐 관리자를 구성할 수 있으며 송신자/수신자 채널 쌍을 사용하여 분산 큐 관리자에게 연결됩니다.

z/OS 그룹간 큐잉을 사용한 분산 큐잉(복수 전달 경로)

짧은 메시지를 처리하는 애플리케이션의 경우, 큐 공유 그룹의 큐 관리자 간에 메시지를 전달하기 위해서만 그룹 내 큐잉을 구성하는 것이 가능할 수 있습니다.

채널 통신에 기반한 그룹 내 큐잉의 선택 항목은 CFSTRUCT 유형 레벨을 통해 제어될 수 있습니다 (4나 5가 아닌 3). SYSTEM.QSQ.TRANSMIT.QUEUE에 설정된 대로 최대 메시지 길이입니다.

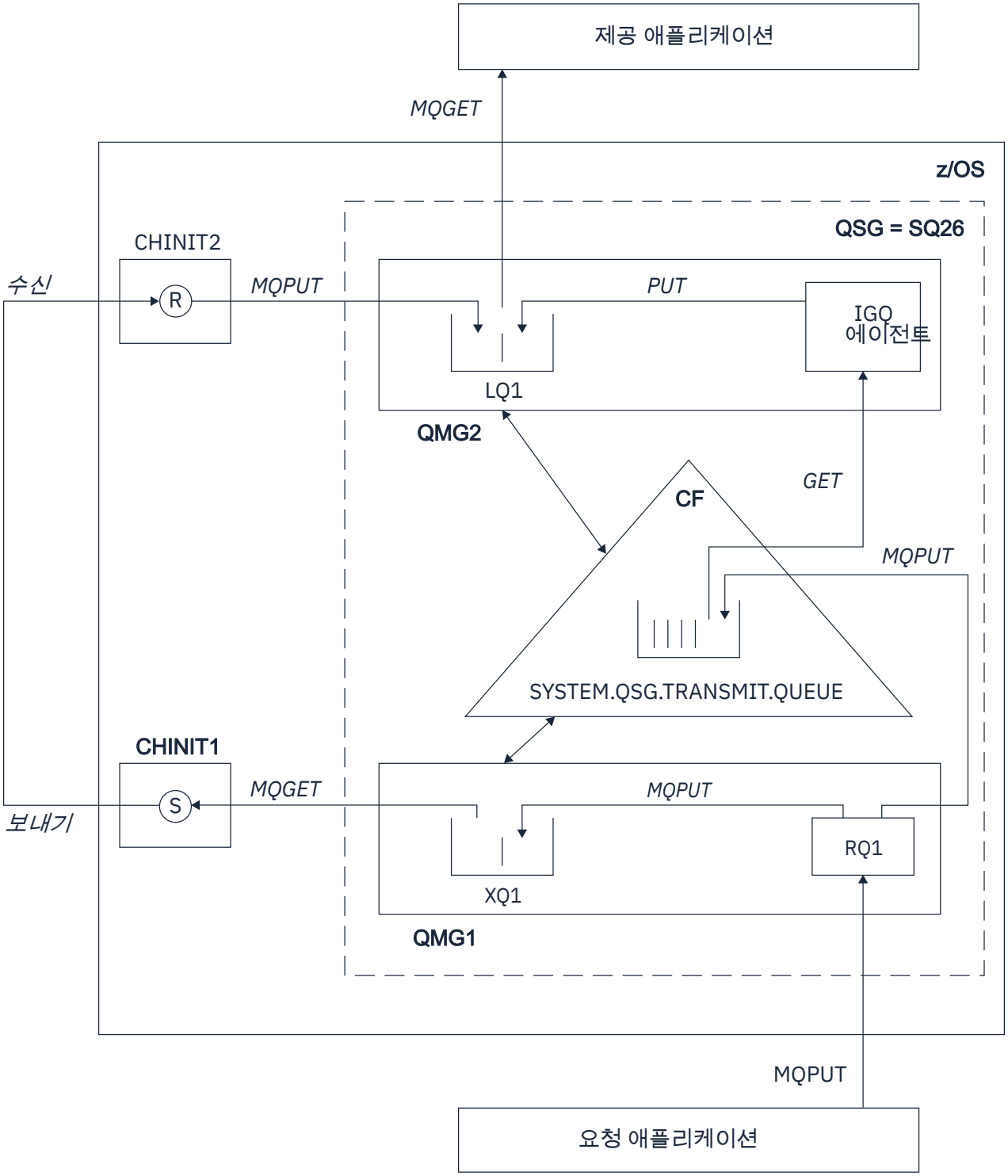


그림 69. 구성 예

열기/넣기 처리

1. 요청 애플리케이션이 리모트 큐 RQ1을 열 때 비공유 전송 큐 XQ1 및 공유 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE 모두에 대해 이름 해석이 이루어진다는 것에 주의하는 것이 중요합니다.
2. 요청 애플리케이션이 메시지를 리모트 큐에 넣을 때, 그룹 내 큐잉이 큐 관리자 및 메시지 특성의 아웃바운드 전송에 사용 가능한지 여부에 따라 메시지를 전송 큐 XQ1 또는 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE에 넣습니다. 큐 관리자는 모든 대용량 메시지를 전송 큐 XQ1에 놓으며 모든 작은 메시지를 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE에 놓습니다.
3. 전송 큐 XQ1이 꽉 차거나 사용 불가능하면, 대용량 메시지에 대한 넣기 요청이 적당한 리턴 및 이유 코드로 동기로 실패합니다. 그러나 작은 메시지에 대한 넣기 요청은 성공하여 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE에 놓입니다.
4. 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE가 꽉 차거나 넣을 수 없는 경우, 작은 메시지에 대한 넣기 요청이 적당한 리턴 및 이유 코드로 동기로 실패합니다. 그러나 대용량 메시지에 대한 넣기 요청은 계속 성공하여 전송 큐 XQ1에 놓입니다. 이 경우, 작은 메시지를 전송 큐에 넣는 시도를 할 수 없습니다.

대용량 메시지의 플로우

1. 요청 애플리케이션은 대용량 메시지를 리모트 큐, RQ1에 넣습니다.
2. 큐 관리자 QMG1은 메시지를 전송 큐 XQ1에 넣습니다.
3. 큐 관리자 QMG1의 송신자 MCA(S)는 전송 큐 XQ1의 메시지를 검색하여 큐 관리자 QMG2에 송신합니다.
4. 큐 관리자 QMG2의 수신자 MCA(R)는 메시지를 수신하여 목적지 큐 LQ1에 놓습니다.
5. 제공되는 애플리케이션은 큐 LQ1에서 메시지를 검색한 다음 처리합니다.

작은 메시지의 플로우

1. 요청 애플리케이션은 작은 메시지를 리모트 큐, RQ1에 넣습니다.
2. 큐 관리자 QMG1은 메시지를 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE에 넣습니다.
3. 큐 관리자 QMG2의 IGQ는 메시지를 검색하여 목적지 큐 LQ1에 놓습니다.
4. 제공되는 애플리케이션은 큐 LQ1에서 메시지를 검색합니다.

참고할 사항

1. 애플리케이션 요청은 메시지 전달에 사용되는 근본적인 메커니즘을 인식하지 않아도 됩니다.
2. 작은 메시지에 대해서 잠재적으로 더 빠른 메시지 전달 메커니즘이 달성될 수 있습니다.
3. 다중 경로를 메시지 전달에 사용할 수 있습니다(즉, 정상 채널 경로 및 그룹 내 큐잉 경로).
4. 잠재적으로 보다 빠른 그룹 내 큐잉 경로는 정상 채널 경로보다 우선하여 선택됩니다. 메시지 특성에 따라 메시지 전달은 두 경로로 나뉠 수 있습니다. 그러므로 메시지는 순서 없이 전달될 수 있습니다(정상 채널 경로만을 사용하여 메시지를 전달하는 경우에도 이 전달이 가능하더라도).
5. 경로가 선택되어 있고 메시지가 전송 큐에 놓인 경우, 선택된 경로만이 메시지 전달에 사용됩니다. SYSTEM.QSG.TRANSMIT.QUEUE에서의 미처리 메시지가 전송 큐 XQ1로 전환되지 않습니다.

z/OS 그룹 내 큐잉을 통한 클러스터링(다중 전달 경로)

큐 공유 그룹뿐 아니라 클러스터에 있도록 큐 관리자를 구성할 수 있습니다.

메시지가 클러스터 큐로 전송되고 로컬 및 원격 목적지 큐 관리자가 동일한 큐 공유 그룹에 있는 경우, 내부 그룹 큐잉이 메시지의 크기를 지원하면 작은 메시지의 전달(SYSTEM.QSG.TRANSMIT.QUEUE 사용) 및 큰 메시지의 전달에 내부 그룹 큐잉이 사용됩니다. 또한, SYSTEM.CLUSTER.TRANSMIT.QUEUE는 클러스터에 있지만 큐 공유 그룹 밖에 있는 큐 관리자에게 메시지를 전달하는 데 사용됩니다. 다음 다이어그램은 이 구성을 설명합니다(채널 시작기는 표시되지 않음).

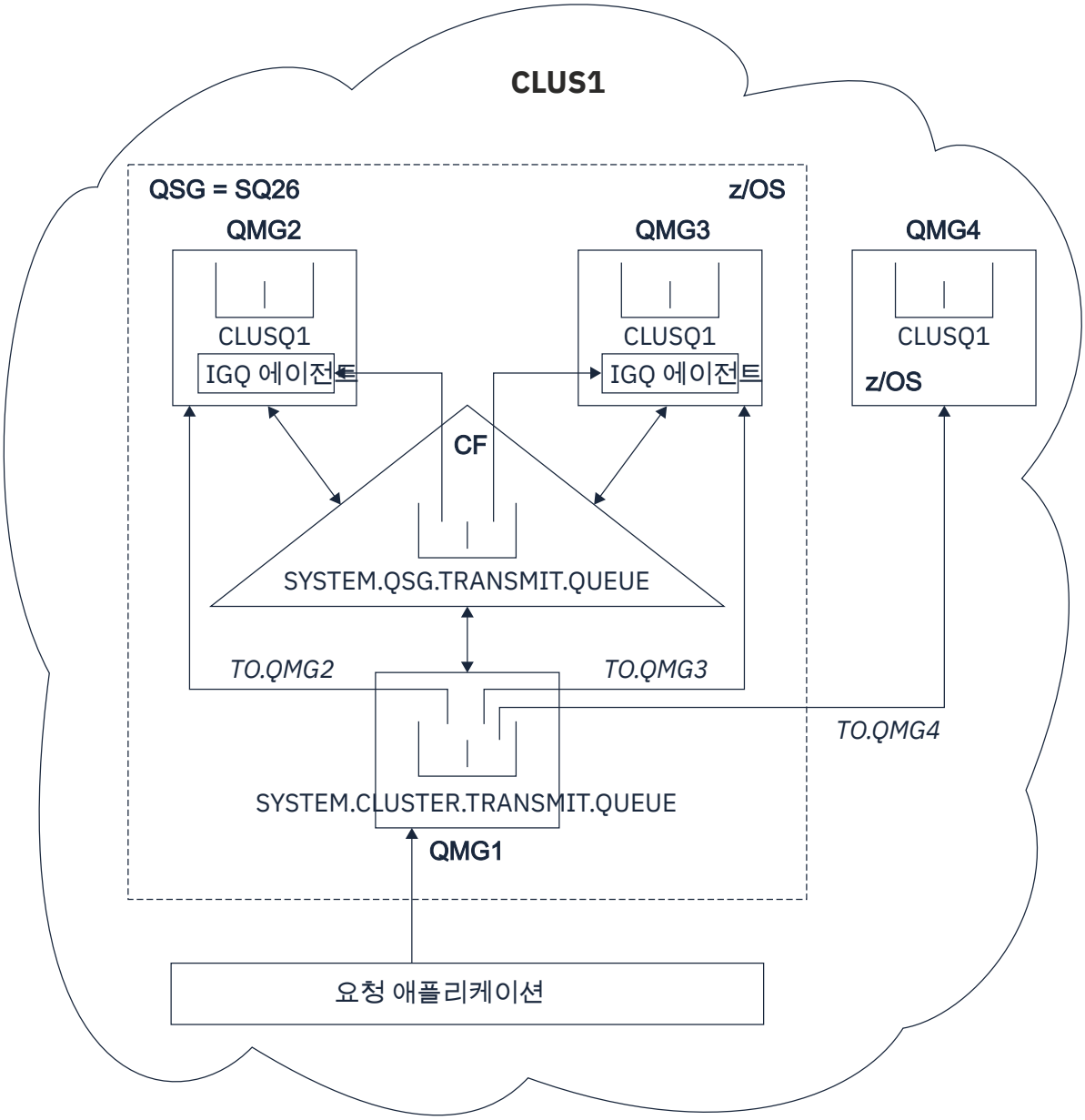


그림 *70.* 그룹 내 큐잉을 포함한 클러스터링 예제

다이어그램은 다음을 표시합니다.

- 클러스터 CLUS1에 구성된 4개의 z/OS 큐 관리자 QMG1, QMG2, QMG3 및 QMG4
- 큐 공유 그룹 SQ26에 구성된 큐 관리자 QMG1, QMG2 및 QMG3.
- 큐 관리자 QMG2 및 QMG3에서 실행 중인 IGQ 에이전트.
- QMG1에서 정의된 로컬 SYSTEM.CLUSTER.TRANSMIT.QUEUE.

  참고: 명확성을 위해 다른 큐 관리자에는 SYSTEM.CLUSTER.TRANSMIT.QUEUE가 표시되지 않습니다.
- CF에 정의된 공유 SYSTEM.QSG.TRANSMIT.QUEUE . 이는 CFLEVEL (3) RECOVER (YES) 속성으로 구성된 IBM MQ 구조에 있습니다.
- 클러스터 채널 TO.QMG2(QMG1을 QMG2에 연결), TO.QMG3(QMG1을 QMG3에 연결) 및 TO.QMG4(QMG1을 QMG4에 연결).
- 큐 관리자 QMG2, QMG3 및 QMG4에서 호스팅되는 클러스터 큐 CLUSQ1.

애플리케이션 요청은 MQOO_BIND_NOT_FIXED 옵션으로 클러스터 큐를 열어 넣을 때 클러스터 큐에 대한 대상 큐 관리자가 선택된다고 가정합니다.

선택된 대상 큐 관리자가 QMG2인 경우:

- 요청하는 애플리케이션에서 넣은 모든 큰 메시지는
  - SYSTEM.QSG.TRANSMIT.QUEUE가 CFLEVEL(3) 구조에 있으므로 QMG1의 SYSTEM.CLUSTER.TRANSMIT.QUEUE에 넣어집니다.
  - 클러스터 채널 TO.QMG2를 사용한 QMG2에 있는 클러스터 큐 CLUSQ1에 전송됩니다.
- 애플리케이션 요청에서 넣은 모든 작은 메시지는
  - 공유 전송 큐 SYSTEM.QSG.TRANSMIT.QUEUE에 넣어집니다. 이 큐는 RECOVER(YES) 속성으로 구성된 구조에 있으므로 지속 및 비지속 둘 다의 작은 메시지에 사용됩니다.
  - QMG2에 있는 IGQ 에이전트에서 검색됩니다.
  - QMG2에 있는 클러스터 큐 CLUSQ1에 넣어집니다.

선택된 대상 큐 관리자가 QMG4인 경우:

- QMG4는 큐 공유 그룹 SQ26의 멤버가 아니기 때문에, 애플리케이션 요청에서 넣은 모든 메시지는
  - QMG1에 있는 SYSTEM.CLUSTER.TRANSMIT.QUEUE에 넣어집니다.
  - 클러스터 채널 TO.QMG4를 사용한 QMG4에 있는 클러스터 큐 CLUSQ1에 전송됩니다.

참고할 사항

- 애플리케이션 요청은 메시지 전달에 사용되는 근본적인 메커니즘을 인식하지 않아도 됩니다.
- 큐 공유 그룹에서 큐 관리자 간에 작은 비지속 메시지를 전달하는 데 잠재적으로 보다 빠른 전달 메커니즘이 이루어집니다(동일 큐 관리자가 클러스터에 있더라도).
- 다중 경로를 메시지 전달에 사용할 수 있습니다(즉, 클러스터 경로 및 그룹 내 큐잉 경로 모두).
- 잠재적으로 보다 빠른 그룹 내 큐잉 경로는 클러스터 경로보다 우선하여 선택됩니다. 메시지 특성에 따라 메시지 전달은 두 경로로 나뉠 수 있습니다. 그러므로 메시지는 순서 없이 전달될 수 있습니다. 애플리케이션에서 지정된 MQOO_BIND_* 옵션에 상관 없이 이 전달이 가능할 수 있다는 점을 참고하는 것이 중요합니다. 열 때 MQOO_BIND_NOT_FIXED, MQOO_BIND_ON_OPEN, MQOO_BIND_ON_GROUP 또는 MQOO_BIND_AS_Q_DEF가 지정되었는지 여부에 따라, 그룹 내 큐잉은 메시지를 클러스터링이 수행한 것과 동일한 방식으로 분배합니다.
- 경로가 선택되어 있고 메시지가 전송 큐에 놓인 경우, 선택된 경로만이 메시지 전달에 사용됩니다. SYSTEM.QSG.TRANSMIT.QUEUE 상의 미처리 메시지는 SYSTEM.CLUSTER.TRANSMIT.QUEUE로 전환되지 않습니다.

z/OS 클러스터링, 그룹 내 큐잉 및 분산 큐잉

큐 공유 그룹 외에 클러스터 멤버인 큐 관리자를 구성할 수 있으며 송신자/수신자 채널 쌍을 사용하여 분산 큐 관리자에게 연결됩니다.

이 구성은 그룹 내 큐잉을 포함한 분산 큐잉 및 그룹 내 큐잉을 포함한 클러스터링의 조합입니다.

그룹 내 큐잉은 195 페이지의 『그룹간 큐잉을 사용한 분산 큐잉(복수 전달 경로)』에 설명되어 있습니다.

그룹 내 큐잉을 포함한 클러스터링은 196 페이지의 『그룹 내 큐잉을 통한 클러스터링(다중 전달 경로)』에서 설명되어 있습니다.

z/OS 그룹 내 큐잉 메시지

이 절에서는 SYSTEM.QSG.TRANSMIT.QUEUE에 넣는 메시지에 대해 설명합니다.

메시지 구조

전송 큐에 넣은 다른 모든 메시지와 마찬가지로, SYSTEM.QSG.TRANSMIT.QUEUE에 넣은 메시지는 전송 큐 헤더(MQXQH)로 접두부가 지정됩니다.

메시지 지속성

IBM WebSphere MQ 5.3 이상에서 공유 큐는 지속 및 비지속 메시지 모두를 지원합니다.

IGQ 에이전트가 비지속 메시지를 처리하는 동안 큐 관리자가 종료되거나 메시지 처리 중간에 IGQ 에이전트가 비정상적으로 종료된 경우 처리 중인 비지속 메시지가 유실될 수 있습니다. 복구가 필요한 경우 비지속 메시지 복구를 위해 애플리케이션이 지정되어야 합니다.

비지속 메시지에 대한 넣기(put) 요청이 IGQ 에이전트에서 발행되면 예상치 못하게 실패하여 처리 중인 메시지가 유실됩니다.

메시지 전달

IGQ 에이전트는 동기점 지점 범위 밖의 모든 비지속 메시지 및 동기점 지점 범위 내의 모든 지속 메시지를 검색하여 전달합니다. 이 경우, IGQ 에이전트는 동기점 통합기로 동작합니다. 그러므로 IGQ 에이전트는 빠른 비지속 메시지가 메시지 채널에서 처리되는 방식과 같이 비지속 메시지를 처리합니다. 빠른 비지속 메시지를 참조하십시오.

메시지 일괄 처리

IGQ 에이전트는 50MB 크기의 고정된 배치 크기를 사용합니다. 배치 내에서 검색된 지속 메시지는 50개의 메시지 간격으로 커미트됩니다. SYSTEM.QSG.TRANSMIT.QUEUE에서의 검색에 사용 가능한 추가 메시지가 없는 경우 에이전트는 지속 메시지로 구성된 배치를 커미트합니다.

메시지 크기

SYSTEM.QSG.TRANSMIT.QUEUE로 넣을 수 있는 메시지의 최대 크기는 공유 큐에 대해 지원되는 최대 메시지 길이에서 전송 큐 헤더(MQXQH) 길이를 뺀 크기입니다.

기본 메시지 지속성 및 기본 메시지 우선순위

SYSTEM.QSG.TRANSMIT.QUEUE가 열릴 때 설정된 큐 이름 해석 경로에 있는 경우 기본 지속 및 기본 우선순위(또는 기본 지속 또는 기본 우선순위가 있는)로 넣은 메시지에 대한 일반적인 규칙이 사용된 지속 값 및 기본 우선순위가 있는 큐의 선택 부분에 적용됩니다. (큐 선택 규칙에 대한 자세한 정보는 IBM MQ 메시지 절을 참조하십시오.)

관련 개념

199 페이지의 『미전달/미처리된 메시지』
이 주제에서는 SYSTEM.QSG.TRANSMIT.QUEUE에서 미전달되고 미처리된 메시지에 발생하는 내용을 설명합니다.

200 페이지의 『보고 메시지 - 내부 그룹 큐잉』
이 주제는 보고 메시지인 도착 확인, 전달 확인, 만기 보고 및 예외 보고를 설명합니다.

z/OS 미전달/미처리된 메시지

이 주제에서는 SYSTEM.QSG.TRANSMIT.QUEUE에서 미전달되고 미처리된 메시지에 발생하는 내용을 설명합니다.

IGQ 에이전트가 메시지를 목적지 큐에 전달할 수 없는 경우 IGQ 에이전트는 다음을 수행합니다.

- MQRO_DISCARD_MSG 보고 옵션(미전달 메시지에 대한 MQMD의 보고 옵션 필드가 그와 같아야 함을 표시하는 경우)을 이행하며 미전달 메시지를 제거합니다.
- 메시지를 아직 제거하지 않은 경우 미전달 메시지를 목적지 큐 관리자의 데드 레터 큐에 놓으려고 합니다. IGQ 에이전트는 데드 레터 큐 헤더(MQDLH)를 포함한 메시지에 접두부를 붙입니다.

데드 레터 큐가 정의되지 않았거나 미전달 메시지를 데드 레터 큐에 넣을 수 없는 경우 및 미전달 메시지가 다음과 같은 경우:

- 지속적인 경우, IGQ 에이전트는 처리 중인 지속 메시지의 현재 배치를 백아웃하며 재시도 상태를 입력합니다. 자세한 정보는 200 페이지의 『그룹 내 큐잉의 특정 특성』의 내용을 참조하십시오.
- 비지속인 경우, IGQ 에이전트는 메시지를 제거하며 다음 메시지를 계속 처리합니다.

연관된 IGQ 에이전트가 모든 메시지를 처리할 시간을 갖기 전에 큐 공유 그룹의 큐 관리자가 종료되면, 다음에 큐 관리자가 시작될 때까지 처리되지 않은 메시지가 SYSTEM.QSG.TRANSMIT.QUEUE에 남습니다. 그런 다음 IGQ 에이전트는 메시지를 검색하여 목적지 큐에 전달합니다.

SYSTEM.QSG.TRANSMIT.QUEUE의 모든 메시지가 처리되기 전에 커플링 기능이 실패하면 처리되지 않은 비지속 메시지가 유실됩니다.

IBM에서는 애플리케이션이 메시지를 바로 전송 큐에 넣지 않기를 권장합니다. 애플리케이션이 메시지를 바로 SYSTEM.QSG.TRANSMIT.QUEUE에 넣지 않으면, IGQ 에이전트는 이 메시지를 처리할 수 없어 SYSTEM.QSG.TRANSMIT.QUEUE에 남습니다. 그러면 사용자는 자체 방법을 사용하여 처리되지 않은 메시지를 다뤄야 합니다.

z/OS 보고 메시지 - 내부 그룹 큐잉

이 주제는 보고 메시지인 도착 확인, 전달 확인, 만기 보고 및 예외 보고를 설명합니다.

COA(Confirmation Of Arrival)/COD(Confirmation Of Delivery) 보고 메시지
그룹 내 큐잉이 사용되면 COA 및 COD 메시지는 큐 관리자에서 생성됩니다.

만기 보고 메시지
만기 보고 메시지는 큐 관리자에서 생성됩니다.

예외 보고 메시지

미전달 메시지의 메시지 디스크립터의 보고 옵션에서 지정된 MQRO_EXCEPTION_* 보고 옵션에 따라, IGQ 에이전트가 필수 예외 보고를 생성하여 지정된 응답 대상 큐에 놓습니다. 그룹 내 큐잉을 사용하여 예외 보고를 목적지 응답 대상 큐에 전달할 수 있습니다.

보고 메시지의 지속성은 미전달 메시지의 지속성과 동일합니다. IGQ 에이전트가 목적지 응답 대상 큐의 이름을 해석하는 데 실패하거나 응답 메시지를 전송 큐에 넣는 데 실패한 경우(목적지 응답 대상 큐에 후속 전송하기 위한), 예외 보고를 보고 메시지가 생성된 큐 관리자의 데드 레터 큐에 넣으려고 합니다. 불가능한 경우 미전달 메시지가 다음의 경우일 때:

- 지속적인 경우, IGQ 에이전트는 예외 보고를 제거하며 현재 메시지 배치를 백아웃하며 재시도 상태를 입력합니다. 자세한 정보는 200 페이지의 『그룹 내 큐잉의 특정 특성』의 내용을 참조하십시오.
- 비지속인 경우, IGQ 에이전트는 예외 보고를 제거하며 SYSTEM.QSG.TRANSMIT.QUEUE에서 다음 메시지를 계속 처리합니다.

z/OS 그룹 내 큐잉의 보안

이 주제에서는 그룹 내 큐잉의 보안 대책에 대해 설명합니다.

큐 관리자 속성 IGQAUT(IGQ 권한) 및 IGQUSER(IGQ 에이전트 사용자 ID)를 설정하여 IGQ 에이전트가 목적지 큐를 열 때 수행되는 보안 점검 레벨을 제어할 수 있습니다.

그룹 내 큐잉 권한(IGQAUT)

IGQAUT 속성을 설정하여 수행될 보안 검사 유형을 표시하고 목적지 큐에 메시지를 넣을 수 있는 권한을 설정할 때 IGQ 에이전트가 사용할 사용자 ID를 판별하는 데 사용할 수 있습니다.

IGQAUT 속성은 채널 정의에서 사용할 수 있는 PUTAUT 속성과 유사합니다.

그룹 내 큐잉 사용자 ID(IGQUSER)

IGQUSER 속성은 목적지 큐에 메시지를 넣을 수 있는 권한을 설정할 때 IGQ 에이전트가 사용할 사용자 ID를 지정하는 데 사용할 수 있습니다.

IGQAUT 속성은 채널 정의에서 사용할 수 있는 MCAUSER 속성과 유사합니다.

z/OS 그룹 내 큐잉의 특정 특성

이 절은 유효하지 않은 오브젝트 핸들, 그룹 내 큐잉 에이전트의 자체 복구 및 재시도 용량 그리고 그룹 내 큐잉 에이전트 및 직렬화를 포함하는 그룹 내 큐잉의 특정 특성에 대해 설명합니다.

유효하지 않은 오브젝트 핸들(MQRC_OBJECT_CHANGED)

오브젝트를 연 후 오브젝트의 속성이 변경된 것을 발견하면 다음 사용 시 큐 관리자는 MQRC_OBJECT_CHANGED로 오브젝트 핸들의 유효성을 검증하지 않습니다.

그룹 내 큐잉은 오브젝트 핸들 유효성 검증 실패를 위한 다음 규칙을 소개합니다.

- 열 때 그룹 내 큐잉이 사용 가능하지만 넣을 때 그룹 내 큐잉이 사용 불가능하기 때문에 열린 프로세스 중 SYSTEM.QSG.TRANSMIT.QUEUE가 이름 해석 경로에 포함되면, 큐 관리자는 오브젝트 핸들의 유효성 검증을 하지 않으며 MQRC_OBJECT_CHANGED로 넣기 요청이 실패합니다.
- 열 때 그룹 내 큐잉이 사용 불가능하지만 넣을 때 그룹 내 큐잉이 사용 가능하기 때문에 열린 프로세스 중 SYSTEM.QSG.TRANSMIT.QUEUE가 이름 해석 경로에 포함되지 않으면, 큐 관리자는 오브젝트 핸들의 유효성 검증을 하지 않으며 MQRC_OBJECT_CHANGED로 넣기 요청이 실패합니다.
- 열 때 그룹 내 큐잉이 사용 가능하지만 넣을 때 SYSTEM.QSG.TRANSMIT.QUEUE 정의가 변경되기 때문에 열린 프로세스 중 SYSTEM.QSG.TRANSMIT.QUEUE가 이름 해석 경로에 포함되면, 큐 관리자는 오브젝트 핸들의 유효성 검증을 하지 않으며 MQRC_OBJECT_CHANGED로 넣기 요청이 실패합니다.

그룹 내 큐잉 에이전트의 자체 복구

IGQ 에이전트가 비정상적으로 종료된 경우, 메시지 CSQM067E가 발행되며 IGQ 에이전트가 다시 시작됩니다.

그룹 내 큐잉 에이전트의 재시도 용량

IGQ 에이전트에 SYSTEM.QSG.TRANSMIT.QUEUE 접근 문제가 있는 경우(정의되지 않았기 때문이거나 예를 들어 올바르지 않은 속성으로 정의되거나 Get로 상속되거나 또는 기타 다른 이유로), IGQ 에이전트가 재시도 상태가 됩니다.

IGQ 에이전트는 짧고 긴 재시도 수 및 간격을 적용합니다. 변경될 수 없는 수와 간격의 값은 다음과 같습니다.

표 *20.* 짧고 긴 재시도 수 및 간격 값

상수	**값**
짧은 재시도 수	10
짧은 재시도 간격	60초 = 1분
긴 재시도 수	999,999,999
긴 재시도 간격	1200초 = 20분

그룹 내 큐잉 에이전트 및 직렬화

피어 복구가 여전히 진행 중인 동안 IGQ 에이전트에서 공유 큐에 대한 액세스 직렬화가 실패할 수 있습니다.

IGQ 에이전트가 공유 큐나 큐에서 커미트되지 않은 메시지를 다루는 동안 큐 공유 그룹에 큐 관리자가 실패하면, IGQ 에이전트가 종료되며 공유된 큐 피어 복구가 실패 큐 관리자에 대해 발생합니다. 공유된 큐 피어 복구가 비동기 활동이기 때문에, 실패 큐 관리자 및 해당 큐 관리자의 IGQ 에이전트가 공유된 큐 피어 복구가 완료되기 전에 다시 시작할 수 있는 가능성이 있습니다. 커미트된 메시지가 앞서 처리되고 메시지 순서에 관계 없이 복구될 수 있는 가능성이 있습니다. 메시지를 순서 없이 처리하지 않게 위해 MQCONNX API 호출을 발행하여 IGQ 에이전트는 공유 큐에 대한 액세스를 직렬화합니다.

피어 복구가 여전히 진행 중인 동안 IGQ 에이전트에서 공유 큐에 대한 액세스 직렬화가 실패할 수 있습니다. 오류 메시지가 발행되며 IGQ 에이전트가 재시도 상태에 놓입니다. 큐 관리자 피어 복구가 완료되면, 예를 들어 다음 재시도 시 IGQ 에이전트를 시작할 수 있습니다.

z/OS z/OS의 스토리지 관리

IBM MQ for z/OS에서는 영구 및 임시 데이터 구조가 필요하며 이 데이터를 저장하기 위해 페이지 세트 및 메모리 버퍼를 사용합니다. 이 주제에서는 IBM MQ에서 이러한 페이지 세트 및 버퍼를 사용하는 방법에 대한 자세한 정보를 제공합니다.

관련 개념

202 페이지의 『IBM MQ for z/OS의 페이지 세트』
IBM MQ for z/OS가 메시지를 저장하는 데 페이지 세트를 사용하는 방법을 이해하려면 이 토픽을 사용하십시오.

202 페이지의 『IBM MQ for z/OS의 스토리지 클래스』
스토리지 클래스는 큐 관리자가 페이지 세트에 큐를 맵핑할 수 있는 IBM MQ for z/OS 개념입니다. 스토리지 클래스를 사용하여 어느 데이터 세트가 어느 큐에서 사용될지를 제어할 수 있습니다.

204 페이지의 『IBM MQ for z/OS의 버퍼 및 버퍼 풀』
IBM MQ for z/OS는 버퍼 및 버퍼 풀을 사용하여 임시로 데이터를 캐시합니다. 버퍼가 구성되고 사용되는 방식을 자세히 이해하는 데 이 토픽을 사용하십시오.

관련 참조

205 페이지의 『IBM MQ for z/OS의 스토리지 관리에 대한 자세한 정보를 제공하는 위치』
IBM MQ for z/OS에 대한 스토리지 관리의 추가 정보를 찾으려면 이 토픽을 참조로 사용하십시오.

z/OS IBM MQ for z/OS의 페이지 세트

IBM MQ for z/OS가 메시지를 저장하는 데 페이지 세트를 사용하는 방법을 이해하려면 이 토픽을 사용하십시오.

페이지 세트는 특별히 IBM MQ에서 사용하기 위해 형식화되는 VSAM 선형 데이터 세트입니다. 페이지 세트는 대부분의 메시지와 오브젝트 정의를 저장하는 데 사용됩니다.

이에 대한 예외로, Db2의 공유 저장소에 저장되는 글로벌 정의와 공유 큐의 메시가 있습니다. 이들은 큐 관리자 페이지 세트에 저장되지 않습니다. 공유 큐에 대한 자세한 정보는 149 페이지의 『공유 큐 및 큐 공유 그룹』을 참조하고 글로벌 정의에 대한 자세한 정보는 개인용 및 글로벌 정의를 참조하십시오.

IBM MQ 페이지 세트는 최대 크기가 64GB일 수 있습니다. 각 페이지 세트는 범위가 정수 00 - 99 사이인 페이지 세트 ID(PSID)로 식별됩니다. 각 큐 관리자에는 고유한 페이지 세트가 있습니다.

IBM MQ는 페이지 세트 0(PSID=00)을 사용하여 오브젝트 정의 및 큐 관리자와 관련된 기타 중요한 정보를 저장합니다. IBM MQ의 정상 조작 시 메시지를 저장하는 데 이를 사용하지 않도록 페이지 세트 0이 가득차지 않는 것이 중요합니다.

또한 시스템 성능을 향상시키려면 다른 페이지 세트에 배치하여 장기간 메시지에서 단기간 메시지를 분리해야 합니다.

페이지 세트를 형식화해야 하며, 이러한 형식화를 위해 IBM MQ가 FORMAT 유틸리티를 제공합니다. 페이지 세트 형식화(FORMAT)를 참조하십시오. IBM MQ 서브시스템에 페이지 세트도 정의해야 합니다.

가득찬 경우 동적으로 페이지 세트를 확장하도록 IBM MQ for z/OS를 구성할 수 있습니다. IBM MQ은(는) 사용 가능한 디스크 스토리지 공간이 충분한 경우 123개 논리 익스텐트가 존재할 때까지 필요한 경우 페이지 세트를 계속 확장합니다. 범위는 이러한 방식으로 선형 데이터 세트가 정의된 경우 볼륨으로 확장될 수 있습니다. 그러나 IBM MQ는 64GB 넘게 페이지 세트를 확장할 수 없습니다.

한 IBM MQ 큐 관리자의 페이지 세트를 다른 IBM MQ 큐 관리자에서 사용하거나 큐 관리자 이름을 변경할 수 없습니다. 큐 관리자에서 다른 큐 관리자로 데이터를 전송하려는 경우 첫 번째 큐 관리자에서 모든 오브젝트 및 메시지를 로드 해제하고 다른 곳으로 다시 로드해야 합니다.

V6 이전의 릴리스를 실행하는 큐 관리자에서 4GB보다 큰 페이지 세트를 사용할 수 없습니다. 마이그레이션 기간 중에 코드의 이전 릴리스로 폴백해야 할 수도 있습니다.

- 4GB를 초과하도록 페이지 세트 0을 변경하지 마십시오.
- 이전 릴리스에서 큐 관리자를 재시작할 때 4GB를 초과하는 다른 페이지 세트는 오프라인 상태로 남아 있습니다.

4GB를 초과하여 확장할 수 있는 기존 페이지 세트를 마이그레이션하는 방법에 대한 추가 정보는 4GB보다 크게 페이지 세트 정의를 참조하십시오.

관리자가 동적으로 실행 중인 큐 관리자에 페이지 세트를 추가하거나 실행 중인 큐 관리자에서 페이지 세트(페이지 세트 0 제외)를 제거할 수 있습니다. DEFINE PSID 명령은 큐 관리자 재시작을 완료한 후 명령이 DSN 키워드를 포함하는 경우에만 실행할 수 있습니다.

z/OS IBM MQ for z/OS의 스토리지 클래스

스토리지 클래스는 큐 관리자가 페이지 세트에 큐를 맵핑할 수 있는 IBM MQ for z/OS 개념입니다. 스토리지 클래스를 사용하여 어느 데이터 세트가 어느 큐에서 사용될지를 제어할 수 있습니다.

스토리지 클래스 소개

스토리지 클래스는 하나 이상의 큐를 페이지 세트에 맵핑합니다. 즉, 해당 큐의 메시지가 해당 페이지 세트에 저장됨을 의미합니다.

스토리지 클래스에서는 관리, 데이터 세트 공간 및 로드 관리 또는 애플리케이션 분리 목적으로 비공유 메시지 데이터가 저장되는 위치를 제어할 수 있습니다. 또한 IMS 브리지를 사용하는 경우 스토리지 클래스를 사용하여 ' IMS 영역의 XCF 그룹 및 구성원 이름을 정의할 수 있습니다(' 253 페이지의 『IBM MQ 및 IMS』 ' 설명 참조).

공유 큐의 메시지는 페이지 세트에 저장되지 않으므로 공유 큐는 페이지 세트 맵핑을 얻기 위해 스토리지 클래스를 사용하지 않습니다.

스토리지 클래스가 작동하는 방식

- 페이지 세트 ID(PSID)를 지정하여 DEFINE STGCLASS 명령을 통해 스토리지 클래스를 정의합니다.
- 큐를 정의하는 경우 STGCLASS 속성에서 스토리지 클래스를 지정합니다.

다음 예에서 로컬 큐 QE5는 스토리지 클래스 ARC2를 통해 페이지 세트 21에 맵핑됩니다.

```
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE QLOCAL(QE5) STGCLASS(ARC2)
```

즉, 큐 QE5에 넣은 메시지가 페이지 세트 21에 저장됨을 의미합니다(DASD에 쓸 수 있을 정도로 충분히 오래 큐에 머물러 있는 경우).

둘 이상의 큐가 동일한 스토리지 클래스를 사용할 수 있으며 원하는 만큼 많은 스토리지 클래스를 정의할 수 있습니다. 예를 들어 이전 예를 확장하여 다음과 같이 많은 스토리지 클래스 및 큐 정의를 포함할 수 있습니다.

```
DEFINE STGCLASS(ARC1) PSID(05)
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE STGCLASS(MAXI) PSID(05)
DEFINE QLOCAL(QE1) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE2) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE3) STGCLASS(MAXI) ...
DEFINE QLOCAL(QE4) STGCLASS(ARC2) ...
DEFINE QLOCAL(QE5) STGCLASS(ARC2) ...
```

203 페이지의 그림 71에서 스토리지 클래스 ARC1 및 MAXI 모두가 페이지 세트 05에 연관됩니다. 따라서 큐 QE1, QE2 및 QE3은 페이지 세트 05에 맵핑됩니다. 마찬가지로 스토리지 클래스 ARC2는 큐 QE4 및 QE5를 페이지 세트 21에 연관합니다.

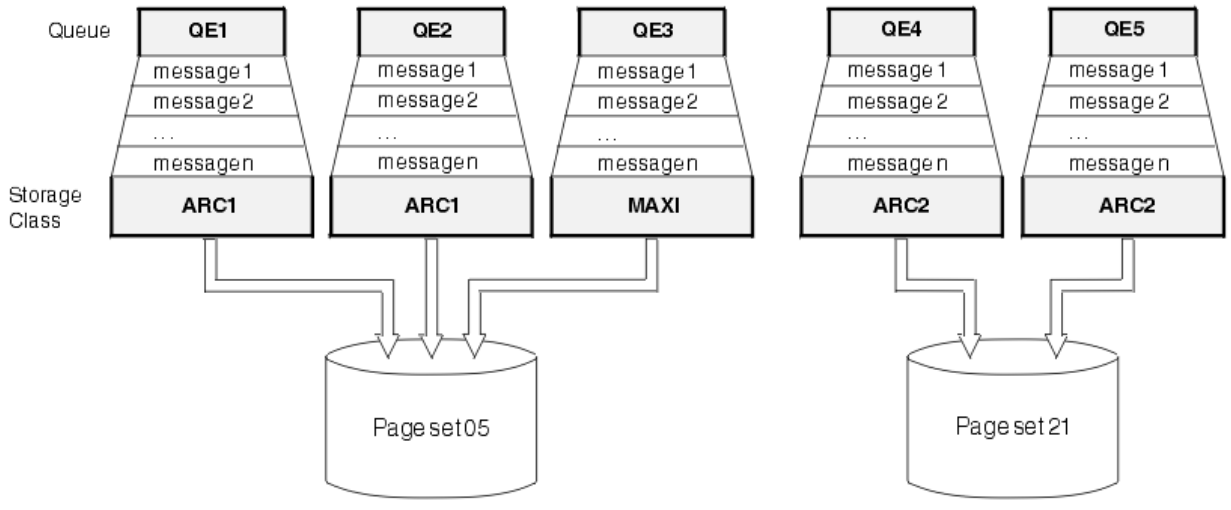


그림 *71.* 스토리지 클래스를 통해 페이지 세트에 큐 맵핑

스토리지 클래스를 지정하지 않고 큐를 정의하는 경우 IBM MQ는 기본 스토리지 클래스를 사용합니다.

존재하지 않는 스토리지 클래스 이름을 지정하는 큐에 메시지를 넣으면 애플리케이션은 오류를 수신합니다. 기존 스토리지 클래스 이름을 제공하거나 큐에서 이름을 지정한 스토리지 클래스를 작성하려면 큐 정의를 대체해야 합니다.

다음 경우에만 스토리지 클래스를 변경할 수 있습니다.

- 이 스토리지 클래스를 사용하는 모든 큐가 비어 있으며 커미트되지 않은 활동이 없습니다.
- 이 스토리지 클래스를 사용하는 모든 큐가 닫혔습니다.

z/OS IBM MQ for z/OS의 버퍼 및 버퍼 풀

IBM MQ for z/OS는 버퍼 및 버퍼 풀을 사용하여 임시로 데이터를 캐시합니다. 버퍼가 구성되고 사용되는 방식을 자세히 이해하는 데 이 토픽을 사용하십시오.

효율성을 위해 IBM MQ는 캐싱 양식을 사용하므로 메시지 및 오브젝트 정의는 DASD에서 페이지 세트에 저장되기 전에 버퍼에 임시로 저장됩니다. 단기간 메시지, 즉, 수신한 후 잠시 큐에서 검색되는 메시지는 버퍼에 계속 저장될 수 있습니다. 이 캐싱 활동은 IBM MQ의 컴포넌트인 버퍼 관리자에서 제어됩니다.

버퍼는 버퍼 풀로 구성됩니다. 각 큐 관리자에 대해 최대 100개(0 - 99)까지 버퍼 풀을 정의할 수 있습니다.

205 페이지의 그림 72에 간략히 설명된 오브젝트 및 메시지 유형 분리, 애플리케이션이 보유할 수 있는 데이터 격리 요구사항과 일치하는 최소 버퍼 풀 수를 사용하는 것이 좋습니다. 각 버퍼 길이는 4KB입니다. 버퍼 풀은 이 모드에서 기본적으로 31비트 스토리지를 사용하며, 최대 버퍼 수는 큐 관리자 주소 공간에서 사용 가능한 31비트 스토리지 크기로 판별됩니다. 버퍼에 대해 약 70%를 초과하여 사용하지 마십시오. 또는 64비트스토리지에서 버퍼 풀 스토리지를 할당할 수 있습니다 (**DEFINE BUFFPOOL** 명령의 LOCATION 속성 사용). 64비트 스토리지가 사용되도록 LOCATION(ABOVE)를 사용하는 데는 두 가지 이점이 있습니다. 첫째는 버퍼 풀이 훨씬 커질 수 있도록 훨씬 많은 64비트 스토리지를 사용할 수 있다는 점이고 둘째는 31비트 스토리지를 다른 기능에 의해 사용 가능하다는 점입니다. 일반적으로 버퍼가 많을수록 버퍼링 효율과 IBM MQ의 성능이 향상됩니다.

205 페이지의 그림 72에서는 메시지, 버퍼, 버퍼 풀 및 페이지 세트 사이의 관계를 표시합니다. 버퍼 풀은 하나 이상의 페이지 세트와 연관됩니다. 각 페이지 세트는 단일 버퍼 풀과 연관됩니다.

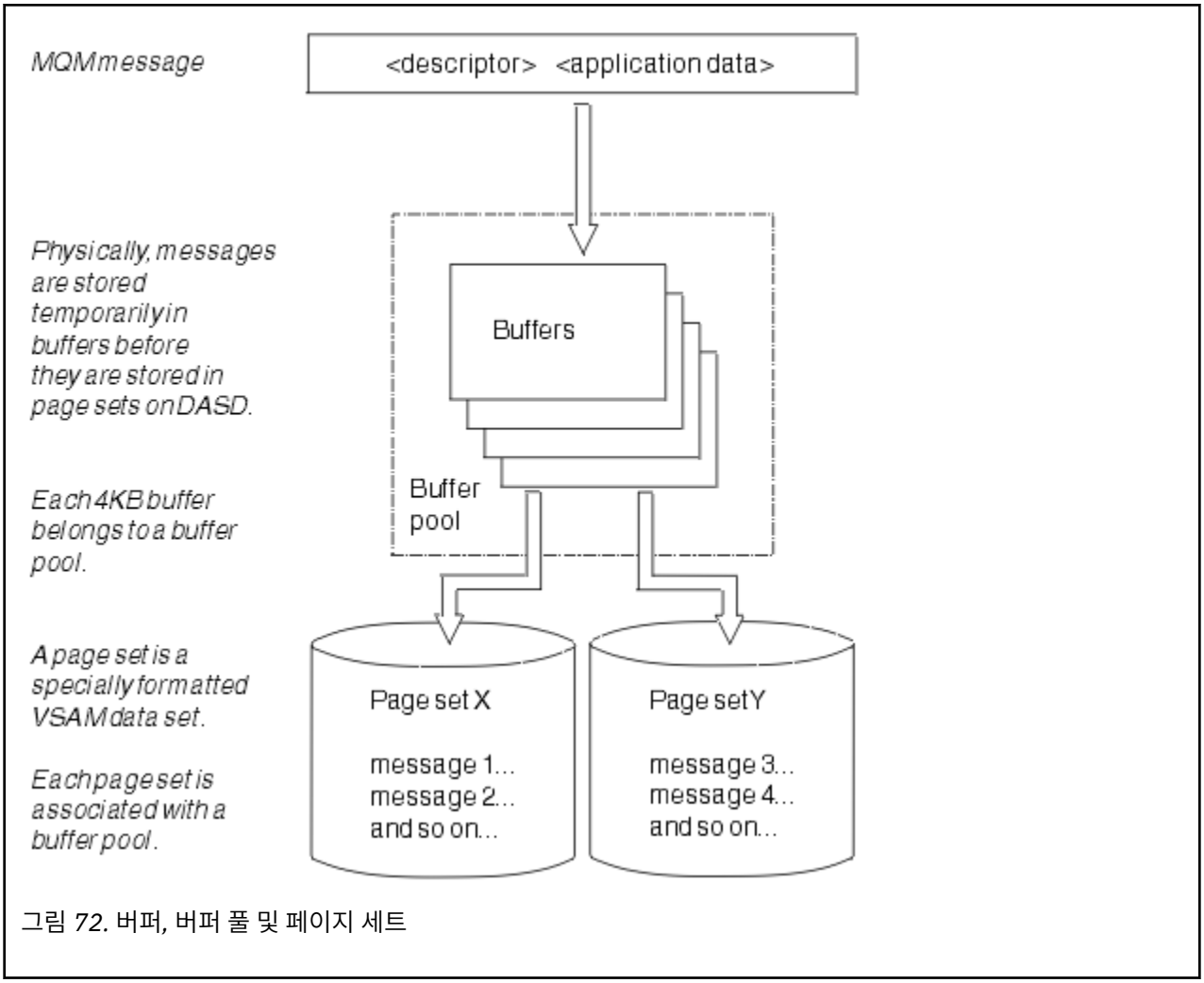


그림 72. 버퍼, 버퍼 풀 및 페이지 세트

ALTER BUFFPOOL 명령을 사용하여 버퍼 풀 크기 및 위치를 수정하는 명령을 동적으로 실행할 수 있습니다. **DEFINE PSID** 명령을 사용하여 페이지 세트를 동적으로 추가하거나 **DELETE PSID** 명령을 사용하여 삭제할 수 있습니다.

버퍼 풀이 너무 작으면 IBM MQ는 CSQP020E 메시지를 발행합니다. 그런 다음 관련된 버퍼 풀에 버퍼를 동적으로 더 추가할 수 있습니다(이를 수행하기 위해 다른 버퍼 풀에서 버퍼를 제거해야 할 수도 있음).

DEFINE BUFFPOOL 명령으로 풀의 버퍼 수를 지정하며, **ALTER BUFFPOOL** 명령으로 버퍼 풀의 크기를 동적으로 조정할 수 있습니다. **DISPLAY USAGE** 명령을 사용하여 버퍼 풀을 사용하는 페이지 세트를 표시하여 동적으로 풀의 현재 버퍼 수를 판별합니다.

성능상의 이유로 동일한 버퍼 풀에 메시지 및 오브젝트 정의를 넣지 마십시오. 오브젝트 정의를 보관하는 페이지 세트 0에 대해 독점적으로 하나의 버퍼 풀(번호 0)을 사용하십시오. 마찬가지로 단기간 메시지와 장기간 메시지를 다른 버퍼 풀, 결과적으로 다른 페이지 세트와 다른 큐에 보관하십시오.

새 버퍼 풀을 작성하기 위해 재시작한 후에는 **DEFINE BUFFPOOL** 명령을 사용할 수 없습니다. 그 대신에 **DEFINE PSID** 명령이 DSN 키워드를 사용하는 경우에 이는 현재 정의되지 않은 버퍼 풀을 명시적으로 식별할 수 있습니다. 해당 새 버퍼 풀이 작성됩니다.

z/OS IBM MQ for z/OS의 스토리지 관리에 대한 자세한 정보를 제공하는 위치

IBM MQ for z/OS에 대한 스토리지 관리의 추가 정보를 찾으려면 이 토픽을 참조로 사용하십시오.

다음 소스에서 이 절의 토픽에 대한 자세한 정보를 찾을 수 있습니다.

표 21. 스토리지 관리에 대한 자세한 정보를 제공하는 위치	
주제	**찾을 위치**
필요한 스토리지 크기	z/OS에서 스토리지 및 성능 요구사항 계획
페이지 세트 및 버퍼 풀을 작성할 크기	페이지 세트 및 버퍼 풀 계획
페이지 세트 관리	페이지 세트 관리
MQSC 명령	MQSC 명령

z/OS IBM MQ for z/OS에서 로깅

IBM MQ는 발생하는 데이터 변경사항 및 중요한 이벤트의 로그를 유지보수합니다. 이러한 로그는 필요한 경우 데이터를 이전 상태로 복구하는 데 사용할 수 있습니다.

부트스트랩 데이터 세트(BSDS)는 로그를 포함하는 데이터 세트에 대한 정보를 저장합니다.

로그는 통계, 추적 또는 성능 평가에 대한 정보를 포함하지 않습니다. IBM MQ가 수집하는 통계 및 모니터링 정보에 대한 추가적인 자세한 내용은 모니터링 및 통계를 참조하십시오.

로깅에 대한 자세한 정보는 다음 토픽을 참조하십시오.

- 206 페이지의 『IBM MQ for z/OS의 로그 파일』
- 210 페이지의 『로그가 구성되는 방식』
- 210 페이지의 『IBM MQ for z/OS 로그 기록 방법』
- 213 페이지의 『대형 로그 상대 바이트 주소』
- 214 페이지의 『부트스트랩 데이터 세트』

관련 태스크

로깅 환경 계획

시스템 매개변수 모듈을 사용하여 로그 설정

z/OS z/OS 관리

관련 참조

z/OS IBM MQ for z/OS에 대한 메시지

z/OS IBM MQ for z/OS의 로그 파일

로그 파일은 트랜잭션 복구에 필요한 정보를 포함합니다. 장기간 로그 데이터를 보존할 수 있도록 활성 로그 파일을 아카이브할 수 있습니다.

로그 파일의 개념

IBM MQ는 모든 중요한 이벤트가 발생하면 활성 로그에 기록합니다. 로그는 다음을 복구하는 데 필요한 정보를 포함합니다.

- 지속 메시지
- IBM MQ 오브젝트(예: 큐)
- IBM MQ 큐 관리자

활성 로그는 주기적으로 사용되는 데이터 세트(최대 310) 의 콜렉션을 포함합니다.

활성 로그를 채울 때 아카이브 데이터 세트에서 복사가 수행되도록 로크 아카이브를 사용할 수 있습니다. 아카이브를 사용하면 확장된 기간 동안 로그 데이터를 보존할 수 있습니다. 아카이브를 사용하지 않는 경우 로그 줄이

바뀌고 이전 데이터는 덮어쓰여집니다. 페이지 세트를 복구하거나 CF 구조에서 데이터를 복구하려면 페이지 세트 또는 구조의 백업을 취할 때 로그 데이터가 필요합니다. 아카이브 로그는 디스크 또는 테이프에서 작성될 수 있습니다.

아카이브

활성 로그의 크기는 고정되어 있으므로 IBM MQ는 정기적으로 아카이브 로그에 각 로그 데이터 세트의 컨텐츠를 복사합니다. 이는 일반적으로 직접 액세스 스토리지 디바이스(DASD) 또는 자기 테이프의 데이터 세트입니다. 서브시스템 또는 트랜잭션 실패가 발생한 경우 IBM MQ는 활성 로그와 필요한 경우 복구를 위해 아카이브 로그를 사용합니다.

아카이브 로그는 최대 순차 데이터 세트 1000개를 포함할 수 있습니다. z/OS 통합 카탈로그 기능(ICF)을 사용하여 각 데이터 세트를 카탈로그화할 수 있습니다.

아카이브는 IBM MQ 복구의 핵심 컴포넌트입니다. 복구 단위가 장기 실행 항목인 경우 해당 복구 단위 내 로그 데이터는 아카이브 로그에서 찾을 수 있습니다. 이 경우 복구에는 아카이브 로그의 데이터가 필요합니다. 그러나 아카이브가 사용 불가능하면 새 로그 레코드 내 활성 로그는 줄이 바뀌고 이전 로그 레코드를 겹쳐씁니다. 즉, IBM MQ에서는 복구 단위를 백아웃할 수 없으며, 메시지는 유실될 수 있습니다. 그런 다음 큐 관리자가 비정상적으로 종료됩니다.

따라서 프로덕션 환경에서는 **아카이브를 전환하지 마십시오**. 이렇게 하면 시스템 또는 트랜잭션 실패 이후에 데이터 손실 위험이 있습니다. 테스트 환경에서 실행하는 경우에만 아카이브 전환을 고려할 수 있습니다. 이를 수행해야 하는 경우에는 CSQ6LOGP 사용에서 설명한 CSQ6LOGP 매크로를 사용하십시오.

계획되지 않은 장기 실행 작업 단위에서 문제점을 방지하려면 활성 로그 오프로드 처리 중에 장기 실행 작업 단위가 발견된 경우 IBM MQ는 메시지(CSQJ160I 또는 CSQJ161I)를 발행합니다.

이중 로깅

이중 로깅의 경우 각 로그 레코드는 재시작 중에 데이터 손실 문제점 가능성을 최소화하도록 서로 다른 두 개의 활성 로그 데이터 세트에 기록됩니다.

단일 로깅 또는 이중 로깅을 사용하여 실행하도록 IBM MQ 를 구성할 수 있습니다. 단일 로깅의 경우 로그 레코드는 활성 로그 데이터 세트에 한 번 기록됩니다. 각 활성 로그 데이터 세트는 단일 익스텐트 VSAM 선형 데이터 세트(LDS)입니다. 이중 로깅에서 각 로그 레코드는 두 개의 서로 다른 활성 로그 데이터 세트에 기록됩니다. 이중 로깅은 재시작 중 데이터 손실 문제점 가능성을 최소화합니다.

로그 전환

로그 전환으로 인해 일부 작업 단위의 로그 레코드가 로그 아래에 추가로 기록됩니다. 그러면 장기 실행 또는 장기 실행 인다우트(in-doubt) 작업 단위에 대해 큐 관리자 재시작 또는 백아웃에서 읽어야 하는 로그 데이터 크기가 줄어듭니다.

작업 단위가 길다고 간주되면 각 로그 레코드의 표현이 로그 아래에 기록됩니다. 이 기법을 전환이라고 합니다. 전체 작업 단위가 처리되면 작업 단위는 전환된 상태입니다. 전환된 작업 단위와 관련된 모든 백아웃 또는 재시작 활동은 원래 작업 단위 로그 레코드를 사용하는 대신 전환된 로그 레코드를 사용할 수 있습니다.

장기 실행 작업 단위 발견은 체크포인트 프로세스의 기능입니다. 체크포인트 시점에서 전환해야 하는지 여부를 설정하도록 각 활성 작업 단위가 검사됩니다. 작성 또는 마지막 전환된 이후로 두 개의 사전 체크포인트를 통해 작업 단위가 전달되면 작업 단위는 전환에 적당합니다. 즉, 단일 작업 단위는 두 번 이상 전환될 수 있음을 의미합니다. 이를 다중 전환된 작업 단위라고 합니다.

작업 단위는 3개의 체크포인트마다 전환됩니다. 그러나 체크포인트는 로그 스위치(또는 LOGLOAD가 초과되게 만드는 로그 레코드의 쓰기)에 대해 비동기로 수행됩니다.

한 번에 하나의 체크포인트만 발생하므로 체크포인트가 완료되기 전에 다중 로그 스위치가 있을 수 있습니다.

이는 활성 로그가 충분하지 않거나 너무 적은 경우 모든 로그를 채우기 전에 대형 작업 단위의 처리 지연이 완료되지 않을 수 있음을 의미합니다.

처리 지연을 완료할 수 없는 경우 메시지 CSQR027I가 발생합니다.

로그 아카이브를 끄면 처리 지연이 실패한 작업 단위를 백아웃하려는 시도가 있는 경우 이유가 00D1032A인 ABEND 5C6이 발생합니다. 이 문제점을 피하려면 OFFLOAD=YES를 사용해야 합니다.

로그 전환은 항상 활성이며 로그 아카이브의 사용 여부에 상관없이 실행됩니다.

참고: 작업 단위의 모든 로그 레코드가 전환되어도 각 레코드의 전체 컨텐츠는 전환되지 않으며, 백아웃에 필요한 부분만 전환됩니다. 즉, 기록된 로그 데이터 크기는 최소로 보관되며 페이지 세트 장애가 발생한 경우 전환된 레코드는 사용할 수 없습니다. 장기 실행 작업 단위는 네 개 이상의 큐 관리자 체크포인트에 대해 실행 중인 항목입니다.

로그 전환에 대한 자세한 정보는 로그 관리를 참조하십시오.

로그 압축

로그 데이터 세트에서 기록하고 읽으면 로그 레코드를 압축하고 압축을 풀도록 IBM MQ for z/OS를 구성할 수 있습니다.

로그 압축은 개인 큐의 지속 메시지에 대한 로그에 기록된 데이터의 크기를 줄이도록 사용할 수 있습니다. 아카이브되는 압축 크기는 메시지에 포함된 데이터 유형에 따라 달라집니다. 예를 들어 실행 길이 인코딩(RLE)은 구조화된 또는 레코드 중심 데이터에서 효과적으로 좋은 결과를 제공할 수 있는 바이트의 반복되는 인스턴스를 압축하여 작동합니다.

주의: 공유 큐에 넣는 지속 메시지는 로그 압축의 대상이 아닙니다.

시스템 관리 기능 115(SMF) 레코드의 로그 관리자 섹션 내 필드를 사용하여 데이터 압축이 아카이브되는 방법을 모니터할 수 있습니다. SMF에 대한 자세한 정보는 시스템 관리 기능 사용 및 회계 및 통계 메시지를 참조하십시오.

로그 압축은 시스템의 프로세서 활용도를 높입니다. 로그 데이터 세트를 보유하는 데 필요한 디스크 스토리지에서 제한되거나 큐 관리자의 처리량이 로그 데이터 세트에 기록되는 입출력 대역폭으로 제한되는 경우에만 압축 사용을 고려해야 합니다. 공유 큐를 사용하는 경우 입출력 대역폭 제한조건은 큐 공유 그룹에 추가 큐 관리자를 추가하고 추가 큐 관리자에서 워크로드를 분배하여 완화할 수 있습니다.

로그 압축 옵션은 큐 관리자를 중지하고 재시작할 필요 없이 필요하면 사용하거나 사용하지 않을 수 있습니다. 큐 관리자는 현재 로그 압축 설정에 상관없이 압축된 로그 레코드를 읽을 수 있습니다.

큐 관리자는 로그 압축에 대해 세 가지 설정을 지원합니다.

NONE
로그 데이터 압축이 사용되지 않습니다. 이는 기본값입니다.

RLE
실행 길이 인코딩을 사용하여 로그 데이터 압축이 수행됩니다(RLE).

ANY
가장 높은 단계로 로그 레코드를 압축하는 압축 알고리즘을 큐 관리자가 선택할 수 있습니다. 이 옵션을 사용하면 RLE 압축이 수행됩니다.

다음 중 하나를 사용하여 로그 레코드의 압축을 제어할 수 있습니다.

- MQSC의 SET 및 DISPLAY LOG 명령은 SET LOG 및 DISPLAY LOG 참조
- PCF 인터페이스의 로그 설정 및 조회 로그 기능은 로그 설정 및 로그 조회 참조
- 시스템 매개변수 모듈의 CSQ6LOGP 매크로는 CSQ6LOGP 사용 참조

또한 로그 인쇄 유틸리티 CSQ1LOGP에는 압축된 로그 레코드 확장에 대한 지원이 있습니다.

로그 데이터

로그는 최대 1800만 x 백만 제곱($1.8*10^{19}$)바이트를 포함할 수 있습니다. 각 바이트는 로그 시작 위치에서 오프셋으로 주소를 지정할 수 있으며 해당 오프셋은 상대적 바이트 주소(RBA)라고 합니다.

RBA는 사용 중인 로그 RBA가 6바이트인지 아니면 8바이트인지에 따라 2^{48}바이트 또는 2^{64} 바이트의 총 주소 지정 범위를 제공하는 6바이트 또는 8바이트 필드에서 참조됩니다.

그러나 IBM MQ에서 사용된 범위가 F00000000000(6바이트 RBA를 사용 중인 경우) 또는 FFFF800000000000(8바이트 로그 RBA를 사용 중인 경우) 이상임을 감지하면 CSQI045, CSQI046, CSQI047 및 CSQJ032 메시지가 발행되어 로그 RBA를 재설정하도록 경고합니다.

RBA 값이 FFF800000000(6바이트 로그 RBA가 사용 중인 경우) 또는 FFFFFFC000000000(8바이트 로그 RBA가 사용 중인 경우)에 근접하는 경우 큐 관리자가 종료되고 이유 코드 00D10257이 표시됩니다.

사용된 로그 범위에 대한 경고 메시지가 발행되면 큐 관리자 가동 중지를 계획해야 합니다. 이 기간 동안 큐 관리자가 8바이트 로그 RBA를 사용하도록 변환하거나 로그를 재설정할 수 있습니다. 로그를 재설정하는 프로시저는 큐 관리자의 로그 재설정에 설명되어 있습니다.

큐 관리자가 6바이트 로그 RBA를 사용 중인 경우 큐 관리자의 로그를 재설정하지 않고 대형 로그 상대 바이트 주소 구현에 설명된 프로시저에 따라 8바이트 로그 RBA를 사용하도록 큐 관리자를 변환하십시오.

로그는 로그 레코드로 구성되며, 각각은, 단일 단위로 처리되는 로그 데이터 세트에 해당합니다. 로그 레코드는 헤더의 첫 번째 바이트에 해당하는 RBA 또는 해당 LRSN(log record sequence number)로 식별됩니다. RBA 또는 LRSN은 로그의 특정 지점에서 시작되는 레코드를 고유하게 식별합니다.

로그 지점을 식별하기 위해 RBA 또는 LRSN 중 어느 것을 사용할 것인지 여부는 큐 공유 그룹 사용 여부에 따라 달라집니다. 큐 공유 환경에서는 상대적 바이트 주소를 사용하여 로그 지점을 고유하게 식별할 수 없습니다. 다중 큐 관리자가 동시에 동일한 큐를 업데이트할 수 있으며 각각은 고유한 로그를 포함합니다. 이를 해결하기 위해 LRSN(log record sequence number)은 시간소인 값에서 파생되며 로그 내 로그 레코드의 물리적 배치를 반드시 표현하지 않아도 됩니다.

각 로그 레코드에는 해당 유형, 레코드를 작성하는 IBM MQ 서브컴포넌트 및 복구 단위 레코드의 경우, 복구 단위 ID를 제공하는 헤더가 있습니다.

다음 표제 아래에서 설명하는 네 가지 유형의 로그 레코드가 있습니다.

- 복구 단위 로그 레코드
- 체크포인트 레코드
- 페이지 세트 제어 레코드
- CF 구조 백업 레코드

복구 단위 로그 레코드

대부분의 로그 레코드에서는 IBM MQ 큐에 대한 변경사항을 설명합니다. 이러한 모든 변경사항은 복구 단위에서 수행됩니다.

IBM MQ는 재시작 시간을 줄이고 시스템 사용 가능성을 향상시키도록 로그 레코드 보충 및 실행 취소/다시 실행과 관련된 특수 로깅 기술을 사용합니다.

이에 대한 한 가지 효과는 재시작 시간이 제한된다는 점입니다. 재시작 중에 실패가 발생하여 큐 관리자를 두 번째로 재시작해야 하는 경우 첫 번째 재시작 실패 지점에서 완료된 모든 복구 활동은 두 번째 재시작 중에 다시 적용되지 않아도 됩니다. 즉, 연속된 재시작은 완료하는 데 더 오랜 시간이 걸리지 않습니다.

체크포인트 레코드

재시작 시간을 줄이기 위해 IBM MQ는 정상적인 조작 중에 정기적인 체크포인트를 정합니다. 다음과 같이 수행됩니다.

- 사전 정의된 로그 레코드 수가 기록됩니다. 이 수는 CSQ6SYSP 사용에서 설명한 대로, 시스템 매개변수 매크로 CSQ6SYSP라고 하는 체크포인트 빈도 피연산자에 의해 정의됩니다.
- 성공적인 재시작의 종료 시
- 정상 종료 시
- IBM MQ는 순환에서 다음 활성 로그 데이터 세트로 전환합니다.

체크포인트가 지정된 시점에 IBM MQ는 DISPLAY CONN 명령(DISPLAY CONN에 설명됨)을 내부적으로 실행하므로 현재 인다우트 상태인 연결 목록을 z/OS 콘솔 로그에 씁니다.

페이지 세트 제어 레코드

이러한 레코드는 각 체크포인트에 IBM MQ 큐 관리자에 알려진 페이지 세트 및 버퍼 풀을 등록하고 체크포인트 시점에 페이지 세트의 매체 복원을 수행하는 데 필요한 로그 범위에 대한 레코드 정보를 등록합니다.

페이지 세트 및 버퍼 풀에 대한 특정 동적 변경도 페이지 세트 제어 레코드로 기록되므로 변경은 복원 가능하며 다음 큐 관리자 재시작 시 자동으로 다시 인스턴스화할 수 있습니다.

CF 구조 백업 레코드

이러한 레코드는 BACKUP CFSTRUCT 명령에 대한 응답으로 커플링 기능 목록 구조에서 읽은 데이터를 보유합니다. 커플링 기능 구조에서 장애가 발생한 경우와는 다르게, 이러한 레코드는 RECOVER CFSTRUCT 명령이 장애 지점에 대해 커플링 기능 구조의 매체 복구를 수행하는 데 복구 단위 레코드와 함께 사용됩니다.

관련 태스크

대형 로그 상대 바이트 주소 구현

z/OS 로그가 구성되는 방식

이 토픽을 사용하여 로그 레코드를 설명하는 데 사용되는 용어를 이해하십시오.

각 활성 로그 데이터 세트는 VSAM 선형 데이터 세트(LDS)여야 합니다. 활성 로그 데이터 세트에 기록된 물리적 출력 단위는 4KB 제어 간격(CI)입니다. 각 CI는 하나의 VSAM 레코드를 포함합니다.

물리적 로그 레코드와 논리적 로그 레코드

하나의 VSAM CI는 물리적 레코드입니다. 특정 시간에 기록되는 정보는 CI에서 사용 가능한 공간과는 별개로 길이가 달라지는 논리 레코드를 구성합니다. 따라서 하나의 물리적 레코드는 다음을 포함할 수 있습니다.

- 여러 논리 레코드
- 하나 이상의 논리 레코드와 다른 논리 레코드의 일부
- 하나의 논리 레코드만에서 일부

로그 레코드 용어는 이를 저장하는 데 필요한 물리적 레코드 수에 상관없이 논리 레코드를 참조합니다.

z/OS IBM MQ for z/OS 로그 기록 방법

이 토픽을 사용하여 IBM MQ가 로그 파일 레코드를 처리하는 방법을 이해하십시오.

IBM MQ는 각 로그 레코드를 활성 로그라 하는 DASD 데이터 세트에 작성합니다. 활성 로그가 가득차면 IBM MQ가 컨텐츠를 아카이브 로그라 하는 테이프 데이터 세트 또는 DSAD로 복사합니다. 이 프로세스는 오프-로딩이라고 합니다.

211 페이지의 그림 73에서는 로깅 프로세스를 보여줍니다. 일반적으로 로그 레코드는 다음 주기로 진행됩니다.

1. IBM MQ는 데이터에 대한 변경사항 및 중요한 이벤트를 복구 로그 레코드에 기록합니다.
2. IBM MQ는 복구 로그 레코드를 처리하고 필요한 경우 이들을 세그먼트로 구분합니다.
3. 로그 레코드는 VSAM 제어 간격(CI)로 형식화된 출력 로그 버퍼에서 순차적으로 배치됩니다. 각 로그 레코드는 0에서 2^{64}-1 사이의 범위에서 상대 바이트 주소로 식별됩니다.
4. CI는 순차적으로 사용되며 재순환되는 사전 정의된 DASD 활성 로그 데이터의 세트에 기록됩니다.
5. 아카이브가 활성 상태인 경우 각 활성 로그 데이터 세트가 가득 차면 해당 컨텐츠는 새 아카이브 로그 데이터 세트에 자동으로 오프로드됩니다.

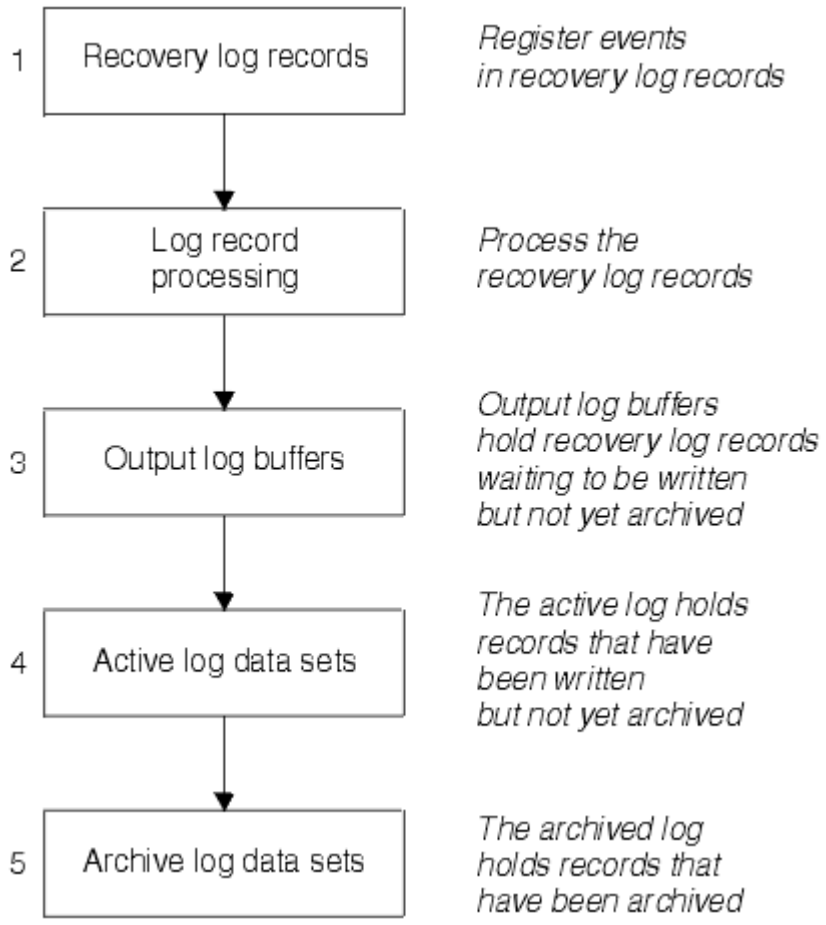


그림 73. 로깅 프로세스

활성 로그가 기록되는 시점

스토리지 내 로그 버퍼는 다음 상황이 발생할 때마다 활성 로그 데이터 세트에 기록됩니다.

- 로그 버퍼가 가득 찬 경우.
- 쓰기 임계값에 도달한 경우(CSQ6LOGP 매크로에서 지정된 대로).
- 특정 이벤트가 발생한 경우(예: 커미트 지점) 또는 IBM MQ BACKUP CFSTRUCT 명령이 실행된 경우.

큐 관리자가 초기화되면 BSDS에서 이름이 지정된 활성 로그 데이터 세트는 큐 관리자에 의해 독점적 사용을 위해 동적으로 할당되며 큐 관리자가 종료될 때까지 IBM MQ에 독점적으로 할당된 상태입니다.

동적으로 로그 데이터 세트 추가

큐 관리자가 실행 중인 동안 새 활성 로그 데이터 세트를 동적으로 정의할 수 있습니다. 이 기능은 아카이브에서 일시적인 문제점으로 인해 활성 로그를 오프로드할 수 없는 경우 정지된 큐 관리자의 문제점을 완화시킵니다. 자세한 정보는 DEFINE LOG 명령을 참조하십시오.

참고: 활성 로그를 재정의하거나 제거하려면 큐 관리자를 종료하고 재시작해야 합니다.

IBM MQ 및 스토리지 관리 서브시스템

IBM MQ 매개변수를 사용하면 IBM MQ 아카이브 로그 데이터 세트를 동적으로 할당할 때 스토리지 관리 서브시스템 (MVS/DFP SMS) 스토리지 클래스를 지정할 수 있습니다. IBM MQ는 로그 데이터 세트의 아카이브를 시작하지만, 사용자는 SMS를 사용하여 아카이브 데이터 세트의 할당을 수행할 수 있습니다.

관련 참조

212 페이지의 『IBM MQ for z/OS 아카이브 로그가 기록되는 시점』
이 토픽을 사용하여 활성 로그를 아카이브 로그에 복사하는 프로세스 및 프로세스 수행 시점을 이해하십시오.

z/OS *IBM MQ for z/OS* 아카이브 로그가 기록되는 시점

이 토픽을 사용하여 활성 로그를 아카이브 로그에 복사하는 프로세스 및 프로세스 수행 시점을 이해하십시오.

활성 로그를 아카이브 로그에 복사하는 프로세스는 오프-로딩이라고 합니다. 다른 로깅 이벤트에 대한 오프-로딩의 관계는 212 페이지의 그림 74에서 체계적으로 표시됩니다.

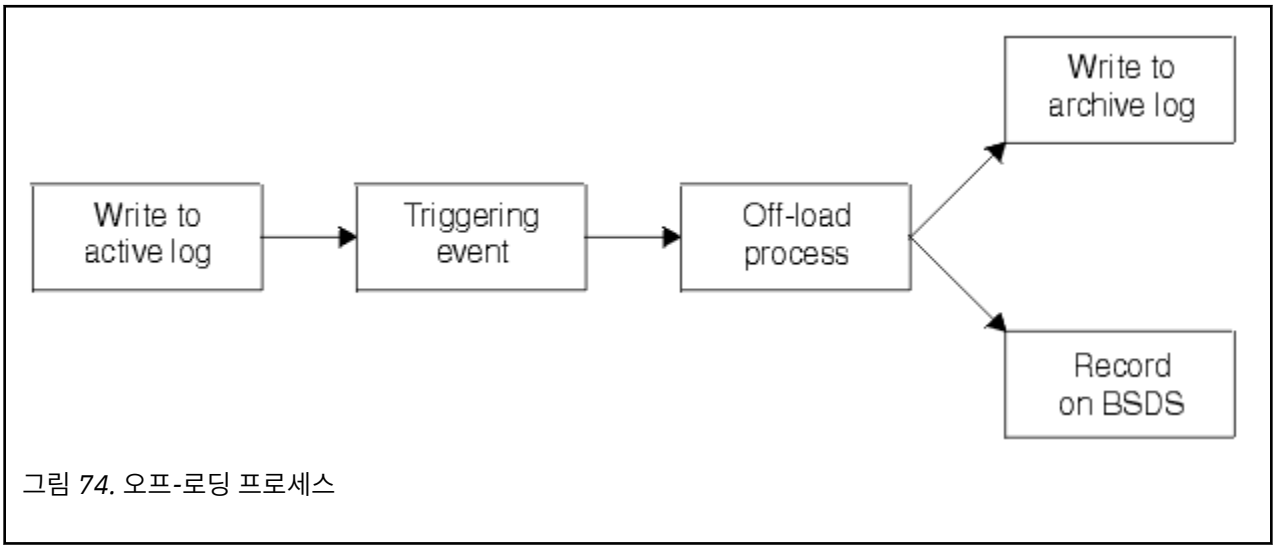


그림 *74.* 오프-로딩 프로세스

오프-로딩 프로세스 트리거

아카이브 로그에 대한 활성 로그의 오프로드 프로세스는 여러 이벤트로 트리거될 수 있습니다. 예를 들면, 다음과 같습니다.

- 활성 로그 데이터 세트를 채웁니다.
- MQSC ARCHIVE LOG 명령을 사용합니다.
- 활성 로그 데이터 세트에 기록하는 중 오류가 발생합니다.

데이터 세트는 실패 지점 이전에 잘리며, 기록되지 않은 레코드는 새 데이터 세트의 첫 번째 레코드가 됩니다. 오프-로딩은 정상적인 전체 로그 데이터 세트로 잘린 데이터 세트에서 트리거됩니다. 이중 활성 로그가 있는 경우 두 사본이 동기화되도록 두 사본이 잘립니다.

마지막 사용 가능한 활성 로그가 5% 채워지고 이후에 5% 증분 시 사용 중인 로그 용량의 백분율을 언급하는 메시지 CSQJ110E가 발행됩니다. 모든 활성 로그가 가득차면 IBM MQ는 오프-로딩이 수행될 때까지 처리를 중지하며 이 메시지를 발행합니다.

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

오프로드 프로세스

모든 활성 로그가 가득차면 IBM MQ는 오프-로딩 프로세스를 실행하고 오프-로딩 프로세스를 완료할 때까지 처리를 정지합니다. 활성 로그가 가득찬 경우 오프로드 처리에 실패하면 IBM MQ가 이상종료됩니다.

활성 로그가 오프로드될 준비가 된 경우 DASD 단위를 준비하거나 테이프를 마운트하도록 z/OS 콘솔 연산자에 요청이 전송됩니다. ARCWTOR 로깅 옵션의 값은(추가적인 정보는 CSQ6ARVP 사용 참조) 요청의 수신 여부를

판별합니다. 오프-로딩을 사용 중인 경우, ARCWTOR=YES를 지정하십시오. 값이 YES인 경우 요청에서는 WTOR(메시지 번호 CSQJ008E)이 앞에 오며 할당할 활성 로그 데이터 세트를 준비하도록 연산자에 지시합니다.

연산자는 이 메시지에 즉시 응답하지 않아도 됩니다. 그러나 응답이 지연되면 오프로드 프로세스가 지연됩니다. 연산자에서 응답이 너무 지연되어 IBM MQ에서 활성 로그가 부족해지지 않는 한, IBM MQ 성능에는 영향을 주지 않습니다.

연산자는 오프로드 프로세스를 취소하여 응답할 수 있습니다. 이 경우 이중 아카이브 데이터 세트의 첫 번째 사본에 대한 할당이면 오프로드 프로세스는 다음 활성 로그 데이터 세트가 가득 찰 때까지 지연됩니다. 두 번째 사본에 대한 할당이면 아카이브 프로세스는 이 데이터 세트에 대해서만 단일 복사 모드로 전환됩니다.

오프-로딩 중 발생한 인터럽트 및 오류

큐 관리자를 중지하는 요청은 오프로드 처리가 완료될 때까지 적용되지 않습니다. 오프로딩이 진행 중인 동안에 IBM MQ가 실패하면 큐 관리자가 재시작될 때 다시 오프-로딩이 시작됩니다.

오프로드 처리 중 발행된 메시지

오프로드된 메시지는 IBM MQ 및 오프로드 프로세스에 의해 z/OS 콘솔로 전송됩니다. 이러한 메시지를 사용하여 다양한 로그 데이터 세트에서 RBA 범위를 찾을 수 있습니다.

z/OS 대형 로그 상대 바이트 주소

이 기능은 로그를 재설정해야 하는 시점까지의 기간을 늘려 큐 관리자의 가용성을 향상시킵니다.

큐 관리자를 다시 시작할 때 지속 메시지를 사용할 수 있도록 복구 데이터를 로그에 씁니다. 로그 상대 바이트 주소(로그 RBA)라는 용어는 로그의 시작부터 오프셋으로 데이터의 위치를 나타내는 데 사용합니다.

IBM MQ 8.0 전에는 6바이트 로그 RBA는 최대 256TB의 데이터를 처리할 수 있었습니다. 이 크기의 로그 레코드를 작성하기 전에 큐 관리자의 로그 재설정에 설명된 프로시저에 따라 큐 관리자의 로그를 재설정해야 합니다.

큐 관리자 로그를 재설정하는 프로세스는 간단하지 않으며 프로세스의 일부로 페이지 세트를 재설정해야 하므로 오랜 시간 동안 가동을 중지해야 할 수 있습니다. 사용률이 높은 큐 관리자의 경우 일반적으로 이 조작은 1년에 한 번 수행할 수 있습니다.

IBM MQ 8.0부터는 로그 RBA의 길이가 8바이트가 될 수 있으므로 큐 관리자는 이제 로그 RBA를 재설정하기 전에 64,000배 이상 많은 데이터(16엑사바이트)를 처리할 수 있습니다. 대형 로그 RBA를 사용함에 따라 작성되는 로그 데이터의 크기가 몇 바이트 증가하게 됩니다.

이 기능을 사용할 수 있는 경우

V 9.3.0 IBM MQ 9.3.0 이상에서 작성된 큐 관리자에는 이미 이 기능이 사용으로 설정되어 있습니다.

현재 로그 RBA가 로그 RBA 범위의 끝에 접근하고 있는 경우 큐 관리자의 로그를 재설정하지 않고 큐 관리자가 8바이트 로그 RBA를 사용하도록 변환하는 것을 고려하십시오. 큐 관리자가 8바이트 로그 RBA를 사용하도록 변환하려면 로그를 재설정하는 경우보다 가동 중단 시간이 짧아지므로 로그를 재설정해야 하는 시점까지의 기간을 늘릴 수 있습니다.

큐 관리자 초기화 중에 발행된 CSQJ034I 메시지는 큐 관리자에 대해 구성된 로그 RBA 범위의 끝을 표시하며 사용 중인 로그 RBA가 6바이트인지 아니면 8바이트인지를 판별하는 데 사용할 수 있습니다.

이 기능을 사용하는 방법

8바이트 로그 RBA는 버전 2 형식 BSDS가 있는 큐 관리자를 시작하여 사용합니다. 요약하자면 이 작업은 다음을 통해 수행됩니다.

1. 큐 공유 그룹의 모든 큐 관리자가 8바이트 로그 RBA 사용 요구사항을 충족하는지 확인
2. 큐 관리자를 완전히 시스템 종료
3. BSDS 변환 유틸리티를 실행하여 버전 2 형식의 BSDS 사본 작성
4. 변환된 BSDS로 큐 관리자 재시작

8바이트 로그 RBA를 사용하도록 큐 관리자를 변환하고 나면 6바이트 로그 RBA를 사용하도록 되돌릴 수 없습니다.

8바이트 로그 RBA를 사용하는 방법에 대한 자세한 프로시저는 대형 로그 상대 바이트 주소 구현을 참조하십시오.

관련 태스크

주소 지정 가능 최대 로그 범위 증가 계획

관련 참조

BSDS 변환 유틸리티(CSQJUCNV)

z/OS 부트스트랩 데이터 세트

부트스트랩 데이터 세트는 로그 데이터 세트 및 로그 레코드를 참조하는 메커니즘으로 IBM MQ에 필요합니다. 이 정보는 정상적인 처리 및 재시작 복구 중에 필요합니다.

부트스트랩 데이터 세트의 용도

부트스트랩 데이터 세트(BSDS)는 IBM MQ에 필요한 정보를 보유하는 VSAM 키 순차 데이터 세트(KSDS)입니다. 여기에는 다음이 포함됩니다.

- IBM MQ에 알려진 모든 활성 및 아카이브 로그 데이터 세트의 재고. IBM MQ는 이 재고를 사용하여 다음을 수행합니다.
  - 활성 및 아카이브 로그 데이터 세트 추적
  - 정상적인 처리 중에 로그 읽기 요청을 만족할 수 있도록 로그 레코드 찾기
  - 재시작 처리를 처리할 수 있도록 로그 레코드 찾기

  IBM MQ는 활성 로그 데이터 세트를 재사용하거나 아카이브 로그 데이터 세트를 정의할 때마다 재고에 정보를 저장합니다. 활성 로그의 경우 재고는 가득 찬 항목과 재사용 가능한 항목을 표시합니다. 재고는 해당 데이터 세트에 보유된 로그의 각 부분에 대한 상대 바이트 주소(RBA)를 보유합니다.
- 모든 최근 IBM MQ 활동의 랩어라운드 자원 명세. 이는 큐 관리자를 재시작해야 하는 경우 필요합니다.

BSDS는 큐 관리자에 오류가 있고 이를 재시작해야 하는 경우 필요합니다. IBM MQ에 **반드시** BSDS가 있어야 합니다. 재시작 중에 문제점의 발생 가능성을 최소화하도록 각각 동일한 정보를 기록하는 이중 BSDS를 포함한 IBM MQ를 구성할 수 있습니다. 이중 BSDS 사용은 이중 모드에서의 실행이라고도 합니다. 가능한 경우 별도의 볼륨에 사본을 배치하십시오. 이는 볼륨이 파손되거나 손상된 경우 이들이 유실된 위험을 줄입니다. DASD에 대한 이중 쓰기 대신, 이중 BSDS를 사용하십시오.

BSDS는 IBM MQ가 사용자 정의되는 경우 설정되며 변경 로그 재고 유틸리티(CSQJU003)를 사용하여 재고를 관리할 수 있습니다. 이 유틸리티에 대한 자세한 정보는 관리IBM MQ for z/OS의 내용을 참조하십시오. 이는 큐 관리자 시동 프로시저에서 DD 문으로 참조됩니다.

정상적으로 IBM MQ는 BSDS의 중복 사본을 보관합니다. I/O 오류가 발생한 경우 실패한 사본을 할당 해제하고 단일 BSDS를 계속합니다. 이중 모드 조작을 복원할 수 있습니다. 이는 관리IBM MQ for z/OS에서 설명됩니다.

활성 로그는 IBM MQ가 설치된 경우 BSDS에 처음 등록됩니다. 큐 관리자를 종료한 후 재시작하지 않으면 활성 로그를 바꿀 수 없습니다.

아카이브 로그 데이터 세트는 동적으로 할당됩니다. 하나가 할당되면 데이터 세트 이름이 BSDS에 등록됩니다. 아카이브가 추가되면 아카이브 로그 데이터 세트 목록이 펼쳐지고 사용자가 판별한 항목 수에 도달하면 줄이 바뀝니다. 최대 항목 수는 단일 아카이브 로깅의 경우 1000, 이중 로깅의 경우 2000입니다.

테이프 관리 시스템을 사용하여 아카이브 로그 데이터 세트를 삭제할 수 있습니다(IBM MQ는 자동화된 방법을 보유하지 않음). 따라서 아카이브 로그 데이터 세트를 시스템 관리자가 삭제한 후에 아카이브 로그 데이터 세트에 대한 정보는 BSDS에 있을 수 있습니다.

반대로 아카이브 로그 데이터 세트의 최대 수를 초과할 수 있으며 데이터 세트가 만기 날짜에 도달하기 전에 BSDS의 데이터가 삭제됩니다.

다음 MQSC 명령을 사용하여 로그 범위와 큐 관리자 복구 또는 다양한 매체 유형에 필요한 가장 이른 로그 RBA를 보유하는 활성 또는 아카이브 로그 데이터 세트의 이름을 판별할 수 있습니다.

```
DISPLAY USAGE TYPE(DATASET)
```

시스템 매개변수 모듈에서 할당 시 아카이브 로그 데이터 세트를 카탈로그화하도록 지정한 경우 BSDS는 나중 할당에 필요한 정보를 위해 통합 카탈로그 기능(ICF) 카탈로그를 가리킵니다. 그렇지 않으면 각 볼륨에 대한 BSDS 항목은 나중 할당에 필요한 볼륨 일련 번호 및 단위 정보를 등록합니다.

BSDS 버전

BSDS의 형식은 버전에 따라 다릅니다. BSDS의 버전이 증가되면 새 기능을 사용할 수 있습니다. 다음 BSDS 버전은 IBM MQ에서 지원됩니다.

버전 1
IBM MQ의 모든 릴리스에서 지원됩니다. 버전 1 BSDS는 6바이트 로그 RBA 값을 지원합니다.

버전 2
IBM MQ 8.0 이상에서 지원됩니다. 버전 2 BSDS에서는 8바이트 로그 RBA 값과 각 활성 로그 사본에 있는 최대 310개의 데이터 세트를 사용합니다.

V 9.3.0 IBM MQ 9.3.0 이상에서 작성된 큐 관리자에 대해 기본적으로 사용됩니다.

버전 3
IBM MQ 8.0 이상에서 지원됩니다. 어느 활성 로그 사본에든 32개 이상의 데이터 세트가 추가되면 BSDS가 자동으로 버전 2에서 버전 3으로 변환됩니다.

로그 맵 인쇄 유틸리티 (CSQJU004)를 실행하여 BSDS 버전을 판별할 수 있습니다. BSDS를 버전 1에서 버전 2로 변환하려면 BSDS 변환 유틸리티 (CSQJUCNV)를 실행하십시오.

6바이트 및 8바이트 로그 RBA에 대한 자세한 정보는 213 페이지의 『대형 로그 상대 바이트 주소』의 내용을 참조하십시오.

아카이브 로그 데이터 세트 및 BSDS 사본

새 아카이브 로그 데이터 세트를 작성할 때마다 BSDS의 사본도 작성됩니다. 아카이브 로그가 테이프에 있으면 BSDS는 첫 번째 출력 볼륨의 첫 번째 데이터 세트입니다. 아카이브 로그가 DASD에 있으면 BSDS는 별도의 데이터 세트입니다.

BSDS 사본 및 아카이브 로그의 데이터 세트 이름은 동일합니다. 단, 아카이브 로그 이름의 최하위 레벨 규정자가 A로 시작하고 BSDS 사본이 B로 시작하는 경우는 예외입니다. 예를 들어 다음과 같습니다.

아카이브 로그 이름
CSQ.ARCHLOG1.E00186.T2336229.*A*0000001

BSDS 사본 이름
CSQ.ARCHLOG1.E00186.T2336229.*B*0000001

BSDS를 복사하는 중 읽기 오류가 발생한 경우 메시지 CSQJ125E가 발행되고 새 아카이브 로그 데이터 세트에 대한 오프-로딩은 BSDS 사본 없이 계속됩니다.

z/OS z/OS 의 시스템 정의

IBM MQ for z/OS는 많은 기본 오브젝트 정의를 사용하고 해당 기본 오브젝트를 작성하도록 샘플 JCL을 제공합니다. 이 토픽을 사용하여 이러한 기본 오브젝트 및 샘플 JCL을 이해하십시오.

시스템 매개변수 설정

IBM MQ for z/OS에서는 시스템 매개변수 모듈이 IBM MQ가 해당 조작에서 사용하는 연결 환경, 추적, 아카이브 및 로깅을 제어합니다. 시스템 매개변수는 다음과 같이 세 가지 어셈블러 매크로에서 지정됩니다.

CSQ6SYSP
연결 설정 및 환경 추적을 포함한 시스템 매개변수입니다.

CSQ6LOGP
로깅 매개변수입니다.

CSQ6ARVP
로그 아카이브 매개변수입니다.

기본 매개변수 모듈은 IBM MQ for z/OS와 함께 제공됩니다. 사용하려는 값을 포함하지 않는 경우 IBM MQ에서 제공하는 샘플을 사용하여 사용자 고유의 매개변수 모듈을 작성할 수 있습니다. 샘플은 `thlqual.SCSQPROC(CSQ4ZPRM)`입니다.

큐 관리자가 실행되는 동안 일부 시스템 매개변수를 대체할 수 있습니다. MQSC 명령에서 SET SYSTEM, SET LOG및 SET ARCHIVE 명령을 참조하십시오.

정의에 대한 자세한 정보는 다음 주제를 참조하십시오.

- 216 페이지의 『IBM MQ for z/OS의 시스템 오브젝트 정의』
- 220 페이지의 『IBM MQ for z/OS에서 큐 관리자 성능 조정』
- 221 페이지의 『IBM MQ for z/OS에서 제공하는 샘플 정의』

관련 개념
샘플 초기화 입력 데이터 세트 사용자 정의
IBM MQ for z/OS 에서 MQSC및 PCF 명령을 실행할 수 있는 소스
관련 태스크
z/OS 관리
클러스터 구성
IBM MQ 모니터링

z/OS IBM MQ for z/OS의 시스템 오브젝트 정의

IBM MQ for z/OS에서는 발행/구독 애플리케이션, 클러스터 및 채널 제어에 대한 사전 정의된 추가 오브젝트와 기타 시스템 관리 기능이 필요합니다.

IBM MQ for z/OS에 필요한 시스템 오브젝트는 다음 범주로 구분할 수 있습니다.

- 발행/구독 오브젝트
- 시스템 기본 오브젝트
- 시스템 명령 오브젝트
- 시스템 관리 오브젝트
- 채널 큐
- 클러스터 큐
- 큐 공유 그룹 큐
- 스토리지 클래스
- 시스템 오브젝트 데드-레터 큐 정의
- 기존 전송 큐
- 내부 큐
- 220 페이지의 『채널 인증 큐』

발행/구독 오브젝트

IBM MQ for z/OS에서 발행/구독 애플리케이션을 사용하려면 먼저 정의해야 하는 여러 시스템 오브젝트가 있습니다. 이러한 오브젝트를 정의하는 데 도움을 주는 샘플 정의를 IBM MQ에서 제공합니다. 해당 샘플은 CSQ4INSG에 설명되어 있습니다.

발행/구독을 사용하려면 다음 오브젝트를 정의해야 합니다.

- 큐 관리자에서 각 보유된 발행의 사본을 보유하는 데 사용되는 로컬 큐 SYSTEM.RETAINED.PUB.QUEUE. 각 전체 토픽 이름은 이 큐에 저장된 최대 1개의 보유된 발행물을 포함할 수 있습니다. 애플리케이션이 많은 다른 토픽에서 보유된 발행물을 사용하거나 보유된 발행 메시지가 너무 많은 메시지인 경우 이 큐의 스토리지에 대

한 요구사항은 이에 대한 스토리지 요구사항이 너무 큰 경우 고유한 페이지 세트를 지정하는 등 신중하게 계획되어야 합니다. 성능을 향상시키려면 제공된 샘플 큐 정의에 표시된 대로, 색인 유형이 MSGID로 이 큐를 정의해야 합니다.

- 큐 관리자에서 지속 가능 구독의 지속 사본을 보유하는 데 사용되는 로컬 큐 SYSTEM.DURABLE.SUBSCRIBER.QUEUE. 성능을 향상시키려면 제공된 샘플 큐 정의에 표시된 대로, 색인 유형이 CORRELID로 이 큐를 정의해야 합니다.
- 관리되는 지속 가능 구독의 모델로 사용되는 로컬 큐 SYSTEM.DURABLE.MODEL.QUEUE.
- 관리되는 지속 불가능한 구독의 모델로 사용되는 로컬 큐 SYSTEM.NDURABLE.MODEL.QUEUE.
- 큐에 있는 발행/구독 인터페이스에서 모니터하는 큐 이름 목록을 포함하는 이름 목록 SYSTEM.QPUBSUB.QUEUE.NAMELIST.
- 큐에 있는 발행/구독 인터페이스가 구독 지점에 토픽 오브젝트를 일치시키기 위해 사용하는 토픽 오브젝트 목록을 포함하는 이름 목록 list called SYSTEM.QPUBSUB.SUBPOINT.NAMELIST.
- 속성을 해석하는 기본 토픽을 사용되는 토픽 SYSTEM.BASE.TOPIC.
- 큐에 있는 발행/구독 인터페이스에서 사용하는 기본 스트림인 토픽 SYSTEM.BROKER.DEFAULT.STREAM.
- 큐에 있는 발행/구독 인터페이스에서 사용하는 기본 RFH2 구독 지점인 토픽 SYSTEM.BROKER.DEFAULT.SUBPOINT.
- 큐에 있는 발행/구독 인터페이스에서 사용하는 관리 스트림인 토픽 SYSTEM.BROKER.ADMIN.STREAM.
- DEFINE SUB 명령에서 기본값을 제공하는 데 사용되는 기본 구독 오브젝트인 구독 SYSTEM.DEFAULT.SUB.

시스템 기본 오브젝트

시스템 기본 오브젝트는 오브젝트를 정의할 때 기본 속성을 제공하는 데 사용되며 정의를 기반으로 하는 다른 오브젝트 이름을 지정하지 않습니다.

기본 시스템 오브젝트 정의의 이름은 문자 "SYSTEM.DEFAULT" 또는 "SYSTEM.DEF"로 시작합니다. 예를 들어 시스템 기본 로컬 큐 이름은 SYSTEM.DEFAULT.LOCAL.QUEUE입니다.

이러한 오브젝트는 이러한 IBM MQ 오브젝트의 속성에 대한 시스템 기본값을 정의합니다.

- 로컬 큐
- 모델 큐
- 알리어스 큐
- 리모트 큐
- 프로세스
- 이름 목록
- 채널
- 스토리지 클래스
- 인증 정보

공유 큐는 로컬 큐의 특수 유형이므로 공유 큐를 정의할 때 정의는 SYSTEM.DEFAULT.LOCAL.QUEUE에 기반합니다. 기본 정의에 지정되지 않았으므로 커플링 기능 구조 이름에 대한 값을 제공해야 함을 명심해야 합니다. 또는 모두 필수 속성을 상속하도록 공유 큐에 대해 기본으로 사용할 사용자 고유의 기본 공유 큐 정의를 정의할 수 있습니다. 큐 공유 그룹에서만 하나의 큐 관리자에서 공유 큐를 정의해야 한다는 점을 기억하십시오.

시스템 명령 오브젝트

시스템 명령 오브젝트의 이름은 문자 SYSTEM.COMMAND로 시작합니다. IBM MQ 조작 및 제어판을 사용하여 IBM MQ 서브시스템에 명령을 실행하기 전에 이러한 오브젝트를 정의해야 합니다.

다음과 같은 두 개의 시스템 명령 오브젝트가 있습니다.

1. 시스템 명령 입력 큐는 IBM MQ 명령 프로세서에서 처리하기 전에 명령이 배치되는 로컬 큐입니다. SYSTEM.COMMAND.INPUT이 호출되어야 합니다. SYSTEM.ADMIN.COMMAND.QUEUE 도 IBM MQ for

Multiplatforms와의 호환성을 위해, 그리고 IBM MQ Console 및 administrative REST API에서 사용하도록 정의되어야 합니다.

2. SYSTEM.COMMAND.REPLY.MODEL은 시스템 명령 응답 대상 큐를 정의하는 모델 큐입니다.

IBM MQ Explorer에서 사용할 다음과 같은 2개의 추가 오브젝트가 있습니다.

- SYSTEM.MQEXPLORER.REPLY.MODEL 큐
- SYSTEM.ADMIN.SVRCONN 채널

SYSTEM.REST.REPLY.QUEUE 는 IBM MQ administrative REST API에서 사용되는 응답 큐입니다.

명령은 일반적으로 비지속 메시지를 사용하여 전송됩니다. 따라서 이들을 사용하는 애플리케이션(유틸리티 프로그램과 같은 제공된 애플리케이션 및 조작과 제어판 포함)에서 기본적으로 비지속 메시지를 가져오도록 두 시스템 명령 오브젝트 모두 DEFPSIST(NO) 속성을 보유해야 합니다. 명령에 대한 지속 메시지를 사용하는 애플리케이션을 보유한 경우 이러한 명령에 대한 응답 메시지는 지속되므로 응답 대상 큐에 대해 DEFTYPE(PERMDYN) 속성을 설정하십시오.

시스템 관리 오브젝트

시스템 관리 오브젝트의 이름은 문자 SYSTEM.ADMIN으로 시작합니다.

다음과 같은 7개의 시스템 관리 오브젝트가 있습니다.

- SYSTEM.ADMIN.CHANNEL.EVENT 큐
- SYSTEM.ADMIN.COMMAND.EVENT 큐
- SYSTEM.ADMIN.CONFIG.EVENT 큐
- SYSTEM.ADMIN.PERFM.EVENT 큐
- SYSTEM.ADMIN.QMGR.EVENT 큐
- SYSTEM.ADMIN.TRACE.ROUTE.QUEUE 큐
- SYSTEM.ADMIN.ACTIVITY.QUEUE 큐

채널 큐

분산 큐잉을 사용하려면 다음 오브젝트를 정의해야 합니다.

- 채널의 논리적 작업 단위 ID(LUWID) 및 순서 번호를 유지보수하는 데 사용되는 이름이 SYSTEM.CHANNEL.SYNCQ인 로컬 큐. 채널 성능을 향상시키려면 제공된 샘플 큐 정의에 표시된 대로, 색인 유형이 MSGID로 이 큐를 정의해야 합니다.
- 채널 명령에 사용되는 이름이 SYSTEM.CHANNEL.INITQ인 로컬 큐.

이러한 큐는 공유 큐로 정의할 수 없습니다.

클러스터 큐

IBM MQ 클러스터를 사용하려면 다음 오브젝트를 정의해야 합니다.

- 큐 관리자 사이에서 저장소 변경을 통신하는 데 사용되는 로컬 큐 SYSTEM.CLUSTER.COMMAND.QUEUE. 이 큐에 기록된 메시지는 저장소 데이터에 대한 요청 또는 저장소의 로컬 사본에 적용될 저장소 데이터에 대한 업데이트를 포함합니다.
- 저장소의 지속 사본을 보유하는 데 사용되는 로컬 큐 SYSTEM.CLUSTER.REPOSITORY.QUEUE.
- 클러스터의 모든 목적지에 대한 전송 큐에 해당하는 로컬 큐 SYSTEM.CLUSTER.TRANSMIT.QUEUE. 성능상의 이유로 샘플 큐 정의에 표시된 대로, 색인 유형이 CORRELID로 이 큐를 정의해야 합니다.

일반적으로 이러한 큐는 많은 양의 메시지를 포함합니다.

이러한 큐는 공유 큐로 정의할 수 없습니다.

큐 공유 그룹 큐

공유 채널 및 그룹 내 큐잉을 사용하려면 다음 오브젝트를 정의해야 합니다.

- 공유 채널에 대한 동기화 정보를 보유하는 데 사용되는 이름이 SYSTEM.QSG.CHANNEL.SYNCQ인 공유 큐.
- 그룹 내 큐잉에 대한 전송 큐로 사용되는 이름이 SYSTEM.QSG.TRANSMIT.QUEUE인 공유 큐. 큐 공유 그룹에서 실행하는 경우 그룹 내 큐잉을 사용하지 않아도 이 큐를 정의해야 합니다.

스토리지 클래스

다음 6개의 스토리지 클래스를 정의하는 것이 좋습니다. IBM MQ에 필요하므로 이들 중 4개를 정의해야 합니다. 이들은 샘플 큐 정의에 사용되므로 다른 스토리지 클래스 정의가 권장됩니다.

DEFAULT(필수)
이 스토리지 클래스는 성능에 중요하지 않으며, 다른 스토리지 클래스에 적합하지 않은 모든 메시지 큐에서 사용됩니다. 또한 큐를 정의할 때 지정하지 않은 경우 제공되는 기본 스토리지 클래스이기도 합니다.

NODEFINE(필수)
큐를 정의할 때 지정된 스토리지 클래스가 정의되지 않은 경우 이 스토리지 클래스가 사용됩니다.

REMOTE(필수)
이 스토리지 클래스는 1차적으로 전송 큐(즉, 성능에 중요한 단기간 메시지를 포함하는 시스템 관련 큐)에 대해 사용됩니다.

SYSLNGLV
성능에 중요한 장기간 메시지에 대해 이 스토리지 클래스가 사용됩니다.

SYSTEM(필수)
이 스토리지 클래스는 성능에 중요한 시스템 관련 메시지 큐 (예: SYSTEM.CHANNEL.SYNQ 및 SYSTEM.CLUSTER.* 큐.

SYSVOLAT
성능에 중요한 단기간 메시지에 대해 이 스토리지 클래스가 사용됩니다.

해당 속성을 수정하고 필요한 경우 다른 스토리지 클래스 정의를 추가할 수 있습니다.

시스템 오브젝트 데드-레터 큐 정의

메시지 목적지가 올바르지 않은 경우 데드-레터 큐가 사용됩니다. IBM MQ는 데드-레터 큐라고 하는 로컬 큐에 이러한 메시지를 배치합니다. 데드-레터 큐가 필수사항이 아니어도 특히 IBM MQ 브릿지 중 하나 또는 분산 큐잉을 사용하는 경우 이를 중요한 항목으로 간주해야 합니다.

데드-레터 큐를 공유 큐로 정의하지 **마십시오**. 하나의 큐 관리자에서 로컬 큐에 넣으면 데드-레터 큐에 넣어질 수 있습니다. 데드-레터 큐가 공유 큐이면, 다른 시스템의 데드-레터 큐 핸들러가 메시지를 처리하여 이를 같은 이름을 가진 큐에 넣지만 이 핸들러가 다른 큐 관리자에 있기 때문에 해당 메시지는 올바르지 않은 큐가 되거나 다른 보안 프로파일을 가집니다. 큐가 존재하지 않으면 해당 메시지를 재처리하는 데 실패합니다.

데드-레터 큐를 정의하도록 결정한 경우 해당 이름을 큐 관리자에게 알려야 합니다. 이를 수행하려면 ALTER QMGR DEADQ(*queue-name*) 명령을 사용하십시오. 자세한 정보는 큐 관리자 속성 표시 및 변경을 참조하십시오.

기본 전송 큐

다른 큐 관리자로 메시지를 송신할 때 사용 가능한 다른 적합한 전송 큐가 없으면 기본 전송 큐가 사용됩니다. 기본 전송 큐를 정의하는 경우 큐 역할을 할 채널도 정의해야 합니다. 이를 수행하지 않은 경우 기본 전송 큐에 배치된 메시지는 리모트 큐 관리자로 전송되지 않으며 큐에 남아 있습니다.

기본 전송 큐를 정의하도록 결정한 경우 해당 이름을 큐 관리자에게 알려야 합니다. 이를 수행하려면 ALTER QMGR 명령을 사용하십시오.

내부 큐

- **보류 중인 데이터 큐**
  - 내부 사용을 위해 정의된 큐, SYSTEM.PENDING.DATA.QUEUE는 JMS 발행/구독 환경에서 지속 가능한 구독의 사용을 지원합니다.
- **JMS 2.0 전달 지연 스테이징 큐**
  - JMS 2.0 에서 제공하는 전달 지연 기능이 사용되면 내부 스테이징 큐, SYSTEM.DDELAY.LOCAL.QUEUE를 정의해야 합니다. 이 큐는 전달 지연이 완료될 때까지 0이 아닌 전달 지연으로 송신된 메시지를 임시로 저장하기 위해 큐 관리자에 의해 사용되며 이 메시지는 해당 대상 목적지에 배치됩니다. 이 큐에 대해 CSQ4INSG에서 주석 처리된 샘플 정의가 제공됩니다.
  - SYSTEM.DDELAY.LOCAL.QUEUE 큐를 정의할 경우, 전달 지연과 함께 전송될 예상 메시지 수에 대해 STGCLASS, MAXMSGL 및 MAXDEPTH 속성을 설정해야 합니다. 또한 SYSTEM.DDELAY.LOCAL.QUEUE 큐를 정의할 때 큐 관리자만 메시지를 이 큐에 넣을 수 있는지 확인하십시오. 사용자 ID에는 메시지를 이 큐에 넣을 수 있는 권한이 없는지 확인해야 합니다.

채널 인증 큐

채널 인증을 내부적으로 사용하려면 SYSTEM.CHLAUTH.DATA.QUEUE 큐가 필요합니다. 이러한 오브젝트를 정의하는 데 도움을 주는 샘플 정의를 IBM MQ에서 제공합니다. 이 샘플에 대해서는 CSQ4INSA에서 설명하며, 여기서는 몇 가지 기본 규칙도 함께 정의합니다.

z/OS IBM MQ for z/OS에서 큐 관리자 성능 조정

기본 성능 문제점을 방지하기 위해 큐 관리자를 조정하도록 취할 수 있는 몇 가지 간단한 단계가 있습니다.

ALTER QMGR 명령에서 설정된 큐 관리자 속성으로 제어되는 큐 관리자의 성능을 향상시킬 수 있는 몇 가지 방법이 있습니다. 이 절에는 큐 관리자에서 '정리'를 수행하거나 큐 관리자에 허용된 최대 메시지 수를 설정하여 이를 수행할 수 있는 방법에 대한 정보가 포함됩니다. IBM MQ SupportPac MP16 -IBMMQ ~을 위한z/OS 용량 계획 및 튜닝는 성능과 성능 조정에 대한 자세한 정보를 제공합니다.

동기점

큐 관리자의 역할 중 하나는 애플리케이션의 동기점 제어입니다. 애플리케이션은 MQCMIT 호출로 종료된 임의의 수의 MQPUT 또는 MQGET 호출을 포함하는 작업 단위를 구성합니다.

하나의 MQCMIT 범위 내 MQPUT 또는 MQGET 호출 수가 증가함에 따라 커미트의 성능 비용이 크게 증가할 수 있습니다. 일반적으로 애플리케이션은 단일 동기점 내에서 많은 수의 메시지를 MQPUT/MQGET하지 않도록 설계해야 합니다.

MAXUMSGS 큐 관리자 속성을 사용하여 단일 동기점 내 메시지 수를 관리적으로 제한할 수 있습니다. 애플리케이션이 이 한계를 초과하는 경우 한계를 초과하는 MQPUT,MQPUT1 또는 MQGET 호출에서 MQRC_SYNCPOINT_LIMIT_REACHED를 수신합니다. 그런 다음 애플리케이션이 MQCMIT 또는 MQBACK을 적절히 발행해야 합니다.

MAXUMSGS의 기본값은 10000입니다. 하한을 적용하려는 경우 이 값을 낮출 수 있으며, 이는 루프 애플리케이션으로부터 보호하는 데 도움이 될 수도 있습니다. MAXUMSGS를 줄이기 전에 기존 애플리케이션을 파악하여 한계를 초과하지 않거나 MQRC_SYNCPOINT_LIMIT_REACHED 리턴 코드를 허용할 수 있는지 확인하십시오.

만기된 메시지

만기된 메시지는 다음의 적절한 MQGET 호출에서 제거됩니다. 그러나 이러한 호출이 발생하지 않으면 만기된 메시지가 제거되지 않으며 일부 큐, 특히, MessageId, CorrelId 또는 GroupId로 메시지 검색이 수행되고 성능을 위해 큐가 색인화된 항목에서 많은 만기된 메시지가 누적될 수 있습니다. 큐 관리자는 만기된 메시지가 있는지 큐를

정기적으로 스캔할 수 있습니다. 그러면 이 메시지가 삭제됩니다. 적어도 이 스캔을 수행하는 간격을 선택할 수 있습니다. 이를 수행하는 다음과 같은 두 가지 방법이 있습니다.

명확한 요청
스캔하는 큐 및 시기를 제어할 수 있습니다. 스캔하려는 하나 이상의 큐를 지정하여 REFRESH QMGR TYPE(EXPIRY) 명령을 실행하십시오.

정기적 스캔
EXPRYINT 속성을 사용하여 큐 관리자 오브젝트에서 만기 간격을 지정할 수 있습니다. 큐 관리자는 각 큐에서 만기된 메시지에 대한 정보를 유지보수하며 만기된 메시지의 스캔이 중요한 시점을 확인합니다. EXPRYINT 간격에 도달할 때마다 큐 관리자는 만기된 메시지를 스캔할만한 후보 큐를 찾고 적합하다고 판단되는 해당 큐만 스캔합니다. 모든 큐를 스캔하지는 않습니다. 이를 통해 필요하지 않은 스캔에 프로세서 시간이 낭비되는 것을 피할 수 있습니다.

공유 큐는 큐 공유 그룹에서 하나의 큐 관리자에 의해서만 스캔됩니다. 일반적으로 재시작할 큐 관리자 또는 EXPRYINT가 설정된 첫 번째 항목이 스캔을 수행합니다.

참고: 큐 공유 그룹 내의 모든 큐 관리자에 대해 동일한 EXPRYINT 값을 설정해야 합니다.

z/OS IBM MQ for z/OS에서 제공하는 샘플 정의

IBM MQ for z/OS에서 제공하는 샘플 JCL 및 코드에 대한 참조로 이 토픽을 사용하십시오.

다음과 같은 샘플 정의는 IBM MQ의 thlqual.SCSQPROC 라이브러리에서 제공됩니다. 이들을 사용하여 시스템 오브젝트를 정의하고 사용자 고유의 오브젝트로 사용자 정의할 수 있습니다. 초기화 입력 데이터 세트 (초기화 명령어 에 설명되어 있음)에 일부를 포함시킬 수 있습니다.

표 *22.* 시스템 오브젝트에 대한 *IBM MQ* 샘플 정의

초기화 입력 데이터 세트	**샘플 이름**
CSQINP1	CSQ4INP1 CSQ4INPR
CSQINP2	CSQ4INSA CSQ4INYS[1] CSQ4INSX CSQ4INSG CSQ4INSR CSQ4INSS CSQ4INSJ CSQ4INSM CSQ4INYG CSQ4INYR CSQ4INYC CSQ4INYD CSQ4INSC
CSQINPT	CSQ4INST CSQ4INYT
기타	CSQ4DISP CSQ4INPX CSQ4IVPQ CSQ4IVPG CSQ4MSTR CSQ4MSRR CSQ4QMIN

참고:

1. 이러한 샘플 정의의 순서가 중요합니다. INYS, INSX 및 INSG가 잘못 정렬된 경우 오류가 발생합니다.

CSQINP1 샘플

메시지의 주요 클래스에 대해 다중 페이지 세트를 사용하는 경우 thlqual.SCSQPROC(CSQ4INPR) 또는 메시지의 각 클래스에 대해 하나의 페이지 세트를 사용하는 경우 샘플 CSQINP1 데이터 세트 thlqual.SCSQPROC(CSQ4INP1)를 사용하십시오. 버퍼 풀, 버퍼 풀 연관에 대한 페이지 세트 및 ALTER SECURITY 명령의 정의가 이에 포함됩니다. 큐 관리자 시작된 태스크 프로시저의 CSQINP1 연결에 샘플을 포함하십시오.

CSQINP2 샘플

CSQ4INSG 시스템 오브젝트 샘플

샘플 CSQINP2 데이터 세트 thlqual.SCSQPROC(CSQ4INSG)는 일반 사용을 위해 다음 시스템 오브젝트에 대한 정의를 포함합니다.

- 시스템 기본 오브젝트
- 시스템 명령 오브젝트
- 시스템 관리 오브젝트
- 시스템 사용을 위한 기타 오브젝트

이 샘플에서 오브젝트를 정의해야 합니다. 그러나 서브시스템이 먼저 시작된 경우 한 번만 이를 수행해야 합니다. CSQINP2 데이터 세트에 정의를 포함하는 것이 이를 수행하는 최상의 방법입니다. 이들은 큐 관리자 시스템 종료 및 재시작에서 유지보수됩니다. 오브젝트 이름은 변경해서는 안되지만, 필요한 경우 해당 속성을 변경할 수 있습니다.

다음 조건이 충족되면 하나의 메시지가 SYSTEM.DURABLE.SUBSCRIBER.QUEUE 큐에 넣어집니다(발행/구독이 활성이 아닌 경우에도).

- QMGR 속성 PSMODE가 DISABLED로 설정됨
- 샘플 오브젝트 CSQ4INST문 `DEFINE SUB('SYSTEM.DEFAULT.SUB')`가 있음.

위의 동작을 예방하려면 `DEFINE SUB('SYSTEM.DEFAULT.SUB')`문을 삭제하거나 주석 처리하십시오.

JMS 2.0 전달 지연이 사용되는 경우 JMS 2.0 전달 지연 스테이징 큐인 SYSTEM.DDELAY.LOCAL.QUEUE만 정의해야 합니다. 기본적으로 큐 정의는 주석 처리되며, 필요한 경우 주석 처리를 취소할 수 있습니다.

CSQ4INSA 시스템 오브젝트 및 인증 샘플

샘플 CSQINP2 데이터 세트 thlqual.SCSQPROC(CSQ4INSA)에는 채널 인증 시스템 큐 정의가 포함됩니다. 이 큐에는 채널 인증 레코드가 들어 있습니다. 또한 기본 채널 인증 규칙도 포함되어 있습니다.

큐 관리자에서 CHLAUTH가 ENABLED 상태이고 사용자가 채널을 실행하거나 SET 또는 DISPLAY CHLAUTH 레코드를 사용하려는 경우 이 샘플에서 오브젝트를 정의해야 합니다. 서브시스템이 처음 시작된 경우에만 오브젝트를 정의해야 합니다. CSQINP2 데이터 세트에 정의를 포함하는 것이 이를 수행하는 최상의 방법입니다. 대기열 관리자가 종료되고 다시 시작되는 동안에도 유지됩니다. 대기열 이름을 변경해서는 안 됩니다.

CSQ4INSS 시스템 오브젝트 샘플

큐 공유 그룹을 사용하는 경우 추가 시스템 오브젝트를 정의할 수 있습니다.

샘플 데이터 세트 thlqual.SCSQPROC(CSQ4INSS)는 공유 채널 및 그룹 내 큐잉에 필요한 시스템 오브젝트의 정의 세트와 CF 구조와 함께 사용할 샘플 명령을 포함합니다.

이 샘플은 그대로 사용할 수 없습니다. 사용하기 전에 사용자 정의해야 합니다. 그런 다음 큐 관리자 시동 프로시저의 CSQINP2 DD 연결에 이 멤버를 포함하거나 CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 이를 사용하여 필수 명령을 실행할 수 있습니다.

그룹 또는 공유 오브젝트를 정의하는 경우 큐 공유 그룹에 있는 큐 관리자 하나에 대해서만 CSQINP2 DD 연결에 이들을 포함해야 합니다.

CSQ4INSX 시스템 오브젝트 샘플

분산 큐잉 및 클러스터링을 사용하는 경우 추가 시스템 오브젝트를 정의해야 합니다.

샘플 데이터 세트 thlqual.SCSQPROC(CSQ4INSX)는 필요한 큐 정의를 포함합니다. 큐 관리자 시동 프로시저의 CSQINP2 DD 연결에 이 멤버를 포함하거나 CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 이를 사용하여 필수 DEFINE 명령을 실행할 수 있습니다.

다음과 같은 두 가지 유형의 오브젝트 정의가 있습니다.

- SYSTEM.CHANNEL.xx(분산 큐잉에 필요함)
- SYSTEM.CLUSTER.xx(클러스터에 필요함)

CSQ4INSJ 시스템 JMS 오브젝트 샘플

JMS 발행/구독 도메인에 사용되는 큐를 정의합니다.

CSQ4INSM 시스템 오브젝트 샘플

고급 메시지 보안을 사용 중인 경우, 추가 시스템 오브젝트를 정의해야 합니다. 샘플 데이터 세트 thlqual.SCSQPROC(CSQ4INSM)는 필요한 큐 정의를 포함합니다.

CSQ4INSR 오브젝트 샘플

WebSphere Application Server 및 브로커에서 사용하는 큐를 정의합니다.

CSQ4INYD 오브젝트 샘플

분산 큐잉을 사용하고 사용자 고유의 큐, 프로세스 및 채널을 설정해야 하는 경우.

샘플 데이터 세트 thlqual.SCSQPROC(CSQ4INYD)는 분산 큐잉 오브젝트를 사용자 정의하는 데 사용할 수 있는 샘플 정의를 포함합니다. 다음과 같이 구성됩니다.

- 송신 끝에 대한 정의 세트
- 수신 끝에 대한 정의 세트
- 클라이언트 사용에 대한 정의 세트

이 샘플은 그대로 사용할 수 없습니다. 사용하기 전에 사용자 정의해야 합니다. 그런 다음 큐 관리자 시동 프로시저의 CSQINP2 DD 연결에 이 멤버를 포함하거나 CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 이를 사용하여 필수 DEFINE 명령을 실행할 수 있습니다. (큐 관리자를 재시작할 때마다 이러한 오브젝트를 재정의하지 않아도 됨을 의미하므로 선호됩니다.)

CSQ4INYC 오브젝트 샘플

클러스터를 사용하는 경우 분산 큐잉의 리모트 큐 정의 및 채널 정의와 동등한 정의가 필요하면 자동 작성됩니다. 그러나 하나 이상의 클러스터 저장소 큐 관리자에 대한 클러스터-송신자 정의 및 클러스터에 대한 클러스터-수신자 채널과 같은 일부 수동 채널 정의가 필요합니다.

샘플 데이터 세트: thlqual.SCSQPROC(CSQ4INYD)는 클러스터링 오브젝트를 사용자 정의하는 데 사용할 수 있는 다음과 같은 샘플 정의를 포함합니다.

- 큐 관리자에 대한 정의
- 수신 채널에 대한 정의
- 송신 채널에 대한 정의
- 클러스터 큐에 대한 정의
- 클러스터 목록에 대한 정의

이 샘플은 그대로 사용할 수 없습니다. 사용하기 전에 사용자 정의해야 합니다. 그런 다음 큐 관리자 시동 프로시저의 CSQINP2 DD 연결에 이 멤버를 포함하거나 CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 이를 사용하여 필수 DEFINE 명령을 실행할 수 있습니다. IBM MQ를 재시작할 때마다 이러한 오브젝트를 재정의하지 않아도 됨을 의미하므로 선호됩니다.

CSQ4INYG 오브젝트 샘플

샘플 데이터 세트: thlqual.SCSQPROC(CSQ4INYG)는 일반 용도로 사용하도록 오브젝트를 사용자 정의하는 데 사용할 수 있는 다음과 같은 샘플 정의를 포함합니다.

- 데드-레터 큐
- 기본 전송 큐
- CICS 어댑터 오브젝트

이 샘플은 그대로 사용할 수 없습니다. 사용하기 전에 사용자 정의해야 합니다. 그런 다음 큐 관리자 시동 프로시저의 CSQINP2 DD 연결에 이 멤버를 포함하거나 CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 이를 사용하여 필수 DEFINE 명령을 실행할 수 있습니다. IBM MQ를 재시작할 때마다 이러한 오브젝트를 재정의하지 않아도 됨을 의미하므로 선호됩니다.

여기의 샘플 정의 외에도 자원 정의에 대한 기초로 시스템 오브젝트 정의를 사용할 수 있습니다. 예를 들어 SYSTEM.DEFAULT.LOCAL.QUEUE의 작업 사본을 작성하고 이름을 MY.DEFAULT.LOCAL.QUEUE로 지정할 수 있습니다. 그런 다음 필요한 경우 이 사본에서 매개변수를 변경할 수 있습니다. 그리고 해당 유형의 자원을 작성할 권한이 있는 경우 선택한 메소드로 DEFINE 명령일 실행할 수 있습니다.

기본 전송 큐
기본 전송 큐를 정의할 것인지 여부를 결정하기 전에 기본 전송 큐 설명을 읽으십시오.

- 기본 전송 큐를 정의하도록 결정한 경우 이를 지원할 채널도 정의해야 함을 명심하십시오.
- 정의하지 않도록 결정한 경우 샘플의 ALTER QMGR 명령에서 DEFXMITQ 문을 제거하는 것을 명심하십시오.

CICS 어댑터 오브젝트
샘플에서는 이니시에이션 큐 CICS01.INITQ를 정의합니다. 이 큐는 IBM MQ에서 제공하는 CKTI 트랜잭션에서 사용합니다. 이 큐 이름은 변경할 수 있습니다. 그러나 INITPARM 문에 있는 SYSIN 대체 또는 CICS 시스템 초기화 테이블(SIT)에 지정된 이름과 일치해야 합니다.

CSQ4INYS/CSQ4INYR 오브젝트 샘플

사용할 스토리지 클래스 정의:

- 각 메시지 클래스에 대한 하나의 페이지 세트
- 메시지의 주요 클래스에 대한 다중 페이지 세트

예를 들어, SYSTEM.COMMAND.INPUT 는 STGCLASS ('SYSVOLAT') 및
SYSTEM.CLUSTER.TRANSMIT.QUEUE 는 STGCLASS ('REMOTE') 를 사용합니다. CSQ4INYS에서는 두 스토리지 클래스가 모두 동일한 페이지 세트를 사용합니다. CSQ4INYR에서 해당 스토리지 클래스는 전송 큐 채우기의 영향을 줄이기 위해 다른 페이지 세트를 사용합니다.

CSQINPT 샘플

CSQ4INST
샘플 데이터 세트: thlqual.SCSQPROC(CSQ4INST)는 시스템 기본 구독에 대한 정의를 포함합니다.

이 샘플에서 오브젝트를 정의해야 합니다. 그러나 발행/구독 엔진이 먼저 시작된 경우 한 번만 이를 수행해야 합니다. CSQINPT데이터 세트에 정의를 포함하는 것이 이를 수행하는 최상의 방법입니다. 이는 큐 관리자 시스템 종료 및 재시작에서 유지보수됩니다. 오브젝트 이름은 변경해서는 안되지만, 필요한 경우 해당 속성을 변경할 수 있습니다.

CSQ4INYT
샘플 데이터 세트: thlqual.SCSQPROC(CSQ4INYT)는 발행/구독 엔진이 시작될 때 실행하려는 명령 세트를 포함합니다. 이 샘플은 토픽 및 구독 정보를 표시합니다.

기타

CSQ4DISP 표시 샘플

샘플 데이터 세트: thlqual.SCSQPROC(CSQ4DISP)는 큐 관리자에 있는 모든 정의된 자원을 표시하는 일반 DISPLAY 명령의 세트를 포함합니다. 여기에는 스토리지 클래스 및 추적과 같은 정의 및 모든 IBM MQ 오브젝트에 대한 정의가 포함됩니다. 이러한 명령은 많은 양의 출력을 생성할 수 있습니다. CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 또는 CSQINP2 데이터 세트에서 이 샘플을 사용할 수 있습니다.

CSQ4INPX 샘플

샘플 데이터 세트: thlqual.SCSQPROC(CSQ4INPX)는 채널 시작기가 시작될 때 실행하려는 명령 세트를 포함합니다. 사용하기 전에 이 샘플을 사용자 정의해야 합니다. 그런 다음 채널 시작기에 대해 CSQINPX 데이터 세트에 이를 포함할 수 있습니다.

CSQ4IVPQ 및 CSQ4IVPG 샘플

샘플 데이터 세트: thlqual.SCSQPROC(CSQ4IVPQ) 및 thlqual.SCSQPROC(CSQ4IVPG)는 설치 확인 프로그램(IVP)을 실행하는 데 필요한 DEFINE 명령의 세트를 포함합니다.

CSQINP2 데이터 세트에 이러한 샘플을 포함할 수 있습니다. IVP를 성공적으로 실행한 경우 큐 관리자를 재시작할 때마다 이들을 다시 실행하지 않아도 됩니다. 따라서 CSQINP2 연결에 이 샘플을 영구적으로 보관하지 않아도 됩니다.

CSQ4MSTR 및 CSQ4MSRR 샘플

다음은 thlqual.SCSQPROC(CSQ4MSTR) 및 thlqual.SCSQPROC(CSQ4MSRR) 큐 관리자에 대한 시작 태스크 프로시저 샘플입니다.

CSQ4MSRR 은 중요한 큐가 다른 페이지 세트에 분산되도록 CSQINP2 연결에서 CSQ4INYR 을 사용합니다.

필요한 경우 새로 작성된 큐 관리자에 대해 CSQMINI 카드를 사용할 수 있도록 주석을 제거할 수 있습니다.

CSQ4QMIN 샘플

샘플 QMINI 데이터 세트, thlqual.SCSQPROC(CSQ4QMIN).

QMINI 데이터 세트 및 **TransportSecurity** 스탠자에 대한 세부사항은 QMINI 데이터 세트 를 참조하십시오.

z/OS z/OS에서의 복구 및 재시작

이 토픽의 링크를 사용하여 재시작 및 복구를 위해 IBM MQ for z/OS의 기능에 대해 확인하십시오.

IBM MQ for z/OS에는 재시작 및 복구를 위한 강력한 기능이 있습니다. 중지된 후 큐 관리자가 복구하는 방법과 재시작 시 발생하는 상황에 대한 정보는 다음 하위 주제를 참조하십시오.

- 226 페이지의 『IBM MQ for z/OS의 데이터 변경 방법』
- 227 페이지의 『IBM MQ for z/OS에서 일관성 유지보수 방법』
- 229 페이지의 『IBM MQ for z/OS에서 종료 중에 발생하는 사항』
- 230 페이지의 『IBM MQ for z/OS에서 재시작 및 복구 중에 발생하는 사항』
- 232 페이지의 『복구 인다우트 단위를 해석하는 방법』
- 234 페이지의 『공유 큐 복구』

관련 개념

z/OS IBM MQ for z/OS 복구 조치

관련 태스크

백업 및 복구 계획

z/OS z/OS 관리

관련 참조

z/OS IBM MQ for z/OS에 대한 메시지

z/OS IBM MQ for z/OS의 데이터 변경 방법

IBM MQ for z/OS는 모든 데이터를 일관되게 보존하기 위해 다른 서브시스템과 상호작용해야 합니다. 이 토픽은 복구 단위, 복구 단위의 개념 및 백아웃에서 이들을 사용하는 방법에 대한 정보를 포함합니다.

복구 단위

복구 단위는 한 일관성 지점에서 다른 지점으로 IBM MQ 데이터를 변경하는, 애플리케이션 프로그램에 대한 단일 큐 관리자에서 수행하는 처리입니다. 일관성 지점(동기점 또는 커미트 지점이라고도 함)은 애플리케이션 프로그램에 액세스하는 모든 복구 가능한 데이터가 일관된 특정 시점입니다.

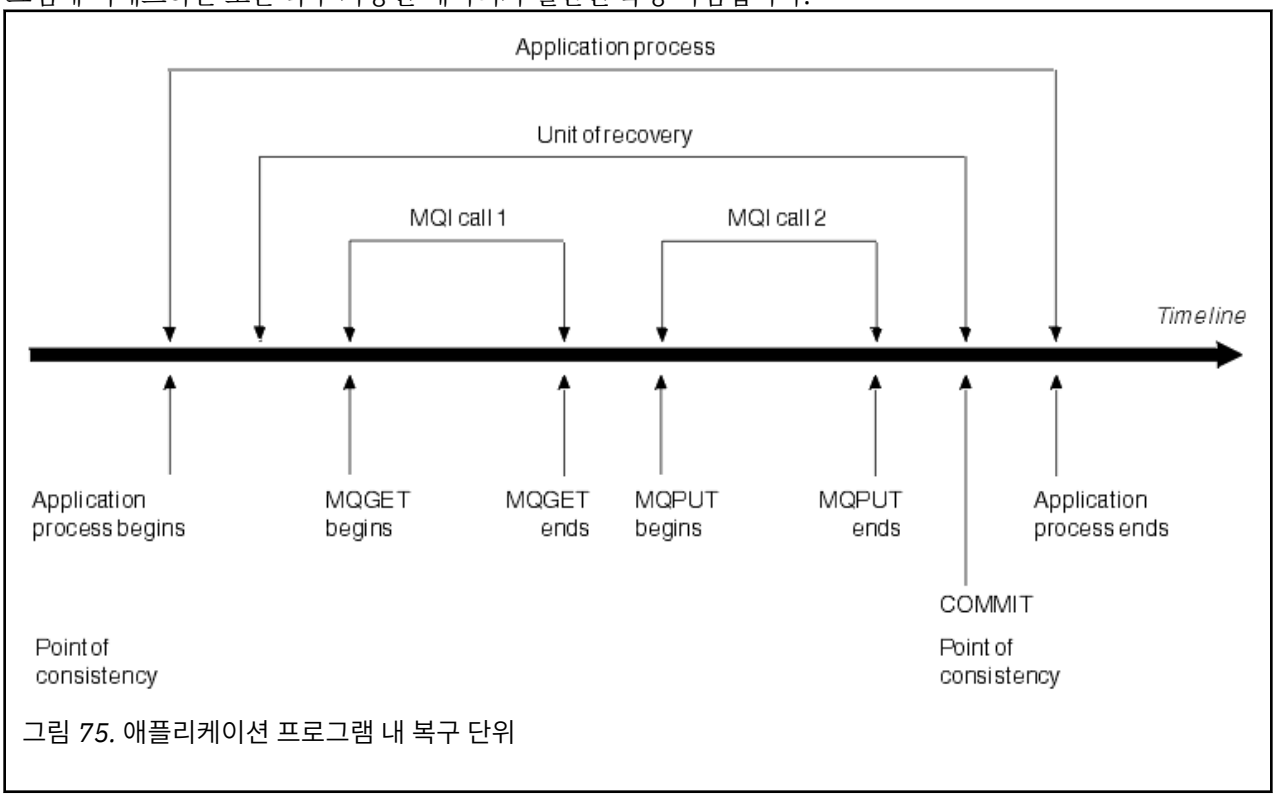


그림 75. 애플리케이션 프로그램 내 복구 단위

복구 단위는 이전 일관성 지점 뒤에서 또는 프로그램의 시작 이후에 데이터의 첫 번째 변경으로 시작됩니다. 그리고 이후 일관성의 지점으로 종료됩니다. 226 페이지의 그림 75에서는 복구 단위, 일관성 지점 및 애플리케이션 프로그램 사이의 관계를 보여줍니다. 이 예에서 애플리케이션 프로그램은 MQI 호출 1 및 2를 통해 큐를 변경합니다. 애플리케이션 프로그램은 둘 이상의 복구 단위 또는 하나만 포함할 수 있습니다. 그러나 모든 전체 복구 단위는 커미트 지점에서 종료됩니다.

예를 들어 은행 트랜잭션은 한 계좌에서 다른 계좌로 자금을 이체합니다. 먼저 프로그램은 첫 번째 계좌, 계좌 A에서 금액을 인출합니다. 그런 다음, 두 번째 계좌 B로 해당 금액을 추가합니다. A에서 금액을 인출한 후에는 두 계좌가 불일치하므로 IBM MQ는 커미트할 수 없습니다. 금액이 계좌 B에 추가되면 두 계좌가 일치하게 됩니다. 두 단계가 모두 완료되면 프로그램은 커미트를 통해 일관성 지점을 발표하여 다른 애플리케이션에서 변경사항을 볼 수 있도록 합니다.

애플리케이션 프로그램이 정상적으로 종료되면 일관성 지점이 자동으로 생성됩니다. CICS 및 IMS 프로그램에서 일부 프로그램 요청도 일관성 지점(예: EXEC CICS SYNCPOINT)을 생성합니다.

작업 백아웃

복구 단위 내에서 오류가 발생하면 IBM MQ는 데이터에 대한 변경을 제거하여, 데이터를 복구 단위 시작 시 해당 상태로 되돌립니다. 즉, IBM MQ는 작업을 백아웃합니다. 이벤트는 227 페이지의 그림 76에 표시됩니다.

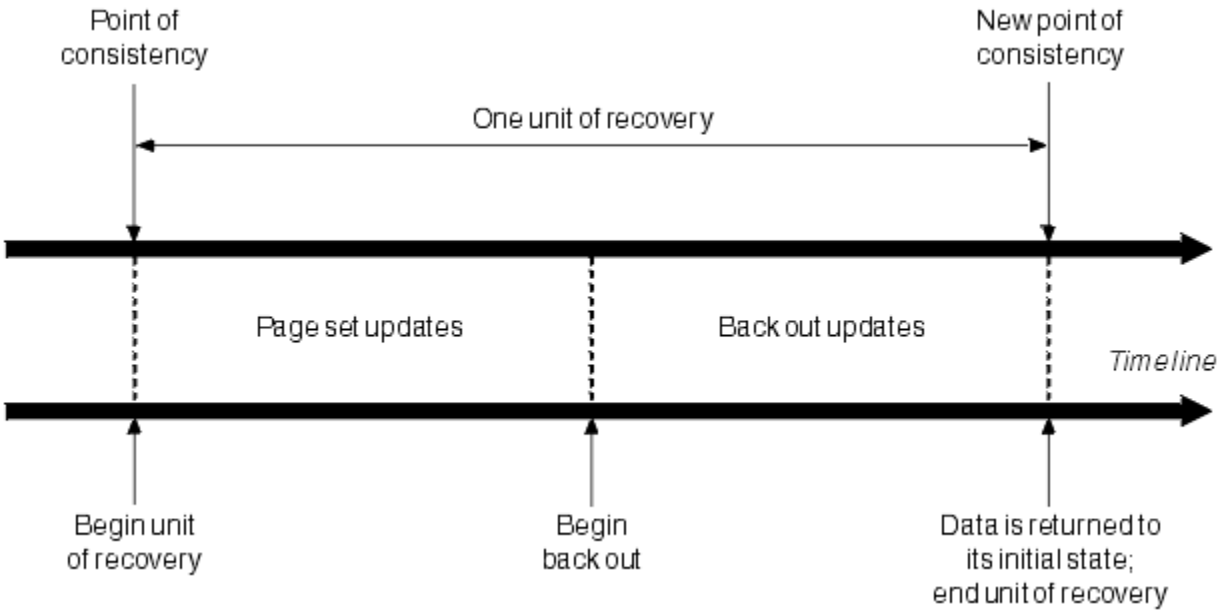


그림 76. 백아웃을 표시하는 복구 단위

z/OS IBM MQ for z/OS에서 일관성 유지보수 방법

IBM MQ for z/OS의 데이터는 배치, CICS, IMS 또는 TSO와 일관되어야 합니다. 한 곳에서 변경된 데이터는 다른 위치에서의 변경과 일치되어야 합니다.

한 시스템에서 변경된 데이터를 커미트하기 전에 다른 시스템이 해당 변경을 수행할 수 있는지 확인해야 합니다. 그러므로 시스템은 서로 통신해야 합니다.

*2*단계 커미트(예: CICS 하에서) 중에 한 서브시스템이 프로세스를 통합합니다. 해당 서브시스템은 통합기라고 하고 다른 항목을 참가자라고 합니다. CICS 또는 IMS는 IBM MQ와의 상호작용에서 통합기 역할을 하며, IBM MQ는 항상 참가자입니다. 배치 또는 TSO 환경에서 IBM MQ는 z/OS RRS에 의해 통합된 2단계 커미트 프로토콜에 참가할 수 있습니다.

*1*단계 커미트(예: TSO 또는 배치 하에서) 중에 IBM MQ는 항상 상호작용에서 통합기 역할을 하며 커미트 프로세스를 완전히 제어합니다.

WebSphere Application Server 환경에서 JMS 세션 오브젝트의 구문은 사용되는 항목(1단계 또는 2단계 커미트)을 판별합니다.

CICS 또는 IMS와의 일관성

IBM MQ와 CICS 또는 IMS 사이의 연결은 다음 동기점 프로토콜을 지원합니다.

- 2단계 커미트 - 둘 이상의 자원 관리자가 소유한 자원을 업데이트하는 트랜잭션의 경우.

  이는 표준 분산 동기점 프로토콜입니다. 1단계 커미트보다 더 많은 로깅 및 메시지 플로우를 포함합니다.
- 1단계 커미트 - 단일 자원 관리자(IBM MQ)가 소유한 자원을 업데이트하는 트랜잭션의 경우.

  이 프로토콜은 로깅 및 메시지 플로우에 대해 최적화되었습니다.
- 동기점 생략 - IBM MQ와 관련된 트랜잭션의 경우. 단, 동기점이 필요한 큐 관리자에서는 아무 것도 수행하지 않습니다(예: 큐 찾아보기).

각각의 경우에 CICS 또는 IMS가 동기점 관리자 역할을 수행합니다.

IBM MQ가 CICS 또는 IMS와 통신하는 데 사용하는 2단계 커미트 단계는 다음과 같습니다.

1. 1단계에서 각 시스템은 로그에 충분한 복구 정보가 기록되었는지 여부를 독립적으로 판별하고 해당 작업을 커미트할 수 있습니다.

   단계 종료 시 시스템이 통신합니다. 동의하는 경우 각각 다음 단계를 시작합니다.
2. 2단계에서 변경은 영구적으로 됩니다. 시스템 중 하나가 2단계 동안 이상종료되면 재시작 중에 복구 프로세스로 조작이 완료됩니다.

2단계 커미트 프로세스 설명

228 페이지의 그림 77에서는 2단계 커미트 프로세스를 설명합니다. CICS 또는 IMS 통합기의 이벤트는 위의 행에 표시되며 IBM MQ의 이벤트는 아래 행에 표시됩니다.

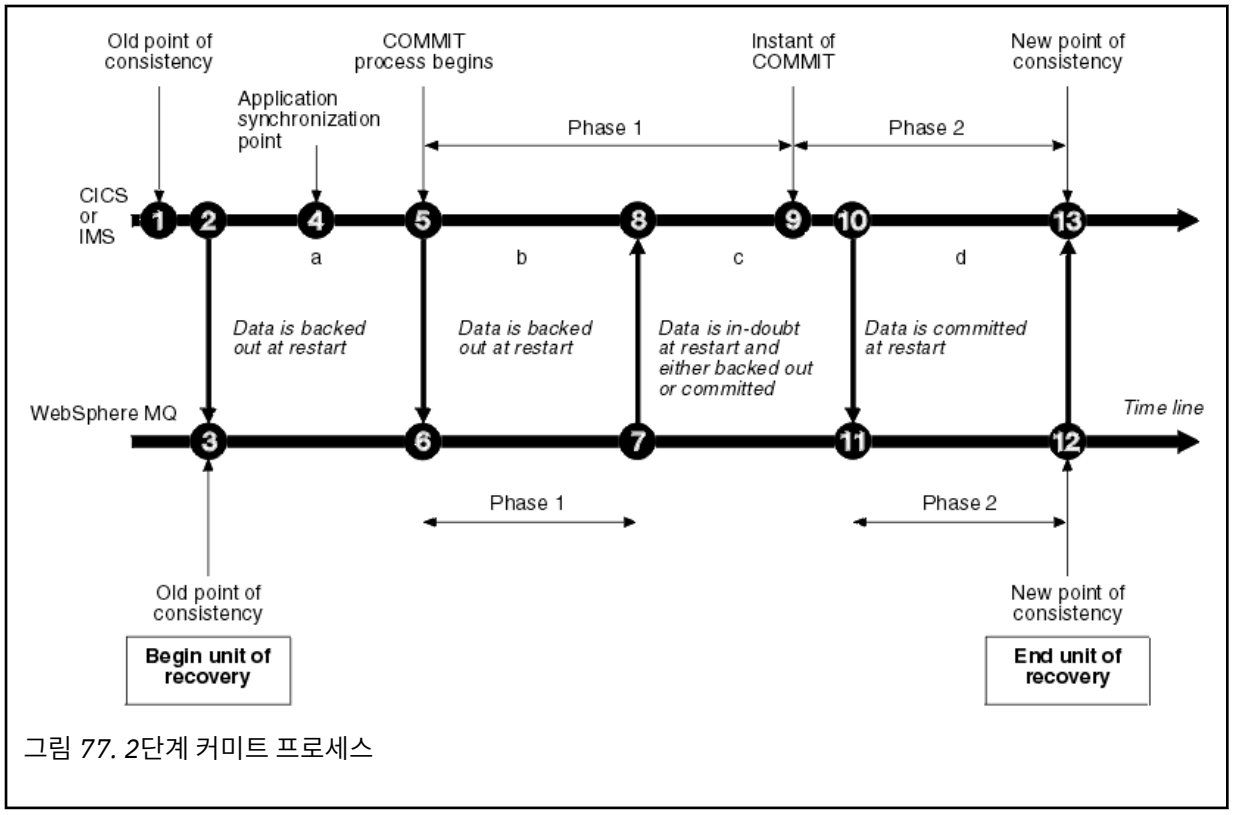


그림 77. 2단계 커미트 프로세스

다음 절의 숫자는 그림에 표시된 항목과 링크되어 있습니다.

1. 통합기의 데이터는 일관성 지점에 있습니다.
2. 통합기의 애플리케이션 프로그램은 IBM MQ를 호출하여 메시지를 추가해 큐를 업데이트합니다.
3. 이는 IBM MQ에서 복구 단위를 시작합니다.
4. 애플리케이션 동기화 지점에 도달할 때까지 통합기에서 처리가 계속됩니다.
5. 그런 다음 통합기는 커미트 처리를 시작합니다. CICS 프로그램은 SYNCPOINT 명령 또는 정상적인 애플리케이션 종료를 사용하여 커미트를 시작합니다. IMS 프로그램은 CHKP 호출, SYNC 호출, IOPCB에 대한 GET UNIQUE 호출 또는 정상적인 애플리케이션 종료를 사용하여 커미트를 시작할 수 있습니다. 커미트 처리의 1단계가 시작됩니다.
6. 통합기에서 1단계 처리가 시작되면 IBM MQ에서도 수행됩니다.
7. IBM MQ는 성공적으로 2단계를 완료하고 이 사실을 해당 로그에 기록하며 통합기에 알립니다.
8. 통합기가 알림을 수신합니다.
9. 통합기는 성공적으로 해당 1단계 처리를 완료합니다. 모두 1단계를 완료하고 오류에서 복구될 수 있으므로 이제 두 서브시스템은 데이터 변경을 커미트하는 데 동의합니다. 통합기는 두 서브시스템의 취소 불가능한 의사결정에 대한 커미트 시점을 해당 로그에 기록하여 변경을 수행합니다.

   이제 통합기는 실제 커미트에 해당하는 처리의 2단계를 시작합니다.
10. 통합기는 해당하는 2단계를 시작하도록 IBM MQ에 알립니다.

11. IBM MQ는 2단계의 시작을 기록합니다.
12. 2단계가 성공적으로 완료되고 이제 이는 IBM MQ의 새 일관성 지점이 됩니다. 그런 다음 IBM MQ는 해당하는 2단계 처리가 완료되었음을 통합기에 알립니다.
13. 통합기는 해당하는 2단계 처리를 완료합니다. 두 서브시스템이 제어하는 데이터는 이제 일관되며 다른 애플리케이션에서 사용 가능합니다.

비정상 종료 이후 일관성이 유지보수되는 방법

비정상 종료 이후 큐 관리자가 재시작되면 종료 시점에서 활성 상태였던 복구 단위를 커미트 또는 백아웃할 것인지 여부를 결정해야 합니다. 일부 복구 단위에 대해 IBM MQ는 의사결정을 내릴 수 있도록 충분한 정보를 보유합니다. 다른 경우 그렇지 않으며, 연결이 재설정될 때 통합기에서 정보를 가져와야 합니다.

228 페이지의 그림 77에서는 2단계 내의 4개 기간(a, b, c, d)을 표시합니다. 복구 단위의 상태는 종료가 발생한 기간에 따라 달라집니다. 상태는 다음 중 하나입니다.

인플라이트
큐 관리자는 1단계가 종료되기 전에 종료됩니다(기간 a 또는 b). 재시작 중에 IBM MQ는 업데이트를 백아웃합니다.

인다우트(in-doubt)
큐 관리자는 1단계 종료 후, 2단계 시작 전에 종료됩니다(기간 c). 통합기만 커미트 전 또는 커미트 후에 발생했는지 여부(지점 9)를 확인합니다. 이전에 발생된 경우에는 IBM MQ가 해당 변경을 백아웃하고, 이후에 발생된 경우에는 IBM MQ가 해당 변경을 수행하고 이를 커미트해야 합니다. 재시작 시에 IBM MQ는 이 복구 단위를 처리하기 전에 통합기에서 정보를 기다립니다.

커미트 시
큐 관리자는 해당 2단계 처리가 시작된 후에 종료됩니다(기간 d). 커미트된 변경이 수행됩니다.

백아웃 시
큐 관리자는 복구 단위 백아웃이 시작된 후에 종료되지만, 재시작 중에 프로세스가 완료되기 이전에(그림에 표시되지 않음) IBM MQ는 계속해서 변경을 백아웃합니다.

z/OS IBM MQ for z/OS에서 종료 중에 발생하는 사항

큐 관리자는 STOP QMGR 명령에 대한 응답으로 정상적으로 종료됩니다. 다른 이유로 큐 관리자가 중지된 경우 해당 종료는 비정상적인 조건입니다.

참고로, 큐 관리자가 종료하는 동안 IBM MQ는 내부적으로 다음 명령을 실행합니다.

```
DISPLAY CONN(*) TYPE(CONN) ALL WHERE (APPLTYPE NE SYSTEMAL)
```

이에 따라 큐 관리자가 종료를 완료하지 못하도록 하는 스레드를 인식할 수 있습니다.

SYSTEMAL은 SYSTEM 또는 CHINIT 중 하나의 APPLTYPES와 일치합니다. 따라서 SYSTEMAL과 일치하지 않는 애플리케이션 유형을 필터링하는 DISPLAY CONN 명령은 정상 종료에 방해가 되는 스레드에 대한 joblog 정보로 리턴됩니다.

정상 종료

정상 종료 시에 IBM MQ는 정해진 순서대로 모든 활동을 중지합니다. 일시정지, 강제 실행 또는 재시작 모드를 사용하여 IBM MQ를 중지할 수 있습니다. 효과는 229 페이지의 표 23에 제공됩니다.

표 *23. QUIESCE, FORCE* 및 *RESTART*를 사용하여 종료

스레드 유형	QUIESCE	FORCE	RESTART
활성 스레드	완료하려면 실행	백아웃	백아웃
새 스레드	시작할 수 있음	허용되지 않음	허용되지 않음
새 연결	허용되지 않음	허용되지 않음	허용되지 않음

애플리케이션이 계속 연결된 동안 종료가 발생하면 배치 애플리케이션에 이를 알립니다.

CICS에서 현재 스레드는 복구 단위 종료 시에만 실행됩니다. CICS에서 큐 관리자를 일시정지 모드로 중지하면 CICS 어댑터가 중지되므로 활성 태스크가 둘 이상의 복구 단위를 포함하는 경우 태스크를 실행하여 완료하지 않아도 됩니다.

강제 실행 또는 재시작 모드로 큐 관리자를 중지하면 새 스레드는 할당되지 않으며 연결된 스레드의 작업은 롤백됩니다. 이 모드를 사용하면 커미트 처리 단계 사이에 있는 스레드에 대한 복구 인다우트 단위를 작성할 수 있습니다. 이는 IBM MQ가 CICS, IMS 또는 RRS 서브시스템을 제어하며 다시 연결되면 해결됩니다.

큐 관리자를 중지하면 어떤 모드에서든 단계는 다음과 같습니다.

1. 연결이 종료됩니다.
2. IBM MQ가 명령 승인을 중지합니다.
3. IBM MQ는 페이지 세트에 대한 미해결 업데이트가 완료되었는지 확인합니다.
4. DISPLAY USAGE 명령은 IBM MQ에서 내부적으로 실행되어 재시작 RBA가 z/OS 콘솔 로그에 기록됩니다.
5. 시스템 종료 체크포인트가 생성되고 BSDS가 업데이트됩니다.

일시정지 모드를 지정하는 종료는 복구 인다우트 단위에 영향을 주지 않습니다. 인다우트(in doubt) 상태의 단위는 인다우트(in doubt)로 남습니다.

비정상 종료

비정상 종료는 데이터를 일관되지 않은 상태로 둘 수 있습니다. 예를 들어 다음과 같습니다.

- 일관성 지점에 도달하기 전에 복구 단위가 인터럽트됩니다.
- 커미트된 데이터가 페이지 세트에 기록되지 않습니다.
- 커미트되지 않은 데이터가 페이지 세트에 기록됩니다.
- 애플리케이션 프로그램이 커미트 프로세스의 1단계 및 2단계 사이에서 인터럽트되며 복구 단위가 인다우트(in doubt) 상태로 남습니다.

IBM MQ는 재시작 및 복구 중에 비정상 종료에서 발생하는 데이터 불일치를 해결합니다.

z/OS IBM MQ for z/OS에서 재시작 및 복구 중에 발생하는 사항

IBM MQ는 해당 복구 로그 및 부트스트랩 데이터 세트(BSDS)를 사용하여 재시작할 때 복구할 항목을 판별합니다. BSDS는 로그에서 최신 IBM MQ 체크포인트의 위치와 활성 및 아카이브 로그 데이터 세트를 식별합니다.

재시작 및 복구에 대한 소개

IBM MQ를 초기화한 후에 다음과 같이 큐 관리자 재시작 프로세스가 수행됩니다.

- 로그 초기화
- 현재 상태 다시 빌드
- 이전 로그 복구
- 이후 로그 복구
- 큐 색인 다시 빌드

복구를 완료한 경우:

- 커미트된 변경이 데이터에 반영됩니다.
- 인다우트(in-doubt) 활동이 데이터에 반영됩니다. 그러나 데이터는 잠기고 IBM MQ에서 인다우트 의사결정을 인식하고 이에 대한 조치를 수행할 때까지 사용할 수 없습니다.
- 인터럽트된 작동 시 및 중단 시 변경은 큐에서 제거됩니다. 메시지는 일관되며 이를 사용할 수 있습니다.
- 새 체크포인트가 생성됩니다.
- 지속 메시지를 포함하는 색인화된 큐에서 새 색인이 빌드됩니다(232 페이지의 『큐 색인 다시 빌드』에서 설명함).

이중 BSDS가 사용 중인 경우, IBM MQ는 BSDS에서 시간소인의 일관성을 검사합니다.

- BSDS의 두 사본이 모두 최신인 경우 IBM MQ는 두 시간소인이 동일한지 여부를 테스트합니다. 동일하지 않은 경우 IBM MQ는 메시지 CSQJ120E를 발행하고 종료됩니다. BSDS의 두 사본이 별도의 DASD 볼륨에서 유지보수되고 큐 관리자가 중지된 동안 볼륨 중 하나가 복원된 경우 이 상황이 발생할 수 있습니다. IBM MQ는 재시작 시 상황을 감지합니다.
- BSDS 중 사본 하나가 할당 취소되고 단일 BSDS에서 로깅이 계속되면 문제점이 발생할 수 있습니다. BSDS의 두 사본이 단일 볼륨에서 유지보수되고 볼륨이 복원되거나 두 BSDS 사본이 별도로 복원되는 경우, IBM MQ에서 복원을 감지하지 못할 수도 있습니다. 이 경우 BSDS에 기록되지 않은 로그 레코드는 시스템에 알려지지 않습니다.

애플리케이션이 연결을 요청한 후에 재시작이 수행되어도 배치 애플리케이션에 알리지 않습니다.

복구에 필요한 로그 범위 이해

재시작 중에 읽어야 하는 로그 데이터 범위는 다음과 같은 많은 요소에 종속됩니다.

- 비정상 종료 시 일반적으로 시스템에 불완전한 많은 작업 단위가 있습니다. 앞서 설명한 대로, 재시작 처리가 수행되면 시스템은 일관된 상태가 되고, 인플라이트 작업 단위를 백아웃하거나 인다우트(in-doubt) 작업 단위에서 잠금을 복구할 수 있습니다. 작업 단위 복구에서는 인플라이트, 백아웃 시 및 인다우트(in-doubt) 상태의 작업 단위에 대한 작업 단위 로그 레코드가 모두 사용 가능해야 합니다. IBM MQ는 작업 단위 복구가 더 작은 범위의 많은 로그 데이터를 사용하여 수행될 수 있도록 오래된 작업 단위를 '전환'합니다.
- 비정상 종료 시 일반적으로 버퍼 풀 캐시에만 보유되는 많은 지속 업데이트가 있습니다. 이들은 아직 디스크에 기록되지는 않습니다. 이러한 변경은 로그에서 읽어야 하며 페이지 세트에 보유된 데이터에 재적용되어야 합니다. 체크포인트에서 페이지 세트 복구 RAB는 일관된 상태로 페이지 세트를 업데이트하는 데 필요한 최하위 로그 RBA를 설명합니다.
- 오래된 페이지 세트가 시스템에 도입된 경우(예: 매체 장애에서 복구하도록 페이지 세트 백업이 도입된 경우) 모든 변경은 백업이 수행된 시점에서 로그로부터 읽어야 합니다. 이러한 변경은 복구할 페이지 세트에 보유된 데이터에 재적용됩니다. 페이지 세트의 페이지 0에 보유된 페이지 세트 복구 RBA는 페이지 세트의 매체 복원에 필요한 최하위 로그 RBA를 설명합니다.
- 공유 큐에서 지속 메시지를 사용하는 경우 지속 메시지를 보유하는 CFSTRUCT를 복구하는 데 로그 데이터 범위가 필요합니다. CFSTRUCT 복구를 수행하는 데 필요한 가장 이른 로그 데이터는 이전 CFSTRUCT BACKUP의 시점 근처의 항목입니다.

정상 실행 중에 DISPLAY USAGE TYPE(DATASET) 명령을 사용하여 이러한 요소와 연관된 복구 로그 범위를 볼 수 있습니다(물론 오래된 페이지 세트를 다시 도입하기 때문에 정보를 제공할 수 없음). 비정상 종료 시 큐 관리자 재시작이 연기되는 문제를 방지하려면 DISPLAY USAGE TYPE(DATASET)에서 값 출력을 정기적으로 모니터하십시오.

또한 큐 관리자는 이러한 요소와 관련된 정보 메시지를 발행합니다.

- CSQJ160I 및 CSQJ161I은 장기 실행 작업 단위를 경고합니다.
- CSQR026I 및 CSQR027I에서는 이러한 장기 실행 작업 단위가 성공적으로 전환되었는지 여부에 대한 정보를 제공합니다.
- CSQE040I 및 CSQE041E는 구조 백업이 오래되어 결과적으로 RECOVER CFSTRUCT 조작 시간이 오래 걸림을 경고합니다.

장기 실행 작업 단위가 있는 애플리케이션 판별

장기 실행 작업 단위를 포함하는 애플리케이션을 판별할 수 있습니다. 이를 수행하려면 DISPLAY CONN 명령을 사용하십시오.

DISPLAY CONN 명령은 큐 관리자에 연결된 모든 애플리케이션에 대한 연결 정보와 현재 장기 실행 작업 단위를 보유한 애플리케이션을 판별하는 데 도움이 되는 추가 정보를 함께 리턴합니다. DISPLAY CONN 명령에서 리턴되는 정보는 DISPLAY QSTATUS 명령에서 리턴된 정보와 유사하지만, 기본적인 차이는 특정 오브젝트에 연관된 연결의 세부사항 대신 DISPLAY CONN은 오브젝트에 대한 정보와 특정 연결에 대한 트랜잭션 정보를 표시하는 것입니다.

연결된 각 애플리케이션의 경우 DISPLAY CONN 명령은 다음 정보를 리턴합니다.

- 연결 ID 및 PID를 포함하는 기본 정보.
- 트랜잭션이 작성된 시간 및 날짜(즉, 첫 번째 MQGET/PUT가 동기점 아래에서 작성된 시점) 및 처음에 트랜잭션이 로그에 기록한 시점을 포함하여 해당 연결에 대한 트랜잭션 정보.
- 장기 실행 작업 단위를 계속 보유하는 애플리케이션을 표시하는 로그 시간 정보.
- 연결이 현재 열려 있는 모든 오브젝트의 목록. 각 오브젝트에 대한 자세한 내용은 키처럼 사용되는 연결 ID와 함께, 별도의 메시지로 리턴됩니다. 큐 및 큐 관리자와 같은 서로 다른 유형의 오브젝트가 있으므로 오브젝트에 표시된 정보는 해당 오브젝트 유형에 특정합니다.

큐 색인 다시 빌드

메시지가 순차적으로 검색되지 않은 큐에서 MQGET 조작의 속도를 늘리려는 경우, IBM MQ에서 해당 큐에 있는 모든 메시지의 그룹 ID 또는 메시지나 상관 ID의 색인을 유지보수하도록 지정할 수 있습니다.

큐 관리자가 재시작되면 각 큐에 대해 이러한 색인이 다시 빌드됩니다. 이는 지속 메시지에만 적용되며 비지속 메시지는 재시작 시 삭제됩니다. 색인화된 큐가 많은 양의 지속 메시지를 포함하는 경우 큐 관리자를 재시작하는 데 걸리는 시간이 늘어납니다.

CSQ6SYSP 매크로의 QINDXBLD 매개변수를 사용하여 큐 관리자 시동에 비동기적으로 색인을 다시 빌드하도록 선택할 수 있습니다. QINDXBLD=NOWAIT를 설정하는 경우, 색인 다시 빌드를 기다리지 않고 IBM MQ가 재시작됩니다.

z/OS 복구 인다우트 단위를 해석하는 방법

IBM MQ가 다른 자원 관리자에 대한 연결이 끊길 경우, 일반적으로 재시작 시에 일관되지 않은 모든 오브젝트를 복구하려고 시도합니다.

IBM MQ가 CICS, IMS또는 RRS에 대한 연결이 끊길 경우, 일반적으로 재시작 시에 일관되지 않은 모든 오브젝트를 복구하려고 시도합니다. 복구 인다우트 단위를 해석하는 데 필요한 정보는 통합 시스템에서 파생되어야 합니다. 다음 절에서는 서로 다른 환경에서 해석 프로세스를 설명합니다.

- 인다우트(in-doubt) 복구 단위가 CICS에서 해석되는 방법
- 인다우트(in-doubt) 복구 단위가 IMS에서 해석되는 방법
- 인다우트(in-doubt) 복구 단위가 RRS에서 해석되는 방법
- GROUP 복구 단위 속성 지정을 포함하는 인다우트(in-doubt) 복구 단위를 해석하는 방법

CICS에서 복구 인다우트 단위를 해석하는 방법

어떤 경우에는 CICS가 복구 인다우트 단위를 해석하는 데 IBM MQ 프로세스를 실행할 수 없습니다. 이 경우 IBM MQ는 다음 메시지 중 하나를 송신합니다.

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E

이 뒤에 메시지 CSQC408I가 나옵니다.

이러한 메시지의 의미에 대한 자세한 내용은 IBM MQ for z/OS 메시지, 완료 및 이유 코드 매뉴얼을 참조하십시오.

인다우트 단위의 해석은 CICS 자원에 영향을 주지 않습니다. CICS는 복구 조정의 제어 하에 있으며, 재시작 시에 커미트 시작 위치를 표시하는 로그 레코드가 있는지 여부에 따라 단위를 자동으로 커미트하거나 백아웃합니다. 인다우트 (in-doubt) 오브젝트의 존재는 IBM MQ 가 다시 연결되는 동안 CICS 자원을 잠그지 않습니다.

CICS 어댑터의 기능 중 하나는 CICS와 IBM MQ 사이에서 데이터 동기화를 유지하는 것입니다. CICS에 연결되어 있는 동안 큐 관리자가 이상 종료되는 경우, CICS 는 IBM MQ 가 인식하지 못하는 상태에서 작업을 커미트하거나 백아웃할 수 있습니다. 큐 관리자가 재시작되면 해당 작업은 인다우트*(in doubt)* 상태가 됩니다.

IBM MQ 는 CICS 에 대한 연결이 다시 시작되거나 다시 연결될 때까지 이러한 인다우트 (in-doubt) 복구 단위를 해석할 수 없습니다 (즉, IBM MQ 자원에 대한 변경사항을 커미트하거나 백아웃).

복구 인다우트 단위를 해석하는 프로세스는 CICS 어댑터의 시동 시 시작됩니다. 어댑터는 복구 인다우트 단위의 목록을 요청하는 경우 시작됩니다. 그런 다음 다음과 같습니다.

- 어댑터는 IBM MQ에서 이 연결 ID에 대한 복구 인다우트 단위의 목록을 수신하고, 해석을 위해 CICS에 전달합니다.
- CICS는 이 목록의 입력 항목과 고유한 로그의 입력 항목을 비교합니다. CICS는 고유한 목록에서 각 복구 인다우트 단위에 대해 취하는 조치를 판별합니다.

해석된 모든 단위에 대해 IBM MQ는 필요한 경우 큐를 업데이트하고 해당 잠금을 해제합니다. 해석되지 않은 단위는 재시작 이후에 남아 있을 수 있습니다. 관리IBM MQ for z/OS에서 설명한 방법으로 이들을 해석하십시오.

IMS에서 복구 인다우트 단위를 해석하는 방법

IMS에서 복구 인다우트 단위 해석은 DL/I 자원에 영향을 주지 않습니다. IMS는 복구 협업의 제어 하에 있으며, 재시작 시에 불완전한 DL/I 작업을 자동으로 커미트하거나 백아웃합니다. 온라인 리젼(빠른 경로 아님)을 커미트하거나 백아웃하는 의사결정은 IMS 로그 레코드 유형 X'3730' 및 X'3801'이 있는지에 따라 달라집니다. 복구 인다우트 단위의 존재는 IBM MQ가 연결될 때까지 DL/I 레코드가 잠김을 의미하지는 않습니다.

큐 관리자 재시작 중에 IBM MQ는 복구 인다우트 단위의 목록을 작성합니다. IMS는 여분의 복구 입력 항목(RRE) 목록을 빌드합니다. RRE는 모든 입력 항목이 해석될 때까지 IMS 체크포인트에서 로그됩니다.

IMS 리젼을 IBM MQ에 다시 연결하는 동안 IMS는 IBM MQ에 IBM MQ에서 인다우트(in doubt)로 표시되는 작업 단위를 커미트 또는 백아웃할 것인지 여부를 표시합니다.

인다우트 단위가 해석되는 경우:

1. IBM MQ가 커미트를 위해 입력 항목을 표시했음을 인식하고 IMS가 이를 백아웃하려고 표시한 경우, IBM MQ는 CSQQ010E 메시지를 발행합니다. IBM MQ는 IBM MQ와 IMS 사이에서 이러한 유형의 모든 불일치에 대해 이 메시지를 발행합니다.
2. IBM MQ에 나머지 인다우트 단위가 있는 경우 어댑터는 메시지 CSQQ008I를 발행합니다.

해석된 모든 단위에 대해 IBM MQ는 필요한 경우 큐를 업데이트하고 해당 잠금을 해제합니다.

IBM MQ는 해석되지 않은 인다우트 작업에 대한 잠금을 유지보수합니다. 이는 중요한 잠금이 보유 중인 경우 시스템에서 백로그를 생성시킬 수 있습니다. 연결은 활성 상태로 있으므로 IMS RRE를 해석할 수 있습니다. 관리 IBM MQ for z/OS에서 설명한 방법으로 인다우트(in-doubt) 스레드를 복구하십시오.

모든 인다우트 작업은 IMS 콜드 스타트와 같이 소프트웨어 또는 조작 문제점이 없는 경우에 한하여 해석되어야 합니다. IMS 제어 리젼에 의한 인다우트 해석은 다음 두 가지 상황에서 발생합니다.

1. IBM MQ에 대한 연결 시작 시 해석이 동기적으로 수행되는 동안.
2. 프로그램이 이상종료된 경우 해석이 비동기적으로 수행되는 동안.

RRS에서 복구 인다우트 단위를 해석하는 방법

RRS 어댑터의 기능 중 하나는 IBM MQ와 기타 RRS 참가 자원 관리자 사이에서 데이터 동기화를 유지하는 것입니다. IBM MQ가 커미트 중 1단계를 완료하고 RRS(커미트 통합기)에서 의사결정을 대기하는 동안 실패가 발생하면 복구 단위는 인다우트 상태가 됩니다.

IBM MQ 사이에서 통신이 재설정되면 RRS는 커미트의 시작을 표시하는 로그 레코드가 있는지 여부에 따라 각 복구 단위를 자동으로 커미트하거나 백아웃합니다. IBM MQ는 RRS에 대한 연결이 재시작되거나 이에 다시 연결될 때까지 이러한 복구 인다우트 단위를 해석할 수 없습니다(즉, IBM MQ 자원에서 수행된 변경을 커미트 또는 백아웃함).

일부 상황에서 RRS는 복구 인다우트 단위를 해석할 수 없습니다. 이 경우 IBM MQ는 다음 메시지 중 하나를 z/OS 콘솔에 송신합니다.

- CSQ3011I

- CSQ3013I
- CSQ3014I
- CSQ3016I

이러한 메시지의 의미에 대한 자세한 내용은 IBM MQ for z/OS 메시지, 완료 및 이유 코드 매뉴얼을 참조하십시오.

해석된 모든 단위에 대해 IBM MQ는 필요한 경우 큐를 업데이트하고 해당 잠금을 해제합니다. 해석되지 않은 복구 단위는 재시작 이후에 남아 있을 수 있습니다. 관리IBM MQ for z/OS에서 설명한 방법으로 이들을 해석하십시오.

GROUP 복구 단위 속성 지정을 포함하는 인다우트 복구 단위를 해석하는 방법

GROUP 복구 단위 속성 지정을 보유한 인다우트(in-doubt) 트랜잭션은 GROUPUR 큐 관리자 속성이 사용 가능한 큐 공유 그룹(QSG)에 있는 모든 큐 관리자에 의해 트랜잭션 통합기에서 해석할 수 있습니다. 트랜잭션 통합기가 이에 다시 연결되면 일반적으로 미해결 인다우트(in-doubt) 트랜잭션 목록을 요청한 후 해당 로그의 정보를 사용하여 이들을 해석합니다.

GROUP 복구 단위 속성 지정에 연결된 트랜잭션 통합기가 인다우트(in-doubt) 트랜잭션 목록을 요청하면 리턴된 목록은 큐 공유 그룹 전체에 존재하는 GROUP 복구 단위 속성 지정을 포함하는 모든 인다우트(in-doubt) 트랜잭션을 포함합니다. 이 목록은 해당 인다우트(in-doubt) 트랜잭션이 시작된 큐 관리자에 종속되지 않습니다. 요청과 같은 큐 관리자 처리는 SYSTEM.QSG.UR.RESOLUTION.QUEUE를 사용하여 큐 공유 그룹에서 모든 다른 활성 큐 관리자와 통신하여 목록을 컴파일합니다. 그런 다음 큐 관리자는 마지막 체크포인트에서 비활성 큐 관리자의 로그를 읽어 이들이 활성 상태로 보고한 추가 인다우트(in-doubt) 트랜잭션을 식별합니다.

트랜잭션 통합기가 인다우트(in-doubt) 트랜잭션의 해석을 요청하면 연결된 큐 관리자가 트랜잭션이 자체에서 생성되었는지 여부를 식별하고 이 경우 QMGR 복구 단위 속성 지정을 포함하는 트랜잭션과 같은 방식으로 이를 해석합니다. 트랜잭션이 QSG의 다른 활성 큐 관리자에서 생성된 경우 해석을 완료하는 요청은 SYSTEM.QSG.UR.RESOLUTION.QUEUE를 사용하여 해당 큐 관리자로 라우트됩니다. 트랜잭션이 QSG의 비활성 큐 관리자에서 생성된 경우 모든 공유 큐 작업은 즉시 해석되며 나머지 개인 큐 작업을 해석하는 요청은 SYSTEM.QSG.UR.RESOLUTION.QUEUE에 배치됩니다. 비활성 큐 관리자는 새 작업을 승인하기 전에 시동 시 이 요청을 처리합니다. 이 시나리오에서 원래 큐 관리자의 로그는 재시작되고 요청을 처리할 때까지 복구 단위가 인다우트(in doubt) 상태임을 계속 반영합니다.

z/OS 공유 큐 복구

이 토픽을 사용하여 IBM MQ 복구와 큐 공유 그룹 환경에서 여러 컴포넌트의 복원력을 이해하십시오.

- 234 페이지의 『트랜잭션 복구』
- 235 페이지의 『피어 복구』
- 235 페이지의 『공유 큐 정의』
- 235 페이지의 『로그 기록』
- 235 페이지의 『커플링 기능 및 구조 실패』
- 236 페이지의 『구조 실패 시나리오』
- 237 페이지의 『커플링 기능 연결 실패에 대한 복원력』
- 237 페이지의 『커플링 기능 연결 실패에 대한 복원력 관리』
- 239 페이지의 『조작 작동』

트랜잭션 복구

애플리케이션이 MQBACK 호출을 발행하거나 비정상적으로 종료할 경우(예: EXEC CICS ROLLBACK 또는 IMS 이상종료 때문에) 큐 관리자에 저장된 스레드 레벨 정보는 인플라이트 작업 단위를 롤백하도록 보장합니다. 공유 큐의 동기점 내 MQPUT 및 MQGET 조작은 비공유 큐에 대한 업데이트와 동일한 방식으로 롤백됩니다.

피어 복구

큐 관리자가 실패할 경우 관리자는 현재 연결되어 있는 커플링 기능 구조와의 연결을 비정상적으로 끊습니다. z/OS 인스턴스와 커플링 기능 사이의 연결이 실패할 경우(예: 실제 링크 실패, 커플링 기능 또는 파티션의 전원 차단 등) 이는 큐 관리자와 관련 커플링 기능 구조 사이 연결의 비정상 종료로도 간주됩니다. 해당 구조에 연결된 상태로 남아 있는 동일한 큐 공유 그룹의 다른 큐 관리자는 비정상적인 연결 끊기와 해당 구조에서 실패한 큐 관리자에 대해 피어 복구를 시작하려는 모든 시도를 감지합니다. 이러한 큐 관리자 중 하나만 피어 복구를 성공적으로 시작합니다. 그러나 나머지 모든 큐 관리자는 실패한 큐 관리자가 소유한 복구 작업 단위에서 협업합니다.

구조에 연결된 피어가 없을 때 큐 관리자가 실패한 경우 다른 큐 관리자가 해당 구조에 연결하거나 실패한 큐 관리자를 재시작하면 복구가 수행됩니다.

피어 복구(PLR(Peer Level Recovery)이라고 함)는 구조 대 구조 기반에서 수행되며 단일 큐 관리자가 동시에 둘 이상의 구조의 복구에 참여할 수 있습니다. 그러나 서로 다른 구조의 복구에 협력하는 피어 세트는 실패 시 서로 다른 구조에 연결된 큐 관리자에 따라 달라질 수 있습니다.

실패한 큐 관리자가 재시작되면 실패 시 연결된 구조에 다시 연결하고 피어 복구로 복구되지 않은 남아 있는 해석되지 않은 작업 단위를 복구합니다.

피어 복구는 여러 단계의 프로세스입니다. 첫 번째 단계 중에 인플라이트 단계 이후로 진행한 작업 단위가 복구됩니다. 이는 커미트 상태인 작업 단위에 대한 메시지를 커미트하고 인다우트(in-doubt) 상태의 작업 단위에 대한 메시지를 잠그는 작업을 포함할 수 있습니다. 두 번째 단계에서는 실패한 큐 관리자에서 활성 상태인 스레드를 포함하는 큐를 검사하고, 인플라이트 작업 단위와 관련된 커미트되지 않은 메시지가 롤백되며, 실패한 큐 관리자의 공유 큐에 있는 활성 핸들에 대한 정보가 재설정됩니다. 즉, IBM MQ가 입력을 위해 다른 활성 큐 관리자가 큐를 열 수 있도록 허용하며 실패한 큐 관리자가 독점적 입력을 위해 공유 큐를 열었음을 알리는 표시기를 재설정합니다.

공유 큐 정의

공유 큐의 속성을 표시하는 큐 오브젝트는 큐 공유 그룹에서 사용하는 공유 Db2 저장소에 보유됩니다. IBM MQ 오브젝트를 보유하는 데 사용되는 Db2 테이블의 백업 및 복구에 적절한 프로시저가 있는지 확인하십시오. 또한 IBM MQ CSQUTIL 유틸리티를 사용하여 큐 관리자로 재생하기 위한 MQSC 명령을 작성하여 Db2에 저장된 공유 큐 및 그룹 정의를 포함하여 IBM MQ 오브젝트를 재정의할 수 있습니다.

로그 기록

공유된 큐의 메시지는 큐 관리자 로그에 로그될 수 있기 때문에 큐 공유 그룹은 지속 메시지를 지원할 수 있습니다.

커플링 기능 및 구조 실패

커플링 기능(CF, coupling facility) 구조에는 실패로 보고될 수 있는 유형이 두 가지 있으며, 이는 구조 실패와 연결성 유실입니다. 데이터 공유에 대한 Sysplex 서비스(XES)는 IBM MQ에 구조 실패 이벤트와 함께 CF 실패 또는 CF 구조 실패를 알립니다. XES가 연결성 유실 이벤트를 작성하면 구조에 문제가 있음을 반드시 표시하지 않아도 됩니다. 이는 구조와 통신할 때 사용할 수 있는 연결이 없기 때문일 수 있습니다. 일부 큐 관리자는 구조에 대한 연결성 유실 이벤트를 수신하지 못할 수도 있습니다. 이는 CF에 대한 연결의 구성에 따라 달라집니다. 연산자 명령(예: VARY PATH OFFLINE 또는 CONFIG CHP OFFLINE)으로 인해 연결성 유실 이벤트가 수신될 수도 있습니다.

IBM MQ에서 사용되는 CF 구조는 시스템 관리 양방향을 사용하도록 구성될 수 있습니다. 즉, 단일 실패인 경우 시스템 관리 장애 복구 처리는 구조의 실패 또는 연결성 유실을 숨기고 큐 관리자는 이 실패에 대한 알림을 받지 않습니다. 양방향 구조 또는 연결의 두 인스턴스에 대한 실패가 있는 경우 큐 관리자는 적절한 이벤트를 수신하고 단순한 구조에 대한 실패 이벤트와 동일한 방식으로 이를 처리합니다. 큐 관리자가 이벤트를 처리하는 방식의 자세한 내용은 시나리오에서 설명됩니다.

CF 또는 구조 실패의 경우와 달리 영향을 받는 애플리케이션 구조에 저장된 비지속 메시지가 유실됩니다. RECOVER CFSTRUCT 명령을 사용하여 지속 메시지를 복구할 수 없습니다. 복구 가능한 애플리케이션 구조에서 장애가 발생하면 이 구조에 대한 추가 애플리케이션 활동은 구조가 복구될 때까지 방지됩니다.

합리적인 시간 안에 CF 구조를 복구할 수 있으려면 BACKUP CFSTRUCT 명령을 사용하여 잦은 백업을 수행하십시오. 큐 공유 그룹에 있는 큐 관리자에서 백업을 수행하거나 모든 백업을 수행하도록 하나의 큐 관리자를 전용으로 설정할 수 있습니다. 정기적으로 수행되도록 백업 수행 프로세스를 자동화하십시오.

각 백업은 백업을 수행하는 큐 관리자의 활성 로그 데이터 세트에 기록됩니다. 공유 큐 Db2 저장소는 백업할 CF 구조의 이름, 백업을 수행하는 큐 관리자 이름, 해당 큐 관리자 로그에서 이 백업의 RBA 범위 및 백업 시간을 기록합니다.

관리 구조는 애플리케이션 구조 실패 시 공유 큐에서 불완전한 작업 단위에 대한 정보를 포함합니다. 따라서 관리 구조가 RECOVER CFSTRUCT 처리 중에 사용 가능해야 합니다. 관리 구조에서 실패하면 RECOVER CFSTRUCT 명령을 실행하려면 큐 공유 그룹의 모든 큐 관리자가 해당 관리 구조를 다시 빌드해야 합니다.

큐 관리자는 해당 관리 구조 입력 항목을 종료 없이 자동으로 다시 빌드합니다. 큐 관리자가 실패 시 실행 중이 아닌 경우, 동일한 레벨 또는 상위 레벨에서 실행 중인 큐 공유 그룹의 다른 큐 관리자에 의해 해당 관리 구조 입력 항목이 다시 빌드될 수 있습니다.

애플리케이션 구조를 복구하려면 복구를 수행하려는 큐 관리자에서 RECOVER CFSTRUCT 명령을 실행하십시오. 단일 CF 구조를 복구하거나 동시에 여러 CF 구조를 복구할 수 있습니다. 큐 공유 그룹에서 큐 관리자를 사용하여 복구할 수 있습니다. 백업을 수행하거나 이전에 실패한 구조에 연결된 큐 관리자일 필요는 없습니다.

RECOVER CFSTRUCT 명령은 Db2 저장소 정보를 통해 배치된 백업을 사용하고(따라서 Db2는 복구가 수행되는 큐 관리자에서 수행할 수 있어야 함), 이를 실패 지점으로 복구합니다.

RECOVER CFSTRUCT 명령은 CF 구조에 맵핑된 공유 큐에 대해 실패 시간 및 백업 시작 사이에서 MQPUT 또는 MQGET을 수행하는 큐 공유 그룹의 모든 큐 관리자에서 로그 레코드를 적용하여 이를 수행합니다. 결과적인 로그 병합에서는 상당한 양의 로그 레코드를 읽어야 할 수도 있습니다. 백업 이후 참여하는 큐 관리자에서 쓴 모든 로그 데이터를 읽어야 하기 때문입니다. 특히 백업 내 큰 메시지가 있는 경우 예를 들어 매시간과 같이 자주 백업을 수행하는 것이 좋습니다.

구조 실패 시나리오

시나리오

CF 구조에서 실패가 보고된 경우 연결된 큐 관리자가 수행하는 조치는 다음에 따라 달라집니다.

- z/OS 의 XES 컴포넌트가 IBM MQ에 보고하는 실패 유형입니다.
- 구조 유형(애플리케이션 또는 관리)
- 큐 관리자 레벨
- IBM MQ CFSTRUCT 오브젝트의 CFLEVEL(2, 3, 4 또는 5. CFCC 마이크로 코드의 CFLEVEL이 아님)
- CFLEVEL(5)에서 IBM MQ CFSTRUCT 오브젝트의 RECAUTO 속성

다음 시나리오에서는 관리 구조에 대해 실패가 보고된 경우 발생하는 상황을 설명합니다.

- 구조 실패 이벤트가 관리 구조에서 수신되는 경우, 구조는 큐 관리자를 종료하지 않고도 자동으로 재할당 및 다시 빌드됩니다. 새 구조의 인스턴스는 큐 관리자가 연결하려고 시도할 때 XES에 의해 할당됩니다.

  큐 관리자가 구조의 새 인스턴스에 연결되면 큐 관리자는 구조로 자체의 입력 항목을 씁니다. 이 처리는 큐 관리자에서 수행되며 XES 빌드 처리의 일부가 아닙니다.

  큐 관리자가 실패 시 실행 중이 아니거나 해당 관리 구조 파트의 복구가 완료되기 전에 종료되는 경우, 큐 공유 그룹의 다른 큐 관리자에 의해 해당 관리 구조 입력 항목이 다시 빌드될 수 있습니다.

  동일한 레벨 또는 상위 레벨에서 실행 중인 다른 큐 관리자에 의해서만 큐 관리자의 관리 구조 입력 항목이 다시 빌드될 수 있습니다. 큐 공유 그룹의 다른 큐 관리자에 의해 큐 관리자의 관리 구조 입력 항목이 다시 빌드될 수 없는 경우, 해당 구조 파트의 다시 빌드를 완료할 수 있도록 큐 관리자를 재시작하십시오.

  모든 큐 관리자의 관리 구조 입력 항목이 다시 빌드될 때까지 특정 조치가 일시중단됩니다. 일시중단된 조치로는 다음이 포함됩니다.

  - 공유 큐 열기 및 닫기.
  - 복구 단위 커미트 또는 백아웃.
  - 큐 관리자에 연결되거나 연결이 끊어진 직렬화된 애플리케이션.
  - 애플리케이션 구조의 백업 또는 복구.

이미 큐 관리자에 연결된 직렬화된 애플리케이션은 처리를 계속할 수 있습니다. MQCNO_SERIALIZE_CONN_TAG_QSG 또는 MQCNO_RESTRICT_CONN_TAG_QSG 매개변수에 연결하려고 하는 직렬화된 애플리케이션은 MQRC_CONN_TAG_NOT_USABLE 리턴 코드를 수신합니다.

큐 관리자의 관리 구조 입력 항목이 다시 빌드되면 일시중단된 조치가 계속됩니다.

다음 시나리오에서는 애플리케이션 구조에 대해 실패가 보고된 경우 발생하는 상황을 설명합니다.

- 애플리케이션 구조에서 구조 실패 이벤트가 수신되고 CFLEVEL이 1 또는 2이면 큐 관리자가 종료됩니다. 큐 관리자를 재시작하십시오. 구조에 다시 연결하려는 첫 번째 큐 관리자로 인해 XES에서 구조의 새 인스턴스를 할당합니다.
- 애플리케이션 구조에서 구조 실패 이벤트가 수신되고 CFLEVEL이 3, 4 또는 5이면 구조에 연결된 큐 관리자는 계속 실행됩니다. 실패한 구조의 큐를 사용하지 않는 애플리케이션은 정상 처리를 계속할 수 있습니다.

  그러나 실패한 구조의 큐에 조작을 시도하는 애플리케이션은 구조가 다시 빌드를 완료할 때까지, 즉 애플리케이션이 큐를 다시 열 수 있는 시점까지 MQRC_CF_STRUC_FAILED 오류를 수신합니다.

  RECAUTO(YES)로 정의된 CFLEVEL(5) 애플리케이션 구조에 대해 구조 다시 빌드가 자동으로 시작됩니다. 그렇지 않으면 RECOVER CFSTRUCT 명령이 실행될 때 구조가 다시 빌드됩니다.

커플링 기능 연결 실패에 대한 복원력

커플링 기능 연결 실패에 대한 복원력의 개념

커플링 기능 연결 실패에 대한 복원력이란 큐 공유 그룹에서 종료되지 않고 커플링 기능 구조에 대한 연결성 유실을 허용하는 큐 관리자의 능력을 의미합니다. 또한 이 기능은 공유된 큐에 최대한 빠르게 다시 액세스할 수 있도록 연결성이 더 좋은 다른 커플링 기능에 구조를 다시 빌드하려고 시도합니다.

부분적 연결성 유실의 개념

IBM MQ는 부분적 연결성 유실을 sysplex에서 하나 이상의 시스템이 시스템에 의해 액세스되고 있는 구조가 할당되어 있는 커플링 기능과의 연결성을 유실했지만, sysplex 안의 적어도 하나의 시스템이 같은 커플링 기능과의 연결성을 유지하고 있는 상황으로 정의합니다.

완전 연결성 유실의 개념

IBM MQ는 완전 연결성 유실을 sysplex에서 커플링 기능과 그 안에 할당된 구조에 대해 연결성을 가진 시스템이 없는 상황으로 정의합니다.

이 기능을 사용하는 이유

커플링 기능 연결 실패에 대한 복원력은 큐 관리자가 하나 이상의 커플링 기능 구조에 대한 연결성을 유실한 후에도 공유되지 않은 큐가 사용 가능하도록 하여 IBM MQ의 가용성을 향상시킵니다. 또한 커플링 기능 구조에 대한 연결성을 유실한 큐 관리자는 사용 가능한 다른 커플링 기능에 구조를 자동으로 다시 빌드하여 큐 공유 그룹 내에서 공유 큐의 가용성을 향상시킵니다.

이 기능 사용 시 고려사항

종료되지 않고 커플링 기능 구조에 대한 연결성 유실을 허용하는 큐 관리자는 사용 가능한 대체 커플링 기능이 없을 경우 커플링 기능 구조에 어느 정도의 시간 동안 다시 연결하지 못할 수 있습니다. 연결성이 유실된 구조에 정의되어 있는 공유 큐는 구조에 대한 연결성이 복원되어야 사용할 수 있습니다. 이 상황에서 공유된 큐 작업을 수행하기 위해 큐 공유 그룹 멤버에 연결하는 애플리케이션이 액세스해야 하는 공유된 큐가 사용 불가능할 경우가 있습니다. 이런 상황을 피하기 위해서는 커플링 기능 구조에 대한 연결성이 유실되었을 때는 종료되도록 큐 관리자를 설정하는 것을 권장합니다. 이 종료는 애플리케이션이 애플리케이션에 필요한 공유된 큐가 정의되어 있는 커플링 기능 구조에 대한 연결성을 가진 큐 공유 그룹의 다른 멤버에 연결하도록 강제합니다.

커플링 기능 연결 실패에 대한 복원력 관리

기능성 사용 방법

커플링 기능 연결성에 대한 복원력을 사용하려면 다음 단계를 수행해야 합니다.

1. CFRM 커플 데이터 세트가 시스템 관리 다시 빌드를 지원하도록 포맷되었는지 확인하십시오. 이는 큐 관리자가 사용 가능한 커플링 기능에 구조를 다시 작성할 수 있도록 시스템 관리 다시 빌드를 시작할 수 있게 합니다. CFRM 커플 데이터 세트의 형식을 판별하려면 **DISPLAY XCF,COUPLE,TYPE=CFRM** 명령을 사용하십시오. 시스템 관리 다시 빌드를 지원하려면 CFRM 커플 데이터 세트를 다음을 지정하여 포맷해야 합니다.

   ```
   "ITEM NAME(SMREBLD) NUMBER(1)"
   ```

   CFRM 커플 데이터 세트 형식화에 대한 자세한 정보는 z/OS MVS Setting Up a Sysplex 문서를 참조하십시오.
2. 대체 커플링 기능이 사용 가능하며 모든 IBM MQ 커플링 기능 구조에 대한 CFRM 환경 설정 목록에 있는지 확인하십시오. 이는 큐 관리자가 최대한 빠르게 구조에 대한 액세스를 복원할 수 있도록 사용 가능한 대체 커플링 기능에 구조를 다시 빌드할 수 있게 합니다.

   IBM MQ에서 구조를 재할당해야 하는 경우 XCF가 최적의 CF를 선택할 수 있도록 IBM MQ 구조는 CFRM 정책에서 ENFORCEORDER(NO)으로 정의되어야 합니다.

   구조 환경 설정 목록에 대한 자세한 정보는 z/OS MVS Setting Up a Sysplex 문서를 참조하십시오.
3. 연결성 유실을 허용해야 하는 모든 애플리케이션 커플링 기능 구조를 CFLEVEL(5)로 변경하십시오. 이는 연결성 유실을 허용할 수 있는 최소 레벨입니다.
4. **QMGR CFCONLOS** 및 **CFSTRUCT CFCONLOS** 속성에 필요한 값을 판별하고 이에 따라 변경하십시오. **QMGR CFCONLOS** 속성은 관리 구조에 대한 연결성 유실에 대한 허용 여부를 제어하며 **CFSTRUCT CFCONLOS** 속성은 각 애플리케이션 커플링 기능 구조의 연결성 유실 허용 여부를 제어합니다. 이 속성들에 대해 기본 값을 유지할 경우 큐 관리자는 커플링 기능 구조에 대한 연결성을 더 이상 유실되지 않게 합니다.
5. 각 애플리케이션 커플링 기능 구조에 대해 **CFSTRUCT RECAUTO** 속성에 필요한 값을 판별하고 이를 알맞게 변경하십시오. 이 속성은 완전 연결성 유실 후 로그된 데이터를 사용한 커플링 기능 구조의 자동적 복구 여부를 제어합니다. 이 속성에 기본값을 유지할 경우 연결성이 모두 유실된 후 어떤 애플리케이션 구조에 대해서도 자동 복구가 수행되지 않습니다.

시나리오 1 - 관리 구조에 대한 연결성 유실

큐 관리자는 종료하지 않고 관리 구조에 대한 연결 유실을 허용할 수 있습니다.

관리 구조에 대한 연결성 유실을 허용하도록 구성된 큐 관리자가 관리 구조에 대한 연결성을 유실했을 경우 큐 공유 그룹의 모든 멤버들이 관리 구조와의 연결을 끊습니다. 그 후 큐 공유 그룹의 모든 활성 큐 관리자는 관리 구조에 다시 연결을 시도하며, 이 때 이들은 sysplex의 모든 시스템에 대해 최상의 연결성을 가진 커플링 기능으로 다시 할당되어 관리 구조 데이터를 다시 빌드합니다.

참고: 이것이 반드시 활성 큐 관리자를 가진 모든 시스템에 최상의 연결성을 가진 커플링 기능인 것은 아닙니다.

예를 들어 관리 구조에 대한 CFRM 환경 설정 목록에 사용 가능한 커플링 기능이 없거나 하는 이유로 큐 관리자가 관리 구조에 다시 연결할 수 없는 경우, 큐 관리자가 관리 구조에 다시 연결해 관리 구조 데이터를 다시 빌드할 때까지 몇몇 공유된 큐 조작을 사용할 수 없습니다. 이는 시스템에서 알맞은 커플링 기능이 사용 가능해질 때 자동적으로 일어납니다.

커플링 기능에 대한 연결성 부족으로 인한 큐 관리자 시작 중 관리 구조와의 연결 실패나 구조를 할당할 사용 가능한 알맞은 커플링 기능 부족은 허용되지 않습니다. 그 후 큐 공유 그룹의 모든 활성 큐 관리자는 관리 구조에 다시 연결을 시도하며, 이 때 이들은 사용 가능한 다른 커플링 기능이 있을 경우 여기에 다시 할당되어 관리 구조 데이터를 다시 빌드합니다.

시나리오 2 - 애플리케이션 구조에 대한 연결성 유실

큐 관리자를 종료하지 않고 **CFLEVEL(5)** 이상에서 애플리케이션 구조에 대한 연결 유실을 허용할 수 있습니다. 구조에 대한 연결성이 유실될 때 **CFLEVEL(4)** 이하의 애플리케이션 구조 또는 연결성 유실을 허용하도록 구성되지 않은 **CFLEVEL(5)**의 구조에 연결된 큐 관리자는 이상종료되며 그 이유 코드는 00C510AB입니다.

연결성 유실을 허용하도록 구성된 애플리케이션 구조에 대한 연결성이 유실되었을 경우 구조에 대한 연결성을 유실한 모든 큐 관리자가 연결을 끊습니다. 큐 관리자의 후속 작동은 연결성 유실이 부분적인지 전체적인지에 따라 다릅니다.

애플리케이션 구조에 대한 부분적 연결성 유실

연결성 유실이 부분적으로 판별될 경우 구조에 대한 연결성을 유실한 큐 관리자는 연결성이 향상된 다른 커플링 기능으로 구조를 이동시키기 위해 시스템 관리를 다시 빌드하기 시작합니다. 다시 빌드가 완료되면 구조에 있는 지속적 및 비지속 메시지는 다른 커플링 기능으로 복사되며 구조에 있는 큐에 대한 액세스가 복원됩니다. 연결성을 유실하지 않은 큐 관리자는 구조에 연결된 채로 남아 있지만 구조에 액세스하는 조작은 시스템 관리 재빌드 프로세스 동안 지연됩니다.

연결성이 향상된 다른 커플링 기능에 애플리케이션 구조를 다시 빌드할 수 없거나 다른 커플링 기능에 구조가 다시 빌드된 후에도 연결성을 갖지 못한 큐 관리자가 있는 경우 커플링 기능에 대한 연결성이 복원될 때까지 구조에 대한 연결성이 없는 큐 관리자는 구조에 정의된 큐를 사용할 수 없습니다. 큐 관리자는 구조가 사용 가능해지면 자동적으로 다시 연결하며 구조에 정의되어 있는 공유된 큐에 대한 액세스가 복원됩니다.

애플리케이션 구조에 대한 총 연결성 유실

sysplex의 모든 MVS 시스템이 애플리케이션 구조가 할당되어 있는 커플링 기능에 대한 연결성을 유실한 경우 z/OS는 구조에 다시 연결을 시도할 때마다 커플링 기능에서 구조를 할당 해제합니다. 애플리케이션의 공유된 큐 열기 시도나 시스템의 새 커플링 기능 자원 사용 가능 알림과 같은 몇 가지 이유로 큐 관리자가 구조에 다시 연결을 시도하는 경우가 있습니다. 따라서 이 구조에 있는 모든 비지속 메시지는 애플리케이션 구조에 연결성을 모두 유실한 후 잃게 될 가능성이 큽니다.

연결성을 모두 유실한 후 복구 가능한 애플리케이션은 **RECAUTO(YES)**로 정의되어 있을 경우 자동으로 복원됩니다. 복구는 구조를 할당할 대체 커플링 구조가 있을 경우 또는 이러한 커플링 기능이 사용 가능해질 때 거의 즉시 시작됩니다. 구조가 **RECAUTO(YES)**로 정의되지 않은 경우 **RECOVER CFSTRUCT** 명령을 실행하여 복구를 시작할 수 있습니다. 이는 구조의 모든 지속 메시지를 복구하지만, 모든 비지속 메시지는 잃게 됩니다. 이 프로세스는 큐 관리자 로그 읽기를 포함하기 때문에 완료에 시간이 걸릴 수 있으며 따라서 이 시간을 줄이기 위해 구조의 공유된 큐에 대한 액세스가 복원될 때까지 정기적으로 구조 백업을 하는 것을 권장합니다.

큐 관리자는 애플리케이션이 구조에 정의된 공유 큐를 열려고 시도함과 동시에, 또는 새 커플링 기능 자원이 사용 가능하다는 시스템의 알림을 수신하는 동시에 복구 불가능한 애플리케이션 구조에 대한 다시 연결을 시도합니다. 구조를 할당하기에 알맞은 커플링 기능이 사용 가능한 경우 새 구조가 할당되며 구조에 정의된 공유 큐에 대한 액세스가 복원됩니다. 지속 메시지는 복구 불가능한 구조에 정의된 큐에 넣을 수 없기 때문에 공유 큐의 모든 메시지는 잃게 됩니다.

조작 작동

특정 커플링 기능 구조에 대한 연결성 유실을 허용하도록 구성된 IBM WebSphere MQ 7.1 이상의 큐 관리자가 연결성을 유실할 경우 큐 공유 그룹의 멤버는 자동적으로 실패로부터 복원하여 구조에 다시 연결을 시도합니다. 이 활동은 사용 가능할 경우 향상된 연결성의 다른 커플링 기능에 구조를 다시 할당하는 것을 포함할 수 있습니다. 그러나 연결성 유실을 복구하기 위해서는 여전히 운영자 개입이 필요할 수 있습니다.

일반적으로 필요한 운영자 조치는 다음과 같습니다.

1. 연결성 유실을 일으키는 실패의 원인을 해결합니다.
2. IBM MQ 구조가 할당될 수 있는 커플링 기능이 sysplex의 모든 시스템에서 사용 가능한지 확인하십시오.

연결성 유실 이벤트 후 자동적으로 다른 커플링 기능으로 다시 할당된 모든 구조는 큐 공유 그룹의 모든 큐 관리자에 대한 최적 연결성을 갖는 커플링 기능으로 이동시킬 수 있습니다. 필요한 경우 z/OS MVS 시스템 명령 참조에 설명된 대로 시스템 관리 다시 빌드 명령 **SETXCF START,REBUILD** 를 시작하여 이를 수행할 수 있습니다.

애플리케이션 구조에 대한 연결성이 부분적으로 유실된 경우 구조에 대한 연결성을 유실한 큐 관리자는 시스템 관리를 다시 빌드하려고 시도합니다. 이 프로세스는 다른 커플링 기능에 구조를 다시 할당하며, 이는 이 커플링 기능이 현재 구조에 연결되어 있는 모든 활성 큐 관리자에 대해 연결성을 가질 때만 수행됩니다. 따라서 큐 공유 그룹에 있는 대부분의 큐 관리자가 애플리케이션 구조에 대한 연결성을 유실했을 때 아직 기존 구조에 연결되어 있는 큐 관리자들로 인해 다른 커플링 기능에 구조를 다시 빌드하지 못할 경우가 있습니다. 이 상황에서 아직 기존 구조에 연결되어있는 큐 관리자는 구조가 다시 빌드될 수 있도록 종료될 수 있으며 또는 구조를 실패시키기 위

해 **RESET CFSTRUCT ACTION(FAIL)** 명령을 발행할 수도 있습니다. **RECOVER CFSTRUCT** 명령을 발행하여 적용 가능한 구조의 복구를 시작할 수 있습니다.

참고: 구조 실패 및 복구 시에는 구조의 모든 비지속 메시지를 잃게 됩니다.

z/OS IBM MQ for z/OS의 보안 개념

IBM MQ에서 보안의 중요성과 시스템에서 적절한 보안 설정이 되지 않은 경우 의미를 이해하는 데 이 토픽을 사용하십시오.

IBM MQ 자원을 보호해야 하는 이유

IBM MQ는 잠재적으로 중요한 정보 전송을 처리합니다. 보안을 적용하면 자원 IBM MQ에서 소유하고 관리하는 자원이 비인가 액세스로부터 보호됩니다. 이러한 액세스는 정보의 손실 또는 공개로 이어질 수 있습니다.

권한이 없는 사용자 또는 프로세스가 다음 자원을 액세스하거나 변경하지 못하도록 해야 합니다.

- IBM MQ에 연결
- IBM MQ 오브젝트(예: 큐, 프로세스 및 이름 목록)
- IBM MQ 전송 링크(즉, IBM MQ 채널)
- IBM MQ 시스템 제어 명령
- IBM MQ 메시지
- 메시지와 연관된 컨텍스트 정보

필요한 보안을 제공하기 위해 IBM MQ는 z/OS 시스템 인증 기능(SAF)을 사용하여 외부 보안 관리자(ESM)(예: 보안 서버(이전에 RACF임))로 인증 요청을 라우트합니다. IBM MQ에는 고유한 보안 확인이 없습니다. 분산 큐잉 또는 클라이언트가 사용 중인 경우 IBM MQ에서 채널, 인증 레코드, 채널 엑시트, MCAUSER 채널 속성 및 TLS를 제공하는 추가 보안 수단이 필요할 수도 있습니다.

오브젝트에 대한 액세스를 허용하는 의사결정은 ESM에서 수행되며, IBM MQ는 이 의사결정을 따릅니다. ESM에서 의사결정을 내릴 수 없는 경우 IBM MQ는 오브젝트에 대한 액세스를 방지합니다.

IBM MQ 자원을 보호하지 않는 경우 발생하는 상황

보안에 대해 아무것도 수행하지 않은 경우 가장 유력한 영향은 모든 사용자가 모든 자원에 액세스하고 이를 변경할 수 있다는 점입니다. 여기에는 로컬 사용자 뿐만 아니라, 로그온 보안 제어가 일반적으로 z/OS의 경우보다 덜 엄격한 클라이언트 또는 분산 큐잉을 사용하는 원격 시스템의 사용자도 포함됩니다.

보안 검사를 사용 가능하게 하려면 다음을 수행해야 합니다.

- ESM을 설치하고 활성화합니다(예: 보안 서버).
- 보안 서버 이외의 ESM을 사용하는 경우 MQADMIN 클래스를 정의합니다.
- MQADMIN 클래스를 사용으로 설정합니다.

대소문자가 혼합된 자원 이름을 사용하면 엔터프라이즈에 유리한지 여부를 고려해야 합니다. ESM 프로파일에서 대소문자가 혼합된 자원 이름을 사용하는 경우 MXADMIN 클래스를 정의 및 활성화해야 합니다.

z/OS 데이터 세트 암호화

데이터 세트 암호화(DSE)는 z/OS 데이터 세트를 암호화하는 기능을 제공하므로 포함된 데이터는 특정 권한이 부여된 사용자 ID만 보거나 수정할 수 있습니다. 이는 파일 시스템에 있는 정지 상태인 데이터에 대한 암호화를 제공하며 데이터 세트 자체를 관리하기 위해 정당한 비즈니스 요구사항 및 권한이 있는 사용자에게 민감한 정보를 의도치 않게 노출하는 것을 방지합니다.

IBM MQ for z/OS 9.1.4 이전에는 IBM MQ for z/OS에서 IBM MQ 메시지에 대한 기본 지속성 메커니즘을 제공하는 활성 로그, 페이지 세트 및 공유 메시지 데이터 세트(SMDS)가 포함된 DSE의 사용을 지원하지 않았습니다.

대신 Advanced Message Security는 IBM MQ 메시징에 대한 엔드-투-엔드 암호화 솔루션을 제공하는데, 이는 전체 IBM MQ 네트워크, 전송 중이거나 정지 상태인 데이터 암호화 그리고 IBM MQ가 처리하는 런타임 내부까지도 포함합니다.

IBM MQ 서브시스템에 사용되는 기타 VSAM 및 순차 데이터 세트는 DSE를 사용하여 암호화할 수 있습니다. 예를 들면, 다음과 같습니다.

- 부트스트랩 데이터 세트(BSDS)
- CSQINPx DDNAME을 사용하여 시작 시 시스템 구성(MQSC) 명령 읽기를 보유하는 순차 파일
- 감사를 위해 IBM MQ 로그 데이터의 장기 아카이브에 종종 사용되는 IBM MQ 아카이브 로그

데이터 세트 키 레이블로 정의되는 데이터 클래스를 할당하여 DSE를 사용하여 암호화할 수 있습니다. 자세한 정보는 로그 아카이브 스토리지 계획을 참조하십시오.

IBM MQ for z/OS 9.1.4부터는 IBM MQ for z/OS에서 이전 릴리스에서 제공된 지원은 물론 활성 로그 및 페이지 세트가 포함된 DSE의 사용도 지원됩니다.

IBM MQ for z/OS에서는 공유 메시지 데이터 세트(SMDS)에 대한 DSE의 사용이 지원되지 않습니다.

데이터 세트 암호화를 사용하는 IBM MQ for z/OS의 저장 데이터에 대한 기밀성 섹션을 참조하십시오. 참조하십시오.

관련 개념

보안 개념

채널 인증 레코드

z/OS 에서 IBM MQ 오브젝트에 대해 작업할 수 있는 권한

암호화 보안 프로토콜: TLS

관련 태스크

z/OS에서 보안 설정

링크 레벨 보안과 애플리케이션 레벨 보안 비교

관련 참조

IBM MQ for z/OS에 대한 메시지

z/OS IBM MQ for z/OS의 보안 제어 및 옵션

전체 IBM MQ 서브시스템에서 보안이 켜져 있는지 여부와 보안 검사를 수행하는 레벨(큐 관리자 또는 큐 공유 그룹 레벨)을 지정할 수 있습니다. 또한 API 자원 보안에 대해 검사하는 사용자 ID 수를 제어할 수도 있습니다.

서브시스템 보안

서브시스템 보안은 전체 큐 관리자에서 보안 검사를 수행하는지 여부를 지정하는 제어입니다. 보안 검사가 필요하지 않은 경우(예: 테스트 시스템) 또는 클라이언트 및 채널을 포함하여 IBM MQ에 연결할 수 있는 모든 자원에서 보안 레벨에 만족하는 경우 추가 보안 검사를 수행하지 않도록 큐 관리자 또는 큐 공유 그룹에서 보안 검사를 끌 수 있습니다.

이는 보안을 완전히 끌 수 있고 다른 보안 검사의 수행 여부를 판별하는 유일한 검사입니다. 즉, 큐 관리자 또는 큐 공유 그룹에서 검사를 끄는 경우 다른 IBM MQ 검사는 수행되지 않습니다. 켠 상태로 두면 IBM MQ는 다른 IBM MQ 자원에 대한 보안 요구사항을 검사합니다.

또한 명령과 같은 특정 자원 세트에서 보안을 켜거나 끌 수 있습니다.

큐 관리자 또는 큐 공유 그룹 레벨 검사

큐 관리자 레벨 또는 큐 공유 그룹에서 보안을 구현할 수 있습니다. 큐 공유 그룹 레벨에서 보안을 구현하는 경우 그룹의 모든 큐 관리자는 동일한 프로파일을 공유합니다. 즉, 보안 관리자가 쉽게 작업할 수 있도록 정의 및 유지보수할 프로파일 수가 적음을 의미합니다. 또한 이는 기존 보안 프로파일을 상속하므로 큐 공유 그룹에 새 큐 관리자를 더 쉽게 추가합니다.

설치에 필요한 경우(예: 마이그레이션 도중) 또는 큐 공유 그룹의 한 큐 관리자를 보유하지만, 이때 그룹의 다른 큐 관리자에 대해 다른 레벨의 보안이 필요한 경우 모두의 결합을 구현할 수도 있습니다.

큐 공유 그룹 레벨 보안

큐 공유 그룹 레벨의 보안 검사는 전체 큐 공유 그룹에서 수행됩니다. 이 경우 정의해야 하는 보안 프로파일 수가 더 적으므로 보안 관리를 단순화할 수 있습니다. 특정 자원을 사용할 사용자 ID의 권한 부여는 큐 공유 그룹 레벨에서 처리되며 자원에 액세스하는 데 사용자 ID가 사용하는 큐 관리자와는 독립적입니다.

예를 들어, 사용자 ID SERVER 아래에서 서버 애플리케이션이 실행되고 큐 SERVER.REQUEST에 액세스하려고 하며, sysplex의 각 z/OS 이미지에서 SERVER의 인스턴스를 실행하려고 합니다. 개별적으로 각 큐 관리자에서 SERVER.REQUEST를 열도록 SERVER를 허용하는 대신(큐 관리자 레벨의 보안), 큐 공유 그룹 레벨에서만 액세스를 허용할 수 있습니다.

큐 공유 그룹 레벨의 보안 프로파일을 사용하여 로컬 또는 공유에 상관없이 모든 유형의 자원을 보호할 수 있습니다.

큐 관리자 레벨 보안
큐 관리자 레벨의 보안 프로파일을 사용하여 로컬 또는 공유에 상관없이 모든 유형의 자원을 보호할 수 있습니다.

두 레벨의 결합
큐 관리자 및 큐 공유 그룹 레벨의 보안 모두의 조합을 사용할 수 있습니다.

해당 그룹의 멤버인 특정 큐 관리자에 대한 큐 공유 그룹 레벨 보안 설정을 대체할 수 있습니다. 즉, 개별 큐 관리자에서 그룹의 다른 큐 관리자에서 수행된 항목과는 다른 레벨의 보안 검사를 수행할 수 있음을 의미합니다.

자세한 정보는 큐 공유 그룹 또는 큐 관리자 레벨의 보안을 제어할 프로파일을 참조하십시오.

검사하는 사용자 ID 수 제어

RESLEVEL은 IBM MQ 자원 보안에 대해 검사하는 사용자 ID 수를 제어하는 보안 서버 프로파일입니다. 일반적으로 사용자가 IBM MQ 자원에 액세스하려고 하면 보안 서버가 해당 자원에 대한 액세스가 허용되는지를 확인하기 위해 하나 이상의 관련 사용자 ID를 검사합니다. RESLEVEL 프로파일을 정의하면 0, 1 또는 가능한 경우 2개 중에서 검사하는 사용자 ID 수를 제어할 수 있습니다.

이러한 제어는 연결 기반으로 연결에서 수행되며 연결 수명 동안 지속됩니다.

각 큐 관리자에 대해 하나의 RESLEVEL 프로파일만 존재합니다. 제어는 이 프로파일에 대해 사용자 ID가 보유한 액세스로 구현됩니다.

대소문자 혼용 또는 대문자 IBM MQ RACF 클래스

이제 대소문자가 혼합된 RACF 프로파일 지원을 사용할 수 있습니다. 이를 통해 대소문자가 혼합된 자원 이름을 사용하고 이들을 보호하도록 IBM MQ RACF 프로파일을 정의할 수 있습니다.

다음 중에서 선택할 수 있습니다.

- 이전 릴리스에서와 같이 IBM MQ RACF 클래스만 계속 대소문자를 사용합니다. 또는
- 대소문자가 혼합된 새 IBM MQ RACF 클래스를 사용합니다.

대소문자가 혼합된 RACF 프로파일을 사용하지 않아도 IBM MQ for z/OS 에서 대소문자가 혼합된 자원 이름을 사용할 수 있습니다. 그러나 이러한 자원 이름은 대문자 IBM MQ 클래스의 일반 RACF 프로파일에 의해서만 보호될 수 있습니다. 대소문자가 혼합된 IBM MQ RACF 프로파일 지원을 사용하는 경우 대소문자가 혼합된 IBM MQ 클래스에서 IBM MQ RACF 프로파일을 정의하여 보다 세분화된 레벨의 보호를 제공할 수 있습니다.

z/OS IBM MQ for z/OS에서 보호할 수 있는 자원

큐 관리자가 시작되거나 연산자 명령으로 지시된 경우 IBM MQ for z/OS는 보호할 자원을 판별합니다.

각 개별 큐 관리자에 대해 수행된 보안 검사를 제어할 수 있습니다. 예를 들어 테스트 큐 관리자에서는 구현하지 않고 프로덕션 큐 관리자에서 여러 보안 검사를 구현할 수 있습니다.

연결 보안

연결 보안 검사는 애플리케이션 프로그램이 큐 관리자에 연결하려고 할 때 수행됩니다. 이는 MQCONN 또는 MQCONNX 요청을 발행함으로써, 또는 채널 시작기나 CICS 또는 IMS 어댑터가 연결 요청을 발행할 때 수행됩니다.

큐 관리자 레벨의 보안을 사용하는 경우 특정 큐 관리자에서 연결 보안 검사를 끌 수 있습니다. 그러나 이를 수행하면 모든 사용자가 해당 큐 관리자에 연결할 수 있습니다.

CICS 어댑터의 경우 개별 CICS 터미널 사용자 ID가 아닌 CICS 주소 공간 사용자 ID만 연결 보안 검사에 사용됩니다. IMS 어댑터의 경우 IMS 제어 또는 종속 영역이 IBM MQ에 연결된 경우 IMS 주소 공간 사용자 ID가 검사됩니다. 채널 시작기의 경우 채널 시작기 주소 공간에 사용된 사용자 ID가 검사됩니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 연결 보안 검사를 켜거나 끌 수 있습니다.

명령 보안

명령 보안 검사는 사용자가 IBM MQ for z/OS에서 MQSC및 PCF 명령을 실행할 수 있는 소스에 설명된 소스에서 MQSC 명령을 실행할 때 수행됩니다. 243 페이지의 『명령 자원 보안』 에서 설명한 대로 명령에서 지정한 자원에서 별도의 검사를 수행할 수 있습니다.

명령 검사를 끄면 명령을 실행할 권한이 있는지 확인하기 위해 명령 실행자를 검사하지 않습니다.

MQSC 명령이 콘솔에서 입력된 경우 콘솔에는 z/OS SYS 콘솔 권한 속성이 있어야 합니다. CSQINP1이나 CSQINP2 데이터 세트에서 실행되었거나 큐 관리자에서 내부적으로 실행된 명령은 모든 보안 검사에서 제외되는 반면, CSQINPX의 경우에는 채널 시작기 주소 공간의 사용자 ID를 사용합니다. 정상 데이터 세트 보호를 통해 이러한 데이터 세트를 업데이트할 수 있는 사용자를 제어해야 합니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 명령 보안 검사를 켜거나 끌 수 있습니다.

명령 자원 보안

일부 MQSC 명령(예: 로컬 큐 정의)은 IBM MQ 자원 조작을 포함합니다. 명령 자원 보안이 활성 상태인 경우 자원과 관련된 명령이 실행될 때마다 IBM MQ는 사용자가 해당 자원 정의를 변경할 수 있는지 여부를 확인합니다.

명령 자원 보안을 사용하여 이름 지정 표준을 시행하는 데 도움을 받을 수 있습니다. 예를 들어 지급 관리자는 이름이 "PAYROLL"로 시작하는 큐만 삭제 및 정의할 수 있습니다. 명령 자원 보안이 비활성 상태인 경우 명령으로 조작 중인 자원에서 보안 검사는 수행되지 않습니다. 명령 자원 보안과 명령 보안을 혼동하지 마십시오. 이 둘은 서로 별개의 것입니다.

명령 자원 보안 검사를 꺼도 특히 명령과 관련이 없는 다른 처리 유형에서 수행되는 자원 검사에는 영향을 주지 않습니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 명령 자원 보안 검사를 켜거나 끌 수 있습니다.

채널 보안 고려사항

채널 보안

채널을 사용하는 경우 사용 가능한 보안 기능은 사용할 통신 프로토콜에 따라 다릅니다. TCP를 사용하는 경우 TLS를 사용할 수 있어도 통신 프로토콜에서 제공하는 보안 기능은 없습니다. APPC를 사용하는 경우 확인을 위해 송신 MCA에서 네트워크를 거쳐 목적지 MCA로 사용자 ID 정보를 전달할 수 있습니다.

두 프로토콜의 경우 보안 목적으로 확인할 수 있는 사용자 ID 및 해당 수를 지정할 수 있습니다. 다시 사용자가 사용할 수 있는 선택은 사용하는 프로토콜, 채널을 정의할 때 정의하는 내용 및 채널 시작기에 대해 RESLEVEL 설정에 따라 달라집니다.

사용 가능한 채널 보안 유형에 대한 자세한 정보는 채널 인증 레코드 및 보안 엑시트 개요를 참조하십시오.

관련 참조

244 페이지의 『IBM MQ for z/OS의 API 자원 보안』

MQOPEN 또는 MQPUT1 호출로 애플리케이션이 오브젝트를 여는 경우 자원을 검사합니다. 오브젝트를 여는 데 필요한 액세스는 큐가 열린 경우 지정된 열기 옵션에 따라 달라집니다.

z/OS *IBM MQ for z/OS*의 *API* 자원 보안

MQOPEN 또는 MQPUT1 호출로 애플리케이션이 오브젝트를 여는 경우 자원을 검사합니다. 오브젝트를 여는 데 필요한 액세스는 큐가 열린 경우 지정된 열기 옵션에 따라 달라집니다.

API 자원 보안은 다음 검사로 구분됩니다.

- 큐
- 프로세스
- 이름 목록
- 대체 사용자
- 컨텍스트

큐 관리자 오브젝트를 열거나 스토리지 클래스 오브젝트에 액세스하는 경우 보안 검사는 수행되지 않습니다.

큐

큐 보안 검사는 어떤 큐를 누가 열 수 있는지와 이를 열 때 허용되는 옵션을 제어합니다. 예를 들어 사용자는 큐 PAYROLL.INCREASE.SALARY를 열어 큐에서 메시지를 찾아보도록 할 수 있습니다(MQOO_BROWSE 옵션 사용). 그러나 큐에서 메시지를 제거하는 작업은 허용하지 않습니다(MQOO_INPUT_* 옵션 사용). 큐 검사를 끄는 경우 모든 사용자가 올바른 열기 옵션(즉, MQOPEN 또는 MQPUT1 호출에서 올바른 MQOO_* 옵션)으로 모든 큐를 열 수 있습니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 큐 보안 검사를 켜거나 끌 수 있습니다.

프로세스

프로세스 보안 검사는 사용자가 프로세스 정의 오브젝트를 여는 경우 수행됩니다. 프로세스 검사를 끄는 경우 모든 사용자가 모든 프로세스를 열 수 있습니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 프로세스 보안 검사를 켜거나 끌 수 있습니다.

이름 목록

이름 목록 보안 검사는 사용자가 이름 목록을 여는 경우 수행됩니다. 이름 목록 검사를 끄는 경우 모든 사용자가 모든 이름 목록을 열 수 있습니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 이름 목록 보안 검사를 켜거나 끌 수 있습니다.

대체 사용자

대체 사용자 보안은 한 사용자 ID가 다른 사용자 ID의 권한을 사용하여 IBM MQ 오브젝트를 열 수 있는지 여부를 제어합니다.

예를 들면, 다음과 같습니다.

- 사용자 ID PAYSERV에서 실행 중인 서버 프로그램은 사용자 ID USER1에서 큐에 배치된 큐로부터 요청 메시지를 검색합니다.
- 서버 프로그램이 요청 메시지를 가져오면 요청을 처리하고 요청 메시지에 지정된 응답 대상 큐로 응답을 다시 넣습니다.
- 소유한 사용자 ID(PAYSERV)를 사용하여 응답 대상 큐 열기 권한을 부여하는 대신, 서버는 다른 사용자 ID(이 경우, USER1)를 지정할 수 있습니다. 이 예에서 대체 사용자 보안은 응답 대상 큐를 열 때 사용자 ID PAYSERV가 대체 사용자 ID로 사용자 ID USER1을 지정하도록 허용되었는지 여부를 제어합니다.

대체 사용자 ID는 오브젝트 디스크립터(MQOD)의 *AlternateUserId* 필드에 지정됩니다.

모든 IBM MQ 오브젝트(예: 프로세스 또는 이름 목록)에서 대체 사용자 ID를 사용할 수 있습니다. 다른 자원 관리자(예: CICS 보안 또는 z/OS 데이터 세트 보안용)에서 사용하는 사용자 ID에는 영향을 주지 않습니다.

대체 사용자 보안이 활성 상태가 아닌 경우 모든 사용자가 대체 사용자 ID로 다른 사용자 ID를 사용할 수 있습니다.

큐 관리자 또는 큐 공유 그룹 레벨에서 대체 사용자 보안 검사를 켜거나 끌 수 있습니다.

컨텍스트

컨텍스트는 특정 메시지에 적용되며 메시지의 일부인 메시지 디스크립터(MQMD)에 포함된 정보입니다. 컨텍스트 정보는 다음과 같은 두 개의 섹션으로 구성됩니다.

ID 섹션
큐에 메시지를 처음 넣는 애플리케이션의 사용자입니다. 이는 다음 필드로 구성됩니다.

- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

원본 섹션
현재 저장된 큐에 메시지를 넣는 애플리케이션. 이는 다음 필드로 구성됩니다.

- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

애플리케이션은 MQPUT 또는 MQPUT1 호출이 실행된 경우 컨텍스트 데이터를 지정할 수 있습니다. 애플리케이션이 데이터를 생성할 수 있거나, 데이터가 다른 메시지로부터 전달될 수 있거나, 기본적으로 큐 관리자가 데이터를 생성할 수 있습니다. 예를 들어 서버 프로그램은 컨텍스트 데이터를 사용하여 요청자의 ID(즉, 올바른 애플리케이션에서 이 메시지가 파생되었는지)를 확인할 수 있습니다. 일반적으로 *UserIdentifier* 필드는 대체 사용자의 사용자 ID를 판별하는 데 사용됩니다.

컨텍스트 보안을 사용하여 MQOPEN 또는 MQPUT 호출에서 컨텍스트 옵션을 지정할 수 있는지 여부를 제어합니다. 컨텍스트 옵션에 대한 정보는 메시지 컨텍스트와 관련된 MQOPEN 옵션을 참조하십시오. 컨텍스트와 관련된 메시지 디스크립터 필드에 대한 설명은 MQMD - 메시지 디스크립터MQMD - 메시지 디스크립터를 참조하십시오.

컨텍스트 보안 검사를 끄는 경우 큐 보안이 허용하는 모든 컨텍스트 옵션을 모든 사용자가 사용할 수 있습니다.

큐, 큐 관리자 또는 큐 공유 그룹 레벨에서 컨텍스트 보안 검사를 켜거나 끌 수 있습니다.

z/OS z/OS의 가용성

IBM MQ for z/OS에는 고가용성을 위한 많은 기능이 있습니다. 이 주제에서는 가용성에 대한 몇 가지 고려사항을 설명합니다.

큐 관리자 또는 채널 시작기에서 실패한 경우 IBM MQ의 여러 기능에서 시스템 가용성을 늘릴 수 있습니다. 이 기능에 대한 자세한 정보는 다음 절을 참조하십시오.

- Sysplex 고려사항
- 공유 큐
- 공유 채널
- IBM MQ 네트워크 가용성
- z/OS 자동 재시작 관리자(ARM) 사용
- z/OS XRF(Extended Recovery Facility) 사용
- 큐 공유 그룹에서 복구를 위해 z/OS GROUPUR 속성 사용
- 가용성에 대한 자세한 정보를 제공하는 위치

SYSPLEX 고려사항

*sysplex*에서는 다수의 z/OS 운영 체제 이미지가 단일 시스템 이미지 안에서 협업하며 커플링 기능을 사용하여 통신합니다. IBM MQ는 개선된 가용성을 위해 sysplex 환경의 기능을 사용할 수 있습니다.

큐 관리자와 특정 z/OS 이미지 사이의 연관관계를 제거하면 이미지 실패 시 다른 z/OS 이미지에서 큐 관리자를 재시작할 수 있습니다. 재시작 메커니즘은 수동이거나 ARM을 사용하거나 다음을 보장하는 경우 시스템 자동화를 사용할 수 있습니다.

- 모든 페이지 세트, 로그, 부트스트랩 데이터 세트, 코드 라이브러리 및 큐 관리자 구성 데이터 세트는 공유 볼륨에 정의됩니다.
- 서브시스템 정의에는 SYSPLEX 범위와 SYSPLEX 내 고유한 이름이 있습니다.
- IPL시 모든 z/OS 이미지에 설치된 초기 코드 의 레벨은 동일한 레벨입니다.
- TCP 가상 IP 주소(VIPA)는 SYSPLEX의 각 TCP 스택에서 사용 가능하며, 기본 호스트 이름 대신 VIPA를 사용하도록 IBM MQ TCP 리스너와 인바운드 연결을 구성했습니다.

sysplex에서 TCP 사용에 대한 자세한 정보는 *TCP/IP in a sysplex*, SG24-5235, IBM Redbooks® 서적을 참조하십시오.

또한 SYSPLEX의 다른 운영 체제 이미지에서 실행 중인 여러 큐 관리자가 더 높은 가용성 및 워크로드 조정을 위해 공유 큐 및 공유 채널을 활용할 수 있는 큐 공유 그룹으로 작동하도록 구성할 수 있습니다.

공유 큐

큐 공유 그룹 환경에서 애플리케이션은 큐 공유 그룹 내 큐 관리자에 연결할 수 있습니다. 큐 공유 그룹의 모든 큐 관리자가 동일한 공유 큐 세트에 액세스할 수 있으므로 애플리케이션은 특정 큐 관리자의 가용성에 종속되지 않으며, 큐 공유 그룹의 큐 관리자는 모든 큐를 지원할 수 있습니다. 큐 공유 그룹의 다른 모든 큐 관리자가 큐를 계속 처리할 수 있으므로 큐 관리자가 중지되면 더 많은 가용성을 제공합니다. 공유 큐의 고가용성에 대한 정보는 166 페이지의 『공유 큐 사용 이점』의 내용을 참조하십시오.

큐 공유 그룹 내에서 메시지의 가용성을 더욱 강화시키기 위해 IBM MQ는 커플링 기능과 그룹 내의 다른 큐 관리자와의 비정상적 연결 끊김을 발견하고 가능한 경우 아직 보류 상태인 해당 큐 관리자의 작업 단위를 대신 완료합니다. 이는 피어 복구라고 하며 235 페이지의 『피어 복구』에서 설명됩니다.

피어 복구는 실패 시 인다우트(in doubt) 상태였던 작업 단위를 복구할 수 없습니다. 자동 재시작 관리자를 사용하여 실패와 관련된 모든 시스템(예: CICS, Db2 및 IBM MQ)을 재시작할 수 있으며 동일한 새 프로세서에서 모두 재시작되었는지 확인할 수 있습니다. 이는 다시 동기화할 있음을 의미하므로 인다우트(in-doubt) 작업 단위의 빠른 복구를 제공합니다. 이 프로그램은 247 페이지의 『z/OS 자동 재시작 관리자(ARM) 사용』에서 설명합니다.

공유 채널

큐 공유 그룹 환경에서 IBM MQ에서는 네트워크에 고가용성을 제공하는 기능을 제공합니다. 채널 시작기를 사용하면 적격 서버 세트에서 네트워크 요청 균형을 조정하는 네트워킹 제품을 사용하고 네트워크에서 서버 실패를 숨길 수 있습니다(예: VTAM 일반 자원). IBM MQ는 첨부 요청을 큐 공유 그룹의 사용 가능한 채널 시작기로 라우팅할 수 있도록 인바운드 요청에 대한 일반 포트를 사용합니다. 이 프로그램은 185 페이지의 『공유 채널』에서 설명합니다.

공유 아웃바운드 채널은 공유 전송 큐에서 송신하는 메시지를 가져옵니다. 공유 채널 상태에 대한 정보는 전체 큐 공유 그룹 레벨에 대한 한 위치에 보유됩니다. 즉, 채널 시작기, 큐 관리자 또는 통신 서브시스템에 실패하면 큐 공유 그룹의 다른 채널 시작기에서 채널이 자동으로 재시작될 수 있음을 의미합니다. 이는 피어 채널 복구라고 하며 공유 아웃바운드 채널에서 설명됩니다.

IBM MQ 네트워크 가용성

IBM MQ 메시지는 큐 관리자에서 채널을 사용하여 IBM MQ 네트워크의 큐 관리자로 전달됩니다. 네트워크 문제점을 감지하고 다시 연결하는 IBM MQ 채널의 기능 및 큐 관리자의 네트워크 가용성을 향상시키도록 여러 레벨에서 구성을 변경할 수 있습니다.

TCP *Keepalive*는 TCP/IP 채널에 대해 사용 가능합니다. 이를 통해 네트워크 실패를 감지하도록 세션 사이에서 TCP가 정기적으로 패킷을 송신합니다. KAINT 채널 속성은 채널에 대한 이러한 패킷의 빈도를 결정합니다.

*AdoptMCA*를 사용하면 네트워크 정전의 결과로 처리 수신이 차단된 채널이 새 연결 요청에 의해 종료되고 바뀔 수 있습니다. PCF(Programmable Command Format) 인터페이스에서 AdoptNewMCAType 특성 또는 MQSC 유틸리티에서 ADOPTMCA 큐 관리자 특성을 사용하여 AdoptMCA를 제어합니다.

*ReceiveTimeout*은 네트워크 수신 호출에서 채널을 영구적으로 차단하지 못하게 합니다. RCVTIME 및 RCVTMIN 채널 시작기 매개변수는 하트비트 간격의 기능으로 채널의 수신 제한시간 특성을 결정합니다. 자세한 내용은 큐 관리자 매개변수를 참조하십시오.

z/OS 자동 재시작 관리자(ARM) 사용

z/OS 자동 재시작 관리자 (ARM) 와 함께 IBM MQ for z/OS 를 사용할 수 있습니다. 큐 관리자 또는 채널 시작기가 실패한 경우 ARM은 이를 동일한 z/OS 이미지에서 재시작합니다. z/OS에 실패하면 관련 서브시스템 및 애플리케이션의 전체 그룹도 실패합니다. ARM은 sysplex 내의 다른 z/OS 이미지에서 사전 정의된 순서로 실패한 모든 시스템을 자동으로 재시작할 수 있습니다. 이는 교차 시스템 재시작이라고 합니다.

ARM에서는 공유 큐 환경에서 인다우트(in-doubt) 트랜잭션의 빠른 복구가 가능합니다. 또한 큐 공유 그룹을 사용하지 않는 경우 더 높은 가용성을 제공합니다.

ARM을 사용하여 z/OS 장애 시에 sysplex 내의 다른 z/OS 이미지에서 큐 관리자를 재시작할 수 있습니다.

자동 재시작을 사용 가능하게 하려면 다음을 수행해야 합니다.

1. ARM 커플링 데이터 세트를 설정하십시오.
2. z/OS가 *ARM* 정책에서 수행하려는 자동 재시작 조치를 정의하십시오.
3. ARM 정책을 시작하십시오.

다른 z/OS 이미지에서 큐 관리자를 자동으로 재시작하려면 해당 큐 관리자를 재시작해야 하는 각 z/OS 이미지의 모든 큐 관리자가 sysplex 전체 고유한 4자의 서브시스템 이름으로 정의되어야 합니다.

IBM MQ와 함께 ARM을 사용하는 데 대해서는 IBM MQ 네트워크에서 ARM 사용에 설명되어 있습니다.

z/OS XRF(Extended Recovery Facility) 사용

XRF(Extended Recovery Facility) 환경에서 IBM MQ를 사용할 수 있습니다. 모든 IBM MQ 소유 데이터 세트(실행 가능 코드, BSDS, 로그 및 페이지 세트)는 활성 및 대체 XRF 프로세스 사이에서 공유되는 DASD에 있어야 합니다.

복구를 위해 XRF를 사용하는 경우 활성 프로세서에서 큐 관리자를 중지하고 대체 프로세서에서 시작해야 합니다. CICS의 경우, CICS에서 제공하는 명령 목록 테이블(CLT)을 사용하여 이를 수행하거나 시스템 운영자가 이를 자동으로 수행할 수 있습니다. IMS의 경우 이는 수동 조작이며 통합하는 IMS 시스템이 프로세서 스위치를 완료한 후 이를 수행해야 합니다.

IBM MQ 유틸리티는 큐 관리자를 대체 프로세서로 교환하기 전에 완료 또는 종료되어야 합니다. XRF 복구 계획을 계획할 때 이러한 잠재적인 인터럽션의 영향을 신중하게 고려하십시오.

활성 프로세서에서 큐 관리자를 종료하기 전에 대체 프로세서에서 큐 관리자가 시작하지 못하도록 주의하십시오. 이른 시작은 데이터, 카탈로그 및 로그에서 심각한 무결성 문제점을 일으킬 수 있습니다. 글로벌 자원 직렬화(GRS)를 사용하면 두 시스템에서 IBM MQ의 동시 사용을 방지하여 무결성 문제점을 회피하는 데 도움이 됩니다. 보호되는 자원으로 BSDS를 포함하고 GRS 링에 활성 및 대체 XRF 프로세서를 포함해야 합니다.

큐 공유 그룹의 복구에 z/OS GROUPUR 속성 사용

큐 공유 그룹(QSG)에서는 이 토픽에서 설명하는 추가 트랜잭션 기능을 허용합니다. GROUPUR 속성에서는 XA 클라이언트 애플리케이션에서 필요할 수 있는 인다우트(in-doubt) 트랜잭션 복구를 QSG의 멤버에서 수행할 수 있습니다.

XA 클라이언트 애플리케이션이 SYSPLEX를 통해 큐 공유 그룹(QSG)에 연결하는 경우 연결되는 특정 큐 관리자를 보장할 수는 없습니다. 큐 공유 그룹 내에서 큐 관리자가 GROUPUR 속성을 사용하면 QSG의 모든 멤버에서 발

생해야 할 수 있는 인다우트(in-doubt) 트랜잭션 복구를 사용할 수 있습니다. 애플리케이션이 처음에 연결된 큐 관리자를 사용할 수 없어도 트랜잭션 복구가 수행될 수 있습니다.

이 기능은 QSG의 특정 멤버에 대한 종속성에서 XA 클라이언트 애플리케이션을 해제하므로 큐 관리자의 가용성을 확장합니다. 큐 공유 그룹은 단일 큐 관리자 실패 지점 없이 모든 IBM MQ 기능을 제공하며 트랜잭션 애플리케이션에 단일 엔티티로 나타납니다.

이 기능은 트랜잭션 애플리케이션에 투명하지 않습니다.

가용성에 대한 자세한 정보를 제공하는 위치

다음 소스에서 이 토픽에 대한 자세한 정보를 찾을 수 있습니다.

표 *24.* 가용성에 대한 자세한 정보를 제공하는 위치

주제	찾을 위치
큐 공유 그룹	149 페이지의 『공유 큐 및 큐 공유 그룹』
시스템 매개변수	시스템 매개변수 구성
자동 재시작 관리자 사용 유틸리티 프로그램	IBM MQ 네트워크에서 ARM 사용
MQSC 명령	MQSC 명령

z/OS IBM MQ for z/OS에 대한 모니터링 및 통계

IBM MQ for z/OS에는 큐 관리자를 모니터링하고 통계를 수집하는 기능 세트가 있습니다.

IBM MQ는 시스템을 모니터링하고 통계를 수집하는 기능을 제공합니다. 이 기능에 대한 추가적인 정보는 다음 절을 참조하십시오.

- 248 페이지의 『온라인 모니터링』
- 248 페이지의 『IBM MQ 추적』
- 249 페이지의 『이벤트』

온라인 모니터링

IBM MQ에는 IBM MQ 오브젝트의 상태를 모니터링하는 다음 명령이 포함됩니다.

- DISPLAY CHSTATUS는 지정된 채널의 상태를 표시합니다.
- DISPLAY QSTATUS는 지정된 큐의 상태를 표시합니다.
- DISPLAY CONN은 지정된 연결의 상태를 표시합니다.

위의 명령에 대한 자세한 정보는 MQSC 명령을 참조하십시오.

IBM MQ 추적

IBM MQ는 큐 관리자가 실행 중인 동안 다음 정보를 수집하기 위해 사용할 수 있는 추적 기능을 제공합니다.

성능 통계

통계 추적은 성능을 모니터하고 시스템을 성능 조정할 수 있도록 도움을 주는 다음 정보를 수집합니다.

- 서로 다른 MQI 요청의 수(메시지 관리자 통계)
- 서로 다른 오브젝트 요청의 수(데이터 관리자 통계)
- Db2 사용에 대한 정보(Db2 관리자 통계)

- 커플링 기능 사용에 대한 정보(커플링 기능 관리자 통계)
- SMDS 사용에 대한 정보(공유 메시지 데이터 세트 통계)
- 버퍼 풀 사용에 대한 정보(버퍼 관리자 통계)
- 로깅에 대한 정보(로그 관리자 통계)
- 스토리지 사용에 대한 정보(스토리지 관리자 통계)
- 잠금 요청에 대한 정보(잠금 관리자 통계)

계정 데이터

- 회계 추적은 MQI 호출의 처리에 사용된 프로세서 시간에 대한 정보와 특정 사용자가 작성한 MQPUT 및 MQGET 요청의 수에 대한 정보를 수집합니다.
- IBM MQ는 IBM MQ를 사용하는 각 태스크에 대한 정보도 수집할 수 있습니다. 이 데이터는 스레드 레벨 회계 레코드로 수집됩니다. 각 스레드마다 IBM MQ는 해당 스레드가 사용하는 각각의 큐에 대한 정보도 수집합니다.

추적에 의해 생성된 데이터는 시스템 관리 기능(SMF) 또는 일반 추적 기능(GTF)으로 전송됩니다.

이벤트

IBM MQ 이벤트는 큐 관리자의 오류, 경고 및 기타 중요한 발생에 대한 정보를 제공합니다. 이 이벤트를 자체 시스템 관리 애플리케이션에 통합함으로써 다중 IBM MQ 애플리케이션에 대해 여러 큐 관리자의 활동을 모니터할 수 있습니다. 특히 단일 큐 관리자에서 시스템의 모든 큐 관리자를 모니터할 수 있습니다.

이벤트는 사용자 작성 보고 메커니즘을 통해 운영자에 이벤트의 프리젠테이션을 제공하는 관리 애플리케이션에 보고될 수 있습니다. 또한 이벤트는 NetView와 같은 기타 관리 네트워크의 에이전트로서 작동하는 애플리케이션이 보고서를 모니터하고 적절한 경보를 작성할 수 있도록 합니다.

관련 태스크

IBM MQ 추적 사용

IBM MQ 이벤트 사용

z/OS z/OS에서 복구 단위 지정

특정 트랜잭션 애플리케이션은 큐 관리자 이름 대신 연결 시 QSG 이름을 지정하여 큐 공유 그룹(QSG)에서 큐 관리자에 연결할 때 GROUP 복구 단위 속성 지정(QMGR이 아님)을 사용할 수 있습니다. 이를 통해 트랜잭션 복구는 QSG의 동일한 큐 관리자에 다시 연결하는 데 필요한 요구사항을 제거하여 보다 유연하고 강력해질 수 있습니다.

큐 공유 그룹 이름을 사용하여 연결된 애플리케이션에서 시작된 트랜잭션에는 GROUP 복구 단위 속성 지정도 있습니다.

트랜잭션 애플리케이션이 GROUP 복구 단위 속성 지정으로 연결된 경우 이는 논리적으로 큐 공유 그룹에 연결되며 특정 큐 관리자에 대한 연관관계가 없습니다. 커미트 프로세스의 1단계를 완료하고 시작된 모든 2단계 커미트 트랜잭션(즉, 인다우트(in doubt) 상태임)은 QSG에서 큐 관리자에 연결된 경우 조회 및 해석 가능합니다. 복구 시나리오에서 이는 트랜잭션 통합기에서 동일한 큐 관리자에 다시 연결하지 않아도 됨을 의미합니다. 이는 당시에 사용 불가능할 수 있습니다.

QMGR 복구 단위 속성 지정으로 연결된 애플리케이션에는 이들이 연결된 큐 관리자에 대한 직접적인 연관관계가 있습니다. 복구 시나리오에서 큐 관리자가 큐 공유 그룹에 속해있는지 여부에 상관없이 인다우트(in-doubt) 트랜잭션을 해결하려면 트랜잭션 통합기를 동일한 큐 관리자에 다시 연결해야 합니다.

애플리케이션이 큐 공유 그룹 이름을 지정하고 GROUP 복구 단위 속성 지정으로 QSG에 있는 큐 관리자에 연결하는 경우 큐 공유 그룹은 논리적으로 별도의 자원 관리자입니다. 즉, 동일한 복구 단위 속성 지정으로 다시 연결된 경우 인다우트(in-doubt) 트랜잭션은 애플리케이션에만 보일 수 있음을 의미합니다. QMGR 복구 단위 속성 지정을 포함하는 인다우트(in-doubt) 트랜잭션은 GROP 복구 단위 속성 지정에 연결된 애플리케이션에는 보이지 않으며 그 반대의 경우도 마찬가지입니다.

관련 개념

250 페이지의 『복구의 GROUP 단위 사용』

큐 공유 그룹은 GROUP 복구 단위를 구성하고 이에 대한 지원을 사용 가능하게 할 수 있습니다.

250 페이지의 『애플리케이션 지원』
이 페이지를 사용하여 GROUP 복구 단위 속성 지정을 통해 연결할 수 있는 애플리케이션을 판별하십시오.

z/OS 복구의 GROUP 단위 사용

큐 공유 그룹은 GROUP 복구 단위를 구성하고 이에 대한 지원을 사용 가능하게 할 수 있습니다.

QSG 내 큐 관리자에서 GROUP 복구 단위를 사용하려면 GROUPUR 큐 관리자 속성을 사용 가능하게 하십시오. 이 개념에 대한 자세한 정보를 얻으려면 이 주제의 나머지 내용을 읽기 전에 249 페이지의 『z/OS에서 복구 단위 지정』의 내용을 참조하십시오.

GROUPUR 큐 관리자 속성을 사용 가능해지면 큐 관리자는 GROUP 복구 단위 배치와 새 연결을 허용합니다. 이 속성을 사용하지 않을 경우 이 속성 지정과의 새 연결은 허용되지 않지만, 이미 연결되어 있는 애플리케이션은 연결을 끊을 때까지 이에 영향을 받지 않습니다.

애플리케이션이 GROUP 복구 단위 속성 지정을 통해 연결하고 인다우트(in doubt) 상태의 트랜잭션을 조회하거나 큐 공유 그룹(QSG)의 임의의 위치에서 시작된 트랜잭션을 해석하려는 경우 현재 연결된 큐 관리자는 요청을 처리할 수 있도록 큐 공유 그룹의 다른 멤버와 통신할 수 있어야 합니다. 이를 수행하기 위해 공유 큐 SYSTEM.QSG.UR.RESOLUTION.QUEUE를 사용합니다. 이 큐는 복구 가능한 애플리케이션 구조 CSQSYSAPPL에 있어야 합니다. 해결 요청을 처리할 때 이 큐에 지속 메시지가 저장되므로 구조는 복구 가능해야 합니다.

GROUP 복구 단위를 사용하기 전에 커플링 기능 구조 및 공유 큐가 정의되어 있는지 확인해야 합니다. CSQ4INSS 샘플에서 정의를 사용할 수 있습니다. 큐가 정의되었거나 시동 시 감지된 경우 큐 공유 그룹의 각 큐 관리자는 수신 요청을 수신할 수 있도록 큐를 엽니다. 올바르게 정의되지 않아서 큐를 삭제하거나 이동하려는 경우 MQGET 요청을 금지하도록 큐 오브젝트를 업데이트하여 큐 관리자가 열린 핸들을 닫도록 요청할 수 있습니다. 필수 정정을 수행한 경우 다시 한 번 큐에서 메시지를 가져오도록 애플리케이션을 허용하면 각 큐 관리자에 이를 다시 열도록 지시합니다. DISPLAY QSTATUS 명령을 사용하여 큐에서 열린 핸들을 식별하십시오.

이 설정을 완료하고 나면 복구의 GROUP 복구 단위 배치로 트랜잭션 애플리케이션과 연결이 가능했으면 하는 각 큐 관리자에서 GROUP 복구 단위를 사용할 수 있습니다. 이 경우 큐 공유 그룹에 모든 큐 관리자가 없어도 되지만, 큐 공유 그룹의 서브세트에서 이 기능을 사용으로 설정하려고 선택하는 경우 애플리케이션이 사용 가능하게 한 큐 관리자에만 연결하려고 하는지 확인해야 합니다. 자세한 정보는 250 페이지의 『애플리케이션 지원』의 내용을 참조하십시오.

GROUPUR 큐 관리자 속성을 사용 가능하게 하려는 경우 많은 구성 검사가 수행됩니다. 큐 관리자는 다음을 검사합니다.

- 이 항목이 큐 공유 그룹에 속합니다.
- CSQ4INSS의 정의에 따라 공유 큐 SYSTEM.QSG.UR.RESOLUTION.QUEUE가 정의됩니다.
- SYSTEM.QSG.UR.RESOLUTION.QUEUE가 복구 가능한 CF 구조 CSQSYSAPPL에 있습니다.

위 검사에 실패하면 GROUPUR 속성은 사용 불가능 상태로 남아 있으며 메시지 코드가 리턴됩니다.

큐 관리자 속성이 사용 가능한 경우 이러한 구성 검사는 큐 관리자 시동 시에도 수행됩니다. 시동 중 검사에 실패하면 GROUP 복구 단위가 사용 불가능하고 큐 관리자는 실패한 검사를 식별하는 메시지를 발행합니다. 필수 정정 조치를 수행한 경우 큐 관리자 속성을 다시 사용 가능하게 해야 합니다.

z/OS 애플리케이션 지원

이 페이지를 사용하여 GROUP 복구 단위 속성 지정을 통해 연결할 수 있는 애플리케이션을 판별하십시오.

GROUP 복구 단위 속성 지정에 대한 지원은 IBM MQ for z/OS가 트랜잭션 통합기가 아닌 자원 관리자인 트랜잭션 애플리케이션의 특정 유형으로만 제한됩니다. 현재 지원되는 트랜잭션 애플리케이션은 다음과 같습니다.

- IBM MQ 확장된 트랜잭션 클라이언트 애플리케이션
- 애플리케이션 서버에서 실행 중인 IBM MQ classes for JMS 애플리케이션 (예: WebSphere Application Server).
- CICS Transaction Server 4.2 이상에서 실행 중인 CICS 애플리케이션 (CICS MQCONN 자원 정의가 RESYNCMEMBER (GROUPRESYNC) 로 구성된 경우).

관련 개념

251 페이지의 『IBM MQ 확장된 트랜잭션 클라이언트 애플리케이션』
이 페이지를 사용하여 IBM MQ 확장 트랜잭션 클라이언트 애플리케이션이 GROUP 복구 단위 속성 지정을 사용할 수 있는 방법을 판별하십시오.

251 페이지의 『CICS 애플리케이션』
이 페이지를 사용하면 CICS가 GROUP 복구 단위 배치를 사용하는 방법을 판별할 수 있습니다.

z/OS *IBM MQ* 확장된 트랜잭션 클라이언트 애플리케이션

이 페이지를 사용하여 IBM MQ 확장 트랜잭션 클라이언트 애플리케이션이 GROUP 복구 단위 속성 지정을 사용할 수 있는 방법을 판별하십시오.

IBM MQ 확장된 트랜잭션 클라이언트 애플리케이션의 예로 JMS를 사용하고 WebSphere Application Server에서 실행되는 항목이 있으며, 이는 로컬 바인딩이 아닌 TCP/IP로 IBM MQ에 연결합니다. 이러한 클라이언트 애플리케이션은 TCP/IP와 같은 네트워크 연결에서 IBM MQ for z/OS에 연결합니다. 이러한 애플리케이션의 경우 이는 QMGR 또는 GROUP 복구 단위 속성 지정이 사용되는지 여부를 지정하는 xa_open 호출에서 전달되는 xa_info 문자열의 QMNAME 매개변수에 대해 지정된 값입니다. xa_open에 대한 자세한 정보는 xa_open 문자열의 형식 및 xa_open의 추가 오류 처리를 참조하십시오. JMS 애플리케이션의 경우, 이는 특정 큐 관리자 이름 대신 ConnectionFactory에서 큐 공유 그룹(QSG)의 이름을 지정하여 수행됩니다.

XA 클라이언트 애플리케이션이 GROUP 복구 단위 속성 지정을 사용하도록 하려면 클라이언트 애플리케이션이 특정 큐 관리자가 아닌 GROUPUR 속성이 사용 가능한 큐 공유 그룹에서 큐 관리자로 라우팅될 수 있도록 TCP/IP 설정을 구성해야 합니다. 이를 수행하는 데 사용할 수 있는 동적 가상 IP 주소 기술 중 하나가 z/OS SysPlex Distributor입니다. 자세한 정보는 z/OS 통신 서버 및 z/OS 기본 기술: 동적 가상 주소 지정의 내용을 참조하십시오. 큐 공유 그룹의 큐 관리자 서브세트에서 GROUP 복구 단위를 사용하려면, 사용되지 않는 큐 관리자에 클라이언트 애플리케이션이 라우팅될 수 없는지 확인하십시오.

클라이언트 애플리케이션은 공유 채널을 사용하여 큐 공유 그룹에 연결하지 않아도 됩니다.

z/OS *CICS* 애플리케이션

이 페이지를 사용하면 CICS가 GROUP 복구 단위 배치를 사용하는 방법을 판별할 수 있습니다.

CICS 4.2 이상은 MQCONN 자원 정의에서 그룹 재동기화 옵션인 RESYNCMEMBER (GROUPRESYNC) 를 제공합니다. 이 옵션으로 구성된 CICS는 CICS 리전과 동일한 LPAR에서 실행 중인 큐 공유 그룹에 있는 알맞은 모든 큐 관리자와 연결할 수 있습니다. CICS GROUPRESYNC 옵션을 지원하려면 큐 관리자는 MQ V7.1 이상에서 실행 중이어야 하며 GROUPUR 지원을 사용 중이어야 합니다.

GROUPRESYNC를 사용하여 MQ에 연결되어 있는 CICS 리전 내에서 실행 중인 트랜잭션은 GROUP 복구 단위 배치로 작업 단위를 작성합니다.

RESYNCMEMBER (GROUPRESYNC) 를 사용하면 CICS 리전이 큐 관리자 재시작을 기다리지 않고 필요에 따라 인다우트 트랜잭션을 해결하여 동일한 LPAR에서 실행 중인 대체 적격 큐 관리자에 즉시 연결할 수 있으므로 큐 관리자 실패 후 더 빠른 복구를 사용할 수 있습니다.

RESYNCMEMBER (GROUPRESYNC) 도 CICS에 대해 보다 유연한 다시 시작 옵션을 사용합니다. GROUPRESYNC와 MQ 공유 큐를 사용하도록 구성된 MQ 연결이 있는 CICS 리전은 동일한 큐 공유 그룹의 멤버로서 실행 중인 큐 관리자가 있는 모든 LPAR에서 다시 시작될 수 있습니다.

z/OS IBM MQ 및 기타 z/OS 제품

이 토픽을 사용하여 IBM MQ를 다른 z/OS 제품과 함께 사용할 수 있는 방법을 이해하십시오.

관련 개념

252 페이지의 『IBM MQ 및 CICS』
IBM MQ 9.0.0이상에서 지원되는 모든 CICS 버전은 어댑터 및 브릿지의 CICS 제공 버전을 사용합니다.

257 페이지의 『IBM MQ for z/OS 및 WebSphere Application Server』
이 토픽을 사용하여 WebSphere Application Server의 IBM MQ for z/OS 사용을 이해하십시오.

관련 참조

253 페이지의 『IBM MQ 및 IMS』
이 토픽을 사용하여 IBM MQ가 IMS와 함께 작동하는 방법을 이해하십시오. IMS 어댑터를 사용하면 큐 관리자를 IMS에 연결하고 MQI를 사용하도록 IMS 애플리케이션을 사용 가능하게 할 수 있습니다.

256 페이지의 『IBM MQ와 z/OS 배치, TSO 및 RRS 어댑터』
IBM MQ가 z/OS 배치, TSO 및 RRS 어댑터와 함께 작동하는 방식을 이해하는 데 이 토픽을 사용하십시오.

z/OS IBM MQ 및 CICS

IBM MQ 9.0.0이상에서 지원되는 모든 CICS 버전은 어댑터 및 브릿지의 CICS 제공 버전을 사용합니다.

IBM MQ CICS 어댑터 및 IBM MQ CICS bridge 컴포넌트 구성에 대한 자세한 정보는 CICS 문서의 IBM MQ에 대한 연결 구성 절을 참조하십시오.

관련 태스크

CICS과(와) 함께 IBM MQ 사용

z/OS CICS 그룹 첨부

CICS 그룹 첨부는 개별 큐 관리자를 지정하기보다 CICS 리젼이 동일한 LPAR의 IBM MQ 큐 공유 그룹의 활성 멤버에 연결할 수 있는 기능을 제공합니다. CICS는 여전히 한 번에 단일 큐 관리자에 연결됩니다.

CICS 그룹 첨부를 지원하려면 LPAR에 최소한 두 명의 큐 관리자가 필요합니다. 그룹 첨부를 사용하면 특정 큐 관리자가 활성 상태일 필요가 없기 때문에 더 높은 가용성이 제공됩니다. CICS는 LPAR의 큐 공유 그룹에 있는 큐 관리자에 연결됩니다.

자세한 정보는 MQCONN 자원의 CICS 문서를 참조하십시오.

CICS는 자신이 큐 관리자인 것처럼 전달된 MQNAME에 대한 연결을 시도합니다.

- 큐 관리자가 있고 활성 상태이면 연결이 작동합니다.
- 연결에 실패하면 CICS는 동일한 LPAR에서 활성인 항목을 확인하기 위해 그룹에서의 큐 관리자 상태를 조회합니다.
- 다중 큐 관리자가 활성 상태인 경우 CICS 는 RESYNCMEMBER (YES) 및 UOW 상태를 검사하여 CICS 가 특정 멤버에 연결해야 하는지 또는 연결해야 하는지 또는 활성 상태가 아닌 경우 대기해야 하는지 여부를 판별합니다.
- 특정 멤버에 대한 연결이 필요하지 않은 경우, CICS는 큐 관리자를 선택합니다(임의 추출 알고리즘 사용).
- CICS는 선택된 큐 관리자에 대한 연결을 시도합니다.
- 시도에 실패하면 리턴 코드에 따라 CICS는 다음 멤버를 선택한 후에 선택 루프를 다시 진행합니다.
- 활성인 큐 관리자가 없는 경우, CICS는 큐 관리자 목록에 다중 연결을 발행하고 첫 큐 관리자가 사용 가능해질 때까지 ECBLIST에서 대기합니다.

관련 개념

252 페이지의 『CICS용 그룹 복구 단위(GROUPUR)』
IBM MQ GROUPUR for CICS 는 큐 공유 그룹 (QSG) 에서 인다우트 (in-doubt) 작업 단위에 대한 피어 복구를 제공합니다. 한 IBM MQ 큐 관리자는 큐 공유 그룹의 다른 큐 관리자를 대신하여 인다우트 작업 단위를 해결할 수 있습니다. 이는 CICS가 그룹 접속을 통해 QSG의 다른 큐 관리자에 재연결하는 경우, 이전 IBM MQ 연결로부터 인다우트(in-doubt) 트랜잭션을 해결할 수 있음을 의미합니다.

관련 정보

IBM MQ 큐 공유 그룹 지원

z/OS CICS용 그룹 복구 단위(GROUPUR)

IBM MQ GROUPUR for CICS 는 큐 공유 그룹 (QSG) 에서 인다우트 (in-doubt) 작업 단위에 대한 피어 복구를 제공합니다. 한 IBM MQ 큐 관리자는 큐 공유 그룹의 다른 큐 관리자를 대신하여 인다우트 작업 단위를 해결할 수 있습니다. 이는 CICS가 그룹 접속을 통해 QSG의 다른 큐 관리자에 재연결하는 경우, 이전 IBM MQ 연결로부터 인다우트(in-doubt) 트랜잭션을 해결할 수 있음을 의미합니다.

CICS 리젼이 큐 관리자를 사용 중이고 이 큐 관리자가 비정상적으로 종료된 경우, 인다우트 트랜잭션이 복구됩니다. 이렇게 되면 CICS 리젼이 사용 중이던 큐 관리자가 다시 시작될 때까지 기다릴 필요가 없어지고 인다우트 작업 단위를 해결할 수 있게 됩니다. 이는 첫 번째 큐 관리자의 비정상 종료 시에 CICS가 다른 큐 관리자에 연결할 수 있도록 LPAR에 최소한 두 명의 큐 관리자가 있어야 한다는 의미입니다.

CICS MQCONN 정의의 새 RESYNCMEMBER (GROUPRESYNC) 설정:

- IBM MQ 그룹 첨부 기능 및 피어 복구를 사용합니다.
- GROUPUR 속성이 사용으로 설정된 큐 관리자가 필요합니다.
- 여전히 기존 CICS MQCONN RESYNCMEMBER 설정 (YES및 NO) 을 지원합니다.
  - 기존 CICS 그룹 첨부 기능을 사용하며 피어 복구는 사용하지 않습니다.
  - RESYNCMEMBER 설정 변경은 다음에 CICS 가 IBM MQ에 연결할 때 적용됩니다.

관련 개념

250 페이지의 『복구의 GROUP 단위 사용』

큐 공유 그룹은 GROUP 복구 단위를 구성하고 이에 대한 지원을 사용 가능하게 할 수 있습니다.

z/OS IBM MQ 및 IMS

이 토픽을 사용하여 IBM MQ가 IMS와 함께 작동하는 방법을 이해하십시오. IMS 어댑터를 사용하면 큐 관리자를 IMS에 연결하고 MQI를 사용하도록 IMS 애플리케이션을 사용 가능하게 할 수 있습니다.

선택적 추가 IBM MQ-IMS 브릿지를 사용하면 애플리케이션이 MQI를 사용하지 않는 IMS 애플리케이션을 실행할 수 있습니다. 즉, 다시 기록하지 않고도 IBM MQ에서 레거시 애플리케이션을 사용할 수 있음을 의미합니다.

이 컴포넌트에 대한 자세한 정보는 다음 하위 주제를 참조하십시오.

관련 개념

IBM MQ for z/OS 의 IMS 및 IMS 브릿지 애플리케이션

관련 태스크

IMS 어댑터 설정

IMS 브릿지 설정

IMS 어댑터 작동

관련 참조

MQIIH - IMS 정보 헤더

z/OS IMS 어댑터

IMS 어댑터는 IMS 애플리케이션 프로그램과 IBM MQ 서브시스템 사이의 인터페이스입니다.

IBM MQ 어댑터를 사용하면 서로 다른 애플리케이션 환경에서 메시지 큐잉 네트워크를 통해 메시지를 송수신할 수 있습니다. IMS 어댑터는 IMS 애플리케이션 프로그램과 IBM MQ 서브시스템 사이의 인터페이스입니다. 그러면 IMS 애플리케이션 프로그램에서 MQI를 사용할 수 있습니다.

IMS 어댑터는 IMS에서 제공하는 ESAF (External Subsystem Attach Facility) 를 사용하여 IBM MQ 에 대한 액세스 요청을 수신하고 해석합니다. 일반적으로 IMS는 운영자 개입 없이 자동으로 IBM MQ에 연결합니다.

IMS 어댑터는 다음 모드나 상태에서 실행 중인 프로그램에 대해 IBM MQ 자원에 대한 액세스를 제공합니다.

- 태스크(TCB) 모드
- 문제점 상태
- 비교차 메모리 모드
- 비액세스 등록 모드

어댑터는 애플리케이션 TCB(Task Control Block)에서 IBM MQ로의 연결 스레드를 제공합니다.

어댑터는 IMS 가 동기점 조정자 역할을 하는 IBM MQ 가 소유하는 자원에 대한 변경사항에 대해 2단계커미트 프로토콜을 지원합니다. IMS가 동기점 통합기가 아닌 대화(예: APPC 보호(SYNCLVL=SYNCPT) 대화)는 IMS 어댑터에서 지원되지 않습니다.

또한 어댑터는 트리거 모니터 트랜잭션(CSQQTRMN)을 제공합니다. 이 프로그램은 254 페이지의 『IMS 트리거 모니터』에서 설명합니다.

IBM MQ 를 IMS XRF (Extended Recovery Facility) 와 함께 사용하여 IMS 오류로부터의 복구를 지원할 수 있습니다.

참고: IMS 15.2 XRF (Extended Recovery Facility) 는 더 이상 지원되지 않습니다. 자세한 정보는 IMS 문서를 참조하십시오.

어댑터 사용

애플리케이션 프로그램 및 IMS 어댑터는 동일한 주소 공간에서 실행됩니다. 큐 관리자는 고유한 주소 공간에서 별도로 존재합니다.

적절한 IMS 언어 인터페이스 모듈에 대한 하나 이상의 MQI 호출 및 동적 MQI 호출을 사용하지 않는 한, IBM MQ 제공 API 스텁 프로그램, CSQQSTUB를 실행하는 각 프로그램을 링크-편집해야 합니다. 애플리케이션이 MQI 호출을 실행하면 스텁은 메시지 큐 관리자가 요청 처리를 관리하는 IMS 외부 서브시스템 인터페이스를 통해 어댑터로 제어를 전송합니다.

IMS를 사용하여 시스템 관리 및 조작

권한이 부여된 IMS 터미널 운영자는 IMS 명령을 실행하여 IBM MQ에 대한 연결을 제어 및 모니터할 수 있습니다. 그러나 IMS 터미널 운영자는 IBM MQ 주소 공간에 대한 제어 권한이 없습니다. 예를 들어, 운영자는 IMS 주소 공간에서 IBM MQ 를 종료할 수 없습니다.

제한사항

다음 IBM MQ API 호출은 IMS 어댑터를 사용하는 애플리케이션 내에서 지원되지 않습니다.

- MQCB
- MQCB_FUNCTION
- MQCTL

IMS 트리거 모니터

IMS 트리거 모니터 (**CSQQTRMN**) IBM MQ 이벤트가 발생할 때 (예: 메시지를 특정 큐에 넣을 때) IMS 트랜잭션을 시작하는 IBM MQ제공 IMS 애플리케이션입니다.

작동 방식

애플리케이션 메시지 큐에 메시지를 넣은 경우 트리거 조건을 만족하면 트리거가 생성됩니다. 그러면 큐 관리자는 일부 사용자 정의 데이터를 포함하는 메시지(트리거 메시지라고도 함)를 해당 메시지 큐에 지정된 이니시에이션 큐에 씁니다. IMS 환경에서 CSQQTRMN의 인스턴스를 시작하여 이니시에이션 큐를 모니터하고 도달하면 여기에서 트리거 메시지를 검색할 수 있습니다. 일반적으로 CSQQTRMN은 IMS 메시지 큐에 대한 INSERT(ISRT) 작업을 통해 다른 IMS 트랜잭션을 스케줄합니다. 시작된 IMS 애플리케이션은 애플리케이션 메시지 큐에서 메시지를 읽고 이를 처리합니다. CSQQTRMN은 비메시지 BMP로 실행되어야 합니다.

CSQQTRMN의 각 사본은 단일 이니시에이션 큐를 지원합니다. IBM MQ 또는 IMS가 종료될 때까지 트리거 모니터가 실행됩니다.

CSQQTRMN의 APPLCTN 매크로는 SCHDTYP=PARALLEL을 지정해야 합니다.

트리거 모니터는 배치 중심 BMP이므로 트리거 모니터에서 시작된 IMS 트랜잭션은 다음을 포함합니다.

- IOPCB의 LTERM 필드에서 공백
- IOPCB의 사용자 ID 필드에서 트리거 모니터 BMP의 PSB 이름

대상 IMS 트랜잭션이 보안 서버(이전에 RACF라고 함)에 의해 보호되는 경우 보안 서버에 대해 사용자 ID로 CSQQTRMN을 정의해야 할 수 있습니다.

z/OS IBM MQ-IMS 브릿지

IBM MQ - IMS 브릿지는 IBM MQ 애플리케이션에서 IMS 시스템의 애플리케이션으로의 직접 액세스를 허용하는 IBM MQ for z/OS 의 컴포넌트입니다.

IBM MQ-IMS 브릿지를 사용하면 암시적 *MQI* 지원이 가능합니다. 즉, 이를 다시 기록하거나, 다시 컴파일하거나, 다시 링크하지 않고도 3270 연결 터미널에서 제어하는 레거시 애플리케이션을 IBM MQ 메시지에서 제어하도록 리엔지니어링할 수 있음을 의미합니다. 브릿지는 IMS *Open Transaction Manager Access*(OTMA) 클라이언트입니다.

브릿지 애플리케이션에서는 IMS 애플리케이션 내에 IBM MQ 호출이 없습니다. 애플리케이션은 GET UNIQUE (GU)를 사용하여 해당 입력으로 IOPCB로 가져오고 ISRT를 사용하여 해당 출력을 IOPCB로 송신합니다. IBM MQ 애플리케이션은 메시지 데이터의 IMS 헤더(MQIIH 구조)를 사용하여 비프로그래밍 터미널에서 구동된 경우 전에 이들이 수행될 때 애플리케이션이 실행할 수 있는지 확인합니다. 다중 세그먼트 메시지를 처리하는 IMS 애플리케이션을 사용하는 경우, 모든 세그먼트가 하나의 IBM MQ 메시지에 포함되어야 함을 명심하십시오.

IMS 브릿지는 255 페이지의 그림 78에서 설명됩니다.

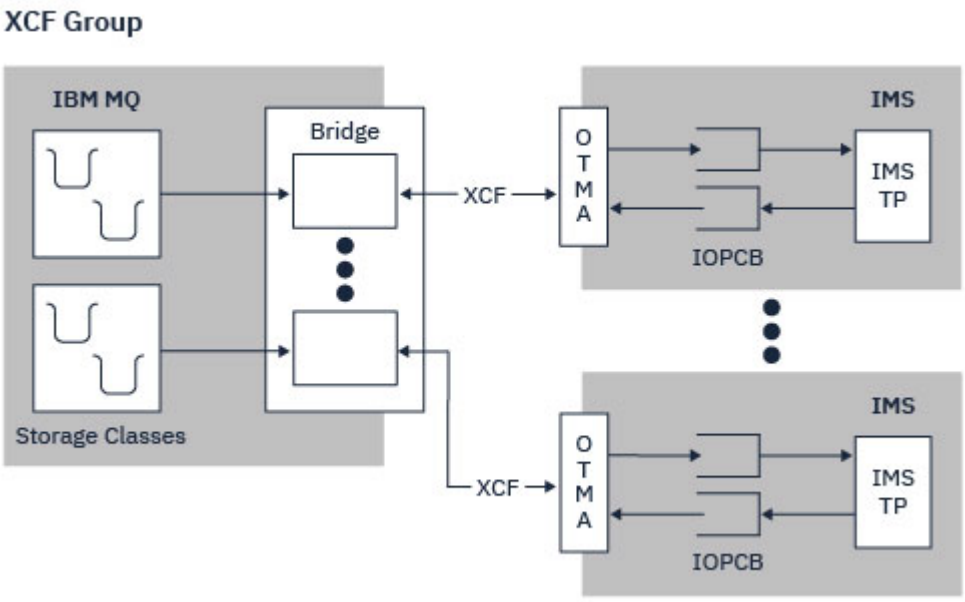


그림 *78. IBM MQ-IMS* 브릿지

하나의 큐 관리자를 하나 이상의 IMS 시스템에 연결할 수 있으며, 둘 이상의 큐 관리자를 하나의 IMS 시스템에 연결할 수 있습니다. 유일한 제한사항은 이들이 모두 동일한 XCF 그룹에 속해야 하며 동일한 sysplex에 있어야 한다는 점입니다.

IMS 브릿지 설정 및 동일한 큐 관리자에 추가 IMS 연결 추가에 대한 정보는 IMS 브릿지 설정 을 참조하십시오.

OTMA의 개념

IMS OTMA 기능은 IMS에서 실행되는 트랜잭션 기반의 무연결 클라이언트/서버 프로토콜입니다. z/OS XCF (Cross Systems Coupling Facility)를 통해 IMS TM 애플리케이션에 액세스하는 호스트 기반 통신 서버의 인터페이스 역할을 합니다.

OTMA에서는 클라이언트가 IMS에 연결하여 대형 네트워크 또는 많은 수의 세션에 대해 클라이언트 및 IMS 간 상호작용을 위해 고성능을 제공할 수 있습니다. OTMA는 z/OS sysplex 환경에서 구현됩니다. 따라서 OTMA의 도메인은 XCF의 도메인으로 제한됩니다.

OTMA 자원 모니터링

IMS v10 이상에서 사용 가능한 x '3C' OTMA 프로토콜 메시지에 대한 지원은 IBM MQ for z/OS의 IBM MQ - IMS 브릿지에 포함되어 있습니다. 이 메시지는 해당 상태를 보고할 수 있도록 IMS를 통해 OTMA 클라이언트로 송신됩니다.

IMS 파트너가 송신되는 트랜잭션 요청 볼륨을 처리할 수 없는 경우 범람 경고가 발생했음을 IBM MQ에 알립니다. 그에 따라 IBM MQ는 브릿지를 통해 요청이 송신되는 속도를 늦춥니다.

IMS가 계속 트랜잭션 요청을 처리할 수 없는 경우 완전 범람 조건이 발생하고 IMS 파트너에 대한 모든 TPIPE가 일시중단됩니다. 범람 또는 범람 경고 조건이 완화되었다는 IMS 파트너의 알림을 받으면 IBM MQ는 일시중단된 모든 TPIPE를 다시 시작하고, 적절하다면 트랜잭션 요청이 송신되는 속도를 최대 속도에 도달할 때까지 점진적으로 높입니다. 콘솔 메시지는 IMS 파트너의 상태 변경에 대한 응답으로 IBM MQ 에서 발행됩니다.

IMS v10 파트너를 사용 중인 경우 PTF UK45082가 적용되었는지 확인해야 합니다.

IBM MQ 에서 IMS 트랜잭션 제출

브릿지를 사용하는 IMS 트랜잭션을 제출하려면 애플리케이션은 보통과 같이 IBM MQ 큐에 메시지를 넣습니다. 메시지는 IMS 트랜잭션 데이터를 포함합니다. 이들은 IMS 헤더(MQIIH 구조)를 포함하거나 IBM MQ-IMS 브릿지에서 메시지의 데이터에 대한 가정을 수행할 수 있습니다.

그런 다음 IBM MQ는 IMS 큐에 메시지를 넣습니다(IBM MQ의 큐에 먼저 넣어 데이터 무결성을 보장하도록 동기점을 사용할 수 있음). IBM MQ 큐의 스토리지 클래스는 큐가 *OTMA* 큐(즉, IBM MQ-IMS 브릿지에 메시지를 전송하는 데 사용되는 큐) 및 메시지 데이터가 송신되는 특정 IMS 파트너인지 여부를 판별합니다.

리모트 큐 관리자는 IBM MQ for z/OS에서 이러한 OTMA큐에 기록하여 IMS 트랜잭션을 시작할 수도 있습니다.

IMS시스템에서 리턴된 데이터는 메시지 디스크립터 구조(MQMD)에 지정된 IBM MQ 응답 대상 큐에 직접 기록됩니다. (이는 MQMD의 **ReplyToQMgr** 필드에 지정된 큐 관리자에 대한 전송 큐일 수 있습니다.)

관련 개념

IBM MQ for z/OS 의 IMS 및 IMS 브릿지 애플리케이션

관련 태스크

IMS 브릿지 사용자 정의

관련 참조

253 페이지의 『IBM MQ 및 IMS』

이 토픽을 사용하여 IBM MQ가 IMS와 함께 작동하는 방법을 이해하십시오. IMS 어댑터를 사용하면 큐 관리자를 IMS에 연결하고 MQI를 사용하도록 IMS 애플리케이션을 사용 가능하게 할 수 있습니다.

z/OS IBM MQ와 z/OS 배치, TSO 및 RRS 어댑터

IBM MQ가 z/OS 배치, TSO 및 RRS 어댑터와 함께 작동하는 방식을 이해하는 데 이 토픽을 사용하십시오.

배치 어댑터에 대한 소개

배치/TSO 어댑터는 JES, TSO 또는 z/OS UNIX System Services에서 실행되는 IBM MQ 및 z/OS 애플리케이션 프로그램 간의 인터페이스입니다. 이 어댑터를 통해 z/OS 애플리케이션 프로그램에서 MQI를 사용할 수 있습니다.

어댑터는 다음 모드나 상태에서 실행 중인 프로그램에 대해 IBM MQ 자원에 대한 액세스를 제공합니다.

- 태스크(TCB) 모드
- 문제점 또는 수퍼바이저 상태
- 비교차 메모리 모드
- 비액세스 등록 모드

애플리케이션 프로그램과 IBM MQ 간의 연결은 태스크 레벨에 존재합니다. 어댑터는 애플리케이션 TCB(Task Control Block)에서 IBM MQ로의 연결 스레드를 제공합니다.

배치/TSO 어댑터는 IBM MQ에서 소유한 자원에서 수행된 변경을 위해 1단계 커미트 프로토콜을 지원합니다. 여기에서는 다중 단계 커미트 프로토콜을 지원하지 않습니다. RRS 어댑터는 IBM MQ 애플리케이션이 z/OS RRS(Resource Recovery Services)에서 통합한 다른 RRS 사용 가능 제품과 함께 2단계 커미트 프로토콜에 참여할 수 있게 합니다.

어댑터는 z/OS STIMERM 서비스를 사용하여 매초마다 비동기 이벤트를 스케줄합니다. 이 이벤트는 배치 애플리케이션의 태스크에서 대기를 포함하지 않는 인터럽트 요청 블록(IRB)을 실행합니다. 이 IRB는 IBM MQ 종료 ECB가 게시되었는지 확인합니다. 종료 ECB가 게시된 경우 IRB는 IBM MQ에서 이벤트를 대기 중인 애플리케이션 ECB를 게시합니다(예: 신호 또는 대기).

배치/TSO 어댑터

IBM MQ Batch/TSO 어댑터는 z/OS Batch및 TSO 애플리케이션에 대한 IBM MQ 지원을 제공합니다. z/OS 배치 또는 TSO 아래에서 실행되는 모든 애플리케이션 프로그램에서는 API 스텁 프로그램 CSQBSTUB가 이들과 링크 편집되어 있어야 합니다. 스텁에서는 모든 MQI 호출에 대한 액세스 권한을 애플리케이션에 제공합니다. MQI 호출 **MQCMIT** 및 **MQBACK**을 실행하여 애플리케이션에 대한 1단계 커미트 및 백아웃을 사용합니다.

RRS 어댑터

RRS(Resource Recovery Services)는 z/OS 제품에서 2단계 커미트를 통합하기 위해 시스템 전체 서비스를 제공하는 z/OS의 서브컴포넌트입니다. IBM MQ Batch/TSO RRS 어댑터 (RRS 어댑터) 는 이러한 서비스를 사용하려는 z/OS 배치 및 TSO 애플리케이션에 대한 IBM MQ 지원을 제공합니다. RRS 어댑터를 사용하면 IBM MQ가 RRS 통합에 완전히 참여할 수 있게 됩니다. 애플리케이션은 RRS를 지원하는 다른 제품(예: Db2)을 통해 2단계 커미트 처리에 참가할 수 있습니다.

RRS 어댑터는 두 개의 스텁을 제공합니다. 이러한 스텁 중 하나에서 RRS를 사용하려는 애플리케이션 프로그램을 링크-편집해야 합니다.

CSQBRSTB
이 스텁에서는 MQI 호출 **MQCMIT** 및 **MQBACK** 대신 RRS 호출 가능 자원 복구 서비스를 사용하여 애플리케이션에 대한 2단계 커미트 및 백아웃을 사용할 수 있습니다.

또한 라이브러리 SYS1.CSSLIB의 모듈 ATRSCSS를 애플리케이션과 링크-편집해야 합니다. MQI 호출 **MQCMIT** 및 **MQBACK**를 사용하는 경우 리턴 코드 `MQRC_ENVIRONMENT_ERROR`를 수신합니다.

CSQBRRSI
이 스텁을 사용하면 MQI 호출 **MQCMIT** 및 **MQBACK**을 사용할 수 있습니다. IBM MQ는 실제로 이러한 호출을 **SRRCMIT** 및 **SRRBACK** RRS 호출로 구현합니다.

RRS 어댑터를 사용하는 애플리케이션 프로그램의 빌드에 대한 정보는 RRS 배치 어댑터를 참조하십시오.

z/OS 배치, TSO 및 RRS 어댑터에 대한 자세한 정보를 찾을 위치

다음 소스에서 이 절의 토픽에 대한 자세한 정보를 찾을 수 있습니다.

표 *25. z/OS Batch with IBM MQ* 사용에 대한 자세한 정보를 찾을 수 있는 위치

주제	찾을 위치
배치 어댑터 설정	태스크 19: 배치, TSO 및 RRS 어댑터 설정
RRS 호출 가능 자원 복구 서비스	*MVS* 프로그래밍*:* 상위 레벨 언어에 대한 호출 가능 서비스

z/OS IBM MQ for z/OS 및 WebSphere Application Server

이 토픽을 사용하여 WebSphere Application Server의 IBM MQ for z/OS 사용을 이해하십시오.

WebSphere Application Server 에서 실행 중인 Java 로 작성된 애플리케이션은 Java Message Service (JMS) 스펙을 사용하여 메시징을 수행할 수 있습니다. 이 환경에서 포인트-투-포인트 메시징은 IBM MQ for z/OS 큐 관리자가 제공할 수 있습니다.

메시지 제공을 위해 IBM MQ for z/OS 큐 관리자를 사용하여 얻을 수 있는 혜택은 JMS 애플리케이션과 연결하여 IBM MQ 네트워크 기능성에 완전히 참여할 수 있다는 것입니다. 예를 들어, IMS 브릿지를 사용하거나 다른 플랫폼에서 실행 중인 큐 관리자와 메시지를 교환할 수 있습니다.

WebSphere Application Server 및 큐 관리자 사이의 연결

자세한 정보는 IBM MQ 및 WebSphere Application Server 함께 사용 을 참조하십시오.

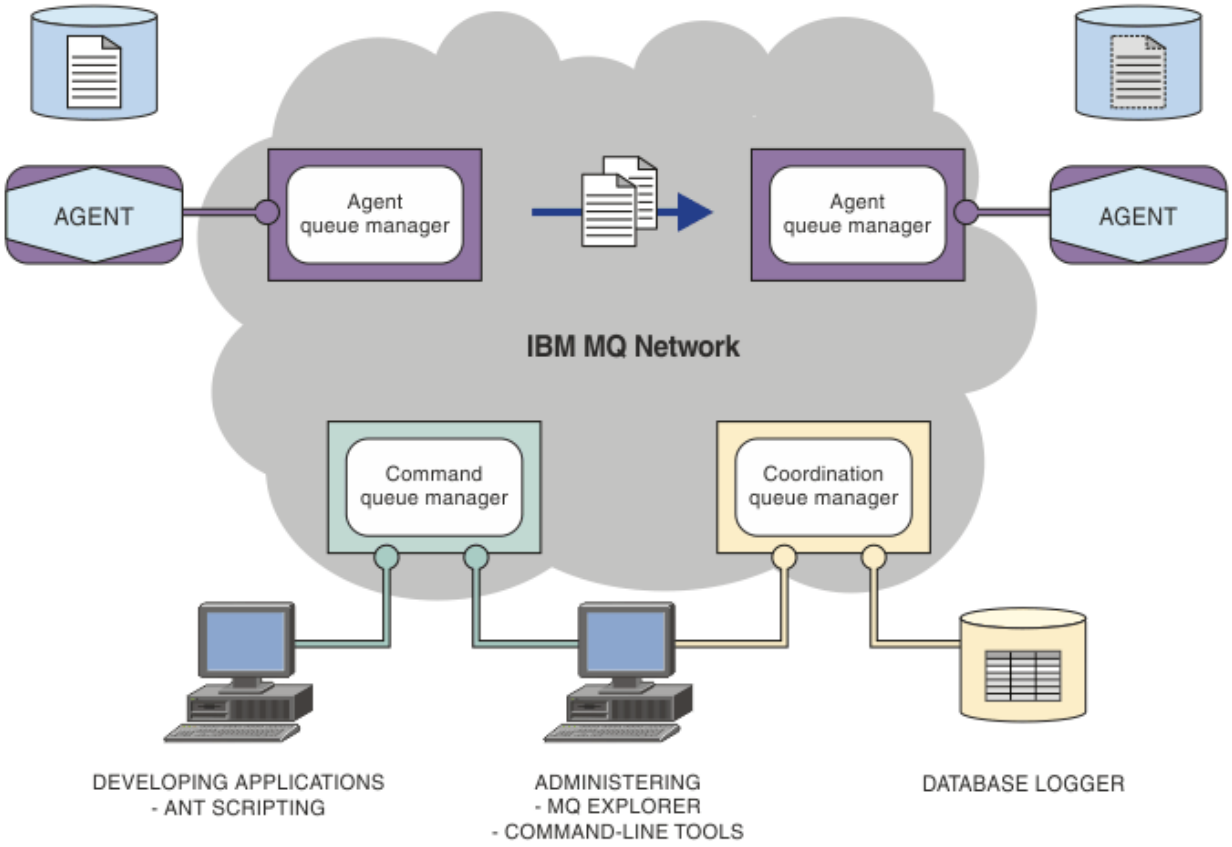
JMS 애플리케이션에서 IBM MQ 함수 사용

기본적으로 IBM MQ 큐에 보유된 JMS 메시지는 MQRFH2 헤더를 사용하여 일부 JMS 메시지 헤더 정보를 보유합니다. 많은 레거시 IBM MQ 애플리케이션은 이러한 헤더를 포함하는 메시지를 처리할 수 없으며 고유한 특성 헤더(예: CICS 브릿지의 경우 MQCIH, IBM MQ 워크플로우 애플리케이션의 경우 MQWIH)가 필요합니다. 이러한 특수 고려사항에 대한 자세한 정보는 JMS 메시지를 IBM MQ 메시지에 맵핑을 참조하십시오.

Managed File Transfer

Managed File Transfer는 파일 크기 또는 사용되는 운영 체제에 관계 없이 관리되고 감사 가능한 방법으로 시스템 간에 파일을 전송합니다.

Managed File Transfer를 사용하여 파일 전송을 관리하고 신뢰 가능하며 보안을 유지할 수 있게 해주는 사용자 정의되고 확장 가능하며 자동화된 솔루션을 빌드할 수 있습니다. Managed File Transfer는 많은 비용이 드는 불필요한 중복을 제거하고 유지보수 비용을 낮추며 기존 IT 환경을 최대한 활용합니다.



다이어그램에서는 간단한 Managed File Transfer 토폴로지를 보여줍니다. 두 개의 에이전트가 있으며 각각 IBM MQ 네트워크 내에서 해당하는 고유 에이전트 큐 관리자에 연결됩니다. 파일은 다이어그램의 한쪽에 있는 에이전트에서 IBM MQ 네트워크를 통해 다이어그램의 다른 쪽에 있는 에이전트로 전송됩니다. 또한 IBM MQ 네트워크에는 조정 큐 관리자와 명령 큐 관리자가 있습니다. 애플리케이션 및 도구는 이러한 큐 관리자에 연결하여 IBM MQ 네트워크에서 Managed File Transfer 활동을 구성, 관리, 조작 및 로깅합니다.

Managed File Transfer는 운영 체제 및 전체 설정에 따라 네 가지의 다른 옵션으로 설치할 수 있습니다. Managed File Transfer Agent, Managed File Transfer Logger, Managed File Transfer Service 또는 Managed File Transfer Tools가 이러한 옵션에 해당합니다. 자세한 정보는 Managed File Transfer 제품 옵션 을 참조하십시오.

Managed File Transfer를 사용하여 다음 태스크를 수행할 수 있습니다.

- 관리 파일 전송을 작성합니다.
  -  Linux 또는 Windows 플랫폼의 IBM MQ Explorer 에서 새 파일 전송을 작성하십시오.
  - 지원되는 모든 플랫폼의 명령행에서 새 파일 전송을 작성합니다.
  - 파일 전송 기능을 Apache Ant 도구에 통합하십시오.
  - 에이전트 명령 큐에 메시지를 넣어 Managed File Transfer를 제어하는 애플리케이션을 씁니다.
  - 나중에 수행할 파일 전송을 스케줄합니다. 또한 파일 시스템 이벤트(예: 작성된 새 파일) 등을 기반으로 하여 스케줄된 파일 전송을 트리거할 수 있습니다.
  - 디렉토리와 같은 자원을 계속 모니터하며 해당 자원의 컨텐츠가 일부 사전정의된 조건을 충족하는 경우 태스크를 시작합니다. 이 태스크는 파일 전송, Ant 스크립트 또는 JCL 작업일 수 있습니다.
  - IBM MQ 큐로나 큐로부터 파일을 전송합니다.
  - FTP, FTPS 또는 SFTP 서버 간에 파일을 전송합니다.
  - Connect:Direct® 노드 간 파일 전송
  - 텍스트와 2진 파일 모두를 전송합니다. 텍스트 파일은 소스 및 목적지 시스템의 행의 끝 규칙과 코드 페이지 사이에서 자동 변환됩니다.
  - SSL(Secure Socket Layer) 기반 연결에 대한 산업 표준을 사용하여 전송 시 보안을 설정할 수 있습니다.
- 진행 중인 전송을 보고, 네트워크의 모든 전송에 대한 정보를 로그합니다.
  -  Linux 또는 Windows 플랫폼의 IBM MQ Explorer 에서 진행 중인 전송 상태를 보십시오.
  -  Linux 또는 Windows 플랫폼에서 IBM MQ Explorer 를 사용하여 완료된 전송의 상태를 확인하십시오.
  - Managed File Transfer 데이터베이스 로거 기능을 사용하여 Db2 또는 Oracle 데이터베이스에 로그 메시지를 저장합니다.

Managed File Transfer는 애플리케이션 사이에서 문제 없이 메시지를 확실히 전달하는 IBM MQ에 빌드됩니다. 사용자는 IBM MQ의 다양한 기능을 활용할 수 있습니다. 예를 들어, 채널 압축을 사용하여 IBM MQ 채널을 통해 에이전트 간에 보내는 데이터를 압축하고 SSL 채널을 사용하여 에이전트 간에 보내는 데이터의 보안을 유지할 수 있습니다. 파일이 안정적으로 전송되고 실행된 파일 전송이 수행되는 인프라의 실패에 영향을 받지 않을 수 있습니다. 네트워크의 연결이 끊긴 경우, 연결이 복원되면 파일 전송이 중지된 위치에서 재시작됩니다.

파일 전송을 기존 IBM MQ 네트워크와 통합하면 두 개의 개별 인프라를 유지보수하는 데 필요한 자원의 소비를 피할 수 있습니다. IBM MQ 고객이 아닌 경우 IBM MQ 네트워크를 작성하여 Managed File Transfer를 지원함으로써 나중에 SOA를 구현하기 위한 백본을 빌드합니다. 이미 IBM MQ 고객인 경우 Managed File Transfer 는 IBM MQ Internet Pass-Thru 및 IBM Integration Bus를 포함한 기존 IBM MQ 인프라를 이용할 수 있습니다.

IBM MQ 고가용성 솔루션을 활용하여 Managed File Transfer 구성의 복원성을 개선할 수 있습니다. 에이전트가 복제된 데이터 큐 관리자 (RDQM) 를 사용하는 경우 부동 IP 주소 기능을 사용하도록 구성해야 합니다. 이는 에이전트가 동일한 IP 주소를 사용하여 현재 실행 중인 세 개의 RDQM 인스턴스 중 하나와 통신하고 장애 복구 시 자동으로 다시 연결함을 의미합니다 (RDQM 고가용성 및 부동 IP 주소 작성 및 삭제참조). 다중 인스턴스 큐 관리자 솔루션을 사용하는 경우 애플리케이션은 서로 다른 IP 주소를 사용하여 각 인스턴스와 통신하며, 이는 장애 복구 시 클라이언트 다시 연결에 의해 처리됩니다 (다중 인스턴스 큐 관리자 및 채널 및 클라이언트 다시 연결참조).

Managed File Transfer는 다양한 기타 IBM 제품과 통합됩니다.

IBM Integration Bus

IBM Integration Bus 플로우의 일부로 Managed File Transfer 에 의해 전송된 파일을 처리합니다. 자세한 정보는 IBM Integration Bus에서 MFT 에 대한 작업을 참조하십시오.

IBM Sterling™ Connect:Direct

Managed File Transfer Connect:Direct 브릿지를 사용하여 기존 Connect:Direct 네트워크 간에 파일을 전송합니다. 자세한 정보는 Connect:Direct 브릿지를 참조하십시오.

IBM Tivoli® Composite Application Manager

IBM Tivoli Composite Application Manager가 조정 큐 관리자에 발행된 정보를 모니터하는 데 사용할 수 있는 에이전트를 제공합니다.

관련 개념

Managed File Transfer 제품 옵션

260 페이지의 『MFT 토폴로지 개요』
Managed File Transfer 에이전트를 IBM MQ 네트워크의 조정 큐 관리자와 연결하는 방법에 대한 개요입니다.

260 페이지의 『MFT가 IBM MQ에 대해 작업하는 방법』
Managed File Transfer는 IBM MQ와 다양한 방법으로 상호작용합니다.

MFT가 IBM MQ에 대해 작업하는 방법

Managed File Transfer는 IBM MQ와 다양한 방법으로 상호작용합니다.

- Managed File Transfer는 각 파일을 하나 이상의 메시지로 나누고 IBM MQ 네트워크를 통해 메시지를 전송하여 에이전트 프로세스 간에 파일을 전송합니다.
- 에이전트 프로세스는 IBM MQ 로그에 대한 영향을 최소화하기 위해 비지속 메시지를 사용하여 파일 데이터를 이동합니다. 에이전트 프로세스는 다른 프로세스와 통신하여 파일 데이터가 포함된 메시지 플로우를 조절합니다. 이는 IBM MQ 전송 큐에 빌드되는 파일 데이터가 메시지에 포함되는 것을 막으며 비지속 메시지가 전달되지 않는 경우 파일 데이터가 다시 송신되도록 합니다.
- Managed File Transfer 에이전트는 다수의 IBM MQ 큐를 사용합니다. 자세한 정보는 MFT 시스템 큐 및 시스템 토픽을 참조하십시오.
- 이러한 큐 중 일부는 내부용으로 엄격하게 제한되어 있지만 에이전트는 에이전트가 읽는 특정 큐로 송신된 특별하게 형식화된 명령 메시지 양식의 요청을 승인할 수 있습니다. 명령행 명령 및 IBM MQ Explorer 플러그인 모두 IBM MQ 메시지를 에이전트에 전송하여 에이전트가 원하는 조치를 수행하도록 지시합니다. 이러한 방법으로 에이전트와 상호작용하는 IBM MQ 애플리케이션을 작성할 수 있습니다. 자세한 정보는 에이전트 명령 큐에 메시지를 넣어 MFT 제어를 참조하십시오.
- Managed File Transfer 에이전트는 조정 큐 관리자로 지정된 MQ 큐 관리자에 대한 전송의 진행 및 결과와 해당 상태에 대한 정보를 송신합니다. 이 정보는 조정 큐 관리자에 의해 발행되며 전송 진행 상태를 모니터하거나 일어난 전송을 기록할 애플리케이션에 의해 구독될 수 있습니다. 명령행 명령 및 IBM MQ Explorer 플러그인은 둘 다 발행되는 정보를 사용할 수 있습니다. 이 정보를 사용하는 IBM MQ 애플리케이션을 작성할 수 있습니다. 정보가 발행되는 토픽에 대한 자세한 정보는 SYSTEM.FTE 주제.
- Managed File Transfer의 주요 컴포넌트는 IBM MQ 큐 관리자의 기능을 활용하여 메시지를 저장하고 전달합니다. 이는 정지 상태가 되는 경우, 이에 영향을 받지 않는 인프라의 일부가 계속 파일을 전송할 수 있음을 의미합니다. 이러한 내용은 조정 큐 관리자에게도 해당되어, 저장 및 전달의 조합 및 지속 가능 구독을 사용하면 조정 큐 관리자가 발생한 파일 전송에 관한 핵심 정보를 잃지 않고 사용 불가능 상태에서도 작동할 수 있습니다.

MFT 토폴로지 개요

Managed File Transfer 에이전트를 IBM MQ 네트워크의 조정 큐 관리자와 연결하는 방법에 대한 개요입니다.

Managed File Transfer 에이전트는 전송되는 파일을 송수신합니다. 각 에이전트는 연관된 큐 관리자에 고유한 큐 세트를 가지고 있으며 에이전트는 바인딩 또는 클라이언트 모드로 큐 관리자에 연결됩니다. 에이전트는 조정 큐 관리자를 큐 관리자로 사용할 수도 있습니다.

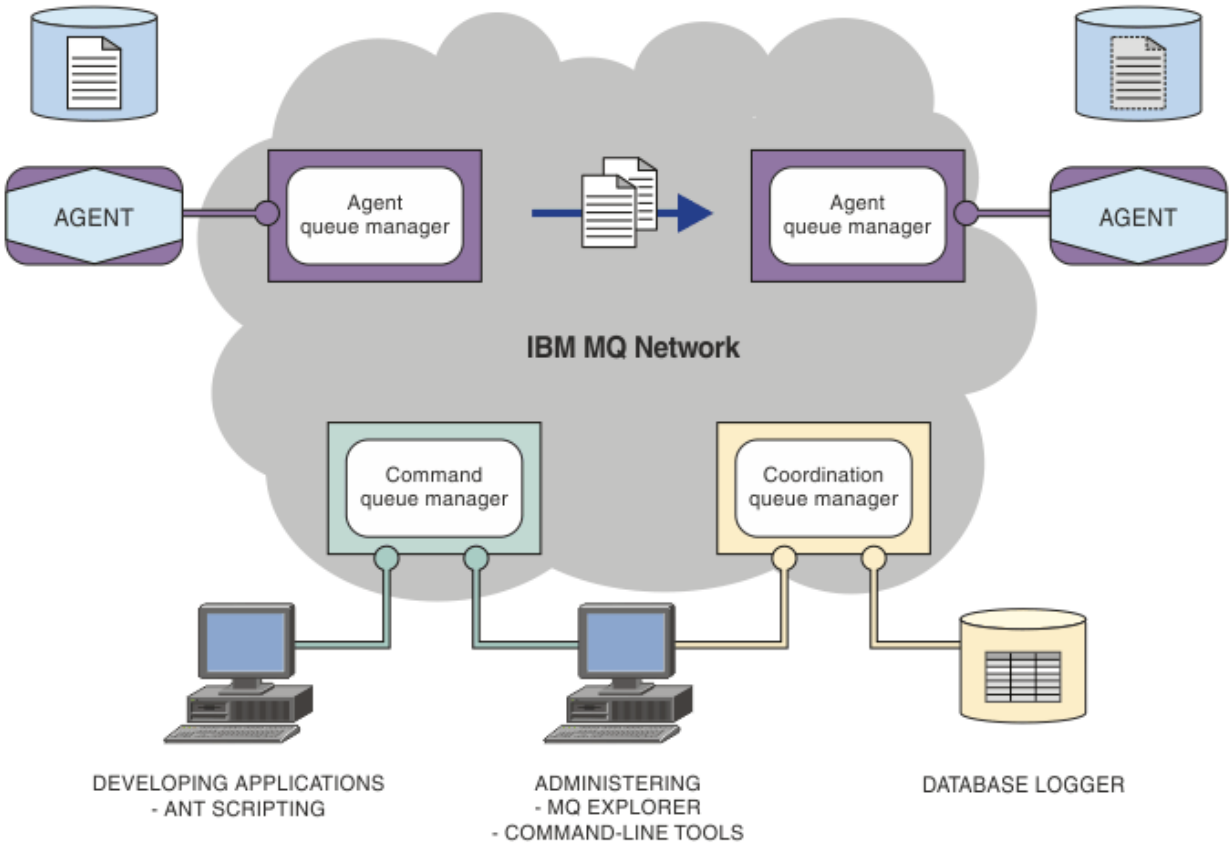
조정 큐 관리자는 감사 및 파일 전송 정보를 브로드캐스트합니다. 조정 큐 관리자는 에이전트, 전송 상태 및 전송 감사 정보 콜렉션의 단일 지점을 나타냅니다. 전송이 이루어지도록 하기 위해 조정 큐 관리자가 사용 가능해야 하는 것은 아닙니다. 조정 큐 관리자를 일시적으로 사용할 수 없는 경우에도 전송은 정상적으로 계속 됩니다. 조정

큐 관리자가 사용 가능하게 되어 정상적으로 처리될 수 있을 때까지 감사 및 상태 메시지가 에이전트 큐 관리자에 저장됩니다.

에이전트는 조정 큐 관리자와 함께 등록되며 세부사항을 해당 큐 관리자에 발행합니다. 이 에이전트 정보는 Managed File Transfer 플러그인이 IBM MQ Explorer에서 전송 시작을 사용으로 설정하는 데 사용됩니다. 또한 에이전트 정보 및 에이전트 상태를 표시하는 명령이 조정 큐 관리자에 수집된 에이전트 정보를 사용하기도 합니다.

전송 상태 및 전송 감사 정보는 조정 큐 관리자에서 발행됩니다. 전송 상태 및 전송 감사 정보는 Managed File Transfer 플러그인이 IBM MQ Explorer에서 전송 진행 상황을 모니터하는 데 사용됩니다. 조정 큐 관리자에 저장된 전송 감사 정보는 감사 가능성을 제공하기 위해 보유할 수 있습니다.

명령 큐 관리자는 IBM MQ 네트워크에 연결하는 데 사용되며 Managed File Transfer 명령을 실행할 때 연결된 큐 관리자입니다.



관련 개념

258 페이지의 『Managed File Transfer』
Managed File Transfer는 파일 크기 또는 사용되는 운영 체제에 관계 없이 관리되고 감사 가능한 방법으로 시스템 간에 파일을 전송합니다.

260 페이지의 『MFT가 IBM MQ에 대해 작업하는 방법』
Managed File Transfer는 IBM MQ와 다양한 방법으로 상호작용합니다.

Managed File Transfer 시나리오

MFT REST API 개요

REST API는 파일 전송 에이전트에 대한 세부사항 및 전송 목록을 포함하여 특정 Managed File Transfer 명령을 지원합니다.

IBM MQ 9.1.0부터 REST API는 현재 모든 Managed File Transfer 전송을 나열하고 Managed File Transfer 에이전트의 상태를 조회하는 옵션을 포함합니다. 자세한 정보는 REST API MFT 시작하기를 참조하십시오.

IBM MQ Internet Pass-Thru

IBM MQ Internet Pass-Thru(MQIPT)는 인터넷을 통한 원격 사이트 간 메시징 솔루션을 구현하는 데 사용할 수 있는, IBM MQ의 선택적 컴포넌트입니다.

IBM MQ 9.2.0부터는 MQIPT가 IBM MQ의 선택적 컴포넌트입니다. IBM MQ 9.3.x에 대한 MQIPT 설치 파일을 얻으려면 IBM MQ 용 IBM Fix Central로 이동하십시오. IBM MQ 9.2.0이전에는 MQIPT를 지원 팩으로 사용할 수 있었습니다.

IBM MQ 9.3에서 MQIPT 를 사용하기 위해 IBM MQ 9.3 를 실행할 필요가 없습니다. MQIPT 를 사용하여 지원되는 버전의 IBM MQ를 연결할 수 있으며, MQIPT와 동일한 버전의 다른 IBM MQ 컴포넌트를 설치할 필요가 없습니다.

IBM MQ 인타이틀먼트를 구매한 경우 필요한 만큼 MQIPT의 사본을 설치할 수 있습니다. MQIPT 설치는 구매한 IBM MQ 인타이틀먼트로 계수되지 않습니다. IBM MQ 라이센싱에 대한 자세한 정보는 IBM MQ 라이센스 정보를 참조하십시오.

참고: 이 문서는 IBM MQ 9.3의 MQIPT 와 관련되어 있습니다. IBM Documentation의 MQIPT 지원 팩 (버전 2.1) 문서는 IBM MQ 9.0 문서의 MQIPT (SupportPac MS81) 을 참조하십시오.

참고: MQIPT 2.1 또는 이전 버전을 사용하는 경우 IBM MQ 9.3용 MQIPT 로 업그레이드하는 것이 좋습니다. MQIPT 지원 팩에 대한 지원 종료 날짜가 2020년 9월 30th 이기 때문입니다.

IBM MQ Internet Pass-Thru는 두 개의 IBM MQ 큐 관리자 또는 IBM MQ 클라이언트와 IBM MQ 사이에서 IBM MQ 메시지 플로우를 수신 및 전달할 수 있는 독립형 서비스로 실행됩니다.

클라이언트 및 서버가 동일한 물리적 네트워크에 존재하지 않을 경우 MQIPT에서 이 연결을 사용으로 설정합니다.

하나 이상의 MQIPT 인스턴스를 두 개의 IBM MQ 큐 관리자 또는 IBM MQ 클라이언트와 IBM MQ 큐 관리자 간의 통신 경로에 배치할 수 있습니다. MQIPT 인스턴스를 사용하는 경우 두 개의 시스템 간에 직접 TCP/IP 연결 없이도 두 개의 IBM MQ 시스템에서 메시지를 교환할 수 있습니다. 이 기능은 방화벽 구성에서 두 시스템 간의 직접 TCP/IP 연결을 금지하는 경우에 유용합니다.

MQIPT는 하나 이상의 TCP/IP 포트에서 일반 IBM MQ 메시지, HTTP 내에서 터널링된 IBM MQ 메시지 또는 TLS(Transport Layer Security)나 SSL(Secure Sockets Layer)을 사용하여 암호화된 메시지를 전달할 수 있는 수신 연결을 청취합니다. MQIPT는 복수의 동시 연결을 핸들링할 수 있습니다.

초기 TCP/IP 연결 요청을 작성하는 IBM MQ 채널은 호출자로 참조되고, 연결을 시도하는 대상 채널은 응답자로 참조되며, 최종적으로 연결을 시도하는 대상 큐 관리자는 대상 큐 관리자로 참조됩니다.

MQIPT는 소스에서 목적지로 전달할 때 메모리에 데이터를 보관합니다. 디스크에는 데이터가 저장되지 않습니다(운영 체제에서 디스크에 페이징하는 메모리는 제외). 구성 파일을 읽어들인 후 연결 로그 및 추적 레코드를 작성하는 경우에만 MQIPT에서 명시적으로 디스크에 액세스합니다.

IBM MQ 채널 유형의 전체 범위는 하나 이상의 MQIPT 인스턴스를 통해 작성할 수 있습니다. 통신 경로에 MQIPT가 존재하는 경우 연결된 IBM MQ 컴포넌트의 기능적인 특성에는 영향을 미치지 않지만 메시지 전송의 성능에 약간의 영향이 있을 수 있습니다.

MQIPT 는 265 페이지의 『MQIPT의 가능한 구성』 에 설명된 대로 IBM MQ 및 IBM Integration Bus와 함께 사용할 수 있습니다.

MQIPT를 설치하려면 MQIPT 설치를 참조하십시오.

관련 태스크

IBM MQ Internet Pass-Thru 구성

IBM MQ Internet Pass-Thru 관리 및 구성

관련 참조

IBM MQ Internet Pass-Thru 구성 참조

MQIPT의 사용법

IBM MQ Internet Pass-Thru(MQIPT)에 대한 다양한 잠재적 사용법이 존재합니다.

MQIPT를 채널 집선기로 사용할 수 있음

이 방법으로 MQIPT를 사용하는 경우 방화벽에 독립적인 복수의 호스트와 통신하는 채널을 모두 MQIPT 호스트와 통신하는 것처럼 표시할 수 있습니다. 이 경우 더 간단히 방화벽 필터링 규칙을 정의하고 관리할 수 있습니다.

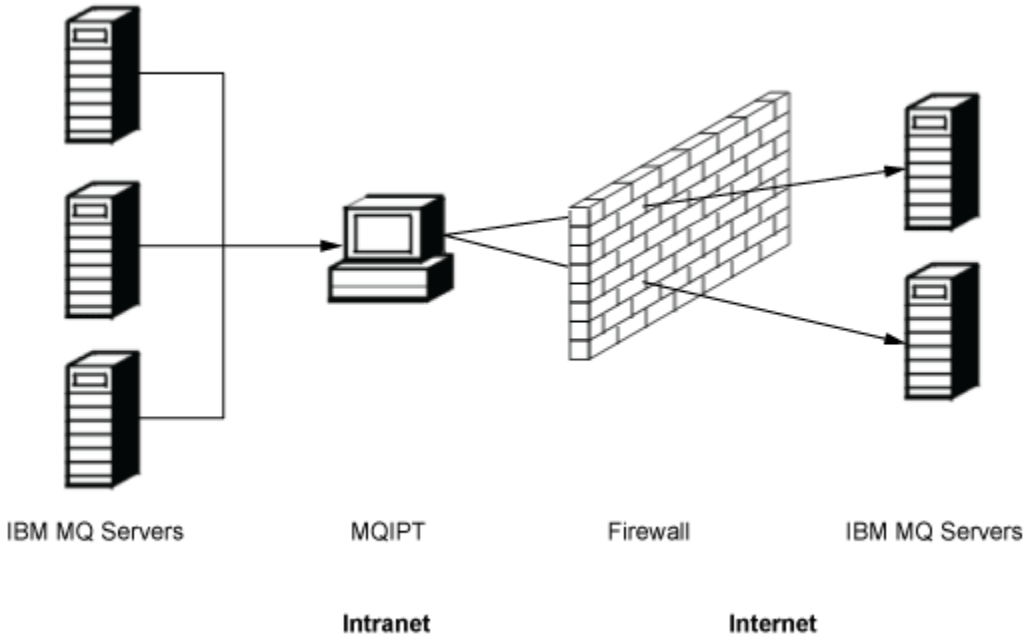


그림 *79.* 채널 집선기로 사용하는 *MQIPT* 예제

단일 액세스 지점을 제공하기 위해 MQIPT를 DMZ에 배치할 수 있음

DMZ 방화벽(때로는 "비무장 지대" 방화벽이라고도 함)은 근거리 통신망을 보호하기 위한 방화벽 구성입니다. 그것은 일반적으로 인터넷과 같은 신뢰할 수 없는 네트워크에 대한 노출 지점 역할을 합니다. DMZ 방화벽에 MQIPT 가 배치되어 있고, 인터넷 프로토콜(IP) 주소가 알려진 신뢰할 수 있는 컴퓨터에 있는 경우, MQIPT 를 사용하여 들어오는 IBM MQ 채널 연결을 수신 대기한 다음, 신뢰할 수 있는 인트라넷으로 전달할 수 있습니다. 내부 방화벽은 이 신뢰할 수 있는 컴퓨터가 인바운드 연결을 할 수 있도록 허용해야 합니다. 이 구성에서는 MQIPT가 액세스에 대한 외부 요청이 신뢰할 수 있는 인트라넷에 있는 컴퓨터의 실제 IP 주소를 수신할 수 없도록 차단합니다. 이러한 방식으로 MQIPT는 단일 액세스 지점을 제공합니다. 필요한 경우, MQIPT 는 TLS 연결을 승인하고 별도의 TLS 연결을 사용하여 데이터를 대상으로 전달하도록 구성할 수 있으므로 DMZ에서 TLS 세션을 종료합니다.

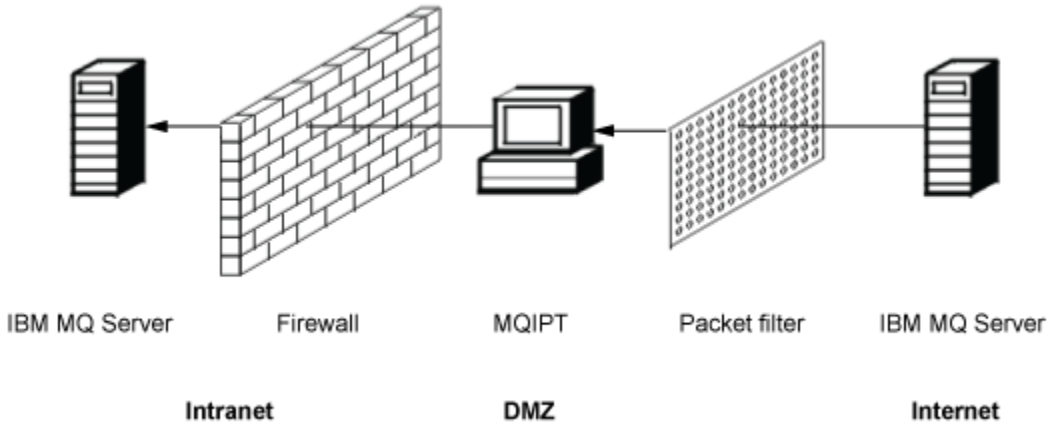


그림 *80. DMZ* 방화벽의 *MQIPT* 예제

MQIPT는 HTTP 터널링 방식으로 통신할 수 있음

두 개의 MQIPT 인스턴스가 나란히 배치되어 있는 경우 해당 인스턴스는 HTTP를 사용하여 통신할 수 있습니다. HTTP 터널링 기능을 사용하는 경우 기존 HTTP 프록시를 사용하여 방화벽을 통해 요청을 전송할 수 있습니다. 첫 번째 MQIPT는 IBM MQ 프로토콜을 HTTP에 삽입하고, 두 번째는 해당 HTTP 랩퍼에서 IBM MQ 프로토콜을 추출한 후 해당 프로토콜을 목적지 큐 관리자로 전달합니다.

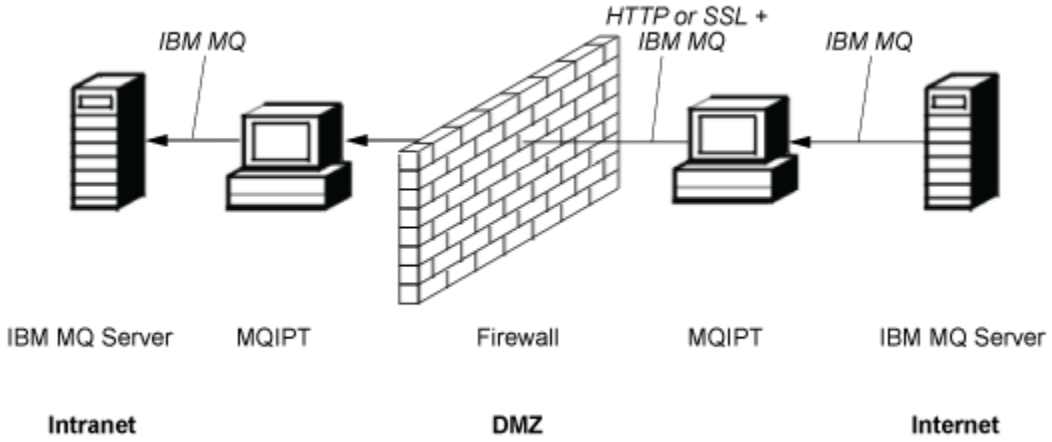


그림 *81. MQIPT* 및 *HTTP* 터널링 예제

MQIPT는 메시지를 암호화할 수 있음

MQIPT가 이전 예제와 같이 구성된 경우 방화벽을 통해 전송되기 전에 요청을 암호화할 수 있습니다. 첫 번째 MQIPT는 데이터를 암호화하고 두 번째는 목적지 큐 관리자로 송신하기 전에 SSL/TLS를 사용하여 데이터를 복호화합니다.

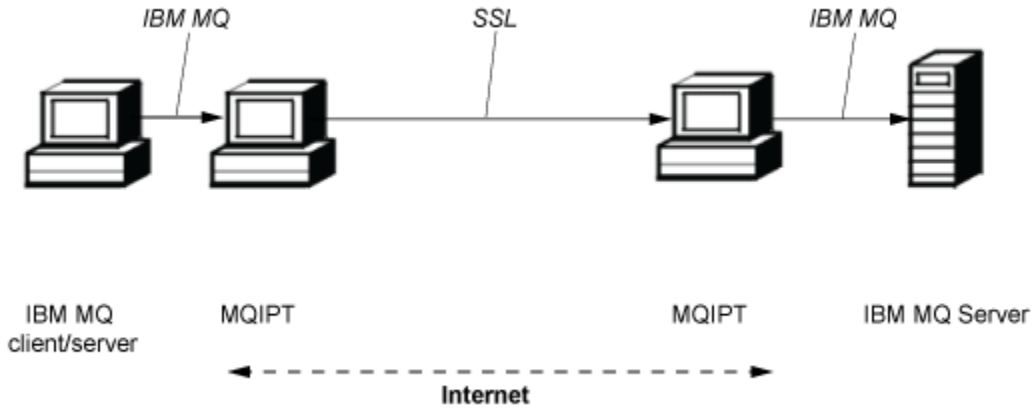


그림 *82. MQIPT* 및 *SSL/TLS* 예제

MQIPT의 작동 방식

가장 단순한 구성에서 MQIPT는 IBM MQ 프로토콜 전달자로 작동합니다. TCP/IP 포트를 청취하고 IBM MQ 채널로부터의 연결 요청을 승인합니다.

올바르게 구성된 요청이 수신되는 경우 MQIPT는 자신과 목적지 IBM MQ 큐 관리자 사이에 추가적인 TCP/IP 연결을 설정합니다. 그런 다음 해당 수신 연결에서 수신되는 모든 프로토콜 패킷을 목적지 큐 관리자로 전달한 후 목적지 큐 관리자의 프로토콜 패킷을 다시 원래 수신 연결로 리턴합니다.

어느 쪽에서도 중개자의 존재를 직접 인식하지 못하기 때문에 IBM MQ 프로토콜에 대한 변경사항(클라이언트/서버 또는 큐 관리자에서 큐 관리자로)이 포함되지 않습니다. 새 버전의 IBM MQ 클라이언트 또는 서버 코드는 필요하지 않습니다.

MQIPT를 사용하려면 목적지 큐 관리자의 호스트 이름 및 포트가 아닌 MQIPT 호스트 이름 및 포트를 사용하도록 호출자 채널이 구성되어야 합니다. 이는 IBM MQ 채널의 **CONNAME** 특성으로 정의됩니다. MQIPT는 수신 데이터를 읽어들인 후 단순히 목적지 큐 관리자로 전달합니다. 클라이언트/서버 채널의 사용자 ID 및 비밀번호와 같은 다른 구성 필드도 마찬가지로 목적지 큐 관리자로 전달됩니다.

다중 큐 관리자

MQIPT를 사용하여 둘 이상의 목적지 큐 관리자에 대한 액세스를 허용할 수 있습니다. 이 작업을 수행하려면 MQIPT에 연결할 큐 관리자를 지정하는 메커니즘이 존재해야 하기 때문에 MQIPT는 수신 TCP/IP 포트 번호를 사용하여 연결할 큐 관리자를 판별합니다.

따라서 복수의 TCP/IP 포트를 청취하도록 MQIPT를 구성할 수 있습니다. 각각의 청취 포트는 MQIPT 라우트를 통해 목적지 큐 관리자에 맵핑됩니다. 이러한 라우트는 최대 100개까지 정의할 수 있으며 청취 TCP/IP 포트를 목적지 큐 관리자의 호스트 이름 및 포트와 연관시킵니다. 즉, 목적지 큐 관리자의 호스트 이름(IP 주소)은 원래 채널에 절대 표시되지 않습니다. 각각의 라우트는 해당 청취 포트와 목적지 사이에서 복수의 연결을 핸들링할 수 있으며, 각각의 연결은 독립적으로 수행됩니다.

MQIPT 구성 파일

MQIPT는 `mqipt.conf`라는 구성 파일을 사용합니다. 이 파일에는 모든 라우트 정의 및 연관된 특성이 포함되어 있습니다. `mqipt.conf`에 대한 자세한 정보는 IBM MQ Internet Pass-Thru 관리 및 구성을 참조하십시오.

MQIPT가 시작되면 구성 파일에 나열된 각각의 라우트가 시작됩니다. 시스템 콘솔에는 각 라우트의 상태를 표시하는 메시지가 기록됩니다. 라우트에 대해 MQCPI078 메시지가 표시되면 해당 라우트에서 연결 요청을 승인할 준비가 된 것입니다.

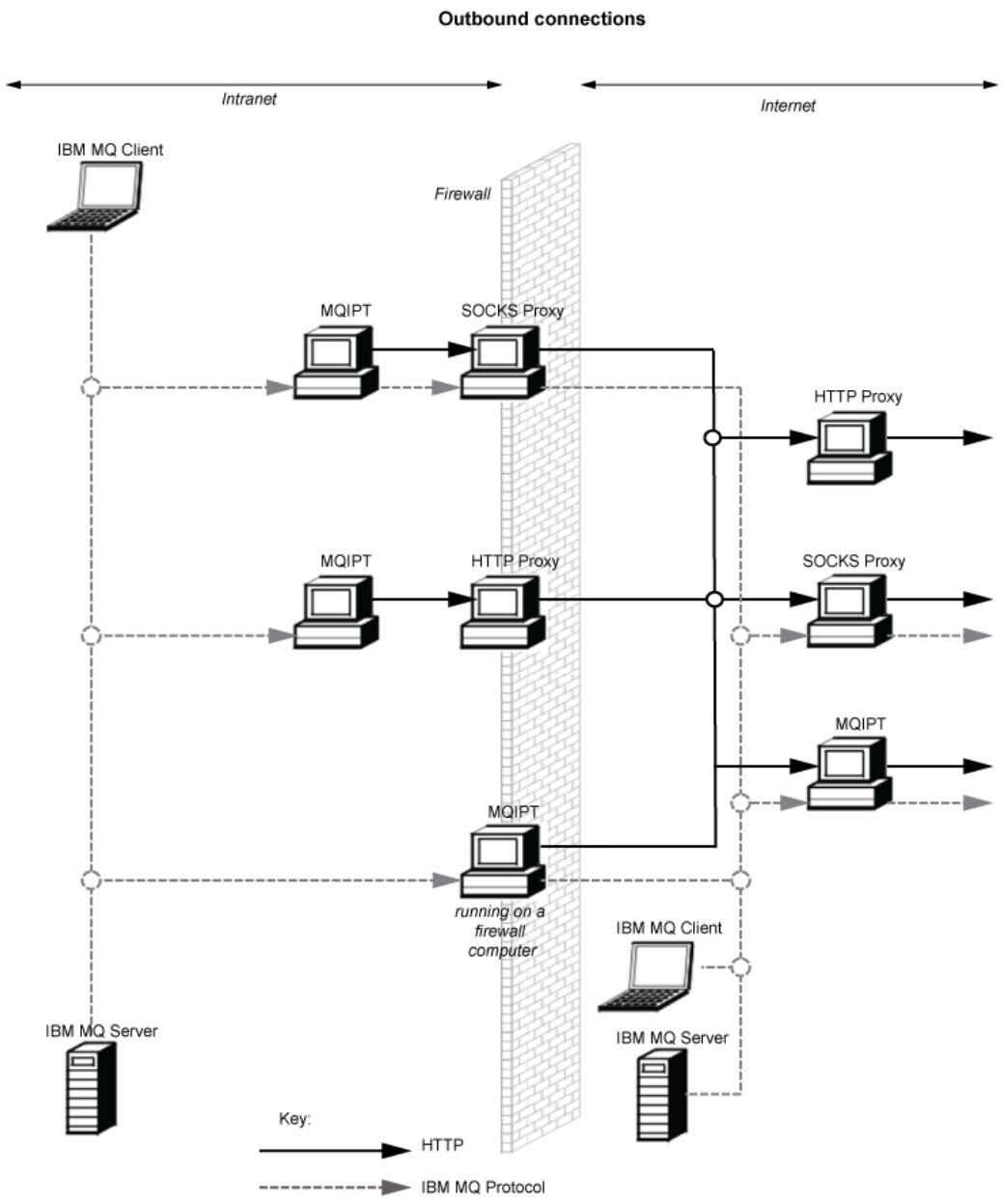
MQIPT의 가능한 구성

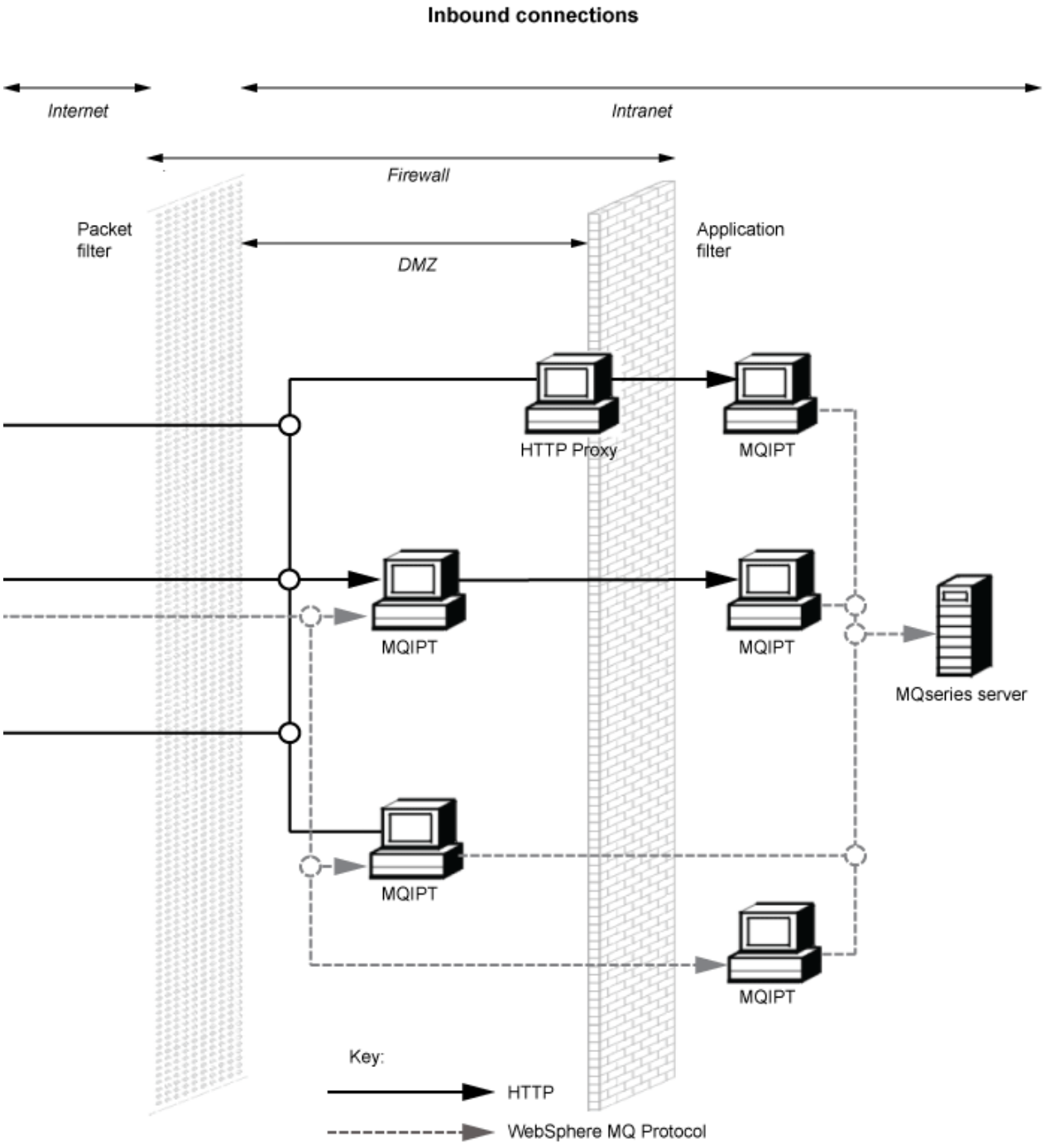
MQIPT는 IBM MQ 및 IBM Integration Bus과 함께 사용할 수 있습니다.

다음 다중 파트 그림은 IBM MQ 토폴로지에서 MQIPT 에 대해 가능한 많은 구성을 보여줍니다. 이 그림은 MQIPT에서 메시지를 송신하는 다양한 방법을 보여줍니다. MQIPT, HTTP 프록시 또는 SOCKS 프록시로 메시지를 전달하여 인트라넷, 방화벽 내부 및 방화벽 외부의 인터넷에서 해당 메시지를 전달하는 클라이언트 및 서버를 보여줍니다.

메시지는 인바운드 방화벽을 통해 서버로 전달하기 전에 DMZ의 MQIPT 프록시 또는 HTTP 프록시에서 수신됩니다.

방화벽의 인트라넷 측에 있는 HTTP 프록시, SOCKS 프록시 및 MQIPT 컴퓨터는 인터넷에서 서로 연결되어 있는 복수의 컴퓨터가 존재할 가능성을 나타냅니다. 예를 들어 MQIPT 컴퓨터는 대상에 도달하기 전에 하나 이상의 SOCKS 또는 HTTP 프록시 컴퓨터나 추가적인 MQIPT 컴퓨터를 통해 통신할 수 있습니다.


Outbound connections
Intranet
Internet
IBM MQ Client
Firewall
MQIPT
SOCKS Proxy
HTTP Proxy
MQIPT
HTTP Proxy
SOCKS Proxy
MQIPT
MQIPT
running on a firewall computer
IBM MQ Client
IBM MQ Server
IBM MQ Server
Key:
HTTP
IBM MQ Protocol



호환 가능 구성

IBM MQ 클라이언트 또는 큐 관리자가 MQIPT와 통신하는 호환 가능 연결 시나리오입니다. 목적지 큐 관리자와 통신하기 위해 동일한 또는 두 번째 MQIPT 라우트가 사용됩니다.

하나의 MQIPT 라우트가 포함된 호환 가능 구성

하나의 MQIPT 라우트를 사용하여 IBM MQ와 통신할 수 있습니다.

268 페이지의 표 26의 열에는 다음과 같은 정보가 포함되어 있습니다.

1. IBM MQ와 MQIPT 라우트 사이에 사용되는 프로토콜. 이 연결은 IBM MQ 클라이언트 또는 큐 관리자에서 작성할 수 있으며 IBM MQ FAP(Formats and Protocols) 또는 SSL/TLS 프로토콜을 사용할 수 있습니다.
2. MQIPT 라우트가 작동하는 모드. MQIPT와 IBM MQ 간의 인터넷을 통한 통신 형식은 MQIPT 라우트의 구성에 따라 판별됩니다. 테이블에서 SSL이 언급되는 위치를 메모해 두십시오. TLS를 사용할 수도 있습니다.
3. MQIPT 라우트와 목적지 큐 관리자 사이에 사용되는 프로토콜.

표 *26.* 하나의 *MQIPT* 인스턴스가 포함된 올바른 구성

1. IBM MQ 소스 프로토콜	**2. MQIPT 라우트의 모드**	**3. IBM MQ 대상 프로토콜**
FAP	FAP 프록시(기본값)	FAP
	FAP 서버 및 SSL 클라이언트	SSL/TLS
SSL/TLS	SSL 프록시	SSL/TLS
	SSL 서버 및 FAP 클라이언트	FAP
	SSL 서버 및 SSL 클라이언트	SSL/TLS

둘 이상의 MQIPT 라우트가 포함된 호환 가능 구성

둘 이상의 MQIPT 인스턴스에서 둘 이상의 라우트를 사용하여 IBM MQ와 통신하도록 선택할 수도 있습니다.

268 페이지의 표 27의 열에는 다음과 같은 정보가 포함되어 있습니다.

1. IBM MQ와 첫 번째 MQIPT 라우트 사이에 사용되는 프로토콜. 이 연결은 IBM MQ 클라이언트 또는 큐 관리자에서 작성할 수 있으며 IBM MQ FAP(Formats and Protocols) 또는 SSL/TLS 프로토콜을 사용할 수 있습니다.
2. 첫 번째 MQIPT 라우트가 작동하는 모드. MQIPT와 IBM MQ 간의 인터넷을 통한 통신 형식은 MQIPT 라우트의 구성에 따라 판별됩니다. 테이블에서 SSL이 언급되는 위치를 메모해 두십시오. TLS를 사용할 수도 있습니다.
3. 두 번째 MQIPT 라우트가 작동하는 모드.
4. 두 번째 MQIPT 라우트와 목적지 큐 관리자 사이에 사용되는 프로토콜.

표 *27.* 복수의 *MQIPT* 인스턴스가 포함된 올바른 구성

1. IBM MQ 소스 프로토콜	**2. 첫 번째 MQIPT 라우트의 모드**	**3. 두 번째 MQIPT 라우트의 모드**	**4. IBM MQ 대상 프로토콜**
FAP(기본값)	FAP 프록시(기본값)	FAP 프록시(기본값)	FAP
	FAP 서버 및 SSL 클라이언트	SSL 프록시	SSL/TLS
		SSL 서버 및 FAP 클라이언트	FAP
		SSL 서버 및 SSL 클라이언트	SSL/TLS
	HTTP 클라이언트	HTTP 서버 및 SSL 클라이언트	SSL/TLS
	HTTPS 클라이언트	HTTPS 서버 및 SSL 클라이언트	SSL/TLS
	HTTP 클라이언트	HTTP 서버	FAP
	HTTPS 클라이언트	HTTPS 서버	FAP

표 *27.* 복수의 *MQIPT* 인스턴스가 포함된 올바른 구성 *(계속)*

1. IBM MQ 소스 프로토콜	**2. 첫 번째 MQIPT 라우트의 모드**	**3. 두 번째 MQIPT 라우트의 모드**	**4. IBM MQ 대상 프로토콜**
SSL/TLS	SSL 프록시	SSL 프록시	SSL/TLS
		SSL 서버 및 FAP 클라이언트	FAP
		SSL 서버 및 SSL 클라이언트	SSL/TLS
	HTTP 클라이언트	HTTP 서버	FAP
	HTTPS 클라이언트	HTTPS 서버	SSL/TLS
	HTTP 클라이언트	HTTP 서버 및 SSL 클라이언트	FAP
	HTTPS 클라이언트	HTTPS 서버 및 SSL 클라이언트	SSL/TLS

지원되는 채널 구성

모든 IBM MQ 채널 유형이 지원되지만 구성은 TCP/IP 연결로 제한됩니다. IBM MQ 클라이언트 또는 큐 관리자에는 MQIPT가 목적지 큐 관리자인 것처럼 표시됩니다. 채널 구성에 목적지 호스트 및 포트 번호가 필요한 경우 MQIPT 호스트 이름 및 목적지 포트 번호가 지정됩니다.

클라이언트/서버 채널
MQIPT는 수신 클라이언트 연결 요청을 청취한 후 HTTP 터널링, SSL/TLS 또는 표준 IBM MQ 프로토콜 패킷을 사용하여 해당 요청을 전달합니다. MQIPT에서 HTTP 터널링 또는 SSL/TLS를 사용하는 경우 연결 시 해당 요청을 두 번째 MQIPT로 전달합니다. HTTP 터널링을 사용하지 않을 경우 연결 시 해당 요청을 목적지 큐 관리자로 표시되는 항목으로 전달합니다(결과적으로 추가적인 MQIPT가 될 수도 있지만). 목적지 큐 관리자에서 클라이언트 연결을 승인하면 클라이언트와 서버 사이에서 패킷이 릴레이됩니다.

클러스터 송신자/수신자 채널
MQIPT가 클러스터 송신자 채널로부터 수신 요청을 수신하는 경우 큐 관리자에서 SOCKS가 사용으로 설정되어 있으며 SOCKS 데이터 교환 프로세스 중에 실제 목적지 주소를 확보하는 것으로 가정합니다. 이 경우 요청을 클라이언트 연결 채널과 정확하게 동일한 방식으로 MQIPT 또는 목적지 큐 관리자로 전달합니다. 여기에는 자동 정의 클러스터 송신자 채널도 포함됩니다.

송신자/수신자
MQIPT가 송신자 채널로부터 수신 요청을 수신하는 경우 요청을 클라이언트 연결 채널과 정확하게 동일한 방식으로 다음 MQIPT 또는 목적지 큐 관리자로 전달합니다. 목적지 큐 관리자는 수신 요청을 유효성 검증한 후 적절한 경우 수신자 채널을 시작합니다. 송신자와 수신자 채널 간의 모든 통신(보안 플로우 포함)이 릴레이됩니다.

요청자/서버
이 결합은 이전 구성과 동일한 방식으로 핸들링됩니다. 연결 요청에 대한 유효성 검증은 목적지 큐 관리자의 서버 채널에서 수행됩니다.

요청자/송신자
두 개의 큐 관리자가 서로 직접 연결을 설정하도록 허용되지 않지만 둘 다 MQIPT에 연결하여 연결을 승인하도록 허용된 경우 "콜백" 구성을 사용할 수 있습니다.

서버/요청자 및 서버/수신자
이는 `Sender/Receiver` 구성을 처리하는 것과 동일한 방식으로 MQIPT 에 의해 처리됩니다.

채널 종료 및 실패 조건

MQIPT에서 IBM MQ 채널의 폐쇄(정상 또는 비정상)를 감지하는 경우 채널 폐쇄를 전파합니다. MQIPT를 사용하여 라우트를 닫을 경우 해당 채널을 통해 이동하는 모든 채널이 닫힙니다.

MQIPT는 선택적 유휴 제한시간 기능을 제공합니다. MQIPT에서 특정 채널이 제한시간을 초과하는 기간 동안 유효 상태임을 감지하는 경우 문제의 두 연결에서 즉시 종료를 수행합니다.

채널 양쪽의 IBM MQ 시스템은 이러한 비정상 종료 조건이 네트워크 장애인지 또는 파트너에 의한 채널 종료인지를 관찰합니다. 그런 다음 채널을 다시 시작하고 MQIPT가 사용되지 않은 것처럼 복구할 수 있습니다(프로토콜 인다우트 기간 중에 장애가 발생한 경우).

메시지의 안전성

IBM MQ 분산 큐 관리를 통해 메시지가 올바르게 전달되도록 할 수 있습니다. 두 채널 사이에 MQIPT가 존재하는 경우에도 해당됩니다. MQIPT는 메시지 데이터를 저장하거나 올바른 메시지 전달을 보증하는 동기점 프로시저에 참가하지 않습니다.

빠르고 비지속적인 IBM MQ 메시지를 사용할 때 MQIPT 라우트가 실패하거나 IBM MQ 메시지가 전송 중일 때 재시작되면 메시지가 유실될 수 있습니다. 라우트를 재시작하기 전에 MQIPT 라우트를 사용하는 모든 IBM MQ 채널이 비활성 상태인지 확인하십시오.

IBM MQ에서 메시지의 안전성에 대한 자세한 정보는 메시지의 안전성을 참조하십시오.

다중 인스턴스 큐 관리자 및 고가용성

MQIPT는 고가용성 환경에서 다중 인스턴스 큐 관리자와 함께 사용할 수 있습니다.

MQIPT에는 지속적 상태가 없기 때문에 MQIPT를 다른 시스템으로 장애 복구하는 경우 이점이 없습니다. 대신 동일한 `mqipt.conf` 구성 파일을 사용하는 다중 MQIPT 인스턴스가 서로 다른 시스템에서 실행되도록 하십시오. 고가용성을 위한 각각의 MQIPT 인스턴스를 모니터하고 필요한 경우 인스턴스를 다시 시작하십시오(동일한 시스템에서). 이 경우 연결을 라우팅하기 위해 사용할 수 있는 동일한 MQIPT 인스턴스가 제공됩니다. 그런 다음 IBM MQ에서 연결을 MQIPT로 라우팅할 수 있으며 MQIPT에서 해당 연결을 목적지 큐 관리자로 전달할 수 있는지 확인해야 합니다.

아웃바운드 IBM MQ 채널은 다양한 방법으로 사용 가능한 MQIPT 인스턴스로 전달할 수 있습니다. 예를 들면 다음과 같습니다.

- WebSphere Edge Components 제품의 IBM Network Dispatcher 와 같은 로드 밸런서 또는 고가용성 라우터를 사용하십시오.
- 쉼표로 구분된 목록을 사용하여 IBM MQ 채널 정의에서 복수의 연결 이름을 지정합니다. 이 경우 IBM MQ는 사용 가능한 MQIPT 인스턴스를 찾을 때까지 차례로 각각의 MQIPT 주소에 연결하려고 시도합니다.

또한 MQIPT에서 목적지 큐 관리자로의 연결도 전달해야 합니다. 고가용성 구성을 통해 목적지 큐 관리자에 대한 IP 주소 장애 복구가 보장된 경우 별도의 MQIPT 구성이 필요하지 않습니다. **Destination** 라우트 특성에 목적지 IP 주소를 지정하고 장애 복구 조작에서 해당 큐 관리자의 IP 주소를 이동할 수 있도록 허용하십시오.

하지만 장애 복구 이후에 큐 관리자의 IP 주소가 변경되는 경우 MQIPT에서 연결을 올바른 목적지로 전달하도록 조정해야 합니다. 이 작업은 다음 방법 중 하나를 사용하여 수행할 수 있습니다.

- 액세스할 수 있는 IP 주소 및 포트 번호를 검사하는 라우팅 엑시트를 작성한 후 각각의 연결에 대한 라우트 목적지를 대체하십시오. MQIPT에서는 몇 가지 샘플 라우팅 엑시트가 제공됩니다. 이러한 엑시트를 이 용도로 채택할 수 있습니다.
- 고가용성 로드 밸런서를 사용하여 연결을 경로 재지정하십시오.
- 큐 관리자를 실행할 수 있는 각각의 IP 주소 및 포트별로 하나씩, 복수의 MQIPT 라우트를 정의하십시오. 그런 다음 IBM MQ 연결을 다양한 MQIPT 라우트로 전달하십시오(예: 아웃바운드 채널의 연결 이름에서 쉼표로 구분된 목록으로 모든 라우트 IP 주소 및 포트 번호 나열).

또한 네트워크 경로에서 모든 엔드-투-엔드 컴포넌트를 성능 조정하는 것이 중요합니다.

1. 사용 불가능한 시스템에 대한 연결을 시도하는 경우 처음으로 사용 가능한 대상으로 재연결 시도가 이동할 수 있도록 즉시 실패해야 합니다.

   MQIPT SSL 라우트의 경우 사용 불가능한 대상에 대한 연결이 즉시 실패하도록 **SSLClientConnectTimeout** 라우트 특성을 성능 조정하십시오. IBM MQ 성능 조정 매개변수에 대한 자세한 정보는 IBM MQ 문서를 참조하십시오. 또한 운영 체제의 TCP/IP 성능 조정에 대한 자세한 정보는 운영 체제 문서를 참조하십시오. 모든 경우에 실패한 연결 시도는 신속하게 네트워크 실패(예: TCP 재설정 패킷)를 리턴하거나 과도한 지연 없이 제한시간이 초과되어야 합니다.

2. 실패한 시스템에 대한 활성 연결은 새 연결을 설정할 수 있도록 즉시 연결을 끊어야 합니다.

   또한 연결에서 MQIPT를 사용 중인 경우 장애 복구의 영향도 고려해야 합니다. 장애 복구 중에는 네트워크 연결이 끊어질 수 있습니다. 클라이언트 애플리케이션의 경우 IBM MQ 자동 클라이언트 재연결 기능을 사용하여 끊어진 연결을 재설정할 수 있습니다. 메시지 채널의 경우 채널이 즉시 재연결되도록 짧은 재시도 간격을 지정할 수 있습니다. 자동 클라이언트 재연결 및 메시지 채널 재시도 구성에 대한 자세한 정보는 IBM MQ 문서를 참조하십시오.

V 9.3.5 IBM MQ Console 및 REST API

IBM MQ Console 및 REST API 를 사용하여 IBM MQ를 관리하고 HTTP를 사용하여 메시징 조작을 수행할 수 있습니다.

- IBM MQ Console 를 사용하여 웹 브라우저에서 기본 관리 태스크를 수행할 수 있습니다. 자세한 정보는 IBM MQ Console를 사용한 관리를 참조하십시오.
- administrative REST API 를 사용하여 큐 관리자 및 큐, Managed File Transfer 에이전트 및 전송과 같은 IBM MQ 오브젝트를 관리할 수 있습니다. 자세한 정보는 REST API를 사용하여 관리를 참조하십시오.
- messaging REST API 를 사용하여 단순 지점간 및 공개 메시징을 수행할 수 있습니다. 자세한 정보는 REST API를 사용한 메시징을 참조하십시오.

설치 옵션

IBM MQ Console 및 REST API 는 mqweb라고 하는 WebSphere Liberty 서버에서 실행됩니다. IBM MQ 9.3.5부터는 IBM MQ 설치에서 선택적 구성요소로 또는 독립형 IBM MQ Web Server 설치로 mqweb 서버를 설치할 수 있습니다.

V 9.3.5 Linux 독립형 IBM MQ Web Server 설치

IBM MQ 9.3.5부터 mqweb 서버는 IBM MQ Web Server의 독립형 설치에서 실행될 수 있습니다. 독립형 IBM MQ Web Server 설치를 사용하면 IBM MQ 설치와 별도의 시스템에 mqweb 서버를 설치하고 실행할 수 있습니다. 독립형 IBM MQ Web Server 를 설치하면 mqweb 서버를 실행하도록 선택하는 시스템 및 시스템 수에 대해 더 큰 유연성을 제공합니다. 필요한 경우, mqweb 서버의 여러 인스턴스를 다른 시스템에서 실행하여 필요한 확장성 및 가용성을 제공할 수 있습니다.

IBM MQ 인타이틀먼트를 구매한 경우 독립형 IBM MQ Web Server의 필요한 수만큼의 사본을 설치할 수 있습니다. IBM MQ Web Server 설치는 구매한 IBM MQ 인타이틀먼트로 계수되지 않습니다. IBM MQ 라이센싱에 대한 자세한 정보는 IBM MQ 라이센스 정보를 참조하십시오.

다음 제한사항은 독립형 IBM MQ Web Server 설치에 적용됩니다.

- IBM MQ Console 는 리모트 큐 관리자만 관리하는 데 사용할 수 있습니다.
- messaging REST API 는 리모트 큐 관리자에서만 사용할 수 있습니다.
- administrative REST API 를 사용할 수 없습니다.

독립형 IBM MQ Web Server 는 Linux 플랫폼에서만 지원됩니다.

독립형 IBM MQ Web Server설치에 대한 자세한 정보는 독립형 IBM MQ Web Server설치를 참조하십시오.

IBM MQ 설치의 선택적 구성요소

IBM MQ Console 및 REST API 구성요소를 IBM MQ 설치의 일부로 설치하도록 선택할 수 있습니다.

모든 IBM MQ Console 및 REST API 기능은 mqweb 서버가 IBM MQ 설치에서 실행될 때 사용 가능합니다.

- IBM MQ Console 를 사용하여 로컬 및 리모트 큐 관리자를 관리할 수 있습니다.
- messaging REST API 는 로컬 및 리모트 큐 관리자와 함께 사용할 수 있습니다.
- administrative REST API 를 사용하여 로컬 및 리모트 큐 관리자를 관리할 수 있습니다.

IBM MQ Console 및 REST API 구성요소를 사용하려면 IBM MQ 설치의 일부로 다음 구성요소를 설치하십시오.

- AIX AIX에서 `mqm.web.rte` 파일 세트를 설치하십시오.

- IBM i IBM i에 WEB 컴포넌트를 설치하십시오.
- Linux Linux에서 MQSeriesWeb 구성요소를 설치하십시오.
- Windows Windows에서 Web Administration 기능을 설치하십시오.
- z/OS z/OS에서 IBM MQ for z/OS UNIX System Services Web Components 기능을 설치하십시오.

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. 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.

07326
서울특별시 영등포구
국제금융로 10, 3IFC
한국 아이.비.엠 주식회사
U.S.A.

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

Intellectual Property Licensing
2-31 Roppongi 3-chome, Minato-Ku
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 명시적 또는 묵시적인 일체의 보증 없이 이 책을 "현상태대로" 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

서울특별시 영등포구
서울특별시 강남구 도곡동 467-12,
군인공제회관빌딩
한국 아이.비.엠 주식회사
U.S.A.

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등)하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 단계의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을

통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 애플리케이션 프로그래밍 인터페이스(API)에 부합하는 애플리케이션을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이들 샘플 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다.

이 정보를 소프트카피로 확인하는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

프로그래밍 인터페이스 정보

프로그래밍 인터페이스 정보는 본 프로그램과 함께 사용하기 위한 응용프로그램 소프트웨어 작성을 돕기 위해 제공됩니다.

이 책에는 고객이 프로그램을 작성하여 WebSphere MQ서비스를 얻을 수 있도록 하는 계획된 프로그래밍 인터페이스에 대한 정보가 포함되어 있습니다.

그러나 본 정보에는 진단, 수정 및 성능 조정 정보도 포함되어 있습니다. 진단, 수정 및 성능 조정 정보는 응용프로그램 소프트웨어의 디버그를 돕기 위해 제공된 것입니다.

중요사항: 이 진단, 수정 및 튜닝 정보는 변경될 수 있으므로 프로그래밍 인터페이스로 사용하지 마십시오.

상표

IBM, IBM 로고, ibm.com®는 전세계 여러 국가에 등록된 IBM Corporation의 상표입니다. 현재 IBM 상표 목록은 웹 "저작권 및 상표 정보"(www.ibm.com/legal/copytrade.shtml) 에 있습니다. 기타 제품 및 서비스 이름은 IBM 또는 타사의 상표입니다.

Microsoft 및 Windows는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.

Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.

이 제품에는 Eclipse 프로젝트 (https://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: