

IBM MQ

MQI Test Program (mqpgf)

Ver 1.4.2.12
20 October, 2025

Pulsar Integration Inc.
e-mail : support@pulsarintegration.com

Program Version 1.4.2.12

本プログラムの 検証済み MQ バージョン / OS

- ・ Windows 10 64bit / IBM MQ 9.0 / 9.1 / 9.2.0 / 9.2.3
- ・ Windows 10 64bit / IBM MQ 9.1 / 9.2.1 / 9.2.4 Client
- ・ Linux RHEL Server release 7.4 (Maipo) / IBM MQ 9.0 / 9.2.3
- ・ CentOS Linux release 7.7.1908 64bit / WebSphere MQ 9.1 / 9.2.4
- ・ HP NonStop i J06.21.01 / IBM MQ 8.1, WebSphere MQ 5.3.1
- ・ HP NonStop X L23.08.00 / IBM MQ 8.1

コンパイル / 稼働実績のある MQ バージョン / OS

- ・ SunOS 5.10 sparc / WebSphere MQ 7.5
- ・ SunOS 5.10 sparc / IBM MQ 9.0
- ・ HP-UX 11iV2 (11.23) HP rp3410-2 (PA8900) / WebSphere MQ 7.0.1
- ・ HP-UX 11iV2 (11.23) HP rx1620-2 (IA-64, IPF) / WebSphere MQ 7.0.1
- ・ HP-UX 11iV3 (11.31) ia64 / IBM MQ 9.0
- ・ Linux ppc64 / WebSphere MQ 7.5
- ・ AIX 6.1 / WebSphere MQ 8.0
- ・ AIX 5.3 / WebSphere MQ 7.0.1
- ・ Linux RED Hat 5.5 x86 32bit / WebSphere MQ 7.5
- ・ Linux RED Hat 5.8 x86 64bit / WebSphere MQ 7.5
- ・ Windows 7 64bit / IBM MQ 9.0
- ・ HP NonStop i J06.14.01 / WebSphere MQ 5.3.1
- ・ HP NonStop i J06.20.00, J06.21.01 / IBM MQ 8.0, 8.1, WebSphere MQ 5.3.1
- ・ HP NonStop X L16.05.00 / IBM MQ 8.0, 8.1
- ・ HP NonStop X L20.05.00, L20.10.00 / IBM MQ 8.1

※本プログラムのWindows版は32bitでコンパイルされている為、64bit/32bit Windows OSの両方で動作可能です。Linux x86版のプログラムはバージョン1.4.2.6から32bit版と64bit版の両方が提供されています。Linux x86 64ビットOS上に32bitのランタイム・ライブラリがインストールされている場合は、本プログラムの32bit版も稼働できます。

本プログラムは上記の検証済みの環境以外の多くのOSレベルで実行できる可能性があります。

目次

1. 製品の概要.....	1-1
本プログラムについて	1-1
バージョンの命名規則	1-1
2. プログラムの実行環境.....	2-1
MQインストール環境	2-1
MQライブラリの参照	2-1
実行ユーザー	2-1
3. 使用方法の説明.....	3-1
USAGEの表示	3-1
Ex. 3.1 使用方法の表示.....	3-1
ライセンス情報、バージョン情報、指定可能なコンスタントの表示.....	3-5
Ex. 3.2 ライセンス情報、バージョン情報、指定可能なコンスタントの表示	3-5
クライアント・モードの使用.....	3-7
4. 基本的なテストの実施方法.....	4-1-1
4.1 共通パラメータ.....	4-1-1
4.2 コマンドラインで指定したメッセージをPUT	4-2-1
Ex. 4.2.1 メッセージ件数、PUT間隔、メッセージ長を指定してPUT	4-2-1
4.3 コマンドラインで指定するPUTメッセージをHexaDecimalで指定.....	4-3-1
Ex. 4.3.1 メッセージを16進表記で指定してPUT	4-3-1
4.4 GETしたメッセージを標準出力へ書き込み(可視文字のみ)	4-4-1
Ex. 4.4.1 不可視文字を含むメッセージ (バイナリデータ) のGET	4-4-1
4.5 ファイルデータをPUT.....	4-5-1
Ex. 4.5.1 ファイルのデータをのPUT回数、間隔を指定	4-5-1
4.6 GETしたメッセージをファイルへ出力	4-6-1
Ex. 4.6.1 GETしたメッセージをファイルに出力	4-6-1
Ex. 4.6.2 MQMDのバージョンを指定してGET	4-6-2
Ex. 4.6.3 伝送キュー上のメッセージをGET	4-6-3
Ex. 4.6.4 デッド・レター・キュー上のメッセージをGET	4-6-4
4.7 GETしたメッセージをキューへ出力(リキュー)	4-7-1

Ex. 4.7.1 入力キューの全てのメッセージを他のキューへ出力	4-7-1
4.8 PUTしたメッセージの応答を受信	4-8-1
Ex. 4.8.1 PUT後、その応答メッセージを受信	4-8-1
4.9 ディレクトリ内のファイルデータを全てPUT	4-9-1
Ex. 4.9.1 ディレクトリ内のファイルをPUT回数、間隔を指定	4-9-1
4.10 キュー内のメッセージをGETしディレクトリに出力	4-10-1
Ex. 4.10.1 キュー上の全てのメッセージを指定間隔でGETし、ディレクトリに出力 ...	4-10-1
4.11 キュー上のメッセージのブラウズ/ダンプ (通常)	4-11-1
Ex. 4.11.1 キュー上の全てのメッセージをブラウズ	4-11-1
Ex. 4.11.2 GET時にMQMDEヘッダが生成される例	4-11-2
4.12 キュー上のメッセージのブラウズ/ダンプ (詳細)	4-12-1
表 4.12.1 表示モード指定が有効な項目	4-12-1
Ex. 4.12.1 伝送キュー内のメッセージの詳細ダンプ	4-12-2
Ex. 4.12.2 デッドレターキュー内のメッセージのダンプ	4-12-3
Ex. 4.12.3 RFH2ヘッダを含むJMSメッセージのダンプ	4-12-4
Ex. 4.12.4 PCFフォーマットのダンプ	4-12-5
4.13 GETしたメッセージを標準出力へ書き込み (rawモード)	4-13-1
Ex. 4.13.1 バイナリデータをリダイレクトしてファイルへ保存	4-13-1
4.14 GETしたメッセージを標準出力へ書き込み (16進表記)	4-14-1
Ex. 4.14.1 バイナリデータを16進表記で標準出力へ書き込み	4-14-1
4.15 PCFフォーマット・メッセージをPUT	4-15-1
Ex. 4.15.1 ユーザー定義のフォーマットのPCFメッセージを作成しPUT	4-15-4
Ex. 4.15.2 コマンド・サーバーに start channel コマンドを送信	4-15-8
4.16 MQSET呼び出し	4-16-1
表 4.16.1 キューに関する MQSET 属性セレクター	4-16-1
Ex. 4.16.1 MQSETに単独のパラメータを指定する例	4-16-2
Ex. 4.16.2 MQSETに複数のパラメータを指定する例	4-16-2
4.17 MQINQ呼び出し	4-17-1
表 4.17.1 キューに関する MQINQ 属性セレクター	4-17-1
表 4.17.2 名前リストに関する MQINQ 属性セレクター	4-17-4
表 4.17.3 プロセス定義に関する MQINQ 属性セレクター	4-17-5
表 4.17.4 キュー・マネージャーに関する MQINQ 属性セレクター	4-17-5
Ex. 4.17.1 ローカルキューの属性の照会	4-17-12
Ex. 4.17.2 リモートキューの属性の照会	4-17-13
Ex. 4.17.3 エイリアスキューの属性の照会	4-17-13
Ex. 4.17.4 ネームリストの属性の照会	4-17-13

Ex. 4.17.5 プロセスの属性の照会	4-17-13
Ex. 4.17.6 キューマネージャーの属性の照会	4-17-14
4.18 メッセージ・プロパティの指定	4-18-1
表 4.18.1 プロパティ値のデータ型	4-18-1
Ex. 4.18.1 メッセージ・プロパティの指定例（数値型に最大値を指定）	4-18-1
Ex. 4.18.2 メッセージ・プロパティの数値型（数値型に最小値を指定）	4-18-1
4.19 配布リストの使用	4-19-1
表 4.19.1 PUTメッセージ・レコードのフィールド	4-19-2
Ex. 4.19.1 オブジェクト・レコードに複数のローカル・キューを指定してPUT	4-19-3
Ex. 4.19.2 クラスタ・キューにオブジェクト・キューマネージャーを指定してPUT ...	4-19-3
Ex. 4.19.3 メッセージ・レコードを指定（オブジェクト・レコードの先頭のキューのみメッ ジ・レコードを指定する場合）	4-19-3
Ex. 4.19.4 クライアント接続で配布リストを使用	4-19-4
Ex. 4.19.5 メッセージ・レコード内の省略	4-19-5
Ex. 4.19.6 メッセージ・レコードに特定のフィールドのみ指定	4-19-6
Ex. 4.19.7 一部のクラスタ・キューのPUTに失敗した場合	4-19-7
Ex. 4.19.8 MQRC_MULTIPLE_REASONSが返却される例	4-19-9
4.20 キューマネージャーによるセグメント化	4-20-1
Ex. 4.20.1 キューマネージャーによるセグメント化	4-20-1
4.21 アプリケーションによるセグメント化	4-21-1
Ex. 4.21.1 アプリケーションによるセグメント化	4-21-1
4.22 キューマネージャーによる論理メッセージの再組立て	4-22-1
Ex. 4.22.1 キューマネージャーによる論理メッセージの再組立て	4-22-1
4.23 アプリケーションによる論理メッセージの再組立て	4-23-1
Ex. 4.23.1 アプリケーションによる論理メッセージの再組立て	4-23-1
4.24 論理メッセージのグループ化	4-24-1
Ex. 4.24.1 論理メッセージのグループ化	4-24-1
4.25 論理メッセージのグループ化と論理メッセージのセグメント分割	4-25-1
Ex. 4.25.1 論理メッセージのグループ化と論理メッセージのセグメント分割	4-25-1
4.26 グループ・メッセージの読み込み	4-26-1
Ex. 4.26.1 グループ・メッセージの読み込み	4-26-1
5. 全パラメータ・リファレンス	5-1-1
5.1 基本パラメータ	5-1-1
キューマネージャー名 (-qm)	5-1-1

Ex. 5.1.1 複数のキューマネージャーに接続する場合	5-1-1
キュー名 (-q)	5-1-2
入力メッセージ (-m)	5-1-3
入力メッセージ (16進表記) (-mx)	5-1-3
入力メッセージ・ファイル名 (-f)	5-1-3
出力メッセージ・ファイル名 (-o)	5-1-4
出力キュー名 (-oq)	5-1-4
入力キュー名 (-iq)	5-1-4
入力ディレクトリ名 (-d)	5-1-5
出力ディレクトリ名 (-g)	5-1-5
キュー上の全てのメッセージをGET (-r)	5-1-5
強制バックアウト (-b)	5-1-5
Ex. 5.1.2 PUT時に指定する場合	5-1-6
Ex. 5.1.3 GET時に指定する場合	5-1-6
PUTするメッセージのサイズ指定 (-l)	5-1-7
Ex. 5.1.4 -mオプションと共に使用した例	5-1-7
Ex. 5.1.5 -mxオプションと共に使用した例	5-1-7
Ex. 5.1.6 -fオプションと共に使用した例	5-1-8
Ex. 5.1.7 -dオプションと共に使用した例	5-1-9
PUT/GETするメッセージの数の指定 (-n)	5-1-10
Ex. 5.1.8 -dオプションと共に使用した例	5-1-10
PUT/GETするメッセージの間隔の指定 (-i)	5-1-11
GETするメッセージの最大サイズの指定 (-sz)	5-1-11
Ex. 5.1.9 受信バッファをデフォルトより大きく指定する例	5-1-11
Ex. 5.1.10 受信バッファをデフォルトより小さく指定する例	5-1-12
標準出力への書き込みサイズの指定 (-ds)	5-1-12
Ex. 5.1.11 PUT/GET時に大きなサイズのメッセージの表示サイズを調節	5-1-12
Ex. 5.1.12 受信したメッセージの全てを表示	5-1-13
キュー上のメッセージのブラウズ/ダンプ (通常) (-br)	5-1-13
キュー上のメッセージのブラウズ/ダンプ (詳細) (-brv)	5-1-14
GETしたメッセージをダンプ表示 (通常) (-dp)	5-1-14
GETしたメッセージをダンプ表示 (詳細) (-dpv)	5-1-14
rawモード出力 (-raw)	5-1-14
16進表記で出力 (-hex)	5-1-15
指定MQI呼び出し実行前で停止 (-s)	5-1-15
Ex. 5.1.13 -s MQGET を -rと共に指定した場合	5-1-16
プロセス名の指定 (-p)	5-1-16
ネームリスト名の指定 (-nl)	5-1-16
PCFフォーマット・メッセージをPUT (-pcf)	5-1-17
以降のパラメータを2次側に切り替える (-ss)	5-1-17

	以降のパラメータを1次側に戻す (-sp)	5-1-17
	送信MsgIdをCorrelIdに持つメッセージをGET (-mc)	5-1-17
	受信MQMDを送信メッセージに引き継ぐ (-im)	5-1-17
	セグメント・サイズ (-as)	5-1-18
	論理メッセージのデリミタ (-dl)	5-1-18
	起動スレッド数 (-nt)	5-1-18
Ex.	5.1.14 起動スレッド数の指定	5-1-18
	起動スレッド数(スレッド内でMQCONN/MQDISC呼び出し) (-ni)	5-1-20
Ex.	5.1.15 起動スレッド数の指定(スレッド内でMQCONN/MQDISC呼び出し)	5-1-20
	APIトレース (-tr)	5-1-23
	処理開始の同期用ファイル名 (-sf)	5-1-24
	接続繰り返し回数 (-c)	5-1-24
	MQDISC呼び出しの省略 (-sd)	5-1-24
	接続繰り返し時の待ち時間 (-wp)	5-1-24
	MQI失敗後も処理を継続 (-ca)	5-1-24
	送信メッセージにカウンターを付加 (-ac)	5-1-25
	接続リトライの回数 (-cr)	5-1-25
	Thread内MQI呼び出し後CPUの使用権を放棄 (-y)	5-1-25
	ファイルまたはディレクトリのオーバーライト確認省略 (-nm)	5-1-25
	ディレクトリに出力するファイル名の拡張子を指定 (-ext)	5-1-25
	RFHヘッダを強制的に削除 (-rh)	5-1-25
	標準出力/エラーをフラッシュ (-ff)	5-1-26
	特定のMQIまたはTNS関数の呼び出しをスキップする (-sk)	5-1-26
表	5.1.1 スキップ可能(-skに指定可能)な関数	5-1-26
	最大未コミットメッセージ数の指定 (-u)	5-1-27
5.2	プラットフォーム固有のオプション	5-2-1
	グローバル作業単位の使用 (NSK) (-gt)	5-2-1
	グローバル作業単位の使用 (MQI呼び出し毎) (NSK) (-gti)	5-2-1
5.3	特定のMQIの指定	5-3-1
	MQSET (-set)	5-3-1
	MQINQ (-inq)	5-3-1
	MQSETMP (-smp)	5-3-1
5.4	MQCDのフィールド	5-4-1
	ConnectionNameの指定 (-x) (MQCHAR264) (-)	5-4-1
	ChannelNameの指定 (-ch) (MQCHAR20) (-)	5-4-1
	LocalAddressの指定 (-la) (MQCHAR48) (-)	5-4-1
	CertificateLabel (-cl) (MQCHAR64) (-)	5-4-1
	SSLCipherSpec (-cs) (MQCHAR32) (-)	5-4-1
	SSLPeerName (-er) (-) (-)	5-4-2

5.5 MQMDのフィールド	5-5-1
Ex. 5.5.1 MsgIdとCorrelIdの指定の例	5-5-1
Ex. 5.5.2 UserIdentifier、AccountingToken、ApplIdentityDataの指定	5-5-1
Ex. 5.5.3 PutApplName、PutDate、PutTime、ApplOriginDataの指定	5-5-2
Expiry (-ex) (MQLONG) (MQEI_UNLIMITED)	5-5-3
Encoding (-ec) (MQLONG) (MQENC_NATIVE)	5-5-3
Ex. 5.5.4 MQRFH2ヘッダのエンコーディングの指定例	5-5-3
CodedCharSetId (-cc) (MQLONG) (MQCCSI_Q_MGR)	5-5-4
Priority (-pr) (MQLONG) (MQPRI_PRIORITY_AS_Q_DEF)	5-5-4
MsgId (-mi) (MQBYTE24) (MQMI_NONE_ARRAY)	5-5-4
CorrelId (-ci) (MQBYTE24) (MQCI_NONE_ARRAY)	5-5-5
ReplyToQ (-rq) (MQCHAR48) ("")	5-5-5
ReplyToQMgr (-rm) (MQCHAR48) ("")	5-5-5
UserIdentifier (-ui) (MQCHAR12) ("")	5-5-5
AccountingToken (-at) (MQBYTE32) (MQACT_NONE_ARRAY)	5-5-5
ApplIdentityData (-ap) (MQCHAR32) ("")	5-5-5
PutApplName (-pn) (MQCHAR28) ("")	5-5-6
PutDate (-pd) (MQCHAR8) ("")	5-5-6
PutTime (-pt) (MQCHAR8) ("")	5-5-6
ApplOriginData (-ao) (MQCHAR4) ("")	5-5-6
5.6 MQMD Version 2のフィールド	5-6-1
Ex. 5.6.1 MQMD Version 2の指定例	5-6-1
GroupId (-gi) (MQBYTE24) ("")	5-6-1
MsgSeqNumber (-ms) (MQLONG) (1)	5-6-2
Offset (-of) (MQLONG) (0)	5-6-2
OriginalLength (-ol) (MQLONG) (MQOL_UNDEFINED)	5-6-2
5.7 MQRFH2のフィールド	5-7-1
Ex. 5.7.1 MQRFH2フィールドの指定例	5-7-2
Encoding (-re) (MQLONG) (MQENC_NATIVE)	5-7-2
表 5.7.1 指定可能なMQENC_*	5-7-2
Ex. 5.7.2 Encodingフィールドの指定例	5-7-4
CodedCharSetId (-rc) (MQLONG) (MQCCSI_INHERIT)	5-7-4
表 5.7.2 指定可能なMQCCSI_*	5-7-4
Format (-rf) (MQCHAR8) (MQFMT_NONE_ARRAY)	5-7-5
表 5.7.3 指定可能なMQFMT_*	5-7-5
Flags (-fg) (MQLONG) (MQRFH_NONE)	5-7-6
NameValueCCSID (-nc) (MQLONG) (1208)	5-7-6

NameValueData (-nd) (MQCHARn) (-)	5-7-7
5.8 MQCSPのフィールド	5-8-1
CSPUserId (-cu) (-) (-)	5-8-1
CSPPassword (-cp) (-) (-)	5-8-1
Ex. 5.8.1 MQCSPにユーザー/パスワードを指定して接続	5-8-1
5.9 MQODのフィールド	5-9-1
ObjectQMgrName (-om) (MQCHAR48) ("")	5-9-1
Ex. 5.9.1 オブジェクト・キューマネージャの指定例	5-9-1
AlternateUserId (-au) (MQCHAR12) ("")	5-9-2
Ex. 5.9.2 代替ユーザーIDを指定してPUT/GET	5-9-2
ObjectRec (MQOR) (-or) (-) (-)	5-9-2
DynamicQName (-dq) (MQCHAR48) ("AMQ.*")	5-9-3
Ex. 5.9.3 動的一時キューを作成し、メッセージを書き込む	5-9-3
Ex. 5.9.4 動的一時キューを作成し、レポートメッセージを待つ	5-9-5
5.10 MQPMOのフィールド	5-10-1
PutMsgRec (MQPMR) (-mr) (-) (-)	5-10-1
5.11 MQGMOのフィールド	5-11-1
WaitInterval (-wi) (MQLONG) (-)	5-11-1
Ex. 5.11.1 GET時にメッセージが到着するまで指定時間待機	5-11-1
MsgToken (-mt) (MQBYTE16) (MQMTOK_NONE_ARRAY)	5-11-1
5.12 MQIMPOのフィールド	5-12-1
RequestedEncoding (-pe) (MQLONG) (MQENC_NATIVE)	5-12-1
RequestedCCSID (-pc) (MQLONG) (MQCCSI_APPL)	5-12-1
5.13 MQCBのフィールド	5-13-1
Operation (-op) (MQLONG) (-)	5-13-1
表 5.13.1 指定可能なMQOP_*	5-13-1
5.14 MQCBDのフィールド	5-14-1
CallbackFunction (-cf) (MQPTR) (-)	5-14-1

5.15 MQSCOのフィールド	5-15-1
KeyRepository (-kr) (MQCHAR256) (-)	5-15-1
5.16 MQAIRのフィールド	5-16-1
OCSPResponderURL (-ru) (MQCHAR256) (-)	5-16-1
6. コンスタントの指定	6-1-1
6.1 MQMDパラメータ	6-1-1
MQMD_*	6-1-1
表 6.1.1 指定可能なMQMD_*	6-1-1
Ex. 6.1.1 MQMDのバージョンをGET時に指定	6-1-1
MQRO_*	6-1-2
表 6.1.2 指定可能なMQRO_*	6-1-2
Ex. 6.1.2 レポートオプションの指定例	6-1-4
MQMT_*	6-1-4
表 6.1.3 指定可能なMQMT_*	6-1-4
Ex. 6.1.3 メッセージタイプの指定例	6-1-5
MQEI_*	6-1-5
表 6.1.4 指定可能なMQEI_*	6-1-5
MQFB_*	6-1-6
表 6.1.5 指定可能なMQFB_*	6-1-6
Ex. 6.1.4 フィードバックの指定例	6-1-8
MQENC_*	6-1-8
MQCCSI_*	6-1-9
MQFMT_*	6-1-9
表 6.1.6 指定可能なMQFMT_*	6-1-9
Ex. 6.1.5 MQMDのFormatとMQRFH2のFormatの指定例	6-1-10
MQPRI_*	6-1-10
表 6.1.7 指定可能なMQPRI_*	6-1-11
MQPER_*	6-1-11
表 6.1.8 指定可能なMQPER_*	6-1-11
Ex. 6.1.6 パーシステント属性を指定例	6-1-11
MQMI_*	6-1-12

表 6.1.9 指定可能なMQMI_*.....	6-1-12
MQCI_*.....	6-1-12
表 6.1.10 指定可能なMQCI_*.....	6-1-12
MQACT_*.....	6-1-13
表 6.1.11 指定可能なMQACT_*.....	6-1-13
MQACTT_*.....	6-1-13
表 6.1.12 指定可能なMQACTT_*.....	6-1-13
Ex. 6.1.7 MQACTT_*と-at の指定例.....	6-1-14
MQAT_*.....	6-1-14
表 6.1.13 指定可能なMQAT_*.....	6-1-14
Ex. 6.1.8 アプリケーションタイプの指定例.....	6-1-16
MQMF_*.....	6-1-17
表 6.1.14 指定可能なMQMF_*.....	6-1-17
Ex. 6.1.9 メッセージフラグの指定例.....	6-1-17
6.2 MQRFH2パラメータ.....	6-2-1
MQRFH_*.....	6-2-1
表 6.2.1 指定可能なMQRFH_*.....	6-2-1
6.3 MQCNOパラメータ.....	6-3-1
MQCNO_* (MQCNO*VERSION*、MQCNO_STRUC_ID 以外)	6-3-1
表 6.3.1 指定可能なMQCNO_* (MQCNO*VERSION*、MQCNO_STRUC_ID 以外)	6-3-1
Ex. 6.3.1 SHARED_BINDINGで接続する例.....	6-3-2
Ex. 6.3.2 FASTPATH_BINDINGで接続する例.....	6-3-2
Ex. 6.3.3 ISOLATED_BINDINGで接続する例.....	6-3-3
6.4 MQOPENオプション.....	6-4-1
MQOO_*.....	6-4-1
表 6.4.1 指定可能なMQOO_*.....	6-4-1
6.5 MQODパラメータ.....	6-5-1
MQOT_*.....	6-5-1
表 6.5.1 指定可能なMQOT_*.....	6-5-1
MQOD_* (MQOD*VERSION*)	6-5-1
表 6.5.2 指定可能なMQOD_* (MQOD*VERSION*)	6-5-1

6.6 MQPMOパラメータ	6-6-1
MQPMO_* (MQPMO*VERSION*)	6-6-1
表 6.6.1 指定可能なMQPMO*VERSION*	6-6-1
MQPMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)	6-6-1
表 6.6.2 指定可能なMQPMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)	6-6-1
MQPMRF_*	6-6-2
表 6.6.3 指定可能なMQPMRF_*	6-6-2
6.7 MQGMOパラメータ	6-7-1
MQGMO_* (MQGMO*VERSION*)	6-7-1
表 6.7.1 指定可能なMQGMO*VERSION*	6-7-1
MQGMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)	6-7-1
表 6.7.2 指定可能なMQGMO_* (MQGMO*VERSION*、MQGMO_STRUC_ID、MQGMO_LENGTH_* 以外)	6-7-1
Ex. 6.7.1 MQGMO_CONVERTを指定してユーザーデータ部をコード変換する例	6-7-3
Ex. 6.7.2 MQGMO_CONVERTを指定してRFH2とユーザーデータを同時にコード変換する例	6-7-4
Ex. 6.7.3 MQGMO_CONVERTを指定してPCFメッセージをコード変換する例	6-7-5
MQWI_*	6-7-7
表 6.7.3 指定可能なMQWI_*	6-7-7
MQMO_*	6-7-8
表 6.7.4 指定可能なMQMO_*	6-7-8
Ex. 6.7.4 MQMO_MATCH_MSG_IDを指定して特定のメッセージのみをGET	6-7-8
6.8 MQCLOSEオプション	6-8-1
MQCO_*	6-8-1
表 6.8.1 指定可能なMQCO_*	6-8-1
Ex. 6.8.1 永続動的キューをMQCLOSE()時に削除する例	6-8-1
6.9 MQSETMPオプション	6-9-1
MQPD_*	6-9-1
表 6.9.1 指定可能なMQPD_*	6-9-1
Ex. 6.9.1 入力の識別、起点、ユーザー・コンテキストを引き継いでリキューする例 ..	6-9-1
6.10 MQINQMPオプション	6-10-1
MQIMPO_*	6-10-1
表 6.10.1 指定可能なMQIMPO_*	6-10-1
Ex. 6.10.1 MQIMPO_CONVERT_VALUEを指定してプロパティのデータ変換を行う例	6-10-1

6.11 MQCRTMHオプション	6-11-1
MQCMHO_*	6-11-1
表 6.11.1 指定可能なMQCMHO_*	6-11-1
Ex. 6.11.1 プロパティ名をMQSETMP()で検証させる例	6-11-1
6.12 MQCBDパラメータ	6-12-1
MQCBT_*	6-12-1
表 6.12.1 指定可能なMQCMHO_*	6-12-1
6.13 MQAIRパラメータ	6-13-1
MQAIR_* (MQAIR*VERSION*)	6-13-1
表 6.13.1 指定可能なMQGMO*VERSION*	6-13-1
MQAIT_*	
表 6.13.2 指定可能なMQAIT_*	6-13-1
おわりに	7-1

1. 製品の概要

本プログラムについて

本プログラムはWebSphere MQ／IBM MQおよびその提供するAPIであるMQIの機能／使用方法を検証／確認する目的で作成されています。（MQIはC言語用のライブラリを使用しています。）細かい機能検証が可能です、一つのオペレーションを実行する為に多くのオプションやコンスタントを指定することが必要になる場合があります。本プログラムはプロジェクトの設計工程からシステムの運用段階まで、どのような工程でも使用することが可能です。ただし、MQIの機能についての詳細を理解することが必要です。

本書では、IBM MQ自体の詳細の解説は行いません。必要に応じて製品のマニュアルを参照してください。

全てのバージョンの製品のマニュアルが下記URLから参照できます。

IBM MQ (formerly IBM WebSphere® MQ)

https://www.ibm.com/support/knowledgecenter/SSFKSJ/com.ibm.mq.helphome.doc/product_welcome_wmq.htm

mqpgfコマンドの実行結果の確認には、MQ製品提供のプログラム以外に、MQAIプログラム(mqpcf) コマンドも使用しています。MQAIプログラム(mqpcf) コマンドの詳細については、資料「MQAI Program (mqpcf)」を参照してください。

バージョンの命名規則

本製品ではIBM MQと類似のバージョンの命名方式を使用しています。

バージョン、リリース、保守、フィックス（VRMF）レベル・コードの 4 桁で構成されています。

V: Version

R: Revision

M: Modification

F: Fix

mqpgf/mqpcfのバージョンにはIBM MQ製品自体のバージョンとの対応はありません。

- ・各レベルの説明

各レベルの意味は下記の通りです。

Version : 大きな機能の追加／変更があり、ソース・コードの互換ありません。但し、オペレーションの互換は極力維持されます。ユーザーズ・ガイドは別に作成されます。

Revision : ソース・コードのほとんどは維持されていますが、大きな機能の追加があります。ユーザーズ・ガイドは別に作成されます。

Modification : ソース・コードのほとんどは維持されていますが、細かい新機能の為のコードが追加されています。ユーザーズ・ガイドの新機能部分の説明にバージョン情報も付記されます。

Fix : ソース・コードに一つ以上の製品障害に対しての修正が適用されています。

・バージョン・アップの方法

それぞれのレベルが上がる場合、それよりも下位で行われた機能追加、修正も同時に適応されます。例えば、Modificationレベルが上がる場合、それ以前の全ての修正(Fix)も適用されています。

mqpgf/mqpcfはそれぞれ単一のモジュールですので、修正の適用はモジュール自体の置き換えになります。

・バージョン・アップのタイミング

Revision、Modificationレベルのバージョン・アップは、ユーザーからの要望に基づいて行われる場合以外にも、不定期に実施されます。

基本的に特定のユーザー専用のバージョンは作成しません。汎用的な仕様での機能追加を検討します。

・修正版の作成

ご要望により、特定のV.R.Mに対して修正(Fix)を行うことは可能ですが、特定の修正のみを適用することはできません。それ以前の全ての修正が適用されます。例えば、障害が見つかったバージョンが1.4.0.1で、その時点の最新のFixレベルが1.4.0.15の場合、その最新のFixレベルのソース・コードに修正が適用され、1.4.0.16がリリースされます。

但し、修正の適用の要望があったV.R.Mが著しく前のレベルの場合は、修正の適用が困難な場合があります。その場合はその修正が適用された最新のVersionのご使用をお願いします。

2. プログラムの実行環境

mqpgf/mqpgfc および mqpcf/mqpcfを使用する前提として、ご利用のマシンにWebSphere MQ7.0.1以上（HP NonStopの場合は5.3.1以上）のMQサーバーまたはクライアントがインストールされており、実際にWebSphere MQ/IBM MQを操作できる環境であることが必要です。

mqpgf(c)/mqpcf(c)自体は特別なインストール作業はありません。ご利用のプラットフォームにあったモジュールをダウンロードし、そのモジュールに適切なパーミッションを設定し、PATH環境変数でコマンドを参照可能にするだけで使用可能となります。ただし、ご利用の環境によっては下記に示す作業が必要になる場合があります。

※mqpgf、mqpcfはバインドモード用、mqpgfc、mqpcfはクライアントモード用です。

MQインストール環境

MQ7.1 以上を使用している場合、使用中の環境によってはその使用するMQのインストールの環境を読み込むことが必要になります。もし、ログインシェルなどの起動環境でMQの実行環境が読み込まれていない場合は下記を実行して使用するMQ環境をセットアップしてください。

```
$ . <MQ Install Directory>/bin/setmqenv -s
```

MQライブラリの参照

UNIX環境で、プログラム実行時に、MQのライブラリを参照できないというメッセージ（下記はSolarisの例）が表示される場合は、LD_LIBRARY_PATH(AIXの場合はLIBPATH)を設定し、exportしてください。

```
$ mqpgf
ld.so.1: mqpgf: fatal: libmqm.so: open failed: No such file or directory Killed

$ export LD_LIBRARY_PATH=<MQ Install Directory>/lib64:$ LD_LIBRARY_PATH
H
または
$ export LIBPATH=<MQ Install Directory>/lib64:$LIBPATH
```

実行ユーザー

プログラムの実行には実行ユーザーにキューマネージャーに設定される適切なアクセ

ス権限が必要です。権限の詳細が不明な場合は、mqmグループ(MQ管理者)のメンバーであるユーザーを使用するか、使用しているユーザーをmqmグループに含めてください。

3. 使用方法の説明

USAGEの表示

mqpgfを引数なしで実行すると、使用方法、指定できるパラメータが表示されます。

Ex. 3.1 使用方法の表示

```
-----
$ mqpgf
USAGE:
-qm : queue manager name(e.g. -qm qm1,qm2,...)
-q  : queue name
-m  : input message
-mx : input message(hexadecimal notation e.g. 09af..)
-f  : input file name
-o  : output file name
-oq : output queue name(for queue to queue)('*: ReplyToQ, '**': + ReplyToQMGr)
-iq : input queue name(for send and receive)
-d  : input directory name
-g  : output directory name
-r  : get repeatedly
-b  : force backout
-l  : message length for writing
-n  : message count for writing or reading
-i  : interval(ms) for writing or reading
-sz : max message size for reading(byte)(default 12KByte)
-ds : max message display size(byte)(default 128Byte) (all: entire message)
-br : browse message
-brv: browse message(verbose)
-dp : dump message
-dpv: dump message(verbose)
-raw: raw mode output
-hex: dump hexadecimal
-s  : stop before MQI call (e.g. -s MQCMIT)
-p  : process name
-nl : namelist
-pcf: pcf format file name
-ss : switch to parameters for secondary
-sp : switch to parameters for primary
-mc : get the message with the same CorrelId as MsgId sent
-im : Inherit MQMD
```

-as : segmentation size
 -dl : delimiter for logical messages
 -nt : number of threads
 -ni : number of threads that call MQCONN/MQDISC internally
 -tr : enable api trace
 -sf : the file for synchronization start
 -c : connection loop count
 -sd : skip MQDISC
 -wp : wait time to next processing
 -ca : continue processing after MQCONN(X) fails
 -ac : append the counter to message automatically
 -cr : The number of connection retry
 -y : Invoke yield function after every MQI call
 -nm : Non-Interactive Mode
 -ext : File Extension(use with -g)
 -rh : Forcibly Remove RFH
 -ff : Invoke fflush() each time
 -sk : Skip invoking function
 -u : Maximum number of uncommitted messages

Platform-specific options :

-gt : Using Global UOW for NSK
 -gti : Using Global UOW for NSK(TMFAPI: per PUT/GET)

MQI functions :

-set : MQSET (e.g. MQIA_INHIBIT_GET:MQQA_GET_ALLOWED,...:..)
 -inq : MQINQ (e.g. MQCA_ALTERATION_DATE,MQIA_CLWL_Q_PRIORITY,...)
 -smp : MQSETMP (e.g. MQTYPE_STRING:property name:value,...:..)

MQCD fields (use MQCONNX) :

-x : ConnectionName (e.g. -x "localhost(1414)")
 -ch : ChannelName -la : LocalAddress
 -cl : CertificateLabel -cs : SSLCipherSpec
 -er : SSLPeerName

MQMD fields :

-ex : Expiry(par 100ms) -ec : Encoding
 -cc : CodedCharSetId -pr : Priority
 -mi : MsgId -ci : CorrelId
 -rq : ReplyToQ -rm : ReplyToQMgr
 -ui : UserIdentifier -at : AccountingToken
 -ap : ApplIdentityData -pn : PutApplName

-pd : PutDate -pt : PutTime
 -ao : ApplOriginData
 MQMD Version 2 fields :
 -gi : GroupId -ms : MsgSeqNumber
 -of : Offset -ol : OriginalLength

MQRFH2 fields:
 -re : Encoding(<encoding> or MQENC_*)
 -rc : CodedCharSetId(<ccsid> or MQCCSI_*)
 -rf : Format (e.g. -rf MQFMT_STRING)
 -fg : Flags -nc : NameValueCCSID(<ccsid> or MQCCSI_*)
 -nd : NameValueData (e.g. -nd "data1,data2,data3")

MQCSP fields:
 -cu : CSPUserId -cp : CSPPassword

MQSCO fields:
 -kr : KeyRepository

MQAIR fields:
 -ru : OCSPResponderURL (e.g. -ru "url1,url2,url3")

MQOD fields:
 -om : ObjectQMgrName -au : AlternateUserId
 -or : ObjectRec(MQOR) (e.g. ObjectName:ObjectQMgrName,...:...,...)
 -dq : DynamicQName (for receive queue, e.g. "DQ*", "*" or 33bytes full name)

MQPMO fields:
 -mr : PutMsgRec (MQPMR) (e.g. <MsgId>:<CorrelId>:<GroupId>:MQFB_xx:<AccountingToken>,...)

MQGMO fields:
 -wi : WaitInterval(ms) -mt : MsgToken

MQIMPO fields:
 -pe : RequestedEncoding(<encoding> or MQENC_*)
 -pc : RequestedCCSID(<ccsid> or MQCCSI_*)

MQCB fields :
 -op : Operation (e.g. -op MQOP_REGISTER -op MQOP_SUSPEND ...)

MQCBD fields:

-cf : CallbackFunction (e.g. EventHandler)

Constants:

MQMD	: MQMD_*, MQRO_*, MQMT_*, MQEI_*, MQFB_*, MQENC_*, MQCCSI_*, MQFMT_*,
MQPRI_*	MQPER_*, MQMI_*, MQCI_*, MQACT_*, MQACTT_*, MQAT_*
MQMDV2	: MQGI_*, MQMF_*, MQOL_*
MQRFH2	: MQRFH_*
MQCONN	: MQCNO_*, MQCSP_*, MQCD_*, MQAIR_*, MQAIT_*
MQOPEN	: MQOO_*, MQOT_*, MQOD_*
MQPUT	: MQPMO_*, MQPMRF_*
MQGET	: MQGMO_*, MQWI_*, MQMO_*
MQCLOSE	: MQCO_*
MQSETMP	: MQPD_*
MQINQMP	: MQIMPO_*
MQCRTMH	: MQCMHO_*
MQCB	: MQCBT_*

ライセンス情報、バージョン情報、指定可能なコンスタントの表示

mqpgfに `-v` を指定すると、USAGEの表示に加え、ライセンス情報、本プログラムおよびリンクされたライブラリのバージョン情報が表示されます。さらに `-v` につづいて `all` を指定すると、本プログラムに指定可能なコンスタントの一覧が表示されます。ここで言うコンスタントとはC言語での `cmqc.h` に定義されている `#define` のことです。mqpgfは、その文字列を指定するだけで、適切なMQ内部の構造体 (MQMD、MQGMO etc) のフィールドやオプションに設定します。また、その際、適切に `or` もしくは `overwrite` を自動的に判断します。400種類以上の任意のコンスタント (MQMD_*, MQPMO_*, MQGMO_* 他) が設定可能です。各々のテストでは、これらを指定しない場合は、それぞれのMQIで使用する構造体やフィールドのデフォルト値 (MQMD_DEFAULT等) が使用されます。

(注) 全てのコンスタントについて本プログラムの動作検証を行っているわけではありません。

Ex. 3.2 ライセンス情報、バージョン情報、指定可能なコンスタントの表示

※ 「System number」はHPE NonStopの場合のみ表示されます。

```
-----
$ mqpgf -v
....
[ License information ]
System number    999999
Expires          2025. 03. 31

version 1.4.2.12 2025.10.20
library version 1.0.0.2 2024.04.05
$
$ mqpgf -v all
....
[ License information ]
System number    999999
Expires          2026. 03. 31

version 1.4.2.12 2025.10.20
library version 1.0.0.2 2024.04.05
MQMD_VERSION_1
MQMD_VERSION_2
MQMD_CURRENT_VERSION
```

MQRO_EXCEPTION

MQRO_EXCEPTION_WITH_DATA

....

クライアント・モードの使用

クライアント・モードで利用する場合は、mqpgfc コマンドを使用します。

クライアント接続のためのオプションを除き、バインドモード用の mqpgf と使用方法は同じです。

mqpgfcは -x オプションで接続先のIPアドレスまたはホスト名、および接続ポート番号、-ch オプションでMQIチャネル名、-la オプションでローカル・アドレスを受け取ります。-x が指定された場合は、MQCMO.Version に MQCNO_VERSION_2 を自動的に設定します。

-x のパラメーターの書式は "ipaddr or hostname(port)" です。Windowsの場合はダブルクォートもしくはシングルクォートで囲む必要はありません。

-x が指定された場合、mqpgfc は MQCONN() に直接接続のパラメーターを渡す為、チャネル定義テーブルなど他の接続の設定は不要です。

-x を指定しない場合は、チャネル定義テーブル、MQSERVER環境変数、mqclient.ini のいずれかで接続のパラメーターを指定することが必要です。

送信元の情報(source ipaddr/hostname, source port, tcpip process(HP NonStop))の指定が必要な場合は-la でLOCLADDRを指定します。

-la のパラメーターの書式は"local ipaddr or hostname (sender port,port) [/tcp process name]"です。"/tcp process name"はHP NonStopでのみ指定可能です。

-ch でチャネル名を指定しない場合のデフォルトは SYSTEM.DEF.SVRCONN です。

```
mqpgfc -qm <qmgr> -q <queue> -br -x <"ipaddr or hostname(port)"> -ch <channel name> -la <"source ipaddr or hostname(source port)/tcpip processname"
```

e. g.

```
mqpgfc -qm SampleQM -q SampleQ -br -x "hostname(1414)" -ch PULSAR.MQICHL -la "localhost(1234) "
```

*HP NonStopで特定のTCPIPプロセスを指定する必要がある場合は、-la "localhost(1234)/ztc3"の様にします。（\$ZTC3を指定する場合）

4. 基本的なテストの実施方法

4.1 共通パラメータ

mqpgf は、ほとんど全ての場合に、キューマネージャー名 (-qm)、キュー名 (-q) の指定が必要です。

以降、これらのパラメータの説明は省略します。

```
mqpgf -qm <qmgr> -q <queue> ....
```

-qm: キューマネージャー名

-q: キュー名

※本プログラムでは、テスト内容によって非常に多くのパラメータを指定することが必要になる場合があります。Solaris等の一部のプラットフォームでは、コマンドラインの一行に指定可能な文字数がデフォルトでより小さく設定されている場合があります。一度に入力できない場合は、'¥' で区切って複数行にして入力してください。

4.2 コマンドラインで指定したメッセージをPUT

PUTするメッセージをコマンドラインで直接指定します。特に指定が無い場合は、MQMDはデフォルトの値 (MQMD_DEFAULT) が使用されます。

```
mqpgf -qm <qmgr> -q <queue> -m <message> -n <count> -i <interval> -l <size>
```

-m: 入力メッセージ

(任意のオプション例)

-n: PUTする回数

-i: PUTする間隔(ms) (-nを指定時に有効)

-l: PUTするメッセージの長 (指定したメッセージより大きい場合、後続が0x00で埋められる。指定したメッセージより小さい場合、そのサイズにカットされる。)

Ex. 4.2.1 メッセージ件数、PUT間隔、メッセージ長を指定してPUT

※3件、5秒間隔、メッセージ長に20バイトを指定

```
$ mqpgf -qm HM8A -q LQ1 -m "test message" -n 3 -i 5000 -l 20
```

```
[20/01/23 17:48:35.994310789] 1: message length: 20 put message: test message.....
```

```
[20/01/23 17:48:40.995207421] 2: message length: 20 put message: test message.....
```

```
[20/01/23 17:48:45.995712958] 3: message length: 20 put message: test message.....
```

```
Elapsed time = 10.012609 sec
```

```
$
```

```
$ mqpgf -qm HM8A -q LQ1 -r
```

```
[20/01/23 17:49:04.255936507] 1: message length: 20 get message : test message.....
```

```
[20/01/23 17:49:04.256519249] 2: message length: 20 get message : test message.....
```

```
[20/01/23 17:49:04.256687651] 3: message length: 20 get message : test message.....
```

```
no message available : LQ1 CompCd=02 ReasonCd=2033
```

```
Elapsed time = 0.010942 sec
```

※-r は RC=2033 (MQRC_NO_MSG_AVAILABLE) までGETを繰り返す。“-l 20”により付加された0x00 は不可視文字の為、“.”で表示される。

4.3 コマンドラインで指定するPUTメッセージをHexaDecimalで指定

コマンドラインで指定するメッセージを16進形式で指定します。

```
mqpgf -qm <qmgr> -q <queue> -mx <input message(hexadecimal notation)>  
-n <count> -i <interval> -l <size>
```

Ex. 4.3.1 メッセージを16進表記で指定してPUT

```
-----  
$ mqpgf -qm HM8A -q LQ1 -mx 0123abcdABCD  
[20/01/23 17:52:25.875333411] 1: message length: 6 put message: 0x0123ABCDABCD  
Elapsed time = 0.011225 sec  
$  
$ mqpgf -qm HM8A -q LQ1 -hex  
[20/01/23 17:52:35.760981978] 1: message length: 6 get message : 0x0123ABCDABCD  
Elapsed time = 0.011612 sec  
※-hexはGETしたメッセージを16進表記で表示させます。  
-----
```

4.4 GETしたメッセージを標準出力へ書き込み(可視文字のみ)

GETしたメッセージを標準出力へ書き込みます。デフォルトでは不可視文字は”.”で出力されます。MQMDは破棄されます。さらに伝送キューからGETする場合は、伝送キューヘッダとメッセージ本体のMQMD、MQMDEも削除されます。デッドレターキューからGETする場合は、MQDLHも削除されます。

```
mqpgf -qm <qmgr> -q <queue> -r
```

(任意のオプション例)

-r: キュー内にメッセージが無くなる迄、繰り返しGETする。

Ex. 4.4.1 不可視文字を含むメッセージ (バイナリデータ) のGET

```
-----
$ mqpgf -qm HM8A -q LQ1 -mx 01020304057FFF
[20/01/23 17:54:06.975709566] 1: message length: 7 put message: 0x01020304057FFF
Elapsed time = 0.010685 sec
$
$ mqpgf -qm HM8A -q LQ1
[20/01/23 17:54:13.377889722] 1: message length: 7 get message : .....
Elapsed time = 0.008381 sec
-----
```

4.5 ファイルデータをPUT

ファイルデータをそのままキューにPUTします。ファイルデータはMQMDを除いた部分のデータです。特に指定が無い場合は、MQMDはデフォルトの値 (MQMD_DEFAULT) が使用されます。

下記の-n オプション、-i オプションは他のテストでも使用可能な場合があります。

```
mqpgf -qm <qmgr> -q <queue> -f <filename> -n <count> -i <interval> -l <size>
```

-f: 入力ファイル名

(任意のオプション例)

-n: ファイルをPUTする回数

-i: PUTする間隔(ms) (-nを指定時に有効)

-l: PUTするメッセージの長さ(指定したメッセージより大きい場合、後続が0x00で埋められる。指定したメッセージより小さい場合、そのサイズにカットされる。)

Ex. 4.5.1 ファイルのPUT回数、間隔を指定

```
-----
$ ls -l input.txt
-rw-r--r--  1 mq80      mqm                19 Dec 14 19:13 input.txt
$
$ cat input.txt
input file message
$
$ od -x input.txt
0000000  696e 7075 7420 6669 6c65 206d 6573 7361
0000020  6765 0a00
0000023
$
※PUT回数に2回、PUTする間隔に3秒を指定
$ mqpgf -qm HM8A -q LQ1 -f input.txt -n 2 -i 3000
[20/01/23 17:56:22.285117540] 1: put from: input.txt
[20/01/23 17:56:25.285613927] 2: put from: input.txt
Elapsed time = 3.010767 sec
$
$ mqpgf -qm HM8A -q LQ1
[20/01/23 17:56:32.492878183] 1: message length: 19 get message : input file mes
sage

Elapsed time = 0.009688 sec
$ mqpgf -qm HM8A -q LQ1 -hex
```

[20/01/23 17:56:38.486576036] 1: message length: 19 get message : 0x696E70757420
66696C65206D6573736167650A
Elapsed time = 0.008977 sec

4.6 GETしたメッセージをファイルへ出力

GETしたメッセージをファイルに書き込みます。MQMDは破棄されます。さらに伝送キューからGETする場合は、伝送キューヘッダとメッセージ本体のMQMD、MQMDEも削除されます。デッドレターキューからGETする場合は、MQDLHも削除されます。

```
mcpqgf -qm <qmgr> -q <queue> -o <filename>
```

-o: 出力ファイル名

-o で指定されたファイルが既に存在する場合、そのファイルを上書きして良いかどうか確認されます。

```
file already exist. overwrite ? y/n : y
```

処理を継続する場合は(y|Y)、処理を中断する場合は、(n|N)を入力します。

※下記のオプションは同時に指定しても無視されます。

-r(キュー上の全てメッセージをGET)と-n(PUT/GETするメッセージの数の指定)および-raw(rawモード出力)と-hex(16進表記で出力)

Ex. 4.6.1 GETしたメッセージをファイルに出力

```
-----
$ mcpqgf -qm HM8A -q LQ1 -m "line 1
> line 2
> line 3
> "
[20/01/23 18:19:05.753343645] 1: message length: 21 put message: line 1
line 2
line 3

Elapsed time = 0.013499 sec
※Unixのシェルでは上記の様に改行も入力も可能
$
$ mcpqgf -qm HM8A -q LQ1 -o /home/mqm/tmp/output.txt
file already exist. overwrite? y/n : y
[20/01/23 18:20:51.591537336] 1: message length: 21 output filename : /home/mqm/
tmp/output.txt
Elapsed time = 1.311320 sec
$ cat /home/mqm/tmp/output.txt
```

line 1

line 2

line 3

\$

※出力に指定したファイルが既に存在する場合は、上書きしてよいかどうか確認されます。

MQMD の Version2 が使用されているメッセージを GET する際に MQMD の Version1 を指定した場合、MQMDE (拡張メッセージ記述子) がキューマネージャーによって自動的に作成されます。ただし、Version2 で拡張されたフィールドが実際に使用されていない場合 (デフォルト値のままの場合) には MQMDE ヘッダは作成されません。

下記はMQMDEが生成された場合のテスト例です。mqpgfはMQMDEを削除し「cut MQMDE header」というメッセージを出力します。

Ex. 4.6.2 MQMDのバージョンを指定してGET

※MQMD V2を使用するメッセージをPUTする。

```
$ mqpgf -qm HM8A -q LQ1 -m test -gi GID -ms 3 -of 100 -ol 1000 MQMD_VERSION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP
[20/01/24 09:43:01.721510414] 1: message length: 4 put message: test
Elapsed time = 0.014961 sec
```

※PUT時に、MQMD Version2を指定 (MQMD_VERSION_2) し、Version2で拡張されたフィールドをデフォルト以外に設定しています。Offset (-of)、GroupId (-gi)を設定する為にMQMF_SEGMENT、OriginalLength (-ol)を設定する為に MQMT_REPORT、さらに MsgSeqNumber (-ms)を設定する為に MQMF_MSG_IN_GROUP を指定しています。

※MQMD V2を使用しているメッセージをMQMDのV1を指定してGETする。(デフォルトで MQMD_VERSION_1 が使用されます。)

```
$ mqpgf -qm HM8A -q LQ1 -o /home/mqm/tmp/output.txt
file already exist. overwrite? y/n : y
Cut MQMDE Header
[20/01/24 09:43:51.817819504] 1: message length: 4 output filename : /home/mqm/tmp/output.txt
Elapsed time = 1.919485 sec
```


MQMD の Version2 が使用されているメッセージは伝送キュー上では、MQMD V1 と MQMD E に分かれている状態になっています。

Ex. 4.6.3 伝送キュー上のメッセージをGET

```
-----  
C:\¥Users¥mqm>mqpgfc -qm HM8A -q REQ8B1 -x "remotehost(1414)" -ch PULSAR.MQICHL -  
f input.txt -gi GID -ms 3 -of 100 -ol 1000 MQMD_VERSION_2 MQMF_SEGMENT MQMT_REPO  
RT MQMF_MSG_IN_GROUP  
[2012/01/30 11:08:13.801] 1: put from: input.txt  
Elapsed time = 187 msec
```

※伝送キューは通常読み込み禁止になっていますので、GETする前に読み込みが可能に設定します。

```
C:\¥Users¥mqm>mqpcfc que -qm HM8A -q HM8B -x "remotehost(1414)" -ch PULSAR.MQICHL  
L GET  
1: QUEUE(HM8B) TYPE(QLOCAL) GET(DISABLED)
```

```
C:\¥Users¥mqm>mqpcfc get enable -qm HM8A -q HM8B -x "remotehost(1414)" -ch PULSAR.MQICHL  
Get Enabled : HM8B
```

```
C:\¥Users¥mqm>mqpcfc que -qm HM8A -q HM8B -x "remotehost(1414)" -ch PULSAR.MQICHL  
L GET  
1: QUEUE(HM8B) TYPE(QLOCAL) GET(ENABLED)
```

```
C:\¥Users¥mqm>mqpgfc -qm HM8A -q HM8B -x "remotehost(1414)" -ch PULSAR.MQICHL -o  
output.txt  
file already exist. overwrite? y/n : y  
Cut Xmit Queue Header and message body MQMD  
Cut MQMDE Header  
[2012/01/30 11:08:54.287] 1: message length: 12 output filename : output.txt  
Elapsed time = 2578 msec
```

```
C:\¥Users¥mqm>type output.txt  
test message  
-----
```

MQMD の Version2 が使用されているメッセージがデッドレターキューに入れられた場合、MQDLH は MQMD V1 と MQMDE の間に挿入されます。（MQMD V1、MQDLH、MQMDE の順）

Ex. 4.6.4 デッド・レター・キュー上のメッセージをGET

```
-----
$ mqpgfc -qm HM8A -q SYSTEM.DEAD.LETTER.QUEUE -x "172.21.10.50(18
591)" -ch PULSAR.MQICHL -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[4] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQDEAD ] Priority[0] Persistence[0] MsgId[0x414D
5120484D394C2020202020202087A9405E03630A22] CorrelId[0x00000000000000000000
000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[HM9L
] UserIdentifier[mqm ] AccountingToken[0x04313030300000000000000000000000
0000000000000000000000000000000000] ApplIdentityData[
] PutApplType[6] PutApplName[mqpgf ] PutDate[2020021
0] PutTime[00550607] ApplOriginData[ ]

GroupId[0x00000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

*StrucId[DLH ] Version[1] Reason[2085] DestQName[LLQ
] DestQMgrName[HM8A
] Encoding[546] CodedCharSetId[1208] Format[MQHMDE ] PutApplType[13] PutApplNa
me[amqrmppa ] PutDate[20120210] PutTime[00560505]

*StrucId[MDE ] Version[2] StrucLength[72] Encoding[546] CodedCharSetId[1208] For
mat[ ] Flags[0] GroupId[0x474944000000000000000000000000000000000000000000
0] MsgSeqNumber[3] Offset[100] MsgFlags[10] OriginalLength[1000]

data length: 13
00000000: 7465 7374 206D 6573 7361 6765 0A 'test message. '

Elapsed time = 0.105855 sec

$
$ mqpgfc -qm HM8A -q SYSTEM.DEAD.LETTER.QUEUE -x "172.21.10.50(18
591)" -ch PULSAR.MQICHL -o output.txt
file already exist. overwrite? y/n : y
Cut Dead Letter Header
Cut MQMDE Header
[20/02/10 11:35:27.245865904] 1: message length: 13 output filename : output.txt
```

Elapsed time = 1.536663 sec

\$

4.7 GETしたメッセージをキューへ出力(リキュー)

GETしたメッセージを他のキューにリキューします。

出力側（2次側）のキューにパラメータ／オプションを設定する場合は `-ss` オプションを使用して以降のパラメータ／オプションを出力側（2次側）のキューに切り替えます。（「5. 全パラメータ・リファレンス」で以降のパラメータを2次側に切り替える（`-ss`）」参照）

```
mqpgf -qm <qmgr> -q <queue> -oq <output queue> -r
```

`-oq`: 出力キュー名

(任意のオプション例)

`-r`: キュー内にメッセージが無くなる迄、繰り返しGETする。

Ex. 4.7.1 入力キューの全てのメッセージを他のキューへ出力

※入力キュー上に3件メッセージをPUT

```
$ mqpgf -qm HM8E2 -q LQ1 -m "sample message" -n 3
```

```
[20/02/28 15:38:15.736915] 1: message length: 14 put message: sample message
```

```
[20/02/28 15:38:15.742337] 2: message length: 14 put message: sample message
```

```
[20/02/28 15:38:15.742439] 3: message length: 14 put message: sample message
```

```
Elapsed time = 0.179671 sec
```

※入力キューのメッセージを繰り返しGETし、その全てを出力キューへ転送。その際出力側（2次側）に対してMQMDのMsgTypeとCodedCharSetIdを指定する。

```
$ mqpgf -qm HM8E2 -q LQ1 -oq LQ2 -r -ss MQMT_REPLY -cc 930
```

```
[20/02/28 15:39:16.385869] 1: message length: 14 get message : sample message
```

```
[20/02/28 15:39:16.393922] 1: message length: 14 put message : sample message
```

```
[20/02/28 15:39:16.394323] 2: message length: 14 get message : sample message
```

```
[20/02/28 15:39:16.394436] 2: message length: 14 put message : sample message
```

```
[20/02/28 15:39:16.394974] 3: message length: 14 get message : sample message
```

```
[20/02/28 15:39:16.395086] 3: message length: 14 put message : sample message
```

```
no message available : LQ1 CompCd=02 ReasonCd=2033
```

```
Elapsed time = 0.129444 sec
```

```
$ mqpgf -qm HM8E2 -q LQ2 -dp -r
```

```
message number: 1
```

```
*StrucId[MD ] .... MsgType[2] .... CodedCharSetId[930] ....
```

```
....
```

```
00000000: 7361 6D70 6C65 206D 6573 7361 6765 'sample message'
```

```
message number: 2
```

```
*StrucId[MD  ] .... MsgType[2] .... CodedCharSetId[930] ....  
....  
00000000: 7361 6D70 6C65 206D 6573 7361 6765      'sample message '  
  
message number: 3  
*StrucId[MD  ] .... MsgType[2] .... CodedCharSetId[930] ....  
....  
00000000: 7361 6D70 6C65 206D 6573 7361 6765      'sample message '  
  
no message available : LQ2 CompCd=02 ReasonCd=2033  
Elapsed time = 0.063891 sec  
-----
```

4.8 PUTしたメッセージの応答を受信

メッセージをPUTし、その応答を受信します。

入力側（2次側）のキューにパラメータ／オプションを設定する場合は `-ss` オプションを使用して以降のパラメータ／オプションを入力側（2次側）のキューに切り替えます。（「5. 全パラメータ・リファレンス — 以降のパラメータを2次側に切り替える（-ss）」参照）

```
mqpgf -qm <qmgr> -q <queue> -m <input message> -iq <input queue> -n <count> -i <interval>
```

-iq: 入力キュー名

(任意のオプション例)

-n: PUTする回数

-i: PUTする間隔(ms) (-nを指定時に有効)

Ex. 4.8.1 PUT後、その応答メッセージを受信

※PUT後、-iq に指定した受信キューから応答メッセージを受信する処理を1秒間隔で3回実施

```
$ mqpgf -qm HM8E2 -q RemoteQ -m "request reply test" -iq ReplyQ MQMT_REQUEST -n 3 -i 1000 -ss MQGMO_WAIT MQWI_UNLIMITED
[20/02/28 16:05:40.042057] 1: message length: 18 put message: request reply test
[20/02/28 16:05:40.047693] 1: message length: 18 get message : request reply test
[20/02/28 16:05:41.048099] 2: message length: 18 put message: request reply test
[20/02/28 16:05:41.048588] 2: message length: 18 get message : request reply test
[20/02/28 16:05:42.049564] 3: message length: 18 put message: request reply test
[20/02/28 16:05:42.050027] 3: message length: 18 get message : request reply test
Elapsed time = 2.332243 sec
```

4.9 ディレクトリ内のファイルデータを全てPUT

ディレクトリ内のファイルデータを全てPUTします。

ファイルデータはMQMDを除いた部分のデータです。特に指定のない場合は、MQMDはデフォルトの値(MQMD_DEFAULT)が使用されます。

```
mqpgf -qm <qmgr> -q <queue> -d <directory> -n <count> -i <interval> -l <size>
```

-d: 入力ディレクトリ名

(任意のオプション例)

-n: 個々のファイルをPUTする回数

-i: PUTする間隔(ms) (-nを指定時に有効)

-l: PUTするメッセージの長さ(指定したメッセージより大きい場合、後続が0x00で埋められる。指定したメッセージより小さい場合、そのサイズにカットされる。)

Ex. 4.9.1 ディレクトリ内のファイルを回数、間隔を指定してPUT

※ディレクトリ” input” 内のファイルを2回ずつ3秒間隔でPUTする。

```
$ mqpgf -qm HM8E2 -q LQ1 -d input -n 2 -i 3000
```

```
[20/02/28 16:30:16.551001] 1: put from: input/test1.txt
```

```
[20/02/28 16:30:19.558533] 2: put from: input/test1.txt
```

```
[20/02/28 16:30:22.559016] 1: put from: input/test2.txt
```

```
[20/02/28 16:30:25.560174] 2: put from: input/test2.txt
```

```
[20/02/28 16:30:28.560654] 1: put from: input/test3.txt
```

```
[20/02/28 16:30:31.561816] 2: put from: input/test3.txt
```

```
Elapsed time = 15.299729 sec
```

```
$
```

```
$ amqsbcbg LQ1 HM8E2 | grep PutDate
```

```
PutDate : '20120228' PutTime : '07301655'
```

```
PutDate : '20120228' PutTime : '07301955'
```

```
PutDate : '20120228' PutTime : '07302255'
```

```
PutDate : '20120228' PutTime : '07302556'
```

```
PutDate : '20120228' PutTime : '07302856'
```

```
PutDate : '20120228' PutTime : '07303156'
```

※MQMDのPutDate/PutTimeはGMT (JST-9)

4.10 キュー内のメッセージをGETしディレクトリに出力

キュー内のメッセージを指定メッセージ数、または全てGETし、そのデータをメッセージ毎にユニークなファイル名で保存します。MQMDは破棄されます。さらに伝送キューからGETする場合は、伝送キューヘッダとメッセージ本体のMQMD、MQMDEも削除されます。デッドレターキューからGetする場合は、MQDLHも削除されます。

```
mqpgf -qm <qmgr> -q <queue> -g <directory> [-n <count> | -r] -i <interval>
```

-g: 出力ディレクトリ名

(任意のオプション例)

-n: GETの回数 (-r と同時に指定できない)

-r: キュー上のメッセージを全てGET (-n と同時に指定できない)

-i: GETする間隔(ms) (-r または -n を指定時に有効)

出力ファイル名形式:

yyyymmdd_HHMMSS_msec_seqno

yyyymmdd、HHMMSS、msecはGETしたMQMDのPutDate、PutTimeから取得されます。

同じ時刻のメッセージが複数ある場合は、seqnoが初回なし、2回目から 1, 2, 3... の様に付加されます。

-g で指定されたディレクトリが既に存在する場合、そのディレクトリにファイルを出力して良いかどうか確認されます。

```
directory already exist. overwrite ? y/n : y
```

処理を継続する場合は(y|Y)、処理を中断する場合は、(n|N)を入力します。

Ex. 4.10.1 キュー上の全てのメッセージを指定間隔でGETし、ディレクトリに出力

※0.5秒間隔でGETするように指定する。

```
$ mqpgf -qm HM8E2 -q LQ1 -g /home/mqm/output -r -i 500
```

```
directory already exist. overwrite? y/n : y
```

```
[20/02/28 16:55:39.711720] 1: write file : /home/mqm/output/20120228_163016_55
```

```
[20/02/28 16:55:40.262528] 2: write file : /home/mqm/output/20120228_163019_55
```

```
[20/02/28 16:55:40.799099] 3: write file : /home/mqm/output/20120228_163022_55
```

```
[20/02/28 16:55:41.336690] 4: write file : /home/mqm/output/20120228_163025_56
```

```
[20/02/28 16:55:41.879415] 5: write file : /home/mqm/output/20120228_163028_56
```

```
[20/02/28 16:55:42.423150] 6: write file : /home/mqm/output/20120228_163031_56
```

```
no message available : LQ1 CompCd=02 ReasonCd=2033
```


Elapsed time = 5.614104 sec

※-r を指定した場合は、RC=2033 (MQRC_NO_MSG_AVAILABLE)になるまでGETが実施されます。

4.11 キュー上のメッセージのブラウズ／ダンプ（通常）

キュー上のメッセージをブラウズし、結果を16進ダンプ表示します。MQMDに関するフィールド毎に表示します。

※ブラウズ／ダンプ時は例外として、デフォルトでMQMD_VERSION_2を使用します。
（他のMQGETを使用するテストではMQMD_DEFAULTが使用される為、MQMD_VERSION_1がデフォルトです。）特別にMQMD_VERSION_1を使用したい場合は、引数に MQMD_VERSION_1を指定する必要があります。

```
mqpgf -qm <qmgr> -q <queue> -br -r
```

-br: ブラウズ(通常)

(任意のオプション例)

-r: キュー内にメッセージが無くなる迄、繰り返しGETする。

MQMD_VERSION_1: MQMD_VERSION_2で追加されたフィールドが使用された場合、MQMDEヘッダが生成されます。

Ex. 4.11.1 キュー上の全てのメッセージをブラウズ

```
-----
$ mqpgf -qm HM8E2 -q LQ1 -br -r
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[          ] Priority[0] Persistence[0] MsgId[0x414D
5120484D384532202020202020205E58AD7820003B0D] CorrelId[0x0000000000000000000000
000000000000000000000000] BackoutCount[0] ReplyToQ[
          ] ReplyToQMgr[HM8E2
    ] UserIdentifier[mqm          ] AccountingToken[0x0534343033310000000000000000
000000000000000000000000000000000006] ApplIdentityData[
          ] PutApplType[13] PutApplName[mqpgf          ] PutDate[201202
28] PutTime[08343590] ApplOriginData[    ]

GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

data length: 8
00000000:  6D65 7373 6167 6531          'message1          '

message number: 2
....
```

Elapsed time = 0.062943 sec

Ex. 4.11.2 GET時にMQMDEヘッダが生成される場合の例

※MQMD V2を使用しているメッセージをPUTする。

```
$ mqpgf -qm HM8E2 -q LQ1 -m test -gi GID -ms 3 -of 100 -ol 1000 MQMD_VERSION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP
```

```
[16/12/22 19:55:43] 1: message length: 4 put message : test
```

```
[20/02/28 17:39:47.920439] 1: message length: 4 put message: test
```

Elapsed time = 0.180878 sec

\$

※MQMD V2を使用しているメッセージをMQMDのV1を指定してブラウズする。

```
$ mqpgf -qm HM8E2 -q LQ1 -br MQMD_VERSION_1
```

message number: 1

```
*StrucId[MD ] Version[1] Report[0] MsgType[4] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[819] Format[MQHMDE ] Priority[0] Persistence[0] MsgId[0x414D5120484D384532202020202020205E58AD7820003F08] CorrelId[0x0000000000000000000000000000000000000000000000000000000000000000] BackoutCount[0] ReplyToQ[ ] ReplyToQMgr[HM8E2] UserIdentifier[mqm ] AccountingToken[0x0534343033310000000000000000000000000000000000000000000000000000] ApplIdentityData[ ] PutApplType[13] PutApplName[mqpgf ] PutDate[20120228] PutTime[08422337] ApplOriginData[ ]
```

```
GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]
```

data length: 76

```
00000000: 4D44 4520 0000 0002 0000 0048 0000 0111 'MDE .....H....'
00000010: 0000 0333 2020 2020 2020 2020 0000 0000 '...3 ....'
00000020: 4749 4400 0000 0000 0000 0000 0000 0000 'GID.....'
00000030: 0000 0000 0000 0000 0000 0003 0000 0064 '.....d'
00000040: 0000 000A 0000 03E8 7465 7374 '.....test'
```

Elapsed time = 0.061495 sec

※MQMD. Formatに"MQHMDE "(MQFMT_MD_EXTENSION)がセットされ、データ部の先頭が"MDE "(MQMDE_STRUC_ID)になっている。

4.12 キュー上のメッセージのブラウズ／ダンプ（詳細）

キュー上のメッセージをブラウズし、結果を16進ダンプ表示します。MQMD(独立メッセージ記述子)、MQXQH(伝送キュー・ヘッダー)、メッセージ・データのMQMD(組み込みメッセージ記述子)、MQMDE(拡張メッセージ記述子)、MQDLH(送達不能ヘッダー)、MQRFH2(規則および書式ヘッダー 2)およびPCF(プログラマブル・コマンド・フォーマット)がフィールド毎に表示されます。

下表に示す項目については、可視文字のみ出力され、不可視文字については”.”が代替文字として表示されます。これらのフィールドは表示方法を変更することが可能です。-hex を指定した場合は16進表記、-rawを指定した場合はそのまま標準出力にその文字コードが送られます。

表 4.12.1 表示モード指定が有効な項目

データタイプ	項目	注記
MQCA_ALTERATION_DATE	数値以外の項目	PCF
MQCFT_BYTE_STRING_FILTER	数値以外の項目	PCF
MQCFT_BYTE_STRING	数値以外の項目	PCF
MQCFT_STRING_FILTER	数値以外の項目	PCF
MQCFT_STRING_LIST	数値以外の項目	PCF
MQCFT_STRING	数値以外の項目	PCF
MQDLH	数値以外の項目	
MQRFH2	NameValueData も含む数値以外の項目	
MQMD	Report	
MQMD	MsgFlags	MQMD Ver 2
MQMDE	MsgFlags	

mqpgf -qm <qmgr> -q <queue> **-brv** -hex

-brv: ブラウズ(詳細)

(任意のオプション例)

-raw: Rawモード出力

-hex: 16進表記で出力

Ex. 4.12.1 伝送キュー内のメッセージの詳細ダンプ

※MQMDのVer2が指定され、実際にMQMD Ver2で拡張されたフィールドが使用されている(デフォルト以外)メッセージが伝送キューにPUTされた場合、その伝送キュー内のメッセージはMQMD Ver1にMQMDEが付加された状態になっています。

※独立MQMD、伝送キューヘッダ、メッセージのMQMD（組み込みMQMD）およびMQMDEの文字コードとエンコーディングの変換は、GET時にキューマネージャーが自動的行います。後続に伝送キューヘッダがある場合のその前のMQMDの Encoding フィールドは、そのプラットフォームのエンコーディングが設定されます。その為、mqpgfはこの3種類のヘッダに関しては表示する際にエンコーディングの変換を行いません。

※チャネルを停止後、MQMD_VERSION_2 (mqpgfのPUT時のデフォルトはMQMD_VERSION_1)、そして、MQMD V2で拡張されたフィールドをデフォルト以外にする為に、MQMF_LAST_MSG_IN_GROUP (0x00000010) を指定してリモートキューのローカル定義にPUTします。MQMF_LAST_MSG_IN_GROUPを引数に指定すると、MQMD V2の MsgFlags に自動的に設定されます。

```
$ mqpgrf -qm HM8B -q REP5A1 -m "dump xmitq" MQMD_VERSION_2 MQMF_LAST_MSG_IN_GROUP
[20/03/02 17:47:06.595354] 1: message length: 10 put message: dump xmitq
MQCMIT success : CompCd=00 ReasonCd=00
Elapsed time = 0.119847 sec
```

```
※伝送キューを読み込み許可にします。
$ mqpcf get enable -qm HM8B -q HM5A
Get Enabled : HM5A
```

```
$ mqpgf -qm HM8B -q HM5A -brv
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQXMIT  ] Priority[0] Persistence[0] MsgId[0x414D
5120484D3842202020202020202020205E5CC1B120002A0A] CorrelId[0x414D5120484D3842202020
020202020205E5CC1B120002A08] BackoutCount[0] ReplyToQ[
                ] ReplyToQMgr[HM8B
                ] UserIdentifier[mqm                ] AccountingToken[0x05343430333100000000000000000000
000000000000000000000000000000000006] ApplIdentityData[
                ] PutApplType[7] PutApplName[HM8B                ] PutDate[2012030
2] PutTime[08470659] ApplOriginData[                ]

GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]
```

```

*StrucId[XQH ] Version[1] RemoteQName[REP1
    ] RemoteQMgrName[HM5A
]

*StrucId[MD ] Version[1] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQHMDE ] Priority[0] Persistence[0] MsgId[0x414D
5120484D38422020202020202020205E5CC1B120002A08] CorrelId[0x00000000000000000000
000000000000000000000000] BackoutCount[0] ReplyToQ[
    ] ReplyToQMgr[HM8B
] UserIdentifier[mqm
] AccountingToken[0x05343430333100000000000000000000
00000000000000000000000000000000000000000000] ApplIdentityData[
    ] PutApplType[13] PutApplName[mqpgf
] PutDate[201203
02] PutTime[08470659] ApplOriginData[
]

*StrucId[MDE ] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[819] Form
at[
] Flags[0] GroupId[0x414D5120484D38422020202020202020205E5CC1B120002A0
9] MsgSeqNumber[1] Offset[0] MsgFlags[24] OriginalLength[-1]

```

```

data length: 10
00000000:  6475 6D70 2078 6D69 7471          'dump xmitq      '

```

```

MQCMIT success : CompCd=00 ReasonCd=00
Elapsed time = 0.074383 sec

```

※MQMF_LAST_MSG_IN_GROUP(0x00000010)が指定された場合、キューマネージャーはMQMF_MSG_IN_GROUP(0x00000008)も追加設定します。その結果、ブラウズした時に現れるMQMDEのMsgFlagsには、2つの合計である0x00000018 = 24 が表示されます。

※伝送キューを読み込み禁止に戻します。

```

$ mqpcf get disable -qm HM8B -q HM5A
Get Disabled : HM5A

```

Ex. 4.12.2 デッドレターキュー内のメッセージのダンプ

※DLHはデッドレターキューがあるマシンのエンコーディングで作成されPUTされます。その為、MQMDのエンコーディングもDLQを持つマシンのエンコーディングが設定されます。mqpgf はダンプ表示時は数値フィールドに対してエンコーディングの変換を行った値を表示します。

```

mqpgfc -qm HM8E2 -q SYSTEM. DEAD. LETTER. QUEUE -x "16.147.169.57(18502)" -ch PULSA
R. MQICHL -brv

```

```

message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQDEAD ] Priority[0] Persistence[0] MsgId[0x414D
5120484D384D31202020202020205E5DB9A820003002] CorrelId[0x00000000000000000000
0000000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[HM8M1
] UserIdentifier[mqm ] AccountingToken[0x05343430333100000000000000000000
0000000000000000000000000000000000000000000000000000] ApplIdentityData[
] PutApplType[13] PutApplName[mqpgf ] PutDate[201203
03] PutTime[02405706] ApplOriginData[ ]

```

```

GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

```

```

*StrucId[DLH ] Version[1] Reason[2053] DestQName[CQE2
] DestQMgrName[HM8E2
] Encoding[546] CodedCharSetId[819] Format[MQHMDE ] PutApplType[13] PutApplName[amqrmppa
] PutDate[20120303] PutTime[02410719]

```

```

*StrucId[MDE ] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[819] Format[
] Flags[0] GroupId[0x414D5120484D384D31202020202020205E5DB9A820003003] MsgSeqNumber[1] Offset[0] MsgFlags[24] OriginalLength[-1]

```

```

data length: 4
00000000: 7465 7374 'test'

```

```

no message available : SYSTEM.DEAD.LETTER.QUEUE CompCd=02 ReasonCd=2033
Elapsed time = 1835 msec

```

Ex. 4.12.3 RFH2ヘッダを含むJMSメッセージのダンプ

```

$ mqpgf -qm HM8E2 -q LQ1 -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[1208] Format[MQHRF2 ] Priority[4] Persistence[1] MsgId[0x414
D5120484D38453220202020202020205E5DB3D420004002] CorrelId[0x00000000000000000000
0000000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[HM8E2
] UserIdentifier[mqm ] AccountingToken[0x05343430333100000000000000000000
0000000000000000000000000000000000000000000000000000] ApplIdentityData[

```

```

] PutApplType[13] PutApplName[java] PutDate[20120
303] PutTime[03132220] ApplOriginData[ ]

```

```

GroupId[0x00000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

```

```

*StrucId[RFH ] Version[2] StrucLength[148] Encoding[273] CodedCharSetId[1208] Fo
rmat[MQSTR ] Flags[0] NameValueCCSID[1208]
NameValueLength[32] NameValueData[<mcd><Msd>jms_text</Msd></mcd> ]
NameValueLength[72] NameValueData[<jms><Dst>queue:///LQ1</Dst><Tms>1583205202175
</Tms><Dlv>2</Dlv></jms> ]
data length: 11
00000000: 4A4D 5320 4D65 7373 6167 65 'JMS Message '

```

Elapsed time = 0.066488 sec

Ex. 4.12.4 PCFフォーマットのダンプ

※下記はクラスタ伝送キューをブラウズした例の抜粋です。

```

$ mqpgf -qm HM8E2 -q SYSTEM.CLUSTER.TRANSMIT.QUEUE -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[22535239] Feedback[0] Enco
ding[273] CodedCharSetId[819] Format[MQXMIT ] Priority[0] Persistence[1] MsgId
[0x414D5120484D38453220202020202020205E58AD782000020A] CorrelId[0x544F2E484D384D32
20202020202020202020202020202020] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[HM8E2
] UserIdentifier[mqm ] AccountingToken[0x00000000000000000000000000000000]
0000000000000000000000000000000000000000000000000000000] ApplIdentityData[
] PutApplType[7] PutApplName[HM8E2] PutDate
[20120228] PutTime[06044187] ApplOriginData[ ]

```

```

GroupId[0x00000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

```

```

*StrucId[XQH ] Version[1] RemoteQName[SYSTEM.CLUSTER.COMMAND.QUEUE
] RemoteQMgrName[HM8M2]

```

```

*StrucId[MD ] Version[1] Report[0] MsgType[8] Expiry[22535239] Feedback[0] Enco
ding[273] CodedCharSetId[819] Format[MQADMIN ] Priority[0] Persistence[1] MsgId

```



```
[0x414D5120484D38453220202020202020205E58AD7820000204] CorrelId[0x0000000000000000
00000000000000000000000000000000] BackoutCount[0] ReplyToQ[
    ] ReplyToQMgr[HM8E2
    ] UserIdentifier[mqm    ] AccountingToken[0x0534343033310000000000
00000000000000000000000000000000000000000000000000000000000000000006] ApplIdentityData[
    ] PutApplType[7] PutApplName[amqrrmfa    ] PutDate
[20120228] PutTime[06044185] ApplOriginData[    ]
```

```
*MQCFH(MQCFT_COMMAND) Type[1] StrucLength[36] Version[1] Command[1001] MsgSeqNu
mber[1] Control[1] CompCode[0] Reason[0] ParameterCount[83]
(MQCFT_STRING) Type[4] StrucLength[68] Parameter[5514] CodedCharSetId[0] String
Length[48] String[HM8E2_2012-02-03_16.49.24    ]
(MQCFT_STRING) Type[4] StrucLength[68] Parameter[5502] CodedCharSetId[0] String
Length[48] String[HM8M2_2011-11-26_15.28.42    ]
(MQCFT_STRING) Type[4] StrucLength[68] Parameter[5515] CodedCharSetId[0] String
Length[48] String[HM8E2_2012-02-03_16.49.24    ]
...
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1502] Value[50]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1503] Value[0]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1505] Value[10]
...
(MQCFT_INTEGER_LIST) Type[5] StrucLength[24] Parameter[1575] Count[2] Values[0,
-1]
(MQCFT_INTEGER_LIST) Type[5] StrucLength[80] Parameter[1576] Count[16] Values
[0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
...
(MQCFT_STRING) Type[4] StrucLength[20] Parameter[3516] CodedCharSetId[0] String
Length[0] String[]
(MQCFT_STRING_LIST) Type[6] StrucLength[72] Parameter[5500] CodedCharSetId[0] C
ount[1] StringLength[48] String1[PLDCL
    ]
```

Elapsed time = 0.371101 sec

4.13 GETしたメッセージを標準出力へ書き込み(rawモード)

メッセージをそのままの内容でファイルにリダイレクトしたい場合などは、Rawモードで出力します。`-o` または `-g` オプションでファイルに出力する場合は、`-raw` オプションは必要ありません。(標準出力へ書き込む場合のみ有効)

```
mppgf -qm <qmgr> -q <queue> -raw -r
```

`-raw`: Rawモード出力

(任意のオプション例)

`-r`: キュー内にメッセージが無くなる迄、繰り返しGETする。

Ex. 4.13.1 バイナリデータをリダイレクトしてファイルへ保存

```
-----  
$ mppgf -qm HM8A -q LQ1 -mx 010231324142  
[20/03/05 13:43:09.883428] 1: message length: 6 put message: 0x010231324142  
Elapsed time = 0.388001 sec  
  
$ mppgf -qm HM8A -q LQ1 -raw > redirect.msg  
Elapsed time = 0.148884 sec  
  
$ od -x redirect.msg  
00000000 0102 3132 4142  
00000006  
-----
```

4.14 GETしたメッセージを標準出力へ書き込み(16進表記)

GETしたメッセージを16進表記で出力します。`-o` または `-g` オプションと共に指定された場合は無視されます。(標準出力へ書き込む場合のみ有効)

```
mqpgf -qm <qmgr> -q <queue> -hex -r
```

`-hex`: 16進表記で出力

(任意のオプション例)

`-r`: キュー内にメッセージが無くなる迄、繰り返しGETする。

Ex. 4.14.1 バイナリデータを16進表記で標準出力へ書き込み

```
-----  
$ mqpgf -qm HM5A -q LQ1 -mx 010231324142  
[20/03/05 14:06:08.975989] 1: message length: 6 put message: 0x010231324142  
Elapsed time = 0.480365 sec  
$ mqpgf -qm HM5A -q LQ1 -mx 080938396162  
[20/03/05 14:06:19.078512] 1: message length: 6 put message: 0x080938396162  
Elapsed time = 0.194654 sec  
%mqpgf -qm HM5A -q LQ1 -hex -r  
[20/03/05 14:06:27.181172] 1: message length: 6 get message : 0x010231324142  
[20/03/05 14:06:27.181743] 2: message length: 6 get message : 0x080938396162  
no message available : LQ1 CompCd=02 ReasonCd=2033  
Elapsed time = 0.207737 sec  
-----
```

4.15 PCFフォーマット・メッセージをPUT

プレーンテキストファイルに下記のようにPCFの定義を記述することで、バイナリのPCFフォーマットを作成し、指定キューへPUTすることができます。

PUT時には、作成されたフォーマットを標準出力へも表示します。下記のパラメータ構造には個別にCCSIDが設定可能です。通常は表示可能なコードのみを表示しようと思いますが、プラットフォームの違うccsidを指定した場合は、通常は正しく表示されません。その場合、`-hex` を指定すると文字列のパラメータ部分を16進表記で表示できます。

MQCFT_STRING: スtring・パラメーター

MQCFT_STRING_FILTER: スtring・フィルター・パラメーター

MQCFT_STRING_LIST: スtring・リスト・パラメーター

```
#-----
# MQMD
#-----
# Version <1 or 2>
#MD_VERSION=1
MD_VERSION=2

# MsgType <REQUEST or REPLY or DATAGRAM or REPORT>
#MD_MSGTYPE=REQUEST
#MD_MSGTYPE=REPLY
MD_MSGTYPE=DATAGRAM
#MD_MSGTYPE=REPORT

# Format <ADMIN or EVENT or PCF>
#MD_FORMAT=ADMIN
MD_FORMAT=EVENT
#MD_FORMAT=PCF

# ReplyToQ
MD_REPLYTOQ=<応答先キュー名>

# ReplyToQMgr
MD_REPLYTOQMGR=<応答先キューマネージャー名>

#-----
# MQCFH
# PCF hedder
```

```

#-----
#Type <COMMAND or COMMAND_XR or RESPONSE or XR_MSG or XR_ITEM or XR_SUMMARY or U
SER or NONE or EVENT or TRACE_ROUTE or REPORT or GROUP or STATISTICS or ACCOUNTI
NG or APP_ACTIVITY>
#MQCFH_TYPE=COMMAND
#MQCFH_TYPE=COMMAND_XR
#MQCFH_TYPE=RESPONSE
#MQCFH_TYPE=XR_MSG
#MQCFH_TYPE=XR_ITEM
#MQCFH_TYPE=XR_SUMMARY
MQCFH_TYPE=USER
#MQCFH_TYPE=NONE
#MQCFH_TYPE=EVENT
#MQCFH_TYPE=TRACE_ROUTE
#MQCFH_TYPE=REPORT
#MQCFH_TYPE=GROUP
#MQCFH_TYPE=STATISTICS
#MQCFH_TYPE=ACCOUNTING
#MQCFH_TYPE=APP_ACTIVITY

# Version <MQCFH_VERSION_1 or MQCFH_VERSION_2 or MQCFH_VERSION_3 or MQCFH_CURREN
T_VERSION>
MQCFH_VERSION=MQCFH_VERSION_1
MQCFH_VERSION=MQCFH_VERSION_2
MQCFH_VERSION=MQCFH_VERSION_3
MQCFH_VERSION=MQCFH_CURRENT_VERSION

# Command <MGR or PERFM or CHANNEL> or <command number>
# イベント・メッセージの場合は下記を指定可能
#MQCFH_COMMAND=MGR
#MQCFH_COMMAND=PERFM
#MQCFH_COMMAND=CHANNEL
#
# それ以外は直接コマンドの番号を指定します。
# 下記例の 28 は start channel
# $ grep MQCMD_START_CHANNEL /usr/mqm/inc/cmqcfc.h
# #define MQCMD_START_CHANNEL          28
MQCFH_COMMAND=28

# MsgSeqNumber <1 or 2>
MQCFH_MSGSEQNUM=1
#MQCFH_MSGSEQNUM=2

```

```

#Control
MQCFH_CTRL=MQCFC_LAST
#MQCFH_CTRL=MQCFC_NOT_LAST

# CompCode <OK or WARNING or FAILED>
MQCFH_COMPCODE=OK
#MQCFH_COMPCODE=WARNING
#MQCFH_COMPCODE=FAILED

# Reason < number >
MQCFH_REASON=0

# ParameterCount( for MQCFGR( MQCFT_GROUP ) )
# MQCFGRを指定する場合は、その数を明示的にMQCFHのParameterCountに指定します。MQC
FGRを使用しない場合は、このパラメータを省略できます。省略した場合は、mqpgfによっ
て後続のパラメータの数が自動的に設定されます。

MQCFH_PARAMCOUNT=2

#-----
# MQCFST/MQCFIN etc
# PCF data
#-----
# MQCFGR Structure - PCF グループ・パラメーター
# MQCFGR=Parameter, ParameterCount
#
# MQCFBS - PCF バイト・ストリング・パラメーター
# MQCFBS=Parameter, StringLength, String
#
# MQCFBF - PCF バイト・ストリング・フィルター・パラメーター
# MQCFBF=Parameter, Operator, FilterValueLength, FilterValue
#
# MQCFST - PCF ストリング・パラメーター
# MQCFST=Parameter, CodedCharSetId, StringLength, String
#
# MQCFSF - PCF ストリング・フィルター・パラメーター
# MQCFSF=Parameter, Operator, CodedCharSetId, FilterValueLength, FilterValue
#
# MQCFSL - PCF ストリング・リスト・パラメーター
# MQCFSL=Parameter, CodedCharSetId, Count, StringLength, String1, String2, ...
#

```

```

# MQCFIN - PCF 整数パラメーター
# MQCFIN=Parameter, Value
#
# MQCFIF - PCF 整数フィルター・パラメーター
# MQCFIF=Parameter, Operator, FilterValue
#
# MQCFIL - PCF 整数リスト・パラメーター
# MQCFIL=Parameter, Count, Value1, Value2, ...
#
# MQCFIN64 - 64 ビット整数パラメーター
# MQCFIN64=Parameter, Value
#
# MQCFIL64 - 64 ビット整数リスト・パラメーター
# MQCFIL64=Parameter, Count, Value1, Value2, ...
#-----
#
# <パラメータの各フィールドの説明>
#
# Parameter : パラメーターID
#
# 例) MQCACH_CHANNEL_NAME を指定したい場合は、3501 を指定する。
#
# $ grep MQCMD_START_CHANNEL /usr/mqm/inc/cmqcfc.h.h
# #define MQCACH_CHANNEL_NAME          3501
#
# ParameterCount : MQCFGR(MQCFT_GROUP)に含まれるPCFパラメーターの数
# StringLength : スtringの長さ
# String : スtring
# Operator : MQCFOP_*に対応する数値を指定する。
#
# Operator:
# MQCFOP_LESS          1
# MQCFOP_EQUAL         2
# MQCFOP_NOT_GREATER  3
# MQCFOP_GREATER      4
# MQCFOP_NOT_EQUAL    5
# MQCFOP_NOT_LESS     6
#
# FilterValueLength : フィルター・バリュウの長さ
# FilterValue : フィルター・バリュウ
# CodedCharSetId : スtringのCCSID
# Count : リストの数

```

```
# Value : 整数値
# MQCFIN, MQCFIF, MQCFILのValue/FilterValueに指定可能な範囲
# 0x00000000 - 0xffffffff
# -2147483648 - (+)2147483647
# MQCFIN64, MQCFIL64のValueに指定可能な範囲
# (最上位ビットはセットできない= 16進表記ではマイナスは指定できない)
# 0x00000000 00000000 - 0x7fffffff ffffffff
# -9223372036854775808 - (+)9223372036854775807
```

```
mcpqgf -qm <qmgr> -q <queue> -pcf <pcf format file>
```

-pcf: PCFフォーマット定義ファイル

Ex. 4.15.1 ユーザー定義のフォーマットのPCFメッセージを作成しPUT

```
-----
$ cat sample1.def
MD_VERSION=2

MD_MSGTYPE=DATAGRAM

MD_FORMAT=EVENT

MD_REPLYTOQ=PCF.ANSWER
MD_REPLYTOQMGR=TESTQM

MQCFH_TYPE=USER

MQCFH_VERSION=MQCFH_VERSION_3

MQCFH_COMMAND=99

MQCFH_MSGSEQNUM=1

MQCFH_CTRL=MQCFC_LAST

MQCFH_COMPCODE=OK

MQCFH_REASON=0

MQCFBS=1111,10,1234567890
```



```

MQCFBF=2222, 1, 5, 0x3141324233
MQCFST=3501, 943, 17, TESTQM. to. TESTQM2
MQCFSF=4444, 2, 1208, 30, 123456789012345678901234567890
MQCFSL=5555, 930, 5, 3, 0xf1f1f1, 0xf1f2f3, 0xf3f3f3, 0xf5f6f7, 0xf5f5f5
MQCFIN=6666, 1234567890
MQCFIF=7777, 6, -3
MQCFIL=8888, 3, 1234, 0xffffffff, 5678
MQCFIN64=9999, 0x7fffffffffffffff
MQCFIL64=1234, 5, 4294967294, 0x00000000fffffffb, -5, 0x7fffffffffffffff, 4294967290

```

```

$ mqpgf -qm HM8E2 -q LQ1 -pcf sample1.def
Command : 99
Id : 1111, MQCFT_BYTE_STRING : 10, 1234567890
Id : 2222, MQCFT_BYTE_STRING_FILTER : 1 5 1A2B3
Id : 3501, MQCFT_STRING : 943 17 TESTQM. to. TESTQM2
Id : 4444, MQCFT_STRING_FILTER : 2 1208 30 123456789012345678901234567890
Id : 5555, MQCFT_STRING_LIST : 930 5 3 [...], [...], [...], [...], [...]
Id : 6666, MQCFT_INTEGER : 1234567890
Id : 7777, MQCFT_INTEGER_FILTER : 6 -3
Id : 8888, MQCFT_INTEGER_LIST : 3 [1234], [-1], [5678]
Id : 9999, MQCFT_INTEGER64 : 9223372036854775807
Id : 1234, MQCFT_INTEGER64_LIST : 5 [4294967294], [4294967291], [-5], [922337203685
4775803], [4294967290]
[20/03/06 09:24:05.389505] 1: put from sample1.def
Elapsed time = 0.109636 sec

```

※この例では、MQCFT_STRING_LISTで日本語EBCDIC(ccsid 930)を指定している為、文字列の部分が正しく表示されていません。-hex を指定すると、下記のように16進表記で表示されます。

```

$ mqpgf -qm HM8E2 -q LQ1 -pcf sample1.def -hex
Command : 99
Id : 1111, MQCFT_BYTE_STRING : 10, 1234567890
Id : 2222, MQCFT_BYTE_STRING_FILTER : 1 5 1A2B3
Id : 3501, MQCFT_STRING : 943 17 0x54455354514D2E746F2E54455354514D32
Id : 4444, MQCFT_STRING_FILTER : 2 1208 30 0x3132333435363738393031323334353637
38393031323334353637383930
Id : 5555, MQCFT_STRING_LIST : 930 5 3 [0xF1F1F1], [0xF1F2F3], [0xF3F3F3], [0xF5F6
F7], [0xF5F5F5]
Id : 6666, MQCFT_INTEGER : 1234567890
Id : 7777, MQCFT_INTEGER_FILTER : 6 -3

```



```
(MQCFT_INTEGER_LIST) Type[5] StrucLength[28] Parameter[8888] Count[3] Values[123
4, -1, 5678]
(MQCFT_INTEGER64) Type[23] StrucLength[24] Parameter[9999] Value[922337203685477
5807]
(MQCFT_INTEGER64_LIST) Type[25] StrucLength[56] Parameter[1234] Count[5] Values
[4294967294, 4294967291, -5, 9223372036854775803, 4294967290]
```

Elapsed time = 0.075103 sec

※次は、MQCFGR (MQCFT_GROUP : PCF Group Parameter) を使用した例です。

```
$ cat sample1_2.def
MD_VERSION=2
MD_MSGTYPE=DATAGRAM
MD_FORMAT=EVENT
```

```
MQCFH_TYPE=EVENT
MQCFH_VERSION=MQCFH_VERSION_3
MQCFH_COMMAND=99
```

```
MQCFH_MSGSEQNUM=1
```

```
MQCFH_CTRL=MQCFC_LAST
```

```
MQCFH_COMPCODE=OK
```

```
MQCFH_REASON=2413
```

```
# ParameterCount( for MQCFGR( MQCFT_GROUP ) )
MQCFH_PARAMCOUNT=2
```

```
#-----
```

```
# Command Event
```

```
#-----
```

```
#           1           2           3           4           5           6
#           123456789012345678901234567890123456789012345678901234567890
```

```
MQCFGR=8001, 2
```

```
MQCFST=3045, 819, 12, mqm
```

```
MQCFIN=1011, 3
```

```
MQCFGR=8002, 1
```

```
MQCFIN=99, 1
```

```
$
```

```
$ mqpgf -qm PLQMTS -q LQ1 -pcf sample1_2.def
```

```

Command : 99
Id : 8001, MQCFT_GROUP : 2
Id : 3045, MQCFT_STRING : 819 12 mqm
Id : 1011, MQCFT_INTEGER : 3
Id : 8002, MQCFT_GROUP : 1
Id : 99, MQCFT_INTEGER : 1
[20/11/05 19:55:53.088531] 1: put from sample1_2.def
MQCMIT success : CompCd=00 ReasonCd=00
Elapsed time = 0.081116 sec

```

```

$ mqpgef -qm PLQMTS -q LQ1 -dpv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQEVENT ] Priority[0] Persistence[0] MsgId[0x414D
5120504C514D54532020202020205FA3D34120002504] CorrelId[0x00000000000000000000
000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[PLQMTS
] UserIdentifier[mqm ] AccountingToken[0x05343430333100000000000000000000
000000000000000000000000000000000006] ApplIdentityData[
] PutApplType[13] PutApplName[mqpgef ] PutDate[202011
05] PutTime[10555308] ApplOriginData[ ]

```

```

GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

```

```

*MQCFH(MQCFT_EVENT) Type[7] StrucLength[36] Version[3] Command[99] MsgSeqNumber
[1] Control[1] CompCode[0] Reason[2413] ParameterCount[2]
(MQCFT_GROUP) Type[20] StrucLength[16] Parameter[8001] ParameterCount[2]
(MQCFT_STRING) Type[4] StrucLength[32] Parameter[3045] CodedCharSetId[819] Strin
gLength[12] String[mqm ]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[1011] Value[3]
(MQCFT_GROUP) Type[20] StrucLength[16] Parameter[8002] ParameterCount[1]
(MQCFT_INTEGER) Type[3] StrucLength[16] Parameter[99] Value[1]

```

```

MQCMIT success : CompCd=00 ReasonCd=00
Elapsed time = 0.076464 sec
-----

```

Ex. 4.15.2 コマンド・サーバーに start channel コマンドを送信

※コマンド・サーバーにPCFを送信する場合は、コマンド・サーバーの応答用に MD_REPLY

TOQMGR と MD_REPLYTOQ を指定し、キューも定義しておきます。

```
-----
$ cat sample2.def
MD_VERSION=2

MD_MSGTYPE=REQUEST

MD_FORMAT=ADMIN

MD_REPLYTOQ=PCF.ANSWER
MD_REPLYTOQMGR=HM8E2

MQCFH_TYPE=COMMAND

MQCFH_VERSION=MQCFH_VERSION_1

# start channel
MQCFH_COMMAND=28

MQCFH_MSGSEQNUM=1

MQCFH_CTRL=MQCFC_LAST

MQCFH_COMPCODE=OK

MQCFH_REASON=0

MQCFST=3501, 943, 8, TO.HM8M1

$ mqpcf chs -qm HM8E2 -c TO.HM8M1 STATUS
1: CHLINSTYPE(CURRENT) CHANNEL(TO.HM8M1) STATUS(STOPPED) CHLTYPE(CLUSSDR) CONNAM
E(remotehost(1414)) RQMNAME(HM8M1) STOPREQ(NO) SUBSTATE(OTHER) XMITQ(SYSTEM.CLUS
TER.TRANSMIT.QUEUE)

$
$ mqpgf -qm HM8E2 -q SYSTEM.ADMIN.COMMAND.QUEUE -pcf sample2.def
Command : 28
Id : 3501, MQCFT_STRING : 943 8 TO.HM8M1
[20/03/06 10:21:37.857249] 1: put from sample2.def
Elapsed time = 0.055745 sec

$ mqpcf chs -qm HM8E2 -c TO.HM8M1 STATUS
```

```
1: CHLINSTTYPE(CURRENT) CHANNEL(TO.HM8M1) STATUS(RUNNING) CHLTYPE(CLUSDDR) CONNAM
E(remotehost(1414)) RQMNAME(HM8M1) STOPREQ(NO) SUBSTATE(MQGET) XMITQ(SYSTEM.CLUS
TER.TRANSMIT.QUEUE)
```

※ 下記はコマンドサーバーは返した start channel コマンドの応答メッセージです。

\$ mqp gf -qm HM8E2 -q PCF.ANSWER -brv

message number: 1

```
*StrucId[MD  ] Version[2] Report[0] MsgType[2] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQADMIN ] Priority[0] Persistence[0] MsgId[0x414D
5120484D38453220202020202020205E61952120001E12] CorrelId[0x414D5120484D38453220202
0202020205E61952120003002] BackoutCount[0] ReplyToQ[
```

] ReplyToQMgr[HM8E2

```
] UserIdentifier[mqm] AccountingToken[0x05343430333100000000000000000000  
000000000000000000000000000006] ApplIdentityData[
```

```

] PutApplType[7] PutApplName[amqpcsea] PutDate[2012030
6] PutTime[01213802] ApplOriginData[ ]

```

```
GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]
```

```
*MQCFH(MQCFT_RESPONSE) Type[2] StrucLength[36] Version[1] Command[28] MsgSeqNum
ber[1] Control[1] CompCode[0] Reason[0] ParameterCount[0]
```

Elapsed time = 0.061344 sec

4.16 MQSET呼び出し

指定キューに対し、MQSET()を呼び出します。複数のセレクターとアトリビュートの組み合わせを指定することができます。

MQSETは、キューに対してのみ属性の変更が可能です。プロセス、キューマネージャなど、他のオブジェクトの属性は変更できません。

また、モデルキューは変更できず、クラスター・キューの場合はローカル・インスタンスが必要です。

```
mqpgf -qm <qmgr> -q <queue> -set: selector:attribute,...:., (e.g. MQIA_INHIBIT_GET:MQQA_GET_ALLOWED,...:.)
```

-set: セレクター

下表のセレクターが使用可能です。

表 4.16.1 キューに関する MQSET 属性セレクター	
セレクター	説明(設定可能な値)
MQCA_TRIGGER_DATA	トリガー・データ (最大 MQ_TRIGGER_DATA_LENGTH(64) までの文字列 ※削除する時は、シングルのクォートでスペース文字を指定します。: ' ' または" ")
MQIA_DIST_LISTS	配布リストのサポート。 (MQDL_SUPPORTED/ MQDL_NOT_SUPPORTED)
MQIA_INHIBIT_GET	読み取り操作が許されているかどうかを判別します。 (MQQA_GET_INHIBITED/MQQA_GET_ALLOWED)
MQIA_INHIBIT_PUT	書き込み操作が許されているかどうかを判別します。 (MQQA_PUT_INHIBITED/MQQA_PUT_ALLOWED)
MQIA_TRIGGER_CONTROL	トリガー制御。 (MQQA_PUT_INHIBITED/MQQA_PUT_ALLOWED)
MQIA_TRIGGER_DEPTH	トリガー項目数。 (MQTC_ON/MQTC_OFF)
MQIA_TRIGGER_MSG_PRIORITY	トリガーに対するしきい値メッセージ優先度。 (1 以上の整数)
MQIA_TRIGGER_TYPE	トリガー・タイプ。 (MQTT_NONE/ MQTT_FIRST/ MQTT_EVERY/ MQTT_DEPTH)

Ex. 4.16.1 MQSETに単独のパラメータを指定する例

```
-----  
$ mqpgf -qm HM8E2 -q LQ1 -set MQIA_INHIBIT_GET:MQQA_GET_INHIBITED  
[20/03/06 11:05:57.627602] 1: MQSET MQIA_INHIBIT_GET:MQQA_GET_INHIBITED  
Elapsed time = 0.308728 sec
```

```
$ mqpcf que -qm HM8E2 -q LQ1 GET  
1: QUEUE(LQ1) TYPE(QLOCAL) GET(DISABLED)  
-----
```

Ex. 4.16.2 MQSETに複数のパラメータを指定する例

```
-----  
$ mqpcf que -qm HM8E2 -q LQ1 GET PUT TRIGDATA DISTL  
1: QUEUE(LQ1) TYPE(QLOCAL) DISTL(NO) GET(DISABLED) PUT(ENABLED) TRIGDATA(trigger  
data)
```

```
$ mqpgf -qm HM8E2 -q LQ1 -set MQIA_INHIBIT_GET:MQQA_GET_ALLOWED, MQIA_INHIBIT_PUT:MQQA_PUT_ALLOWED, MQCA_TRIGGER_DATA:" ", MQIA_DIST_LISTS:MQDL_SUPPORTED  
[20/03/06 11:15:36.565387] 1: MQSET MQIA_INHIBIT_GET:MQQA_GET_ALLOWED, MQIA_INHIBIT_PUT:MQQA_PUT_ALLOWED, MQCA_TRIGGER_DATA: , MQIA_DIST_LISTS:MQDL_SUPPORTED  
Elapsed time = 0.089051 sec
```

```
$ mqpcf que -qm HM8E2 -q LQ1 GET PUT TRIGDATA DISTL  
1: QUEUE(LQ1) TYPE(QLOCAL) DISTL(YES) GET(ENABLED) PUT(ENABLED) TRIGDATA()  
-----
```


4.17 MQINQ呼び出し

MQINQ() を呼び出し、指定キュー（ローカル、リモート、エイリアス）、ネームリスト、プロセス、キューマネージャーの属性を照会することができます。

```
mqpgf -qm <qmgr> -inq: selector(e.g. MQCA_CHANNEL_AUTO_DEF_EXIT, MQCA_CLUSTER_WORKLOAD_DATA,...) MQOT_Q_MGR
mqpgf -qm <qmgr> -q <queue> -inq: selector(e.g. MQCA_ALTERATION_DATE, MQIA_CLWL_Q_PRIORITY,...)
mqpgf -qm <qmgr> -nl <namelist> -inq: selector(e.g. MQIA_NAMELIST_TYPE, M MQCA_NAMES,...) MQOT_NAMELIST
mqpgf -qm <qmgr> -p <process> -inq: selector(e.g. MQCA_APPL_ID, MQCA_ENV_DATA,...) MQOT_PROCESS
```

-q: キュー名 ※キューの属性を照会する場合に必須
 -nl: ネームリスト名 ※ネームリストの属性を照会する場合に必須
 -p: プロセス名 ※プロセスの属性を照会する場合に必須
 -inq: セレクター
 MQOT_Q_MGR ※キューマネージャーの属性を照会する場合に必須
 MQOT_NAMELIST ※ネームリストの属性を照会する場合に必須
 MQOT_PROCESS ※プロセスの属性を照会する場合に必須

下表のセレクターが使用可能です。

表 4.17.1 キューに関する MQINQ 属性セレクター		
セレクター	説明	注記
MQCA_ALTERATION_DATE	最後の変更の日付	
MQCA_ALTERATION_TIME	最後の変更の時刻	
MQCA_BACKOUT_REQ_Q_NAME	超過バックアウト再キューイング用のキュー名	
MQCA_BASE_Q_NAME	別名解決後のキューの名前	
MQCA_CF_STRUC_NAME	カップリング・ファシリティ構造体の名前	z/OS
MQCA_CLUS_CHL_NAME	このキューを伝送キューとして使用するクラスター送信側チャネルの名前。	
MQCA_CLUSTER_NAME	クラスター名	
MQCA_CLUSTER_NAMELIST	クラスター名前リスト	

表 4.17.1 キューに関する MQINQ 属性セレクトター

セレクトター	説明	注記
MQCA_CREATION_DATE	キュー作成日	
MQCA_CREATION_TIME	キュー作成時刻	
MQCA_INITIATION_Q_NAME	開始キュー名	
MQCA_PROCESS_NAME	プロセス定義の名前	
MQCA_Q_DESC	キューの記述	
MQCA_Q_NAME	キュー名	
MQCA_REMOTE_Q_MGR_NAME	リモート・キュー・マネージャーの名前	
MQCA_REMOTE_Q_NAME	リモート・キュー・マネージャー上で認識されているリモート・キューの名前	
MQCA_STORAGE_CLASS	ストレージ・クラスの名前	z/OS
MQCA_TRIGGER_DATA	トリガー・データ	
MQCA_XMIT_Q_NAME	伝送キュー名	
MQIA_ACCOUNTING_Q	キューのアカウンティング・データのコレクションを制御する	z/OS 以外
MQIA_BACKOUT_THRESHOLD	バックアウトしきい値	
MQIA_CLWL_Q_PRIORITY	キューの優先順位	
MQIA_CLWL_Q_RANK	キューのランク	
MQIA_CLWL_USEQ	リモート・キューを使用する	
MQIA_CURRENT_Q_DEPTH	キュー上のメッセージの数	
MQIA_DEF_BIND	デフォルトのバインディング	
MQIA_DEF_INPUT_OPEN_OPTION	デフォルトの入力用オープンオプション	
MQIA_DEF_PERSISTENCE	デフォルトのメッセージ持続性	
MQIA_DEF_PRIORITY	デフォルトのメッセージ優先度	
MQIA_DEFINITION_TYPE	キュー定義タイプ	
MQIA_DIST_LISTS	配布リスト・サポート	z/OS 以外

表 4.17.1 キューに関する MQINQ 属性セクター

セクター	説明	注記
MQIA_HARDEN_GET_BACKOUT	バックアウト・カウントをハード化するかどうか	
MQIA_INDEX_TYPE	キュー用に保守される索引のタイプ	z/OS
MQIA_INHIBIT_GET	取得操作が許可されるかどうか	
MQIA_INHIBIT_PUT	PUT 操作が許可されるかどうか	
MQIA_MAX_MSG_LENGTH	最大メッセージ長	
MQIA_MAX_Q_DEPTH	キューに許可されるメッセージの最大数	
MQIA_MSG_DELIVERY_SEQUENCE	メッセージ優先順位が考慮されるかどうか	
MQIA_NPM_CLASS	非持続メッセージの信頼性レベル	
MQIA_OPEN_INPUT_COUNT	キューを入力用にオープンする MQOPEN 呼び出しの数	
MQIA_OPEN_OUTPUT_COUNT	キューを出力用にオープンする MQOPEN 呼び出しの数	
MQIA_PROPERTY_CONTROL	プロパティ制御属性	
MQIA_Q_DEPTH_HIGH_EVENT	キュー・サイズ上限イベントの制御属性	z/OS 以外
MQIA_Q_DEPTH_HIGH_LIMIT	キュー・サイズの上限	z/OS 以外
MQIA_Q_DEPTH_LOW_EVENT	キュー・サイズ下限イベントの制御属性	z/OS 以外
MQIA_Q_DEPTH_LOW_LIMIT	キュー・サイズの下限	z/OS 以外
MQIA_Q_DEPTH_MAX_EVENT	キュー・サイズ最大イベントの制御属性	z/OS 以外
MQIA_Q_SERVICE_INTERVAL	キュー・サービス間隔の限度	z/OS 以外
MQIA_Q_SERVICE_INTERVAL_EVENT	キュー・サービス間隔イベントの制御属性	z/OS 以外
MQIA_Q_TYPE	キュー・タイプ	

表 4.17.1 キューに関する MQINQ 属性セクター

セクター	説明	注記
MQIA_QSG_DISP	キュー共有グループ後処理	z/OS
MQIA_RETENTION_INTERVAL	キュー保持期間間隔	
MQIA_SCOPE	キュー定義の有効範囲	z/OS 以外
MQIA_SHAREABILITY	入力のためのキューを共有できるかどうか	
MQIA_STATISTICS_Q	キューの統計データのコレクションを制御する	z/OS 以外
MQIA_TRIGGER_CONTROL	トリガー制御	
MQIA_TRIGGER_DEPTH	トリガー項目数	
MQIA_TRIGGER_MSG_PRIORITY	トリガーに対するしきい値メッセージ優先度	
MQIA_TRIGGER_TYPE	トリガー・タイプ	
MQIA_USAGE	使用法	

表 4.17.2 名前リストに関する MQINQ 属性セクター

セクター	説明	注記
MQCA_ALTERATION_DATE	最後の変更の日付	
MQCA_ALTERATION_TIME	最後の変更の時刻	
MQCA_NAMELIST_DESC	名前リストの記述	
MQCA_NAMELIST_NAME	名前リスト・オブジェクトの名前。	
MQIA_NAMELIST_TYPE	名前リストのタイプ	z/OS
MQCA_NAMES	名前リストにある名前。	
MQIA_NAME_COUNT	名前リスト内の名前の数	
MQIA_QSG_DISP	キュー共有グループ後処理	z/OS

表 4.17.3 プロセス定義に関する MQINQ 属性セクター

セクター	説明	注記
MQCA_ALTERATION_DATE	最後の変更の日付	
MQCA_ALTERATION_TIME	最後の変更の時刻	
MQCA_APPL_ID	アプリケーション ID	
MQCA_ENV_DATA	環境データ	
MQCA_PROCESS_DESC	プロセス定義の記述。	
MQCA_PROCESS_NAME	プロセス定義の名前	
MQCA_USER_DATA	ユーザー・データ	
MQIA_APPL_TYPE	アプリケーション・タイプ	
MQIA_QSG_DISP	キュー共有グループ後処理	z/OS

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セクター

セクター	説明	注記
MQCA_ALTERATION_DATE	最後の変更の日付	
MQCA_ALTERATION_TIME	最後の変更の時刻	
MQCA_CHANNEL_AUTO_DEF_EXIT	自動チャネル定義出口名。	
MQCA_CHINIT_SERVICE_PARM	IBM の使用のため予約済み	
MQCA_CLUSTER_WORKLOAD_DATA	クラスター・ワークロード出口に渡されるデータ。	
MQCA_CLUSTER_WORKLOAD_EXIT	クラスター・ワークロード出口名	
MQCA_COMMAND_INPUT_Q_NAME	システム・コマンド入力キュー名。	
MQCA_DEAD_LETTER_Q_NAME	送達不能キューの名前。	
MQCA_DEF_XMIT_Q_NAME	デフォルトの伝送キューの名前。	
MQCA_DNS_GROUP	キュー共有グループのインバウンド伝送を処理する TCP リスナーが参加するグループの名前。この名前は、Workload Manager Dynamic Domain Name Services を使用するとき適用されます。	z/OS

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セレクトター

セレクトター	説明	注記
MQCA_IGQ_USER_ID	グループ内キューイングのユーザー ID。	z/OS
MQCA_INSTALLATION_DESC	関連するインストール済み環境の記述。	z/OS 以外 IBMi 以外
MQCA_INSTALLATION_NAME	キュー・マネージャーに関連付けられたインストールの名前。	z/OS 以外 IBMi 以外
MQCA_INSTALLATION_PATH	関連する IBM MQ がインストールされる場所のパス	z/OS 以外 IBMi 以外
MQCA_LU_GROUP_NAME	使用するキュー共有グループのインバウンド送信を処理する LU 6.2 リスナーの総称 LU 名。	z/OS
MQCA_LU_NAME	アウトバウンド LU 6.2 伝送で使用する LU の名前。 この名前をリスナーがインバウンド伝送で使用するのと同じ LU に設定します。	z/OS
MQCA_LU62_ARM_SUFFIX	SYS1.PARMLIB メンバー APPCPM _{xx} のサフィックス。 このチャネル開始プログラムとして LUADD を指定します。	z/OS
MQCA_PARENT	このキュー・マネージャーの親として候補に挙げられた、階層的に接続されたキュー・マネージャーの名前。	
MQCA_Q_MGR_DESC	キュー・マネージャーの記述。	
MQCA_Q_MGR_IDENTIFIER	キュー・マネージャー ID (H)	
MQCA_Q_MGR_NAME	ローカル・キュー・マネージャーの名前	
MQCA_QSG_NAME	キュー共有グループ名	z/OS
MQCA_REPOSITORY_NAME	キュー・マネージャーがリポジトリ・サービスを提供しているクラスターの名前。	

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セクター

セクター	説明	注記
MQCA_REPOSITORY_NAMELIST	キュー・マネージャーがリポジトリ・サービスを提供しているクラスターの名前を含む名前リスト・オブジェクトの名前。	
MQCA_TCP_NAME	使用している TCP/IP システムの名前	z/OS
MQIA_ACCOUNTING_CONN_OVERWRITE	アカウンティング設定をオーバーライドする	z/OS 以外
MQIA_ACCOUNTING_INTERVAL	中間アカウント・レコードを書き込む頻度	z/OS 以外
MQIA_ACCOUNTING_MQI	MQI データに関するアカウンティング情報の収集を制御します。	z/OS 以外
MQIA_ACCOUNTING_Q	キューのアカウンティング情報のコレクションを制御する	z/OS 以外
MQIA_ACTIVE_CHANNELS	いつでもアクティブにできるチャンネルの最大数	z/OS
MQIA_ADOPTNEWMCA_CHECK	MCA を受け入れるかどうか判別するためにチェックするエレメント。このチェックは、既にアクティブになっている MCA と同じ名前のインバウンド・チャンネルが新たに検出されたときに行われます。	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	孤立したチャンネルが終了するまで新しいチャンネルが待機する時間（秒）	z/OS 以外
MQIA_ADOPTNEWMCA_TYPE	AdoptNewMCACheck パラメーターと一致する新しいインバウンド・チャンネル要求が検出された場合に、指定されたチャンネル・タイプの MCA の孤立したインスタンスを自動的に再始動するかどうか	z/OS
MQIA_AUTHORITY_EVENT	権限イベントの制御属性	z/OS 以外
MQIA_BRIDGE_EVENT	IMS ブリッジ・イベントの制御属性	z/OS
MQIA_CHANNEL_AUTO_DEF	自動チャンネル定義の制御属性	z/OS 以外

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セレクトター

セレクトター	説明	注記
MQIA_CHANNEL_AUTO_DEF_EVENT	自動チャンネル定義イベントの制御属性	z/OS 以外
MQIA_CHANNEL_EVENT	チャンネル・イベントの制御属性	
MQIA_CHINIT_ADAPTERS	IBM MQ 呼び出しの処理に使用するアダプター・サブタスクの数	z/OS
MQIA_CHINIT_DISPATCHERS	チャンネル・イニシエーターで使用するディスパッチャーの数	z/OS
MQIA_CHINIT_TRACE_AUTO_START	チャンネル・イニシエーター・トレースを自動的に開始するかどうか	z/OS
MQIA_CHINIT_TRACE_TABLE_SIZE	チャンネル・イニシエーターのトレース・データ・スペースのサイズ (MB)	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	クラスター・ワークロードの長さ。	
MQIA_CLWL_MRU_CHANNELS	クラスターのワークロード・バランシング用に最近使用されたチャンネルの数	
MQIA_CLWL_USEQ	リモート・キューを使用する	
MQIA_CODED_CHAR_SET_ID	コード化文字セット ID	
MQIA_COMMAND_EVENT	コマンド・イベントの制御属性	
MQIA_COMMAND_LEVEL	キュー・マネージャーでサポートされるコマンド・レベル	
MQIA_CONFIGURATION_EVENT	構成イベントの制御属性	z/OS 以外
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	クラスター送信側チャンネルに使用するデフォルトの伝送キュー・タイプ。	
MQIA_DIST_LISTS	配布リスト・サポート	z/OS 以外
MQIA_DNS_WLM	キュー共有グループのインバウンド伝送を処理する TCP リスナーを Workload Manager for Dynamic Domain Name Services に登録するかどうか	z/OS

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セクター

セクター	説明	注記
MQIA_EXPIRY_INTERVAL	有効期限切れメッセージのスキャンを実行する間隔	z/OS
MQIA_GROUP_UR	このキュー・マネージャーで GROUP リカバリー単位を有効にするかどうかを指定する制御属性。GROUP リカバリー単位処理は、キュー・マネージャーがキュー共有グループのメンバーである場合にのみ使用可能です。	z/OS
MQIA_IGQ_PUT_AUTHORITY	グループ内キューイング書き込み権限	z/OS
MQIA_INHIBIT_EVENT	禁止イベントの制御属性	z/OS 以外
MQIA_INTRA_GROUP_QUEUING	グループ内キューイングのサポート	z/OS
MQIA_LISTENER_TIMER	APPC または TCP/IP が失敗した場合に、IBM MQ がリスナーの再開を試行するまでの時間間隔 (秒)	z/OS
MQIA_LOCAL_EVENT	ローカル・イベントの制御属性	z/OS 以外
MQIA_LOGGER_EVENT	禁止イベントの制御属性	z/OS 以外
MQIA_LU62_CHANNELS	LU 6.2 伝送プロトコルを使用して、現行チャネルとして可能なチャネル、または接続できるクライアントの最大数	z/OS
MQIA_MSG_MARK_BROWSE_INTERVAL	キュー・マネージャーがブラウズ・メッセージからマークを自動的に除去する時間間隔 (ミリ秒)	
MQIA_MAX_CHANNELS	現行チャネルとして可能な最大チャネル数 (接続されたクライアントを使用したサーバー接続チャネルを含む)	z/OS
MQIA_MAX_HANDLES	ハンドルの最大数	
MQIA_MAX_MSG_LENGTH	最大メッセージ長	
MQIA_MAX_PRIORITY	最高の優先順位	

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セレクトター

セレクトター	説明	注記
MQIA_MAX_UNCOMMITTED_MSGS	1 つの作業単位内のコミットされていないメッセージの最大数	
MQIA_OUTBOUND_PORT_MAX	MQIA_OUTBOUND_PORT_MIN を使用して、出力チャネルをバインディングするときに使用するポート番号の範囲を定義する	z/OS
MQIA_OUTBOUND_PORT_MIN	MQIA_OUTBOUND_PORT_MAX を使用して、出力チャネルをバインディングするときに使用するポート番号の範囲を定義する	z/OS
MQIA_PERFORMANCE_EVENT	パフォーマンス・イベントの制御属性	z/OS 以外
MQIA_PLATFORM	キュー・マネージャーがあるプラットフォーム	
MQIA_PROT_POLICY_CAPABILITY	キュー・マネージャーで IBM MQ Advanced Message Security のセキュリティー機能が使用可能かどうかを示します。	
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	同期点における、失敗したコマンド・メッセージの再処理の試行回数。	
MQIA_PUBSUB_MODE	パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースが実行されているかどうかアプリケーション・プログラミング・インターフェースを使用してパブリッシュまたはサブスクライブを行うアプリケーションは、パブリッシュ/サブスクライブ・エンジンが必要です。キュー・パブリッシュ/サブスクライブ・インターフェースがモニターするキューの場合、キュー・パブリッシュ/サブスクライブ・インターフェースが実行されている必要があります。	
MQIA_PUBSUB_NP_MSG	未配信の入力メッセージを廃棄（または保持）するかどうか。	
MQIA_PUBSUB_NP_RESP	未配信の応答メッセージの動作を制御します。	

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セレクター

セレクター	説明	注記
MQIA_PUBSUB_SYNC_PT	持続メッセージのみ（またはすべてのメッセージ）を同期点で処理するかどうかを指定します。	
MQIA_QMGR_CFCONLOS	キュー・マネージャーが管理構造体に対する接続または CFCONLOS が ASQMGR に設定されている CF 構造体に対する接続を失ったときに実行されるアクションを指定します。	z/OS
MQIA_RECEIVE_TIMEOUT	TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機するおおよその時間。値は、MQIA_RECEIVE_TIMEOUT_TYPE で修飾される数値です。	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機する最小時間	z/OS
MQIA_RECEIVE_TIMEOUT_TYPE	TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機するおおよその時間。MQIA_RECEIVE_TIMEOUT_TYPE は、MQIA_RECEIVE_TIMEOUT に適用される修飾子です。	z/OS
MQIA_REMOTE_EVENT	リモート・イベントの制御属性	z/OS 以外
MQIA_SECURITY_CASE	セキュリティー・プロファイルのケース	z/OS
MQIA_SSL_EVENT	チャンネル・イベントの制御属性	
MQIA_SSL_FIPS_REQUIRED	暗号化用に FIPS で証明されているアルゴリズムのみを使用する	
MQIA_SSL_RESET_COUNT	SSL 鍵リセット・カウント	
MQIA_START_STOP_EVENT	開始 / 停止イベントの制御属性	z/OS 以外
MQIA_STATISTICS_AUTO_CLUSSDR	クラスター送信側チャンネルの統計モニター情報のコレクションを制御する	

表 4.17.4 キュー・マネージャーに関する MQINQ 属性セクター		
セクター	説明	注記
MQIA_STATISTICS_CHANNEL	チャンネルの統計データのコレクションを制御する	
MQIA_STATISTICS_INTERVAL	統計モニター・データを書き込む頻度	z/OS 以外
MQIA_STATISTICS_MQI	キュー・マネージャーに関する統計モニター情報の収集を制御します。	z/OS 以外
MQIA_STATISTICS_Q	キューの統計データのコレクションを制御する	z/OS 以外
MQIA_SYNCPOINT	同期点の使用可能性	
MQIA_TCP_CHANNELS	TCP/IP 伝送プロトコルを使用して、現行チャンネルとして可能なチャンネル、または接続できるクライアントの最大数	z/OS
MQIA_TCP_KEEP_ALIVE	接続のもう一方の側が依然として使用可能であることを検査する TCP KEEPALIVE 機能を使用するかどうか	z/OS
MQIA_TCP_STACK_TYPE	チャンネル・イニシエーターが TCPNAME で指定された TCP/IP アドレス・スペースのみを使用できるか、あるいは、任意で選択した TCP/IP アドレスにオプションでバインドできるか	z/OS
MQIA_TRACE_ROUTE_RECORDING	トレース経路指定情報の記録を制御する	z/OS
MQIA_TREE_LIFE_TIME	未使用の非管理トピックの存続時間	
MQIA_TRIGGER_INTERVAL	トリガー間隔	

Ex. 4.17.1 ローカルキューの属性の照会

```

$ mqpgf -qm HM8E2 -q LQ1 -inq MQCA_ALTERATION_DATE, MQCA_ALTERATION_TIME, MQCA_BA
CKOUT_REQ_Q_NAME
[20/03/06 11:29:07.891216] 1: ALTDATE(2012-03-06) ALTTIME(11.28.58) BOQNAME(B04T
Q)
Elapsed time = 0.059579 sec

```

Ex. 4.17.2 リモートキューの属性の照会

```
-----  
$ mqpgf -qm HM8E2 -q RQ1 -inq MQCA_REMOTE_Q_MGR_NAME, MQCA_REMOTE_Q_NAME, MQCA_XMIT_Q_NAME  
[20/03/06 11:39:09.850786] 1: RQMNAME(HM8M1) RNAME(LQ1) XMITQ()  
Elapsed time = 0.060893 sec  
-----
```

Ex. 4.17.3 エイリアスキューの属性の照会

```
-----  
$ mqpgf -qm HM8E2 -q AQ1 -inq MQCA_BASE_Q_NAME  
[20/03/06 11:43:57.045627] 1: TARGET(LQ1)  
Elapsed time = 0.059196 sec  
-----
```

Ex. 4.17.4 ネームリストの属性の照会

```
-----  
$ mqpgf -qm HM8E2 -nl NL1 MQOT_NAMELIST -inq MQCA_ALTERATION_DATE, MQCA_ALTERATION_TIME, MQCA_NAMELIST_DESC, MQCA_NAMELIST_NAME, MQCA_NAMES, MQIA_NAME_COUNT  
[20/03/06 11:59:54.593427] 1: ALTDATE(2012-03-06) ALTTIME(11.58.48) DESCR(sample name list) NAMELIST(NL1) NAMES('NAME1', 'NAME2', 'NAME3') NAMCOUNT(3)  
Elapsed time = 0.060212 sec  
-----
```

*MQOT_NAMELIST(オブジェクト・タイプ)の指定が必要

Ex. 4.17.5 プロセスの属性の照会

```
-----  
>mqpgf -qm HM9S -p SYSTEM.DEFAULT.PROCESS MQOT_PROCESS -inq MQCA_ALTERATION_DATE, MQCA_ALTERATION_TIME, MQCA_APPL_ID, MQCA_ENV_DATA, MQCA_PROCESS_DESC, MQCA_PROCESS_NAME, MQCA_USER_DATA, MQIA_APPL_TYPE  
[2012/03/09 17:03:39.963] 1: ALTDATE(2011-11-26) ALTTIME(15.29.19) APPLICID() ENVRDATA() DESCR() PROCESS(SYSTEM.DEFAULT.PROCESS) USERDATA() APPLTYPE(WINDOWSNT)  
Elapsed time = 101 msec  
-----
```

*MQOT_PROCES(オブジェクト・タイプ)の指定が必要

Ex. 4.17.6 キューマネージャーの属性の照会

```
$ mqpgef -qm TESQM MQOT_Q_MGR -inq MQCA_ALTERATION_DATE, MQCA_ALTERATION_TIME, MQC  
A_CHANNEL_AUTO_DEF_EXIT, MQCA_CLUSTER_WORKLOAD_DATA  
[16/12/20 19:30:48] 1: ALTDATA(2016-12-13) ALTTIME(14.19.03) CHADEXIT() CLWLDATA  
( )
```

*MQOT_Q_MGR(オブジェクト・タイプ)の指定が必要

4.18 メッセージ・プロパティの指定

任意のデータ型のメッセージプロパティを指定してメッセージをPUTすることが可能です。

`mqqpgf -qm <qmgr> -q <queue> -m <message> -smp: Type:Property Name:Property Value`
(e. g. `MQTYPE_STRING:property name:value,...`.)

`-smp:` メッセージ・プロパティ。タイプ:プロパティ名:プロパティ値,... の様に指定します。

MQPMO_VERSION_3: メッセージプロパティを作成する場合は、MQPMO_VERSION_3 を指定することが必要

表 4.18.1 プロパティ値のデータ型

タイプ	プロパティ値	サンプル
MQTYPE_BOOLEAN	TRUE/FAULSE	
MQTYPE_BYTE_STRING	16進表記で指定	01ef
MQTYPE_INT8	8bit整数	-128 - 127
MQTYPE_INT16	16bit整数	-32,768 - 32,768
MQTYPE_INT32	32bit整数	-2,147,483,648 - 2,147,483,647
MQTYPE_INT64	64bit整数	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807
MQTYPE_FLOAT32	32bit浮動小数点	-3.402823e+38 - 3.402823e+38
MQTYPE_FLOAT64	64bit浮動小数点	-1.797693e+308 - 1.797693e+308
MQTYPE_STRING	文字列	
MQTYPE_NULL	※MQTYPE_NULLの場合は、Valueは指定できない	

Ex. 4.18.1 メッセージ・プロパティの指定例（数値型に最大値を指定）

\$ `mqqpgf -qm TESTQM -q TQ -m "test" -smp "MQTYPE_BOOLEAN:boolean1:TRUE, MQTYPE_BYTE_STRING:byteString:0102feff, MQTYPE_INT8:int8:127, MQTYPE_INT16:int16:32767, MQT`

data length: 4

00000000: 7465 7374

'test'

4.19 配布リストの使用

オブジェクト・レコードに複数のキューを指定してPUTします。指定したキュー全てに同一のメッセージがPUTされます。指定するキュー毎にオブジェクト・キューマネージャ名を指定できます。さらに、PUTメッセージ・レコードの指定も可能です。PUTメッセージ・レコードを使用することで、MQMDのいくつかのフィールドをPUTするオブジェクト（キュー）宛てのメッセージ毎に設定することが可能です。

```
mqpgf -qm <qmgr> -or <queue1[:qmgr1]>,<queue2[:qmgr2]>,<queue3[:qmgr3]>,... -mr
<msgId>:<correlId>:<groupId>:<feedback>:<accountingtoken>,... -m "distribution
lists" MQOD_VERSION_2 MQPMO_VERSION_2 [MQPMRF_MSG_ID MQPMRF_CORREL_ID MQPMRF_G
ROUP_ID MQPMRF_FEEDBACK MQPMRF_ACCOUNTING_TOKEN MQOO_SET_ALL_CONTEXT MQPMO_SET_
ALL_CONTEXT MQMD_VERSION_2 MQMF_MSG_IN_GROUP]
```

-or: オブジェクト・レコード。キュー名:オブジェクト・キューマネージャ名,... の様に指定します。例えばクラスタキューの場合は、ホストしているキューマネージャ名をオブジェクト・キューマネージャ名として指定します。オブジェクト・キューマネージャ名は省略可能です。

MQOD_VERSION_2: 配布リストを使用する場合は MQOD_VERSION_2 以上の指定が必要

MQPMO_VERSION_2: 配布リストを使用する場合は、MQPMO_VERSION_2以上の指定が必要

(任意のオプション例)

-mr: PUTメッセージ・レコード。PUT先のキュー毎に設定します。’:’で区切って各パラメータを指定します。レコードの区切りは”, ”を使用します。

下表が指定可能なMQMDのフィールドです。省略することは可能ですが、指定順は下表の順を守る必要があります。対応する MQPMRF_ コンスタントの指定が必要です。

表 4.19.1 PUT メッセージ・レコードのフィールド			
指定順	フィールド	MQPMRF_*	その他
1	MsgId	MQPMRF_MSG_ID	
2	CorrelId	MQPMRF_CORREL_ID	
3	GroupId	MQPMRF_GROUP_ID	MQMD_VERSION_2 および MQMF_MSG_IN_GROUP、MQMF_LAST_MSG_IN_GROUP、MQMF_SEGMENT、MQMF_LAST_SEGMENT、MQMF_SEGMENTATION_ALLO

表 4.19.1 PUT メッセージ・レコードのフィールド			
指定順	フィールド	MQPMRF_*	その他
			WED のいずれかの指定が必要
4	Feedback	MQPMRF_FEEDBACK	
5	AccountingToken	MQPMRF_ACCOUNTING_TOKEN	MQ00_SET_ALL_CONTEXT および MQPMO_SET_ALL_CONTEXT または、MQ00_SET_IDENTITY_CONTEXT および MQPMO_SET_IDENTITY_CONTEXT の指定が必要

Ex. 4.19.1 オブジェクト・レコードに複数のローカル・キューを指定してPUT

```

-----
※オブジェクト・レコードに3つのローカル・キューを指定する。
$ mqpgf -qm TESTQM -or INQ1, INQ2, INQ3 -m "distribution lists" MQOD_VERSION_2 MQP
MO_VERSION_2
[16/12/22 20:15:16] 1: message length: 18 put message : distribution lists
$
$ mqpgf -qm TESTQM -q INQ1 -br
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[5
46] CodedCharSetId[943] Format[ ] Priority[0] Persistence[0] MsgId[0x414D
512053545343514D202020202020DA9B365620001E02] CorrelId[0x0000000000000000000000
0000000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[TESTQM ] UserI
dentifier[testuser ] AccountingToken[0160105150000005CB9193C9FEF8154FF9CFF8AE
8030000000000000000000000000B] ApplIdentityData[ ] Pu
tApplType[11] PutApplName[objects¥mqpgf¥Debug¥mqpgf.exe] PutDate[20151101] PutTim
e[23142905] ApplOriginData[ ]

GroupId[0x000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

data length: 18
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973 'distribution lis'
00000010: 7473 'ts',

```

```
$ mqpgrf -qm QMA -or CLUS_Q1:QMA, CLUS_Q1:QMB, CLUS_Q2:QMB -m "distribution lists"
MQPD_VERSION_2 MQPMO VERSION_2
```

§

```
1: QUEUE (CLUS_Q1) TYPE (QUEUE) CURDEPTH (1)
```

\$

```
1: QUEUE(CLUS_Q1) TYPE(QUEUE) CURDEPTH(1)
```

\$

message number: 1

```
] ReplyToQMgr[QMA] UserIdentifier[testuser] AccountingToken[01601051500000005CB9193C9FEF8154FF9CFF8AE80300000000000000000000B] ApplIdentityData[] PutApplType[11] PutApplName[objects\ymqpgf\Debug\ymqpgf.exe] PutDate[20151101] PutTime[23453955] ApplOriginData[]
```

```
data length: 18
```

4-19-3


```
data length: 18
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973 'distribution lis'
00000010: 7473 'ts',
```

....

Ex. 4.19.5 メッセージ・レコード内の省略

※メッセージ・レコードの途中のオブジェクトは省略できません。その場合は、下記のようにデフォルト値を指定しておく必要があります。

```
-mr ...,MQMI_NONE:MQCI_NONE:MQGI_NONE:MQFB_NONE:MQACT_NONE,...
```

```
-----
$ mqpgf -qm QMA -or CLUS_Q1:QMA, CLUS_Q1:QMB, CLUS_Q2:QMB ¥
-mr msgId1:correlId1:groupId1:MQFB_QUIT:account1,¥
MQMI_NONE:MQCI_NONE:MQGI_NONE:MQFB_NONE:MQACT_NONE,¥
0x111111:0x222222:0x333333:MQFB_EXPIRATION:0x444444¥
-m "distribution lists" MQOD_VERSION_2 MQPMO_VERSION_2 MQPMRF_MSG_ID MQPMRF_COR
REL_ID MQPMRF_GROUP_ID MQPMRF_FEEDBACK MQPMRF_ACCOUNTING_TOKEN MQOO_SET_ALL_CONT
EXT MQPMO_SET_ALL_CONTEXT MQMD_VERSION_2 MQMF_MSG_IN_GROUP
[16/12/22 20:22:25] 1: message length: 18 put message : distribution lists
$
```

※上の様に、msgId, correlId, groupId, accountingTokenは下記の様に16進表記で指定することも可能です。

```
$ mqpgf -qm QMB -q CLUS_Q1 -br
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[5
46] CodedCharSetId[932] Format[ ] Priority[0] Persistence[0] MsgId[0x414D
5120434C41202020202020202020E29B365620004103] CorrelId[0x00000000000000000000
00000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMGr[QMA ] User
Identifier[ ] AccountingToken[00000000000000000000000000000000000000000000
0000000000000000000000000000] ApplIdentityData[ ]
PutApplType[0] PutApplName[ ] PutDate[ ] PutTime[
] ApplOriginData[ ] GroupId[0x414D5120434C41202020202020202020E29B36562
0004102] MsgSeqNumber[1] Offset[0] MsgFlags[8] OriginalLength[-1]
```

```
data length: 18
00000000: 6469 7374 7269 6275 7469 6F6E 206C 6973 'distribution lis'
```



```

] PutApplType[6] PutApplName[mqpgf] ] PutDate[20151
108] PutTime[23022907] ApplOriginData[ ]

```

```

GroupId[0x67726F757049643100000000000000000000000000000000] MsgSeqNumber[1] Off
set[0] MsgFlags[8] OriginalLength[-1]

```

```

data length: 4
00000000: 6465 7374 'dest'
....
-----

```

Ex. 4.19.7 一部のクラスタ・キューのPUTに失敗した場合

※以下は2番目のキューのPUT(DISABLED)に設定した場合のテスト例です。

```

$ mqpcf que -qm QMA -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(0) PUT(ENABLED)
$ mqpcf que -qm QMB -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(0) PUT(DISABLED)
$ mqpcf que -qm QMB -q CLUS_Q2 PUT CURDEPHT
1: QUEUE(CLUS_Q2) TYPE(QLOCAL) CURDEPTH(0) PUT(ENABLED)

$ mqpgf -qm QMA -or CLUS_Q1:QMA, CLUS_Q1:QMB, CLUS_Q2:QMB -m "distribution lists"
MQOD_VERSION_2 MQPMO_VERSION_2
[16/12/22 20:24:35] 1: message length: 18 put message : distribution lists
$

```

※エラーは発生せず、2番目のクラスター・キュー以外はPUTされる。これは、MQPMO_SYNC POINT を指定しても挙動は同じで、ホストしているキューマネージャーがリモートである場合、QMAの具体的にはクラスタ伝送キューにPUTが成功する迄がトランザクションの範囲であるため、アプリケーションにエラーが戻されたり、ロールバックされることはない。

```

$ mqpcf que -qm QMA -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(1) PUT(ENABLED)
$ mqpcf que -qm QMB -q CLUS_Q1 PUT CURDEPHT
1: QUEUE(CLUS_Q1) TYPE(QLOCAL) CURDEPTH(0) PUT(DISABLED)
$ mqpcf que -qm QMB -q CLUS_Q2 PUT CURDEPHT
1: QUEUE(CLUS_Q2) TYPE(QLOCAL) PUT(ENABLED) CURDEPTH(1)
-----

```

Ex. 4.19.8 MQRC_MULTIPLE_REASONSが返却される例

\$ mqpgef -qm QMA -or CQ1:QMB,CQQ:QMA,CQ2:QMC -m "distribution lists" MQOD_VERSION
_2 MQPMO_VERSION_2

MQRC_MULTIPLE_REASONS:MQOPEN for CQ1(QMB) returned CompCode=0, Reason=0

MQRC_MULTIPLE_REASONS:MQOPEN for CQQ(QMA) returned CompCode=2, Reason=2085

MQRC_MULTIPLE_REASONS:MQOPEN for CQ2(QMC) returned CompCode=0, Reason=0

[16/12/22 20:25:39] 1: message length: 18 put message : distribution lists

MQRC_MULTIPLE_REASONS:MQPUT for CQ1(QMB) returned CompCode=0, Reason=0

MQRC_MULTIPLE_REASONS:MQPUT for CQQ(QMA) returned CompCode=2, Reason=2137

MQRC_MULTIPLE_REASONS:MQPUT for CQ2(QMC) returned CompCode=0, Reason=0

※MQRC_MULTIPLE_REASONSが返され、存在しないローカル・キューの指定にMQOPENとMQPUT
で理由コードが設定される。

\$ mqr c 2085

2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME

\$ mqr c 2137

2137 0x00000859 MQRC_OPEN_FAILED

※mqpgefはサンプル・プログラム (amqspt10.c) と同様MQOPEN/MQPUTでエラーが発生して
も処理を進めています。MQPMO_SYNCPOINTを指定した場合はMQCMITが呼び出されます。強
制的にバックアウトさせる場合は”-b”オプションを指定します。

4.20 キューマネージャーによるセグメント化

キューマネージャーによってPUTしたメッセージをセグメントに分割させます。キューまたはキューマネージャーのMAXMSGLの内、どちらか小さい方の値で分割されます。

```
mqpgf -qm <qmgr> -q <queue> -f <filename> MQMF_SEGMENTATION_ALLOWED MQMD_VERSION_2
```

MQMF_SEGMENTATION_ALLOWED: キューマネージャーにセグメント分割させる場合には、MQMD.MsgFlagsにMQMF_SEGMENTATION_ALLOWEDを指定します。

MQMD_VERSION_2: セグメント化には、MQMD_VERSION_2を使用することが必要

(任意のオプション例)

-f: PUTするメッセージデータを含むファイルのパスを指定します。

Ex. 4.20.1 キューマネージャーによるセグメント化

※テストを容易にするためにキューのMAXMSGLを100に設定します。

```
$ echo "alter ql('SampleQ') maxmsgl(100)" | runmqsc SampleQM
```

※128バイト以上の任意のテキストファイルを用意します。

```
$ ls -l largemsg.txt
```

```
-rw-r--r--    1 MQM.MANAGER      MQM          315 Apr 25 17:34 largemsg.txt
```

```
$ cat largemsg.txt
```

```
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz<
```

```
$ mqpgf -qm SampleQM -q SampleQ -f largemsg.txt MQMF_SEGMENTATION_ALLOWED MQMD_VERSION_2
```

```
[18/05/09 15:12:05] 1: put from largemsg.txt
```

```
$ mqpgf: ./mqpgf -qm SampleQM -q SampleQ -dpv -r<
```

```
message number: 1
```

```
....
```

```
GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNumber[1] Offset[0] MsgFlags[3] OriginalLength[96]
```

```

data length: 96
00000000:  3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
....
00000050:  4849 4A4B 4C4D 4E4F 5051 5253 5455 5657 'HIJKLMNOPQRSTUVW'

message number: 2
....
GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNumber[1] Offset[96] MsgFlags[3] OriginalLength[96]

data length: 96
00000000:  5859 5A61 6263 6465 6667 6869 6A6B 6C6D 'XYZabcdefghijklm'
....
00000050:  6F70 7172 7374 7576 7178 797A 0A31 3233 'opqrstuvwxyz.123'

message number: 3
....
GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNumber[1] Offset[192] MsgFlags[3] OriginalLength[96]

data length: 96
00000000:  3435 3637 3839 3041 4243 4445 4647 4849 '4567890ABCDEFGH'
....
00000050:  4B4C 4D4E 4F50 5152 5354 5556 5758 595A 'KLMNOPQRSTUVWXYZ'

message number: 4
....
GroupId[0x414D512053616D706C65514D202020205AF2892120002A03] MsgSeqNumber[1] Offset[288] MsgFlags[7] OriginalLength[27]

data length: 27
00000000:  6162 6364 6566 6768 696A 6B6C 6D6E 6F70 'abcdefghijklmnop'
00000010:  7172 7374 7576 7178 797A 0A          'qrstuvwxyz.'

no message available : SampleQ CompCd=02 ReasonCd=2033

```

※キューマネージャーによるセグメント分割は正確にMAXMSGSLの値では分割されない場合があります。例えば、MQ9.0 for WindowsやMQ8.0 for HPNonStopで確認した結果ではMAXMSGSLを超えない最大の8の倍数で分割されます。

```

$ echo "alter ql('SampleQ') maxmsgsl(4194304)" | runmqsc SampleQM
-----

```

4.21 アプリケーションによるセグメント化

mqpgfは“-as”に続いてセグメントのサイズを指定することで、アプリケーション的にセグメント分割を行います。具体的には、指定されたサイズでメッセージを分割し、分割したセグメントのMsgFlagsにMQMF_SEGMENT、最後のセグメントの場合はMQMF_LAST_SEGMENTを指定します。MQPMO.OptionsにMQPMO_LOGICAL_ORDERが指定されていない場合は、GroupId、Offsetをアプリケーションで設定することが必要です。mqpgfは引数にMQPMO_LOGICAL_ORDERが指定されていない場合、Offsetを設定しますが、GroupIdについては最初のセグメントに自動採番された値を後続のセグメントに使用することをしません。GroupIdが相違すると、MQGET時にMQGMO_COMPLETE_MSGやMQGMO_ALL_SEGMENT S_AVAILABLEが使用できません。メッセージの再組立てを可能にするには、mqpgfの引数にGroupIdを指定するか、MQPMO_LOGICAL_ORDERを指定します。mqpgfは“-gi”でGroupIdを手動で指定した場合、全てのセグメントにそのGroupIdを設定します。アプリケーションがMQPMO_LOGICAL_ORDERを指定した場合は、キューマネージャーがGroupId、MsgSeqNumberおよびOffsetに適切な値を自動的に設定します。アプリケーションは、MsgFlagsにMQMF_SEGMENT、最後のセグメントの場合はMQMF_LAST_SEGMENTを指定することのみが必要です。mqpgfもMQPMO_LOGICAL_ORDERが引数で与えられた場合は、MsgFlagsのみを設定します。

```
mqpgf -qm <qmgr> -q <queue> -f <filename> -as <segment size> MQMD_VERSION_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
```

-as: 指定したサイズでメッセージをセグメントに分割

MQMD_VERSION_2: セグメント化には、MQMD_VERSION_2を使用することが必要

MQPMO_LOGICAL_ORDER: キューマネージャーがGroupId、MsgSeqNumberおよびOffsetに適切な値を自動的に設定

(任意のオプション例)

MQPMO_SYNCPOINT: セグメント化された一連のメッセージを一つのUOW (Unit Of Work)で処理 (Commit/Backout) させます。

Ex. 4.21.1 アプリケーションによるセグメント化

```
$ mqpgf -qm SampleQM -q SampleQ -f largemsg.txt -as 100 MQMD_VERSION_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
[18/05/23 17:55:16] 1: put from: largemsg.txt
[18/05/23 17:55:16] 1: logical message: 1 length: 315 put message: 1234567890ABCDEF
GHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz. 1234567890ABCDEFGHIJKLMN
OPQRSTUVWXYZabcdefghijklmnopqrstuvxyz. 12
[18/05/23 17:55:16] 1: segment: 1 length: 100 put message: 1234567890ABCDEFGHIJ
KLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz. 1234567890ABCDEFGHIJKLMN
OPQRSTUVWXYZ
```

```

a
[18/05/23 17:55:16] 1: segment: 2 length: 100 put message: bcdefghijklmnopqrstu
vqxyz. 1234567890ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz. 1234567890
A
[18/05/23 17:55:16] 1: segment: 3 length: 100 put message: BCDEFGHIJKLMNOPQRSTU
VWXYZabcdefghijklmnopqrstuvwxyz. 1234567890ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijk
l
[18/05/23 17:55:16] 1: segment: 4 length: 15 put message: mnopqrstuvwxyz.
MQCMIT success : CompCd=00 ReasonCd=00

```

※太字が物理メッセージ。

```

$ mqpgef -qm SampleQM -q SampleQ -dpv -r
message number: 1
....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNumber[1] Off
set[0] MsgFlags[2] OriginalLength[100]

data length: 100
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
....
00000060: 5859 5A61                                'XYZa'

message number: 2
....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNumber[1] Off
set[100] MsgFlags[2] OriginalLength[100]

data length: 100
00000000: 6263 6465 6667 6869 6A6B 6C6D 6E6F 7071 'bcdefghijklmnopq'
....
00000060: 3839 3041                                '890A'

message number: 3
....
GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNumber[1] Off
set[200] MsgFlags[2] OriginalLength[100]

data length: 100
00000000: 4243 4445 4647 4849 4A4B 4C4D 4E4F 5051 'BCDEFGHIJKLMNOPQ'
....
00000060: 696A 6B6C                                'ijkl'

```

message number: 4

....

GroupId[0x414D512053616D706C65514D202020205B05070120002A0E] MsgSeqNumber[1] Offset[300] MsgFlags[6] OriginalLength[15]

data length: 15

00000000: 6D6E 6F70 7172 7374 7576 7178 797A 0A 'mnopqrstuvwxyz.'

no message available : SampleQ CompCd=02 ReasonCd=2033

※指定したセグメントサイズ（100）で分割されています。（アプリケーション）

※GroupIdは自動採番され、全て同じ値が設定されています。（キューマネージャー）

※MsgSeqNumberも全て"1"が設定されています。（キューマネージャー）

※Offsetは、"0"から始まり分割されたサイズを加算した値が以降のメッセージに設定されています。（キューマネージャー）

※MsgFlagsは最後のセグメント以外は"2"で、最後のセグメントは"6"が設定されています。"2"はMQMF_SEGMENTで、"6"はMQMF_SEGMENTとMQMF_LAST_SEGMENTでORした値です。アプリケーションで分割したので、今回はMQMF_SEGMENTATION_ALLOWEDは設定されていません。

MQMF_SEGMENTATION_ALLOWED 0x00000001

MQMF_SEGMENT 0x00000002

MQMF_LAST_SEGMENT 0x00000004

ここで、mqpgfは最後のセグメントにはMQMF_LAST_SEGMENTのみを指定しています。MQMF_LAST_SEGMENTが指定された場合、キューマネージャーによって自動的にMQMF_SEGMENTがオン（OR）されてメッセージが送信されます。（アプリケーション+キューマネージャー）

※OriginalLengthにはセグメント分割されたメッセージサイズと同じ値が設定されています。（キューマネージャー）

4.22 キューマネージャーによる論理メッセージの再組立て

キューマネージャーにセグメント化されたメッセージを再組立てさせるには、MQGMO.OptionsにMQGMO_COMPLETE_MSGを設定してMQGET()を呼び出します。

```
mqpgf -qm <qmgr> -q <queue> -dpv MQGMO_COMPLETE_MSG
```

MQGMO_COMPLETE_MSG: セグメント化されているメッセージを再組み立てするようにキュー・マネージャーに要求

(任意のオプション例)

-dpv: GETしたメッセージをダンプ表示 (詳細)

Ex. 4.22.1 キューマネージャーによる論理メッセージの再組立て

※アプリケーションで100バイト毎にセグメント分割したメッセージをPUTしておきます。

```
$ mqpgf -qm SampleQM -q SampleQ -f largemsg.txt -as 100 MQMD_VERSION_2 MQPMO_LOGICAL_ORDER
[18/05/23 17:57:48] 1: put from: largemsg.txt
[18/05/23 17:57:48] 1: logical message: 1 length: 315 put message: 1234567890ABCDEF
GHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz. 1234567890ABCDEFGHIJKLMNQRST
UVWXYZabcdefghijklmnopqrstuvxyz. 12
[18/05/23 17:57:48] 1: segment: 1 length: 100 put message: 1234567890ABCDEFGHIJ
KLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz. 1234567890ABCDEFGHIJKLMNQRSTUVWXY
Z
a
[18/05/23 17:57:48] 1: segment: 2 length: 100 put message: bcdefghijklmnopqrstu
vxyz. 1234567890ABCDEFGHIJKLMNQRSTUVWXYZabcdefghijklmnopqrstuvxyz. 1234567890
A
[18/05/23 17:57:48] 1: segment: 3 length: 100 put message: BCDEFGHIJKLMNOPQRSTU
VWXYZabcdefghijklmnopqrstuvxyz. 1234567890ABCDEFGHIJKLMNQRSTUVWXYZabcdefghijklmnop
1
[18/05/23 17:57:48] 1: segment: 4 length: 15 put message: mnopqrstuvxyz.
```

```
$ mqpcf ques -qm SampleQM -q SampleQ CURDEPTH -t
[18/05/09 17:59:54] 1: QUEUE(SampleQ) TYPE(QUEUE) CURDEPTH(4)
```

※4件(CURDEPTH)にセグメント化されています。(太字部分)

MQGMO_COMPLETE_MSGを指定し、キューマネージャーにセグメント化されたメッセージを再組立てさせます。


```
$ mqpgf -qm SampleQM -q SampleQ -dpv MQGMO_COMPLETE_MSG
```

```
message number: 1
```

```
....
```

```
GroupId[0x414D512053616D706C65514D202020205B05070120002B04] MsgSeqNumber[1] Offsets[0] MsgFlags[6] OriginalLength[315]
```

```
data length: 315
```

```
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghijk'
00000030: 6D6E 6F70 7172 7374 7576 7178 797A 0A31 'mnopqrstuvwxyz.1'
00000040: 3233 3435 3637 3839 3041 4243 4445 4647 '234567890ABCDEF'
00000050: 4849 4A4B 4C4D 4E4F 5051 5253 5455 5657 'HIJKLMNOPQRSTUV'
00000060: 5859 5A61 6263 6465 6667 6869 6A6B 6C6D 'XYZabcdefghijk'
00000070: 6E6F 7071 7273 7475 7671 7879 7A0A 3132 'nopqrstuvwxyz.12'
00000080: 3334 3536 3738 3930 4142 4344 4546 4748 '34567890ABCDEFGH'
00000090: 494A 4B4C 4D4E 4F50 5152 5354 5556 5758 'IJKLMNOPQRSTUVW'
000000A0: 595A 6162 6364 6566 6768 696A 6B6C 6D6E 'YZabcdefghijk'
000000B0: 6F70 7172 7374 7576 7178 797A 0A31 3233 'opqrstuvwxyz.123'
000000C0: 3435 3637 3839 3041 4243 4445 4647 4849 '4567890ABCDEFGH'
000000D0: 4A4B 4C4D 4E4F 5051 5253 5455 5657 5859 'JKLMNOPQRSTUVWXY'
000000E0: 5A61 6263 6465 6667 6869 6A6B 6C6D 6E6F 'Zabcdefghijk'
000000F0: 7071 7273 7475 7671 7879 7A0A 3132 3334 'pqrstuvwxyz.1234'
00000100: 3536 3738 3930 4142 4344 4546 4748 494A '567890ABCDEFGH'
00000110: 4B4C 4D4E 4F50 5152 5354 5556 5758 595A 'KLMNOPQRSTUVWXY'
00000120: 6162 6364 6566 6768 696A 6B6C 6D6E 6F70 'abcdefghijk'
00000130: 7172 7374 7576 7178 797A 0A 'qrstuvwxyz.'
```

※再組立てされて1件の物理メッセージとしてGETされています。

```
$ mqpcf ques -qm SampleQM -q SampleQ CURDEPTH -t
```

```
[18/05/09 18:04:18] 1: QUEUE(SampleQ) TYPE(QUEUE) CURDEPTH(0)
```

※4件のセグメントが1回のGETで全て削除されています。

4.23 アプリケーションによる論理メッセージの再組立て

複数の経路が存在する場合やアプリケーションのスレッド化などの影響で、セグメントが順序通りに到着しないケースを考慮し、MQGMO_LOGICAL_ORDERを指定して必ずセグメントが順序どおりに取り出されるようにします。さらに、MQGMO_ALL_MSGS_AVAILABLEを指定して、全てのセグメントが受信キューに到着するまでMQGET()処理を実施しないように指定します。

```
mqpgf -qm <qmgr> -q <queue> -dp -r MQGMO_VERSION_2 MQGMO_LOGICAL_ORDER MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_SYNCPOINT
```

MQGMO_VERSION_2: MQGMO_LOGICAL_ORDERを指定する場合は、MQGMO_VERSION_2以上を使用することが必要

MQGMO_LOGICAL_ORDER: 必ずセグメントが順序どおりに取り出されるようにする。

MQGMO_ALL_SEGMENTS_AVAILABLE: 全てのセグメントが受信キューに到着するまでMQGET()処理を実施しないようにする。

(任意のオプション例)

-dp: GETしたメッセージをダンプ表示

-r: キュー上のメッセージを繰り返しGET

Ex. 4.23.1 アプリケーションによる論理メッセージの再組立て

※MQGMO_LOGICAL_ORDERとMQGMO_ALL_SEGMENTS_AVAILABLEの挙動が分かるように、ここでは、手動で一つ々のセグメント・メッセージをキューに作成します。

3つにセグメント分割された24バイトの論理メッセージを想定します。分割されたセグメントのサイズは8バイトです。同じ論理メッセージですので、GroupIdとMsgSeqNumberは全てのセグメントで同じです。

※最後（3番目）のセグメントを最初にキューにPUTします。MQMF_LAST_SEGMENTを指定します。Offsetは16です。

```
$ mqpgf -qm SampleQM -q SampleQ -m Segment3 -gi GID -ms 1 -of 16 MQMD_VERSION_2 MQMF_LAST_SEGMENT
[18/05/23 18:01:58] 1: message length: 8 put message: Segment3
```

※続いて、先頭のセグメントをキューにPUTします。MQMF_SEGMENTを指定します。Offsetは0です。

```
$ mqpgf -qm SampleQM -q SampleQ -m Segment1 -gi GID -ms 1 -of 0 MQMD_VERSION_2 MQMF_SEGMENT
```

[18/05/23 18:02:12] 1: message length: 8 put message: Segment1

※キューのCURDEPTHは当然 2 件です。

```
$ mqpcf ques -qm SampleQM -q SampleQ CURDEPTH
1: QUEUE(SampleQ) TYPE(QUEUE) CURDEPTH(2)
```

※ここで、MQGMO_LOGICAL_ORDER、MQGMO_ALL_SEGMENTS_AVAILABLEを指定してGETしてみます。MQGMO_LOGICAL_ORDERを指定する場合は、MQGMO_VERSION_2以上とMQMD_VERSION_2の指定が必要です。mqpgfで“-dp”を指定する場合は、MQMD_VERSION_2がデフォルトで使用されます。

```
$ mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_LOGICAL_ORDER MQGMO
_ALL_SEGMENTS_AVAILABLE
no message available : SampleQ CompCd=02 ReasonCd=2033
```

```
$ mqrc 2033
```

```
2033 0x000007f1 MQRC_NO_MSG_AVAILABLE
```

※全てのセグメントが受信キューに到着していないので、MQGET()からはMQRC_NO_MSG_AVAILABLEが返却され、メッセージは一つもGETされません。

※残りの 2 番目のセグメントをPUTします。MQMF_SEGMENTを指定します。Offsetは8です。

```
$ mqpgf -qm SampleQM -q SampleQ -m Segment2 -gi GID -ms 1 -of 8 MQMD_VERSION_2 M
QMF_SEGMENT
[18/05/23 18:03:35] 1: message length: 8 put message: Segment2
```

※念のためこの時点でキューの中のメッセージを、物理順序/FIFO（デフォルト）の順序でブラウズして確認します。

```
$ mqpgf -qm SampleQM -q SampleQ -br -r
message number: 1
....
```

```
GroupId[0x4749440000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[16] MsgFlags[6] OriginalLength[8]
```

```
data length: 8
00000000: 5365 676D 656E 7433 'Segment3'
```

```
message number: 2
```

....
GroupId[0x47494400] MsgSeqNumber[1] **Offset[0]** MsgFlags[2] OriginalLength[8]

data length: 8
00000000: 5365 676D 656E 7431 'Segment1'

message number: 3

....
GroupId[0x47494400] MsgSeqNumber[1] **Offset[8]** MsgFlags[2] OriginalLength[8]

data length: 8
00000000: 5365 676D 656E 7432 'Segment2'

no message available : SampleQ CompCd=02 ReasonCd=2033

※3番目、先頭、2番目のセグメントの順になっています。

※MQGMO_LOGICAL_ORDER、MQGMO_ALL_SEGMENTS_AVAILABLEを指定して繰り返しGETしてみます。

```
$ mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_LOGICAL_ORDER MQGMO  
_ALL_SEGMENTS_AVAILABLE MQGMO_SYNCPOINT
```

message number: 1

....
GroupId[0x47494400] MsgSeqNumber[1] **Offset[0]** MsgFlags[2] OriginalLength[8]

data length: 8
00000000: 5365 676D 656E 7431 'Segment1'

message number: 2

....
GroupId[0x47494400] MsgSeqNumber[1] **Offset[8]** MsgFlags[2] OriginalLength[8]

data length: 8
00000000: 5365 676D 656E 7432 'Segment2'

message number: 3

....

GroupId[0x474944000] MsgSeqNumber[1] **Offset[16]** MsgFlags[6] OriginalLength[8]

data length: 8

00000000: 5365 676D 656E 7433 'Segment3'

no message available : SampleQ CompCd=02 ReasonCd=2033

MQCMIT success : CompCd=00 ReasonCd=00

※セグメントがOffsetの値に従い、正しく論理順序でGETされています。

※mqpgfは実際の論理メッセージの再組立てまでは行いません。意図した順序でセグメントが受信できることまでを確認できます。

4.24 論理メッセージのグループ化

“-dl”に続いて、指定されたメッセージまたはファイルからグループ化された論理メッセージを作成する為に使用するデリミタを指定できます。具体的には、デリミタを除いた論理メッセージに分割し、分割した論理メッセージのMsgFlagsにMQMF_MSG_IN_GROUP、最後の論理メッセージの場合はMQMF_LAST_MSG_IN_GROUPを設定します。MQPMO_LOGICAL_ORDERは、論理メッセージのグループ化の場合にも機能します。引数にMQPMO_LOGICAL_ORDERが指定されず、かつGroupIdも指定されていない場合、論理メッセージ毎に別々のGroupIdが採番され、MsgSeqNumberもインクリメントされません。MQPMO_LOGICAL_ORDERを指定した場合は、キューマネージャーがGroupId、MsgSeqNumber（およびOffset）に適切な値を自動的に設定します。

```
mqpgef -qm <qmgr> -q <queue> -f <filename> -dl <delimiter> MQMD_VERSION_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
```

-dl: 論理メッセージに分割する為のデリミタを文字列もしくは16進表記で指定
MQMD_VERSION_2: メッセージのグループ化には、MQMD_VERSION_2を使用することが必要
MQPMO_LOGICAL_ORDER: キューマネージャーにGroupId、MsgSeqNumberに適切な値を設定させる。

(任意のオプション例)

-f: PUTするメッセージデータを含むファイルのパスを指定します。

MQPMO_SYNCPOINT: セグメント化された一連のメッセージを一つのUOW (Unit Of Work) で処理 (Commit/Backout) させます。

Ex. 4.24.1 論理メッセージのグループ化

※下記のような複数行の任意のテキストファイルを用意します。

```
$ cat largemsg2.txt
```

```
1234567890
```

```
1234567890ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvqxyz
```

```
1234567890ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvqxyz
```

```
1234567890ABCDEFGHIJKLMNQRSTUUVWXYZ
```

```
1234567890ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvqxyz
```

※この複数行のファイル（改行はLF:0x0a）を0x0a (LF)をデリミタに指定して、行ごとに論理メッセージに分割してメッセージをPUTします。MQPMO_LOGICAL_ORDERも指定します。Windowsの場合は通常改行は0x0d0a (CRLF)ですので、“-dl 0x0d0a”の様に指定します。Unix系のOSやHP NonStopのOSS環境では改行は0x0a (LF)です。

```
$ mqpgef -qm SampleQM -q SampleQ -f largemsg2.txt -dl 0x0a MQMD_VERSION_2 MQPMO_L
```

```

OGICAL_ORDER MQPMO_SYNCPOINT
[18/05/23 18:05:31] 1: put from: largemsg2.txt
[18/05/23 18:05:31] 1: logical message: 1 length: 10 put message: 1234567890
[18/05/23 18:05:31] 1: logical message: 2 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
[18/05/23 18:05:31] 1: logical message: 3 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
[18/05/23 18:05:31] 1: logical message: 4 length: 36 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZ
[18/05/23 18:05:31] 1: logical message: 5 length: 62 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
MQCMIT success : CompCd=00 ReasonCd=00

```

※太字の部分が物理メッセージです。

※分割されたメッセージを確認してみます。このケースでは論理メッセージ=物理メッセージです。

```

$ mqpgf -qm SampleQM -q SampleQ -dp -r
message number: 1
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[1] Off
set[0] MsgFlags[8] OriginalLength[-1]

data length: 10
00000000: 3132 3334 3536 3738 3930                '1234567890'

message number: 2
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[2] Off
set[0] MsgFlags[8] OriginalLength[-1]

data length: 62
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghijk'
00000030: 6D6E 6F70 7172 7374 7576 7178 797A      'mnopqrstuvwxyz'

message number: 3
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[3] Off
set[0] MsgFlags[8] OriginalLength[-1]

```

```

data length: 62
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghij'
00000030: 6D6E 6F70 7172 7374 7576 7178 797A      'mnopqrstuvwxyz'

message number: 4
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[4] Off
set[0] MsgFlags[8] OriginalLength[-1]

data length: 36
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A      'WXYZ'

message number: 5
....
GroupId[0x414D512053616D706C65514D202020205B05070120002C19] MsgSeqNumber[5] Off
set[0] MsgFlags[24] OriginalLength[-1]

data length: 62
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364 6566 6768 696A 6B6C 'WXYZabcdefghij'
00000030: 6D6E 6F70 7172 7374 7576 7178 797A      'mnopqrstuvwxyz'

no message available : SampleQ CompCd=02 ReasonCd=2033

※行ごと (LF:0x0aごと) に論理メッセージに分割されています。(アプリケーション)
※GroupIdは自動採番され、全て同じ値です。(キューマネージャー)
※MsgSeqNumberも全て"1"から順にインクリメントされています。(キューマネージャー)
※MsgFlagsは最後のセグメント以外は"8"で、最後のセグメントは"24"が設定されています。
※"3"はMQMF_MSG_IN_GROUPで、"24"はMQMF_MSG_IN_GROUPとMQMF_LAST_MSG_IN_GROUPをORした値です。

MQMF_MSG_IN_GROUP      0x00000008
MQMF_LAST_MSG_IN_GROUP 0x00000010

```


$\text{MQMF_MSG_IN_GROUP} \mid \text{MQMF_LAST_MSG_IN_GROUP} = 0x00000008 \mid 0x00000010 = 0x00000018$ (Hexadecimal) = 24 (Decimal)

※mqpgfは最後のセグメントにはMQMF_MSG_IN_GROUPのみを指定しています。MQMF_LAST_MSG_IN_GROUPが指定された場合、キューマネージャーによって自動的にMQMF_MSG_IN_GROUPがオン（OR）されてメッセージが送信されます。（アプリケーション+キューマネージャー）

4.25 論理メッセージのグループ化と論理メッセージのセグメント分割

論理メッセージのグループ化と論理メッセージのセグメント分割を組み合わせることも可能です。

```
mqpgf -qm <qmgr> -q <queue> -f <filename> -dl <delimiter> -as <segment size> MQMD_VERSION_2 MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
```

-dl: 論理メッセージに分割する為のデリミタを文字列もしくは16進表記で指定

-as: 指定したサイズでメッセージをセグメントに分割

MQMD_VERSION_2: メッセージのグループ化とセグメント化には、MQMD_VERSION_2を使用することが必要

MQPMO_LOGICAL_ORDER: キューマネージャーにGroupId、MsgSeqNumber、Offsetに適切な値を設定させる。

(任意のオプション例)

-f: PUTするメッセージデータを含むファイルのパスを指定します。

MQPMO_SYNCPOINT: グループ化およびセグメント化された一連のメッセージを一つのUOW (Unit Of Work)で処理 (Commit/Backout) させます。

Ex. 4.25.1 論理メッセージのグループ化と論理メッセージのセグメント分割

※ファイル内のデータをデリミタ (0x0a:LF) で論理メッセージに分割し、引数に“-as 40”を追加指定して40バイトを超える論理メッセージをアプリケーションでセグメント分割します。

```
$ mqpgf -qm SampleQM -q SampleQ -f largemsg2.txt -dl 0x0a -as 40 MQMD_VERSION_2
MQPMO_LOGICAL_ORDER MQPMO_SYNCPOINT
[18/05/25 15:12:29] 1: put from: largemsg2.txt
[18/05/25 15:12:29] 1: logical message: 1 length: 10 put message: 1234567890
[18/05/25 15:12:29] 1: logical message: 2 length: 62 put message: 1234567890ABCD
EFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
[18/05/25 15:12:29] 1: segment: 1 length: 40 put message: 1234567890ABCDEFGHJK
LMNOPQRSTUVWXYZabcd
[18/05/25 15:12:29] 1: segment: 2 length: 22 put message: efghijklmnopqrstuvq
yz
[18/05/25 15:12:29] 1: logical message: 3 length: 62 put message: 1234567890ABCD
EFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
[18/05/25 15:12:29] 1: segment: 1 length: 40 put message: 1234567890ABCDEFGHJK
```

```

LMNOPQRSTUVWXYZabcd
[18/05/25 15:12:29] 1: segment: 2 length: 22 put message: efghijklmnopqrstuvqxy
z
[18/05/25 15:12:29] 1: logical message: 4 length: 36 put message: 1234567890ABC
DEFGHIJKLMNOPQRSTUVWXYZ
[18/05/25 15:12:29] 1: logical message: 5 length: 62 put message: 1234567890ABCD
EFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
[18/05/25 15:12:29] 1: segment: 1 length: 40 put message: 1234567890ABCDEFGHIJK
LMNOPQRSTUVWXYZabcd
[18/05/25 15:12:29] 1: segment: 2 length: 22 put message: efghijklmnopqrstuvqxy
z
MQCMIT success : CompCd=00 ReasonCd=00

```

※太字の部分が物理メッセージです。

※分割されたメッセージを確認します。

```

$ mqpgef -qm SampleQM -q SampleQ -dp -r
message number: 1
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[1] Offs
et[0] MsgFlags[8] OriginalLength[-1]

data length: 10
00000000: 3132 3334 3536 3738 3930                '1234567890'

message number: 2
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[2] Offs
et[0] MsgFlags[10] OriginalLength[40]

data length: 40
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364                'WXYZabcd'

message number: 3
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[2] Offs
et[40] MsgFlags[14] OriginalLength[22]

data length: 22

```

```

00000000: 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 'efghijklmnopqrst'
00000010: 7576 7178 797A                               'uvqxyz',

message number: 4
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[3] Offset[0] MsgFlags[10] OriginalLength[40]

data length: 40
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364                               'WXYZabcd',

message number: 5
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[3] Offset[40] MsgFlags[14] OriginalLength[22]

data length: 22
00000000: 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 'efghijklmnopqrst'
00000010: 7576 7178 797A                               'uvqxyz',

message number: 6
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[4] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 36
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A                               'WXYZ',

message number: 7
....
GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] MsgSeqNumber[5] Offset[0] MsgFlags[26] OriginalLength[40]

data length: 40
00000000: 3132 3334 3536 3738 3930 4142 4344 4546 '1234567890ABCDEF'
00000010: 4748 494A 4B4C 4D4E 4F50 5152 5354 5556 'GHIJKLMNOPQRSTU'
00000020: 5758 595A 6162 6364                               'WXYZabcd',

```

message number: 8

....

GroupId[0x414D512053616D706C65514D202020205B07A8F820002303] **MsgSeqNumber[5] Offset[40]** MsgFlags[30] OriginalLength[22]

data length: 22

00000000: 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 'efghijklmnopqrst'
00000010: 7576 7178 797A 'uvqxyz'

no message available : SampleQ CompCd=02 ReasonCd=2033

MQCMIT success : CompCd=00 ReasonCd=00

※ 2、3、5 番目の論理メッセージ (MsgSeqNumberが2, 3, 5) の物理メッセージにOffsetが設定され複数にセグメント分割されていることが確認できます。

4.26 グループ・メッセージの読み込み

ここで説明する例では、1つのファイルのデータを分割してグループ化した論理メッセージを作成しますが、mqpgfはそれらを読み取った時に一つのメッセージまたはファイルに再組立てはしません。意図した順序で論理メッセージが受信できることが確認できるのみです。

```
mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_COMPLETE_MSG MQGMO_LOGICAL_ORDER MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_WAIT MQWI_UNLIMITED MQGMO_NO_SYNCPOINT
```

MQGMO_VERSION_2: MQGMO_VERSION_2以上を使用することが必要

MQGMO_COMPLETE_MSG: セグメント化されているメッセージを再組み立てするようにキュー・マネージャーに要求

MQGMO_LOGICAL_ORDER: 論理メッセージおよびセグメントが順序どおりに取り出されるようにする。

MQGMO_ALL_MSGS_AVAILABLE: グループ内のすべてのメッセージが到着するまでMQGET()処理を実施しないようにする。

MQGMO_ALL_SEGMENTS_AVAILABLE: 全てのセグメントが受信キューに到着するまでMQGET()処理を実施しないようにする。

(任意のオプション例)

-dp: GETしたメッセージをダンプ表示

-r: キュー上のメッセージを繰り返しGET

MQGMO_WAIT: メッセージの到着を待機

MQWI_UNLIMITED: メッセージの到着を待つ時間は無制限

MQPMO_SYNCPOINT: セグメント化された一連のメッセージを一つのUOW (Unit Of Work)で処理 (Commit/Backout) させます。

Ex. 4.26.1 グループ・メッセージの読み込み

※各オプションの挙動が分かりやすくなるように、ここでは、手動で一つずつ物理メッセージをキューに作成します。

下記の様に分割された2種類のグループメッセージを想定します。

グループ1 論理メッセージ1 セグメント1

グループ1 論理メッセージ1 セグメント2

グループ1 論理メッセージ2

グループ1 論理メッセージ3 セグメント1

グループ1 論理メッセージ3 セグメント2

グループ2 論理メッセージ1

グループ2 論理メッセージ2 セグメント1
グループ2 論理メッセージ2 セグメント2

これらの対応するコマンド・ラインは下記です。

```
mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment1 -gi GroupId1 -ms 1 -of 0
MQMD_VERSION_2 MQMF_SEGMENT MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment2 -gi GroupId1 -ms 1 -of 2
2 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical2 -gi GroupId1 -ms 2 MQMD_VERSION_
2 MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment1 -gi GroupId1 -ms 3 -of 0
MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment2 -gi GroupId1 -ms 3 -of 2
2 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP

mqpgf -qm SampleQM -q SampleQ -m Group2Logical1 -gi GroupId2 -ms 1 MQMD_VERSION_
2 MQMF_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment1 -gi GroupId2 -ms 2 -of 0
MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment2 -gi GroupId2 -ms 2 -of 2
2 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP
```

※まず、別のターミナル／コマンド・プロンプトで下記引数で受信の為のmqpgfを起動しておきます。

```
$ mqpgf -qm SampleQM -q SampleQ -dp -r MQGMO_VERSION_2 MQGMO_COMPLETE_MSG MQGMO_
LOGICAL_ORDER MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_WAIT M
QWI_UNLIMITED MQGMO_NO_SYNCPOINT
```

※MQGMO_WAITとMQWI_UNLIMITEDの指定によって、グループのメッセージが全てキューに到着するまで、無期限で待ちます。すべて到着した時点でそれぞれのグループのメッセージの処理が開始されるはずですが、さらに“-r”オプションの指定により複数のグループでそれを繰り返します。

※ランダムな順序でメッセージをPUTしていきます。

```
$ mqpgf -qm SampleQM -q SampleQ -m Group2Logical2Segment2 -gi GroupId2 -ms 2 -of
22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP
[18/05/25 16:45:38] 1: message length: 22 put message : Group2Logical2Segment2
```

```
$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical3Segment2 -gi GroupId1 -ms 3 -of
22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_LAST_MSG_IN_GROUP
```

```

[18/05/25 16:45:44] 1: message length: 22 put message : Group1Logical3Segment2

$ mqpgef -qm SampleQM -q SampleQ -m Group2Logical2Segment1 -gi GroupId2 -ms 2 -of
0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
[18/05/25 16:45:49] 1: message length: 22 put message : Group2Logical2Segment1

$ mqpgef -qm SampleQM -q SampleQ -m Group1Logical3Segment1 -gi GroupId1 -ms 3 -of
0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_LAST_MSG_IN_GROUP
[18/05/25 16:45:55] 1: message length: 22 put message : Group1Logical3Segment1

$ mqpgef -qm SampleQM -q SampleQ -m Group1Logical2 -gi GroupId1 -ms 2 MQMD_VERSION_2
MQMF_MSG_IN_GROUP
[18/05/25 16:46:01] 1: message length: 14 put message : Group1Logical2

$ mqpgef -qm SampleQM -q SampleQ -m Group1Logical1Segment2 -gi GroupId1 -ms 1 -of
22 MQMD_VERSION_2 MQMF_LAST_SEGMENT MQMF_MSG_IN_GROUP
[18/05/25 16:46:06] 1: message length: 22 put message : Group1Logical1Segment2

$ mqpgef -qm SampleQM -q SampleQ -m Group2Logical1 -gi GroupId2 -ms 1 MQMD_VERSION_2
MQMF_MSG_IN_GROUP
[18/05/25 16:46:11] 1: message length: 14 put message : Group2Logical1

```

※この時点で、グループ2のメッセージを全て書き込んだので、別ターミナルでMQGET() 待機していたmqpgefがメッセージを処理し、それが表示されます。

```

message number: 1
....
GroupId[0x47726F757049643200000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[8] OriginalLength[-1]

data length: 14
00000000: 4772 6F75 7032 4C6F 6769 6361 6C31          'Group2Logical1 '

message number: 2
....
GroupId[0x47726F757049643200000000000000000000000000000000] MsgSeqNumber[2] Offset[0] MsgFlags[30] OriginalLength[44]

data length: 44
00000000: 4772 6F75 7032 4C6F 6769 6361 6C32 5365      'Group2Logical2Se'
00000010: 676D 656E 7431 4772 6F75 7032 4C6F 6769      'gment1Group2Logi'
00000020: 6361 6C32 5365 676D 656E 7432                'cal2Segment2 '

```


※論理メッセージ順にGETされ、分割されたセグメントが一つの論理メッセージになっていることが確認できます。

※最後にグループ1のメッセージの残りの1件を書き込みます。

```
$ mqpgf -qm SampleQM -q SampleQ -m Group1Logical1Segment1 -gi GroupId1 -ms 1 -of
0 MQMD_VERSION_2 MQMF_SEGMENT MQMF_MSG_IN_GROUP
[18/05/25 16:46:16] 1: message length: 22 put message : Group1Logical1Segment1
```

※この時点で、グループ1のメッセージも全て書き込んだので、別ターミナルでMQGET() 待機していたmqpgfがメッセージを処理し、それが表示されます。

message number: 3

....

```
GroupId[0x47726F757049643100000000000000000000000000000000] MsgSeqNumber[1] Off
set[0] MsgFlags[14] OriginalLength[44]
```

data length: 44

```
00000000: 4772 6F75 7031 4C6F 6769 6361 6C31 5365 'Group1Logical1Se'
00000010: 676D 656E 7431 4772 6F75 7031 4C6F 6769 'gment1Group1Logi'
00000020: 6361 6C31 5365 676D 656E 7432 'cal1Segment2'
```

message number: 4

....

```
GroupId[0x47726F757049643100000000000000000000000000000000] MsgSeqNumber[2] Off
set[0] MsgFlags[8] OriginalLength[-1]
```

data length: 14

```
00000000: 4772 6F75 7031 4C6F 6769 6361 6C32 'Group1Logical2'
```

message number: 5

....

```
GroupId[0x47726F757049643100000000000000000000000000000000] MsgSeqNumber[3] Off
set[0] MsgFlags[30] OriginalLength[44]
```

data length: 44

```
00000000: 4772 6F75 7031 4C6F 6769 6361 6C33 5365 'Group1Logical3Se'
00000010: 676D 656E 7431 4772 6F75 7031 4C6F 6769 'gment1Group1Logi'
00000020: 6361 6C33 5365 676D 656E 7432 'cal3Segment2'
```

※グループ1のメッセージについても、論理メッセージ順にGETされ、分割されたセグメ

ントが一つの論理メッセージになっていることが確認できます。

5. 全パラメータ・リファレンス

5.1 基本パラメータ

キューマネージャ名 (-qm)

mqpgf は、全ての場合に接続するキューマネージャ名 (-qm) の指定が必要です。

mqpgf -qm <qmgr1[, qmgr2, qmgr3...]>....

カンマで区切ったキューマネージャのリストを指定することもできます。リストで指定したキューマネージャは全て同じマシンに存在し、同じ名前のキューを持っている必要があります。

Ex. 5.1.1 複数のキューマネージャに接続する場合

※シングル・スレッドから、複数のキューマネージャに接続する場合、MQCNO_HANDLE_SHARE_BLOCKまたはMQCNO_HANDLE_SHARE_NO_BLOCKを指定して、「共用（スレッド独立）接続」を作成する必要があります。下記は“-tr”（簡易APIトレース）を指定して、各キューマネージャへの接続ハンドルと共に各MQI呼び出しの状況を確認しています。

```
$ mqpgf -qm SampleQM,RemoteQM,PartialQM -q LQ1 -m test -tr MQCNO_HANDLE_SHARE_BLOCK MQPMO_SYNCPOINT
[18/07/26 16:24:38.817780] MQCONN start qmgr:SampleQM Options:0x00000040
[18/07/26 16:24:38.876475] MQCONN stop hcon:20971525 qmgr:SampleQM CompCd=00 ReasonCd=00
[18/07/26 16:24:38.876659] MQCONN start qmgr:RemoteQM Options:0x00000040
[18/07/26 16:24:38.912757] MQCONN stop hcon:20971527 qmgr:RemoteQM CompCd=00 ReasonCd=00
[18/07/26 16:24:38.912892] MQCONN start qmgr:PartialQM Options:0x00000040
[18/07/26 16:24:38.958100] MQCONN stop hcon:20971529 qmgr:PartialQM CompCd=00 ReasonCd=00
[18/07/26 16:24:38.958741] MQOPEN start hcon:20971525 ObjectName:LQ1 Options:0x00000010
[18/07/26 16:24:38.959152] MQOPEN stop hcon:20971525 ObjectName:LQ1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.959511] 1: message length: 4 put message: test
```

```

[18/07/26 16:24:38.959655] MQPUT start hcon:20971525 Options:0x00000000
[18/07/26 16:24:38.966288] MQPUT stop hcon:20971525 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.966422] MQCMIT start hcon:20971525
[18/07/26 16:24:38.967003] MQCMIT stop hcon:20971525 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/07/26 16:24:38.967198] MQCLOSE start hcon:20971525 Options:0x00000000
[18/07/26 16:24:38.976426] MQCLOSE stop hcon:20971525 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.976530] MQOPEN start hcon:20971527 ObjectName:LQ1 Options:0x0
0000010
[18/07/26 16:24:38.976956] MQOPEN stop hcon:20971527 ObjectName:LQ1 CompCd=00 Re
asonCd=00
[18/07/26 16:24:38.977041] 1: message length: 4 put message: test
[18/07/26 16:24:38.977168] MQPUT start hcon:20971527 Options:0x00000000
[18/07/26 16:24:38.982164] MQPUT stop hcon:20971527 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.982252] MQCMIT start hcon:20971527
[18/07/26 16:24:38.982710] MQCMIT stop hcon:20971527 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/07/26 16:24:38.982838] MQCLOSE start hcon:20971527 Options:0x00000000
[18/07/26 16:24:38.987732] MQCLOSE stop hcon:20971527 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.987812] MQOPEN start hcon:20971529 ObjectName:LQ1 Options:0x0
0000010
[18/07/26 16:24:38.988205] MQOPEN stop hcon:20971529 ObjectName:LQ1 CompCd=00 Re
asonCd=00
[18/07/26 16:24:38.988289] 1: message length: 4 put message: test
[18/07/26 16:24:38.988414] MQPUT start hcon:20971529 Options:0x00000000
[18/07/26 16:24:38.994190] MQPUT stop hcon:20971529 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.994299] MQCMIT start hcon:20971529
[18/07/26 16:24:38.994816] MQCMIT stop hcon:20971529 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/07/26 16:24:38.994947] MQCLOSE start hcon:20971529 Options:0x00000000
[18/07/26 16:24:38.996036] MQCLOSE stop hcon:20971529 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.996115] MQDISC start hcon:20971525
[18/07/26 16:24:38.996454] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.996535] MQDISC start hcon:20971527
[18/07/26 16:24:38.996830] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/07/26 16:24:38.996917] MQDISC start hcon:20971529
[18/07/26 16:24:39.021688] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
-----

```

キュー名 (-q)

キューに対してのオペレーションを実行する場合、キュー名(-q)の指定が必要です。

```
mppgf -qm <qmgr> -q <queue>
```

入力メッセージ (-m)

PUTするメッセージをコマンドラインで指定します。メッセージにスペースが含まれている場合、ダブルクォート(")またはシングルクォート(')で囲みます。（「4. 基本的なテストの実施方法 - コマンドラインで指定したメッセージをPUT」参照）

```
mppgf -qm <qmgr> -q <queue> -m <input message>
```

例)

```
mppgf -qm <qmgr> -q <queue> -m "input message"
```

入力メッセージ (16進表記) (-mx)

PUTするメッセージをコマンドラインで16進表記で指定します。使用できる文字は、0-9, A-F, a-fで、必ず偶数個指定する必要があります。（「4. 基本的なテストの実施方法 - コマンドラインで指定するPUTメッセージをHexadecimalで指定」参照）

```
mppgf -qm <qmgr> -q <queue> -mx <input message(hexadecimal notation)>
```

入力メッセージ・ファイル名 (-f)

PUTするメッセージデータを含むファイルのパスを指定します。（「4. 基本的なテストの実施方法 - ファイルデータをPUT」参照）

```
mppgf -qm <qmgr> -q <queue> -f <input file path>
```

例)

```
mppgf -qm <qmgr> -q <queue> -f work/inputMessage1.txt
```

出力メッセージ・ファイル名 (-o)

GETしたメッセージデータを書き出すファイルのパスを指定します。（「4. 基本的なテストの実施方法 — GETしたメッセージをファイルへ出力」参照）

```
mqpgf -qm <qmgr> -q <queue> -o <output file path>
```

例)

```
mqpgf -qm <qmgr> -q <queue> -o work/outputMessage1.txt
```

出力キュー名 (-oq)

GETしたメッセージデータを書き出すキューを指定します。（「4. 基本的なテストの実施方法 — GETしたメッセージをキューへ出力(リキュー)」参照）

"*"一文字を指定することもできます。この指定の場合、受信メッセージのMQMD.ReplyToQへメッセージを書き込みます。"**"を指定した場合は、さらにMQMD.ReplyToQMGrをMQOD.ObjectQMGrNameにセットします。

MQPMO_PASS_* が指定されている場合、入力キューのハンドルをMQPMO.Contextに設定します。（「Ex. 6.9.2 入力の識別、起点、ユーザー・コンテキストを引き継いでリキューする例」参照）

出力側（2次側）のキューにパラメータ／オプションを設定する場合は -ss オプションを使用して以降のパラメータ／オプションを出力側（2次側）のキューに切り替えます。（「5. 全パラメータ・リファレンス — 以降のパラメータを2次側に切り替える (-ss)」参照）

```
mqpgf -qm <qmgr> -q <queue> -oq <output queue name >
```

入力キュー名 (-iq)

PUTした後に、応答を受信するキューを指定します。（「4. 基本的なテストの実施方法 — PUTしたメッセージの応答を受信」参照）

ここで指定したキュー名称は、送信メッセージのMQMD.ReplyToQに設定されます。

入力側（2次側）のキューにパラメータ／オプションを設定する場合は -ss オプションを使用して以降のパラメータ／オプションを入力側（2次側）のキューに切り替えます。（「5. 全パラメータ・リファレンス — 以降のパラメータを2次側に切り替える (-ss)」参照）

```
mqpgf -qm <qmgr> -q <queue> -iq <input queue name >
```

入力ディレクトリ名 (-d)

ディレクトリ内のファイルデータを全てPUTします。

ファイルデータはMQMDを除いた部分のデータです。特に指定のない場合は、MQMDはデフォルトの値(MQMD_DEFAULT)が設定されてPUTされます。

ディレクトリ名にスペースが含まれている場合、ダブルクォート(“)またはシングルクォート(‘)で囲みます。(「4. 基本的なテストの実施方法 - ディレクトリ内のファイルデータを全てPUT」参照)

```
mqpgf -qm <qmgr> -q <queue> -d <directory>
```

例)

```
mqpgf -qm <qmgr> -q <queue> -d “work/input message dir”
```

出力ディレクトリ名 (-g)

キュー上のメッセージを書き出すディレクトリのパスを指定します。ディレクトリ名にスペースが含まれている場合、ダブルクォート(“)またはシングルクォート(‘)で囲みます。(「4. 基本的なテストの実施方法 - キュー内のメッセージをGETしディレクトリに出力」参照)

```
mqpgf -qm <qmgr> -q <queue> -g <directory>
```

例)

```
mqpgf -qm <qmgr> -q <queue> -g “work/output message dir”
```

キュー上の全てメッセージをGET (-r)

キュー上の全てのメッセージをMQGETします。

```
mqpgf -qm <qmgr> -q <queue> -r
```

強制バックアウト (-b)

メッセージをMQPUT/MQGETした後、MQBACK()を呼び出します。MQGMO_SYNCPOINTを同時に指定する必要があります。

PUT時 -n、GET時 -r オプションと共に使用することも可能です。

```
mqpgf -qm <qmgr> -q <queue> -m <input message> -n <count> -b MQPMO_SYNCPOINT
mqpgf -qm <qmgr> -q <queue> -r -b MQGMO_SYNCPOINT
```

Ex. 5.1.2 PUT時に指定する場合

```
-----
$ mqpcf que -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(QLOCAL) CURDEPTH(0)

$ mqpgf -qm TESTQM -q TQ -m "test message" -n 3 -b MQPMO_SYNCPOINT
[16/12/22 20:30:59] 1: message length: 12 put message : test message
[16/12/22 20:30:59] 2: message length: 12 put message : test message
[16/12/22 20:30:59] 3: message length: 12 put message : test message
MQBACK success : CompCd=00 ReasonCd=00

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(Queue) CURDEPTH(0)
-----
```

Ex. 5.1.3 GET時に指定する場合

```
-----
$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(Queue) CURDEPTH(3)

$ mqpgf -qm TESTQM -q TQ -r -b MQGMO_SYNCPOINT
[16/12/21 19:56:15] 1: message length: 12 get message : test message
[16/12/21 19:56:15] 2: message length: 12 get message : test message
[16/12/21 19:56:15] 3: message length: 12 get message : test message
no message available : TQ CompCd=02 ReasonCd=2033
MQBACK success : CompCd=00 ReasonCd=00

$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(Queue) CURDEPTH(3)

$ mqpgf -qm TESTQM -q TQ -br -r |grep MD
*StrucId[MD ] .... BackoutCount[1] ....
*StrucId[MD ] .... BackoutCount[1] ....
```



```
*StrucId[MD  ] .... BackoutCount[1] ....
```

PUTするメッセージのサイズ指定 (-l)

PUTするメッセージのサイズを指定します。指定したメッセージよりも小さい場合は残りのデータは切り捨てられます。指定したメッセージよりも大きい場合は、NULL(0x00)が後続に付加されます。**-m**(メッセージ)/**-mx**(メッセージ16進表記)/**-f**(ファイルをPUT)/**-d**(ディレクトリ内のファイルをPUT) オプションと共に使用可能です。

```
mqqpgf -qm <qmgr> -q <queue> -m <input message> -l <size>
mqqpgf -qm <qmgr> -q <queue> -mx <input message(hexadecimal notation)> -l <size>
mqqpgf -qm <qmgr> -q <queue> -f <filename> -l <size>
mqqpgf -qm <qmgr> -q <queue> -d <directory> -l <size>
```

Ex. 5.1.4 -mオプションと共に使用した例

```
$ mqqpgf -qm TESTQM -q TQ -m 1234567890 -l 5
[16/12/22 20:33:08] 1: message length: 5 put message : 12345

$ mqqpgf -qm TESTQM -q TQ -br
....
data length: 5
00000000: 3132 3334 35                                '12345'
```



```
$ mqqpgf -qm TESTQM -q TQ -m 1234567890 -l 15
[16/12/22 20:33:45] 1: message length: 15 put message : 1234567890.....

$ mqqpgf -qm TESTQM -q TQ -br
....
data length: 15
00000000: 3132 3334 3536 3738 3930 0000 0000 00      '1234567890.....'
```

Ex. 5.1.5 -mxオプションと共に使用した例

```
$ mqqpgf -qm TESTQM -q TQ -mx 31323334353637383930 -l 5
[16/12/22 20:34:36] 1: message length: 5 put message : 0x3132333435
```

```

$ mqpgef -qm TESTQM -q TQ -br
....
data length: 5
00000000: 3132 3334 35                                '12345          '

$ mqpgef -qm TESTQM -q TQ -mx 31323334353637383930 -l 15
[16/12/22 20:35:52] 1: message length: 15 put message : 0x3132333435363738393000
00000000

$ mqpgef -qm TESTQM -q TQ -br
message number: 1
....
data length: 15
00000000: 3132 3334 3536 3738 3930 0000 0000 00          '1234567890..... '
-----

```

Ex. 5.1.6 -fオプションと共に使用した例

```

-----
$ ls -l input.txt
-rw-r--r-- 1 guest staff 6 Jun 15 18:44 input.txt

$ cat input.txt
123456

$ mqpgef -qm TESTQM -q TQ -f input.txt -l 3
[16/12/22 20:37:50] 1: put from input.txt

$ mqpgef -qm TESTQM -q TQ -br
....
data length: 3
00000000: 3132 33                                '123          '

$ mqpgef -qm TESTQM -q TQ -f input.txt -l 15
[16/12/22 20:37:55] 1: put input.txt
$ mqpgef -qm TESTQM -q TQ -br
....
data length: 15
00000000: 3132 3334 3536 0000 0000 0000 0000 00          '123456..... '
-----

```

Ex. 5.1.7 -dオプションと共に使用した例

```
-----
$ ls -l input
total 24
-rw-r--r--    1 mq80      mqm          262 Oct 13 21:00 input1.txt
-rw-r--r--    1 mq80      mqm          262 Oct 13 21:00 input2.txt
-rw-r--r--    1 mq80      mqm          262 Oct 13 21:00 input3.txt

$ mqpgef -qm TESTQM -q TQ -d input -l 3
put input/input1.txt
put input/input2.txt
put input/input3.txt
% mqpgef -qm TESTQM -q TQ -br -r
message number: 1
....
data length: 3
00000000: 3530 30                                '500'

message number: 2
....
data length: 3
00000000: 3530 30                                '500'

message number: 3
....
data length: 3
00000000: 3530 30                                '500'

no message available : TQ CompCd=02 ReasonCd=2033

$ mqpgef -qm TESTQM -q TQ -d input -l 500
put input/input1.txt
put input/input2.txt
put input/input3.txt
$ mqpgef -qm TESTQM -q TQ -r
message length: 500
....
message length: 500
....
message length: 500
....
```

no message available : TQ CompCd=02 ReasonCd=2033

PUT/GETするメッセージの数の指定 (-n)

PUT/GETするメッセージの数を指定します。-i (PUT/GETするメッセージの間隔の指定) /-m (メッセージ) /-mx (メッセージ16進表記) /-f (ファイルをPUT) /-d (ディレクトリ内のファイルをPUT) /-g (ディレクトリへ出力) オプションなどと共に使用可能です。

PUT時 :

```
mqqpgf -qm <qmgr> -q <queue> -m <input message> -n <count>
mqqpgf -qm <qmgr> -q <queue> -mx <input message(hexadecimal notation)> -n <count>
mqqpgf -qm <qmgr> -q <queue> -f <filename> -n <count>
mqqpgf -qm <qmgr> -q <queue> -d <directory> -n <count>
```

GET時 :

```
mqqpgf -qm <qmgr> -q <queue> -n <count>
mqqpgf -qm <qmgr> -q <queue> -g <directory> -n <count>
```

Ex. 5.1.8 -dオプションと共に使用した例

```
$ mqqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(Queue) CURDEPTH(0)
```

```
$ ls -l input
total 24
-rw-r--r--  1 mq80      mqm           262 Oct 13 21:00 input1.txt
-rw-r--r--  1 mq80      mqm           262 Oct 13 21:00 input2.txt
-rw-r--r--  1 mq80      mqm           262 Oct 13 21:00 input3.txt
```

```
$ mqqpgf -qm TESTQM -q TQ -d input -n 3
[16/12/22 20:41:24] 1: put from input/test1.txt
[16/12/22 20:41:24] 2: put from input/test1.txt
[16/12/22 20:41:24] 3: put from input/test1.txt
[16/12/22 20:41:24] 1: put from input/test2.txt
[16/12/22 20:41:24] 2: put from input/test2.txt
[16/12/22 20:41:24] 3: put from input/test2.txt
[16/12/22 20:41:24] 1: put from input/test3.txt
[16/12/22 20:41:24] 2: put from input/test3.txt
```

```
[16/12/22 20:41:24] 3: put from input/test3.txt
```

```
> mqpcf ques -qm TESTQM -q TQ CURDEPTH
1: QUEUE(TQ) TYPE(Queue) CURDEPTH(9)
-----
```

PUT/GETするメッセージの間隔の指定 (-i)

PUT/GETするメッセージの間隔をms単位で指定します。-n (PUT/GETするメッセージの数の指定) または -d (ディレクトリ内のファイルをPUT) が指定されて、複数メッセージがPUT/GETされる場合に有効です。

PUT時 :

```
mqpgf -qm <qmgr> -q <queue> -m <input message> -n <count> -i <interval(ms)>
mqpgf -qm <qmgr> -q <queue> -m <input message> -d <directory> -i <interval(ms)>
```

GET時 :

```
mqpgf -qm <qmgr> -q <queue> -n <count> -i <interval(ms)>
mqpgf -qm <qmgr> -q <queue> -g <directory> -n <count> -i <interval(ms)>
```

GETするメッセージの最大サイズの指定 (-sz)

メッセージをGETする際のバッファサイズを指定します。デフォルトは12KBです。デフォルトよりも小さい値も、大きい値も指定可能です。受信するメッセージより指定したサイズが小さい場合、MQRC_TRUNCATED_MSG_FAILED でGETが失敗します。ただし、MQGMO_ACCEPT_TRUNCATED_MSG を同時に指定した場合は、指定したバッファサイズ以降が切り捨てられてGETされます。

Ex. 5.1.9 受信バッファをデフォルトより大きくしてGETする例

```
-----
$ mqpgf -qm TESTQM -q TQ -m "put 1Mbyte" -l 1000000
[16/12/22 20:48:15] 1: message length: 1000000 put message : put 1Mbyt
e.....
.....

$ mqpgf -qm TESTQM -q TQ
MQGET with warning : TQ CompCd=01 ReasonCd=2080 len=1000000
```

```
$ mqrc 2080
```

```
2080 0x00000820 MQRC_TRUNCATED_MSG_FAILED
```

```
$ mqpgf -qm TESTQM -q TQ -sz 1000000
```

```
[16/12/22 20:48:57] 1: message length: 1000000 get message : put 1Mbyt
```

```
e.....  
.....  
-----
```

Ex. 5.1.10 受信バッファをデフォルトより小さく指定する例

※MQGMO_ACCEPT_TRUNCATED_MSGを指定して受信バッファに収まらないデータを切り捨てる。

```
$ mqpgf -qm TESTQM -q TQ -m "put 1Mbyte" -l 1000000
```

```
[16/12/22 20:48:20] 1: message length: 1000000 put message : put 1Mbyt
```

```
e.....  
.....
```

```
$ mqpgf -qm TESTQM -q TQ -sz 10 MQGMO_ACCEPT_TRUNCATED_MSG
```

```
MQGET with warning : TQ CompCd=01 ReasonCd=2079 len=1000000
```

```
message length: 1000000
```

```
get message : put 1Mbyte
```

```
$ mqrc 2079
```

```
2079 0x0000081f MQRC_TRUNCATED_MSG_ACCEPTED
```

```
$ mqpcf ques -qm TESTQM -q TQ CURDEPTH
```

```
1: QUEUE(TQ) TYPE(Queue) CURDEPTH(0)
```

```
-----
```

標準出力への書き込みサイズの指定 (-ds)

GETしたメッセージを標準出力へ書き込む場合、およびPUTするメッセージをコマンドラインで指定する場合の標準出力へのエコーバックのデフォルトの最大サイズは128バイトです。GET時およびPUT時の表示文字数を調節することができます。また、メッセージ本来のサイズ分すべてを表示させることもできます。

```
mqpgf -qm <qmgr> -q <queue> -sz 100000 -ds <display size or all>
mqpgf -qm <qmgr> -q <queue> -m "large message" -l 100000 -ds <display size or al
1>
```

Ex. 5.1.11 PUT/GET時に大きなサイズのメッセージの表示サイズを調節

※1Mbyteのメッセージを表示サイズを256ByteでPUT/GETする。

```
$ mqpgf -qm TESTQM -q TQ -m "put 1Mbyte" -l 1000000 -ds 256
```

```
[16/12/22 21:24:09] 1: message length: 1000000 put message : put 1Mbyt
```

```
e.....
.....
.....
.....
```

```
$ mqpgf -qm TESTQM -q TQ -sz 1000000 -ds 256
```

```
[16/12/22 21:24:15] 1: message length: 1000000 get message : put 1Mbyt
```

```
e.....
.....
.....
.....
```

Ex. 5.1.12 受信したメッセージの全てを表示

※256byteのメッセージをそのままのサイズで表示させる。

```
$ mqpgf -qm TESTQM -q TQ -m "put 256byte" -l 256 -ds all
```

```
[16/12/22 21:39:25] 1: message length: 256 put message : put 256byt
```

```
e.....
.....
.....
.....
```

```
$ mqpgf -qm TESTQM -q TQ -ds all
```

```
[16/12/22 21:39:33] 1: message length: 256 get message : put 256byt
```

```
e.....
.....
.....
.....
```

キュー上のメッセージのブラウズ／ダンプ（通常）（-br）

キュー上のメッセージをブラウズし、結果を16進ダンプ表示します。MQMDに関するフィールド毎に表示します。-r(キュー上の全てのメッセージをGET)と共に使用可能です。（「4. 基本的なテストの実施方法 - キュー上のメッセージのブラウズ／ダンプ（通常）」参照）

```
mqpgf -qm <qmgr> -q <queue> -br -r
```

キュー上のメッセージのブラウズ／ダンプ（詳細）（-brv）

キュー上のメッセージをブラウズし、結果を16進ダンプ表示します。MQMD(独立メッセージ記述子)、MQXQH(伝送キュー・ヘッダー)、メッセージのMQMD(組み込みMQMD)、MQMDE(拡張メッセージ記述子)、MQDLH(送達不能ヘッダー)、MQRFH2(規則および書式ヘッダー 2)およびPCF(プログラマブル・コマンド・フォーマット)がフィールド毎に表示されます。-r(キュー上の全てのメッセージをGET)と共に使用可能です。

PCFのMQCFT_BYTE_STRING_FILTER、MQCFT_BYTE_STRING、MQCFT_STRING_FILTER、MQCFT_STRING_LIST、MQCFT_STRING、MQDLHの数値以外の項目、MQRFH2のNameValueDataも含む数値以外の項目については、可視文字のみ出力され、不可視文字については”.”が代替文字として表示されます。これらのフィールドは表示方法を変更することが可能です。-hex を指定した場合は16進表記、-rawを指定した場合はそのまま標準出力にその文字コードが送られます。（「4. 基本的なテストの実施方法 - キュー上のメッセージのブラウズ／ダンプ（詳細）」参照）

```
mqpgf -qm <qmgr> -q <queue> -brv -r
```

GETしたメッセージをダンプ表示（通常）（-dp）

ブラウズではなく、実際にキュー上のメッセージを読み取ること以外は、「4. 基本的なテストの実施方法 - キュー上のメッセージのブラウズ／ダンプ（通常）」と同じです。

```
mqpgf -qm <qmgr> -q <queue> -dp -r
```

GETしたメッセージをダンプ表示（詳細）（-dpv）

ブラウザではなく、実際にキュー上のメッセージを読み取ること以外は、「4. 基本的なテストの実施方法 — キュー上のメッセージのブラウザ／ダンプ（詳細）」と同じです。

```
mqpgf -qm <qmgr> -q <queue> -dpv -r
```

rawモード出力 (-raw)

ユーザーデータ部、PCFフォーマットのMQCFT_BYTE_STRING_FILTER、MQCFT_BYTE_STRING、MQCFT_STRING_FILTER、MQCFT_STRING_LIST、MQCFT_STRING、およびMQDLHの数値以外の項目、MQRFH2のNameValueDataも含む数値以外の項目については、デフォルトでは可視文字のみ出力され、不可視文字については”.”が代替文字として表示されます。-rawを指定した場合はそのまま標準出力にその文字コードが送られます。-brv(キュー上のメッセージのブラウザ／ダンプ（詳細）)と共に指定できます。

GETしたデータを変更せずにそのままファイルに保存したい場合にも使用します。

(「4. 基本的なテストの実施方法 — GETしたメッセージを標準出力へ書き込み(rawモード)」参照)

```
mqpgf -qm <qmgr> -q <queue> -brv -raw
```

```
mqpgf -qm <qmgr> -q <queue> -o <output file path> -raw
```

```
mqpgf -qm <qmgr> -q <queue> -raw > <output file path> (標準出力のリダイレクト)
```

16進表記で出力 (-hex)

GETしたメッセージを16進表記で出力します。(「4. 基本的なテストの実施方法 — GETしたメッセージを標準出力へ書き込み(16進表記)」参照)

```
mqpgf -qm <qmgr> -q <queue> -hex -r
```

指定MQI呼び出し実行前で停止 (-s)

指定された任意のMQI関数が呼び出される直前で停止します。任意のキーの入力で処理が再開されます。

指定可能なMQI関数は、MQCONN, MQCONN, MQPUT, MQOPEN, MQGET, MQSET, MQINQ, MQCLOSE, MQDISC, MQBACK, MQCMIT, MQCRTMH, MQDLTMH, MQSETMP, MQINQMP です。

MQCONNX は-x オプションでクライアント接続を行う場合、および MQCNO_NONE 以外の MQCNO_* オプションを指定した場合に使用されます。

MQCMIT は MQPMO_SYNCPOINT、または MQGMO_SYNCPOINT を指定した場合に呼び出されます。

MQBACK は MQPMO_SYNCPOINT、または MQGMO_SYNCPOINT および「強制バックアウト (-b)」を指定した場合に呼び出されます。

「キュー上の全てメッセージをGET (-r)」とMQGMO_SYNCPOINT、または「PUT/GETするメッセージの数の指定 (-n)」とMQPMO_SYNCPOINTを共に使用する場合、MQCMIT/MQBACKの呼び出しは1回 (同一UOW) になります。

MQCRTMH/MQDLTMH は「メッセージプロパティの指定 (-smp)」してPUTする場合、および MQGMO_PROPERTIES_IN_HANDLE を指定してGETする際に使用されます。

MQSETMP は「メッセージプロパティの指定 (-smp)」してPUTする際に使用されます。

MQINQMP は MQGMO_PROPERTIES_IN_HANDLE を指定してGETする際に使用されます。

```
mqpgf -qm <qmgr> -q <queue> -s <MQI function name> -r
```

Ex. 5.1.13 -s MQGET を -rと共に指定した場合

```
-----
$ mqpgf -qm TESTQM -q TQ -m "sample message" -n 2
[16/12/22 21:31:19] 1: message length: 14 put message : sample message
[16/12/22 21:31:19] 2: message length: 14 put message : sample message
$ mqpgf -qm TESTQM -q TQ -s MQGET -r
stop before calling MQGET().
Hit Any Key!!!
[16/12/22 21:31:26] 1: message length: 14 get message : sample message
stop before calling MQGET().
Hit Any Key!!!
[16/12/22 21:31:28] 2: message length: 14 get message : sample message
stop before calling MQGET().
Hit Any Key!!!
no message available : TQ CompCd=02 ReasonCd=2033
-----
```

プロセス名の指定 (-p)

-inq (MQINQ呼び出し) と一緒に指定します。MQINQで属性を照会したいプロセスの名前を指定します。MQOT_PROCESSの指定も必要です。（「4. 基本的なテストの実施方法 - MQINQ呼び出し」参照）

```
mqpgf -qm <qmgr> -p <process> -inq: selector(e.g. MQCA_APPL_ID, MQCA_ENV_DATA,...) MQOT_PROCESS
```

ネームリスト名の指定 (-nl)

-inq (MQINQ呼び出し) と一緒に指定します。MQINQで属性を照会したいネームリストの名前を指定します。MQOT_NAMELISTの指定も必要です。（「4. 基本的なテストの実施方法 — MQINQ呼び出し」参照）

```
mqpgf -qm <qmgr> -nl <namelist> -inq: selector(e.g. MQIA_NAMELIST_TYPE, MQCA_NAMES,...) MQOT_NAMELIST
```

PCFフォーマット・メッセージをPUT (-pcf)

プレーンテキストファイルに下記のPCFの定義を記述することで、バイナリのPCFフォーマットを作成し指定キューへPUTできます。-n (PUT/GETするメッセージの数の指定)、-i (PUT/GETするメッセージの間隔の指定) と共に指定可能です。（「4. 基本的なテストの実施方法 — PCFフォーマット・メッセージをPUT」参照）

以降のパラメータを2次側に切り替える (-ss)

-oq (出力キュー) または-iq (入力キュー) と共に使用します。このオプションを指定以降に設定したオプション／パラメーターは、-oq または -iq で指定したキューに対して使用されます。

以降のパラメータを1次側に戻す (-sp)

-oq (出力キュー) または-iq (入力キュー) と共に使用します。このオプションを指定以降に設定したオプション／パラメーターは、-qで指定したキューに対して使用されます。デフォルトでは設定したオプションは全て -q に対して有効です。このオプションは -ss の効果をデフォルトに戻します。

送信MsgIdをCorrelIdに持つメッセージをGET (-mc)

"-iq" (入力キュー名) と共に使用します。送信メッセージのMQMD.MsgIdと同じCorrelIdを持つメッセージをGETします。このオプションを指定時は、MQMO_MATCH_CORREL_ID オプションが自動的に付加されます。(MQGMO_DEFAULTのMatchOptionsはMQMO_MATCH_MSG_ID+MQMO_MATCH_CORREL_IDです。)

受信MQMDを送信メッセージに引き継ぐ (-im)

"-oq" (出力キュー名) と共に使用します。受信メッセージのMQMDを送信メッセージへコピーします。但し、コンテキスト情報はこのオプションでは引き継がれません。コンテキスト情報を引き継ぐ場合は、このオプションは不要です。詳細は「Ex. 6.9.2 入力の識別、起点、ユーザー・コンテキストを引き継いでリキューする例」を参照してください。

セグメント・サイズ (-as)

アプリケーション (キューマネージャーでなく) でセグメンテーションを指定サイズで行うように指示します。バイト単位で指定します。詳細は「Ex. 4.21.1 アプリケーションによるセグメント化」を参照してください。

論理メッセージのデリミタ (-dl)

"-m"または"-f"で指定したメッセージから複数の論理メッセージを作成します。デリミタは文字列または16進表記で指定します。作成された論理メッセージにはデリミタは含まれません。詳細は「Ex. 4.24.1 論理メッセージのグループ化」を参照してください。

起動スレッド数 (-nt)

本パラメータで起動するスレッド数を指定すると、メイン・スレッドで接続ハンドルを作成し、指定した数の子スレッドでそのハンドルを使用してMQDISC()以外のMQI()を呼び出します。

Ex. 5.1.14 起動スレッド数の指定

※スレッド間で接続を共用するには、MQCNO_HANDLE_SHARE_BLOCKまたはMQCNO_HANDLE_SHARE_NO_BLOCKを指定して、「共用（スレッド独立）接続」を作成する必要があります。mqpgfはMQCNO_HANDLE_SHARE_NO_BLOCKを指定すると、2219 MQRC_CALL_IN_PROGRESSが返却された場合、MQIの呼び出しをリトライしません。その為、マルチ・スレッドでのMQI呼び出しを指定する場合は、実際にはMQCNO_HANDLE_SHARE_BLOCKのみが有効です。

```
$ mqpgf -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnn) MQCNO_CLIENT_BINDING -nt 3 -tr MQCNO_HANDLE_SHARE_BLOCK
[2018/07/26 18:19:10.208 tid=0] MQCONN start qmgr:SampleQM Options:0x00000840
[2018/07/26 18:19:10.333 tid=0] MQCONN stop hcon:33554437 qmgr:SampleQM CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.333 tid=31108] MQOPEN start hcon:33554437 ObjectName:SampleQ Options:0x00000010
[2018/07/26 18:19:10.333 tid=38164] MQOPEN start hcon:33554437 ObjectName:SampleQ Options:0x00000010
[2018/07/26 18:19:10.333 tid=38156] MQOPEN start hcon:33554437 ObjectName:SampleQ Options:0x00000010
[2018/07/26 18:19:10.333 tid=31108] MQOPEN stop hcon:33554437 ObjectName:SampleQ CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.333 tid=31108] 1: message length: 4 put message: test
[2018/07/26 18:19:10.349 tid=38164] MQOPEN stop hcon:33554437 ObjectName:SampleQ CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.349 tid=31108] MQPUT start hcon:33554437 Options:0x00000000
[2018/07/26 18:19:10.349 tid=38156] MQOPEN stop hcon:33554437 ObjectName:SampleQ CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.349 tid=38164] 1: message length: 4 put message: test
[2018/07/26 18:19:10.365 tid=38164] MQPUT start hcon:33554437 Options:0x00000000
[2018/07/26 18:19:10.365 tid=38156] 1: message length: 4 put message: test
[2018/07/26 18:19:10.365 tid=38156] MQPUT start hcon:33554437 Options:0x00000000
[2018/07/26 18:19:10.380 tid=31108] MQPUT stop hcon:33554437 CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.380 tid=31108] MQCLOSE start hcon:33554437 Options:0x00000000
[2018/07/26 18:19:10.396 tid=38164] MQPUT stop hcon:33554437 CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.396 tid=38164] MQCLOSE start hcon:33554437 Options:0x00000000
[2018/07/26 18:19:10.396 tid=38156] MQPUT stop hcon:33554437 CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.396 tid=38156] MQCLOSE start hcon:33554437 Options:0x00000000
```

```

00
[2018/07/26 18:19:10.412 tid=31108] MQCLOSE stop hcon:33554437 CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.427 tid=38164] MQCLOSE stop hcon:33554437 CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.443 tid=38156] MQCLOSE stop hcon:33554437 CompCd=00 ReasonCd=00
[2018/07/26 18:19:10.458 tid=0] MQDISC start hcon:33554437
[2018/07/26 18:19:10.490 tid=0] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
-----

```

起動スレッド数(スレッド内でMQCONN/MQDISC呼び出し) (-ni)

本パラメータで起動するスレッド数を指定すると、メイン・スレッドから起動された子スレッド内でMQCONN()/MQCONNX()およびMQDISC()を呼び出します。この場合は、「共用(スレッド独立)接続」を作成する(MQCNO_HANDLE_SHARE_BLOCKを指定する)必要はありません。

Ex. 5.1.15 起動スレッド数の指定(スレッド内でMQCONN/MQDISC呼び出し)

```

-----
$ mqpgf -qm SampleQM -q SampleQ -m "thread test" -ni 3 -tr MQPMO_SYNCPOINT
[18/08/07 17:56:12.161562 tid=44144] MQCONN start qmgr:SampleQM
[18/08/07 17:56:12.163679 tid=47424] MQCONN start qmgr:SampleQM
[18/08/07 17:56:12.163723 tid=50704] MQCONN start qmgr:SampleQM
[18/08/07 17:56:12.284938 tid=44144] MQCONN stop hcon:20971526 qmgr:SampleQM CompCd=00 ReasonCd=00
[18/08/07 17:56:12.285045 tid=44144] MQOPEN start hcon:20971526 ObjectName:SampleQ Options:0x00000010
[18/08/07 17:56:12.285394 tid=44144] MQOPEN stop hcon:20971526 ObjectName:SampleQ CompCd=00 ReasonCd=00
[18/08/07 17:56:12.285453 tid=44144] 1: message length: 11 put message: thread test
[18/08/07 17:56:12.285557 tid=44144] MQPUT start hcon:20971526 Options:0x00000000
[18/08/07 17:56:12.300966 tid=47424] MQCONN stop hcon:20971528 qmgr:SampleQM CompCd=00 ReasonCd=00
[18/08/07 17:56:12.301005 tid=47424] MQOPEN start hcon:20971528 ObjectName:SampleQ Options:0x00000010
[18/08/07 17:56:12.308055 tid=50704] MQCONN stop hcon:20971530 qmgr:SampleQM Com

```

pCd=00 ReasonCd=00
 [18/08/07 17:56:12.308091 tid=50704] MQOPEN start hcon:20971530 ObjectName:SampleQ Options:0x00000010
 [18/08/07 17:56:12.308375 tid=44144] MQPUT stop hcon:20971526 CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.308421 tid=44144] MQCMIT start hcon:20971526
 [18/08/07 17:56:12.308476 tid=47424] MQOPEN stop hcon:20971528 ObjectName:SampleQ CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.308536 tid=47424] 1: message length: 11 put message: thread test
 [18/08/07 17:56:12.308581 tid=47424] MQPUT start hcon:20971528 Options:0x00000000
 [18/08/07 17:56:12.308634 tid=50704] MQOPEN stop hcon:20971530 ObjectName:SampleQ CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.308673 tid=50704] 1: message length: 11 put message: thread test
 [18/08/07 17:56:12.308707 tid=50704] MQPUT start hcon:20971530 Options:0x00000000
 [18/08/07 17:56:12.308753 tid=47424] MQPUT stop hcon:20971528 CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.308787 tid=47424] MQCMIT start hcon:20971528
 [18/08/07 17:56:12.308830 tid=50704] MQPUT stop hcon:20971530 CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.308870 tid=50704] MQCMIT start hcon:20971530
 [18/08/07 17:56:12.309450 tid=44144] MQCMIT stop hcon:20971526 CompCd=00 ReasonCd=00
 MQCMIT success : CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.309516 tid=44144] MQCLOSE start hcon:20971526 Options:0x00000000
 [18/08/07 17:56:12.309560 tid=47424] MQCMIT stop hcon:20971528 CompCd=00 ReasonCd=00
 MQCMIT success : CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.309618 tid=47424] MQCLOSE start hcon:20971528 Options:0x00000000
 [18/08/07 17:56:12.309660 tid=50704] MQCMIT stop hcon:20971530 CompCd=00 ReasonCd=00
 MQCMIT success : CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.309716 tid=50704] MQCLOSE start hcon:20971530 Options:0x00000000
 [18/08/07 17:56:12.309758 tid=44144] MQCLOSE stop hcon:20971526 CompCd=00 ReasonCd=00
 [18/08/07 17:56:12.309791 tid=44144] MQDISC start hcon:20971526

```

[18/08/07 17:56:12.309835 tid=47424] MQCLOSE stop hcon:20971528 CompCd=00 Reason
Cd=00
[18/08/07 17:56:12.309866 tid=47424] MQDISC start hcon:20971528
[18/08/07 17:56:12.315516 tid=50704] MQCLOSE stop hcon:20971530 CompCd=00 Reason
Cd=00
[18/08/07 17:56:12.315556 tid=50704] MQDISC start hcon:20971530
[18/08/07 17:56:12.316555 tid=44144] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/08/07 17:56:12.317559 tid=47424] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/08/07 17:56:12.321606 tid=50704] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00

$ mqpgf -qm SampleQM -q SampleQ -ni 3 -tr MQGMO_SYNCPOINT
[18/08/07 17:56:19.812864 tid=44144] MQCONN start qmgr:SampleQM
[18/08/07 17:56:19.814992 tid=47424] MQCONN start qmgr:SampleQM
[18/08/07 17:56:19.815036 tid=50704] MQCONN start qmgr:SampleQM
[18/08/07 17:56:19.936089 tid=44144] MQCONN stop hcon:20971526 qmgr:SampleQM Com
pCd=00 ReasonCd=00
[18/08/07 17:56:19.936178 tid=44144] MQOPEN start hcon:20971526 ObjectName:Sampl
eQ Options:0x00000001
[18/08/07 17:56:19.936448 tid=44144] MQOPEN stop hcon:20971526 ObjectName:Sample
Q CompCd=00 ReasonCd=00
[18/08/07 17:56:19.936496 tid=44144] MQGET start hcon:20971526 Options:0x0000000
0
[18/08/07 17:56:19.936588 tid=47424] MQCONN stop hcon:20971528 qmgr:SampleQM Com
pCd=00 ReasonCd=00
[18/08/07 17:56:19.936624 tid=47424] MQOPEN start hcon:20971528 ObjectName:Sampl
eQ Options:0x00000001
[18/08/07 17:56:19.936696 tid=50704] MQCONN stop hcon:20971530 qmgr:SampleQM Com
pCd=00 ReasonCd=00
[18/08/07 17:56:19.936731 tid=50704] MQOPEN start hcon:20971530 ObjectName:Sampl
eQ Options:0x00000001
[18/08/07 17:56:19.949521 tid=44144] MQGET stop hcon:20971526 CompCd=00 ReasonCd
=00
[18/08/07 17:56:19.949560 tid=44144] 1: message length: 4 get message : test
[18/08/07 17:56:19.949636 tid=44144] MQCMIT start hcon:20971526
[18/08/07 17:56:19.949686 tid=47424] MQOPEN stop hcon:20971528 ObjectName:Sample
Q CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949757 tid=47424] MQGET start hcon:20971528 Options:0x0000000
0
[18/08/07 17:56:19.949820 tid=50704] MQOPEN stop hcon:20971530 ObjectName:Sample
Q CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949866 tid=50704] MQGET start hcon:20971530 Options:0x0000000
0

```



```

[18/08/07 17:56:19.949924 tid=47424] MQGET stop hcon:20971528 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.949959 tid=47424] 1: message length: 4 get message : test
[18/08/07 17:56:19.949996 tid=47424] MQCMIT start hcon:20971528
[18/08/07 17:56:19.950041 tid=50704] MQGET stop hcon:20971530 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950083 tid=50704] 1: message length: 4 get message : test
[18/08/07 17:56:19.950118 tid=50704] MQCMIT start hcon:20971530
[18/08/07 17:56:19.950521 tid=44144] MQCMIT stop hcon:20971526 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950600 tid=44144] MQCLOSE start hcon:20971526 Options:0x00000000
[18/08/07 17:56:19.950649 tid=47424] MQCMIT stop hcon:20971528 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950714 tid=47424] MQCLOSE start hcon:20971528 Options:0x00000000
[18/08/07 17:56:19.950761 tid=50704] MQCMIT stop hcon:20971530 CompCd=00 ReasonCd=00
MQCMIT success : CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950825 tid=50704] MQCLOSE start hcon:20971530 Options:0x00000000
[18/08/07 17:56:19.950871 tid=44144] MQCLOSE stop hcon:20971526 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950908 tid=44144] MQDISC start hcon:20971526
[18/08/07 17:56:19.950955 tid=47424] MQCLOSE stop hcon:20971528 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.950989 tid=47424] MQDISC start hcon:20971528
[18/08/07 17:56:19.956575 tid=50704] MQCLOSE stop hcon:20971530 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.956619 tid=50704] MQDISC start hcon:20971530
[18/08/07 17:56:19.957617 tid=44144] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.958623 tid=47424] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
[18/08/07 17:56:19.962197 tid=50704] MQDISC stop hcon:-1 CompCd=00 ReasonCd=00
-----

```

API トレース (-tr)

MQI呼び出しに対しての簡易APIトレースを表示します。“-nt”や“-ni”を指定してマル

チ・スレッドで実行した場合は、MQIが呼び出されたスレッドID ("tid") も一緒に表示します。"start"、"stop"の表記は、それぞれMQIの呼び出し開始時、完了時を意味します。開始時には指定されたオプション、完了時には完了コード、理由コードも表示します。

処理開始の同期用ファイル名 (-sf)

このオプションを使用すると、全てのMOCONN()またはMQCONN()とその後の入出力キューのオープン後に、指定されたファイルをオープンしてその存在を1ms間隔でチェックし、ファイルのオープン成功後に処理を開始します。これは、複数のmqpgf(c)プロセスを起動または-qmオプションに複数のキューマネージャーを指定した時に、キューマネージャーへの接続および入出力キューのオープン後から処理開始の同期をとりたい場合に使用できます。プロセスの起動に時間がかかり、処理の開始にばらつきが発生する一部の環境でパフォーマンス・テストを実施する場合等に有用です。本オプションは、「起動スレッド数(スレッド内でMQCONN/MQDISC呼び出し) (-ni)」を指定した場合は無効です。

接続繰り返し回数 (-c)

MQCONN(X)0からMQDISC0までの繰り返し回数です。「接続繰り返し時の待ち時間 (-wp)」と共に使用すると、その繰り返しの間隔を指定できます。

MQDISC呼び出しの省略 (-sd)

このオプションを指定すると、MQDISC0呼び出しが省略されます。

接続繰り返し時の待ち時間 (-wp)

「接続繰り返し回数 (-c)」と共に使用した場合は、MQDISC0から次のMQCONN(X)0までのインターバル時間、「接続リトライの回数 (-cr)」と共に使用した場合はMQCONN(X)0のリトライ間隔になります。ms単位で指定します。

MQI失敗後も処理を継続 (-ca)

基本的に「接続繰り返し回数 (-c)」と共に使用します。このオプションを指定すると、各MQIが失敗した場合でも、MQCONN0から始まる次の繰り返しを継続します。

送信メッセージにカウンターを付加 (-ac)

「入力メッセージ (-m)」と共に使用します。このオプションを指定すると、各送信メッセージの先頭に自動的に8桁のメッセージ・カウンタ ("nnnnnnnn: ") が付加されます。

接続リトライの回数 (-cr)

MQCONN0が失敗した場合のMQCONN0の再呼び出しの回数です。「接続繰り返し時の待ち時間 (-wp)」で指定した間隔で再接続が試行されます。

Thread内MQI呼び出し後CPUの使用権を放棄 (-y)

Thread内で任意のMQIが呼び出された後、一旦CPUの使用権を放棄します。Windowsの場合はこの指定は無視されます。

ファイルまたはディレクトリのオーバーライト確認省略 (-nm)

GETしたメッセージをファイル(-o)またはディレクトリ(-g)に書き込み時、既にファイルまたはディレクトリが存在する場合、オーバーライト確認が省略されます。

ディレクトリに出力するファイル名の拡張子を指定 (-ext)

受信メッセージのディレクトリ出力時、MQMDのPUT Date/Timeがファイル名に使用されます。そのファイルの拡張子を指定します。

RFHヘッダを強制的に削除 (-rh)

メッセージにRFHヘッダが付加されている場合、それを強制的に削除します。RFHヘッダがユーザー・データの先頭にある場合(ユーザー・データの先頭にMQRFH_STRUC_ID "RFH "がある場合)に機能します。MQMD.FormatにMQFMT_RF_HEADER_1 "MQHRF "またはMQFMT_RF_HEADER_2 "MQHRF2 "が設定されていなくても強制的に除去します。

標準出力/エラーをフラッシュ (-ff)

コマンドの実行結果を表示する度に標準出力、標準エラーをフラッシュします。リダイレクトしたファイルI/Oが遅延するシステム等に有効です。

特定のMQIまたはTNS関数の呼び出しをスキップする (-sk)

コマンド実行時、指定したAPIの呼び出しをスキップします。下表に示すAPIを指定可能です。

表 5.1.1 スキップ可能(-sk に指定可能)な関数	
API	摘要
MQCONN	
MQCONNX	
MQDISC	
MQPUT	
MQOPEN	
MQCLOSE	
MQGET	
MQPUT	
MQCMIT	
MQBACK	
MQSET	

表 5.1.1 スキップ可能(-sk に指定可能)な関数

API	摘要
MQINQ	
MQCRTMH	
MQDLTMH	
MQSETMP	
MQINQMP	
MQCB	
BEGINTRANSACTION	HPE-NonStopのみ
ENDTRANSACTION	HPE-NonStopのみ
ABORTTRANSACTION	HPE-NonStopのみ
PUT_BEGINTRANSACTION	HPE-NonStopのみ
PUT_ENDTRANSACTION	HPE-NonStopのみ
PUT_ABORTTRANSACTION	HPE-NonStopのみ
PUT_RESUMETRANSACTION	HPE-NonStopのみ

最大未コミットメッセージ数の指定 (-u)

未コミットのメッセージがこの値に達するとMQCOMITまたはENDTRANSACTIONが呼び出されます。デフォルトでは、MQPMO/MQGMO_SYNCPOINTの場合、全てのPUT/GETを完了するまでMQCOMITまたはENDTRANSACTION（HPE NonStop）は呼び出されません。

5.2 プラットフォーム固有のオプション

グローバル作業単位の使用 (NSK) (-gt)

HPE NonStopでのみ使用可能なオプションです。シングル・スレッド版 (mqpgfs, mqpgfcs) の場合、TMFのトランザクションAPIである、BEGINTRANSACTION/ENDTRANSACTION/ABORTTRANSACTION、マルチ・スレッド版 (mqpgf, mqpgfc) の場合はPUT_BEGINTRANSACTION/PUT_ENDTRANSACTION/PUT_ABORTTRANSACTIONが使用されます。複数QMGRを指定した場合も全体の処理で1回のみTMF APIが呼び出されます。

グローバル作業単位の使用 (MQI呼び出し毎) (NSK) (-gti)

HPE NonStopでのみ使用可能なオプションです。シングル・スレッド版 (mqpgfs, mqpgfcs) の場合、TMFのトランザクションAPIである、BEGINTRANSACTION/ENDTRANSACTION/ABORTTRANSACTION、マルチ・スレッド版 (mqpgf, mqpgfc) の場合はPUT_BEGINTRANSACTION/PUT_ENDTRANSACTION/PUT_ABORTTRANSACTIONが使用されます。複数QMGRを指定した場合は、QMGR毎にTMF APIが呼び出されます。-nまたは-rと共に-oq (出力キュー名) (Queue to Queue) が指定された場合、MQGETからMQPUTの度にBEGINTRANSACTION/ENDTRANSACTIONが呼び出されます。-nと共に-iq (入力キュー名) (Send and Receive) が指定された場合、MQPUTおよびMQGET毎にBEGINTRANSACTION/ENDTRANSACTIONが呼び出されます。

5.3 特定のMQIの指定

MQSET (-set)

指定キューに対し、MQSET()を呼び出します。複数のセレクターとアトリビュートの組み合わせを指定することができます。

MQSETは、キューに対してのみ属性の変更が可能です。プロセス、キューマネージャーなど、他のオブジェクトの属性は変更できません。

また、モデルキューは変更できず、クラスター・キューの場合はローカル・インスタンスである必要があります。（「4. 基本的なテストの実施方法 — MQSET呼び出し」参照）

MQINQ (-inq)

MQINQ()を呼び出し、指定キュー（ローカル、リモート、エイリアス）、ネームリスト、プロセス、キューマネージャーのアトリビュートを照会することができます。

（「4. 基本的なテストの実施方法 — MQINQ呼び出し」参照）

MQSETMP (-smp)

任意のデータ型のメッセージプロパティを指定してメッセージをPUTすることが可能です。（「4. 基本的なテストの実施方法 — メッセージプロパティの指定」参照）

5.4 MQCDのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

ConnectionNameの指定（-x）（MQCHAR264）（-）

MQCDチャネル定義構造体の ConnectinName を指定します。（「3. 使用方法の説明
ー クライアント接続の使用」参照）

ChannelNameの指定（-ch）（MQCHAR20）（-）

MQCDチャネル定義構造体の ChannelName を指定します。指定しない場合のデフォルトは SYSTEM.DEF.SVRCONN です。（「3. 使用方法の説明
ー クライアント接続の使用」参照）

LocalAddressの指定（-la）（MQCHAR48）（-）

MQCDチャネル定義構造体の LocalAddress を指定します。この値を指定すると、mqpgfは例外的にMQCD_VERSION_7をMQCD.Versionに自動的に設定します。

CertificateLabel（-cl）（MQCHAR64）（-）

MQCDチャネル定義構造体の CertificateLabel を指定します。MQCD_VERSION_11以上を指定することが必要です。クライアント・モードでSSL接続を試行させる際に有効です。

SSLCipherSpec（-cs）（MQCHAR32）（-）

MQCDチャネル定義構造体の SSLCipherSpec を指定します。MQCD_VERSION_7以上を指定することが必要です。クライアント・モードでSSL接続を試行させる際に有効です。

SSLPeerName (-er) (-) (-)

SSL Peerの名前を検証する為の文字列をMQ_SSL_PEER_NAME_LENGTH(1024)バイト以内で指定します。mqpgfは指定された文字列から、MQCDチャネル定義構造体の mqcd.SSLPeerNamePtr、SSLPeerNameLength を自動的に設定します。MQCD_VERSION_7以上を指定する必要があります。クライアント・モードでSSL接続を試行させる際に有効です。

5.5 MQMDのフィールド

ここでは、MQMDのフィールドの内、コンスタント(MQFB_**など)ではなく、任意の値(数値、文字列、バイナリ)を指定するフィールドが対象です。コンスタントを指定するMQMDのフィールドについては後述します。

データタイプがMQBYTEnnの場合は、文字列および16進表記で指定できます。16進表記で指定したい場合は先頭に0xを付加し、必ず偶数桁を指定します。AからFは大文字/小文字の区別はありません。指定された長さがそのバイト数(16進表記の場合はその2倍の長さ)に満たない場合は、NULL(0x00)が後続に付加されます。

Ex. 5.5.1 MsgIdとCorrelIdの指定の例

```
-----
$ mqpgf -qm TESTQM -q TQ -f input.txt -mi 0x0123456789abcdefABCDEF -ci CorrelId
[16/12/07 20:01:29] 1: put from input.txt
$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
.... MsgId[0x0123456789ABCDEF0000000000000000000000000000] CorrelId[0x436F7
272656C4964000000000000000000000000000000000000] ....
-----
```

-ui UserIdentifier、-at AccountingToken、-ap ApplIdentityData はMQPMO に MQP
MO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXT を指定しなければ、キュー
マネージャーによってその入力が無視されます。

また、MQPMO_SET_IDENTITY_CONTEXT を指定する場合はオープン・オプションの MQ00
_SET_IDENTITY_CONTEXT、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オブ
ションのMQ00_SET_ALL_CONTEXT の指定が必要となります。

Ex. 5.5.2 UserIdentifier、AccountingToken、ApplIdentityDataの指定

```
-----
※MQPMO_SET_*、MQ00_SET_*を指定しない場合
$ mqpgf -qm TESTQM -q TQ -f input.txt -ui testuser -at 0x6163636F756E74696E67746
F6B656E -ap applidentitydata
[16/12/07 20:29:37] 1: put from input.txt
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
.... UserIdentifier[mq80          ] AccountingToken[0x0332343200000000000000000000
000000000000000000000000000000000000] ApplIdentityData[
          ] ....
-----
```

※結果は指定した値がセットされず、デフォルトのままとなる。

※MQPMO_SET_IDENTITY_CONTEXTのみを指定した場合

```
$ mqpgef -qm TESTQM -q TQ -f input.txt -ui testuser -at 0x6163636F756E74696E67746
F6B656E -ap applidentitydata MQPMO_SET_IDENTITY_CONTEXT
[16/12/07 20:33:03] 1: put from input.txt
MQPUT fail : TQ CompCd=02 ReasonCd=2096
```

```
$ mqrc 2096
```

```
2096 0x00000830 MQRC_NOT_OPEN_FOR_SET_IDENT
```

※MQ00_SET_IDENTITY_CONTEXT が指定されていないため、MQRC_NOT_OPEN_FOR_SET_IDENT
で失敗する。

※MQPMO_SET_IDENTITY_CONTEXT および MQ00_SET_IDENTITY_CONTEXTを指定した場合

```
$ mqpgef -qm TESTQM -q TQ -f input.txt -ui testuser -at 0x6163636F756E74696E67746
F6B656E -ap applidentitydata MQPMO_SET_IDENTITY_CONTEXT MQ00_SET_IDENTITY_CONTE
XT
[16/12/07 20:36:47] 1: put from input.txt
> mqpgef -qm TESTQM -q TQ -br
message number: 1
.... UserIdentifier[testuser      ] AccountingToken[0x6163636F756E74696E67746F6B6
56E000000000000000000000000000000000] ApplIdentityData[applidentitydata
] ....
```

※指定した値が正しく設定される。

-pn PutApplName、-pd PutDate、-pt PutTime、-ao applorigindataは MQPMO に MQP
MO_SET_ALL_CONTEXT を指定しなければ入力が無視されます。

また、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションの MQ00_SET_
ALL_CONTEXT の指定が必要となります。

Ex. 5.5.3 PutApplName、PutDate、PutTime、ApplOriginDataの指定

```
-----
$ mqpgef -qm TESTQM -q TQ -f input.txt -pn testapl -pd 20140101 -pt 00112233 -ao
orig MQPMO_SET_ALL_CONTEXT MQ00_SET_ALL_CONTEXT MQAT_WINDOWS
[16/12/07 20:45:56] 1: put from input.txt
> mqpgef -qm TESTQM -q TQ -br
message number: 1
.... PutApplType[9] PutApplName[testapl                                ] PutDate[20140101]
PutTime[00112233] ApplOriginData[orig] ....
```

※この例では、起点コンテキストである PutApplType に MQAT_WINDOWS を指定している。
MQAT_*を指定する場合は、そのままコンスタントを指定すれば良い、-xx のような何かのオプションの後に指定する必要はありません。

注) MQPMO_SET_ALL_CONTEXT/MQOO_SET_ALL_CONTEXT を指定し、UserIdentifier、AccountingToken、PutApplTypeを指定しなかった場合は、デフォルト値は設定されない。PutApplTypeは MQAT_NO_CONTEXT(0) がセットされる。

以下、フィールド名 (オプション) (データタイプ) (デフォルト値) です。

Expiry (-ex) (MQLONG) (MQEI_UNLIMITED)

「メッセージ存続時間」を 100ms 単位で指定します。本オプション(-ex)を指定せずに直接MQEI_*を指定した場合も、MQMD.Expiryに設定されます。

Encoding (-ec) (MQLONG) (MQENC_NATIVE)

「メッセージ・データの数値エンコード」を指定します。本オプション(-ec)を指定せずに直接MQENC_*を指定した場合も、その設定したコンスタントのORがMQMD.Encodingに設定されます。本フィールドは3桁の16進表記 (12bit) でも指定可能です。16進表記で指定したい場合は先頭に0xを付加し、必ず3桁指定します。

Ex. 5.5.4 MQRFH2ヘッダのエンコーディングの指定例

※MQRFH2ヘッダのエンコーディングをMQENC_NATIVEと逆のエンディアンに指定する。
\$ mqpgf -qm TESTQM -q TQ -m "test" -ec 0x222 -re 273 -rc 1208 -rf MQFMT_STRING -fg 100 -nc 1208 -nd "test1,test22,test333" MQFMT_RF_HEADER_2
[17/01/06 20:09:54] 1: message length: 4 put message : test
※本テスト例でのMQENC_NATIVEは0x111

```
$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[546] CodedCharSetId[943] Format[MQHRF2 ] ....
....
*StrucId[RFH ] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[1208] For
```

```

mat[MQSTR  ] Flags[100] NameValueCCSID[943]
NameValueLength[8] NameValueData[test1  ]
NameValueLength[8] NameValueData[test22  ]
NameValueLength[8] NameValueData[test333 ]
data length: 76
00000000: 7465 7374          'test          '
※0x222の10進表記は546

$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[5
46] CodedCharSetId[943] Format[MQHRF2  ] ....

data length: 76
00000000: 5246 4820 0200 0000 4800 0000 1101 0000  'RFH ....H.....'
00000010: B804 0000 4D51 5354 5220 2020 6400 0000  'ク...MQSTR  d...'
00000020: B804 0000 0800 0000 7465 7374 3120 2020  '.....test1  '
00000030: 0800 0000 7465 7374 3232 2020 0800 0000  '....test22  ....'
00000040: 7465 7374 3333 3320 7465 7374          'test333 test  '
※MQRFH2のエンコーディングをMQENC_NATIVEと逆のエンディアンを指定した場合、本プログラムはMQRFH2内のMQLONGのエンディアンを変換して設定します。
注) MQRFH2のエンコーディングは先行するMQMDヘッダのEncodingであることに注意。
MQRFH2のEncodingは「NameValueData フィールドのあとに続くデータの数値エンコード」を示します。
-----

```

CodedCharSetId (-cc) (MQLONG) (MQCCSI_Q_MGR)

「メッセージ・データの文字セット ID」を指定します。本オプション(-cc)を指定せずに直接MQCCSI_*を指定した場合も、MQMD.CodedCharSetIdに設定されます。

Priority (-pr) (MQLONG) (MQPRI_PRIORITY_AS_Q_DEF)

「メッセージ優先度」を指定します。本オプション(-pr)を指定せずに直接MQPRI_*を指定した場合も、MQMD.Priorityに設定されます。

MsgId (-mi) (MQBYTE24) (MQMI_NONE_ARRAY)

「メッセージ ID」を指定します。文字列および16進表記で指定できます。本オプション(-mi)を指定せずに直接MQMI_*を指定した場合も、MQMD.MsgIdに設定されます。

CorrelId (-ci) (MQBYTE24) (MQCI_NONE_ARRAY)

「相関 ID」を指定します。文字列および16進表記で指定できます。本オプション(-ci)を指定せずに直接MQCI_*を指定した場合も、MQMD.CorrelIdに設定されます。

ReplyToQ (-rq) (MQCHAR48) ("")

「応答キューの名前」を指定します。

ReplyToQMgr (-rm) (MQCHAR48) ("")

「応答キュー・マネージャーの名前」を指定します。

UserIdentifier (-ui) (MQCHAR12) ("")

「ユーザー ID」を指定します。識別コンテキストの一部です。同時にMQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXTの指定が必要です。また、MQPMO_SET_IDENTITY_CONTEXT を指定する場合はオープン・オプションの MQ00_SET_IDENTITY_CONTEXT、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションのMQ00_SET_ALL_CONTEXT の指定が必要となります。

AccountingToken (-at) (MQBYTE32) (MQACT_NONE_ARRAY)

「アカウント・トークン」を指定します。識別コンテキストの一部です。同時にMQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXTの指定が必要です。また、MQPMO_SET_IDENTITY_CONTEXT を指定する場合はオープン・オプションの MQ00_SET_IDENTITY_CONTEXT、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションのMQ00_SET_ALL_CONTEXT の指定が必要となります。

ApplIdentityData (-ap) (MQCHAR32) ("")

「ID に関連するアプリケーション・データ」を指定します。識別コンテキストの一部です。同時にMQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXTの指定が必要です。また、MQPMO_SET_IDENTITY_CONTEXT を指定する場合はオープン・オプションの MQOO_SET_IDENTITY_CONTEXT、 MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションのMQOO_SET_ALL_CONTEXT の指定が必要となります。

PutAppName (-pn) (MQCHAR28) ("")

「メッセージを書き込んだアプリケーションの名前」を指定します。起点コンテキストの一部です。同時にMQPMO に MQPMO_SET_ALL_CONTEXT の指定が必要です。また、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションの MQOO_SET_ALL_CONTEXT の指定が必要となります。

PutDate (-pd) (MQCHAR8) ("")

「メッセージが書き込まれたときの日付」を指定します。起点コンテキストの一部です。同時にMQPMO に MQPMO_SET_ALL_CONTEXT の指定が必要です。また、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションの MQOO_SET_ALL_CONTEXT の指定が必要となります。

PutTime (-pt) (MQCHAR8) ("")

「メッセージが書き込まれたときの時刻」を指定します。起点コンテキストの一部です。同時にMQPMO に MQPMO_SET_ALL_CONTEXT の指定が必要です。また、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションの MQOO_SET_ALL_CONTEXT の指定が必要となります。

ApplOriginData (-ao) (MQCHAR4) ("")

このフィールドはアプリケーション・スイートにより定義される情報で、メッセージの発信元についての追加情報を提供するのに使用できます。起点コンテキストの一部です。

同時にMQPMO に MQPMO_SET_ALL_CONTEXT の指定が必要です。また、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションの MQOO_SET_ALL_CONTEXT の指定が必要となります。

5.6 MQMD Version 2のフィールド

本プログラムはデフォルトでMQMD_VERSION_1(MQMD_DEFAULT)を指定します。(例外として、ブラウズ時(-br/-brv)はMQMD_VERSION_2をデフォルトで使用します)。ブラウズ時以外で、MQMD Version 2のフィールドを指定する場合は、MQMD_VERSION_2 を指定する必要があります。

MQMF_MSG_IN_GROUPを指定すると GroupId、MsgSeqNumber が指定可能になり、MQMF_SEGMENT を指定すると Offset が、MQMT_REPORT を指定すると OriginalLength が指定可能となります。

Ex. 5.6.1 MQMD Version 2の指定例

※全ての MQMD Version 2 フィールドを指定する。

```
$ mqpgef -qm TESTQM -q TQ -f input.txt -gi 0x67726F757069640000000000000000000000000000000000000000000000000000 -ms 3 -of 100 -ol 1000 MQMD_VERSION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP
[16/12/23 00:11:41] 1: put from input.txt
```

```
$ mqpgef -qm TESTQM -q TQ -brv
```

....

```
GroupId[0x67726F757069640000000000000000000000000000000000000000000000000000] MsgSeqNumber[3] Offset[100] MsgFlags[10] OriginalLength[1000]
```

....

※MsgFlagsには下記指定したパラメータの OR が設定される。

```
MQMF_SEGMENT: 0x00000002
```

```
MQMF_MSG_IN_GROUP: 0x00000008
```

以下、フィールド名 (オプション) (データタイプ) (デフォルト値) です。

GroupId (-gi) (MQBYTE24) (")

「グループ ID」を指定します。同時に MQMF_MSG_IN_GROUP、MQMF_LAST_MSG_IN_GROUP、MQMF_SEGMENT、MQMF_LAST_SEGMENT、MQMF_SEGMENTATION_ALLOWED のいずれかの指定が必要です。本オプション(-gi)を指定せずに直接MQGL_*を指定した場合も、MQMD.GroupIdに設定されます。

MsgSeqNumber (-ms) (MQLONG) (1)

「グループ中の論理メッセージの順序番号」を指定します。同時に、MQMF_MSG_IN_GROUP の指定が必要です。

Offset (-of) (MQLONG) (0)

「物理メッセージのデータの、論理メッセージの開始点からの相対位置」を指定します。同時に MQMF_SEGMENT の指定が必要です。

OriginalLength (-ol) (MQLONG) (MQOL_UNDEFINED)

「レポート・メッセージが関係するメッセージ・セグメントの長さ」を指定します。同時に MQMT_REPORT の指定が必要です。

5.7 MQRFH2のフィールド

本プログラムは、MQRFH2のフィールドを一つでも指定すると、MQMDの直後にMQRFH2ヘッダーを作成します。しかし、MQMDのFormatフィールドは指定しない為、正しくMQRFH2ヘッダーを認識させるには、MQFMT_RF_HEADER_2 を同時に指定する必要があります。

本来NameValueLengthは4の倍数にすることが必要ですが、本プログラムでは自動的にブランク (0x20) を埋めて4の倍数にします。StrucLengthも自動的に設定します。

また、MQRFH2のエンコーディングをMQENC_NATIVEと逆のエンディアンを指定する場合、本プログラムはMQRFH2内のMQLONGのエンディアンを変換して設定します。MQRFH2のエンコーディングは先行するMQMDヘッダーのEncodingであることに注意してください。MQRFH2のEncodingは「NameValueData フィールドのあとに続くデータの数値エンコード」を示します。

Ex. 5.7.1 MQRFH2フィールドの指定例

※全ての MQRFH2フィールドを指定する。

Encoding : 273

CodedCharSetId : 1208

Format : MQFMT_STRING

flags : 100

NameValueCCSID : 1208 (※NameValueCCSIDはUTF-8またはUTF-16のCCSIDである1200, 1348, 17584, 1208のいずれかを指定する必要があります。)

NameValueData : "test1,test22,test333"

```
-----
$ mqpgf -qm TESTQM -q TQ -m "test" -re 273 -rc 1208 -rf MQFMT_STRING -fg 100 -nc
1208 -nd "test1,test22,test333" MQFMT_RF_HEADER_2
[16/12/23 00:22:25] 1: message length: 4 put message : test
```

```
$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[943] Format[MQHRF2 ] Priority[0] Persistence[0] MsgId[0x414D
51206F6B61716D38306120202057D9E32620003F06] CorrelId[0x00000000000000000000
00000000000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[TESTQM
] UserIdentifier[mq80 ] AccountingToken[0x03323432000000000000000000000000
00000000000000000000000000000000] ApplIdentityData[
] PutApplType[6] PutApplName[mqpgf ] PutDate[2016091
5] PutTime[08041209] ApplOriginData[ ]
```

```
GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

*StrucId[RFH ] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[1208] Format[MQSTR ] Flags[100] NameValueCCSID[1208]
NameValueLength[8] NameValueData[test1 ]
NameValueLength[8] NameValueData[test22 ]
NameValueLength[8] NameValueData[test333 ]
data length: 4
00000000: 7465 7374                                ' test                                '
-----
```

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

Encoding（-re）（MQLONG）（MQENC_NATIVE）

「NameValueData フィールドのあとに続くデータの数値エンコード」を指定します。本オプション（-re）に続けてMQENC_*を指定することも可能です。複数指定する場合は”：”で区切ります。複数指定された場合はそのORをフィールドに設定します。本オプション（-re）を指定せずにMQENC_*を指定した場合は、MQMD.Encodingに設定されます。本フィールドは3桁の16進表記（12bit）でも指定可能です。16進表記で指定したい場合は先頭に0xを付加し、必ず3桁指定します。

下表のコンスタントが指定可能です。

表 5.7.1 指定可能な MQENC_*			
コンスタント	値	設定場所	設定方法 他
MQENC_NATIVE	0x00000111 0x00000222 (Windows, Linux x86)	MQRFH2. Encoding	OR (default)
MQENC_INTEGER_UNDEFINED	0x00000000	以下同様	以下同様
MQENC_INTEGER_NORMAL	0x00000001		
MQENC_INTEGER_REVERSED	0x00000002		

表 5.7.1 指定可能な MQENC_*

コ ン ス タ ン ト	値	設 定 場 所	設 定 方 法 他
MQENC_DECIMAL_UNDEFINED	0x00000000		
MQENC_DECIMAL_NORMAL	0x00000010		
MQENC_DECIMAL_REVERSED	0x00000020		
MQENC_FLOAT_UNDEFINED	0x00000000		
MQENC_FLOAT_IEEE_NORMAL	0x00000100		
MQENC_FLOAT_IEEE_REVERSED	0x00000200		
MQENC_FLOAT_S390	0x00000300		
MQENC_FLOAT_TNS	0x00000400		
MQENC_NORMAL	MQENC_FLOAT_IEEE_NORMAL MQENC_DECIMAL_NORMAL MQENC_INTEGER_NORMAL		
MQENC_REVERSED	MQENC_FLOAT_IEEE_REVERSED MQENC_DECIMAL_REVERSED MQENC_INTEGER_REVERSED		
MQENC_S390	MQENC_FLOAT_S390 MQENC_DECIMAL_NORMAL MQENC_INTEGER_NORMAL		
MQENC_TNS	MQENC_FLOAT_TNS MQENC_DECIMAL_NORMAL MQENC_INTEGER_NORMAL		
MQENC_AS_PUBLISHED	(-1)		

Ex. 5.7.2 Encodingフィールドの指定例

下記 3 つの指定例では全て同じ値 0x222=546 が設定されます。

```
$ mqpgf -qm TESTQM -q TQ -m test -re MQENC_FLOAT_IEEE_REVERSED:MQENC_DECIMAL_REVERSED:MQENC_INTEGER_REVERSED MQFMT_RF_HEADER_2
[17/01/06 08:53:00] 1: message length: 4 put message : test
$ mqpgf -qm TESTQM -q TQ -m test -re MQENC_REVERSED MQFMT_RF_HEADER_2
```

```

[17/01/06 08:53:06] 1: message length: 4 put message : test
$ mqpgef -qm TESTQM -q TQ -m test -re 0x222 MQFMT_RF_HEADER_2
[17/01/06 08:53:11] 1: message length: 4 put message : test
$
$ mqpgef -qm TESTQM -q TQ -brv -r
message number: 1
....

*StrucId[RFH ] Version[2] StrucLength[36] Encoding[546] CodedCharSetId[943] Form
at[          ] Flags[0] NameValueCCSID[1208]
....

message number: 2
....
*StrucId[RFH ] Version[2] StrucLength[36] Encoding[546] CodedCharSetId[943] Form
at[          ] Flags[0] NameValueCCSID[1208]
....

message number: 3
....
*StrucId[RFH ] Version[2] StrucLength[36] Encoding[546] CodedCharSetId[943] Form
at[          ] Flags[0] NameValueCCSID[1208]
....

no message available : TQ CompCd=02 ReasonCd=2033
-----

```

CodedCharSetId (-rc) (MQLONG) (MQCCSI_INHERIT)

「NameValueData フィールドのあとに続くデータの文字セット ID」を指定します。本オプション(-rc)に続けてMQCCSI_*を指定することも可能です。本オプション(-rc)を指定せずにMQCCSI_*を指定した場合は、MQMD.CodedCharSetIdに設定されます。

下表のコンスタントが指定可能です。

表 5.7.2 指定可能な MQCCSI_*			
コンスタント	値	設定場所	設定方法他
MQCCSI_UNDEFINED	0	MQRFH2.CodedCharSetId	Overwrite

表 5.7.2 指定可能な MQCCSI_*

コンスタント	値	設定場所	設定方法他
MQCCSI_DEFAULT	0	以下同様	以下同様
MQCCSI_Q_MGR	0		(default)
MQCCSI_INHERIT	(-2)		
MQCCSI_EMBEDDED	(-1)		
MQCCSI_APPL	(-3)		
MQCCSI_AS_PUBLISHED	(-4)		

Format (-rf) (MQCHAR8) (MQFMT_NONE_ARRAY)

「NameValueData のあとに続くデータの形式名」を指定します。本オプション(-rf)を指定せずにMQFMT_*を指定した場合は、MQMD.Formatに設定されます。

下表のコンスタントが指定可能です。

表 5.7.3 指定可能な MQFMT_*

コンスタント	値	設定場所	設定方法他
MQFMT_NONE	" "	MQRFH2.Format	Overwrite (default)
MQFMT_ADMIN	"MQADMIN "	以下同様	以下同様
MQFMT_CHANNEL_COMPLETED	"MQCHCOM "		
MQFMT_CICS	"MQCICS "		
MQFMT_COMMAND_1	"MQCMD1 "		
MQFMT_COMMAND_2	"MQCMD2 "		
MQFMT_DEAD_LETTER_HEADER	"MQDEAD "		
MQFMT_DIST_HEADER	"MQHDIST "		
MQFMT_EMBEDDED_PCF	"MQHEPCF "		
MQFMT_EVENT	"MQEVENT "		

表 5.7.3 指定可能な MQFMT_*

コンスタント	値	設定場所	設定方法他
MQFMT_IMS	"MQIMS "		
MQFMT_IMS_VAR_STRING	"MQIMSVS "		
MQFMT_MD_EXTENSION	"MQHMDE "		
MQFMT_PCF	"MQPCF "		
MQFMT_REF_MSG_HEADER	"MQHREF "		
MQFMT_RF_HEADER	"MQHRF "		
MQFMT_RF_HEADER_1	"MQHRF "		
MQFMT_RF_HEADER_2	"MQHRF2 "		
MQFMT_STRING	"MQSTR "		
MQFMT_TRIGGER	"MQTRIG "		
MQFMT_WORK_INFO_HEADER	"MQHWIH "		
MQFMT_XMIT_Q_HEADER	"MQXMIT "		

Flags (-fg) (MQLONG) (MQRFH_NONE)

「フラグ」を指定します。指定しない場合のデフォルトはMQRFH_NONE です。上位の16ビット (MQRFH_FLAGS_RESTRICTED_MASK 0xFFFF0000 で示されるビット) はキューマネージャで使用するよう予約されているので、注意が必要です。本オプション(-fg)を指定せずに直接MQRF_*を指定した場合も、MQRFH2.Flags に設定されます。

NameValueCCSID (-nc) (MQLONG) (1208)

「NameValueData フィールド内のデータのコード化文字セット ID」を指定します。本オプション(-nc)に続けてMQCCSI_*を指定することも可能です。(「表 5.7.2 指定可能なMQCCSI_*」参照) 本オプション(-nc)を指定せずにMQCCSI_*を指定した場合は、MQMD.CodedCharSetIdに設定されます。

NameValueData (-nd) (MQCHARn) (-)

「名前/値ペアまたはメッセージ・プロパティの入ったフォルダーを含む可変長フィールド」を指定します。-nd "data1, data2, data3" の様にカンマで区切って指定します。

5.8 MQCSPのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

CSPUserId (-cu) (-) (-)

「認証で使用するユーザーID」を指定します。MQCSP構造を使用して資格情報を渡す場合のユーザーID の最大長は 1024 文字です。

“-cu”および“-cp”でMQCSPに設定するユーザーおよびパスワードを指定すると、mqpgfは、MQCSP.CSPUserIdPtr、MQCSP.CSPPasswordPtrにそれぞれの文字列のポインターを、MQCSP.CSPUserIdLength、MQCSP.CSPPasswordLengthにそれぞれの文字列の長さを自動的に設定します。 認証を実施させるには、MQCSP.AuthenticationTypeにMQCSP_AUTH_USER_ID_AND_PWDを指定することが必要です。 mqpgfは、MQCSP_AUTH_*が引数に指定されると、それをMQCSP.AuthenticationTypeに設定し、さらにMQCSPを参照させる為に、MQCNO.SecurityParmsPtrにMQCSPへのポインターを自動的に設定します。 また、このMQCNO.SecurityParmsPtrの指定はMQCNO_VERSION_5以上が必要な点に注意してください。 MQCNOのデフォルトはMQCNO_VERSION_1です。 MQCNOのバージョンがMQCNO_VERSION_5未満の場合は、認証が行われません。

CSPPassword (-cp) (-) (-)

「認証で使用するパスワード」を指定します。（上記「MQCSPのフィールド — CSPUserId (-cu)」参照）

Ex. 5.8.1 MQCSPにユーザー/パスワードを指定して接続

```
-----  
$ mqpgfc -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnnn) -ch SampleQ.  
MQICHL -cu pulsar -cp correctPW MQCSP_AUTH_USER_ID_AND_PWD MQCNO_VERSION_5  
[2018/08/23 14:50:46.286] 1: message length: 4 put message: test
```

```
$ mqpgfc -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnnn) -ch SampleQ  
M.MQICHL -cu pulsar -cp wrongPW MQCSP_AUTH_USER_ID_AND_PWD MQCNO_VERSION_5  
MQCONN fail : SampleQM CompCd=02 ReasonCd=2035  
!!! Queue Manager Connect Fail SampleQM !!!
```

```
$ mqrc 2035
```

```
2035 0x000007f3 MQRC_NOT_AUTHORIZED
```

```
$ mqpgfc -qm SampleQM -q SampleQ -m test -x nnn.nnn.nnn.nnn(nnnnn) -ch SampleQ
```

```
M.MQICHL -cu pulsar -cp wrongPW MQCSP_AUTH_USER_ID_AND_PWD  
[2018/08/23 14:51:02.293] 1: message length: 4 put message: test
```

※この例は、間違ったパスワードを指定していますが、MQCNO_VERSION_5が指定されていないので、デフォルトのMQCNO_VERSION_1が使用され、MQCSPに指定されたユーザーとパスワードが認証されなかった為、その結果としてPUTが成功しています。

5.9 MQODのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

ObjectQMgrName (-om) (MQCHAR48) ("")

「オブジェクトが定義されるキュー・マネージャーの名前」を指定します。

Ex. 5.9.1 オブジェクト・キューマネージャーの指定例

```
-----
$ mqpcf cque -qm QMB -q CQ1 TYPE CLUSQMGR
1: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(QMA)
2: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(QMB)
3: QUEUE(CQ1) TYPE(QCLUSTER) CLUSQMGR(QMC)

$ mqpcf que -qm QMB -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(0)

$ mqpcf que -qm QMC -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(0)

$ mqpgf -qm QMB -q CQ1 -m "test" -n 3
[16/12/23 00:45:19] 1: message length: 4 put message : test
[16/12/23 00:45:19] 2: message length: 4 put message : test
[16/12/23 00:45:19] 3: message length: 4 put message : test
$ mqpcf que -qm QMB -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(3)
$ mqpcf que -qm QMC -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(0)
```

※デフォルトでは接続したキューマネージャーのクラスタキューにPUTされる為、オブジェクトキューマネージャーでPUT先のキューマネージャーを指定します。

```
$ mqpgf -qm QMB -q CQ1 -m "test" -om QMC -n 3
[16/12/23 00:46:40] 1: message length: 4 put message : test
[16/12/23 00:46:40] 2: message length: 4 put message : test
[16/12/23 00:46:40] 3: message length: 4 put message : test
$ mqpcf que -qm QMB -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(3)
$ mqpcf que -qm QMC -q CQ1 CURDEPTH
1: QUEUE(CQ1) TYPE(QLOCAL) CURDEPTH(3)
```

AlternateUserId (-au) (MQCHAR12) ("")

「代替ユーザー ID」を指定します。同時に MQ00_ALTERNATE_USER_AUTHORITY を指定することが必要です。また、プログラムの実行ユーザーに altusr 権限も必要です。

Ex. 5.9.2 代替ユーザーIDを指定してPUT/GET

※put/get等の権限の無いユーザーIDに対して altusr 権限をQmgrに対して設定します。

```
$ setmqaut -m TESTQM -t qmgr -p guest +altusr
```

setmqaut コマンドが正常に完了しました。

```
$ dspmqaut -m TESTQM -t qmgr -p guest
```

エンティティー guest はオブジェクト TESTQM について次の許可を持っています:

```
inq
set
connect
altusr
setid
setall
```

```
$ dspmqaut -m TESTQM -t q -n TQ -p guest
```

エンティティー guest はオブジェクト TQ について次の許可を持っています:

```
$ id
```

```
uid=243(guest) gid=1(staff)
```

※-au でPUT/GETの権限のあるユーザーIDを指定しただけでは 2035 エラーが発生する。

```
$ mqpgf -qm TESTQM -q TQ -m "test message" -au mqm
```

```
MQOPEN fail : TESTQM TQ CompCd=02 ReasonCd=2035
```

※さらに MQ00_ALTERNATE_USER_AUTHORITY を指定して MQOPEN するとPUT/GETが可能となる。

```
$ mqpgf -qm TESTQM -q TQ -m "test " -au mqm MQ00_ALTERNATE_USER_AUTHORITY
```

```
[16/12/23 01:01:17] 1: message length: 4 put message : test
```

```
$ mqpgf -qm TESTQM -q TQ -au mqm MQ00_ALTERNATE_USER_AUTHORITY
```

```
[16/12/23 01:01:33] 1: message length: 4 get message : test
```

ObjectRec (MQOR) (-or) (-) (-)

「オブジェクト・レコード」を指定します。
-or <queue1[:qmgr1]>, <queue2[:qmgr2]>, <queue3[:qmgr3]>, ... の様に指定します。
(「3. 使用方法の説明 — 配布リストの使用」参照)

DynamicQName (-dq) (MQCHAR48) ("AMQ.*")

「動的キューの名前」を指定します。メッセージを送信するキューを動的キューにする場合は"-q"に、応答を受信するキューを動的キューにする場合は"-iq"にモデル・キュー名を指定します。

Ex. 5.9.3 動的一時キューを作成し、メッセージを書き込む

※モデル・キュー (SYSTEM.DEFAULT.MODEL.QUEUE) を使用して、"DYNAMICQS1"という名前の動的一時キューを作成して、メッセージを書き込み、MQCLOSE前でプログラムを停止させます。

```
$ mqpgf -qm SampleQM -q SYSTEM.DEFAULT.MODEL.QUEUE -m dyntest -dq "DYNAMICQS1"
-s MQCLOSE
[18/05/31 15:12:00] 1: message length: 7 put message: dyntest
MQCMIT success : CompCd=00 ReasonCd=00
stop before calling MQCLOSE().
Hit Any Key!!!
```

※他のターミナルで動的一時キューが作成され、メッセージがキューにあることを確認し、メッセージをGETします。

```
$ mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE CURDEPTH
1: QUEUE(DYNAMICQS1) TYPE(QLOCAL) CURDEPTH(1) DEFTYPE(TEMPDYN)
```

```
$ mqpgf -qm SampleQM -q DYNAMICQS1
[18/05/31 15:20:13] 1: message length: 7 get message : dyntest
```

※元のターミナルで任意のキーを入力して、MQCLOSE()を呼び出しすと、動的一時キューは削除されます。

```
$ mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE CURDEPTH
1: QUEUE(DYNAMICQS1) TYPE(QLOCAL) CURDEPTH(1) DEFTYPE(TEMPDYN)
/home/okaqm7/work/cprog/mqpgf: mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE C>
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommand
```

```
CC=[2], mqCommandRC=[2085]
```

```
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommand  
CC=[2], mqCommandRC=[3008]
```

```
$ mqrc 2085
```

```
2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME
```

Ex. 5.9.4 動的一時キューを作成し、レポートメッセージを待つ

※キュー (SampleQ) からメッセージの受信を繰り返し待機し、受信したメッセージはMQMD.MsgTypeにMQMT_REPLYを二次側に設定して、MQMD.ReplyToQに設定されているキューに書き込みます。mqpgfは"-oq"に"*"を指定すると、MQMD.ReplyToQにメッセージを書き込みます。

```
$ mqpgf -qm SampleQM -q SampleQ -oq "*" MQGMO_WAIT MQWI_UNLIMITED -dp -r -ss MQMT_REPLY
```

※別ターミナルから、応答受信の為の動的一時キュー ("DYNAMIC*") を作成し、その作成されたキュー名を送信メッセージのMQMD.ReplyToQに設定、MQMD.MsgTypeが"MQMT_REQUEST"であるメッセージを3件をキューにPUTします。この時、MQCLOSE()の呼び出し前で一旦処理を停止します。

```
$ mqpgf -qm SampleQM -q SampleQ -m "dynamic" -iq SYSTEM.DEFAULT.MODEL.QUEUE -rm  
SampleQM MQMT_REQUEST MQPMO_NO_SYNCPOINT -n 3 -i 1000 -ss MQGMO_WAIT MQWI_UNLIMITED MQGMO_NO_SYNCPOINT -dq DYNAMIC* -s MQCLOSE
```

```
[18/05/31 15:51:55] 1: message length: 7 put message : dynamic
```

```
[18/05/31 15:51:55] 1: message length: 7 get message : dynamic
```

```
stop before calling MQCLOSE().
```

```
Hit Any Key!!!
```

```
[18/05/31 15:51:59] 1: message length: 7 put message : dynamic
```

```
[18/05/31 15:51:59] 1: message length: 7 get message : dynamic
```

```
stop before calling MQCLOSE().
```

```
Hit Any Key!!!
```

```
[18/05/31 15:52:01] 1: message length: 7 put message : dynamic
```

```
[18/05/31 15:52:01] 1: message length: 7 get message : dynamic
```

```
stop before calling MQCLOSE().
```

```
Hit Any Key!!!
```

※任意のキーを押す度に、もう一つのターミナルで受信したメッセージのダンプと、送信した応答メッセージが表示されます。

```
message number: 1
```


*StrucId[MD] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[943] Format[] Priority[0] Persistence[0] MsgId[0x414D512053616D706C65514D202020205B0F79E22000360A] CorrelId[0x00000000000000000000000000000000] BackoutCount[0] **ReplyToQ[DYNAMIC5B0F79E220003609**
] ReplyToQMgr[SampleQM
] UserIdentifier[mqm] AccountingToken[0x05343430333100] ApplIdentityData[
] PutApplType[13] PutApplName[mqpgf] PutDate[20180531] PutTime[06515552] ApplOriginData[]

GroupId[0x00] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 7
00000000: 6479 6E61 6D69 63 'dynamic '

[18/05/31 15:51:55] 1: message length: 7 put message : dynamic
MQCMIT success : CompCd=00 ReasonCd=00
message number: 2

*StrucId[MD] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[943] Format[] Priority[0] Persistence[0] MsgId[0x414D512053616D706C65514D202020205B0F79E22000360C] CorrelId[0x00000000000000000000000000000000] BackoutCount[0] **ReplyToQ[DYNAMIC5B0F79E22000360B**
] ReplyToQMgr[SampleQM
] UserIdentifier[mqm] AccountingToken[0x05343430333100] ApplIdentityData[
] PutApplType[13] PutApplName[mqpgf] PutDate[20180531] PutTime[06515956] ApplOriginData[]

GroupId[0x00] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]

data length: 7
00000000: 6479 6E61 6D69 63 'dynamic '

[18/05/31 15:51:59] 2: message length: 7 put message : dynamic
MQCMIT success : CompCd=00 ReasonCd=00
message number: 3

*StrucId[MD] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[943] Format[] Priority[0] Persistence[0] MsgId[0x414D512053616D706C65514D202020205B0F79E22000360E] CorrelId[0x00000000000000000000000000000000] BackoutCount[0] **ReplyToQ[DYNAMIC5B0F79E22000360D**

```

] ReplyToQMgr[SampleQM
] UserIdentifier[mqm] AccountingToken[0x05343430333100000000000000000000
000000000000000000000000000000000006] ApplIdentityData[
] PutApplType[13] PutApplName[mqpgf] PutDate[201805
31] PutTime[06520139] ApplOriginData[ ]

```

```

GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]

```

```

data length: 7
00000000: 6479 6E61 6D69 63 'dynamic'

```

```

[18/05/31 15:52:01] 3: message length: 7 put message : dynamic

```

※応答を受信して、動的一時キューをクローズする度にそのキューは削除されます。作成された3つの動的一時キューは全て削除されています。

```

$ mqpcf que -qm SampleQM -q DYNA* TYPE DEFTYPE CURDEPTH
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommand
CC=[2], mqCommandRC=[2085]
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommand
CC=[2], mqCommandRC=[3008]

```

```

$ mqrc 2085

```

```

2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME
-----

```

5.10 MQPMOのフィールド

以下、フィールド名 (オプション) (データタイプ) (デフォルト値) です。

PutMsgRec (MQPMR) (-mr) (-) (-)

「書き込みメッセージ・レコード」を指定します。

-mr <msgId>:<correlId>:<groupId>:<feedback>:<accountingtoken>,... の様に指定します。 (「3. 使用方法の説明 - 配布リストの使用」参照)

5.11 MQGMOのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

WaitInterval (-wi) (MQLONG) (0)

「メッセージが到着するまで待機するおよその時間をミリ秒」を指定します。同時にMQGMO_WAITを指定する必要があります。本オプション(-wi)を指定せずに直接MQWI_*を指定した場合も、MQGMO.WaitInterval に設定されます。

Ex. 5.11.1 GET時にメッセージが到着するまで指定時間待機

```
-----
※メッセージが到着するまで5秒待機するように指定する。
$ date; mqpgf -qm TESTQM -q TQ -wi 5000 MQGMO_WAIT
Fri Jan  6 22:11:42 JST 2017
[17/01/06 22:11:44] 1: message length: 4 get message : test
-----
```

MsgToken (-mt) (MQBYTE16) (MQMTOK_NONE_ARRAY)

「メッセージ・トークン (MsgToken)」を指定します。指定できる表記方法は16進表記のみです。MQGMO_VERSION_3以上を指定する必要があります。通常はMQMO_MATCH_MSG_TOKENと共に使用します。

5.12 MQIMPOのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

RequestedEncoding（-pe）（MQLONG）（MQENC_NATIVE）

MQIMPO_CONVERT_VALUE または MQIMPO_CONVERT_TYPE の指定時に、照会されるプロパティ値の変換先のエンコードを指定します。本オプション(-pe)に続けてMQENC_*を指定することも可能です。（「表 5.7.1 指定可能なMQENC_*」参照）複数指定する場合は”：”で区切ります。複数指定された場合はそのORをフィールドに設定します。本オプション(-pe)を指定せずにMQENC_*を指定した場合は、MQMD.Encodingに設定されます。本フィールドは3桁の16進表記（12bit）でも指定可能です。16進表記で指定したい場合は先頭に0xを付加し、必ず3桁指定します。（「6. コンスタントの指定 - MQINQMPオプションMQIMPO_*」参照）

RequestedCCSID（-pc）（MQLONG）（MQCCSI_APPL）

MQIMPO_CONVERT_VALUE または MQIMPO_CONVERT_TYPE の指定時に、照会されるプロパティ値の変換先のCCSIDを指定します。本オプション(-pc)に続けてMQCCSI_*を指定することも可能です。（「表 5.7.2 指定可能なMQCCSI_*」参照）本オプション(-pc)を指定せずにMQCCSI_*を指定した場合は、MQMD.CodedCharSetIdに設定されます。（「6. コンスタントの指定 - MQINQMPオプションMQIMPO_*」参照）

5.13 MQCBのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

Operation（-op）（MQLONG）（-）

MQCBのOperationを指定します。指定可能な値はMQOP_*です。複数のオプションを同時に指定したい場合は、“-op MQOP_REGISTER -op MQOP_SUSPEND”の様に複数回指定します。MQCBは“-cf”オプションを使用してMQCBDフィールドにCallBack関数を指定した場合にのみ呼び出されます。（「MQCBDのフィールド - CallbackFunction（-cf）（MQPTR）（-）」参照）

下表のコンスタントが指定可能です。

表 5.13.1 指定可能な MQOP_*			
コンスタント	値	設定場所	設定方法他
MQOP_START	0x00000001	MQCB() の引数 Operation	OR
MQOP_START_WAIT	0x00000002	以下同様	以下同様
MQOP_STOP	0x00000004		
MQOP_REGISTER	0x00000100		
MQOP_DEREGISTER	0x00000200		
MQOP_SUSPEND	0x00010000		
MQOP_RESUME	0x00020000		

5.14 MQCBDのフィールド

以下、フィールド名 (オプション) (データタイプ) (デフォルト値) です。

CallbackFunction (-cf) (MQPTR) (-)

コールバック関数を指定します。現在指定可能なコールバック関数は“EventHandler”のみです。EventHandlerはクライアント自動接続をテストするときに有用で、イベント・ハンドラーがコール・バックされた時に、Context.Reasonに何が設定されていたかと、Context.ReconnectDelayの値を表示します。

5.15 MQSCOのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

KeyRepository (-kr) (MQCHAR256) (-)

クライアント・モードでSSL/TLSを使用する場合に、キー・リポジトリの場所を指定します。GSKitの場合は<ディレクトリ>/<キーDBの拡張子を除く部分>を指定します。OpenSSL (MQ for HPE NonStop等)の場合は証明書ファイルを配置しているディレクトリを指定します。

5.16 MQAIRのフィールド

以下、フィールド名（オプション）（データタイプ）（デフォルト値）です。

OCSPResponderURL（-ru）（MQCHAR256）（-）

クライアント・モードでSSL/TLSを使用し、OCSPリスポンダーに接続して証明書を検証する場合に、OCSPリスポンダーへ接続する為のURLを指定します。接続文字列は”http://”で始まる必要があります。

6. コンスタントの指定

コンスタントは必要なものをいくつでも同時に指定することができます。

6.1 MQMDパラメータ

MQMD_*

MQMD.Versionに設定されます。（「5. 全パラメータ・リファレンス - MQMD Version 2のフィールド」参照）

MQMQ.VesionはPUT／GET時間問わずに常に入力フィールドです。MQMD.Versionを読み出すことはできません。

下表のコンスタントが指定可能です。

表 6.1.1 指定可能な MQMD_*			
コンスタント	値	設定場所	設定方法他
MQMD_VERSION_1	1	MQMD.Version	Overwrite (default)
MQMD_VERSION_2	2	以下同様	以下同様
MQMD_CURRENT_VERSION	2		

Ex. 6.1.1 MQMDのバージョンをGET時に指定

※MQMD_VERSION_2を指定し、Version 2のフィールドをデフォルト以外にしてPUT
\$ mqpgef -qm TESTQM -q TQ -m test -gi GID -ms 3 -of 100 -ol 1000 MQMD_VERSION_2 MQMF_SEGMENT MQMT_REPORT MQMF_MSG_IN_GROUP
[16/12/28 19:57:08] 1: message length: 4 put message : test

※MQMD_VERSION_2 を指定してGET（-br/-brvを指定時のデフォルトは MQMD_VERSION_2 の為、省略可能）
\$ mqpgef -qm TESTQM -q TQ -brv MQMD_VERSION_2
message number: 1
*StrucId[MD] Version[2] Report[0] MsgType[4] Expiry[-1] Feedback[0]
....

```
GroupId[0x4749440000000000000000000000000000000000000000000000000] MsgSeqNumber[3] Offset[100] MsgFlags[10] OriginalLength[1000]
```

```
....
```

※MQMD_VERSION_1 を指定してGET

```
$ mqpgef -qm TESTQM -q TQ -brv MQMD_VERSION_1
```

```
message number: 1
```

```
*StrucId[MD ] Version[1] Report[0] MsgType[4] Expiry[-1] Feedback[0] ....
```

```
....
```

```
GroupId[0x0000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]
```

```
*StrucId[MDE ] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[943] Format[          ] Flags[0] GroupId[0x4749440000000000000000000000000000000000000000000000000] MsgSeqNumber[3] Offset[100] MsgFlags[10] OriginalLength[1000]
```

```
....
```

※MQMD Version 2のフィールドはデフォルトの表示となり、変わりにMQMDEヘッダが生成されて対応するフィールドが設定される。

```
-----
```

MQRO_*

MQMD. Reportに設定されます。複数指定した場合はそれらの OR で設定します。同時に ReplyToQを指定(-rq) を指定する必要があります。

下表のコンスタントが指定可能です。

表 6.1.2 指定可能な MQRO_*			
コンスタント	値	設定場所	設定方法他
MQRO_EXCEPTION	0x01000000	MQMD. Report	OR
MQRO_EXCEPTION_WITH_DATA	0x03000000	以下同様	以下同様
MQRO_EXCEPTION_WITH_FULL_DATA	0x07000000		
MQRO_EXPIRATION	0x00200000		
MQRO_EXPIRATION_WITH_DATA	0x00600000		
MQRO_EXPIRATION_WITH_FULL_DATA	0x00E00000		
MQRO_COA	0x00000100		

表 6.1.2 指定可能な MQRO_*			
コンスタント	値	設定場所	設定方法他
MQRO_COA_WITH_DATA	0x00000300		
MQRO_COA_WITH_FULL_DATA	0x00000700		
MQRO_COD	0x00000800		
MQRO_COD_WITH_DATA	0x00001800		
MQRO_COD_WITH_FULL_DATA	0x00003800		
MQRO_PAN	0x00000001		
MQRO_NAN	0x00000002		
MQRO_ACTIVITY	0x00000004		
MQRO_NEW_MSG_ID	0x00000000		
MQRO_PASS_MSG_ID	0x00000080		
MQRO_COPY_MSG_ID_TO_CORREL_ID	0x00000000		
MQRO_PASS_CORREL_ID	0x00000040		
MQRO_DEAD_LETTER_Q	0x00000000		
MQRO_DISCARD_MSG	0x08000000		
MQRO_PASS_DISCARD_AND_EXPIRY	0x00004000		
MQRO_NONE	0x00000000		(default)

Ex. 6.1.2 レポートオプションの指定例

※レポートオプションを複数指定してPUTする。

```
$ mqpgf -qm TESTQ -q TQ -f input.txt MQRO_EXCEPTION MQRO_COA_WITH_FULL_DATA
[16/12/27 20:27:47] 1: put from input.txt
MQPUT fail : TQ CompCd=02 ReasonCd=2027
$
$ mqrc 2027
```

2027 0x000007eb MQRC_MISSING_REPLY_TO_Q

※ReplyToQを指定(-rq)しないと、RC=2027(MQRC_MISSING_REPLY_TO_Q)でエラーとなる。

```

$ mqpgf -qm TESTQ -q TQ -m "test" -rq RQ MQRO_EXCEPTION MQRO_COA_WITH_FULL_DATA
[16/12/27 20:33:35] 1: message length: 4 put message : test
$ mqpgf -qm TESTQ -q TQ -brv -hex
message number: 1
*StrucId[MD  ] Version[2] Report[0x01000700] MsgType[8] Expiry[-1] Feedback[0] E
ncoding[273] CodedCharSetId[943] Format[          ] Priority[0] Persistence[0] Msg
Id[0x414D51206F6B61716D3830612020202058624CD720002287] CorrelId[0x000000000000000
00000000000000000000000000000000] BackoutCount[0] ReplyToQ[RQ
] ....
....
※下記の2つの OR が Report フィールドに設定される。
MQRO_EXCEPTION : 0x01000000
MQRO_COA_WITH_FULL_DATA : 0x00000700
-----

```

MQMT_*

MQMD.MsgTypeに設定されます。

下表のコンスタントが指定可能です。

表 6.1.3 指定可能な MQMT_*			
コンスタント	値	設定場所	設定方法他
MQMT_SYSTEM_FIRST	1	MQMD.MsgType	Overwrite
MQMT_REQUEST	1	以下同様	以下同様
MQMT_REPLY	2		
MQMT_DATAGRAM	8		(default)
MQMT_REPORT	4		
MQMT_MQE_FIELDS_FROM_MQE	112		
MQMT_MQE_FIELDS	113		
MQMT_SYSTEM_LAST	65535		
MQMT_APPL_FIRST	65536		
MQMT_APPL_LAST	999999999		

Ex. 6.1.3 メッセージタイプの指定例

```
-----
$ mqpgf -qm TESTQM -q TQ -m test MQMT_REQUEST
[16/12/27 21:17:22] 1: message length: 4 put message : test
MQPUT fail : TQ CompCd=02 ReasonCd=2027
$
$ mqrc 2027
```

2027 0x000007eb MQRC_MISSING_REPLY_TO_Q

※MQMT_REQUESTを指定する場合は、ReplyToQを指定(-rq)しないと、RC=2027(MQRC_MISSING_REPLY_TO_Q)でエラーとなる。

```
$ mqpgf -qm TESTQM -q TQ -m test MQMT_REQUEST -rq RQ
[16/12/27 21:17:38] 1: message length: 4 put message : test
$
mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[1] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[943] Format[ ] Priority[0] Persistence[0] MsgId[0x414D
51206F6B61716D3830612020202058624CD720002503] CorrelId[0x00000000000000000000
000000000000000000000000] BackoutCount[0] ReplyToQ[RQ
] ....
-----
```

MQEI_*

MQMD.Expiryに設定されます。

下表のコンスタントが指定可能です。

表 6.1.4 指定可能な MQEI_*			
コンスタント	値	設定場所	設定方法他
MQEI_UNLIMITED	(-1)	MQMD.Expiry	Overwrite (default)

MQFB_*

MQMD. Feedbackに設定されます。

下表のコンスタントが指定可能です。

表 6.1.5 指定可能な MQFB_*			
コンスタント	値	設定場所	設定方法他
MQFB_NONE	0	MQMD. Feedback	Overwrite (default)
MQFB_SYSTEM_FIRST	1	以下同様	以下同様
MQFB_QUIT	256		
MQFB_EXPIRATION	258		
MQFB_COA	259		
MQFB_COD	260		
MQFB_CHANNEL_COMPLETED	262		
MQFB_CHANNEL_FAIL_RETRY	263		
MQFB_CHANNEL_FAIL	264		
MQFB_APPL_CANNOT_BE_STARTED	265		
MQFB_TM_ERROR	266		
MQFB_APPL_TYPE_ERROR	267		
MQFB_STOPPED_BY_MSG_EXIT	268		
MQFB_ACTIVITY	269		
MQFB_XMIT_Q_MSG_ERROR	271		
MQFB_PAN	275		
MQFB_NAN	276		
MQFB_STOPPED_BY_CHAD_EXIT	277		
MQFB_STOPPED_BY_PUBSUB_EXIT	279		
MQFB_NOT_A_REPOSITORY_MSG	280		
MQFB_BIND_OPEN_CLUSRCVR_DEL	281		

表 6.1.5 指定可能な MQFB_*			
コンスタント	値	設定場所	設定方法他
MQFB_MAX_ACTIVITIES	282		
MQFB_NOT_FORWARDED	283		
MQFB_NOT_DELIVERED	284		
MQFB_UNSUPPORTED_FORWARDING	285		
MQFB_UNSUPPORTED_DELIVERY	286		
MQFB_DATA_LENGTH_ZERO	291		
MQFB_DATA_LENGTH_NEGATIVE	292		
MQFB_DATA_LENGTH_TOO_BIG	293		
MQFB_BUFFER_OVERFLOW	294		
MQFB_LENGTH_OFF_BY_ONE	295		
MQFB_I IH_ERROR	296		
MQFB_NOT_AUTHORIZED_FOR_IMS	298		
MQFB_IMS_ERROR	300		
MQFB_IMS_FIRST	301		
MQFB_IMS_LAST	399		
MQFB_CICS_INTERNAL_ERROR	401		
MQFB_CICS_NOT_AUTHORIZED	402		
MQFB_CICS_BRIDGE_FAILURE	403		
MQFB_CICS_CORREL_ID_ERROR	404		
MQFB_CICS_CCSID_ERROR	405		
MQFB_CICS_ENCODING_ERROR	406		
MQFB_CICS_CIH_ERROR	407		
MQFB_CICS_UOW_ERROR	408		
MQFB_CICS_COMMAREA_ERROR	409		
MQFB_CICS_APPL_NOT_STARTED	410		

表 6.1.5 指定可能な MQFB_*			
コンスタント	値	設定場所	設定方法他
MQFB_CICS_APPL_ABENDED	411		
MQFB_CICS_DLQ_ERROR	412		
MQFB_CICS_UOW_BACKED_OUT	413		
MQFB_PUBLICATIONS_ON_REQUEST	501		
MQFB_SUBSCRIBER_IS_PUBLISHER	502		
MQFB_MSG_SCOPE_MISMATCH	503		
MQFB_SELECTOR_MISMATCH	504		
MQFB_NOT_A_GROUPUR_MSG	505		
MQFB_IMS_NACK_1A_REASON_FIRST	600		
MQFB_IMS_NACK_1A_REASON_LAST	855		
MQFB_SYSTEM_LAST	65535		
MQFB_APPL_FIRST	65536		
MQFB_APPL_LAST	999999999		

Ex. 6.1.4 フィードバックの指定例

```

-----
$ mqpgf -qm TESTQM -q TQ -m test MQFB_COA
[16/12/27 21:23:35] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[259] ....
....
-----

```

MQENC_*

MQMD.Encodingに設定されます。（指定可能なコンスタントは「表 5.7.1 指定可能な MQENC_*」 参照）

（注）MQRFH2のEncodingフィールドに指定したい場合は、区別する為に `-re` パラメータに続いて指定します。

MQCCSI_*

MQMD. CodedCharSetIdに設定されます。（指定可能なコンスタントは「表 5.7.2 指定可能なMQCCSI_*」 参照）

（注）MQRFH2のCodedCharSetIdまたはNameValueCCSIDフィールドに指定したい場合は、区別する為にそれぞれ `-rc`、`-nc` パラメータに続いて指定します。

MQFMT_*

MQMD. Formatに設定されます。

（注）MQRFH2のFormatフィールドに指定したい場合は、区別する為に `-rf` パラメータに続いて指定します。

下表のコンスタントが指定可能です。

表 6.1.6 指定可能な MQFMT_*			
コンスタント	値	設定場所	設定方法他
MQFMT_NONE	" "	MQMD. Format	Overwrite (default)
MQFMT_ADMIN	"MQADMIN "	以下同様	以下同様
MQFMT_CHANNEL_COMPLETED	"MQCHCOM "		
MQFMT_CICS	"MQCICS "		
MQFMT_COMMAND_1	"MQCMD1 "		
MQFMT_COMMAND_2	"MQCMD2 "		
MQFMT_DEAD_LETTER_HEADER	"MQDEAD "		
MQFMT_DIST_HEADER	"MQHDIST "		
MQFMT_EMBEDDED_PCF	"MQHEPCF "		
MQFMT_EVENT	"MQEVENT "		

表 6.1.6 指定可能な MQFMT_*			
コンスタント	値	設定場所	設定方法他
MQFMT_IMS	"MQIMS "		
MQFMT_IMS_VAR_STRING	"MQIMSVS "		
MQFMT_MD_EXTENSION	"MQHMDE "		
MQFMT_PCF	"MQPCF "		
MQFMT_REF_MSG_HEADER	"MQHREF "		
MQFMT_RF_HEADER	"MQHRF "		
MQFMT_RF_HEADER_1	"MQHRF "		
MQFMT_RF_HEADER_2	"MQHRF2 "		
MQFMT_STRING	"MQSTR "		
MQFMT_TRIGGER	"MQTRIG "		
MQFMT_WORK_INFO_HEADER	"MQHWIH "		
MQFMT_XMIT_Q_HEADER	"MQXMIT "		

Ex. 6.1.5 MQMDのFormatとMQRFH2のFormatの指定例

```

-----
$ mqpgf -qm TESTQM -q TQ -m test MQFMT_RF_HEADER_2 -rf MQFMT_STRING
[16/12/27 21:44:59] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[943] Format[MQHRF2 ] ....
....
*StrucId[RFH ] Version[2] StrucLength[36] Encoding[273] CodedCharSetId[943] Form
at[MQSTR ] Flags[0] NameValueCCSID[1208]
....
-----

```

MQPRI_*

MQMD.Priorityに設定されます。

下表のコンスタントが指定可能です。

表 6.1.7 指定可能な MQPRI_*			
コンスタント	値	設定場所	設定方法他
MQPRI_PRIORITY_AS_Q_DEF	(-1)	MQMD.Priority	Overwrite (default)
MQPRI_PRIORITY_AS_PARENT	(-2)	以下同様	以下同様
MQPRI_PRIORITY_AS_PUBLISHED	(-3)		
MQPRI_PRIORITY_AS_TOPIC_DEF	(-1)		

MQPER_*

MQMD.Persistenceに設定されます。

下表のコンスタントが指定可能です。

表 6.1.8 指定可能な MQPER_*			
コンスタント	値	設定場所	設定方法他
MQPER_PERSISTENCE_AS_PARENT	(-1)	MQMD.Persistence	Overwrite
MQPER_NOT_PERSISTENT	0	以下同様	以下同様
MQPER_PERSISTENT	1		
MQPER_PERSISTENCE_AS_Q_DEF	2		(default)
MQPER_PERSISTENCE_AS_TOPIC_DEF	2		

Ex. 6.1.6 パーシステント属性を指定例

```
-----  
$ mqpgf -qm TESTQM -q TQ -inq MQIA_DEF_PERSISTENCE  
[16/12/28 19:07:54] 1: DEFPSIST(NO)  
$  
$ mqpgf -qm TESTQM -q TQ -m test MQPER_PERSISTENT
```

```
[16/12/28 19:08:58] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[943] Format[          ] Priority[0] Persistence[1] ....
.....
-----
```

MQMI_*

MQMD.MsgIdに設定されます。下表に示すとおり指定できるコンスタントは1つでデフォルトになっているので、指定しても動作に影響はありません。

下表のコンスタントが指定可能です。

表 6.1.9 指定可能な MQMI_*			
コンスタント	値	設定場所	設定方法他
MQMI_NONE	0x00 (24Byte)	MQMD.MsgId	Overwrite (default)

MQCI_*

MQMD.CorrelIdに設定されます。

下表のコンスタントが指定可能です。

表 6.1.10 指定可能な MQCI_*			
コンスタント	値	設定場所	設定方法他
MQCI_NONE	0x00 (24Byte)	MQMD.CorrelId	Overwrite (default)
MQCI_NEW_SESSION	"AMQ!NEW_SESSION_CORRELID"		

MQACT_*

MQMD.AccountingTokenに設定されます。MQMD.AccountingTokenの最終バイトが指定されていた場合（MQACTT_UNKNOWN 0x00 以外が指定されていた場合）、最終バイトは上書きしません。下表に示すとおり指定できるコンスタントは1つでデフォルトになっているので、指定しても動作に影響はありません。

下表のコンスタントが指定可能です。

表 6.1.11 指定可能な MQACT_*			
コンスタント	値	設定場所	設定方法他
MQACT_NONE	0x00 (32Byte)	MQMD.AccountingToken	Overwrite (default)

MQACTT_*

MQMD.AccountingTokenの最終バイト(32バイト目)に設定されます。同時にMQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXTの指定が必要です。また、MQPMO_SET_IDENTITY_CONTEXT を指定する場合はオープン・オプションの MQ00_SET_IDENTITY_CONTEXT、MQPMO_SET_ALL_CONTEXT を指定する場合はオープン・オプションのMQ00_SET_ALL_CONTEXT の指定が必要となります。

下表のコンスタントが指定可能です。

表 6.1.12 指定可能な MQACTT_*			
コンスタント	値	設定場所	設定方法他
MQACTT_UNKNOWN	0x00	MQMD.AccountingToken(32 バイト目)	Overwrite (default)
MQACTT_CICS_LUOW_ID	0x01		
MQACTT_OS2_DEFAULT	0x04		
MQACTT_DOS_DEFAULT	0x05		
MQACTT_UNIX_NUMERIC_ID	0x06		

表 6.1.12 指定可能な MQUACTT_*			
コンスタント	値	設定場所	設定方法他
MQUACTT_OS400_ACCOUNT_TOKEN	0x08		
MQUACTT_WINDOWS_DEFAULT	0x09		
MQUACTT_NT_SECURITY_ID	0x0B		
MQUACTT_USER	0x19		

Ex. 6.1.7 MQUACTT_*と-at の指定例

```

-----
※MQUACTT_*と-at を同時に指定してPUTする。
※AccountingToken は 32バイトですが、MQUACTT_*が最終バイトに設定されます。上書き
されない様に -at で指定するバイト数は31バイト以下にします。
$ mqpgf -qm TESTQM -q TQ -m test MQUACTT_CICS_LUOW_ID -at 0x0332343200000000000000
000000000000000000000000000000000000 MQUPMO_SET_ALL_CONTEXT MQUOO_SET_ALL_C
ONTEXT

[17/01/10 14:21:38] 1: message length: 4 put message : test
$ mqpgf -qm TESTQM -q TQ -brv -r
message number: 1
*StrucId[MD ] .... AccountingToken[0x033234320000000000000000000000000000
000000000000000000000001] ....
....
-----

```

MQUAT_*

MQUMD.PutApplTypeに設定されます。同時にMQUPMO_SET_ALL_CONTEXT MQUOO_SET_ALL_CON
TEXTを指定することが必要です。

下表のコンスタントが指定可能です。

表 6.1.13 指定可能な MQUAT_*			
コンスタント	値	設定場所	設定方法他
MQUAT_UNKNOWN	(-1)	MQUMD.PutApplType	Overwrite 以下同様

表 6.1.13 指定可能な MQAT_*

コンスタント	値	設定場所	設定方法他
MQAT_NO_CONTEXT	0	以下同様	(default)
MQAT_CICS	1		
MQAT_MVS	2		
MQAT_OS390	2		
MQAT_ZOS	2		
MQAT_IMS	3		
MQAT_OS2	4		
MQAT_DOS	5		
MQAT_AIX	6		
MQAT_UNIX	6		
MQAT_QMGR	7		
MQAT_OS400	8		
MQAT_WINDOWS	9		
MQAT_CICS_VSE	10		
MQAT_WINDOWS_NT	11		
MQAT_VMS	12		
MQAT_GUARDIAN	13		
MQAT_NSK	13		
MQAT_VOS	14		
MQAT_OPEN_TP1	15		
MQAT_VM	18		
MQAT_IMS_BRIDGE	19		
MQAT_XCF	20		
MQAT_CICS_BRIDGE	21		
MQAT_NOTES_AGENT	22		

表 6.1.13 指定可能な MQAT_*			
コンスタント	値	設定場所	設定方法他
MQAT_TPF	23		
MQAT_USER	25		
MQAT_BROKER	26		
MQAT_QMGR_PUBLISH	26		
MQAT_JAVA	28		
MQAT_DQM	29		
MQAT_CHANNEL_INITIATOR	30		
MQAT_WLM	31		
MQAT_BATCH	32		
MQAT_RRS_BATCH	33		
MQAT_SIB	34		
MQAT_SYSTEM_EXTENSION	35		
MQAT_MCAST_PUBLISH	36		
MQAT_DEFAULT	6		
MQAT_USER_FIRST	65536		
MQAT_USER_LAST	999999999		

Ex. 6.1.8 アプリケーションタイプの指定例

```

$ mqpgf -qm TESTQM -q TQ -m test MQAT_CICS MQPMO_SET_ALL_CONTEXT MQOO_SET_ALL_C
ONTEXT
[16/12/28 19:30:53] 1: message length: 4 put message : test
$
$ mqpgf -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD  ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] .... PutApp1Type[1] ....
....

```

MQMF_*

MQMD.MsgFlags に設定されます。複数指定した場合はそれらの OR で設定します。同時に MQMD_VERSION_2 を指定する必要があります。

下表のコンスタントが指定可能です。

表 6.1.14 指定可能な MQMF_*			
コンスタント	値	設定場所	設定方法他
MQMF_SEGMENTATION_INHIBITED	0x00000000	MQMD.MsgFlags	OR
MQMF_SEGMENTATION_ALLOWED	0x00000001	以下同様	以下同様
MQMF_MSG_IN_GROUP	0x00000008		
MQMF_LAST_MSG_IN_GROUP	0x00000010		
MQMF_SEGMENT	0x00000002		
MQMF_LAST_SEGMENT	0x00000004		
MQMF_NONE	0x00000000		(default)

Ex. 6.1.9 メッセージフラグの指定例

※メッセージフラグを複数指定してPUTする。

```
$ mqpgef -qm TESTQM -q TQ -m test MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMD_VERSION_2
```

```
[16/12/28 20:19:23] 1: message length: 4 put message : test
```

```
$
```

```
$ mqpgef -qm TESTQM -q TQ -brv -hex
```

```
message number: 1
```

```
*StrucId[MD ] Version[2] Report[0x00000000] MsgType[8] Expiry[-1] ....
```

```
....
```

```
GroupId[0x414D51206F6B61716D3830612020202058638EF520002703] MsgSeqNumber[1] Offset[0] MsgFlags[0x0000001E] OriginalLength[4]
```

```
....
```

※MsgFlagsには下記指定したパラメータの OR が設定される。

```
MQMF_MSG_IN_GROUP: 0x00000008
```

```
MQMF_LAST_MSG_IN_GROUP: 0x00000010
```

MQMF_SEGMENT: 0x00000002

MQMF_LAST_SEGMENT: 0x00000004

※MQMF_LAST_SEGMENTが指定された場合、キューマネージャーによって自動的にMQMF_SEGMENTも設定されるので、その場合はMQMF_SEGMENTの指定は省略可能です。

MQMF_LAST_MSG_IN_GROUPが指定された場合、キューマネージャーによって自動的にMQMF_MSG_IN_GROUPも設定されるので、その場合はMQMF_MSG_IN_GROUPの指定は省略可能です。

6.2 MQRFH2パラメータ

MQRFH_*

MQRFH2.Flags に設定されます。下記表の通り、指定できるコンスタントの値はどちらも 0 でかつデフォルトですので、指定しても動作に影響はありません。

下表のコンスタントが指定可能です。

表 6.2.1 指定可能な MQRFH_*			
コンスタント	値	設定場所	設定方法他
MQRFH_NONE	0x00000000	MQRFH2.Flags	Overwrite (default)
MQRFH_NO_FLAGS	0	//	//

6.3 MQCNOパラメータ

MQCNO_* (MQCNO*VERSION*、MQCNO_STRUC_ID 以外)

MQCNO.Options に設定されます。

下表のコンスタントが指定可能です。

表 6.3.1 指定可能な MQCNO_* (MQCNO*VERSION*、MQCNO_STRUC_ID 以外)			
コンスタント	値	設定場所	設定方法他
MQCNO_STANDARD_BINDING	0x00000000	MQCNO.Options	OR
MQCNO_FASTPATH_BINDING	0x00000001	以下同様	以下同様
MQCNO_SERIALIZE_CONN_TAG_Q_MGR	0x00000002		
MQCNO_SERIALIZE_CONN_TAG_QSG	0x00000004		
MQCNO_RESTRICT_CONN_TAG_Q_MGR	0x00000008		
MQCNO_RESTRICT_CONN_TAG_QSG	0x00000010		
MQCNO_HANDLE_SHARE_NONE	0x00000020		
MQCNO_HANDLE_SHARE_BLOCK	0x00000040		
MQCNO_HANDLE_SHARE_NO_BLOCK	0x00000080		
MQCNO_SHARED_BINDING	0x00000100		
MQCNO_ISOLATED_BINDING	0x00000200		
MQCNO_LOCAL_BINDING	0x00000400		
MQCNO_CLIENT_BINDING	0x00000800		
MQCNO_ACCOUNTING_MQI_ENABLED	0x00001000		
MQCNO_ACCOUNTING_MQI_DISABLED	0x00002000		
MQCNO_ACCOUNTING_Q_ENABLED	0x00004000		
MQCNO_ACCOUNTING_Q_DISABLED	0x00008000		
MQCNO_NO_CONV_SHARING	0x00010000		
MQCNO_ALL_CONVS_SHARE	0x00040000		
MQCNO_CD_FOR_OUTPUT_ONLY	0x00080000		

表 6.3.1 指定可能な MQCNO_* (MQCNO*VERSION*、MQCNO_STRUC_ID 以外)			
コンスタント	値	設定場所	設定方法他
MQCNO_USE_CD_SELECTION	0x00100000		
MQCNO_RECONNECT_AS_DEF	0x00000000		
MQCNO_RECONNECT	0x01000000		
MQCNO_RECONNECT_DISABLED	0x02000000		
MQCNO_RECONNECT_Q_MGR	0x04000000		
MQCNO_ACTIVITY_TRACE_ENABLED	0x08000000		
MQCNO_ACTIVITY_TRACE_DISABLED	0x10000000		
MQCNO_NONE	0x00000000		(default)

Ex. 6.3.1 SHARED_BINDINGで接続する例

※MQCNO_*の指定なし（デフォルト）、MQCNO_STANDARD_BINDING、MQCNO_SHARED_BINDING、MQCNO_NONE を指定すると、MQCNO_SHARED_BINDING で接続される。

```
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN
stop before calling MQOPEN().
Hit Any Key!!!
```

※以下、mqpcfは他の端末から実行

```
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003301) TYPE(CONN) CONNOPTS(MQ
CNO_SHARED_BINDING)
```

```
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN MQCNO_STANDARD_BINDING
stop before calling MQOPEN().
Hit Any Key!!!
```

```
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003501) TYPE(CONN) CONNOPTS(MQ
CNO_SHARED_BINDING)
```

```
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN MQCNO_SHARED_BINDING
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003502) TYPE(CONN) CONNOPTS(MQ
CNO_SHARED_BINDING)
```

```
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN MQCNO_NONE
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
1: CONN(414D51436F6B61716D3830612020202058746D1420003503) TYPE(CONN) CONNOPTS(MQ
CNO_SHARED_BINDING)
-----
```

Ex. 6.3.2 FASTPATH_BINDINGで接続する例

※MQCNO_FASTPATH_BINDINGで接続する場合は、ユーザーIDも“mqm”にする必要があります。（groupIdがmqmでも不可）
mqm以外のユーザーで接続した場合、下記のエラーが発生して接続に失敗します。

```
$ id
uid=xxx(mq80) gid=xxx(mqm) groups=1(yyyy)
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN MQCNO_FASTPATH_BINDING
MQCONN fail : TESTQM CompCd=02 ReasonCd=2012
!!! Queue Manager Connect Fail !!!
$ mqrc 2012
```

2012 0x000007dc MQRC_ENVIRONMENT_ERROR

```
$ su mqm
$id
uid=xxx(mqm) gid=xxx(mqm) groups=1(yyyy)

$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN MQCNO_FASTPATH_BINDING
stop before calling MQOPEN().
Hit Any Key!!!
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
$ 1: CONN(414D51436F6B61716D3830612020202058746D1420003901) TYPE(CONN) CONNOPTS
(MQCNO_FASTPATH_BINDING)
$
-----
```

トラステッド・アプリケーションの制約事項（製品マニュアル）

=====

WebSphere® MQ (UNIX® システム用) では、すべての MQI 呼び出しに有効なユーザー ID およびグループ ID として、mqm を使用する必要がある。これらの ID は、認証が必要な非 MQI 呼び出し（例えば、ファイルのオープン）を行う前に変更できますが、次の MQ

I 呼び出しを行う前に mqm に戻しておく必要があります。

=====

※さらに、FASTPATH_BINDINGで接続する場合、スレッド・ライブラリーとリンクされている必要があります。mqpgfプログラムはMQIのスレッド・ライブラリーとリンクされています。

MQCONN 呼び出しを使用したキュー・マネージャーへの接続（製品マニュアル）

=====

MQCNO_FASTPATH_BINDING

トラステッド・アプリケーション では、WebSphere MQ アプリケーションとローカル・キュー・マネージャー・エージェントが同じプロセスになるように暗黙的に指定します。エージェント・プロセスではキュー・マネージャーにアクセスするためにインターフェースを使用する必要がなくなったので、これらのアプリケーションはキュー・マネージャーの拡張機能になります。これは、MQCONN 呼び出しで MQCNO_FASTPATH_BINDING オプションによって定義されます。

トラステッド・アプリケーションは、WebSphere MQ のスレッド・ライブラリーにリンクする必要があります。トラステッド・アプリケーションとして実行する WebSphere MQ アプリケーションの設定方法については、「WebSphere MQ アプリケーション・プログラミング・リファレンス」を参照してください。

=====

Ex. 6.3.3 ISOLATED_BINDINGで接続する例

※ MQCNO_ISOLATED_BINDING を指定した場合

```
$ mqpgf -qm TESTQM -q INQ1 -s MQOPEN MQCNO_ISOLATED_BINDING
```

```
stop before calling MQOPEN().
```

```
Hit Any Key!!!
```

```
$ mqpcf con -qm TESTQM conn -ap mqpgf CONNOPTS
```

```
1: CONN(414D51436F6B61716D3830612020202058746D1420003C01) TYPE(CONN) CONNOPTS(MQ  
CNO_ISOLATED_BINDING)
```

6.4 MQOPENオプション

MQOO_*

MQOPEN() 関数の引数 Options に設定されます。後続のMQI呼び出しの種類、MQGET() / MQPUT() / MQSET() / MQINQ()によって自動設定されるオプションが変わります。（下表参照）

下表のコンスタントが指定可能です。

表 6.4.1 指定可能な MQOO_*			
コンスタント	値	設定場所	設定方法他
MQOO_BIND_AS_Q_DEF	0x00000000	MQOPEN() の引数 Options	OR
MQOO_READ_AHEAD_AS_Q_DEF	0x00000000	以下同様	以下同様
MQOO_INPUT_AS_Q_DEF	0x00000001		MQGET() を呼び出す場合で MQOO_BROWSE または MQOO_INPUT*がなにも指定されていない場合に設定される。
MQOO_INPUT_SHARED	0x00000002		
MQOO_INPUT_EXCLUSIVE	0x00000004		
MQOO_BROWSE	0x00000008		ブラウズが指定された場合に自動設定される。
MQOO_OUTPUT	0x00000010		MQPUT() を呼び出す場合に自動設定される。
MQOO_INQUIRE	0x00000020		MQINQ() を呼び出す場合に自動設定される。
MQOO_SET	0x00000040		MQSET() を呼び出す場合に自動設定される。
MQOO_SAVE_ALL_CONTEXT	0x00000080		
MQOO_PASS_IDENTITY_CONTEXT	0x00000100		
MQOO_PASS_ALL_CONTEXT	0x00000200		

表 6.4.1 指定可能な MQOO_*

コンスタント	値	設定場所	設定方法他
MQOO_SET_IDENTITY_CONTEXT	0x00000400		
MQOO_SET_ALL_CONTEXT	0x00000800		
MQOO_ALTERNATE_USER_AUTHORITY	0x00001000		
MQOO_FAIL_IF QUIESCING	0x00002000		
MQOO_BIND_ON_OPEN	0x00004000		
MQOO_BIND_ON_GROUP	0x00400000		
MQOO_BIND_NOT_FIXED	0x00008000		
MQOO_CO_OP	0x00020000		
MQOO_NO_READ_AHEAD	0x00080000		
MQOO_READ_AHEAD	0x00100000		
MQOO_NO_MULTICAST	0x00200000		
MQOO_RESOLVE_LOCAL_Q	0x00040000		
MQOO_RESOLVE_LOCAL_TOPIC	0x00040000		
MQOO_RESOLVE_NAMES	0x00010000		

6.5 MQODパラメータ

MQOT_*

MQOD.ObjectType に設定されます。（「4. 基本的なテストの実施方法 — MQINQ呼び出し」参照）

下表のコンスタントが指定可能です。

表 6.5.1 指定可能な MQOT_*			
コンスタント	値	設定場所	設定方法他
MQOT_NONE	0	MQOD.ObjectType	Overwrite 以下同様
MQOT_Q	1	以下同様	(default)
MQOT_NAMELIST	2		
MQOT_PROCESS	3		
MQOT_Q_MGR	5		
MQOT_TOPIC	8		
MQOT_CLNTCONN_CHANNEL	1014		

MQOD_* (MQOD*VERSION*)

MQOD.Version に設定されます。（「4. 基本的なテストの実施方法 — 配布リストの使用」参照）

下表のコンスタントが指定可能です。

表 6.5.2 指定可能な MQOD_*(MQOD*VERSION*)			
コンスタント	値	設定場所	設定方法他
MQOD_VERSION_1	1	MQOD.Version	Overwrite (default)
MQOD_VERSION_2	2	以下同様	以下同様

表 6.5.2 指定可能な MQOD_*(MQOD*VERSION*)			
コ ン ス タ ン ト	値	設 定 場 所	設 定 方 法 他
MQOD_VERSION_3	3		
MQOD_VERSION_4	4		
MQOD_CURRENT_VERSION	4		

6.6 MQPMOパラメータ

MQPMO_* (MQPMO*VERSION*)

MQPMO.Version に設定されます。（「4. 基本的なテストの実施方法 — メッセージ・プロパティの指定」参照）

下表のコンスタントが指定可能です。

表 6.6.1 指定可能な MQPMO*VERSION*			
コンスタント	値	設定場所	設定方法他
MQPMO_VERSION_1	1	MQPMO.Version	Overwrite (default)
MQPMO_VERSION_2	2	以下同様	以下同様
MQPMO_VERSION_3	3		
MQPMO_CURRENT_VERSION	3		

MQPMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)

MQPMO.Options に設定されます。2つ以上指定した場合は、そのORが設定されます。

下表のコンスタントが指定可能です。

表 6.6.2 指定可能な MQPMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)			
コンスタント	値	設定場所	設定方法他
MQPMO_SYNCPOINT	0x00000002	MQPMO.Options	OR
MQPMO_NO_SYNCPOINT	0x00000004	以下同様	以下同様
MQPMO_DEFAULT_CONTEXT	0x00000020		
MQPMO_NEW_MSG_ID	0x00000040		
MQPMO_NEW_CORREL_ID	0x00000080		

表 6.6.2 指定可能な MQPMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)

コnスタント	値	設定場所	設定方法他
MQPMO_PASS_IDENTITY_CONTEXT	0x00000100		
MQPMO_PASS_ALL_CONTEXT	0x00000200		
MQPMO_SET_IDENTITY_CONTEXT	0x00000400		
MQPMO_SET_ALL_CONTEXT	0x00000800		
MQPMO_ALTERNATE_USER_AUTHORITY	0x00001000		
MQPMO_FAIL_IF QUIESCING	0x00002000		
MQPMO_NO_CONTEXT	0x00004000		
MQPMO_LOGICAL_ORDER	0x00008000		
MQPMO_ASYNC_RESPONSE	0x00010000		
MQPMO_SYNC_RESPONSE	0x00020000		
MQPMO_RESOLVE_LOCAL_Q	0x00040000		
MQPMO_WARN_IF_NO_SUBS_MATCHED	0x00080000		
MQPMO_RETAIN	0x00200000		
MQPMO_MD_FOR_OUTPUT_ONLY	0x00800000		
MQPMO_SCOPE_QMGR	0x04000000		
MQPMO_SUPPRESS_REPLYTO	0x08000000		
MQPMO_NOT_OWN_SUBS	0x10000000		
MQPMO_RESPONSE_AS_Q_DEF	0x00000000		
MQPMO_RESPONSE_AS_TOPIC_DEF	0x00000000		
MQPMO_NONE	0x00000000		(default)
MQPMO_PUB_OPTIONS_MASK	0x00200000		

MQPMRF_*

MQPMO.PutMsgRecFields に設定されます。2つ以上指定した場合は、そのORが設定さ

れます。（「4. 基本的なテストの実施方法 — 配布リストの使用」参照）

下表のコンスタントが指定可能です。

表 6.6.3 指定可能な MQPMRF_*			
コンスタント	値	設定場所	設定方法他
MQPMRF_MSG_ID	0x00000001	MQPMO.PutMsgRecFields	OR
MQPMRF_CORREL_ID	0x00000002	以下同様	以下同様
MQPMRF_GROUP_ID	0x00000004		
MQPMRF_FEEDBACK	0x00000008		
MQPMRF_ACCOUNTING_TOKEN	0x00000010		
MQPMRF_NONE	0x00000000		(default)

6.7 MQGMOパラメータ

MQGMO_* (MQGMO*VERSION*)

MQGMO.Version に設定されます。（「4. 基本的なテストの実施方法 — メッセージ・プロパティの指定」参照）

下表のコンスタントが指定可能です。

表 6.7.1 指定可能な MQGMO*VERSION*			
コンスタント	値	設定場所	設定方法他
MQGMO_VERSION_1	1	MQGMO.Version	Overwrite (default)
MQGMO_VERSION_2	2	以下同様	以下同様
MQGMO_VERSION_3	3		
MQGMO_VERSION_4	4		
MQGMO_CURRENT_VERSION	4		

MQGMO_* (MQPMO*VERSION*、MQPMO_STRUC_ID、MQPMO_LENGTH_* 以外)

MQGMO.Options に設定されます。2つ以上指定した場合は、そのORが設定されます。

※デフォルトはMQGMO_NO_WAIT+ MQGMO_PROPERTIES_AS_Q_DEF（どちらも値は0x00000000）

下表のコンスタントが指定可能です。

表 6.7.2 指定可能な MQGMO_* (MQGMO*VERSION*、MQGMO_STRUC_ID、MQGMO_LENGTH_* 以外)			
コンスタント	値	設定場所	設定方法他
MQGMO_WAIT	0x00000001	MQGMO.Options	OR 以下同様
MQGMO_NO_WAIT	0x00000000	以下同様	(default)

表 6.7.2 指定可能な MQGMO_* (MQGMO*VERSION*、MQGMO_STRUC_ID、MQGMO_LENGTH_* 以外)

コnスタント	値	設定場所	設定方法他
MQGMO_SET_SIGNAL	0x00000008		
MQGMO_FAIL_IF_QUIESCING	0x00002000		
MQGMO_SYNCPOINT	0x00000002		
MQGMO_SYNCPOINT_IF_PERSISTENT	0x00001000		
MQGMO_NO_SYNCPOINT	0x00000004		
MQGMO_MARK_SKIP_BACKOUT	0x00000080		
MQGMO_BROWSE_FIRST	0x00000010		
MQGMO_BROWSE_NEXT	0x00000020		
MQGMO_BROWSE_MSG_UNDER_CURSOR	0x00000800		
MQGMO_MSG_UNDER_CURSOR	0x00000100		
MQGMO_LOCK	0x00000200		
MQGMO_UNLOCK	0x00000400		
MQGMO_ACCEPT_TRUNCATED_MSG	0x00000040		
MQGMO_CONVERT	0x00004000		
MQGMO_LOGICAL_ORDER	0x00008000		
MQGMO_COMPLETE_MSG	0x00010000		
MQGMO_ALL_MSGS_AVAILABLE	0x00020000		
MQGMO_ALL_SEGMENTS_AVAILABLE	0x00040000		
MQGMO_MARK_BROWSE_HANDLE	0x00100000		
MQGMO_MARK_BROWSE_CO_OP	0x00200000		
MQGMO_UNMARK_BROWSE_CO_OP	0x00400000		
MQGMO_UNMARK_BROWSE_HANDLE	0x00800000		
MQGMO_UNMARKED_BROWSE_MSG	0x01000000		
MQGMO_PROPERTIES_FORCE_MQRFH2	0x02000000		
MQGMO_NO_PROPERTIES	0x04000000		

表 6.7.2 指定可能な MQGMO_* (MQGMO*VERSION*、MQGMO_STRUC_ID、MQGMO_LENGTH_* 以外)

コンスタント	値	設定場所	設定方法他
MQGMO_PROPERTIES_IN_HANDLE	0x08000000		
MQGMO_PROPERTIES_COMPATIBILITY	0x10000000		
MQGMO_PROPERTIES_AS_Q_DEF	0x00000000		(default)
MQGMO_NONE	0x00000000		
MQGMO_BROWSE_HANDLE	MQGMO_BROWSE_FIRST MQGMO_UNMARKED_BROWSE_MSG MQGMO_MARK_BROWSE_HANDLE		
MQGMO_BROWSE_CO_OP	MQGMO_BROWSE_FIRST MQGMO_UNMARKED_BROWSE_MSG MQGMO_MARK_BROWSE_CO_OP		

Ex. 6.7.1 MQGMO_CONVERTを指定してユーザーデータ部をコード変換する例

```
-----
$ mqpgef -qm TESTQM -q TQ -m "1234" -cc 943 MQFMT_STRING
[17/01/10 20:49:49] 1: message length: 4 put message : 1234
```

※MQGMO_CONVERTを指定しない場合

```
$ mqpgef -qm TESTQM -q TQ -br
message number: 1
*StrucId[MD ] .... CodedCharSetId[943] Format[MQSTR ] ....

data length: 4
00000000: 3132 3334 ' 1234 '
```

※MQGMO_CONVERTと変換先のCCIDを指定した場合

```
$ mqpgef -qm TESTQM -q TQ -br -cc 930 MQGMO_CONVERT
message number: 1
*StrucId[MD ] .... CodedCharSetId[930] Format[MQSTR ] ....
```

data length: 4

00000000: F1F2 F3F4

' 𠄎𠄎𠄎𠄎 '

※指定したCCSID(930=EBCDIC)に変換される。

Ex. 6.7.2 MQGMO_CONVERTを指定してRFH2とユーザーデータを同時にコード変換する例

※MQMD.Encoding=273(Big Endian)、MQMD.CharSetId=943(SHIFT_JIS) (これらはMQMDではなくRFH2のエンコーディング、文字コードを表していることに注意)、MQRFH2.CodedCharSetId=5050(eucJP) (これはRFH2自体やMQRFH2.NameValueDataではなく、NameValueDataにつづくユーザーデータの文字コードを表していることに注意)に指定

※ユーザーデータとして指定している"8eb1 8eb2 8eb3" はeucJPの半角"アイウ"に相当する16進表記の文字列

```
$ mqpgf -qm TESTQM -q TQ -ec 273 -cc 943 -mx 8eb18eb28eb3 -rf MQFMT_STRING -fg 1
00 -nd "test1,test22,test333" MQFMT_RF_HEADER_2 -rc 5050
[17/01/13 15:04:06] 1: message length: 6 put message : 0x8EB18EB28EB3
```

※コード変換せずにPUTしたメッセージの状態を確認

```
$ mqpgf -qm TESTQM -q TQ -brv
```

message number: 1

*StrucId[MD] Encoding[273] CodedCharSetId[943] Format[MQHRF2]

*StrucId[RFH] Version[2] StrucLength[72] Encoding[273] CodedCharSetId[5050] Format[MQSTR] Flags[100] NameValueCCSID[1208]

NameValueLength[8] NameValueData[test1]

NameValueLength[8] NameValueData[test22]

NameValueLength[8] NameValueData[test333]

data length: 78

00000000: 8EB1 8EB2 8EB3 ' 竺軸宍 '

※ MQMD.Encoding=546(Little Endian)、MQMD.CharSetId=930(EBCDIC)を指定してコード変換を実行

```
$ mqpgf -qm TESTQM -q TQ -br MQGMO_CONVERT -ec 546 -cc 930
```

message number: 1

*StrucId[MD] Encoding[546] CodedCharSetId[930] Format[MQHRF2]

data length: 75

00000000: D9C6 C840 0200 0000 4800 0000 2202 0000 'ルニネ@....H..."...'

00000010: A203 0000 D4D8 E2E3 D940 4040 6400 0000 '「...ヤリ粵ル@@@d...'

00000020: B804 0000 0800 0000 7465 7374 3120 2020 'ク.....test1 '

00000030: 0800 0000 7465 7374 3232 2020 0800 0000 '....test22'

00000040: 7465 7374 3333 3320 **8182 83** 'test333 𐤀. 𐤀'

※MQRFH2自体の文字コードが、SHIFTJIS (CCSID 943) からEBCDIC (CCSID 930) に変換されている。

00000000-00000003 : D9C6 C840 = EBCDICの"RFH "

※MQRFH2自体のエンコードが、Little Endian (Encoding 546) に変換される。

00000004-00000007 : 0200 0000

※MQRFH2.Encodingが546 (Little Endian) に設定されている

0000000C-0000000F : 2202 0000 = 0000 0222 (Big Endian) = 546 (10進)

※MQRFH2.CodedCharSetIdがEBCDIC (CCSID 930) に設定されている

00000010-00000011 : A203 0000 = 0000 03A2 (Big Endian) = 930 (10進) EBCDICのCCSID

※MQRFH2.NameValueCCSIDはUTF-8 (CCSID 1208) のまま (NameValueDataは変換対象ではないことに注意)

00000020-00000023 : B804 0000 = 0000 04B8 (Big Endian) = 1208 (10進) UTF-8のCCSID

※ユーザーデータがeuc.JPから日本語EBCDICにに変換されている。

00000048-0000004a : 8182 83 = 日本語EBCDICの半角"アイ"

Ex. 6.7.3 MQGMO_CONVERTを指定してPCFメッセージをコード変換する例

※複数種類の文字コード (SHIFT_JIS:943, UTF-8:1208, EBCDIC:930) が混在するPCFメッセージをPUTする。PCFメッセージ内で文字コードが混在する場合は、MQMD.CharSetId にMQCCSI_EMBEDDED(-1)を設定する。

```
$ mqpgf -qm TESTQM -q TQ -pcf sample3.def MQCCSI_EMBEDDED
```

Command : 99

Id : 1111, MQCFT_BYTE_STRING : 10, 1234567890

Id : 2222, MQCFT_BYTE_STRING_FILTER : 1 5 1A2B3

Id : 3501, MQCFT_STRING : **943** 3 アイ

Id : 4444, MQCFT_STRING_FILTER : 2 **1208** 9 .スア.スイ.スウ

Id : 5555, MQCFT_STRING_LIST : **930** 5 3 [0 0], [𐤀 𐤁], [𐤂 𐤃], [. .], [. .]

Id : 6666, MQCFT_INTEGER : 1234567890

Id : 7777, MQCFT_INTEGER_FILTER : 6 -3

Id : 8888, MQCFT_INTEGER_LIST : 3 [1234], [-1], [5678]

Id : 9999, MQCFT_INTEGER64 : 9223372036854775807

Id : 1234, MQCFT_INTEGER64_LIST : 5 [4294967294], [4294967291], [-5], [9223372036854775803], [4294967290]

[17/01/13 16:51:13] 1: put from sample3.def

※コード変換せずにPUTしたメッセージの状態を確認

```
$ mqpgef -qm TESTQM -q TQ -brv -hex
```

message number: 1

```
*StrucId[MD ] .... Encoding[273] CodedCharSetId[-1] Format[MQPCF ] ....
```

....

```
*MQCFH(MQCFT_USER) ....
```

....

```
(MQCFT_STRING) Type[4] StrucLength[24] Parameter[3501] CodedCharSetId[943] StringLength[3] String[0xB1B2B3]
```

```
(MQCFT_STRING_FILTER) Type[14] StrucLength[36] Parameter[4444] Operator[2] CodedCharSetId[1208] FilterValueLength[9] FilterValue[0xEFBDDB1EFBDDB2EFBDDB3]
```

```
(MQCFT_STRING_LIST) Type[6] StrucLength[40] Parameter[5555] CodedCharSetId[930] Count[5] StringLength[3] String1[0xF1F1F1] String2[0xF1F2F3] String3[0xF3F3F3] String4[0xF5F6F7] String5[0xF5F5F5]
```

....

※CCSID 943 (SHIFT_JIS) の 0xB1 B2 B3 は 半角"アイ"

※CCSID 1208 (UTF-8) の 0xEFBDDB1 EFBDB2 EFBDB3 は 半角"アイ"

※CCSID 930 (EBCDIC) の 0xF0 F1 F2 F3 ... F9 は 半角"0123 ... 9"

※ MQMD.CharSetId=5050 (eucJP) を指定してコード変換を実行

```
$ mqpgef -qm TESTQM -q TQ -brv MQGMO_CONVERT -cc 5050 -hex
```

message number: 1

```
*StrucId[MD ] .... CodedCharSetId[5050] Format[MQPCF ] ....
```

....

```
*MQCFH(MQCFT_USER) ....
```

....

```
(MQCFT_STRING) Type[4] StrucLength[28] Parameter[3501] CodedCharSetId[5050] StringLength[6] String[0x8EB18EB28EB3]
```

```
(MQCFT_STRING_FILTER) Type[14] StrucLength[32] Parameter[4444] Operator[2] CodedCharSetId[5050] FilterValueLength[6] FilterValue[0x8EB18EB28EB3]
```

```
(MQCFT_STRING_LIST) Type[6] StrucLength[40] Parameter[5555] CodedCharSetId[5050] Count[5] StringLength[3] String1[0x313131] String2[0x313233] String3[0x333333] String4[0x353637] String5[0x353535]
```

....

※全てのパラメータのCodedCharSetIdが5050 (eucJP) に設定されている。

※0x8EB1 8EB2 8EB3 はeucJP (5050) の半角"アイ"

※0x30 31 32 33 ... 39 はeucJP (5050) の半角"0123 ... 9"

MQWI_*

MQGMO.WaitInterval に設定されます。このフィールドのデフォルト値は 0（待ち時間なし）です。

下表のコンスタントが指定可能です。

表 6.7.3 指定可能な MQWI_*			
コンスタント	値	設定場所	設定方法他
MQWI_UNLIMITED	(-1)	MQGMO.WaitInterval	Overwrite

MQMO_*

MQGMO.MatchOptions に設定されます。このフィールドのデフォルト値は MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID です。MQMO_* を 1 つでも指定した場合は、デフォルトのオプションは使用されなくなり、また MQMO_* を 2 つ以上指定した場合は、その引数のORが設定されます。-mi (MsgId) 、-ci (CorrelId) または -gi (GroupId) と組みあせて使用します。

下表のコンスタントが指定可能です。

表 6.7.4 指定可能な MQMO_*			
コンスタント	値	設定場所	設定方法他
MQMO_MATCH_MSG_ID	0x00000001	MQGMO.MatchOptions	OR
MQMO_MATCH_CORREL_ID	0x00000002	以下同様	以下同様
MQMO_MATCH_GROUP_ID	0x00000004		
MQMO_MATCH_MSG_SEQ_NUMBER	0x00000008		
MQMO_MATCH_OFFSET	0x00000010		
MQMO_MATCH_MSG_TOKEN	0x00000020		
MQMO_NONE	0x00000000		

Ex. 6.7.4 MQMO_MATCH_MSG_IDを指定して特定のメッセージのみをGET

※3件メッセージをPUT

```
$ mqpgf -qm TESTQM -q TQ -m test -n 3
[17/01/10 21:29:58] 1: message length: 4 put message : test
[17/01/10 21:29:58] 2: message length: 4 put message : test
[17/01/10 21:29:58] 3: message length: 4 put message : test
```

※自動採番された MsgId を確認

```
$ mqpgf -qm TESTQM -q TQ -br -r
message number: 1
*StrucId[MD ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005E03]
....
message number: 2
*StrucId[MD ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005E04]
.....
message number: 3
*StrucId[MD ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005E05]
....
```

※2番目のメッセージの MsgId を指定してGETする。（デフォルトでMQMO_MATCH_MSG_IDは設定されているので、引数での指定は必須ではありません。）

```
$ mqpgf -qm TESTQM -q TQ -mi 0x414D51206F6B61716D3830612020202058746D1420005E04
MQMO_MATCH_MSG_ID
[17/01/10 21:35:34] 1: message length: 4 get message : test
```

```
$ mqpgf -qm TESTQM -q TQ -br -r
message number: 1
*StrucId[MD ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005E03]
....
message number: 2
*StrucId[MD ] .... MsgId[0x414D51206F6B61716D3830612020202058746D1420005E05]
....
no message available : TQ CompCd=02 ReasonCd=2033
※2番目のメッセージのみがGETされている。
-----
```

6.8 MQCLOSEオプション

MQCO_*

MQCLOSE() 関数の引数 Options に設定されます。このフィールドのデフォルト値は MQCO_NONE です。MQCO_* を 2 つ以上指定した場合は、その引数の OR が設定されます。

表 6.8.1 指定可能な MQCO_*			
コンスタント	値	設定場所	設定方法他
MQCO_IMMEDIATE	0x00000000	MQCLOSE() の引数 Options	OR
MQCO_NONE	0x00000000	以下同様	以下同様
MQCO_DELETE	0x00000001		
MQCO_DELETE_PURGE	0x00000002		
MQCO_KEEP_SUB	0x00000004		
MQCO_REMOVE_SUB	0x00000008		
MQCO_QUIESCE	0x00000020		

Ex. 6.8.1 永続動的キューをMQCLOSE()時に削除する例

※永続動的キューを作成するためにSYSTEM. DURABLE. MODEL. QUEUEを指定する。

```
$ mqpgf -qm SampleQM -q SYSTEM. DURABLE. MODEL. QUEUE -dq DYNAMICQ -s MQCLOSE MQCO_DELETE
no message available : SYSTEM. DURABLE. MODEL. QUEUE CompCd=02 ReasonCd=2033
MQCMIT success : CompCd=00 ReasonCd=00
stop before calling MQCLOSE().
Hit Any Key!!!
```

※他のターミナルから作成された永続動的キューを確認する。

```
$ mqpcf que -qm SampleQM -q DYNAMICQ TYPE DEFTYPE
1: QUEUE(DYNAMICQ) TYPE(QLOCAL) DEFTYPE(PERMDYN)
```

※元のターミナルで任意のキーを入力して処理を継続させMQCLOSE()を呼び出す、


```
$ mqpcf que -qm SampleQM -q DYNAMICQ TYPE DEFTYPE
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommand
CC=[2], mqCommandRC=[2085]
MQExecute : Command Server Error. mqExecuteCC=[2], mqExecuteRC=[3008], mqCommand
CC=[2], mqCommandRC=[3008]
```

mqrc 2085

2085 0x00000825 MQRC_UNKNOWN_OBJECT_NAME

※永続動的キューが削除されている。

6.9 MQSETMPオプション

MQPD_*

MQPD.Contextに設定されます。

下表のコンスタントが指定可能です。

表 6.9.1 指定可能な MQPD_*			
コンスタント	値	設定場所	設定方法他
MQPD_USER_CONTEXT	0x00000000	MQPD.Context	Overwrite
MQPD_NO_CONTEXT	0x00000001	MQPD.Context	Overwrite

Ex. 6.9.1 入力の識別、起点、ユーザー・コンテキストを引き継いでリキューする例

※まず、メッセージ・プロパティをユーザー・コンテキストに関連付けるには、MQPD_USER_CONTEXT を指定することが必要です。

```
$ mqpgf -qm SampleQM -q SampleQ -m "test1" -smp "MQTYPE_BOOLEAN:boolean1:TRUE, MQTYPE_BYTE_STRING:byteString:0102feff" MQPMO_VERSION_3 MQPD_USER_CONTEXT
[18/02/02 09:38:14] 1: message length: 5 put message : test1
```

※MQPD_USER_CONTEXTを設定すると、メッセージ・プロパティのXMLのアトリビュートに” context='user' ” が付加されます。

```
$ mqpgf -qm SampleQM -q SampleQ -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[273] CodedCharSetId[819] Format[MQHRF2 ] Priority[0] Persistence[0] MsgId[0x414D512053616D706C65514D202020205A73B22B20002412] CorrelId[0x00000000000000000000000000000000] BackoutCount[0] ReplyToQ[ ] ReplyToQMgr[SampleQM]
] UserIdentifier[mqm ] AccountingToken[0x0534343033310000000000000000000000000000000000000000000000000000] ApplIdentityData[ ] PutApplType[13] PutApplName[mqpgf] ] PutDate[20180202] PutTime[00381410] ApplOriginData[ ]
```

```
GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offset[0] MsgFlags[0] OriginalLength[-1]
```

```
*StrucId[RFH ] Version[2] StrucLength[164] Encoding[273] CodedCharSetId[819] For
```

```
mat[          ] Flags[0] NameValueCCSID[1208]
NameValueLength[124] NameValueData[<usr><boolean1 dt='boolean' context='user'>1<
/boolean1><byteString dt='bin.hex' context='user'>0102FEFF</byteString></usr> ]
data length: 169
00000000: 7465 7374 31          'test1          '
```

※リキュー時に、MQGMO_PROPERTIES_IN_HANDLEを指定し、一旦内部でメッセージ・プロパティを MsgHandle によって使用可能にするようにします。MQ00_SAVE_ALL_CONTEXT オプションを指定しない場合は、元のメッセージのコンテキスト情報は保管されません。さらに全てのコンテキストを渡すために、MQ00_PASS_ALL_CONTEXTとMQPMO_PASS_ALL_CONTEXTを指定します。mqpgfはMQPMO_PASS_*が指定されている場合、入力キューのハンドルをMQPMO.Contextに設定します。

```
$ mqpgf -qm SampleQM -q SampleQ -oq SampleQ2 MQGMO_PROPERTIES_IN_HANDLE MQGMO_VE
RSION_4 MQ00_SAVE_ALL_CONTEXT -ss MQ00_PASS_ALL_CONTEXT MQPMO_PASS_ALL_CONTEXT
MQPMO_VERSION_3
[18/01/31 19:35:23] 1: message length: 5 get message : test1
[18/01/31 19:35:23] 1: message length: 5 put message : test1
```

```
$ mqpgf -qm SampleQM -q SampleQ2 -brv
message number: 1
*StrucId[MD ] Version[2] Report[0] MsgType[8] Expiry[-1] Feedback[0] Encoding[2
73] CodedCharSetId[819] Format[MQHRF2 ] Priority[0] Persistence[0] MsgId[0x414D
512053616D706C65514D202020205A73B22B20002412] CorrelId[0x00000000000000000000
0000000000000000000000000000] BackoutCount[0] ReplyToQ[
] ReplyToQMgr[SampleQM
] UserIdentifier[mqm ] AccountingToken[0x05343430333100000000000000000000
0000000000000000000000000000000006] ApplIdentityData[
] PutApplType[13] PutApplName[mqpgf ] PutDate[201
80202] PutTime[00381410] ApplOriginData[ ]
```

```
GroupId[0x0000000000000000000000000000000000000000000000000000000000000000] MsgSeqNumber[1] Offs
et[0] MsgFlags[0] OriginalLength[-1]
```

```
*StrucId[RFH ] Version[2] StrucLength[164] Encoding[273] CodedCharSetId[819] For
mat[          ] Flags[0] NameValueCCSID[1208]
NameValueLength[124] NameValueData[<usr><boolean1 dt='boolean' context='user'>1<
/boolean1><byteString dt='bin.hex' context='user'>0102FEFF</byteString> </usr>
]
data length: 169
00000000: 7465 7374 31          'test1          '
```

MQCMIT success : CompCd=00 ReasonCd=00

6.10 MQINQMPオプション

MQIMPO_*

MQIMPO.Options に設定されます。MQIMPO_* を2つ以上指定した場合は、その引数のORが設定されます。本プログラムはMQINQMPを呼び出し時、引数NameにMQPROP_INQUIRE_ALLを指定してすべてのメッセージプロパティを照会します。

下表のコンスタントが指定可能です。

表 6.10.1 指定可能な MQIMPO_*			
コンスタント	値	設定場所	設定方法他
MQIMPO_CONVERT_TYPE	0x00000000	MQIMPO.Options	OR
MQIMPO_QUERY_LENGTH	0x00000001	以下同様	以下同様
MQIMPO_INQ_FIRST	0x00000000		(default)
MQIMPO_INQ_NEXT	0x00000008		2 件名以降のプロパティの呼び出し時には自動的にこの値がセットされます。
MQIMPO_INQ_PROP_UNDER_CURSOR	0x00000010		
MQIMPO_CONVERT_VALUE	0x00000020		
MQIMPO_NONE	0x00000000		

Ex. 6.10.1 MQIMPO_CONVERT_VALUEを指定してプロパティのデータ変換を行う例

※MQTYPE_STRINGとMQTYPE_INT32のプロパティを指定してPUTする。

```
$ mqpgf -qm mqm90a -q TQ -m "test" -smp "MQTYPE_STRING:string:123,MQTYPE_INT32:integer:1" MQPMO_VERSION_3
```

```
[17/01/12 20:26:24] 1: message length: 4 put message : test
```

```
$ mqpgf -qm mqm90a -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VERSION_4 -br
message number: 1
```

```
*StrucId[MD ] .... Encoding[546] CodedCharSetId[1208] ....
```

```
***Message properties***
```

```
string : '123'
int32 : 1
MQINQMP failed : CompCd=02 ReasonCd=2471
....
```

※メッセージ・プロパティのエンコーディングを273、CCSIDを930(EBCDIC)に変換するように指定する。さらに -hex を指定して16進表記で表示させる。

```
$ mqpgf -qm mqm90a -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VERSION_4 -br MQIMPO_CONVERT_VALUE -br -pe 273 -pc 930 -hex
message number: 1
*StrucId[MD ] .... Encoding[546] CodedCharSetId[1208] ....
```

Message properties

```
0xABB39B717668 : '0xF1F2F3'
0x7176B3F3F2 : 16777216
MQINQMP failed : CompCd=02 ReasonCd=2471
....
```

※MQIMPO_CONVERT_VALUEを指定するとプロパティ名も変換されていることが分かります。F1F2F3はEBCDICでの数字の'123'、16777216は0x01を32Bitでエンディアンを逆にした場合の10進数の表現です。(0x01000000=16777216)

※MQINQMPを呼び出し時、引数NameにMQPROP_INQUIRE_ALLを指定してすべてのメッセージプロパティを照会します。理由コード: 2471(MQRC_PROPERTY_NOT_AVAILABLE)でMQINQMP()が終了するまで繰り返し表示します。

6.11 MQCRTMHオプション

MQCMHO_*

MQCRTMH.Options に設定されます。

※ MQCMHO_NO_VALIDATION については下記ドキュメント修正のAPARがありますので、ご注意ください。

PM96938: WMQ BATCH JOB FAILS WITH MQRC 2442 WHEN USING MQCMHO OPTION (MQCMHO_NO_VALIDATION)

<http://www-01.ibm.com/support/docview.wss?uid=swg1PM96938>

下表のコンスタントが指定可能です。

表 6.11.1 指定可能な MQCMHO_*			
コンスタント	値	設定場所	設定方法他
MQCMHO_DEFAULT_VALIDATION	0x00000000	MQCRTMH.Options	Overwrite (default)
MQCMHO_NO_VALIDATION	0x00000001	以下同様	以下同様
MQCMHO_VALIDATE	0x00000002		
MQCMHO_NONE	0x00000000		

Ex. 6.11.1 プロパティ名をMQSETMP()で検証させる例

※不正なプロパティ名として"*"を指定して、MQCMHO_VALIDATEでメッセージ・ハンドルを作成 (MQCRTMH()) し、MQSETMP()にチェックをさせる。

```
$ mqpgf -qm TESTQM -q TQ -m "test" -smp "MQTYPE_STRING:::test" MQPMO_VERSION_3 MQCMHO_VALIDATE
```

```
MQSETMP failed : CompCd=02 ReasonCd=2442
```

```
$ mqrc 2442
```

```
$
```

```
2442 0x0000098a MQRC_PROPERTY_NAME_ERROR
```

※MQRC_PROPERTY_NAME_ERROR で MQSETMP() が終了する。

```
$
```

\$ 同様のプロパティ名の指定で、MQCMHO_NO_VALIDATION を指定してメッセージ・ハンドルを作成 (MQCRTMH()) し、MQSETMP()にチェックさせないようにする。

```
$ mqpgf -qm TESTQM -q TQ -m "test" -smp "MQTYPE_STRING:*:test" MQPMO_VERSION_3 MQCMHO_NO_VALIDATION
```

```
$
```

```
[17/01/11 18:38:39] 1: message length: 4 put message : test
```

```
$
```

```
$ mqpgf -qm TESTQM -q TQ MQGMO_PROPERTIES_IN_HANDLE MQGMO_VERSION_4 -br  
message number: 1
```

```
....
```

```
***Message properties***
```

```
    * : 'test'
```

```
MQINQMP failed : CompCd=02 ReasonCd=2471
```

```
....
```

※バリデーション・チェックが行われずメッセージ・プロパティが作成される。

※MQINQMPを呼び出し時、引数NameにMQPROP_INQUIRE_ALLを指定してすべてのメッセージプロパティを照会します。理由コード: 2471 (MQRC_PROPERTY_NOT_AVAILABLE) でMQINQMP() が終了するまで繰り返し表示します。

6.12 MQCBDパラメータ

MQCBT_*

MQCBD.CallbackType に設定されます。

下表のコンスタントが指定可能です。

表 6.12.1 指定可能な MQCBT_*			
コンスタント	値	設定場所	設定方法他
MQCBT_MESSAGE_CONSUMER	0x00000000	MQCBD.CallbackType	Overwrite (default)
MQCBT_EVENT_HANDLER	0x00000001	〃	〃

6.13 MQAIRパラメータ

MQAIR_* (MQAIR*VERSION*)

MQAIR.Version に設定されます。

下表のコンスタントが指定可能です。

表 6.13.1 指定可能な MQAIR*VERSION*

コンスタント	値	設定場所	設定方法他
MQAIR_VERSION_1	1	MQAIR.Version	Overwrite (default)
MQAIR_VERSION_2	2	〃	〃
MQAIR_CURRENT_VERSION	1	〃	〃

MQAIT_*

MQAIR.AuthInfoType に設定されます。

下表のコンスタントが指定可能です。

表 6.13.2 指定可能な MQAIT_*

コンスタント	値	設定場所	設定方法他
MQAIT_CRL_LDAP	1	MQAIR.AuthInfoType	Overwrite (default)
MQAIT_OCSP	2	〃	〃

おわりに

本プログラムについてのバグが見つかった場合、あるいは機能についてのご要望、その為
ご質問等がございましたら、下記までご連絡ください。

Pulsar Integration Inc.
<https://www.pulsarintegration.com>
<https://www.pulsarintegration.jp>
e-mail: support@pulsarintegration.com

本プログラムのHP NonStop版は、下記に著作権があるsha2署名プログラムを使用していま
す。

* AUTHOR: Aaron D. Gifford - <http://www.aarongifford.com/>
*
* Copyright (c) 2000-2001, Aaron D. Gifford
* All rights reserved.
*