IBM MQ V9.3 機能と構成

3. 主な機能

2023年09月

日本アイ・ビー・エム　システムズ・エンジニアリング（株）

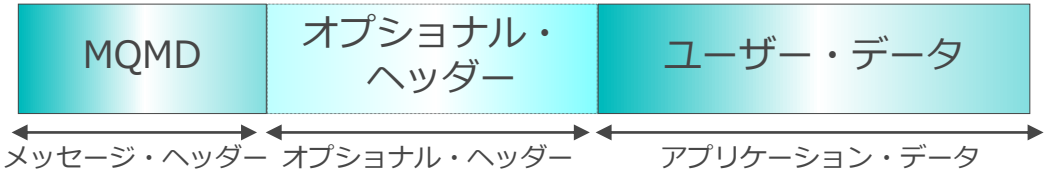
トピック

- メッセージ処理
  - メッセージ
  - メッセージ・タイプ
  - レポート・オプション
  - メッセージの永続性
  - メッセージの存続時間
  - ブラウズ機能
  - メッセージの優先処理
  - メッセージIDと相関ID
  - メッセージ・コンテキスト
  - メッセージのグループ化
  - メッセージのセグメント化
  - 参照メッセージ
  - 文字コード変換
  - 動的キュー
  - 排他使用
  - 配布リスト
- トランザクション・サポート
  - 同期点処理
  - データベース・コーディネーション
- メッセージ・ドリブン処理
  - メッセージの待機受信
  - コールバック機能
  - トリガー
- セキュリティ
  - 許可サービス
  - 代替ユーザーID
  - SSL/TLSチャネル
  - チャネル・アクセス制御
  - 接続認証
- その他の機能
  - Publish/Subscribe
  - EXIT処理
  - Messaging REST API
  - MQ InternetPass-Thru(MQIPT)
  - Trusted Application (FASTPATH BINDING)
  - ストリーミング・キュー

メッセージ処理

メッセージ

- キューを介してプログラム間で送受信されるデータ
- メッセージの構成
  - ◆ メッセージ・ヘッダー（MQMD）
  - ◆ オプショナル・ヘッダー(RFH2やIIHなど)
  - ◆ ユーザー・データ
    - キュー･マネージャーは最大100MBまでのユーザー・データを処理可能



- MQMD
  - ◆ メッセージの識別情報やメッセージの制御情報などを含むメッセージ・ヘッダー
  - ◆ MQMDの主なフィールド

フィールド	説明
Report	レポート・メッセージを受け取るための指定
MsgType	扱うメッセージのタイプ
Expiry	メッセージの有効期限
CodedCharSetID	ユーザー・データのCCSID
Format	ユーザー・データのフォーマット
Priority	メッセージの優先順位
Persistence	メッセージの永続性
MsgID / CorrelID	メッセージの識別ID
ReplyToQ / ReplyToQMgr	サーバー応答先やレポート・メッセージの送信先

メッセージ・タイプ

- MQによって定義されるメッセージのタイプは以下の4つ
  - デフォルトのメッセージ･タイプは“データグラム”

メッセージ・タイプ	MsgTypeフィールド	説明
リクエスト・メッセージ	MQMT_REQUEST	メッセージを受信したアプリケーションからの応答を要求するメッセージ
リプライ・メッセージ	MQMT_REPLY	他のメッセージに応答するメッセージ
データグラム	MQMT_DATAGRAM	キューからメッセージを読み取ったアプリケーションからの応答が不要な一方向の通知メッセージ
レポート・メッセージ	MQMT_REPORT	メッセージ処理中のエラー発生などのイベントをアプリケーションに通知するメッセージ

  - MQPUT時、MQMDの“MsgType”フィールドで指定
  - メッセージをGETするアプリケーションがメッセージ・タイプを判断して適切なアクションをとる
    - キュー・マネージャーのチェックはない
  - リクエストまたはレポート･メッセージの戻り先はMQMDの“ReplyToQ”、および“ReplyToQMgr”フィールドにて指定

レポート・オプション

- MQPUT時に宛先の例外状況に応じたレポート・メッセージを受け取る指定が可能
  - ◆ レポート・メッセージの種類(レポート・タイプ)はMQMDの“Report”フィールドで指定
  - ◆ 以下のレポート・タイプをサポート

レポート・タイプ名	種類	説明
Exception	例外	宛て先キュー・マネージャー上のローカル・キューに書き込めなかった時にレポート・メッセージを生成
Expiry	メッセージの有効期限	指定した時間以上キューに滞留したメッセージが削除された時にレポート・メッセージを生成
Confirm-Of-Arrival（COA）	到着の確認	メッセージが最終宛て先のキューに書き込まれた時にレポート・メッセージを生成
Confirm-Of-Delivery（COD）	配布の確認	メッセージが最終宛て先のキューから読み込まれた時にレポート・メッセージを生成
Positive-Action-Notification(PAN) Negative-Action-Notification(NAN)	アクションの正常終了通知 アクションの異常終了通知	メッセージを受信したアプリケーションの処理結果をレポート・メッセージとして生成（レポート・メッセージはアプリケーションが生成）

  - ◆ 指定方法
    - MQPUT時、 MQMDの“MsgType”フィールドにMQMT_REPORTを指定
      - レポート・メッセージ（MQMT_REPORT)の“Feedback”フィールドに理由（フィードバック・コード）がセットされる
    - 戻り先のキューは、MQMDの“ReplyToQ”、および“ReplyToQMgr”フィールドに指定

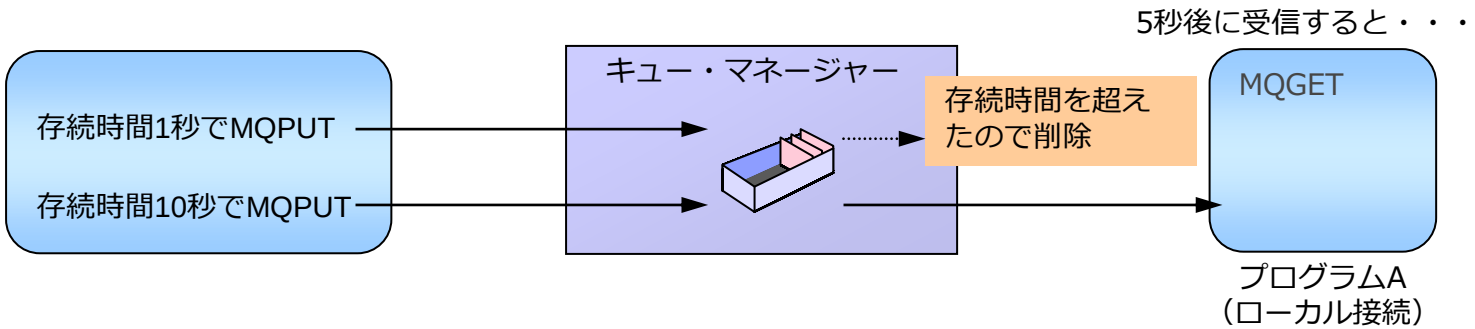
メッセージの永続性

- メッセージの永続性(パーシステント)を選択することでキュー・マネージャー障害時のメッセージ保証を選択
  - ◆ パーシステント・メッセージ
    - キュー・マネージャーの再起動をまたがって、メッセージを保持
    - ログ（ディスク）への書き込みあり
  - ◆ ノン・パーシステント・メッセージ
    - キュー・マネージャーのリスタート時、メッセージは消失
    - ログ（ディスク）への書き込みなし
    - ログへの書き込みがないため、パーシステント・メッセージに比べパフォーマンスが良い

- MQPUT時、MQMDの“Persistence”フィールドで指定
  - ◆ パーシステント・メッセージ
    - MQPER_PERSISTENT
  - ◆ ノン・パーシステント・メッセージ
    - MQPER_NOT_PERSISTENT
  - ◆ メッセージの永続性をキュー属性(DEFPSIST)からセットする場合
    - MQPER_PERSISTENCE_AS_Q_DEF

メッセージの存続時間

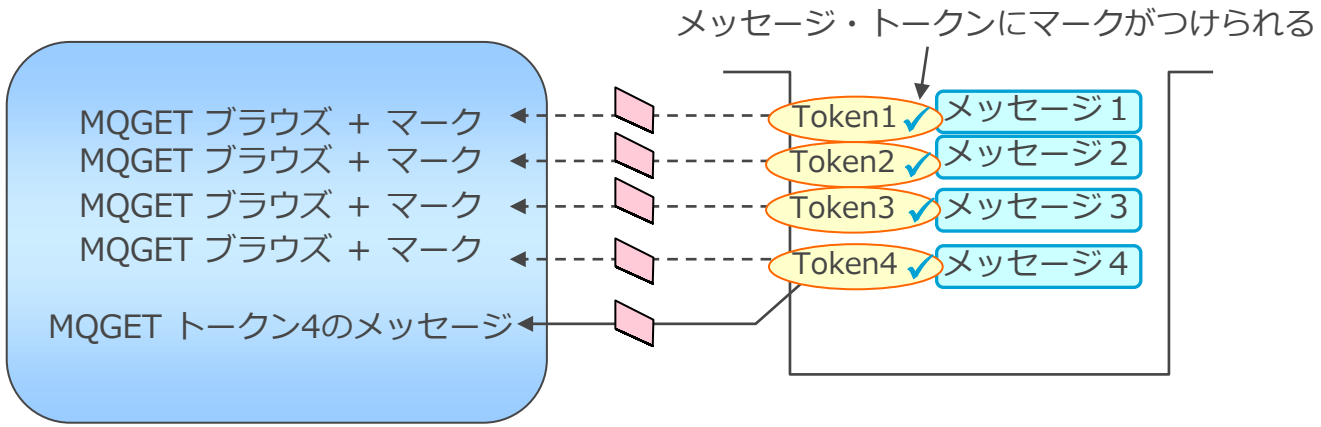
- 指定時間を超えて滞留しているメッセージを自動削除可能
  - ◆ 設定された有効期限のカウント・ダウンは、対象メッセージがキューに書き込まれた時点から開始
    - ● 存続時間のカウントはMQPUTされてから始まるが、ネットワークの転送時間はカウント対象外
  - ◆ 指定された時間を経過しても何の処理も行われなかった場合、一定のタイミングでMQが自動削除
- 指定方法
  - ◆ MQMDの“Expiry”フィールドに存続時間を設定してMQPUT
  - ◆ キューのCAPEXPRY属性でも存続時間を指定可能
    - ● 両方が指定されている場合は、短い方の存続時間が採用される
  - ◆ 存続時間は1/10秒単位で設定



  - ◆ メッセージ同期点処理している場合
    - ● メッセージをPUTした時から存続時間の減算が始まる（コミットのタイミングからではない）
    - ● メッセージをGETしても存続時間の減算は続けられる
      - - バックアウトした際に、再びメッセージをMQGETできるとは限らない

ブラウズ機能

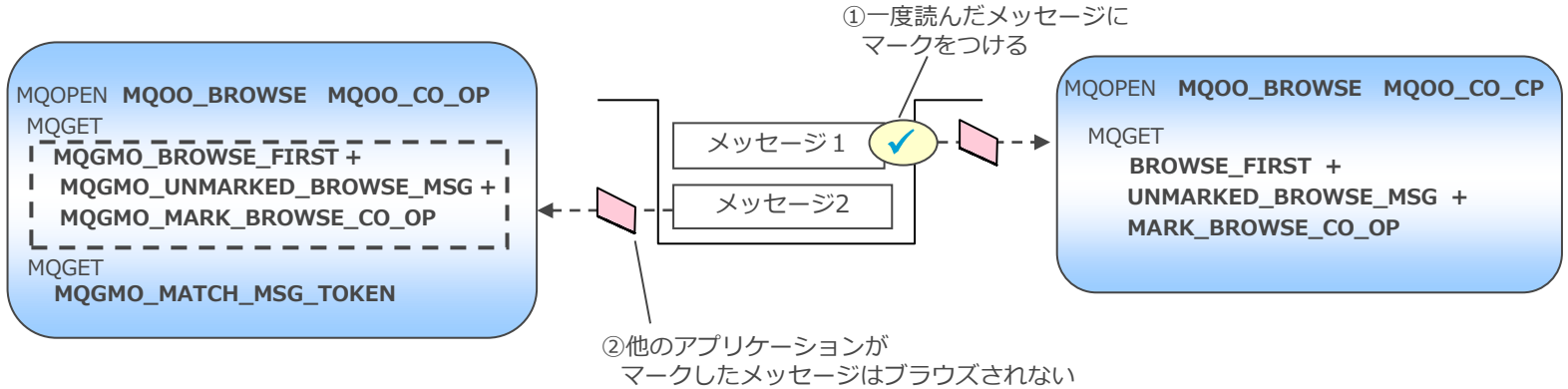
- キュー内にあるメッセージの中身をブラウズ可能
  - ◆ メッセージ・トークンにマークを付けながらメッセージをブラウズすることが可能
    - メッセージ・トークンはキュー内でユニークなIDとしてメッセージに付与される
    - メッセージの読み飛ばしを防ぐことができる
    - ブラウズ時にコミットされていなかったメッセージやバックアウトされたメッセージを容易に読むことができる
    - メッセージ・トークンを指定して該当するメッセージをGETすることも可能
    - １つのキューに対して、複数アプリケーションでマークを共有することが可能
      - 2重読みが発生しない
  - ◆ V6まではカーソルによるブラウズのみサポート
    - カーソルを１つずつ進めながらメッセージをブラウズ
      - メッセージの読み飛ばしが発生することがある
      - 定期的にカーソルを先頭に戻す必要がある



ブラウズ機能

- ブラウズの指定方法
  - ◆ マークのついたメッセージはブラウズしない
    - MQGMO_UNMARKED_BROWSE_MSG
  - ◆ メッセージにマークを付ける（単一アプリケーションでの使用の場合）
    - MQGMO_MARK_BROWSE_HANDLE
  - ◆ メッセージにマークを付ける（マークを複数アプリケーションで共有する場合）
    - MQOO_CO_OP＋MQGMO_MARK_BROWSE_CO_OP
    - 2重読みが発生しない
  - ◆ メッセージ・トークンを指定して該当するメッセージをGETすることも可能
    - MQMO_MATCH_MSG_TOKEN

例)．キューの先頭からマークされていないメッセージの読み込み（複数アプリケーションでのマークの共有）

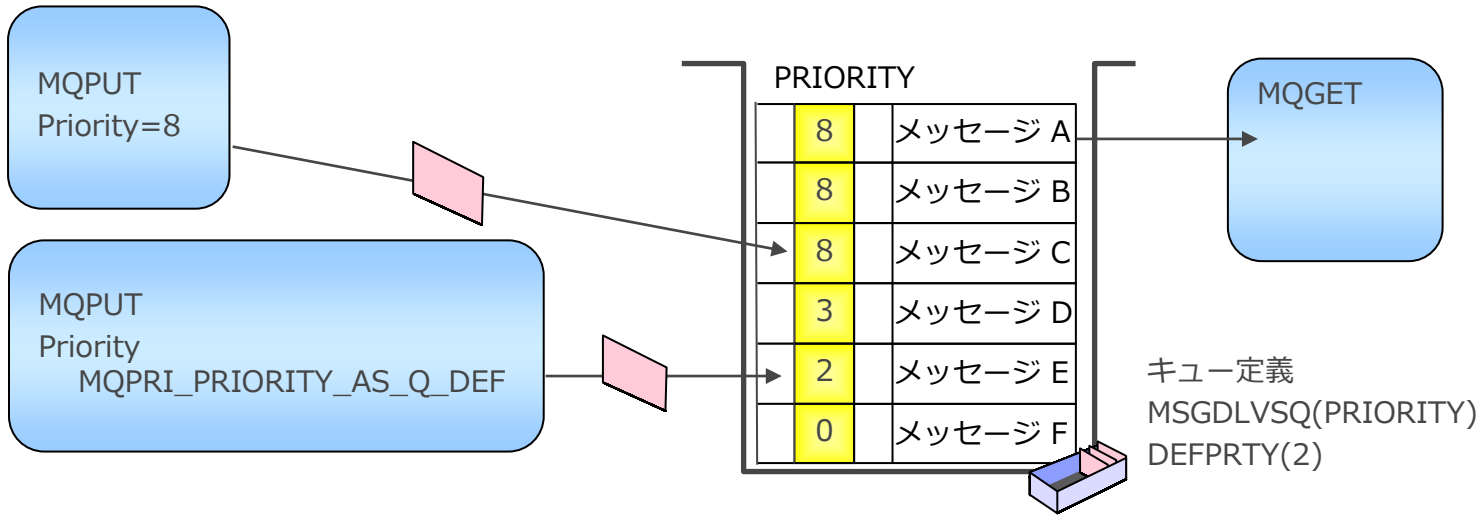


<補足>ブラウズ機能

- マーク付きブラウジングにてマークが外されるタイミング
  - ◆ マークの共有、非共有にかかわらず、下記のタイミングでマークは外れる
    - MQGETによってキューからメッセージが取り出されるとき
    - MQGETがロールバックされたとき
    - Expiry満了によりメッセージが削除されたとき
    - キュー・マネージャー属性MARKINTの設定時間を過ぎたとき
      - ALTER QMGR MARKINT(integer | NOLIMIT)
        - NOLIMIT： MAKRINT属性によるマークの除去はなし
        - integer(ミリ秒)：integer秒経過後、マークが外される
  - ◆ 下記の場合、マークの共有、非共有の違いにより、異なるタイミングでマークが外れる
    - マークを共有しない場合
      - オブジェクト・ハンドルが閉じられたとき
      - MQGMO_UNMARK_BROWSE_HANDLEオプション付でブラウズしたとき
    - マークを共有する場合
      - MQOO_CO_OPでオープンしているすべてのオブジェクト・ハンドルが閉じられたとき
      - MQGMO_UNMARK_BROWSE_CO_OPオプション付でブラウズしたとき

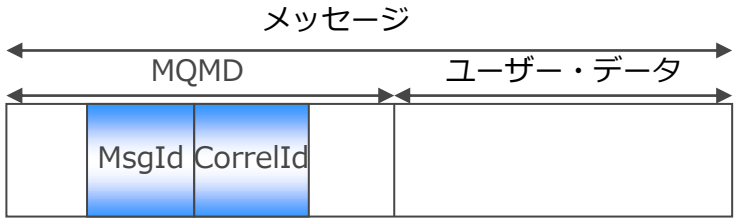
メッセージの優先処理

- メッセージにプライオリティを設定し、プライオリティ順にメッセージを受信可能
  - ◆ プライオリティの付けられたメッセージはMQPUT時に、キュー内でプライオリティ順に従って並び替えられる
    - キューのMSGDLVSQ属性を“PRIORITY”に設定（デフォルト）
    - 同一プライオリティのメッセージはFIFO
    - メッセージ毎に設定可能
  - ◆ MQPUTの際、メッセージの優先順位をMQMDのPriorityフィールドに設定
    - 優先順位：0（最低）～9（最高）
    - MQPRI_PRIORITY_AS_Q_DEFを指定するとキューの属性のデフォルト値(DEFPRTY)がセットされる

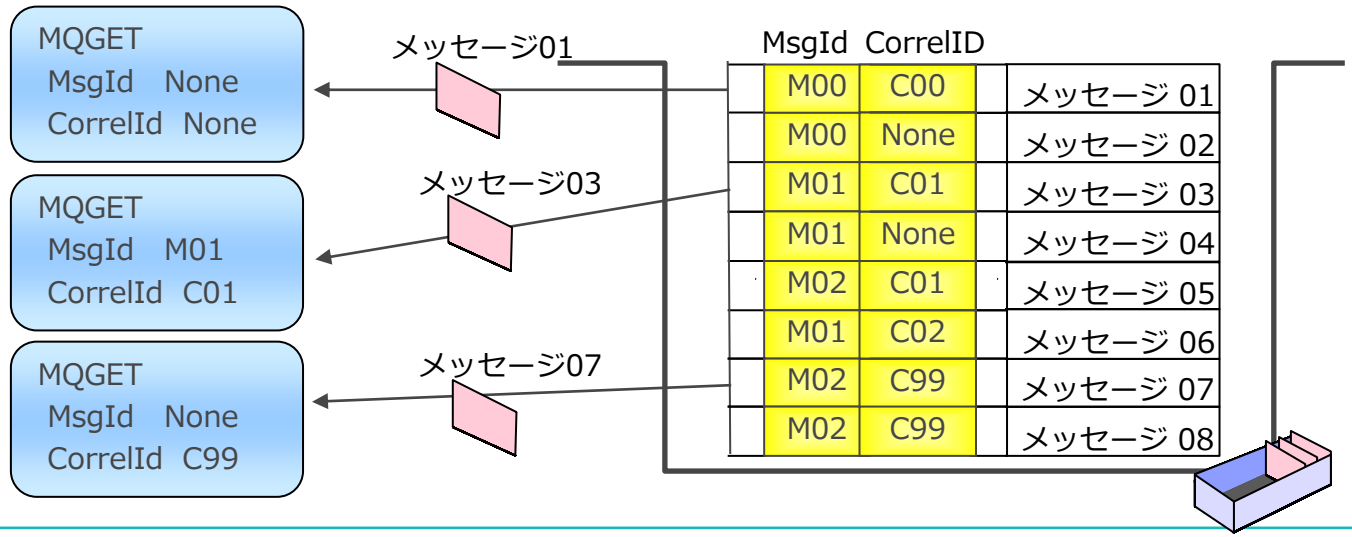


メッセージIDと相関ID

- 同一キュー上の個々のメッセージの識別のために2種類のキーの指定が可能
  - ◆ メッセージID
    - MQMDの“MsgId”フィールド
  - ◆ 相関ID
    - MQMDの“CorrelId”フィールド



- MQGET時にメッセージIDと相関IDを指定することが可能
  - ◆ メッセージID、相関IDはMQBYTEであることに注意
    - CCSIDの異なるプラットフォーム間でのハンドリング（コード変換の対象にはならない）



メッセージIDと相関ID

メッセージIDと相関IDの利用

◆ メッセージIDはキュー・マネージャーが一意のIDを割り振る

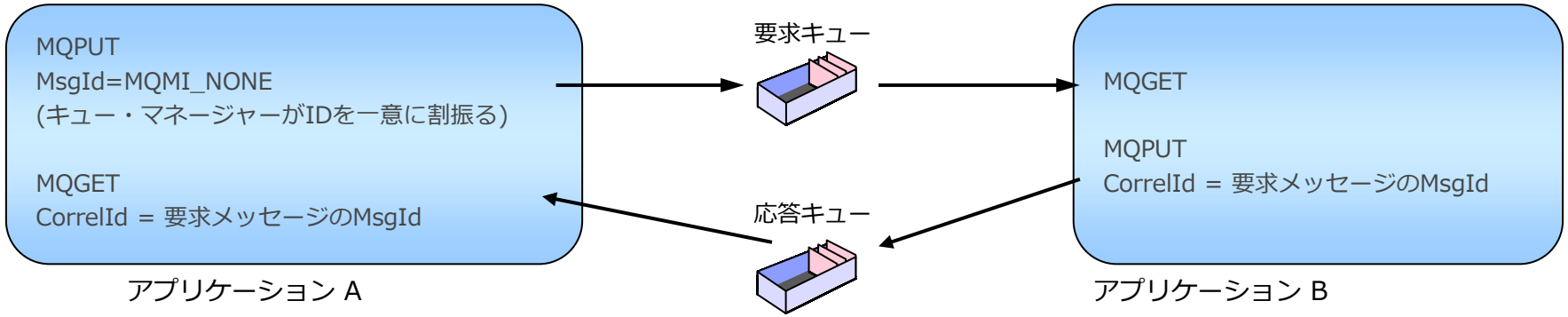
- メッセージID 、相関ID共にアプリケーションが指定することも可能

◆ 要求メッセージと応答メッセージの関連付け

- クライアント・アプリケーション側での処理
  - 要求メッセージを送信した時に、キュー・マネージャーがMsgIdを一意に割振る
  - MQPUT時に割振られたMsgIdを使い、CorrelId指定で応答メッセージを選択してMQGET
- サーバー・アプリケーション側での処理
  - 要求メッセージのMsgIdを応答メッセージのCorrelIdにセットして返信

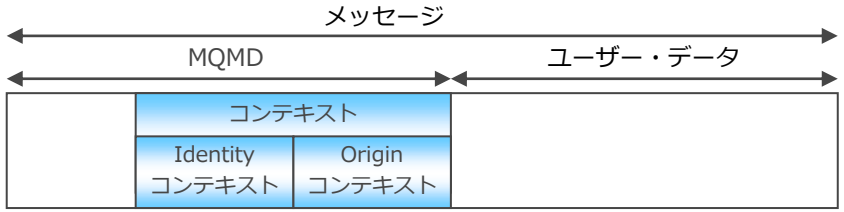
◆ メッセージIDと相関IDの指定

- “MatchOptions”オプションを設定
  - MsgIdに合致する最初のメッセージを検索：MQMO_MATCH_MSG_ID
  - CorrelIdに合致する最初のメッセージを検索：MQMO_MATCH_CORREL_ID



メッセージ・コンテキスト

- MQMDの特定のフィールドにはメッセージ・コンテキスト（発信元の情報）が含まれている
  - ◆ Identityコンテキスト
    - メッセージをキューに入れたアプリケーションのユーザーに関する情報
  - ◆ Originコンテキスト
    - メッセージを入れたアプリケーションに関する情報、キューに入れた時間など

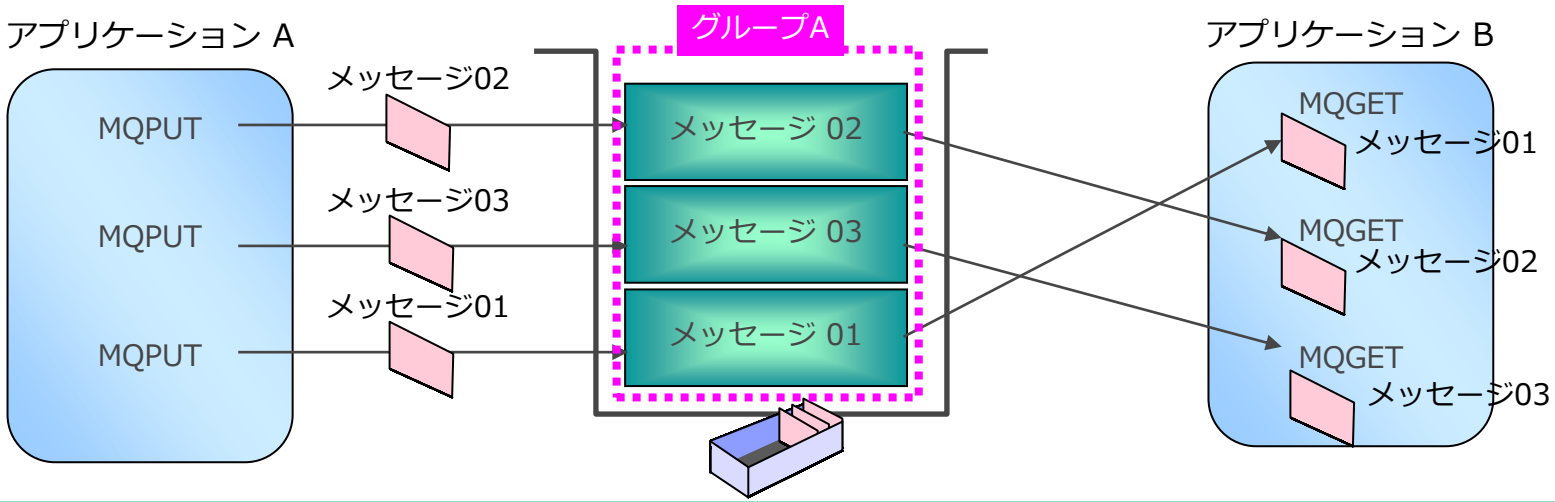


  - ◆ 受信アプリケーションはそのメッセージの発信元を知ることができ、セキュリティのチェックやメッセージのトラッキング（監査ログ）などに利用可能

コンテキスト・タイプ	フィールド名	説明
Identityコンテキスト	UserIdentifierフィールド	ユーザID
	AccountingTokenフィールド	アカウンティング・トークン
	ApplIdentityDataフィールド	アプリケーション任意のデータ
Originコンテキスト	PutApplTypeフィールド	メッセージをPUTしたアプリケーションのタイプ
	PutApplNameフィールド	メッセージをPUTしたアプリケーション名
	PutDateフィールド	メッセージをPUTした日時　(≠COMMITした日時)
	PutTimeフィールド	メッセージをPUTした時刻　(≠COMMITした時刻)
	ApplOriginDataフィールド	アプリケーション任意のデータ

メッセージのグループ化

- 関連する複数のメッセージを論理的に1つのメッセージとして扱うことが可能
  - メッセージをキュー内の物理的な順序ではなく、論理的な順序に組み立てて受信も可能
  - キュー・マネージャーとアプリケーションのどちらでもグループ化が可能
  - MQMD（V2ヘッダー）の3つのフィールドを利用
    - GroupId
      - グループを識別するために付与された一意のID
    - MsgSeqNumber
      - グループ内でのメッセージの順序
    - MsgFlags
      - グループの最終メッセージを識別するためのフラグ



メッセージのグループ化

キュー・マネージャーによるメッセージのグループ化

◆ グループ・メッセージの送信

- MQPMO.OptionsにMQPMO_LOGICAL_ORDERを設定
  - GroupIdとMsgSeqNumberをMQが自動的に割り振る
- MQMD.MsgFlagsにMQMF_MSG_IN_GROUP、MQMF_LAST_MSG_IN_GROUPを設定
  - 最終メッセージではMQMF_LAST_MSG_IN_GROUPを設定

◆ グループ・メッセージの受信

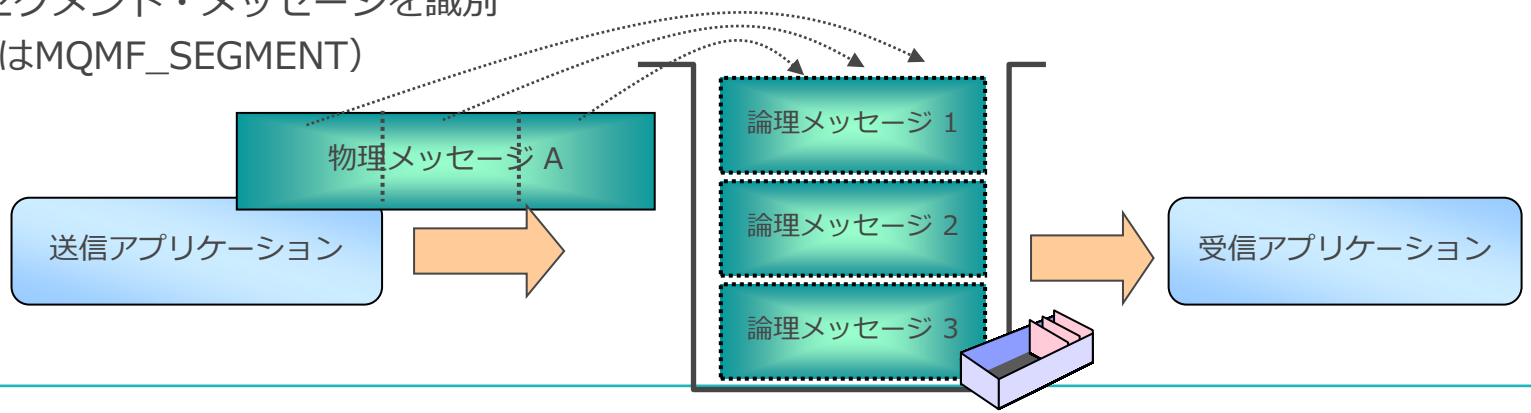
- キュー・マネージャーにグループ内のすべてのメッセージの到着を確認させ、グループ内の論理順序に組み立てさせながら受信
- MQGMO_Optionsに下記を設定
  - MQGMO_ALL_MSGS_AVAILABLE
    - グループのすべてのメッセージの到着を確認（1件目のメッセージの受信にのみ使用可能）
  - MQGMO.OptionsにMQGMO_LOGICAL_ORDER
    - メッセージをグループ内の論理順序で受信

メッセージのグループ化

- アプリケーションによるメッセージのグループ化
  - ◆ グループ・メッセージの送信
    - MQPUTされた順序（物理的な順序）と、グループ内での論理順序が異なる時は、アプリケーションでグループ化する必要がある
    - GroupId、MsgSeqNumber、MsgFlagsをアプリケーションで設定
    - GroupIdは初回のMQPUTで割り振られたものを引き継ぐ
  - ◆ グループ・メッセージの受信
    - アプリケーションで設定したMQMDのGroupID、MsgSeqNumber、Offsetを指定してMQGET
      - MQGMOのMatchOptionで指定

メッセージのセグメント化

- 1つのメッセージを物理的に分割（セグメント）することが可能
  - ◆ MQPUT、MQGETでセグメント化された論理メッセージを扱う
  - ◆ キュー・マネージャーとアプリケーションのどちらでもセグメント化が可能
  - ◆ メッセージをセグメント化するケース
    - ● システム・リソースを考慮してメッセージを小さいサイズに分割して転送する場合
    - ● メッセージ・サイズが大き過ぎて1度にキューに書けない場合
  - ◆ MQMD（V2ヘッダー）の3つのフィールドを利用
    - ● GroupId、MsgSeqNumber
      - 元のメッセージが1つであったことの識別子
    - ● Offset
      - 元のメッセージ内でのデータ・オフセット
    - ● MsgFlags
      - MQMF_LAST_SEGMENTで最終セグメント・メッセージを識別（最終セグメント・メッセージ以外はMQMF_SEGMENT）



メッセージのセグメント化

キュー・マネージャーによるメッセージのセグメント化

◆ セグメント・メッセージの送信

- セグメント化をキュー・マネージャーに任せるために、MQMD.MsgFlagsに“MQMF_SEGMENTATION_ALLOWED”を指定
  - キュー・マネージャーはキューのMAXMSGLENGTH属性に合わせてメッセージを自動で分割

◆ セグメント・メッセージの受信

- セグメント・メッセージをキュー・マネージャーに組み立てさせるために、MQGMO_Optionsに“MQGMO_COMPLETE_MSG”を指定
  - アプリケーションはキュー・マネージャーによって組み立てられたメッセージを受信
  - アプリケーションは組み立てたメッセージを格納できる十分なバッファの用意が必要
  - すべてのセグメント・メッセージが届かない場合は “MQRC_NO_MESSAGE_AVAILABLE” が返される

メッセージのセグメント化

アプリケーションによるメッセージのセグメント化

◆ セグメント・メッセージの送信

- 元のメッセージが1つであったことを識別させるために、GroupId、MsgSeqNumber、Offsetへ値を設定
  - MQPMO.Optionsに“MQPMO_LOGICAL_ORDER”を設定すると
    MQが自動的にGroupId、MsgSeqNumber、Offsetを割り振る（値をセットしても無視される）
  - 設定しない場合はアプリケーションにて適切な値を設定しながらMQPUTする必要がある

- メッセージがセグメント化されていることを知らせるために、MQMD.MsgFlagsにMQMF_SEGMENTを指定して
  MQPUTを繰り返す
  - 最終セグメント・メッセージではMQMF_LAST_SEGMENTを指定

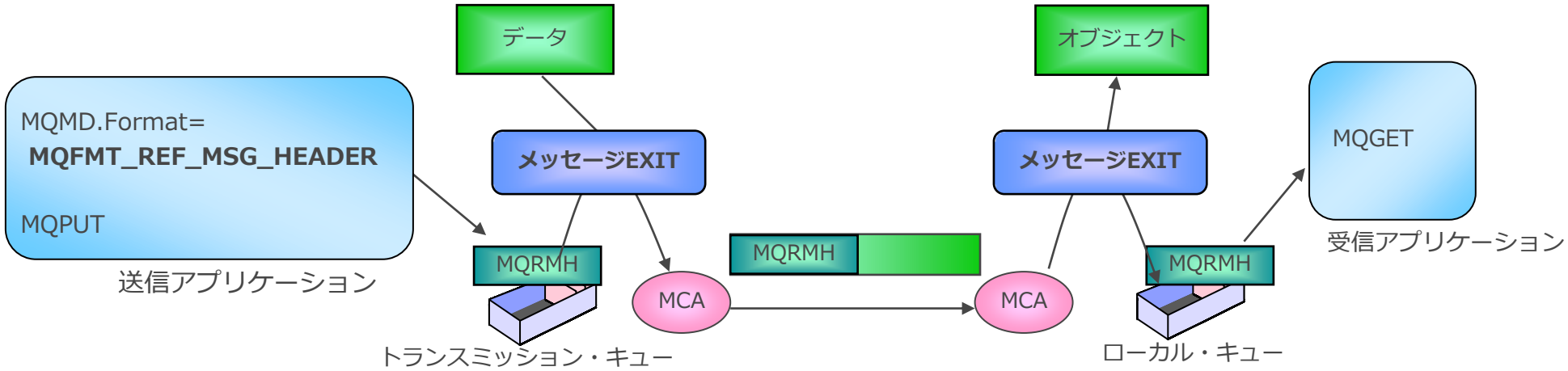
◆ セグメント・メッセージの受信

- セグメント化されたすべてのメッセージの到着を確認するために、
  MQGMO_Optionsに“MQGMO_ALL_SEGMENTS_AVAILABLE”を指定

- セグメント・メッセージを順序どおりに組み立てるために、
  MQGMO.Optionsに“MQGMO_LOGICAL_ORDER”を指定
  - MQGMOのSegmentStatusが“MQSS_LAST_SEGMENT”になるまでMQGETを繰り返す

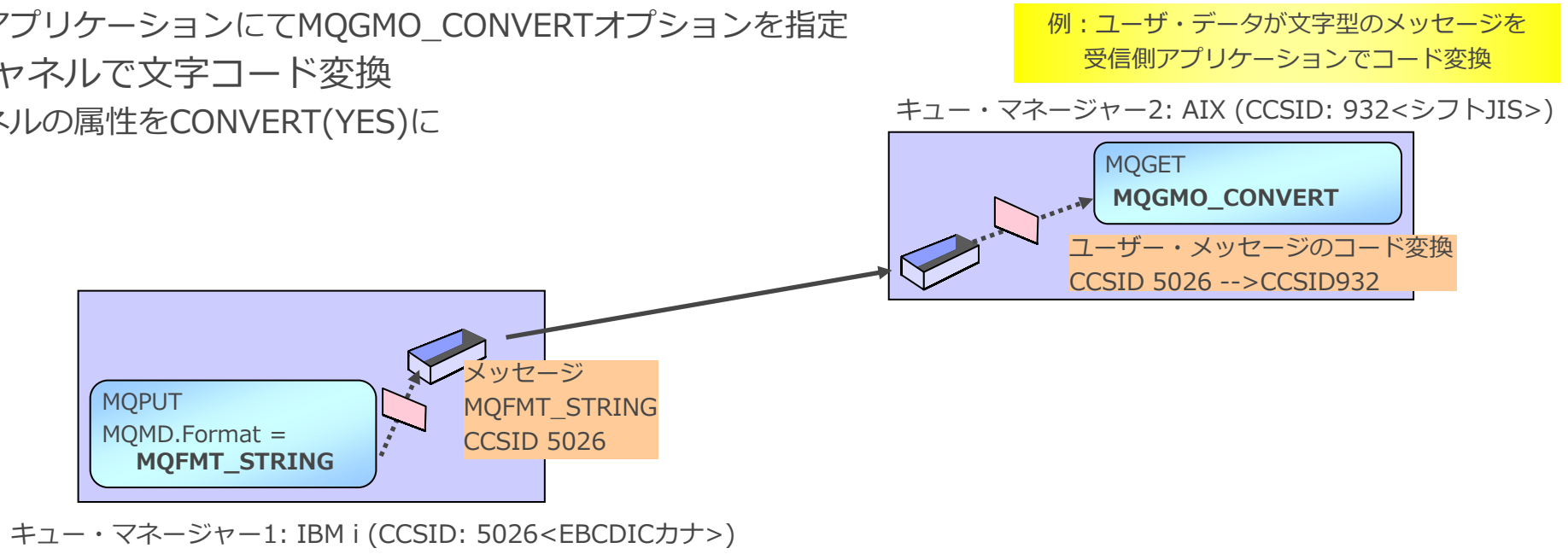
参照メッセージ

- キューを介さずにメッセージを送受信することが可能
  - ラージ・メッセージの送受信が可能
  - チャネルのメッセージEXITにより実現
    - サンプル・プログラム（MQPUT、MQGET、メッセージEXIT用）の提供
    - 送信・受信チャネルの両側でメッセージEXITを指定
  - 処理の流れ
    - MQMDの“Format”フィールドに“MQFMT_REF_MSG_HEADER”を指定し、送信アプリケーションから参照メッセージ・ヘッダー（MQRMH）をMQPUT
    - 送信チャネルのメッセージEXITが、MQRMHの情報から送信データをMQRMHと一緒に送信
    - 受信チャネルのメッセージEXITが、受信オブジェクトを作成し、MQRMHのみをキューに書く
    - 受信アプリケーションは、MQRMHを読み、作成されたオブジェクトにアクセスをする



文字コード変換

- メッセージ（ユーザ・データ）の文字コード変換を行うことが可能
  - ◆ CCSIDとエンコーディングに基づいてコード変換を行う
    - 文字フィールドはCCSIDを使用
    - 数値フィールドはエンコーディングを使用
      - エンコーディングはキュー・マネージャーの稼動環境(OS、アプリケーションの言語）に依存
  - ◆ 文字コード変換が行われる場所
    - 受信側でMQGET時にユーザー・メッセージの文字コード変換
      - 受信アプリケーションにてMQGMO_CONVERTオプションを指定
    - 送信チャネルで文字コード変換
      - チャネルの属性をCONVERT(YES)に



文字コード変換

■ メッセージ（ユーザー・データ）の文字コード変換方法

◆ MQが提供する文字コード変換機能

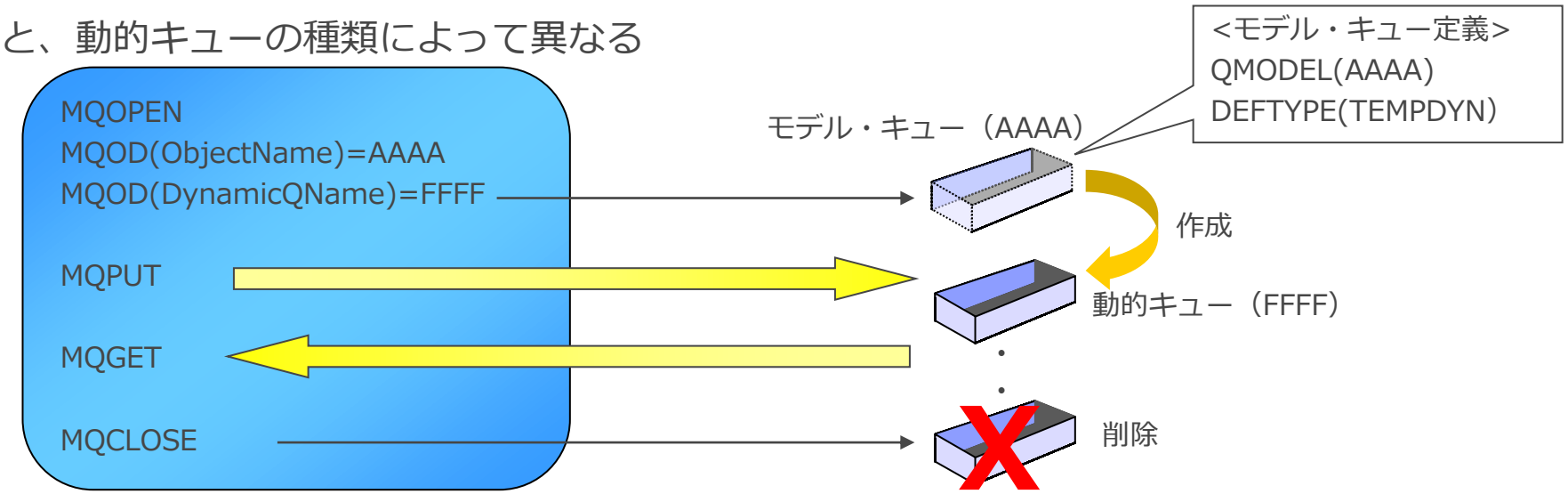
- メッセージが全て文字型の場合
- MQPUT時、MQMDの“Format”フィールドを“MQFMT_STRING”に設定
- 送信チャネルでコード変換を行う場合
  - チャネルの属性をCONVERT(YES)
  - コード変換元：MQMDの“CodedCharSetId”フィールドにCCSIDをセット
  - コード変換先：接続先キュー･マネージャーのCCSIDに変換
- 受信側でMQGET時にコード変換を行う場合
  - コード変換元：MQMDの“CodedCharSetId”フィールドにCCSIDをセット
  - コード変換先：MQMDの“CodedCharSetId”に変換
デフォルトはキュー・マネージャーのCCSIDを使用

◆ ユーザーEXITでの組み込み

- メッセージがキャラクター、バイナリー混在の場合など
- EXITの雛型を提供（データ変換EXIT）
- 作成したEXIT名をMQMDの“Format”フィールドにセット

動的キュー

- アプリケーションが一時的に使用するキューを、動的に作成することが可能
  - テンプレートとなるモデル・キューをMQOPEN時に指定し、一時的にローカル・キューを作成
    - MQOPEN時、MQODの“ObjectName”フィールドにモデル・キュー名を指定
    - 動的キュー名は、MQODの“DynamicQName”フィールドで指定
      - フルネーム指定
      - 部分ネーム指定
        - "＊"の利用 ⇒ キュー・マネージャーがユニークな16文字を追加
      - MQODの“ObjectName”フィールドに生成された名前が戻る
    - 動的キューの属性は、指定されたモデル・キューの属性を使用
  - MQCLOSE、動的キューの削除に関する処理の選択が可能
    - MQCOオプションと、動的キューの種類によって異なる



動的キュー

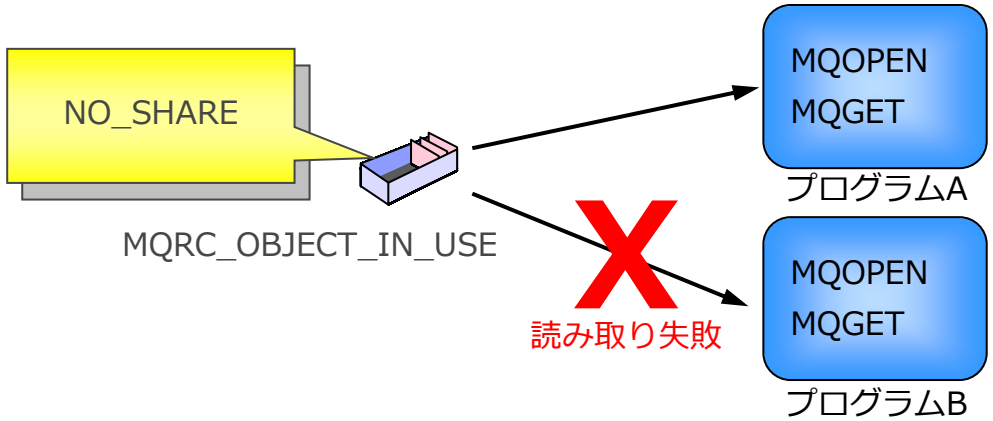
■ 動的キューの種類

◆ MQCLOSE時の動的キューの削除処理、キュー･マネージャー再起動時のリカバリー可否などが異なる

動的キューの種類	DEFTYPE属性	特性	対象メッセージ	MQCLOSE時のキューの削除処理
一時動的キュー	TEMPDYN	リカバリー不可能 キュー・マネージャー始動の際に削除	ノンパーシステント・メッセージ	MQCO_NONE →削除 MQCO_DELETE →削除 MQCO_DELETE_PERGE →削除
永続動的キュー	PERMDYN	システム障害の際リカバリー可能	ノンパーシステント・メッセージ パーシステント・メッセージ	MQCO_NONE →削除されない MQCO_DELETE →キューが空かつ未コミットメッセージがない場合削除 MQCO_DELETE_PERGE →キュー内のメッセージの有無にかかわらず、未コミットメッセージがない場合削除

排他使用

- キューを1つのアプリケーションだけから読み取れるようにし、他のアプリケーションを排他制御する（=読み取り不可）ことが可能
  - MQPUT用にキューをMQOPENするアプリケーションは排他制御の対象外
  - MQOPENオプション（MQOO_INPUT_EXCLUSIVEなど）、またはキューの属性（NOSHARE）にて設定

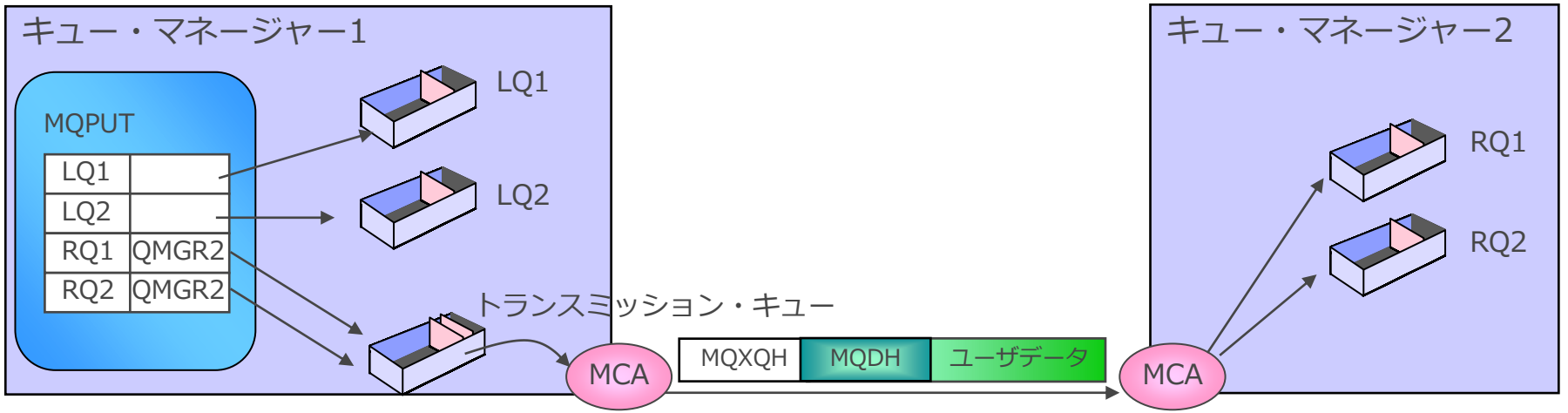


  - キュー属性とMQOPENオプションによる排他設定

キューの属性		MQOPENオプション		
SHARE | NOSHARE	DEFSOPT	AS_Q_DEF	SHARED	EXCLUSIVE
SHARE	SHARED	共用	共用	排他
SHARE	EXCLUSIVE	排他	共用	排他
NOSHARE	SHARED	排他	排他	排他
NOSHARE	EXCLUSIVE	排他	排他	排他

配布リスト

- 1回のMQPUT、 MQPUT1で複数の宛先にメッセージを送ることが可能
  - MQOPEN、MQCLOSEも1回でOK
  - リモートへは伝送キュー・ヘッダー（MQXQH）、及び配布ヘッダー（MQDH）がついたメッセージが1つだけ送られ、宛先で各キューにPUTされる
    - MQXQH、MQDHはキュー・マネージャーにより付与される
    - リモート側も配布リストをサポートする場合のみ使用可能
    - ネットワーク上の転送データを削減
  - アプリケーション内で配布リストを作成
    - MQODで配布リスト（宛先などの情報を設定）を指定してキューをOPEN



配布リスト

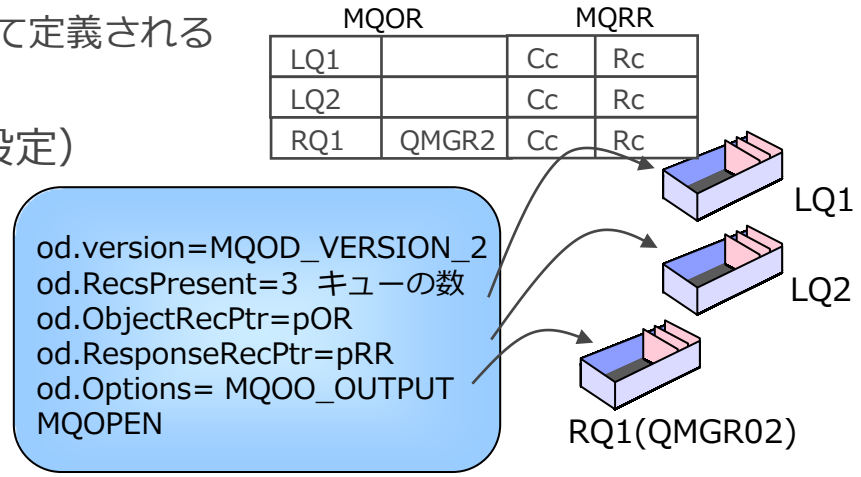
■ 配布リストの作成

◆ MQOD（オブジェクト記述子）で配布リストを指定してキューをOPEN

- MQOR（オブジェクト・レコード）
  - ObjectName：配布先のキュー名
  - ObjectQMgrName：配布先のキュー・マネージャー名
- MQRR（レスポンス・レコード）
  - CompCode：各キューから受け取る完了コード
  - ReasonCode：各キューから受け取る理由コード

◆ MQODの指定

- Versionフィールド ： MQOD_VERSION_2
- RecsPresentフィールド ： キューの数
  - 上記2つによりオブジェクト・ハンドルは1つ以上のキューのリストとして定義される
- ObjectTypeフィールド ： MQOT_Q
- ObjectName、ObjectQMgrNameはブランク（配布リストから設定）
- 次のいずれかのフィールドでMQOR構造体の位置を指定
  - ObjectRecOffset
  - ObjectRecPtr
- 次のいずれかのフィールドでMQRR構造体の位置を指定
  - ResponseRecOffset
  - ResponseRecPtr



配布リスト

■ 配布リストへのメッセージ書き込み

◆ MQPUT時、メッセージ毎に違うMQMDをセットすることが可能（オプション）

- MQPMR（メッセージ・レコード）：個々のメッセージに個別のMQMDフィールドを指定
- MQRR（レスポンス・レコード）　：個々のキューに対する完了コード、理由コードを返す

◆ MQPMOの指定

- Versionフィールド：MQPMO_VERSION_2
- RecsPresentフィールド：MQPMR、MQRRの数を指定
- PutMsgRecFieldsフィールド　：設定するMQMDフィールドを指定
  - MsgId, CorrelId, GroupId, Feedback, AccountingToken のみ
- PutMsgRecOffset, PutMsgRecPtr：どちらかのフィールドでMQPMR構造体の位置を指定
- ResponseRecOffset，ResponseRecPtr：どちらかのフィールドでMQRR構造体の位置を指定

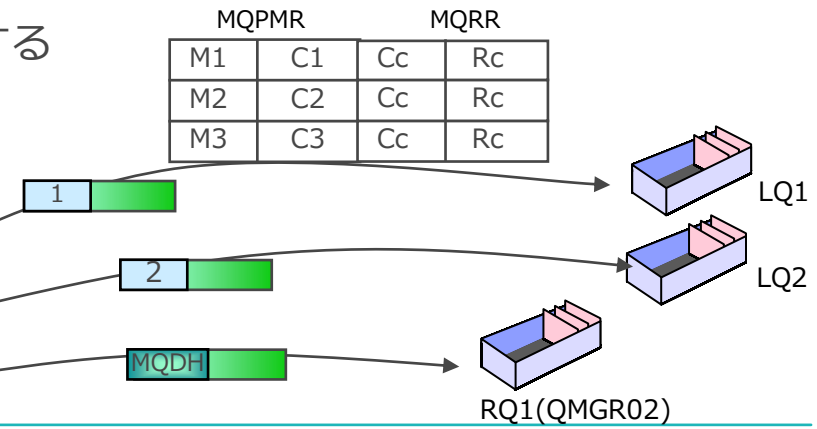
◆ MQPMRの作成

- 固定したレイアウトがないため使用するフィールドに合わせて構造体を定義する

例：MsgIdとCorrelIdをセットしたい場合

```
typedef struct{
MQBYTE24 MsgID
MQBYTE24 CorrelId
} PutMsgRec;
```

```
pmo.version=MQPMO_VERSION_2
pmo.RecsPresent=3
pmo.PutMsgRecPtr=pPMR
pmo.PutMsgRecFields=MQPMRF_MSG_ID
                    |MQPMRF_CORREL_ID
pmo.ResponseRecPtr=pRR
od.Options= MQOO_OUTPUT
MQPUT
```



トランザクション・サポート

同期点処理

- 複数のメッセージ送受信を1つの作業単位（UOW）として処理することが可能
  - ◆ 同期点はMQCMIT / MQBACKで確定される

	オプション項目	パラメーター	説明
メッセージ送信	MQPMO.Options	MQPMO_NO_SYNCPOINT	同期点処理に参加しない
		MQPMO_SYNCPOINT	同期点処理に参加する
メッセージ受信	MQGMO.Options	MQGMO_NO_SYNCPOINT	同期点処理に参加しない
		MQGMO_SYNCPOINT	同期点処理に参加する
		MQGMO_SYNCPOINT_IF_PERSISTENT	メッセージがパーシステントであれば同期点処理に参加

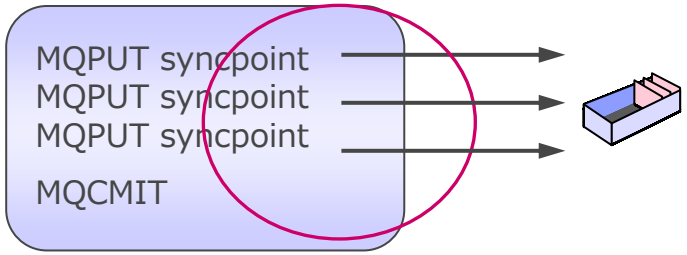
  - ◆ 同期点付でMQGETしたメッセージは、MQCMITでキューから削除される
    - ● ただし、MQGET時点でメッセージは他のアプリケーションからは見えなくなり、同じメッセージが2つのアプリケーションからMQGETされることはない
    - ● アプリケーションで確実にメッセージを受信するためには同期点付のMQGETが必要
      - 同期点なしでMQGETするとキュー内のメッセージは即時に消され、実行結果のメッセージがアプリケーションに返らないことがあるため
  - ◆ 同期点付でMQPUTしても、MQCMITまで他のアプリケーションからメッセージは見えない
    - ● ただし、即時にメッセージはキューに書き込まれるため、カーソル使用時はMQGET BROWSEでのメッセージの呼び飛ばしに注意
  - ◆ 同期点を明示的に指定しなかった場合、プラットフォームに依存したデフォルト値を使用
    - ● z/OS：MQxMO_SYNCPOINT
    - ● IBM i / AIX / Windows / Linux：MQxMO_NO_SYNCPOINT

同期点処理

■ 同期点処理の例1

◆ 1UOWでのメッセージ送信

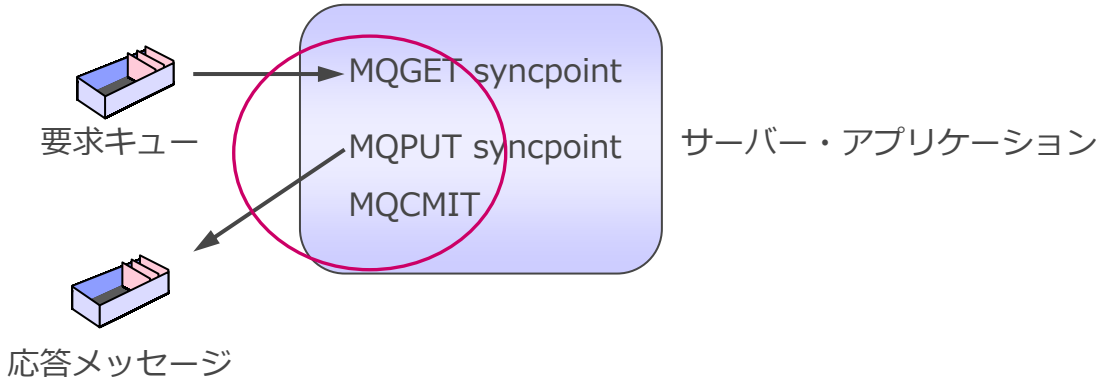
- 関連する複数のメッセージをすべて送信するか / すべて送信しないか制御できる



■ 同期点処理の例2

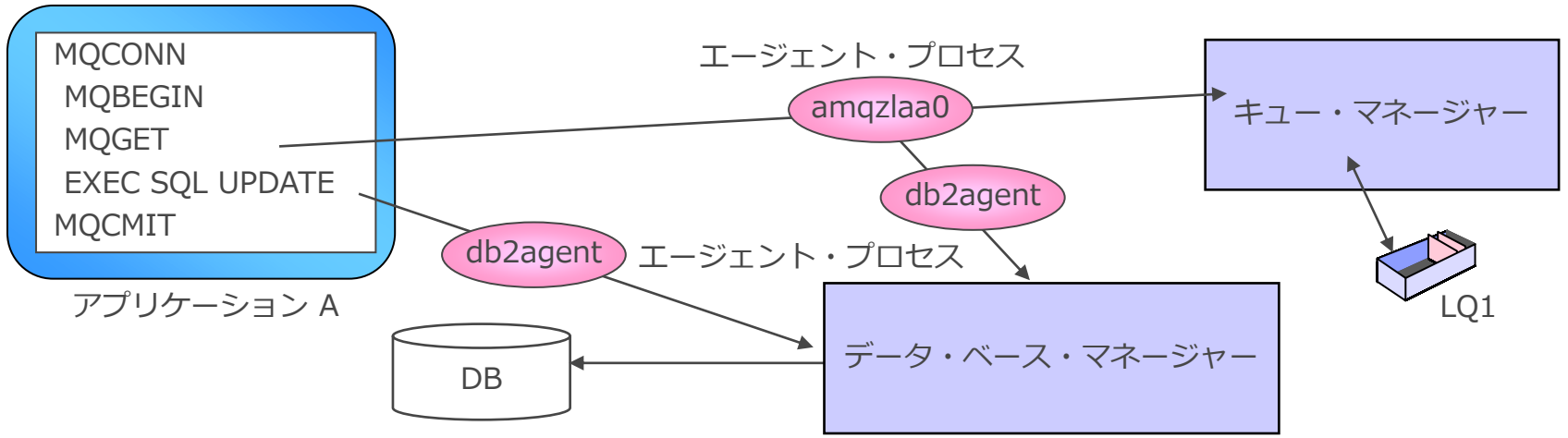
◆ 応答メッセージの確実な返却

- 同期点付きでMQGETしていれば応答メッセージ送信に失敗してもやり直すことができる



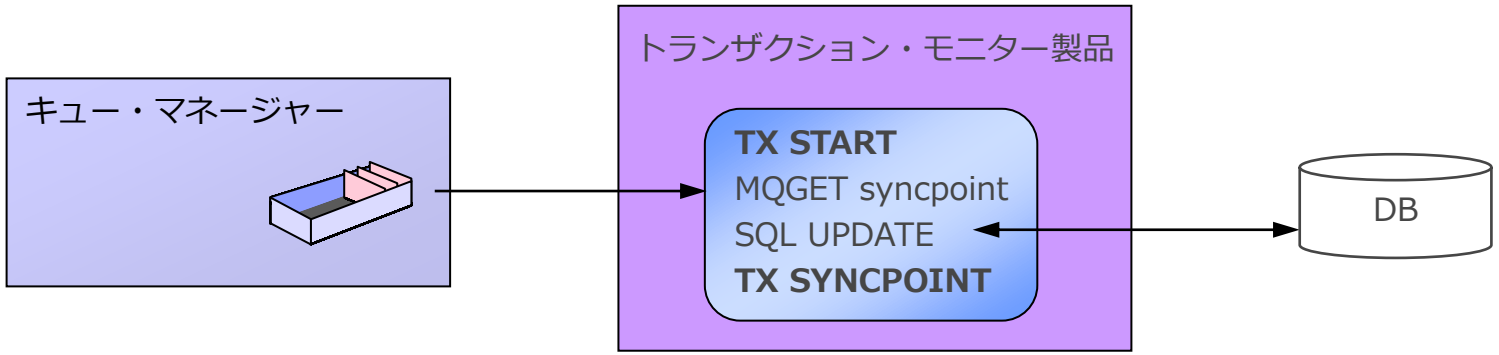
データベース・コーディネーション

- キュー・マネージャーがリソース・コーディネーターとなることが可能（キュー・マネージャー・コーディネーション）
  - ◆ XAインターフェースによる2フェーズ・コミットを行う
    - MQ資源の更新とデータベース更新を１つのUOWで処理が可能
  - ◆ アプリケーションはMQBEGINでグローバル作業単位（UOW）を開始
  - ◆ MQCMIT / MQBACKで作業単位を終了
  - ◆ キュー・マネージャーとアプリケーションはローカル接続している必要がある
    - クライアント接続は不可
    - アプリケーションとデータベースはローカル / リモート接続どちらでもOK
  - ◆ トランザクション内で同期点なしでのMQPUT / MQGETは可能



<補足>データベース・コーディネーション

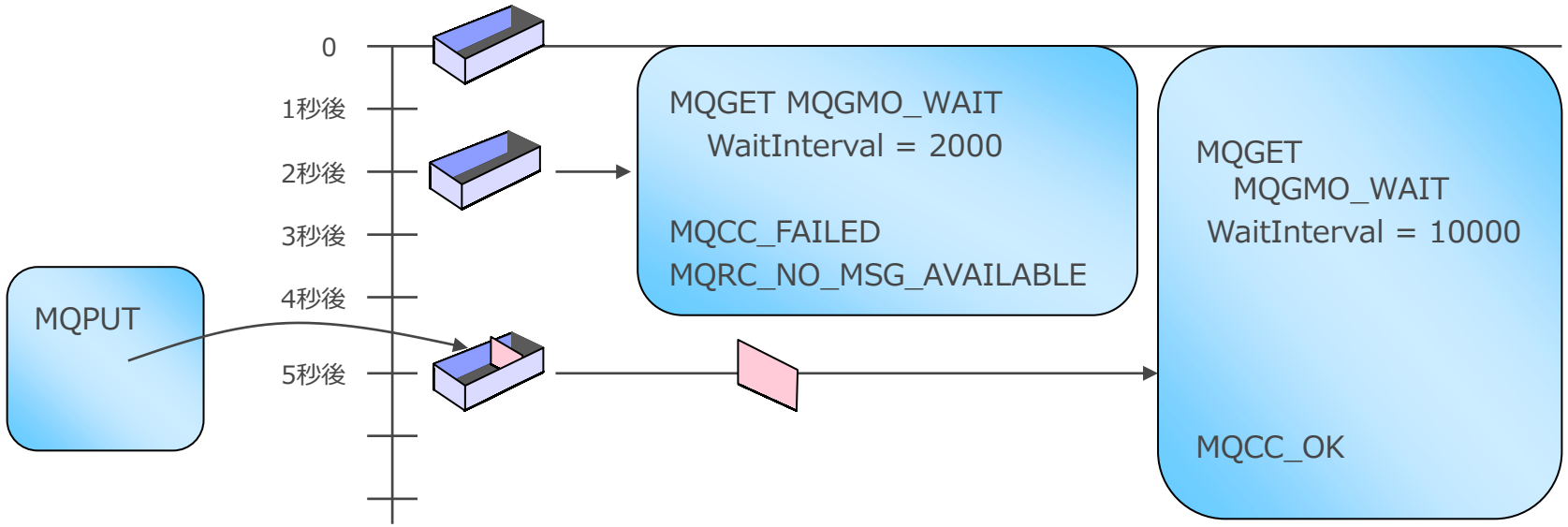
- 外部コーディネーション
  - ◆ トランザクション・モニター製品がトランザクション・コーディネーターとなる
    - トランザクション・モニター製品：WAS、CICSなど
  - ◆ トランザクション・モニター製品の提供するAPIで作業単位の開始と終了を行うためMQBEGIN / MQCMIT / MQBACKを発行することはできない
  - ◆ MQクライアント接続で外部コーディネーションを行う場合、通常のMQクライアントではなく、MQトランザクショナル・クライアントを使用する必要がある
  - ◆ アプリケーションをJavaで作成する場合、JMSを使用する必要がある
    - MQ Base Javaの場合、外部コーディネーションはできない



メッセージ・ドリブン処理

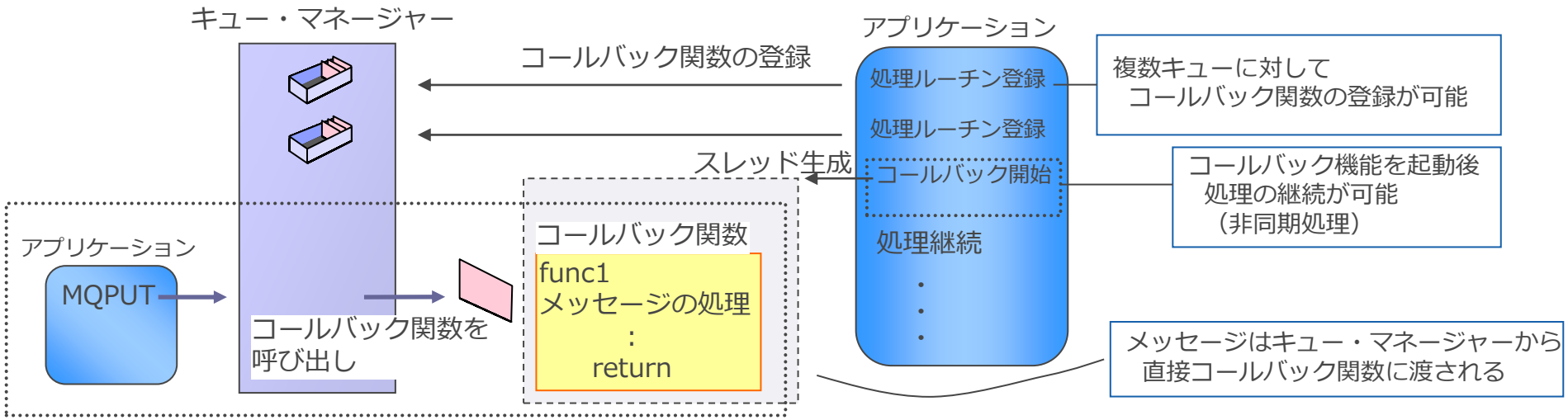
メッセージの待機受信

- キューにメッセージがMQPUTされるのを待ち、受信メッセージを処理可能
  - ◆ 待機時間を経過するか、メッセージがキューに書き込まれるまでメッセージを待機する
    - MQGETはデフォルトではメッセージを待機しない、MQGET時にキューにメッセージがなければエラー
      - MQGMO.Options=MQGMO_NO_WAIT
  - ◆ MQGMO.Optionsに“MQGMO_WAIT”を指定してMQGET
    - 指定時間後にMQGETするのではなく、指定時間MQPUTされるのを待つ
  - ◆ MQGMOの“WaitInterval”フィールドに待ち時間を指定（ms）
    - デフォルト “MQWI_UNLIMITED”指定：メッセージを無制限に待つ



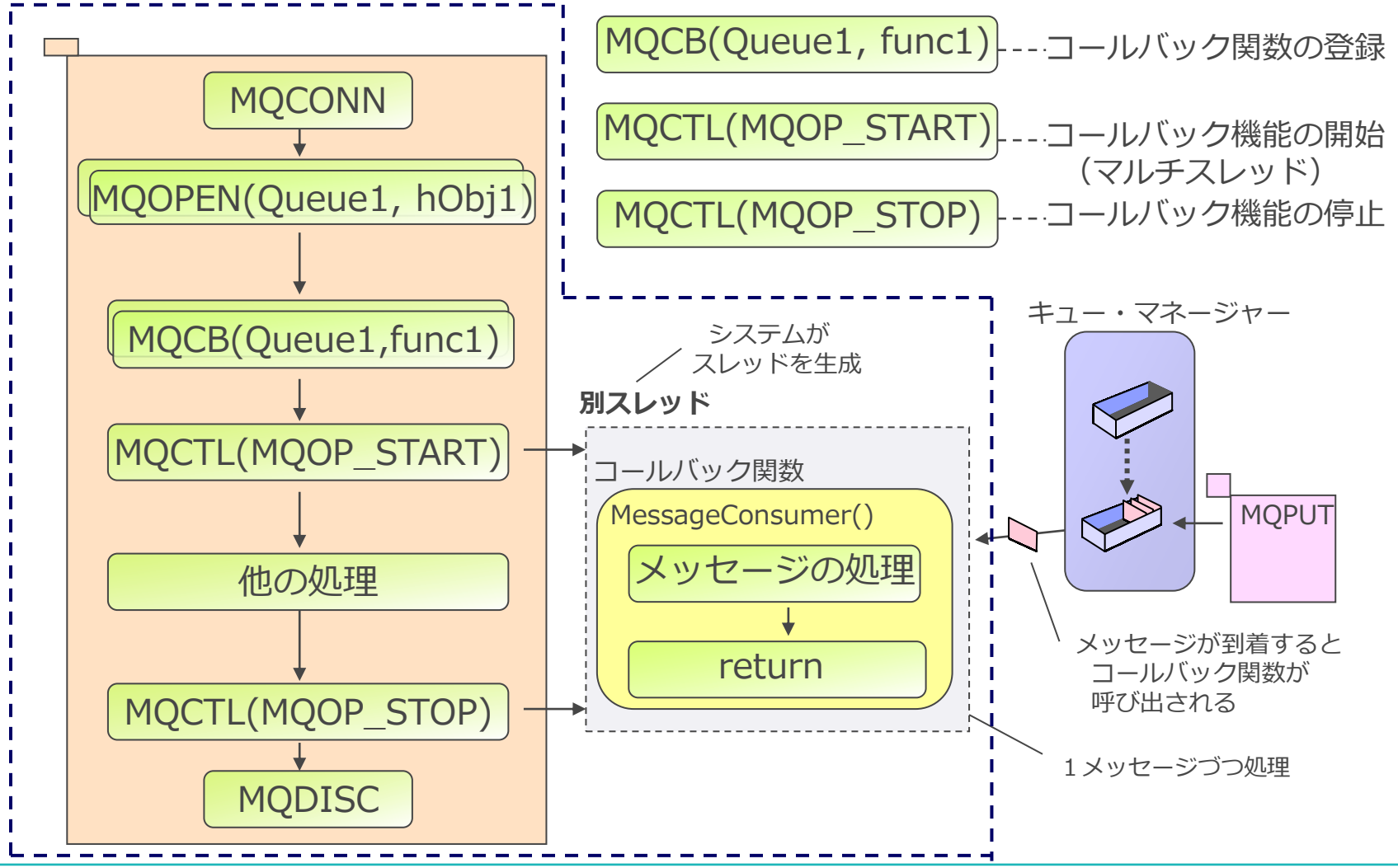
コールバック機能

- 非同期的にメッセージを受信する機能
  - ◆ キュー・マネージャーにコールバック関数の登録（MQCBコール）をし、コールバック機能の開始（MQCTLコール）
    - コールバック関数とは、メッセージがキューに到着したときに、アプリケーションが開始したスレッドから呼び出される処理ルーチン
  - ◆ キューにメッセージが到着すると、コールバック関数が呼び出される
    - プログラムの関数、もしくはダイナミックリンク・ライブラリで実装可能
  - ◆ 複数キューに対してコールバック関数を登録することが可能
  - ◆ アプリケーションはメッセージの到着を待つために、処理をブロックする必要がなくなる



<参考>コールバック機能

■ コールバック機能を用いた基本的なアプリケーションの流れ



<参考>コールバック機能

コールバック機能の制約

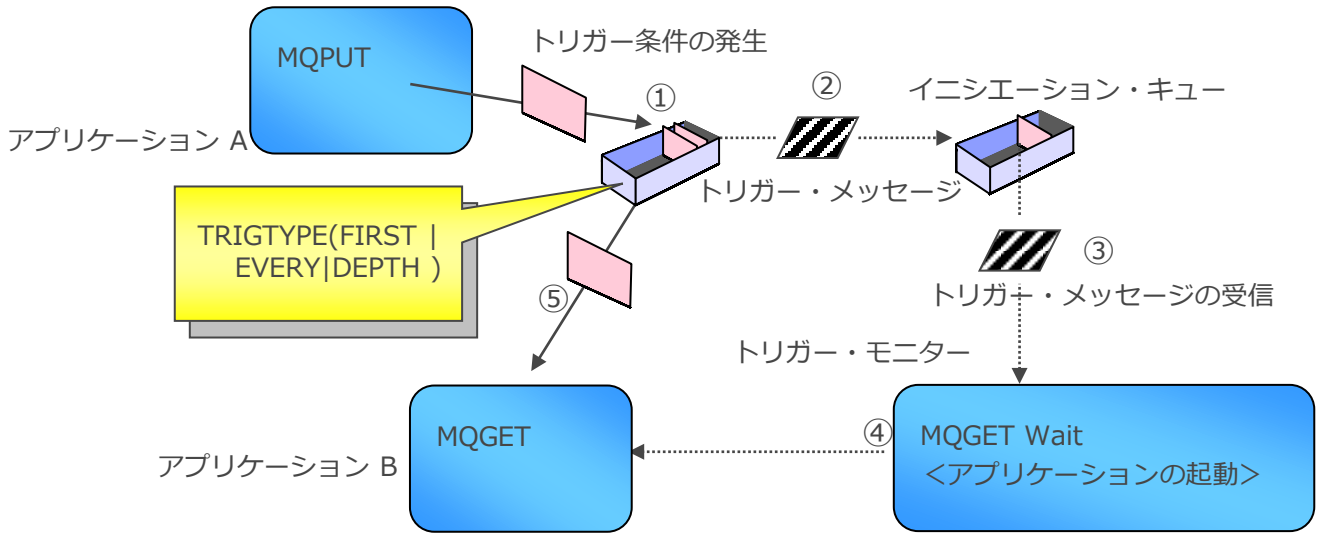
- 1つのコネクション・ハンドルに対して生成されるスレッドは1つのみ
  - メッセージはキューに到着した順に、1メッセージずつ処理される
    - 複数キューに対して、コールバック関数を登録していた場合も同様
  - 複数のスレッドを生成し並行してメッセージを待機するためには、MQCONNを複数回実行する必要がある

- コールバック機能を開始後、コールバック関数を登録したコネクション・ハンドルを通してMQI（MQCTL、MQDISC以外)を実行することはできない
  - MQRC_HCONN_ASYNC_ACTIVEエラーが返る
  - MQCTL、MQDISC以外のMQIを実行するためには、MQCONNをもう一度実行する、もしくは、MQIを用いない処理を行う必要がある

トリガー

- メッセージがMQPUTされたのをトリガーに特定のプログラムの起動が可能
  - トリガー条件を設定し、キューがトリガー条件を満たした時に
    キュー・マネージャーはトリガー・メッセージ（MQTM）をイニシエーション・キューに書き出す
  - 常駐トリガー・モニターがトリガー・メッセージを受信してアプリケーションを起動
    - 起動するアプリケーションはキューごとに固定
  - 起動するアプリケーションの名前はプロセス定義に設定する

  - トリガーに必要な環境
    - ローカル・キューのトリガー設定（トリガー制御：ON）
    - 関連付けされたイニシエーション・キュー
    - 関連付けされたトリガリングのプロセス定義
    - トリガー・モニター・プログラムの常駐



トリガー

■ トリガーの発生条件

◆ ローカル・キューの“TRIGTYPE”属性で指定

トリガリングのタイプ	説 明
FIRST	メッセージ`数が0から1に変わった時にトリガリングを起動 メッセージ到着毎に起動することも可能 (対象のキューが空でなくても､次のメッセージ`が入力された時点でキュー・マネージャー属性のTriggerInterval値を過ぎていればトリガー・メッセージを生成)
EVERY	メッセージの到着毎にトリガリングを起動
DEPTH	メッセージが指定した個数に達した時にトリガリングを起動 トリガリング起動後､トリガー設定は"OFF"になる (ユーザでのトリガー“ON”(MQSETコール)が必要)

◆ 指定したプライオリティー以上のメッセージがPUTされた時のみトリガーの対象とすることが可能

- ローカル・キューの“TRIGMPRI”属性で指定

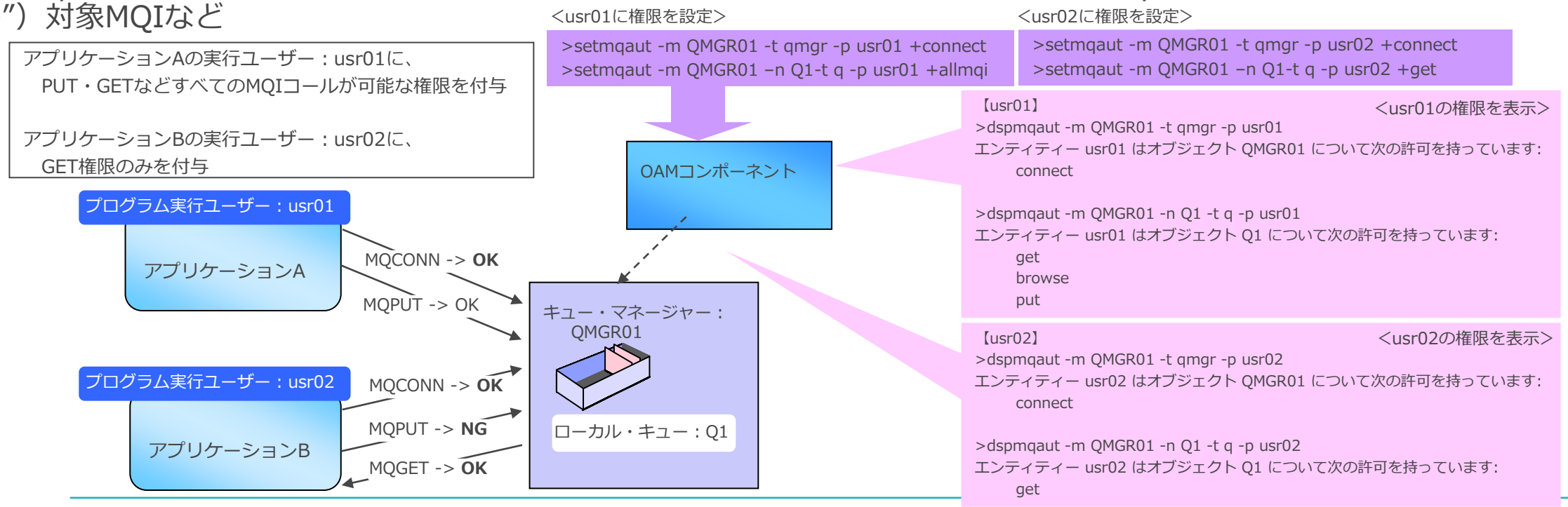
セキュリティ

許可サービス

- キュー・マネージャーやMQオブジェクトに対し、アクセス許可 / 拒否の設定が可能
  - 操作するユーザーやグループに対してアクセスの可否を設定する
  - デフォルトではmqmユーザー、mqmグループがMQの管理権限を持つ
  - LDAPユーザーやOSに存在しないユーザーに対しても権限付与が可能
- OAM（Object Authority Manager）がMQオブジェクト毎の制御を行う
  - アクセス制御可能なオブジェクト
    - キュー･マネージャー
    - キュー
    - プロセス
    - ネームリスト
  - 権限のチェックはMQCONN、MQOPEN時
  - 権限情報はパーシステント・メッセージとしてシステム・キュー（SYSTEM.AUTH.DATA.QUEUE）で保管
  - 以下のコマンドでアクセス制御を設定
    - dspmqaut ： 権限の表示
    - dmpmqaut ： 権限のダンプ
    - setmqaut ： 権限の設定とリセット

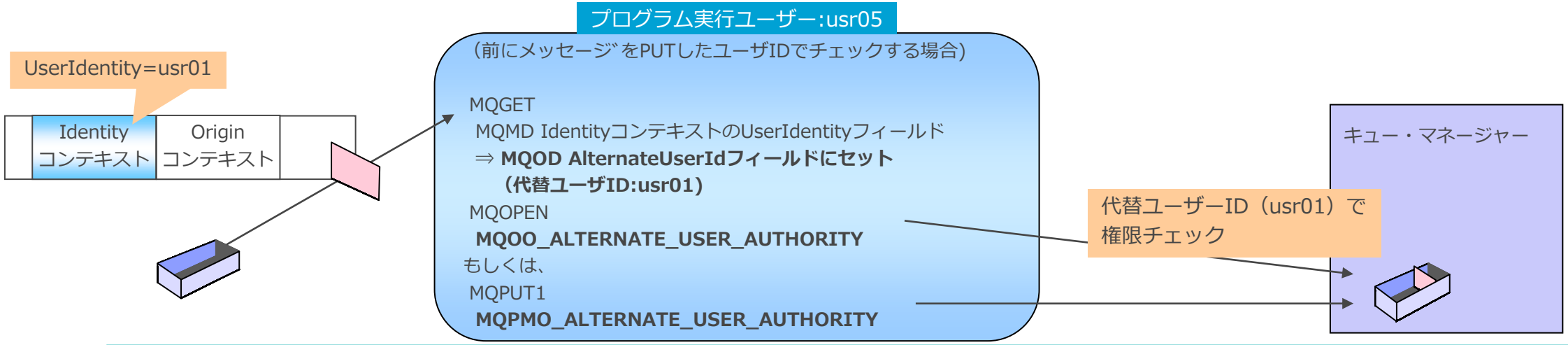
<参考>許可サービス

- OAM管理コマンド文法例
  - dspmqaut
    - dspmqaut -m キュー・マネージャー名 -n オブジェクト名 –t オブジェクト・タイプ -p ユーザ名
  - dmpmqaut
    - dmpmqaut -m キュー・マネージャー名 -n オブジェクト名
  - setmqaut
    - setmqaut -m キュー・マネージャー名 -n オブジェクト名 –t オブジェクト・タイプ -p ユーザー名 +（削除の場合は “-”）対象MQIなど



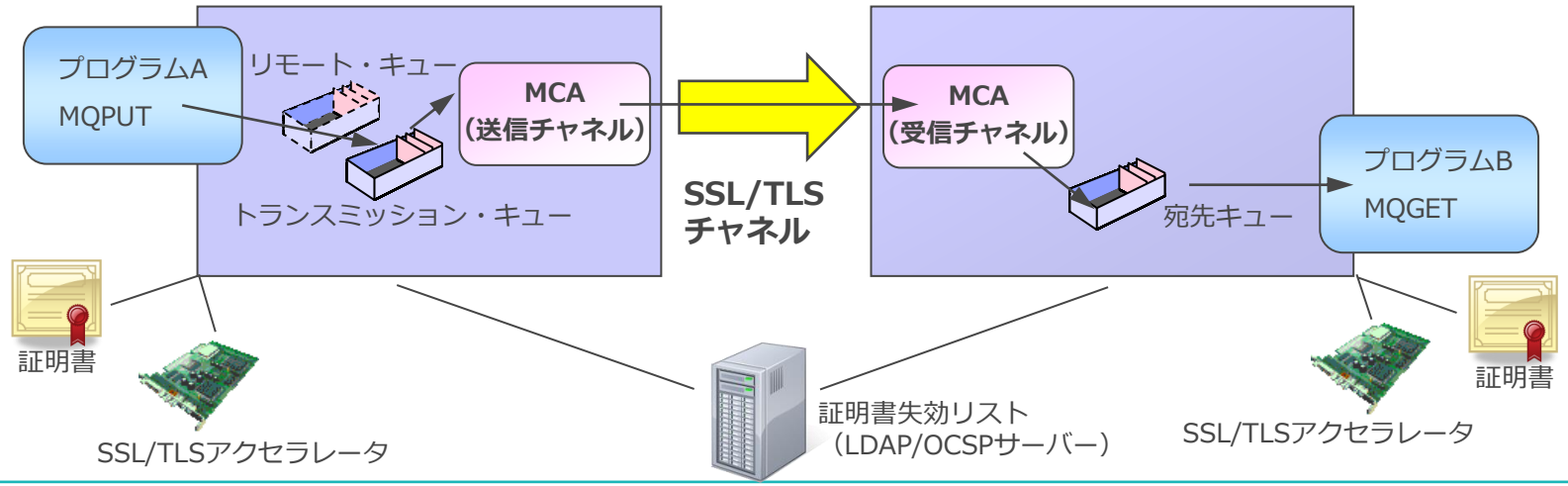
代替ユーザーID

- プログラム実行ユーザー以外のユーザーにてMQOPEN / MQPUT1することが可能
  - キューのOPEN時にセキュリティー・チェックを行う場合下記の方法がある
    1. プログラム実行ユーザーIDでのチェック（デフォルト）
    2. 代替ユーザーIDでのチェック
       - 実行ユーザー以外のユーザーIDでセキュリティ・チェック
  - 代替ユーザーIDを使用する場合
    - 代替ユーザーIDの設定
      - MQODの“AlternateUserId”フィールド
    - 代替ユーザーでのMQOPEN
      - MQOPENオプション（MQOO_ALTERNATE_USER_AUTHORITY）を指定
      - MQPUT1時のMQPMOのオプション（MQPMO_ALTERNATE_USER_AUTHORITY）を指定



SSL/TLSチャネル

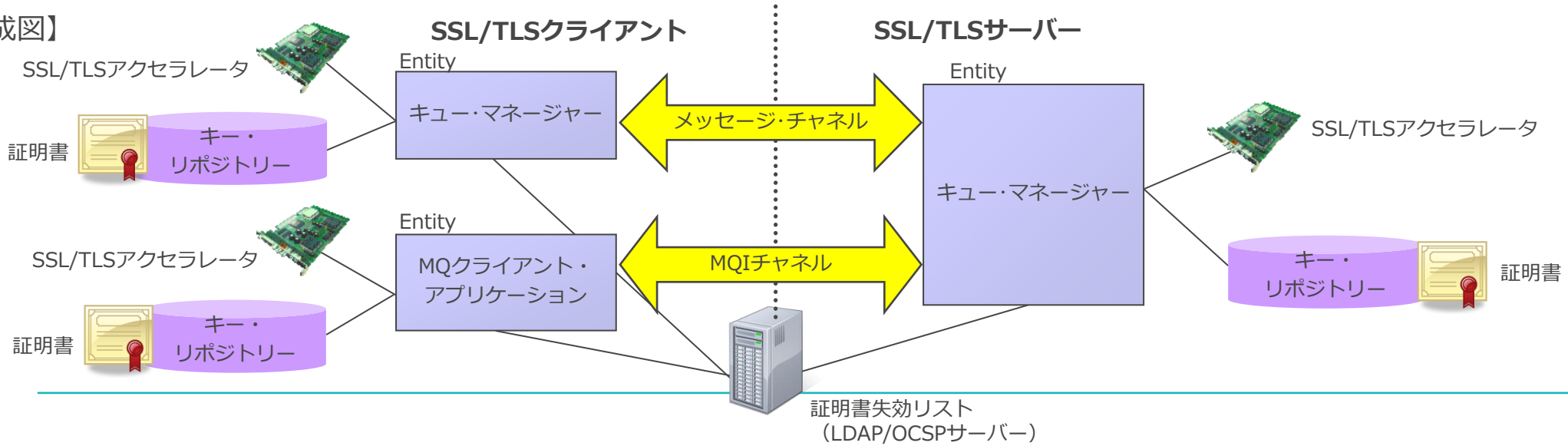
- SSL/TLSを使用して暗号化されたチャネル接続によりセキュアなメッセージの送受信が可能
  - ◆ 送受信中のメッセージ・データの改ざんや、なりすましを防ぐことができる
  - ◆ 全てのチャネル・タイプで使用可能
    - メッセージ・チャネル、MQIチャネル、クラスター・チャネル
  - ◆ CipherSpecの選択が可能
  - ◆ 証明書の識別名（DN）によるフィルタリングが可能
    - 証明書のDNをもとに接続の許可 / 拒否を制御することができる
  - ◆ 証明書失効リスト（CRL）への照会
    - LDAPサーバーやOCSPサーバーへのアクセス設定



SSL/TLSチャネル

- MQでのSSL/TLSの構成
  - ◆ SSL/TLSの構成を行う場合、SSL/TLSのサーバー側とクライアント側の両Entityにそれぞれで証明書、暗号鍵を保管しておく必要がある
    - EntityとはSSL/TLSのサーバーやクライアントとなるキュー･マネージャーおよびMQクライアント･アプリケーション
  - ◆ Entityがキュー・マネージャーの場合
    - キュー･マネージャー属性SSLKEYRに証明書を保存しているキー・リポジトリの名前を設定
    - チャネル属性SSLCIPHに使用するCipherSpecの名前を指定
      - 指定するCipherSpecは通信相手と合わせる必要がある
      - V9.3以降は、条件に合うCipherSpecをネゴシエーションにより決定させる”ANY”の指定が可能
    - CRLを参照する場合は、NAMELIST（参照するAUTHINFO名）、AUTHINFO（LDAP/OCSPサーバーの情報）を定義し、キュー･マネージャー属性SSLCRLNLに作成したNAMELISTを指定

【SSL/TLS構成図】



SSL/TLSチャネル

- MQでのSSL/TLSの構成（続き）
  - ◆ EntityがMQクライアントの場合、接続方法は2種類
    1. チャネル定義テーブルを使用
       - クライアント接続チャネルを定義し、チャネル接続テーブルを作成
         - チャネル属性SSLCIPHに使用するCipherSpecの名前を指定（必須）
         - チャネル属性SSLPEERに接続を許可する証明書の識別名を指定（オプション）
       - 環境変数の設定
         - MQSSLKEYRにSSL/TLSキー・リポジトリーのロケーションを指定（必須）
         - MQSSLCRYP（AIX,Linux,Windowsのみ）に使用する暗号ハードウェア構成を指定（オプション）
       - クライアント接続チャネルを作成したキュー・マネージャーと同一のLDAP/OCSPサーバーを参照するようチャネル定義ファイルに設定されているためLDAP/OCSPサーバーに関する設定はなし

MQでのSSL/TLSの構成

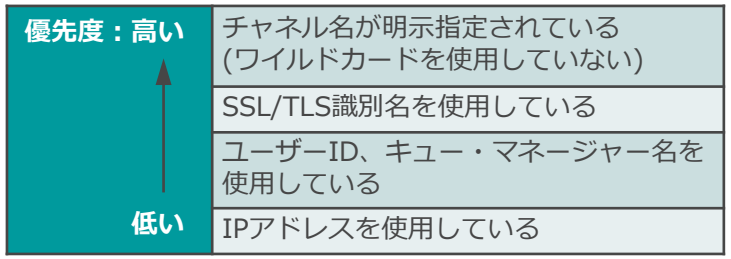
EntityがMQクライアントの場合、接続方法は2種類（続き）

2. MQCONNXを使用
   - チャネル定義(MQCD)構造体の拡張
     - SSLCipherSpecフィールドに使用するCipherSpecの名前を指定
     - SSLPeerNamePtrフィールドに接続を許可する証明書の識別名を指定
   - 接続オプション(MQCNO)構造体の拡張
     - SSLConfigPtrフィールドにSSL/TLS構成オプション（MQSCO）構造体のアドレスを設定
     - SSLConfigOffsetフィールドにSSL/TLS構成オプション（MQSCO）構造体のオフセットを設定
   - SSL/TLS構成オプション（MQSCO）構造体の提供
     - KeyRepositoryフィールドにSSL/TLSキー・リポジトリーのロケーションを指定
     - SSLCryptoHardware（AIX,Linux,Windowsのみ）フィールドにSSL/TLSアクセラレータを設定
     - AuthInfoRecOffsetフィールド、AuthInfoRecPtrに最初のMQAIRの情報を設定
   - 認証情報レコード（MQAIR）構造体の提供
     - LDAPサーバーの場合
       - AuthInfoConnNameフィールドやLDAPUserNamePtrなどにLDAP情報（LDAPサーバー接続名、接続ユーザー名、パスワードなど）を設定
     - OCSPサーバーの場合
       - OCSPResponderURLフィールドにOCSPサーバーに接続可能なURLを指定

チャネル・アクセス制御

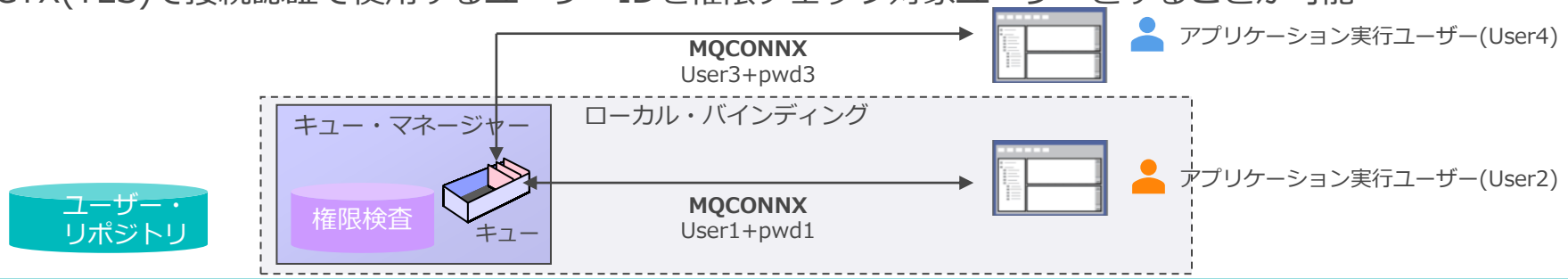
- チャネル認証レコードを使用して、チャネル・レベルで接続システムへのアクセスを制御
  - ◆ 接続要求に対するブロックの実施
    - 接続元の接続情報に応じて、接続をブロックすることが可能
  - ◆ 接続条件に応じた特定ユーザーへのマッピングの実施
    - 接続元の接続情報に応じて、特定の権限ユーザーへのマッピングが可能
    - マッピングに利用する権限ユーザーは、以下を指定可能
      - チャネル認証レコードのMCAUSER属性に指定したユーザー
      - 使用チャネルの接続認証に利用されるユーザー
      - NOACCESS （設定すると、接続のブロックと同義）
    - アクセス制御に使用可能な接続情報
      - IPアドレス または ホスト名
      - ユーザーID（クライアント接続のみ）
      - キュー・マネージャー名（サーバー間接続のみ）
      - SSLまたはTLSのDN
  - ◆ SET CHLAUTHコマンドで認証レコードを作成
    - ブロック、マッピングなどのタイプ別に認証レコードを作成
  - ◆ 1つのチャネル定義に、複数の認証レコードを定義可能
    - チャネル名、キュー・マネージャー名、SSL/TLS識別名、ホスト名、IPアドレスなどに、複数のチャネル認証レコードが合致する場合は、最も具体的に一致するものが使用される
      - ワイルドカードが含まれている場合は、使用位置などを元に判断

チャネル認証レコードの選択アルゴリズム



接続認証

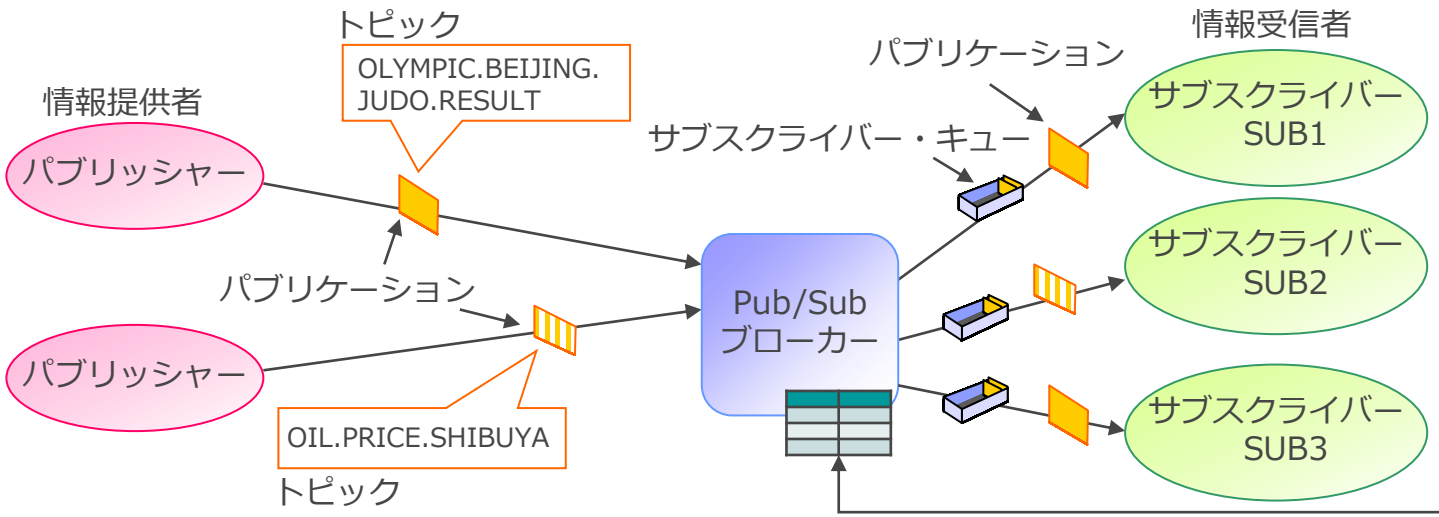
- MQアプリケーションからの接続(ローカル/クライアント)をユーザーID/パスワードで認証する機能
  - ◆ MQ接続(クライアント/ローカル)時にアプリケーションがユーザーID/パスワードを提供
    - アプリケーションはMQCSP構造体を使用してユーザーID/パスワードをキュー・マネージャーに渡す
      - MQCSPのパスワードフィールドは暗号化で保護可能
  - ◆ キュー・マネージャー側で、ユーザーID/パスワードに従って動作するように構成可能
  - ◆ 認証方法はキュー・マネージャーに指定
    - キュー・マネージャーのCONNAUTH属性にAUTHINFOオブジェクト名を指定
  - ◆ MQ外部のユーザー・リポジトリを使用してユーザーID/パスワードの組み合わせをチェック
    - 各キュー・マネージャーで使用するために選択できる認証情報オブジェクトは 1 つのタイプのみ
      - IDPWOS：ローカル・オペレーティング・システムを使用
      - IDPWLDAP：LDAP サーバーを使用(z/OS版は指定不可)
  - ◆ 接続認証で使用するユーザーIDとアプリケーション・ユーザーIDを分離することが可能
    - OAMの権限チェックはアプリケーション・ユーザーIDに対して行われる
      - ADOPTCTX(YES)で接続認証で使用するユーザーIDを権限チェック対象ユーザーとすることが可能



その他の機能

Publish/Subscribe

- Pub/Subとは
  - ◆ 情報の提供者（パブリッシャー）と受信者（サブスクライバー）が、トピックを介して行うメッセージング形態の1つ
    - 提供者と受信者間は、提供者⇒受信者の一方向通信
  - ◆ 疎結合なメッセージ連携
  - ◆ 共通の”話題”を購読要求した受信者に対して、送信側は受信者の数やロケーションを認識せずにメッセージ配信が可能
    - MQの配布リストの場合は、受信者の宛先などを知っている必要がある
  - ◆ パブリッシャーが発信(送信)したパブリケーションを、ブローカーが受信・配信し、それをサブスクライバーが受信



サブスクライバー	トピック
SUB1	OLYMPIC.BEIJING.JUDO.RESULT
SUB2	OIL.PRICE.SHIBUYA
SUB3	OLYMPIC.BEIJING.JUDO.RESULT

Pub/Sub：構成要素

- トピック
  - ◆ 情報提供者 (パブリッシャー)と情報受信者(サブスクライバー)を連携する、共通の”話題”
- パブリケーション
  - ◆ パブリッシャーとサブスクライバーがやり取りするメッセージ
- パブリッシャー
  - ◆ パブリッシャーは、トピックを宛先として情報を配信
- サブスクライバー
  - ◆ パブリケーションを受信するために、購読要求(サブスクリプション)を登録する
- サブスクライバー・キュー
  - ◆ サブスクライバーがパブリケーションを受信するためのキュー
- サブスクリプション
  - ◆ サブスクライバーがパブリケーションを受信するために登録する情報。トピックやサブスクライバー・キューなどの情報を含む

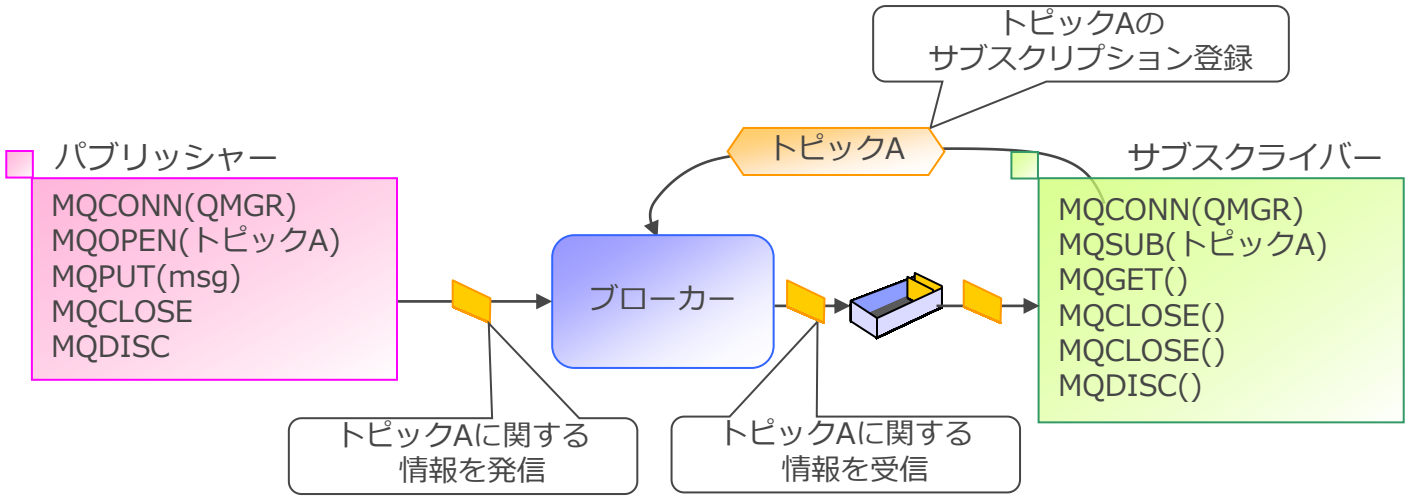
Pub/Sub：構成要素

- ブローカー
  - パブリケーションの受信/配信と、パブリッシャー、サブスクライバーの管理を行う
    - パブリッシャーからパブリケーションを受け取り、サブスクライバーに配信
    - サブスクライバーからのサブスクリプションを登録

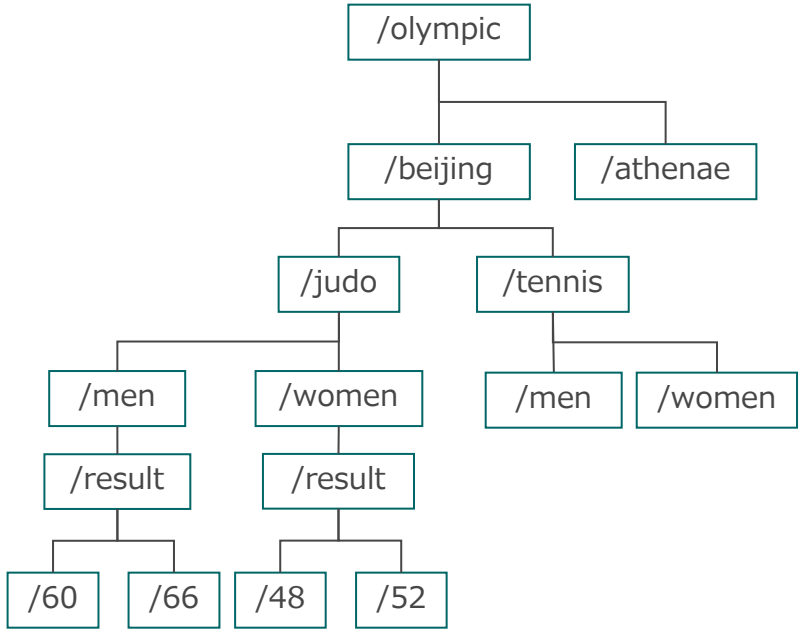
※ ここではブローカーとは、MQでPub/Subの機能を提供するエンジンとします。
App Connect Enterpriseでの機能を提供するということではありません。

Pub/Sub：トピック

- トピック
  - ◆ パブリッシャーとサブスクライバーを紐付ける共通の話題
  - ◆ 階層構造（トピック・ツリー）で表記することができる
  - ◆ パブリッシャーがパブリケーションを発信する宛先
  - ◆ サブスクライバーがパブリケーションを受信するために指定するもの
    - ● ワイルドカードを用いた曖昧指定も可能

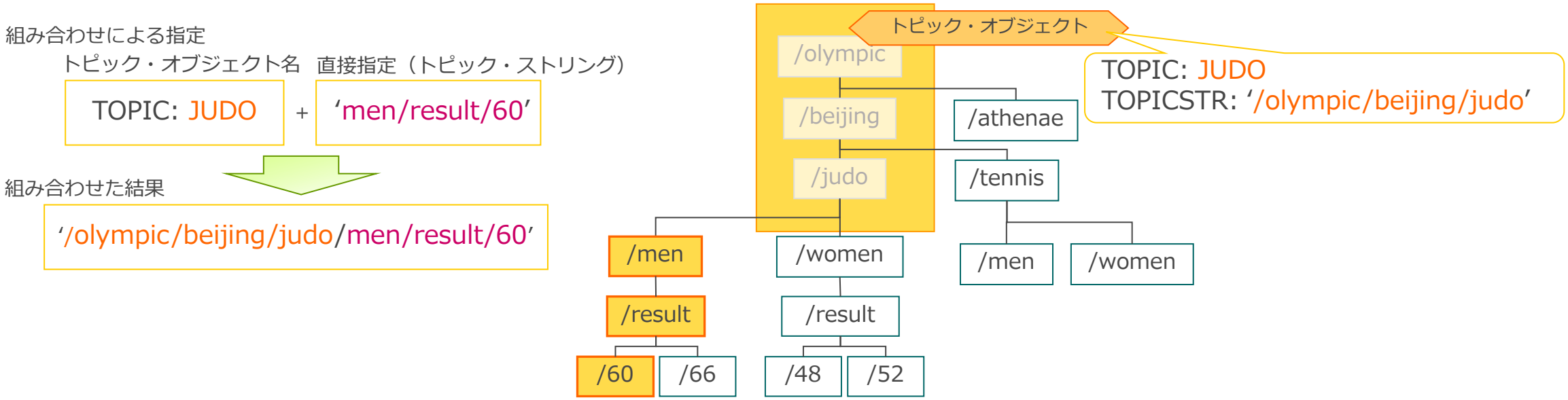


【トピック・ツリー】



Pub/Sub：トピック

- パブリッシャー/サブスクライバーによるトピック指定方法は3通り
  - トピック・ストリングで直接設定
    - スラッシュを階層の区切り文字としてノードを表現する文字列
  - トピック・オブジェクト名で指定
    - トピック・ストリングをキュー・マネージャー・オブジェクトとして定義したもの
    - トピック・オブジェクトに定義されたトピック・ストリングが設定される
  - 上記2つの組み合わせ
    - トピック・オブジェクトに定義されているトピック・ストリング　＋　指定したトピック・ストリングに解決

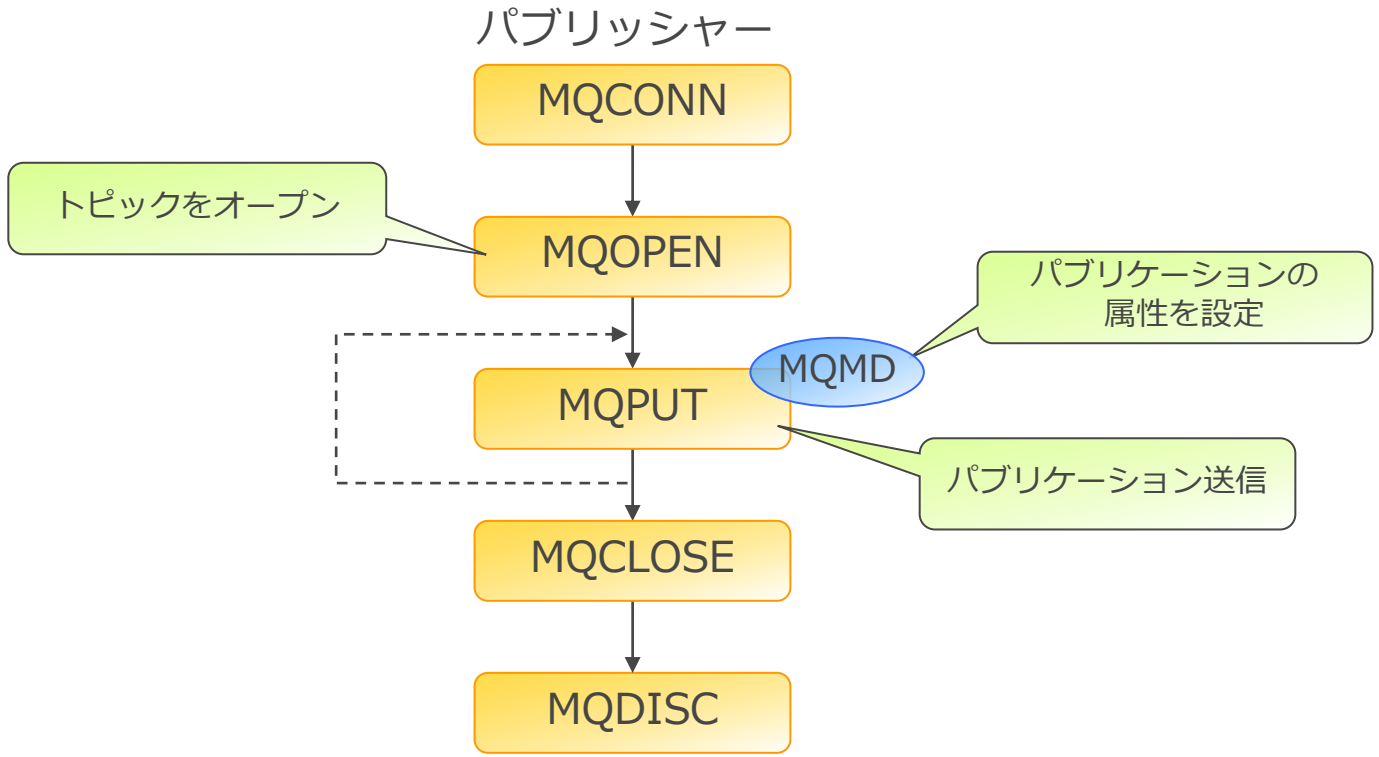


Pub/Sub：パブリケーション

- 2つのパブリケーション・タイプ
  - アプリケーション要件に応じたパブリケーションをMQPMOで選択可能
- ノーマル・パブリケーション(イベント型)
  - パブリッシュした時にサブスクライブしている相手に送信する
  - 継続的に配信されるが、個々には論理的に独立したメッセージ群
    - 株価の売買情報、ブログの更新情報など
  - パブリケーションは、ブローカーによって保管されない
    - パブリッシュ時点で登録されているサブスクライバーにのみ配信される
  - MQでのデフォルト・パブリケーション
- リテイン・パブリケーション(ステート型)
  - サブスクライブ登録やサブスクライバーの稼動に関わらず、トピックの最新情報を送信する
  - 事象の状態を変化を示すもので、時々刻々更新/置換される情報
    - 株価、室内の温度/湿度など
  - パブリケーションは、ブローカーによって保管される
    - パブリッシュ後に登録したサブスクライバーにも最新のパブリケーションが配信される
  - MQではステート型のパブリケーションをリテイン・パブリケーション(リテイン・メッセージ) という

Pub/Sub：パブリッシャー

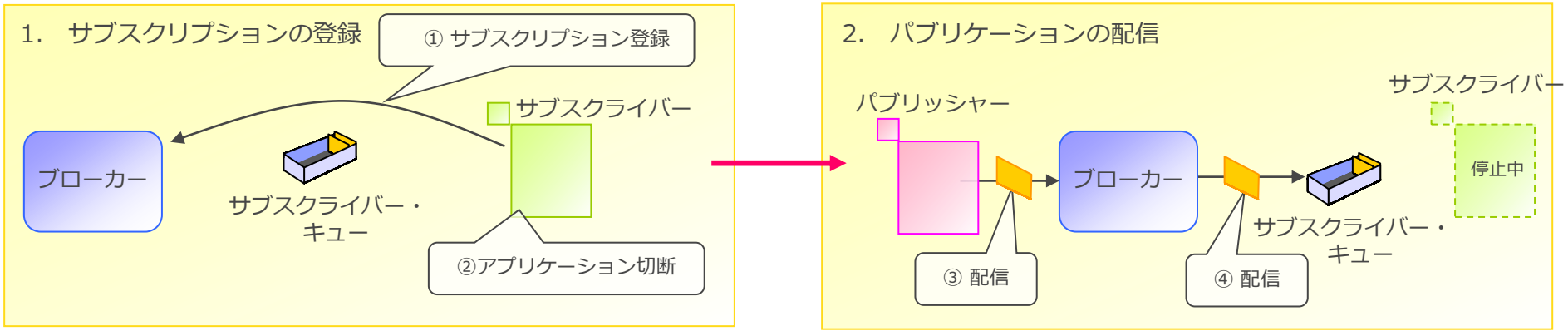
- パブリッシャーはトピックに対してパブリケーションを送信するアプリケーション
  - トピックに対してMQOPENを実行
  - MQMDやトピック・オブジェクトにパブリケーションの属性を指定
  - MQPMOで配信オプションを設定
  - MQPUTでパブリッシュ



Pub/Sub：サブスクリプション

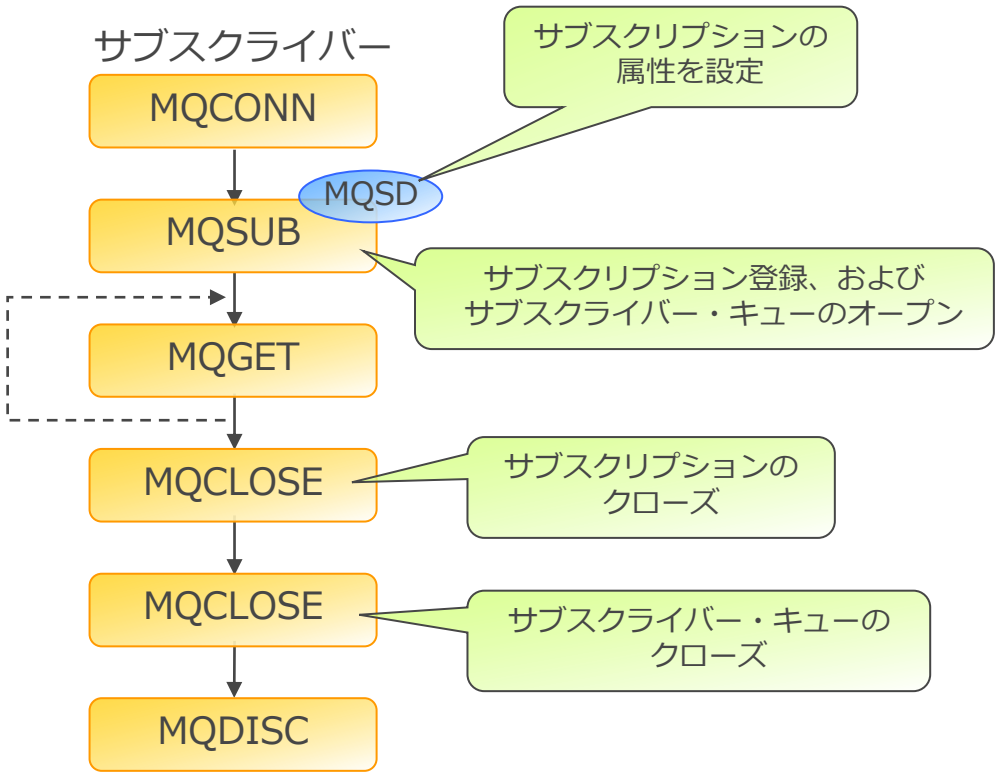
- 2つのサブスクリプション・タイプ
  - アプリケーション要件に応じたサブスクリプションをMQSUB時に選択可
- 継続サブスクリプション
  - アプリケーションの起動をまたがって、サブスクリプションを保存しておくことができる
    - アプリケーションが停止中でもパブリケーションをサブスクリプション・キューに保管しておくことが可能
    - 再開するために、ユニークなサブスクリプション名をもつ必要がある
    - 継続サブスクリプションを破棄する場合は、MQCLOSE時にMQCO_REMOVE_SUBを指定
- 非継続サブスクリプション
  - アプリケーションが終了すると、サブクスリプションは破棄される
    - デフォルトのサブスクリプション・タイプ

【継続サブスクリプションの例】



Pub/Sub：サブスクライバー

- サブスクライバーはトピックに対して送信されたパブリケーションを受信するアプリケーション
  - MQSUBを実行し、トピックなどのサブスクリプションを登録
    - MQSD (MQ Subscription Descriptor) によってサブスクリプションの属性を設定する
  - MQGETでパブリケーションの受信
- サブスクリプションを登録すると・・・
  - ブローカーによってサブスクリプション情報が保管される
  - 実態は、SUBオブジェクト
    - ブローカーが内部的に生成
    - ユーザーが明示的にSUBオブジェクトを定義し、サブスクリプション登録することも可能
- サブスクリプション属性の照会
  - MQSCコマンドでSUBオブジェクトの確認が可能



EXIT処理

- MQコンポーネントやMQI呼び出しの前後にEXITを組み込み、動作をカスタマイズすることが可能
  - ◆ ユーザーEXIT
    - MQの機能を拡張することが可能

EXIT名	説明
チャネルEXIT	独自のチャネル接続動作方法を実行可能 チャネル、キュー・マネージャーの属性にて設定
データ変換EXIT	指定されたメッセージ形式へ変換可能 メッセージ・データがキャラクター、バイナリー混在の場合に使用 サンプル・プログラム提供（amqsvfc0.cなど） MQMDの“Format”フィールドにEXIT名をセット
クラスター・ワークロードEXIT	MQクラスターにおけるワークロード管理において独自のワークロード管理ルーチンを実行することが可能 キュー・マネージャーのCLWLEXIT属性で指定 C言語、system/390アセンブラーのみ
パブリッシュEXIT	パブリッシュされたメッセージがサブスクライバーによって受信される前に、任意のロジックを呼び出すことが可能 キュー・マネージャー構成ファイルで設定

  - ◆ API EXIT
    - MQI呼び出しの直前または直後にEXITプログラムを挿入し独自の動作を組み込ませることが可能
      - mqs.ini、もしくはqm.iniに新規スタンザを追加し指定

EXIT名	説明
MQ_PUT_EXIT	MQPUT処理の出口機能
MQ_GET_EXIT	MQGET処理の出口機能

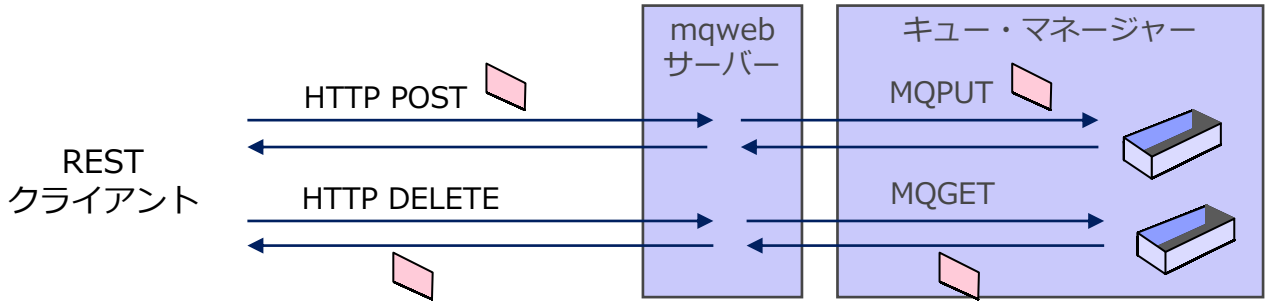
EXIT処理

◆ API EXIT（続き）

EXIT名	説明
MQ_OPEN_EXIT	MQOPEN処理の出口機能
MQ_CLOSE_EXIT	MQCLOSE処理の出口機能
MQ_CMIT_EXIT	コミット処理の出口機能
MQ_PUT1_EXIT	1つのメッセージだけを書き込むMQPUT1処理の 出口機能
MQ_CONNX_EXIT	接続処理の出口機能
MQ_CTL_EXIT	制御コールバック処理の出口機能
MQ_DISC_EXIT	切断処理の出口機能
MQ_INQ_EXIT	MQINQ処理の出口機能
MQ_SET_EXIT	MQSET処理の出口機能
MQ_STAT_EXIT	MQSTAT処理の出口機能
MQ_BACK_EXIT	バックアウト処理の出口機能
MQ_CALLBACK_EXIT	コールバック処理の出口機能
MQ_CB_EXIT	コールバック登録処理の出口機能
MQ_BEGIN_EXIT	MQBEGIN処理の出口機能
MQ_SUB_EXIT	サブスクリプション再登録処理の出口機能
MQ_SUBRQ_EXIT	サブスクリプション要求処理の出口機能
MQ_TERM_EXIT	接続終了処理の出口機能
MQ_INIT_EXIT	接続初期化処理の出口機能

Messaging REST API

- Messaging REST APIはmqwebサーバー上で稼動するアプリケーション
  - ◆ mqwebサーバーの実体はWebSphere Liberty Profile(WLP)
    - mqwebサーバーはRESTクライアントからのリクエストを受信し、対応する処理をキュー・マネージャーに対して行う



- HTTPプロトコルをMQIへマッピングすることが可能
  - ◆ RESTクライアントからMQシステムへの接続が可能

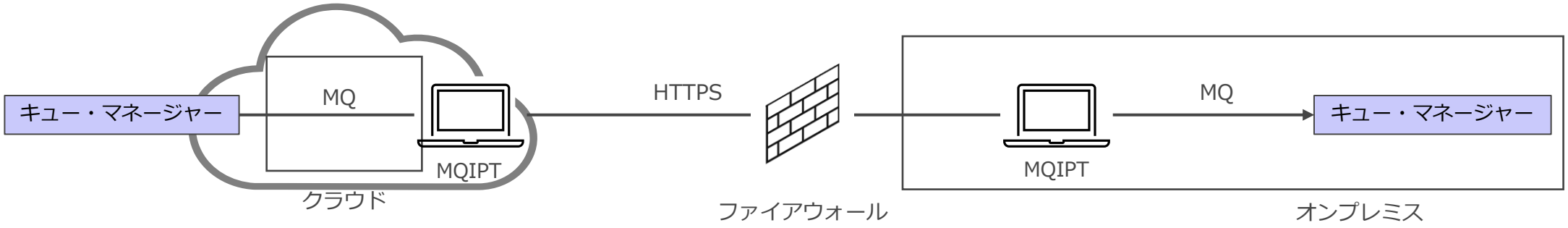
【HTTPプロトコル、リソースURL、MQIの対応表】

HTTPプロトコル	リソースURL	MQI
POST	/messaging/qmgr/{qmgrName}/queue/{queueName}/message	MQPUT
GET		MQGET with browse
DELETE		MQGET
PUT	–	–

＊ その他の実行可能な処理については以下のリンクを参照
[REST API resources]:https://www.ibm.com/docs/en/ibm-mq/9.3?topic=mrar-rest-api-resources

MQ Internet Pass-Thru(MQIPT)

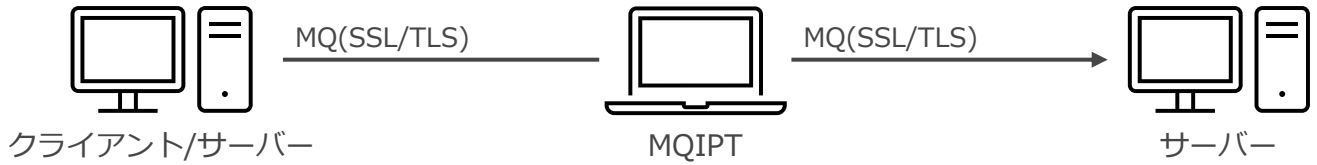
- ファイアウォールを隔てたMQ通信を単純かつ管理しやすくするための機能
  - ◆ V9.2から登場したMQのオプション・コンポーネント
    - AIX、Linux、Windows上でインストール可能
      - プラットフォームを問わず、MQ - MQIPT間の通信は可能
        例）Linux MQ - Linux MQIPT - z/OS MQ
    - mqiptコマンドでMQIPTを開始、mqiptAdminコマンドに-stopキーワードを指定してMQIPTを停止
      - MQIPTはプロセスとして起動
    - mqipt.confファイルに経路を定義し、MQIPTを構成
  - ◆ ハイブリッド/マルチクラウドのMQ ネットワークの相互接続が可能
  - ◆ MQトラフィックをHTTPSに変換しインターネット上のセキュアな相互通信が可能
  - ◆ ファイアウォールのフィルタリング・ルールの定義と管理が容易



MQ Internet Pass-Thru(MQIPT)

- 通信パスで使用可能なプロトコル
  - ◆ MQ - MQIPT間で使用できるプロトコル
    - MQ(FAP) もしくは MQ on SSL/TLS
  - ◆ MQIPT間で使用できるプロトコル
    - MQ(FAP) もしくは MQ on SSL/TLS
    - HTTP もしくは HTTPS
  - ◆ 詳細な組み合わせは以下のリンクを参照
    - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=thru-compatible-configurations

- MQIPTの基本的な3つの構成パターン
  - ◆ 単一のMQIPTを使用した構成
    - 1つのMQIPTを2つのキューマネージャー間、またはクライアントとキューマネージャー間の通信パスに配置
    - MQIPTでMQ接続を集約できる
      - 呼び出し側プロトコル：MQ(SSL/TLS)
        - 元のプロトコルのまま、もしくは、MQ(SSL/TLS)に変換後に宛先キューマネージャーに転送



MQ Internet Pass-Thru(MQIPT)

■ MQIPTの基本的な3つの構成パターン (続き)

◆ SSL/TLS接続構成

- 通信パスに2つのMQIPTを配置し、MQIPT間で暗号化されたメッセージを転送
  - 呼び出し側プロトコル：MQ
    - 1つ目のMQIPTがメッセージを暗号化して転送
    - 2つ目のMQIPTがメッセージをMQ(SSL/TLS)で宛先キューマネージャーに転送
  - 呼び出し側プロトコル：MQ on SSL/TLS
    - 1つ目のMQIPTはメッセージを中継
    - 2つ目のMQIPTがメッセージをMQ(SSL/TLS)で宛先キューマネージャーに転送

◆ HTTPトンネリング接続構成

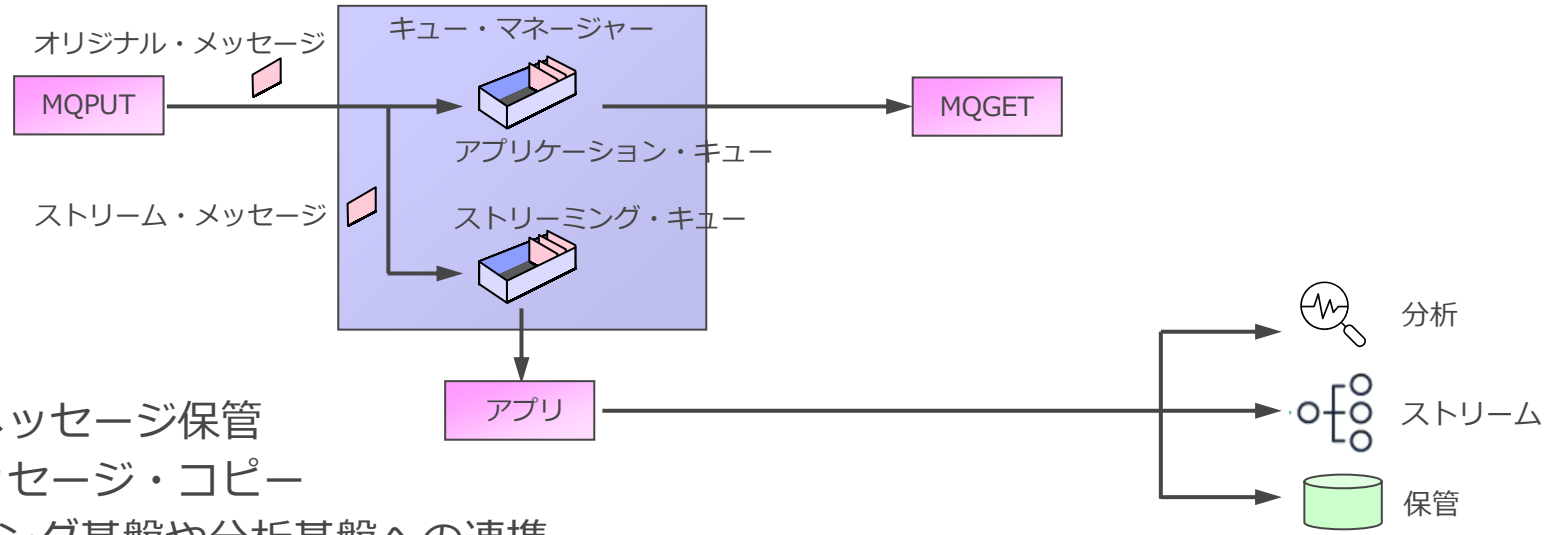
- 通信パスに配置された2つのMQIPT間でHTTP(S)を使用してメッセージを転送
  - 呼び出し側プロトコル：MQ(SSL/TLS)
    - 1つ目のMQIPTがメッセージをHTTP(S)データに変換し転送
    - 2つ目のMQIPTがHTTP(S)データからメッセージを抽出して、MQ(SSL/TLS)で宛先キューマネージャーに転送

Trusted Application(FASTPATH BINDING)

- Trusted Applicationは、FASTPATH BINDINGモードで接続するアプリケーション
- エージェント・プロセスを起動せずにキュー・マネージャーと接続することが可能
  - ◆ エージェント・プロセスはアプリケーションの一部（スレッド）として稼動する
  - ◆ MQアプリケーションがキューマネージャーと同一OS上で直接接続する接続形態(ローカル接続)
  - ◆ 最も高速な接続を提供
    - ● ただし、アプリケーションのエラーがキュー・マネージャーに影響を与える事もあるので十分注意して使用する
    - ● 制限事項の詳細は以下のリンクを参照
      - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=call-restrictions-trusted-applications
  - ◆ 使用方法
    - ● アプリケーションにて“MQCONNX”のオプションでエージェントを起動しない“MQCNO_FASTPATH_BINDING”オプションを指定(デフォルトはMQCNO_STANDARD_BINDING)
    - ● その他の接続オプション
      - MQCNO_STANDARD_BINDING : アプリケーションとMQエージェントがそれぞれ別のプロセスで実行
qm.iniで定義されているキュー・マネージャーのDefaultBindType属性の値に応じて、
SHAREDまたはISOLATEDのいずれかを使用
      - MQCNO_SHARED_BINDING : アプリケーションとMQエージェントが別プロセスで実行し、一部のリソースをシェアする
      - MQCNO_ISOLATED_BINDING : アプリケーションとMQエージェントが別プロセスで実行し、リソースもシェアしない
      - MQCNO_CLIENT_BINDING : クライアント接続のみ試行
      - MQCNO_LOCAL_BINDING : サーバー接続(STANDARD, SHARED, ISOLATEDのいずれか)のみ試行

ストリーミング・キュー

- キューに書き込まれたメッセージを、指定した別のキューにコピーする機能
  - ◆ 既存のアプリケーション設定を変更せずに、メッセージのコピーが可能
  - ◆ ローカル・キューとモデル・キューで利用可能
  - ◆ 2つのキュー属性を使用
    - STREAMQ属性：コピー先のキューを指定
      - 指定可能なキュータイプはローカル・キュー、別名キュー、リモート・キュー
    - STRMQOS属性：サービス品質(Best effort, Must duplicate)を設定
      - Best effort：オリジナル・メッセージの送達を優先し、ストリーム・メッセージは可能な限り送達する
      - Must duplicate：オリジナル・メッセージとストリーム・メッセージを同期的に送達する



  - ◆ 使用例
    - リカバリーのためのメッセージ保管
    - 開発・テスト用のメッセージ・コピー
    - Kafka等のストリーミング基盤や分析基盤への連携