IBM i

バージョン **7.4**

プログラミング
CL 概説および 概念

IBM i

バージョン **7.4**

プログラミング
CL 概説および 概念

注記

本書および本書で紹介する製品をご使用になる前に、 671 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM i 7.4 (製品番号 5770-SS1)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

本書にはライセンス内部コードについての参照が含まれている場合があります。ライセンス内部コードは機械コードであり、 IBM 機械コードのご使用条件に基づいて使用権を許諾するものです。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM i
Version 7.4
Programming
CL overview and concepts

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

© Copyright IBM Corporation 1998, 2018.

目次

© Copyright IBM Corp. 1998, 2018

制御言語

システム・プログラマーおよびシステム管理者は制御言語 (CL) を使用して、IBM® i コマンドおよびその他の IBM 提供のコマンドを使用しプログラムを作成することができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

制御言語の概説

制御言語のトピック・コレクションには、IBM i オペレーティング・システムの一部である制御言語 (CL) のすべてのコマンドについての情報が含まれています。また、IBM i オペレーティング・システムで実行されるライセンス・プログラムについての多数の CL コマンドも含まれています。

CL コマンドを使用する前に、この制御言語のトピック・コレクションで説明する CL プログラミングの概念についてよく理解してください。

対象となるコマンドの使用者

CL コマンドは通常、プログラマー、データ処理マネージャー、およびシステム管理者が使用します。IBM i コマンド・メニュー、強力なコマンド・プロンプター・ツール、およびオンライン・コマンド・ヘルプを使用することで、技術者でないユーザーも CL コマンドを使用できます。CL コマンドを使用するには、IBM i オペレーティング・システムに関する一般的知識が必要です。複数の CL コマンドを組み合わせて制御言語 (CL) プログラムを作成したい場合は、プログラミングの予備知識が役に立ちます。

これらの情報の構成方法

CL コマンド検索プログラムでは、プロダクト、コマンド名、記述名、または名前の一部でコマンドを検索することができます。 また、新しいコマンドおよび変更されたコマンドも検索することができます。すべてのコマンドのリストをアルファベット順に表示することもできます。

関連資料:

2 ページの『制御言語の PDF ファイル』
この情報の PDF ファイルを表示または印刷できます。

IBM i 7.4 の新機能

制御言語に関する新規の情報または大幅に変更された情報については、該当するトピック・コレクションを参照してください。

ILE CL における新規組み込み関数 **%PARMS** のサポート

- 238 ページの『%PARMS 組み込み関数』

他の CL 組み込み関数と同様に、%PARMS 組み込み関数は、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。これらの新しい組み込み関数を使用する CL ソースをコンパイルする際に、`V7R3M0` または `V7R2M0` の `TGTRLS` (ターゲット・リリース) 値を指定できます。この場合、CL プログラムを 7.4 システムから保存し、IBM i 7.3 または IBM i 7.2 を実行中のシステム上でこれを復元し、実行することができます。

© Copyright IBM Corp. 1998, 2018

ILE CL における IFS ソース・ファイルのサポート

ILE CL コンパイラーは、IFS からのメイン・ソース・ファイルと INCLUDE ファイルの両方をコンパイルできます。 IFS ファイル名をサポートするために、CRTCLMOD、CRTBNDCL、および INCLUDE の CL コマンドが機能拡張されました。

CL コマンドの定義における IFS ソース・ファイルのサポート

IFS ソース・ファイルから CL コマンドを定義できます。 IFS ファイル名をサポートするために、CRTCMD CL コマンドが機能拡張されました。

新しいコマンドおよび変更されたコマンド

CL コマンド検索プログラムを使用すると、すべての新しいコマンドまたは変更されたコマンドのリストを表示することができます。

新規情報または変更情報の見分け方

技術上の変更が加えられた場所を見分けるのに役立つように、Information Center では以下のイメージを使用しています。

- » イメージにより、新規または変更された情報の開始点を示します。
- « イメージにより、新規または変更された情報の終了点を示します。

PDF ファイルでは、左マージンに新規および変更情報のリビジョン・バー (|) があります。

今回のリリースの新規情報または変更情報に関するその他の情報は、「プログラム資料説明書」を参照してください。

制御言語の PDF ファイル

この情報の PDF ファイルを表示または印刷できます。

制御言語コマンドの概説および概念情報の PDF 版を表示またはダウンロードするには、「CL 概説および概念」を選択します。 CL コマンド説明の PDF はもう提供されていません。ご使用のシステムでのコマンド情報の印刷に関する詳細は、 52 ページの『システムでの CL コマンドの説明の印刷』を参照してください。

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ご使用のブラウザーで PDF リンクを右クリックする。
2. PDF をローカルに保存するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「保存」をクリックする。

Adobe Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe Reader がご使用のシステムにインストールされている必要があります。Adobe Reader は、Adobe の Web サイト (http://get.adobe.com/reader/) から無償でダウンロードすることができます。

関連概念:

1 ページの『制御言語の概説』
制御言語のトピック・コレクションには、IBM i オペレーティング・システムの一部である制御言語 (CL) のすべてのコマンドについての情報が含まれています。また、IBM i オペレーティング・システムで実行されるライセンス・プログラムについての多数の CL コマンドも含まれています。

52 ページの『システムでの CL コマンドの説明の印刷』
CL コマンドのパラメーターおよび値の説明を印刷する際には、以下の手順に従ってください。

CL の概念

制御言語 (CL) は、全コマンドの集合で、ユーザーはこれを使用してシステム機能を要求します。この情報では、コマンドを使用する前に理解しておく必要のある、CL の基本的な概念について説明します。

システム操作の制御

制御言語 (CL)、メニュー・オプション、およびシステム・メッセージが、システムの操作を制御します。

制御言語

制御言語 (CL) は、オペレーティング・システムとの基本インターフェースです。CL は、異なるワークステーションの複数のユーザーが、バッチ・ジョブおよび対話式ジョブであっても、CL プログラムおよびプロシージャーであっても、同時に使用することができます。

単一の制御言語ステートメントはコマンドと呼ばれます。コマンドを入力する方法は次のとおりです。

- ワークステーションから単独で。
- バッチ・ジョブの一部として。
- CL プログラムまたはプロシージャーを作成するためのソース・ステートメントとして。

コマンドは、任意のコマンド行またはコマンド入力画面から単独で入力することができます。

CL の用法を簡素化するために、すべてのコマンドで一貫した構文が使用されています。さらに、オペレーティング・システムでは、すべてのコマンドでプロンプト・サポートを行い、コマンド・パラメーターの大部分にデフォルト値を設定し、また妥当性検査でコマンドが正しく入力されたかどうかを確認してからコマンドの機能を実行するようになっています。したがって CL は、異なるシステムのユーザーが使用する多種多様なシステム機能に対して単一の、柔軟性のあるインターフェースを提供します。

関連概念:

5 ページの『CL コマンド』
制御言語 (CL) コマンドは、システム機能を要求する単一のステートメントです。

メニュー

システムには多くのメニューが用意されており、ユーザーはメニューからオプションを選択するだけで、さまざまなシステム機能を実行することができます。

メニューには次の利点があります。

- ユーザーは CL コマンドやその構文を理解している必要がありません。
- 入力する回数、および誤った入力をする可能性が大幅に減ります。

関連情報:

Application Display Programming

IBM i グローバリゼーション

メッセージ

メッセージとは、あるユーザーからプログラム、プロシージャーまたは他のユーザーに送られる通信文のことです。

システム・メッセージは、状況情報およびエラー状態の両方を報告することができます。ほとんどのデータ処理システムには、処理中に生じたエラーやその他の状態に対処するために、システムとシステム・オペレーターとの間の通信機能が用意されています。 IBM i オペレーティング・システムには、プログラムとシステム・ユーザーとの間、プログラム相互間、プログラム内のプロシージャー相互間、およびシステム・ユーザー相互間の両方向の通信を可能にするメッセージ処理機能があります。使用できるメッセージには次の 2 つのタイプがあります。

- 即時メッセージ。これは、その送信の時点でプログラムまたはシステムのユーザーによって即時に作成されるもので、システム内に永続的に保持されることのないメッセージです。
- 事前定義メッセージ。これは、実際の使用に先だってあらかじめ作成されているメッセージです。この種のメッセージは作成時にメッセージ・ファイルに入れられ、使用される時点でそのファイルから取り出されます。

メッセージは、プログラム相互間、プログラム内のプロシージャー間、およびプログラムとユーザーとの間の通信に使用することができるので、アプリケーションを開発する際には、IBM i のメッセージ処理機能の使用を考慮しなければなりません。アプリケーションの開発に当たっては、メッセージ処理に関する重要事項として、次のような点を考慮に入れる必要があります。

- メッセージは、それを使用するプログラムの外部にあるメッセージ・ファイルに定義することができ、またメッセージ・テキストには、そのメッセージを送る時点の状況に応じて変更できる可変情報を含めることができます。メッセージはプログラムの外部で定義されるので、メッセージを変更してもプログラムを変更する必要はありません。また、そのメッセージを別の言語に翻訳したメッセージを含むファイル (複数も可) を同じプログラムで使用することもできます。
- メッセージは、システムの 1 つのオブジェクトであるメッセージ待ち行列との間でやりとりされます。待ち行列に送られたメッセージは、プログラムまたはワークステーション・ユーザーが明示的に受信を要求するまで、待ち行列に置かれます。
- プログラムは、そのプログラムを要求したユーザーがどのワークステーションにサインオンしていても、そのユーザーにメッセージを送ることができます。言い換えれば、メッセージは必ずしも特定の装置に送る必要はありません。1 つのプログラムを多数のワークステーションから、変更を加えることなく使用できます。

関連タスク:

536 ページの『メッセージ』
メッセージは、ユーザーとプログラム間の通信に使用されます。

関連情報:

IBM i グローバリゼーション

メッセージ記述:

メッセージ記述は、IBM i オペレーティング・システムへのメッセージを定義します。

メッセージ記述には、メッセージのテキスト、置き換え変数に関する情報、およびメッセージの送信の際にメッセージの送信側が指定する可変データを含めることができます。

メッセージ記述はメッセージ・ファイルに保管されます。各記述には、それぞれのファイル内で固有の識別コードが付いていなければなりません。メッセージが送られる時点で、システムはどのメッセージ記述を使用するかを、メッセージ・ファイルおよびメッセージ識別コードによって識別します。

メッセージ待ち行列:

メッセージは、プロシージャー、プログラム、またはシステム・ユーザーに送られると、そのプロシージャー、プログラムまたはユーザーに関連づけられているメッセージ待ち行列に入れられます。プロシージャー、プログラム、またはユーザーは、待ち行列からメッセージ受信することによってそのメッセージを見ることができます。

IBM i オペレーティング・システムは、以下の項目についてメッセージ待ち行列を提供します。

- システムの各ワークステーション用
- システムに登録されている各ユーザー用
- システム・オペレーター用
- システム活動記録ログ用

上記以外にも、特定のアプリケーション要件に合わせて、新しいメッセージ待ち行列を作成することができます。メッセージ待ち行列に送られたメッセージは保持されるので、メッセージの受信側はすぐにメッセージを処理する必要はありません。

CL コマンド

制御言語 (CL) コマンドは、システム機能を要求する単一のステートメントです。

関連概念:

3 ページの『制御言語』

制御言語 (CL) は、オペレーティング・システムとの基本インターフェースです。CL は、異なるワークステーションの複数のユーザーが、バッチ・ジョブおよび対話式ジョブであっても、CL プログラムおよびプロシージャーであっても、同時に使用することができます。

CL コマンド名

コマンド名は、コマンドの実行時に呼び出されるプログラムが実行する機能を識別します。コマンド名は、動詞 (動作) および動作の対象 (対象となるオブジェクト) を識別する名詞または句を組み合わせたものです (コマンド = 動詞 + 対象となるオブジェクト)。

コマンド名を構成する語は略語 (通常 1 から 3 文字) となっているので、コマンドを入力するのに必要な入力数を少なくすることができます。

例えば、ライブラリーは、作成、削除、または表示することができることから、各動詞の略である CRT、DLT、および DSP が、ライブラリーの略である LIB と組み合わされます。したがって、ライブラリーを処理できる 3 つのコマンドは、**CRTLIB**、**DLTLIB**、および **DSPLIB** です。別の例では、CL コマンドの 1 つにメッセージ送信 (Send Message) コマンドがあります。この**メッセージ送信 (SNDMSG)** コマンドは、ユーザーがメッセージ待ち行列にメッセージを送るために使用します。

動詞およびその対象となるオブジェクトを組み合わせて名前とする規則は、以下のとおりです。

- 第 1 の規則として、上記のようにコマンドの記述名を構成する各語からそれぞれ 3 文字をとり、それを、システムが認識できる省略コマンド名として使用します。

- 第 2 の規則として、コマンド・タイトルの最後の 1 語または複数の語から 1 文字ずつ取り、それをコマンド名の終わりに使用します。例えば、**文書ライブラリー・オブジェクト削除 (DLTDLO)** コマンドでは、3 つの文字 DLO が加えられています。
- ただし、いくつかの例外もあり、語を代表する文字がいくつか、コマンド名の中央で使用される場合もあります (通常、3 文字の動詞と目的語の間)。例えば、**CL プログラム作成 (CRTCLPGM)** コマンドでは、文字 CL が使用されています。

移動 (MOV) コマンドのように動詞のみ、または**データ (DATA)** コマンドのようにオブジェクトのみで構成されているコマンド名もあります。IBM i コマンド名が付いたコマンドもいくつかありますが、IBM i システム以外のシステムを使用しているユーザーは、1 つ以上の使い慣れた代替名で呼ぶこともできます。代替名は、別名とも呼ばれます。例えば、CD は **現行ディレクトリー変更 (CHGCURDIR)** コマンドの別名です。

関連概念:

127 ページの『単純オブジェクト名および修飾オブジェクト名』
ライブラリー内の特定のオブジェクトの名前は、単純名としても修飾名としても指定できます。

130 ページの『オブジェクトの命名規則』
制御言語 (CL) コマンドで使用されるすべての IBM i オブジェクトの命名には、以下の規則が適用されます。単純オブジェクト名、修飾名、または総称名が使用できるかどうかについては、各 CL コマンドのパラメーターの要約表に示しています。

関連資料:

89 ページの『OBJTYPE パラメーター』
オブジェクト・タイプ (OBJTYPE) パラメーターは、オブジェクトが指定されたコマンドにより操作できる IBM i オブジェクトのタイプを指定します。

CL コマンドおよびキーワードで使用される省略形:

IBM i オペレーティング・システムおよびそのオペレーティング・システム上で実行されるその他のライセンス・プログラムの一部である大部分の CL コマンド名は、一貫した命名形式に従います。

コマンド名の最初の 3 文字 (コマンド verb とも呼ばれる) は、実行される処理を表しています。コマンド名の残りの文字は、その処理が実行される対象となるオブジェクトを示しています。

関連タスク:

312 ページの『CL コマンドの定義』
CL コマンドを使用することによって、ユーザーは、広範囲にわたる機能をシステムに要求できます。IBM 提供のコマンドを使用して、コマンド・パラメーター用のデフォルト値を変更し、ユーザー独自のコマンドを定義することができます。

***CL* コマンド動詞の省略形:**

ほとんどの CL コマンドで、次の共通コマンド動詞 の 1 つを使用します。

動詞の省略形	意味
ADD	追加
CHG	変更
CRT	作成
DLT	削除
DSP	表示画面
END	終了
RMV	除去

動詞の省略形	意味
STR	開始
WRK	処理

次の表は、コマンド動詞として使用される省略形をすべてリストしています。

動詞の省略形	意味
ADD	追加
ALC	割り振り
ANS	応答
ANZ	分析
APY	適用
ASK	質問
CFG	構成
CHG	変更
CHK	検査
CLO	クローズ
CLR	消去
CMP	比較
CNL	取り消し
CPR	圧縮
CPY	コピー
CRT	作成
CVT	変換
DCP	圧縮解除
DLC	割り振り解除
DLT	削除
DLY	遅延
DMP	ダンプ
DSC	切断
DSP	表示画面
DUP	複製
EDT	編集
EJT	排出
EML	エミュレート
END	終了
EXP	エクスポート
EXT	抽出
FIL	ファイル
FMT	形式
FND	検出
GEN	生成
GRT	権限認可
HLD	保持
IMP	インポート
INS	インストール
INZ	初期設定
LNK	リンク
LOD	ロード
MGR	移行
MON	モニター
MOV	移動
MRG	組み合わせ
OPN	オープン
ORD	順序

動詞の省略形	意味
OVR	一時変更
PKG	パッケージ
POS	配置
PRT	印刷
PWR	電源
QRY	照会
RCL	再利用
RCV	受信
RGZ	再編成
RLS	解放
RMV	除去
RNM	名前変更
RPL	置換
RQS	要求
RRT	再経路指定
RSM	再開
RST	復元
RTV	検索
RUN	実行
RVK	取り消し
SAV	保管
SBM	投入
SET	設定
SLT	選択
SND	送信
STR	開始
TFR	転送
TRC	トレース
UPD	更新
VFY	検証
VRY	変更
WRK	処理

CL コマンド省略形*:*

この表は、CL コマンド名で使用されるすべての省略形 (コマンド動詞の省略形も含む) をリストしています。

コマンドの省略形	意味
A (接尾部)	属性
ABN	異常
ACC	アクセス・コード
ACCGRP	アクセス・グループ
ACG	会計
ACN	処理
ACP	受諾
ACNE	処理項目
ACT	活動状態、活動、活動化
ADD	追加
ADM	管理、管理の
ADP	借用
ADPI	アダプター情報
ADPP	アダプター・プロファイル
ADPT	アダプター

コマンドの省略形	意味
ADR	アドレス
AFP	拡張機能印刷
AGR	契約
AGT	エージェント
AJE	自動開始ジョブ項目
ALC	割り振り
ALR	警報
ALRD	警報記述
ALRTBL	警報テーブル
ALS	別名
ANS	応答
ANZ	分析
AP	アクセス・パス
APAR	プログラム診断依頼書
APF	拡張印刷機能
APP	アプリケーション
APPC	拡張プログラム間通信
APPN	拡張対等通信ネットワーキング機能
APY	適用
ARA	域
ARC	アーカイブ
ASC	非同期
ASK	質問
ASN	アソシエーション
ASP	補助記憶域プール
AST	援助
ATM	非同期転送モード
ATN	アテンション
ATR	属性
AUD	監査
AUT	権限
AUTE	認証項目
AUTL	権限リスト
BACK	バック
BAL	平衡、平衡化
BAS	BASIC 言語
BCD	バーコード
BCH	バッチ
BCK	バックアップ
BCKUP	バックアップ
BGU	ビジネス・グラフィックス・ユーティリティー
BKP	ブレークポイント
BKU	バックアップ
BND	バインド、バインド済み
BP	ブート・プロトコル
BRM	BRMS (backup recovery and media services)
BSC	2 進データ同期
BSCF	BSC ファイル
BUF	バッファー
C	C 言語
CAD	クラスター管理ドメイン
CAL	カレンダー
CALL	呼び出し
CAP	取り込み

コマンドの省略形	意味
CBL	COBOL 言語
CCF	クレデンシャル・キャッシュ・ファイル
CCS	変更コントロール・サーバー
CDE	コード、コード化
CDS	コード化データ保管
CFG	構成、構成化
CFGL	構成リスト
CFGLE	構成リスト項目
CGY	カテゴリー
CHG	変更
CHK	検査
CHT	図表
CICS®	顧客情報管理システム
CKM	暗号鍵管理
CL	制御言語
CLD	C ロケール記述
CLG	カタログ
CLNUP	終結処置
CLO	クローズ
CLR	消去
CLS	クラス
CLT	クライアント
CLU	クラスター
CMD	コマンド
CMN	通信
CMNE	通信項目
CMNF	通信ファイル
CMP	比較
CMT	コミットメント
CNL	取り消し
CNN	接続
CNNL	接続リスト
CNNLE	接続リスト項目
CNR	コンテナー
CNT	接続
CNV	会話
CODE	連携開発環境
COL	コレクション
COM	コミュニティー
COSD	サービス・クラス記述
CP	検査保留
CPH	暗号
CPIC	通信用の共通プログラミング・インターフェース
CPP	C++ 言語
CPR	圧縮
CPT	構成要素
CPY	コピー
CPYSCN	コピー画面
CRG	クラスター資源グループ
CRL	クローラー
CRP	暗号
CRQ	変更要求
CRSDMN	クロスドメイン
CRT	作成

コマンドの省略形	意味
CSI	通信サイド情報
CSL	コンソール
CST	制約、カスタマイズ
CTG	カートリッジ
CTL	制御
CTLD	制御装置記述
CUR	現行
CVG	適応範囲
CVN	変換
CVT	変換
D (接尾部)	説明
DAT	日付
DB	データベース
DBF	データベース・ファイル
DBG	デバッグ
DCL	宣言
DCP	圧縮解除
DCT	ディクショナリー
DDI	分散データ・インターフェース
DDM	分散データ管理機能
DDMF	分散データ管理機能ファイル
DEP	従属
DEV	装置
DEVD	装置記述
DFN	定義
DFR	据え置き
DFT	デフォルト値
DFU	データ・ファイル・ユーティリティー
DHCP	動的ホスト構成プロトコル
DIG	ドメイン情報グローパー
DIR	ディレクトリー
DIRE	ディレクトリー項目
DIRSHD	ディレクトリーのシャドーイング
DKT	ディスケット
DKTF	ディスケット・ファイル
DL	データ・リンク、データ・ライブラリー
DLC	割り振り解除
DLF	データ・リンク・ファイル
DLFM	データ・リンク・ファイル・マネージャー
DLO	文書ライブラリー・オブジェクト
DLT	削除
DLY	遅延
DMN	ドメイン
DMP	ダンプ
DNS	ドメイン・ネーム・サービス、ドメイン・ネーム・システム
DO	実行
DOC	文書
DOM	Lotus®Domino
DPCQ	DSNX/PC 待ち行列
DPR	DataPropagator Relational
DSC	切断
DSK	ディスク
DSP	表示画面
DSPF	表示装置ファイル

コマンドの省略形	意味
DST	配布
DSTL	配布リスト
DSTLE	配布リスト項目
DSTQ	配布待ち行列
DSTSRV	配布サービス
DTA	データ
DTAARA	データ域
DTAQ	データ待ち行列
DUP	複製
DW	ディスク・ウォッチャー
DWN	ダウン
E (接尾部)	項目
EDT	編集
EDTD	編集記述
EDU	研修
EJT	排出
EML	エミュレート、エミュレーション
ENC	暗号化
END	終了
ENR	登録
ENV	環境
ENVVAR	環境変数
EPM	拡張プログラム・モデル
ERR	エラー
ETH	イーサネット
EWC	拡張無線制御装置
EWL	拡張無線回線
EXIT	出口
EXP	有効期限、満了、エクスポート
EXT	抽出
F (接尾部)	ファイル
FAX	ファクシミリ
FCN	機能
FCT	用紙制御テーブル
FCTE	用紙制御テーブル項目
FD	ファイル記述
FFD	ファイル・フィールド記述
FIL	ファイル
FILL	充てん
FLR	フォルダー
FLT	フィルター
FMT	形式
FMW	ファームウェア
FNC	金融機関
FND	検出
FNT	フォント
FNTRSC	フォント資源
FNTTBL	フォント・テーブル
FORM	用紙
FORMD	用紙記述
FORMDF	書式定義
FR	フレーム・リレー
FRM	元から
FRW	ファイアウォール

コマンドの省略形	意味
FTP	ファイル転送プロトコル
FTR	フィルター
GDF	グラフィック・データ形式
GEN	生成
GO	進む
GPH	グラフィックス
GPHPKG	グラフ・パッケージ
GRP	グループ
GRT	権限認可
GSS	グラフィック記号セット
HDB	ホスト・データベース
HDW	ハードウェア
HDWRSC	ハードウェア資源
HLD	保持、保留
HLL	高水準言語
HLP	ヘルプ
HLR	ホルダー
HOST	ホスト
HST	活動記録、実績
HT	ホスト・テーブル
HTE	ホスト・テーブル項目
HTTP	HTTP
I (接尾部)	情報、項目、ILE
ICF	システム間通信機能
ICFF	ICF ファイル
IDD	対話式データ定義ユーティリティー
IDLC	ISDN データ・リンク制御
IDX	索引
IDXE	索引項目
IFC	インターフェース
IGN	無視
IMG	イメージ
IMP	インポート
INF	情報
INP	入力
INS	インストール
INST	インスタンス
INT	内部、統合
INTR	システム間
INZ	初期設定
IPL	初期プログラム・ロード
IPS	SNA より上のインターネット・プロトコル
ISDB	対話式ソース・デバッガー
ISDN	統合サービス・デジタル網
ITF	端末対話機能
ITG	保全性
ITM	項目
IWS	インテリジェント・ワークステーション
JOB	ジョブ
JOBD	ジョブ記述
JOBE	ジョブ項目
JOBQ	ジョブ待ち行列
JOBQE	ジョブ待ち行列項目
JRN	ジャーナル

コマンドの省略形	意味
JRNRCV	ジャーナル・レシーバー
JS	ジョブ・スケジューラー
JVA	Java™
JVM	Java 仮想マシン
JW	ジョブ・ウォッチャー
KBD	キーボード
KEY	キー
KRB	Kerberos
KSF	鍵ストア・ファイル
KTE	鍵タブ項目
KVV	鍵検証値
L (接尾部)	リスト
LAN	ローカル・エリア・ネットワーク
LANG	言語
LBL	ラベル
LCK	ロック
LCL	内部
LCLA	ローカル属性
LDIF	LDAP データ交換形式
LF	ロジック・ファイル
LFM	ロジック・ファイル・メンバー
LIB	ライブラリー
LIBL	ライブラリー・リスト
LIBM	ライブラリー・メンバー
LIC	ライセンス
LIN	回線
LIND	回線記述
LNK	リンク
LNX	Linux
LOC	ロケーション
LOCALE	ロケール
LOD	ロード
LOF	論理光ディスク・ファイル
LOG	ログ
LOGE	ログ項目
LPD	ライン・プリンター・デーモン
LPDA	リンク問題判別援助機能
LPR	ライン・プリンター・リクエスター
LWS	ローカル・ワークステーション
M (接尾部)	メンバー
MAC	メッセージ認証コード
MAIL	メール
MAP	マップ
MAX	最大
MBR	メンバー
MDL	モデル
MED	媒体
MEDDFN	媒体定義
MEDI	媒体情報
MEM	メモリー
MFS	マウントされたファイル・システム
MGD	管理されている
MGR	移行、管理機能
MGTCOL	管理収集

コマンドの省略形	意味
MLB	媒体ライブラリー
MLM	媒体ライブラリー媒体
MNT	分、保守
MNU	メニュー
MOD	モード、モジュール
MODD	モード記述
MON	モニター
MOV	移動
MRE	モニター対象リソース項目
MRG	組み合わせ
MSF	メール・サーバー・フレームワーク
MSG	メッセージ
MSGD	メッセージ記述
MSGF	メッセージ・ファイル
MSGQ	メッセージ待ち行列
MST	マスター
M36	AS/400 アドバンスト 36 マシン
M36CFG	AS/400 アドバンスト 36 マシン構成
NAM	名前
NCK	ニックネーム
NET	ネットワーク
NETF	ネットワーク・ファイル
NFS	ネットワーク・ファイル・システム
NODGRP	ノード・グループ
NODL	ノード・リスト
NTB	NetBIOS
NTS	LotusNotes
NWI	ネットワーク・インターフェース
NWS	ネットワーク・サーバー
NWSAPP	ネットワーク・サーバー・アプリケーション
NWSCFG	ネットワーク・サーバー構成
NWSD	ネットワーク・サーバー記述
NWSH	ネットワーク・サーバー・ホスト・アダプター
OBJ	オブジェクト
OCL	操作制御言語
OF	光ディスク・ファイル
OFC	オフィス
オフ	オフ
OMC	オブジェクト管理サイクル
OND	OnDemand
OPC	OptiConnect
OPN	オープン
OPT	光ディスク、オプション
ORD	順序
OSPF	最初に一番短いパスをオープン
OUT	アウト、発信、出力
OUTQ	出力待ち行列
OUTQD	出力待ち行列記述
OVL	オーバーレイ
OVLU	オーバーレイ・ユーティリティー
OVR	一時変更
OWN	所有者
PAG	ページ、ページ編集
PAGDFN	ページ定義

コマンドの省略形	意味
PAGS	ページ・セグメント
PAGSEG	ページ・セグメント
PARM	パラメーター
PART	パーツ
PASTHR	パススルー
PC	パーソナル・コンピューター
PCD	パーソナル・コンピューター文書
PCL	プロトコル
PCO	パーソナル・コンピューター・オーガナイザー
PCY	ポリシー
PDF	Portable Document Format
PDG	印刷記述子グループ
PDM	プログラム開発管理機能
PEX	パフォーマンス・エクスプローラー
PF	物理ファイル
PFD	印刷出力形式定義
PFM	物理ファイル・メンバー
PFR	パフォーマンス
PFRG	パフォーマンス・グラフィックス
PFRT	パフォーマンス・ツール
PFU	印刷形式ユーティリティー
PFX	接頭部
PGM	プログラム
PGP	1 次グループ
PGR	ページャー
PHS	フェーズ
PIN	個人識別番号
PJ	事前開始ジョブ
PJE	事前開始ジョブ項目
PKG	パッケージ
PLI	PL/I (プログラミング言語 /I)
PMN	許可
PMT	プロンプト
PNLGRP	パネル・グループ
POF	物理光ディスク・ファイル
POL	プール
POP	POP
PORT	ポート
POS	配置
PPP	Point-to-Point Protocol
PRB	問題
PRC	プロシージャー、処理
PRD	プロダクト
PRF	プロファイル
PRFL	プロファイル・リスト
PRI	1 次
PRJ	プロジェクト
PRM	プロモート
PRP	準備
PRS	パーソナル
PRT	印刷
PRTF	印刷装置ファイル
PRTQ	印刷待ち行列
PRX	プロキシー

コマンドの省略形	意味
PSFCFG	印刷サービス機能構成
PTC	可搬トランザクション・コンピューター
PTF	プログラム一時修正
PTP	2 地点間
PTR	ポインター
PVD	提供者
PVT	私用
PWD	パスワード
PWR	電源
PYM	取り引き
QM	照会管理
QRY	照会
QRYF	照会ファイル
QSH	Qshell インタープリター
QST	質問
RBD	再作成
RCD	レコード
RCL	再利用
RCV	受信
RCY	リカバリー
RDAR	報告書またはデータのアーカイブおよび検索
RDB	リレーショナル・データベース
RDR	読み取りプログラム
REF	参照
REG	登録
REX	REXX (再構造化拡張実行プログラム言語)
RGP	ランキング・グループ
RGPE	ランキング・グループ項目
RGZ	再編成
RIP	ルーティング情報プロトコル
RJE	リモート・ジョブ入力
RLS	解放
RLU	報告書設計ユーティリティー
RMC	報告書管理サイクル
RMT	リモート
RMV	除去
RNDC	リモート名デーモン制御
RNG	範囲
RNM	名前変更
ROLL	ロール
RPC	リモート・プロシージャー呼び出し
RPDS	VM/MVS ブリッジ (以前の遠隔スプーリング通信サブシステム (RSCS) / PROFS 配布サービス)
RPG	報告書作成プログラム
RPL	置換
RPT	報告書
RPY	応答
RPYL	応答リスト
RQS	要求
RRT	再経路指定
RSC	資源
RSE	リモート・システム・エクスプローラー
RSI	リモート・システム情報
RSM	再開

コマンドの省略形	意味
RST	復元
RTD	ルート・デーモン (TCP/IP)
RTE	経路項目
RTGE	経路指定項目
RTL	小売業
RTLF	小売業ファイル
RTN	戻り
RTV	検索
RUN	実行
RVK	取り消し
RWS	リモート・ワークステーション
RXC	REXEC (リモート実行)
SAV	保管
SAVF	保管ファイル
SAVRST	保管および復元
SBM	投入、投入された
SBS	サブシステム
SBSD	サブシステム記述
SCD	スケジュール
SCDE	スケジュール項目
SCHIDX	検索索引
SCHIDXE	検索索引項目
SCN	画面
SDA	画面設計機能
SDLC	同期データ・リンク制御
SEC	セキュリティー
SET	設定
SEU	原始ステートメント入力ユーティリティー
SFW	ソフトウェア
SHD	シャドー、シャドー機能
SHRPOOL	共有プール
SIGN	符号、署名
SIT	状態
SLT	選択
SLTE	選択項目
SMG	システム・マネージャー
SMW	システム・マネージャー・ワークステーション
SMTP	simple mail transfer protocol
SNA	システム・ネットワーク体系
SND	送信
SNI	SNA over IPX
SNMP	simple network management protocol
SNPT	SNA パススルー
SNUF	SNA アップライン機能
SOC	制御範囲
SPADCT	スペル援助ディクショナリー
SPL	スプーリング
SPLF	スプール・ファイル
SPT	サポート
SPTN	サポート・ネットワーク
SQL	構造化照会言語
SRC	ソース
SRCF	ソース・ファイル
SRV	サービス

コマンドの省略形	意味
SRVPGM	サービス・プログラム
SSN	セッション
SSND	セッション記述
SST	システム・サービス・ツール
STC	統計
STG	記憶域
STGL	記憶域リンク
STM	ストリーム、ステートメント
STR	開始
STS	状況
SUBR	サブルーチン
SVR	サーバー
SWA	活動状態での保管
SWL	停止語リスト
SYNC	同期
SYS	システム
SYSCTL	システム制御
SYSDIR	システム・ディレクトリー
SYSVAL	システム値
S34	システム/34
S36	System/36
S38	System/38
TAP	テープ
TAPF	テープ・ファイル
TBL	テーブル
TBLE	テーブル項目
TCP	TCP/IP (伝送制御プロトコル / インターネット・プロトコル)
TDLC	平衡型データ・リンク制御
TELN	TELNET
TFR	転送
TFTP	トリビアル・ファイル転送プロトコル
THD	スレッド
THLD	しきい値
TIE	技術情報交換
TIEF	技術情報交換ファイル
TIMZON	時間帯
TKT	チケット
TNS	トランザクション
TO	先、まで
TOS	サービスのタイプ
TPL	テンプレート
TRC	トレース
TRG	トリガー
TRN	トークンリング・ネットワーク
TRP	トラップ
TXT	テキスト
TYPE	タイプ
T1	トランスポート・クラス 1
UDFS	ユーザー定義ファイル・システム
UPD	更新
UPG	アップグレード
USG	使用法
USR	ユーザー
USRIDX	ユーザー索引

コマンドの省略形	意味
USRPRF	ユーザー・プロファイル
USRPRTI	ユーザー印刷情報
USRQ	ユーザー待ち行列
USRSPC	ユーザー・スペース
VAL	値
VAR	変数
VFY	検証
VLDL	妥当性検査リスト
VOL	ボリューム
VRY	変更
VT	VT100 または VT220
VWS	仮想ワークステーション
WAIT	待機
WCH	監視
WLS	ワイヤレス
WNT	Windows NT
WP	ワード・プロセッシング
WRK	処理
WSE	ワークステーション項目
WSG	ワークステーション・ゲートウェイ
WSO	ワークステーション・オブジェクト
WTR	書き出しプログラム
XREF	相互参照
X25	X.25
ZNE	ゾーン

CL コマンド・キーワードの省略形*:*

それぞれのコマンド・パラメーターには、そのパラメーターと関連付けられているキーワード名があります。そのキーワード名の最大文字数は 10 文字です。

キーワード名はコマンド名とは違い、コマンド動詞で始める必要はありません。可能であれば、1 つの単語または省略形を使用できます。 CL コマンドの共通キーワード名には、OBJ (オブジェクト)、LIB (ライブラリー)、TYPE、OPTION、TEXT、および AUT (権限) などがあります。

複数の単語または省略形を使用してパラメーターを記述するキーワード名を作成してください。キーワード名は、標準のコマンド省略形と省略形ではない短い単語を組み合わせて作成します。例えば OBJTYPE は、省略形 'OBJ' と短い単語 'TYPE' を組み合わせた共通キーワードです。

キーワード名の基本的な 2 つの目的は、分かりやすいことと、同じ機能を提供するコマンドの間に一貫性を持たせることです。簡単な単語と標準の省略形を使用すれば、キーワード名は分かりやすくなります。

次の表は、CL コマンド・パラメーターのキーワード名で使用される省略形をリストしたものです。

キーワードの省略形	意味
A (接尾部)	属性、活動、アドレス
ABS	抽象、絶対
ABN	異常
AC	自動呼び出し
ACC	アクセス、アクセス・コード
ACCMTH	アクセス方式
ACG	会計
ACK	確認

キーワードの省略形	意味
ACN	処理
ACP	受け入れ
ACQ	取得
ACSE	アソシエーション制御サービス要素
ACT	活動状態、活動化、活動、処理
ACTSNBU	交換網バックアップの活動化
ADDR (または ADR)	アドレス
ADJ	隣接、調整
ADL	追加
ADM	管理、アプリケーション開発管理
ADMD	管理ドメイン
ADMDMN	管理ドメイン
ADP	借用、適応
ADPT	アダプター
ADR (または ADDR)	アドレス
ADRLST	アドレス・リスト
ADV	拡張
AFN	類縁性
AGT	エージェント
AIX®	AIX オペレーティング・システム
AJE	自動開始ジョブ項目
AFN	類縁性
AFP	拡張機能印刷
ALC	割り振り
ALM	警報
ALR	警報
ALRD	警報記述
ALS	別名
ALW	許可
ALWFWD	転送許可
ALWPRX	プロキシー許可
ANET	隣接ネットワーク・エンティティー名称
ANS	応答
ANZ	分析
AP	アクセス・パス
APAR	プログラム診断依頼書
APF	拡張印刷機能
APPP	アプリケーション・プロセス
APW	拡張印刷書き出しプログラム
APP	アプリケーション
APPC	拡張プログラム間通信
APPN	拡張対等通信ネットワーキング機能
APY	適用
ARA	域
ARP	アドレス解決プロトコル
ASC	非同期通信
ASCII	ASCII コード
ASMT	割り当て
ASN	割り当て、アソシエーション
ASP	補助記憶域プール
AST	援助
ASYNC	非同期
ATD	在席
ATN (または ATTN)	アテンション

キーワードの省略形	意味
ATR (または ATTR)	属性
ATTACH	付加
ATTN (または ATN)	アテンション・キー
ATTR (または ATR)	属性
AUD	監査
AUT	権限、許可されている、許可、認証
AUTH	認証
AUTL	権限リスト
AUTO	自動
AUX	補助
AVG	平均
AVL	使用可能な
BAL	平衡
BAS	BASIC 言語、基数
BCD	バーコード、ブロードキャスト・データ
BCH	バッチ
BCKLT	バックライト
BCKUP (または BKU)	バックアップ
BDY	境界
BEX	分岐拡張
BGU	ビジネス・グラフィックス・ユーティリティー
BIN	2 進
BIO	ブロック入出力
BITS	データ・ビット
BKP	ブレークポイント
BKT	ブラケット、バックアウト
BKU (または BCKUP)	バックアップ
BLDG	組み立て
BLK	ブロック
BLN	明滅カーソル
BND	バインド、バインド済み
BNR	バナー
BOT	下部
BRK	中断
BSC	2 進データ同期通信
BSCEL	2 進データ同期通信等価リンク
BSP	バックスペース
BUF	バッファー
C	C 言語
CAB	キャビネット
CAL	カレンダー
CAP	容量、取り込み
CB	コールバック
CBL	COBOL 言語
CCD	呼び出し制御データ
CCSID	コード化文字セット ID
CCT	回路
CCF	クレデンシャル・キャッシュ・ファイル
CDE	コード
CDR	発着信詳細記録
CFG	構成
CFGL	構成リスト
CFGLE	構成リスト項目
CFM	確認

キーワードの省略形	意味
CGU	文字作成ユーティリティー
CHAR (または CHR)	文字
CHAP	チャレンジ・ハンドシェーク認証プロトコル
CHG	変更
CHK	検査
CHKSUM	検査合計
CHKVOL	検査ボリューム ID
CHL	チャネル
CHR (または CHAR)	文字
CHRSTR	文字ストリング
CHT	図表
CGY	カテゴリー
CIM	共通情報モデル
CKR	チェッカー
CKS	検査合計
CL	制御言語
CLG	カタログ
CLN	終結処置
CLNS	コネクションレス型ネットワーク・サービス
CLNUP (または CLN)	終結処置
CLO	クローズ
CLR	消去
CLS	クラス
CLSF	クラス・ファイル
CLT	クライアント
CLU	クラスター
CMD	コマンド
CMN	通信
CMNE	通信項目
CMP	比較
CMT	コミット、コミットメント、注記
CNG	輻輳 (ふくそう)
CNL	取り消し
CNLMT (または CNNLMT)	接続制限
CNN	接続
CNNL	接続リスト
CNNLMT	接続制限
CNR	コンテナー
CNS	定数
CNT	接続
CNTL (または CTL)	制御
CNTRY	国
CNV	会話
COD	コード
CODPAG	コード・ページ
CODE	コード、連携開発環境
COL	列、収集、コレクション
COM	共通、コミュニティー
COMPACT	短縮する、短縮
CON	機密
CONCAT	連結
COND	条件
CONS	接続モード・ネットワーク・サービス
CONTIG	連続

キーワードの省略形	意味
CONT	続行
COS	サービス・クラス
COSD	サービス・クラス記述
COSTBYTE	バイトごとのコスト
COSTCNN	接続ごとのコスト
COVER	カバー・レター
CP	制御点
CPB	プロファイル分岐の変更、互換性
CPI	1 インチ当たりの文字数
CPL	完了
CPP	C++ 言語
CPR	圧縮された、圧縮
CPS	呼び出し進行中シグナル
CPT	構成要素
CPU	中央演算処理装置
CPY	コピー
CPYRGT	著作権
CRC	巡回冗長検査
CRDN	認証
CRG	課金、クラスター資源グループ
CRL	相関
CRQ	変更要求
CRQD	変更要求記述
CRSDMNK	クロスドメイン・キー
CRT	作成
CSI	通信サイド情報
CSL	コンソール
CSR	カーソル
CST	制約、コスト
CTG	カートリッジ
CTL	制御装置、制御
CTLD	制御装置記述
CTLP	制御ポート
CTN	競合
CTS	送信可
CTX	コンテキスト
CUR	現行
CURPWD	現行パスワード
CVG	適応範囲
CVN	変換
CVR	カバー
CVT	変換
CXT (または CTX)	コンテキスト
CYC	サイクル
D (接尾部)	説明
DAP	ディレクトリー・アクセス・プロトコル
DAT	日付
DB	データベース
DBCS	2 バイト文字セット
DBF	データベース・ファイル
DBG	デバッグ
DBR	データベースの関連
DCE	データ通信装置、分散コンピューティング環境
DCL	宣言

キーワードの省略形	意味
DCP	圧縮解除
DCT	ディクショナリー
DDI	分散データ・インターフェース
DDM	分散データ管理機能
DDMF	分散データ管理機能ファイル
DDS	データ記述仕様
DEC	10 進
DEGREE	並列処理の度合い
DEL	転送
DEP	従属
DEPT	部門
DES	Data Encryption Standard
DEST	宛先
DEV	装置
DEVD	装置記述
DFN	定義、定義済み
DFR	据え置き
DFT	デフォルト値
DFU	データ・ファイル・ユーティリティー
DIAG	ダイアログ
DIF	差
DIFF	差異化
DIR	ディレクトリー
DKT	ディスケット
DLC	割り振り解除
DLO	文書ライブラリー・オブジェクト
DLT	削除
DLVRY	転送
DLY	遅延
DMD	デマンド
DMN	ドメイン
DMP	ダンプ
DN	識別名
DOC	文書
DPR	DataPropagator Relational
DRAWER	ドロワー
DRT	直接
DRV	ドライブ
DSA	ディレクトリー・システム・エージェント
DSAP	宛先サービス・アクセス・ポイント
DSB	使用不可
DSC	切断
DSK	ディスク
DSP	表示画面
DST	夏時間、専用保守ツール、配布
DTA	データ
DTE	データ端末装置
DTL	詳細
DUP	複製
DVL	開発
DWN	ダウン
DYN	動的
E (接尾部)	項目
EBCDIC	拡張 2 進化 10 進コード

キーワードの省略形	意味
ECN	明示的な輻輳 (ふくそう) 通知
EDT	編集
EDU	研修
EIM	エンタープライズ識別マッピング
EJT	排出
ELAN	イーサネット LAN
ELEM	要素
ELY	早期
EML	エミュレート、エミュレーション
ENB	使用可能
ENC	コード化
ENR	登録
ENT	入力、項目
ENV	環境
EOF	ファイル終わり
EOR	レコード終わり
EOV	ボリュームの終わり
ERR	エラー
ESC	エスケープ
EST	確立、確立されている
ETH	イーサネット
EVT	イベント
EXC	除外
EXCH	交換
EXD	拡張
EXEC	実行
EXIST	存在
EXN	拡張
EXP	有効期限、満了
EXPR	式
EXPTIME	期限切れ
EXT	抽出、拡張
F (接尾部)	ファイル
FAIL	障害、失敗
Fnn	機能キー 'nn'
FA	ファイル属性
FAX	ファクシミリ
FCL	機能
FCN	機能、関数
FCT	用紙制御テーブル
FCTE	用紙制御テーブル項目
FEA	フロントエンド・アプリケーション
FEA	フロントエンド・アプリケーション
FEAT	フィーチャー
FID	ファイル ID
FIL	ファイル
FLD	フィールド
FLG	フラグ
FLIB	ファイル・ライブラリー
FLR	フォルダー
FLW	フロー
FLV	フェイルオーバー
FMA	フォント管理補助
FMT	形式

キーワードの省略形	意味
FNC	金融機関
FNT	フォント
FORMDF	書式定義
FP	フォーカル・ポイント
FRAC	小数部
FRC	強制
FRI	金曜日
FRM	元から、フレーム
FRQ	頻度
FSC	財政上の
FSN	ファイル順序番号
FST	1 番目
FTAM	ファイル転送アクセスおよび管理
FTP	ファイル転送プロトコル (TCP/IP)
FTR	フィルター
GC	不要情報コレクション
GCH	不要情報コレクション・ヒープ
GDF	グラフィックス・データ・ファイル
GDL	ガイドライン
GEN	生成、世代
GID	グループ ID 番号
GIV	提供
GLB	グローバル
GNL	汎用
GOVR	管理プログラム
GPH	グラフ
GRP	グループ
GRT	権限認可
GSS	グラフィック記号セット
GVUP	放棄
HA	高可用性
HCP	ホスト・コマンド・プロセッサー
HDL (または HNDL)	ハンドル
HDR	ヘッダー
HDW	ハードウェア
HEX	16 進数
HFS	階層ファイル・システム
HLD	保持、保留
HLL	高水準言語
HLP	ヘルプ
HLR	ホルダー
HNDL (または HDL)	ハンドル
HPCP	コード・ページをホストから印刷装置へ
HPFCS	フォント文字セットをホストから印刷装置へ
HPR	ハイパフォーマンス経路指定
HRZ	水平
HST	活動記録、実績
HTML	ハイパーテキスト・マークアップ言語
HTTP	HTTP (TCP/IP)
I (接尾部)	情報、ILE
ICF	システム間通信機能
ICV	初期チェーン値
ID	ID、識別
IDD	対話式データ定義

キーワードの省略形	意味
IDL	アイドル
IDLC	統合データ・リンク制御
IDP	交換文書プロファイル
IDX	索引
IE	情報の要素
IFC	インターフェース
IGC	漢字 (2 バイト文字セット)
IGN	無視
IFC	インターフェース
ILE	統合言語環境
IMG	イメージ
IMMED	即時
IN	入力
INAC (または INACT)	非活動
INACT (または INAC)	非活動
INC	組み込み
INCR	増分
IND	間接
INF	情報
INFSKR	Infoseeker
INH	禁止
INIT	開始
INL	初期
INM	中間
INP	入力
INPACING	インバウンド・ペーシング
INQ	照会
INS	インストール
INST	インスタンス
INT	対話式、整数、内部
INTNET	インターネット
INTNETA	インターネット・アドレス
INTR	システム間
INV	出席予定者、インベントリー、起動
INZ	初期設定
IP	インターネット・プロトコル
IP6	インターネット・プロトコル・バージョン 6、IPv6
IPDS	高機能プリンター・データ・ストリーム
IPI	IP over IPX
IPL	初期プログラム・ロード
IPX	Internet Packet Exchange
ISDN	統合サービス・デジタル網
ISP	インターネット・サービス・プロバイダー
IT	中間テキスト、内部テキスト
ITF	端末対話機能
ITM	項目
ITV	間隔
IW2	IPX WAN バージョン 2 プロトコル
J (接尾部)	ジョブ
JDFT	結合省略時
JE (接尾部)	ジョブ項目
JFLD	結合フィールド
JORDER	結合ファイル順序
JRN	ジャーナル

キーワードの省略形	意味
JRNRCV	ジャーナル・レシーバー
JVA	Java
KBD	キーボード
KNW	認識
KPF	漢字印刷装置機能
KWD	キーワード
L (接尾部)	リスト
LADN	ライブラリー割り当て文書名
LAN	ローカル・エリア・ネットワーク
LANG (または LNG)	言語
LBL	ラベル
LCL	内部
LCLE	ローカル・ロケーション項目
LCK	ロック
LDTIME	リード・タイム
LE (接尾部)	リスト項目
LEC	LAN エミュレーション・クライアント
LECS	LAN エミュレーション構成サーバー
LEN	長さ
LES	LAN エミュレーション・サーバー
LF	ロジック・ファイル
LFM	ロジック・ファイル・メンバー
LFT	左
LGL	ロジック
LIB	ライブラリー
LIBL	ライブラリー・リスト
LIC	ライセンス、ライセンス内部コード
LIFTM	存続時間
LIN	回線、回線記述
LMI	ローカル管理インターフェース
LMT	制限
LNG (または LANG)	言語
LNK	リンク
LNR	リスナー
LOC	ロケーション
LOD	ロード
LPI	1 インチ当たりの行数
LRC	水平冗長検査
LRG	ラージ、大規模
LRSP	ローカル応答
LSE	リース
LST	リスト、最後の
LTR	文字
LUW	論理作業単位
LVL	レベル
LWS	ローカル・ワークステーション
LZYWRT	遅延書き込み
M (接尾部)	メンバー、メッセージ
MAC	マクロ、メディア・アクセス制御
MAINT	保守
MAJ	メジャー
MAP	マップ、製造自動化プロトコル
MAX	最大
MBR	メンバー

キーワードの省略形	意味
MBRS	メンバー
MCA	メッセージ・チャネル・エージェント
MCH	マシン
MDL	モデル
MDM	モデム
MDTA	メッセージ・データ
MED	媒体
MEDI	媒体情報
METAFILE	メタテーブル・ファイル
MFR	生産者
MFS	マウントされたファイル・システム
MGR	管理機能
MGRR	管理機能登録
MGT	管理
MID	中間
MIN	最小、最小化
MLB	媒体ライブラリー装置
MLT	乗数、複数
MM	マルチメディア
MNG	管理
MNT	保守、マウント、マウント済み
MNU	メニュー
MOD	モード、モジュール
MODD	モード記述
MODLVL	モディフィケーション・レベル
MON	モニター、月曜日
MOV	移動
MQM	メッセージ待ち行列管理機能
MRG	組み合わせ
MRK	マーク
MRT	要求元端末
MSF	メール・サーバー・フレームワーク
MSG	メッセージ
MSGS	メッセージ
MSGQ	メッセージ待ち行列
MSR	測定
MSS	管理されているシステム・サービス
MST	マスター
MTD	マウント済み
MTG	合致
MTU	最大伝送単位
MTH	方式
MULT (または MLT)	複数
M36	AS/400 アドバンスト 36 マシン
M36CFG	AS/400 アドバンスト 36 マシン構成
N (接尾部)	名前、ネットワーク
NAM	名前
NBC	ブロードキャストでない
NBR	番号
NCK	ニックネーム
NDE	ノード
NDM	正規切断モード
NDS	NetWare ディレクトリー・サービス
NEG	負、折衝

キーワードの省略形	意味
NEP	非終止プログラム
NET	ネットワーク
NFY	通知
NGH	隣接
NL	ネットワーク層
NLSP	NetWare リンク・サービス・プロトコル
NML	名前リスト
NNAM (または NCK)	ニックネーム
NOD	ノード
NODL	ノード・リスト
NORM	正常
NOTVLD	無効
NPRD	非生産
NRM	正常、正規応答モード
NRZI	非ゼロ復帰反転
NT	ネットワーク終端装置
NTB	NetBIOS
NTBD	NetBIOS 記述
NTC	注意
NTF	NetFinity
NTP	Network Time Protocol
NTS	Lotus Notes
NTW	NetWare
NTW3	NetWare 3.12
NUM	数値、番号
NWI	ネットワーク・インターフェース
NWID	ネットワーク・インターフェース記述
NWS	ネットワーク・サーバー
NWSD	ネットワーク・サーバー記述
NXT	次の
OBJ	オブジェクト
OBS	観察可能情報
OFC	オフィス
OFSET	オフセット
OMT	省略
OPN	オープン
OPR	演算子、操作
OPT	オプション、光ディスク、最適
ORD	順序
ORG	編成
ORGUNIT	組織内単位
OS	オペレーティング・システム
OSDB	オブジェクト保管データベース
OUT	出力
OVF	オーバーフロー
OVL	オーバーレイ
OVR	一時変更
OVRFLW	オーバーフロー
OWN	所有者、所有された
PAD	パケット組み立て / 分解
PAG	ページ、ページ編集
PAL	製品アクティビティー・ログ
PARM	パラメーター
PASTHR	パススルー

キーワードの省略形	意味
PBL	可能性
PBX	構内交換
PC	パーソナル・コンピューター
PCD	PC 文書
PCL	プロトコル
PCO	PC 編成プログラム
PCS	パーソナル・コンピューター・サポート
PCT	パーセント
PCTA	パーソナル・コンピューター・テキスト援助
PCY	ポリシー
PDF	Portable Document Format
PDG	印刷記述子グループ
PDM	プログラム開発管理機能
PDU	プロトコル・データ単位
PENWTH	ペンの幅
PERS	パーソナル
PF	物理ファイル
PFnn	プログラム機能キー 'nn'
PFD	印刷出力形式定義
PFM	物理ファイル・メンバー
PFVLM	物理ファイル可変長メンバー
PFR	パフォーマンス
PFX	接頭部
PGM	プログラム
PGP	1 次グループ
PGR	ページャー
PHCP	コード・ページを印刷装置からホストへ
PHFCS	フォント文字セットを印刷装置からホストへ
PHS	フェーズ
PHY	物理
PIN	個人識別番号
PJE	事前開始ジョブ項目
PKA	公開鍵アルゴリズム
PKG	パッケージ
PKT	パケット
PL	プレゼンテーション層
PLC	配置
PLL	ポーリング
PLT	プロッター
PMN	許可
PMP	1 地点から複数の地点へ
PMT	プロンプト
PND	保留中
PNL	パネル
PNT	点
POL	プール
POLL	ポーリングされている、ポーリング
POP	POP (TCP/IP)
PORT	ポート番号
POS	正、位置
PPP	Point-to-Point Protocol
PP	プリプロセッサー
PPR	用紙
PPW	ページ印刷装置書き込み機能

キーワードの省略形	意味
PRB	問題
PRC	プロシージャー、プロシージャーの、プロセス、処理済み
PRD	プロダクト、生産
PREBLT	事前作成
PRED	祖先
PREEST	事前確立
PREF	設定、優先
PREOPR	操作前
PREREQ	前提条件
PRF	プロファイル、プロファイル作成
PRI	1 次
PRJ	プロジェクト
PRL	パラレル
PRM	プロモート、パラメーター
PRMD	私用管理定義域
PRN	親
PRO	推奨
PROC (または PRC)	プロシージャー、処理
PROD	実行用
PROP	プロパティー
PRP	準備、伝搬
PRS	パーソナル
PRT	印刷、印刷装置
PRTQ	印刷待ち行列
PRV	前の
PRX	プロキシー
PSAP	プレゼンテーション層サービス・アクセス・ポイント
PSF	印刷サービス機能
PSN	表示
PSTOPR	操作後
PTC	保護されている、保護、可搬トランザクション・コンピューター
PTF	プログラム一時修正
PTH	パス
PTL	部分的な
PTN	区画、区画化
PTP	2 地点間
PTR	ポインター
PTY	優先度
PUB	公用
PUNS	穿孔
PVC	相手固定接続
PVD	提供者
PVT	私用
PVY	プライバシー
PWD	パスワード
PWR	電源
Q (接尾部)	待ち行列
QE (接尾部)	待ち行列項目
QLTY	品質
QOS	サービス品質
QRY	照会
QST	質問
QSTDB	Q & A データベース
QSTLOD	Q & A ロード

キーワードの省略形	意味
QUAL	修飾子
RAR	経路追加抵抗
RBD	再作成
RCD	レコード
RCDS	レコード
RCL	再利用
RCMS	リモート変更管理サーバー
RCP	宛先
RCR	再帰
RCV	受信
RCY	リカバリー
RDB	リレーショナル・データベース
RDN	相対識別名
RDR	読み取りプログラム
RDRE	読み取りプログラム項目
REACT	再活動化
REASSM	再組み立て
REC	レコード
RECNN	再接続
RECOMMEND	推奨事項
REF	参照
REINZ	再初期設定
REL	関係、リリース
REP	表記、担当者
REQ (または RQS)	必須、要求、要求元
RES	常駐、解像度
RESYNC	再同期
RET	保持期間
REX	REXX 言語
RFS	拒否、拒否された
RGS	登録
RGT	右
RGZ	再編成
RINZ	再初期設定
RIP	ルーティング情報プロトコル
RJE	リモート・ジョブ入力
RJT	拒否
RLS	解放
RMD	想起、覚え書き
RMT	リモート
RMV	除去
RNG	範囲
RNM	名前変更
RPG	RPG 言語
RPL	置換
RPT	報告書
RPY	応答
RQD	必須
RQS (または REQ)	要求、要求元
RQT	必要な
RRSP	リモート応答
RRT	再経路指定
RSB	再組み立て
RSC	資源

キーワードの省略形	意味
RSL	結果、解像度
RSM	再開
RSP	応答
RSRC	資源
RST	復元
RSTD	制限された
RTE	経路
RTG	経路指定
RTL	小売業
RTM	再送
RTN	戻り、戻された、再送
RTR	ルーター
RTT	回転
RTV	検索
RTY	再試行
RU	要求単位
RVK	取り消し
RVS	反転
RWS	リモート・ワークステーション
SAA	システム・アプリケーション体系
SADL	サドル
SAP	サービス・アクセス・ポイント
SAT	土曜日
SAV	保管
SAVF	保管ファイル
SBM	投入、投入された
SBS	サブシステム
SCD	スケジュール、スケジューリングされた
SCH	検索
SCN	画面
SCT	セクター
SDLC	同期データ・リンク制御
SDU	サービス・データ単位
SEC	2 番目の、保護、セキュリティー
SEG	セグメント
SEGMENT	セグメント化
SEL (または SLT)	選択
SENSITIV	機密性
SEP	区切り記号
SEQ	順序、順次
SEV	重大度
SFW	ソフトウェア
SGN	サインオン
SHD	シャドー、シャドー機能
SHF	シフト
SHM	短期保留モード
SHR	共用
SHUTD	シャットダウン
SI	シフトイン
SIG	シグニチャー、署名
SIGN	サインオン
SIZ	サイズ
SL	セッション層
SLR	セレクター

キーワードの省略形	意味
SLT (または SEL)	選択
SMAE	システム管理応用エンティティー
SMG	システム・マネージャー
SMTP	simple mail transfer protocol
SMY	要約
SNA	システム・ネットワーク体系
SNBU	交換網バックアップ
SND	送信
SNG	シングル
SNI	SNA over IPX
SNP	スナップ
SNPT	SNA パススルー
SNUF	SNA アップライン機能
SO	シフトアウト
SOC	制御範囲
SP	サービス・プロセッサー
SPA	スペル援助
SPC	空間、特殊
SPD	提供されている
SPF	特定の
SPID	サービス提供者 ID
SPL	スプーリングされた、スプーリング
SPR	取り替え
SPT	サポート、サポートされている
SPTN	サポート・ネットワーク
SPX	Sequenced Packet Exchange
SQL	構造化照会言語
SRC	ソース
SRCH (または SCH)	検索
SRM	システム資源管理
SRQ	システム要求
SRT	ソート
SRV	サービス
SSAP	ソース・サービス・アクセス・ポイント、セッション層サービス・アクセス・ポイント
SSCP	システム・サービス制御点
SSL	Secure Sockets Layer
SSN	セッション
SSND	セッション記述
SSP	延期
SST	システム・サービス・ツール
STAT	統計データ・レコード
STATION	コンビニエンス・ステーション
STC	統計
STD	標準
STG	記憶域
STK	スタック
STM	ストリーム
STMF	ストリーム・ファイル
STMT	ステートメント
STN	ステーション
STP	ステップ
STPL	ステープル
STR	開始

キーワードの省略形	意味
STS	状況
STT	状態、静的
STX	テキスト開始文字
SUB	置換、サブジェクト
SUBADR	サブアドレス
SUBALC	副次割り振り
SUBDIR	サブディレクトリー
SUBFLR	サブフォルダー
SUBNET (または SUBN)	サブネットワーク
SUBPGM	サブプログラム
SUBR	サブルーチン
SUBST	置換
SUCC	継承者
SUN	日曜日
SUP	抑制、抑止
SURNAM	姓
SVC	スイッチド・バーチャル・サーキット
SVR	サーバー
SWL	停止語リスト
SWS	スイッチ
SWT	スイッチ、交換
SWTSET	スイッチ設定
SYM	記号
SYN	構文
SYNC	同期、同期文字
SYS	システム
SYSLIBL	システム・ライブラリー・リスト
S36	システム/36
TAP	テープ
TAPDEV	磁気テープ装置
TBL	テーブル
TCID	トランスポート接続 ID
TCP	TCP/IP (伝送制御プロトコル / インターネット・プロトコル)
TCS	電話接続サービス
TDC	電話データ収集装置
TDLC	平衡型データ・リンク制御
TEID	端末識別コード
TEL	電話
TELN	TELNET (TCP/IP)
TERM	端末
TFR	転送
TGT	ターゲット
THD	スレッド
THLD	しきい値
THR	スルー、スループット
THRESH (または THLD)	しきい値
THRPUT	スループット
THS	シソーラス
THU	木曜日
TIE	技術情報交換
TIM	時刻
TIMMRK	刻時
TIMO	タイムアウト
TIMOUT (または TIMO)	タイムアウト

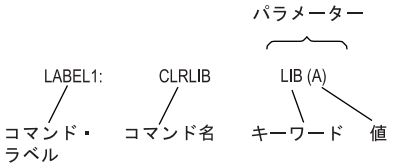
キーワードの省略形	意味
TIMZON	時間帯
TKN	トークン
TKV	テークオーバー
TL	トランスポート層
TM	時刻
TMN	伝送
TMP	一時
TMPL (または TPL)	テンプレート
TMR	タイマー
TMS	伝送
TMT	送信
TNS	トランザクション
TOKN (または TKN)	トークン
TOT	合計
TPDU	トランスポート層プロトコルデータ単位
TPL	テンプレート、トポロジー
TPT	トランスポート
TRANS	中継、トランザクション
TRC	トレース
TRG	トリガー
TRM	用語
TRN	トークンリング・ネットワーク、変換
TRNSPY	透過性
TRP	トラップ
TRS	中継
TRUNC	切り捨て
TSE	タイム・スライス終了
TSP	タイム・スタンプ
TSAP	トランスポート層サービス・アクセス・ポイント
TSK	タスク
TST	テスト
TUE	火曜日
TWR	タワー
TXP	トランスポート
TXT	テキスト
TYP	タイプ
T1	トランスポート・クラス 1
T2	トランスポート・クラス 2
T4	トランスポート・クラス 4
UDFS	ユーザー定義ファイル・システム
UDP	ユーザー・データグラム・プロトコル
UI	ユーザー・インターフェース、無番号の情報
UID	ユーザー ID 番号
UNC	分類していない
UNL	リンク解除
UNPRT	印刷不能
UNQ	固有な
UOM	計測単位
UPCE	汎用製品コード・タイプ E バーコード
UPD	更新
UPG	アップグレード
URL	URL
USG	使用法
USR	ユーザー

キーワードの省略形	意味
VAL	値
VAR	変数
VCT	仮想回線
VDSK	仮想ディスク
VER (または VSN)	バージョン
VFY	検証
VFYPWD	パスワードの確認
VLD	有効、妥当性、妥当性検査
VND	ベンダー
VOL	ボリューム
VRF	検証
VRT	仮想
VRY	変更
VSN (または VER)	バージョン
VWS	仮想ワークステーション
WAN	広域ネットワーク
WCH	監視
WDW	ウィンドウ
WED	水曜日
WIN	勝者
WK	週
WNT	Windows NT
WP	ワード・プロセッシング
WRD	語
WRK	作業、処理
WRT	書き込み
WS	ワークステーション
WSC	ワークステーション制御機構
WSCST	ワークステーション・カスタマイズ・オブジェクト
WSE	ワークステーション項目
WSG	ワークステーション・ゲートウェイ (TCP/IP)
WSO	ワークステーション・オブジェクト
WTR	書き出しプログラム
WTRE	書き出しプログラム項目
WTRS	書き出しプログラム
X25	X.25
X31	X.31
X400	X.400
3270	3270 ディスプレイ

CL コマンドの構成要素

コマンドの構成要素には、コマンド・ラベル (オプション)、コマンド名 (略号)、および 1 つ以上のパラメーターがあります。パラメーターにはキーワードおよび値があります。

この図では、コマンドの各構成要素を示しています。



CL コマンド構文**:**

各コマンドは、CL コマンド名とパラメーターから成っています。

CL コマンドで使用するパラメーターは、キーワード・パラメーターです。キーワードはパラメーターの目的を示すもので、コマンドと同じように略語が使用されます。ただし、コマンドの入力時にパラメーターを所定の順序で指定する (定位置指定) 場合には、キーワードによっては入力を省略できるものもあります。

コマンド構文の一般形式は次のとおりです。大括弧はその中のオプションであることを示します。ただし、パラメーター・セットは、コマンド要件によって、オプションであることもそうでないこともあります。

```
[//] [?] [label-name:][library-name/]command-name
[パラメーター・セット]
```

注**:** // が使用できるのは、DATA コマンドなど一部のバッチ・ジョブ制御コマンドの場合のみです。// は、スプール読み取りプログラムに送られるコマンドであることを示します。スプール読み取りプログラムは、バッチ・ジョブ入力ストリームを読み取ります。

CL コマンド・ラベル**:**

コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

ラベルは、コマンド名の直前に入力されます。ここでは、単純名 (*SNAME) を指定する際の標準的な規則が適用されます。ラベルは、コロンの直後にあります。コロンとコマンド名の間には、必須ではありませんが、ブランクを使用できます。ラベルには、コロンに加えて 10 文字まで含めることができます。 START: および TESTLOOP: は、コマンド・ラベルの例です。

コマンド・ラベルは必須ではありませんが、どのコマンドにもラベルを付けることができます。実行不能のコマンド (**CL 変数宣言 (DCL)** コマンドなど) にラベルを付けた場合、プログラムがそのラベルにブランチすると、そのラベルの次のコマンドが実行されます。ラベルの次のコマンドが実行可能なコマンドではない場合には、プログラムは後続の実行可能なコマンドに進みます。同様に、ラベルは 1 行に 1 つのみ指定することができます。その行にコマンドがない場合は、プログラムは実行可能な次のコマンドに進みます。

複数のラベルを指定する場合には、次の例のようにコマンドの前に、1 行に 1 つの追加ラベルを入れた行を指定する必要があります。

```
LABEL1:
LABEL2:  CMDX
```

コマンド行の前のラベル行で、継続文字 (+ または -) を使用することはできません。

関連概念:

136 ページの『固有名に関するその他の規則』
オブジェクトの命名については、特殊文字 (付加文字) に関して以下のような追加の規則があります。

117 ページの『コマンド内での命名』
名前の指定に使用できる文字は、制御言語 (CL) で指定する名前のタイプによって決まります。

132 ページの『通信名 (*CNAME)』
通信名の名前構文には制限があります。通常は、名前の長さと有効な文字セットが 1 つ以上の通信アーキテクチャーによって制限されている、装置構成オブジェクトを参照するために使用します。

133 ページの『名前 (*NAME)』
基本名および引用符付きの基本名を作成する際は、以下の規則に従ってください。

134 ページの『パス名 (*PNAME)』
パス名は、統合ファイル・システムの中でオブジェクトを位置指定するために使用できる文字ストリングです。

136 ページの『単純名 (*SNAME)』
制御言語 (CL) の構文を簡素化するために、CL 変数、ラベル、およびキーワードには、単純名が使用されます。単純名は、引用符の付いていない基本名と同じですが、相違点が 1 つあります。単純名ではピリオド (.) は使用できません。

CL コマンド・パラメーター**:**

ほとんどの CL コマンドは、そのコマンドを実行するために使用するオブジェクトおよび値を指定する、1 つ以上のパラメーター を持っています。

コマンドを入力する際に、コマンドで使用するコマンド・オブジェクト名、パラメーター・キーワード名、およびパラメーター値を指定します。指定するパラメーターの数はコマンドによって異なります。パラメーターのないコマンド (例えば、 **DO (実行)** コマンドや **ENDBCHJOB (バッチ・ジョブ終了)** コマンドなど) もあれば、 1 つ以上のパラメーターのあるコマンドもあります。

このトピックでは、パラメーター という用語は、通常、パラメーター・キーワードとその値の組み合わせを意味します。 例えば、**オブジェクト移動 (MOVOBJ)** コマンドには OBJ というパラメーターがあり、これにはオブジェクト名を指定しなければなりません。OBJ はパラメーター・キーワードであり、オブジェクト名は OBJ パラメーターに入力する値です。

関連資料:

65 ページの『値のリスト』
値のリストとは、1 つのパラメーターで指定できる 1 つ以上の値のことです。

67 ページの『必須パラメーター、オプション・パラメーター、およびキー・パラメーター』
CL コマンドのパラメーターには、必ず指定しなければならないパラメーター (必須パラメーター) と、必ずしも指定を必要としないパラメーター (オプション・パラメーター) があります。

66 ページの『キーワード形式および定位置形式のパラメーター』
CL におけるパラメーターは、キーワード形式、定位置形式、あるいはこの両方の組み合わせのいずれかを使用して指定できます。

57 ページの『パラメーター値』
パラメーター値は、コマンドの実行中に使用されるユーザー提供の情報です。

CL コマンドの区切り文字**:**

コマンド区切り文字は、コマンドの中で、文字のグループの始まりと終わりを示す特殊文字またはスペースです。

区切り文字は、1 つの文字ストリングを、コマンドを形成する個々の部分に分けるために使用します。コマンドを形成する部分には、コマンド・ラベル、コマンド名、パラメーター・キーワード、およびパラメーター値があります。パラメーター値として指定できるのは、定数、変数名、リスト、または式です。次の図に、コマンドの各種区切り文字を示します。

IBM i 制御言語では、次の区切り文字が使用されます。

- コロン (:) は、コマンド・ラベルの終わりを示すもので、コマンド・ラベルとコマンド名を区切るために使用されます。
- ブランク・スペースは、コマンド名とパラメーターとの間、およびパラメーターとパラメーターとを区切るために使用されます。また、リスト中の値と値の間もブランクで区切ります。複数のブランクは、引用符で囲んだ文字ストリング、または単一引用符で囲んだ注記の中で使われている場合を除いて、1 つのブランクとして扱われます。キーワードと、その値の左括弧との間をブランクで区切ることはできません。
- 括弧 ( ) は、パラメーター値とそのキーワードを区切る場合、値のリストをグループ分けする場合、およびリスト内のリストをグループ分けする場合に使用します。
- 斜線 (/) は、修飾名の各部分またはパス名の各部分を結合します。
  - 修飾オブジェクト名は、ライブラリー修飾子とオブジェクト名の 2 つの部分からなり、次のように斜線で区切られます (LIBX/OBJA)。
  - パス名は、検索する 1 つ以上のディレクトリーとオブジェクト名で構成されます ('/Dir1/Dir2/Dir3/ObjA')。
- 10 進数の小数点にはピリオドまたはコンマを使用することができます (3.14 または 3,14)。1 つの値に対して、小数点は 1 つしか使用できません。
- 単一引用符は、引用符付き文字ストリングの始まりと終わりを示します。引用符付き文字ストリングは、256 個の拡張 2 進化 10 進コード (EBCDIC) 文字を任意に組み合わせたもので、定数として使用されます。例えば、'YOU CAN USE $99@123.45 ()*></ and lowercase letters' は、有効な引用符付き文字ストリングです。

  引用符付き文字ストリング内の単一引用符は、始めの単一引用符 (区切り文字) と対をなすものと見なされ、終わりの単一引用符と解釈されてしまうため、引用符付き文字ストリングの中で単一引用符を使用する場合には、2 重の単一引用符を使用しなければなりません。このように使用した場合の隣接する 2 つの単一引用符は 1 文字として扱われます。
- 日付を、月、日、年の 3 つの部分 (年間通算日の場合は年と月の 2 つの部分) に分けるには、特殊文字を使用します。日付区切り文字として使用できる特殊文字は、斜線 (/)、ハイフン (-)、ピリオド (.)、ブランク ( )、およびコンマ (,) です。コマンド中の日付の区切り文字として使用する特殊文字は、対象のジョブでの日付区切り文字として指定されている特殊文字と、同じでなければなりません。
- /* および */ は、注記の始めと終わりを表し、文字ストリングの中でも使用することができます。注記を開始するには、/* をコマンドの最初の桁に置くか、この 2 文字の前にブランクを 1 つ付けるか、あるいはその後にブランクまたはアスタリスクを 1 つ付けなければなりません。/* または */ がコマンドの後ろの位置に現れた場合には、たいていの場合、単一引用符で囲まれ、例えばパス名の現行ディレクトリーにあるすべてのオブジェクトなどを表すことができます。
- コマンド名の前の疑問符 (?) は、そのコマンドに関するプロンプトが出されることを示します。コマンドにラベルが付いている場合、疑問符は、ラベルの前に置くことも、ラベルとコマンド名の間に置くこともできます。

  CL プログラム内で、コマンド名の前に疑問符がある場合には、プロンプト画面が表示されます。プロンプト画面では、プログラム内でそのコマンド上に指定されていないパラメーターの値を入力することができます。

  コマンドへのプロンプト要求文字の組み込みには、2 つの形式があります。コマンド名の前 (CL プログラム内のコマンド・ラベルの前または後) にシングルの疑問符 (?) をコーディングすれば、そのコマン

ド全体についてのプロンプトが表示されます。いずれかのパラメーター・キーワードの前に選択プロンプト要求文字 (?? または ?*) をコーディングすれば、コマンドの実行時にそのパラメーターに関するプロンプトが表示されます。

コマンド入力画面でコマンド名の前に疑問符を入力すれば、コマンドの入力後に F4 (プロンプト) キーを押した場合と同じ結果が得られます。

CL プログラム内で、コマンド名の前に疑問符が付いていると、プロンプト画面が表示されます。この画面は、コマンド入力画面で F4 キーを押した場合に表示される画面と同じ形式のものです。コマンドのパラメーターのうち、プログラム内で値がコーディングされているパラメーターも、単なる情報として表示されますが、このようにプログラムから提供された値を、ユーザーが変更することはできません。値の指定されていないパラメーターは入力フィールドとして表示されるので、ユーザーはコマンドの処理に使用する値を入力することができます。

選択プロンプト要求を使用すれば、コマンドの特定のパラメーターについて、プロンプトを表示させることができます。選択プロンプトを呼び出すためには、プロンプトを表示したいパラメーター (1 つまたは複数) のキーワード名の直前に、??、?*、または ?- をコーディングします。

注**:**

1. 選択プロンプトは、コマンド・ストリング (*CMDSTR) パラメーターとともに使用することはできません。
2. コマンドのパラメーターのうち、前に ?* のついたパラメーターは、表示はされますが、プログラムが提供する値を変更することはできません。前に ?? のついたパラメーターは、プログラム内でコーディングされている値、またはコマンドのデフォルトが入った入力フィールドとして表示されるため、ユーザーは、コマンドの処理に使用する値を入力または変更することができます。前に ?- のついたパラメーターは、表示画面上で省略されるパラメーターです。選択プロンプト要求で出されたすべてのパラメーターは、キーワード形式、または値付きのキーワード形式でコーディングしなければなりません。1 つのコマンドで、いくつかのパラメーターに、選択プロンプト要求を指定することもできます。選択プロンプトを呼び出す場合には、前に選択プロンプト要求文字が付いているキーワードについてのみ、プロンプトが表示されます。その他のパラメーターは、コマンド上にコーディングされている値 (値がコーディングされていなければ、コマンドのデフォルト) を使用して処理されます。

   CL プログラム内の 1 つのコマンドについて、上記の 2 つのプロンプト方式のいずれも使用することができますが、両方を使用することはできません。コマンド名の前に ? があり、いずれかのキーワードの前に、選択プロンプト要求文字 (?- を除く) がある場合は、エラー・メッセージが出され、プログラムは作成されません。

関連概念:

127 ページの『単純オブジェクト名および修飾オブジェクト名』
ライブラリー内の特定のオブジェクトの名前は、単純名としても修飾名としても指定できます。

45 ページの『CL コマンド・コーディング規則』
ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

45 ページの『CL コマンド定義』
コマンド定義機能を使用することにより、ユーザーはアプリケーションの特殊な要件に対応する新たな制御言語コマンドを作成することができます。ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。

関連タスク:

352 ページの『CL コマンド・パラメーターのプロンプト制御の指定』
プロンプト制御の指定を用いることにより、プロンプトの過程でコマンドのどのパラメーターを表示するかを制御できます。

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

関連情報:

統合ファイル・システム

CL コマンドの継続:

コマンドは自由形式で入力することができます。つまり、コーディング用紙または画面上の決まった位置から、コマンドを開始する必要はありません。コマンドは、その全体を 1 つのレコードに収めることも、複数の行またはレコードに継続させることもできます。

継続するかしないかには関わらず、コマンドの合計長は 32,702 文字を超えることはできません。コマンドの継続を示すためには、その行の最後の非ブランク文字として、2 つの特殊文字、すなわち正符号 (+) または負符号 (-) のいずれかを入力します。+ または - 符号のすぐ前 のブランクは常にコマンドに含まれ、同じレコード 内の + または - のすぐ後のブランクは無視されます。次のレコード の最初の非ブランク文字より前のブランクは、+ を指定した場合には無視されますが、- を指定した場合にはコマンドに含まれます。

一般に、+ はパラメーター相互間または値相互間で使用すると便利です。パラメーター相互間または値相互間で正符号を使用する場合には、符号の前に少なくとも 1 つのブランクを置かなければなりません。正符号と負符号の使い分けは、引用符付きの文字ストリングの途中で継続が生じた場合に、特に重要な意味を持ちます。

次の例でこの違いを示します。

```
CRTLIB LIB(XYZ) TEXT('This is CONT+
   INUED')

CRTLIB LIB(XYZ) TEXT('This is CONT-
   INUED')

+ の場合: CRTLIB LIB(XYZ) TEXT('This is CONTINUED')

- の場合: CRTLIB LIB(XYZ) TEXT('This is CONT   INUED')
```

注:

1. 負符号を指定すると、次の行の先行ブランクがコマンドに含まれます。
2. 継続文字としての + および - は CL プログラム内でのみ使用できます。コマンド入力画面で + または - を使用するとエラーが起こります。
3. + および - は、単一コマンドの例ではなく、複数コマンドの例に使用されています。

CL コマンドの注記:

CL プログラム内の注記は、予想されるコードの振る舞いを説明します。

注記はブランクが使用できる個所であれば、コマンドの文字ストリングの中でも外でも使用することができます。ただし、継続文字は行 (またはレコード) の最後の非ブランク文字にする必要があるので、その行の継続文字の後に注記を入れることはできません。

次の例に示すように、注記は、それに関連したコマンドの前または後に独立した行を設けてコーディングする方が、読みやすくなり便利です。

```
MOVOBJ   OBJA  TOLIB(LIBY)
       /* Object OBJA is moved to library LIBY. */
DLTLIB  LIBX
       /* Library LIBX is deleted. */
```

注記には、256 個の EBCDIC 文字のどれでも使用することができます。ただし、*/ という文字の組み合わせは注記の終わりを示すので、これを注記の中で使用することはできません。注記を開始するには、/* をコマンドの最初の桁に置くか、その前にブランクを 1 つ付けるか、あるいはその後にブランクまたはアスタリスクを 1 つ付けなければなりません。

CL コマンド定義

コマンド定義機能を使用することにより、ユーザーはアプリケーションの特殊な要件に対応する新たな制御言語コマンドを作成することができます。ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。

関連概念:

41 ページの『CL コマンドの区切り文字』
コマンド区切り文字は、コマンドの中で、文字のグループの始まりと終わりを示す特殊文字またはスペースです。

127 ページの『単純オブジェクト名および修飾オブジェクト名』
ライブラリー内の特定のオブジェクトの名前は、単純名としても修飾名としても指定できます。

『CL コマンド・コーディング規則』
ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

関連タスク:

352 ページの『CL コマンド・パラメーターのプロンプト制御の指定』
プロンプト制御の指定を用いることにより、プロンプトの過程でコマンドのどのパラメーターを表示するかを制御できます。

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

関連情報:

統合ファイル・システム

CL コマンド・コーディング規則

ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

区切り文字の CL コマンド・コーディング規則

- ブランクは、コマンドの各部分の間に使用される基本的な区切り文字です。

  - コマンド・ラベルとコマンド名との間 (区切り文字としてコロン (:) が使用されるため、ブランクは必須ではありません)。
  - コマンド名と最初のパラメーターの間、およびパラメーターとパラメーターの間。
  - 値のリスト内の値と値との間 (リスト内のネストされたリストの、終わりの括弧と始めの括弧との間には必要ありません)。
  - // DATA などのジョブ制御コマンドにおける斜線と、名前またはラベルとの間 (必須ではありません)。
- パラメーターのキーワードと、そのパラメーターの値の前にある左括弧との間を、ブランクで区切ることはできません。キーワードを使用する場合には、値を括弧で囲まなければなりません。括弧と値の間にブランクを入れることができます。例えば、KWD(A) のようにすることができます。
- 複数のブランクは、引用符付きストリングまたは注記の中で使用されている場合を除いて、1 個のブランクとして扱われます。
- コマンド・ラベルの直後にはコロンを付けなければなりません。どのコマンドでも、ラベルは 1 つしか使用することができません (LABEL1: DCLF)。
- 引用符付き文字ストリングの始めと終わりには、単一引用符を使用します。ブランクなどの特殊文字を含む文字ストリングの場合には、単一引用符は必須です。引用符付きストリングの中で単一引用符を使用しなければならない場合には、それが引用符付き文字ストリングの終わりを示す単一引用符でないことを示すために、単一引用符を 2 つ連続して使用する必要があります。
- 次の場合には括弧を使用しなければなりません。
  - パラメーターをキーワード形式で指定 (コーディング) する場合。
  - 定位置パラメーターで複数の値をまとめて 1 つのリストにする場合、または式の前後。
  - 他のリストの中にリスト (要素なし、または 1 つ以上の要素のある) をネストして指定する場合。
- 括弧内括弧は、それぞれ対になっていれば、論理式の中では最高 5 レベルまで、値のリストでは最高 3 レベルまでネストすることができます。
- 注記は、行 (またはレコード) の継続文字の後を除いて、ブランクが許されるところであればどこにでも入れることができます。
- 行の終わりの正符号または負符号は、コマンドが次の行に続くことを示します。同じレコード内の + または - 符号の後ろにあるブランクは無視されます。次のレコードの最初の非ブランク文字より前にあるブランクは、 + が指定されている場合には無視され、- が指定されている場合には組み込まれます。パラメーターとパラメーターの間、または値と値の間で + 符号を使用する場合には、その前にブランクを 1 つ置かなければなりません。

パラメーターの CL コマンド・コーディング規則

- 必須パラメーターはすべてコーディングする必要があります。
- オプション・パラメーターをコーディングしなかった場合には、そのパラメーターにデフォルト値があればシステムはそれを使用します。デフォルト値は、パラメーターの要約表の選択項目欄に太字の下線付きテキストの値で示されます。
- コマンドおよびパラメーターの説明の中で英字の大文字で示されている語句または略語は、そのとおりにコーディングしなければなりません。これは、すべての簡略コマンド名、パラメーターのキーワード (使用する場合)、および多くのパラメーターの値に当てはまります。引用符付きストリングまたは注記以外のところで小文字をコーディングした場合には、それぞれ対応する大文字に変換されます。
  CASE(*MIXED) として定義されたパラメーターに値として指定された小文字は、大文字に変換されません。

- キー・パラメーターがある場合には、キー・パラメーターのために使用する値を、残りのパラメーターが表示される前にプロンプトに入力しなければなりません。パラメーターの要約表の「注 (Notes)」欄には、キー・パラメーター (存在する場合) であるパラメーターを示します。
- コマンド・オブジェクトに定義されている定位置パラメーター制限の後のパラメーターは、定位置形式でコーディングすることができません。パラメーターの要約表の「注 (Notes)」欄には、定位置形式で指定できるパラメーターを示します。

パラメーター値の CL コマンド・コーディング規則

- すべての名前は、最初の文字が英字 (A から Z、$、#、@ または 2 重引用符 (")) でなければなりません。名前の長さは 10 文字以下でなければなりません (CL 変数名および組み込み関数名は、名前の最初に付ける & または % も含めて、最高 11 文字まで使用することができます)。コマンドによっては、オブジェクトの名前を、修飾形式 (ライブラリー名/オブジェクト名)、またはパス名形式 (ディレクトリー名/オブジェクト名) で、指定できるものもあります。
- アスタリスクで始まる事前定義値は、注記または引用符付きストリング内で使用する場合を除いて、定められた目的以外の用途に用いることはできません。事前定義値には、事前定義パラメーターの値 (*ALL など)、記号演算子 (*EQ など)、およびヌル値 (*N) があります。
- CL プログラム内では、明示的に禁止されている個所を除いて、すべてのパラメーターに対して変数を指定できます。変数のユーザー指定値は、コマンドでその値が指定された場合と同じに扱われます。
- CL プログラム内では、EXPR(*YES) の指定されているどのパラメーターにも、文字ストリング式を指定できます。式の演算結果の値が、コマンドでその値が指定された場合と同じに扱われます。
- ヌル値 (省略を表す) は、*N で指定します。これは、該当するパラメーターについて値が指定されていないため、デフォルト値があればそれが使用されるということを意味します。 *N が必要となるのは、省略した値の次の値を定位置パラメーターまたはリスト要素として指定する場合のみです。
- 数値内の小数点には、コンマまたはピリオドのいずれかを使用できます。小数点は、数値ストリングの中の数字と数字の間で使用できる唯一の特殊文字です。例えば、3 桁ごとの区切りを示す記号は使用しません。
- パラメーターの反復が指示されている場合には、
  - 事前定義値は、一連の値の中で 2 回以上コーディングすることはできません。
  - ユーザー定義値 (名前または数値の範囲など) は、異なる値や名前がいくつあっても、そのそれぞれについて、許される最大反復回数まで入力できます。例えば、パラメーターの記述に「20 回まで反復可能」とある場合には、最大で値 20 を指定できます。

注: 別々のコマンドで同じ名前のパラメーターを使用する場合、そのパラメーターの意味 (または値) は、コマンドごとに多少異なる場合があります。したがって、パラメーターの説明を読む際には、必ず、使用しようとしているコマンドの項に示されている説明をお読みください。

関連概念:

41 ページの『CL コマンドの区切り文字』
コマンド区切り文字は、コマンドの中で、文字のグループの始まりと終わりを示す特殊文字またはスペースです。

45 ページの『CL コマンド定義』
コマンド定義機能を使用することにより、ユーザーはアプリケーションの特殊な要件に対応する新たな制御言語コマンドを作成することができます。ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

68 ページの『共通に使用されるパラメーター』
一部のパラメーターは、多数の CL コマンドで共通して使用されます。

66 ページの『キーワード形式および定位置形式のパラメーター』
CL におけるパラメーターは、キーワード形式、定位置形式、あるいはこの両方の組み合わせのいずれかを使用して指定できます。

57 ページの『パラメーター値』
パラメーター値は、コマンドの実行中に使用されるユーザー提供の情報です。

CL コマンド情報および文書

IBM は、CL コマンドに関する文書を提供しています。さらに、独自のコマンドについての文書を作成することもできます。

関連タスク:

385 ページの『CL コマンドの文書化』
独自の CL コマンドを定義および作成する場合は、コマンドを説明するオンライン・コマンド・ヘルプを作成することもできます。

CL コマンドの文書形式:

CL コマンドのコマンド文書は、システムのオンライン・ヘルプおよび IBM i Information Center のトピックの形式で提供されます。Information Center では、各コマンドは、同じ形式に従って記述されています。

各コマンドの記述には、以下のサブトピックで説明する構成要素が含まれています。コマンド記述の文書の始めには、「パラメーター」、「例」、および「エラー・メッセージ」のセクションへのリンクがあります。

コマンドは IBM i のオブジェクトであるため、各コマンドは、特定のユーザーに対してのみその使用権限を与えることも、共通権限ユーザー (何らかの形でシステムを使用する権限が与えられているすべてのユーザー) に対して使用権限を与えることも可能である点に注意してください。これはほぼすべてのコマンドについて当てはまるため、各コマンドの記述の項では触れません。

関連情報:

セキュリティー参照

***CL* 環境分類:**

コマンド記述文書の最上部にあるのは環境分類です。環境分類は、コマンドがどこで実行できるのかを記述します。

実行許可環境では、そのコマンドを入力できる環境を示します。この情報は、**コマンド表示 (DSPCMD)** コマンドの出力で示される情報と同じで、コマンド定義オブジェクトが作成されたときの ALLOW パラメーターに指定されたものを表します。実行許可環境 値には、ALLOW パラメーターのために指定された記号特殊値、およびコマンドの実行が許可された環境を説明する短い記述が含まれます。

コマンドの多くは、ALLOW(*ALL) で作成されます。*ALL も ALLOW パラメーターの出荷時のデフォルト値です。この場合、記述は、「すべての環境 (*ALL)」になります。

対話式に実行する必要のあるコマンドの場合は、コマンドが作成されたときに指定された ALLOW 値は通常、 (*INTERACT *IPGM *IREXX *EXEC) または (*INTERACT *IPGM *IMOD *IREXX *EXEC) です。これら 2 つの場合、表示される記述は、「対話環境 (*INTERACT *IPGM *IREXX *EXEC)」または、「対話環境 (*INTERACT *IPGM *IMOD *IREXX *EXEC)」です。

コンパイルされた CL またはインタープリットされた REXX プログラムの場合、コマンドが作成されたときに指定された ALLOW 値は通常、(*BPGM *IPGM *BREXX *IREXX) または、(*BPGM *IPGM *BMOD *IMOD *BREXX *IREXX) です。これら 2 つの場合、表示される記述は「コンパイルされた CL プログラム、インタープリットされた REXX (*BPGM *IPGM *BREXX *IREXX)」または「コンパイルされた CL あるいはインタープリットされた REXX (*BPGM *IPGM *BMOD *IMOD *BREXX *IREXX)」です。

コマンドが作成されたときに ALLOW パラメーターに指定された値の組み合わせが、上記のうちの一つでない場合は、指定されたそれぞれの値の簡単な記述を示す黒丸付きリストが表示されます。

- バッチ・ジョブ (*BATCH)
- 対話式ジョブ (*INTERACT)
- バッチ ILE CL モジュール (*BMOD)
- 対話式 ILE CL モジュール (*IMOD)
- バッチ・プログラム (*BPGM)
- 対話式プログラム (*IPGM)
- バッチ REXX プロシージャー (*BREXX)
- 対話式 REXX プロシージャー (*IREXX)
- QCMDEXEC、QCAEXEC、または QCAPCMD API の使用 (*EXEC)

注: IBM i オペレーティング・システムの一部として出荷されるコマンド定義オブジェクトには、CL コマンドとしての使用を意図していないものがあります。例えば、CMD および PARM コマンド定義オブジェクトは、コマンド定義ソースで使用されます。これら特別な目的のコマンド・オブジェクトには、実行許可環境 情報はありません。

関連概念:

『CL スレッド・セーフの分類』

スレッド・セーフの分類は、コマンドがスレッド・セーフかどうかを示します。各コマンドにはスレッド・セーフの分類があります。

CL スレッド・セーフの分類*:*

スレッド・セーフの分類は、コマンドがスレッド・セーフかどうかを示します。各コマンドにはスレッド・セーフの分類があります。

スレッド・セーフの分類の 3 つのタイプを以下に示します。

- Threadsafe: Yes

  この分類のコマンドは、制約なしで同時に複数のスレッドから安全に呼び出すことができます。また、この分類のコマンドによって呼び出された機能も、すべてスレッド・セーフです。

- Threadsafe: Conditional

  この分類は、コマンドの提供する機能が必ずしもすべてスレッド・セーフではないことを示しています。コマンドの制約条件の項には、スレッド・セーフの制限に関する情報が記載されています。多くのコマンドは、下位のシステム・サポートのいくつかがスレッド・セーフではなかったり、出口点を呼び出したりできるため、条件付きスレッド・セーフに分類されます。

- Threadsafe: No

  この分類のコマンドはスレッド・セーフでないため、マルチスレッド化されたプログラムで使用してはなりません。スレッド・セーフではないコマンドの中には、アクセスを拒否するものもありますが、スレッド・セーフでないコマンドのほとんどはアクセスを拒否しません。診断メッセージ CPD000D がジョブ・ログに送信され、非スレッド・セーフのコマンドが呼び出されたことが示される場合があります。診断メッセージ CPD000D がジョブ・ログに送信されるかは、そのコマンドの「マルチスレッド・ジョブ・アクション」属性によって決まります。この属性は、**コマンド表示 (DSPCMD)** コマンドを使用して判別することができます。指定可能な値および実行可能な処置は次のとおりです。

  – *SYSVAL - システム値 QMLTTHDACN に基づいた処置が行われます。
  – *RUN - コマンドが実行されます。メッセージは送信されません。
  – *MSG - 診断メッセージ CPD000D がジョブ・ログに送信されます。コマンドは実行されます。
  – *NORUN - 診断メッセージ CPD000D がジョブ・ログに送信され、エスケープ・メッセージ CPF0001 が送信されます。コマンドは実行されません。

  このコマンドは、実行されても予測できない結果を生じます。

注**:** IBM i オペレーティング・システムの一部として出荷されるコマンド定義オブジェクトには、CL コマンドとしての使用を意図していないものがあります。例えば、CMD および PARM コマンド定義オブジェクトは、コマンド定義ソースで使用されます。これらの特殊目的コマンド・オブジェクトには、「スレッド・セーフ」情報はありません。

関連概念:

48 ページの『CL 環境分類』

コマンド記述文書の最上部にあるのは環境分類です。環境分類は、コマンドがどこで実行できるのかを記述します。

CL コマンドの説明**:**

一般的説明では、コマンドの機能およびコマンドとプログラムや他のコマンドとの関係について簡潔に説明しています。 コマンドに使用上の制約があれば、制約条件の見出しで説明します。環境およびスレッド・セーフ分類に続いてコマンドの一般的説明があります。

CL パラメーター**:**

パラメーターのセクションでは、パラメーター要約表を示します。

パラメーター要約表には、コマンドで使用できるパラメーターおよび値がすべて示されています。選択項目欄には、指定可能な値を示しています。 IBM 出荷時のデフォルト値には選択項目欄に下線が引かれています。デフォルト値は、ユーザーがパラメーター (またはその一部) をコーディングしなかった場合、またはシステムにより使用される値です。

パラメーターの記述

パラメーターの記述は、パラメーター要約表に従って行われます。パラメーターの記述では、各パラメーターは、パラメーター要約表に示されている順序で提示されます。それぞれのパラメーター記述には、パラメーターの機能の記述に続いて指定可能なパラメーター値についての説明があります。デフォルト・パラメーター値があれば、通常それは最初に示され、デフォルト値であることがわかるように見出しには下線が付けられています。

各パラメーターの説明では、パラメーターの意味、指定する値、およびコマンドの他のパラメーターとの従属関係が説明されています。パラメーターに複数の値がある場合には、パラメーター全体に適用される情報を先に示し、その後に、各値についての個別の情報を値の名前ごとに記述してあります。

パラメーターの要約表

パラメーターの要約表は、CL コマンドのパラメーターと値を要約したものです。

キーワード欄

この欄は、パラメーター・キーワード 名を示します。各 CL コマンド・パラメーターには、そのコマンドに関連したキーワード名があります。ブラウザーを使用してコマンドの資料を表示する場合、コマンドの資料ファイル内にある、パラメーターの情報の始まりにリンクするキーワード名の上でクリックすることができます。

記述欄

この欄は、パラメーター、パラメーター修飾子、またはパラメーター要素に定義されたプロンプト・テキストを示します。修飾子 は、通常修飾オブジェクト名または修飾ジョブ名で使用されます。要素 は、単一のパラメーターに複数の入力フィールドを定義するために使用されます。修飾子または要素の記述には、パラメーター内の修飾子または要素数が含まれています。

選択項目欄

この欄は、パラメーター、修飾子、または要素で使用可能な値を示します。

- 事前定義値 (別名特殊値) は、この欄にリストされています。事前定義値は、通常、アスタリスク (*) または Q で始まり、その後に大文字のみが続きます。
- パラメーター、修飾子、または要素でユーザー定義の値 を使用できる場合、パラメーター・タイプの記述は、イタリック (例えば、名前) で示されます。
- オプション・パラメーターには、デフォルト値 が定義されていることがあります。デフォルト値は、太字の下線付きテキスト (例えば、 ***NO**) で示されます。
- 複数の修飾子または要素を持つ複合パラメーターの場合、あるいはパラメーターまたは要素が値のリストをサポートする場合は、任意の単一値 を選択することができます。単一値は、一度のみ選択できます。
- パラメーターによっては、反復値 を指定できます。反復値では、この欄は使用できる反復の数を示します。

注欄

この欄は、各パラメーターに関する追加情報を示します。

- この欄に表示される「必須」は、入力値を指定するために常に必要なパラメーターである、必須パラメーター を示します。
- この欄に表示される「オプション」は、入力値を必要としないパラメーターである、オプション・パラメーター を示します。
- この欄に表示される「キー」は、プロンプト指定変更プログラムを持つコマンドが使用する、キー・パラメーター を示します。
- コマンド・ストリングでパラメーターを (関連パラメーター・キーワードを使用せずに) 定位置形式で指定できる場合、この欄には「定位置」が表示されます。パラメーターの定位置番号は次の「定位置 (Positional)」に示されます。

CL コマンド・コーディング例*:*

『例』セクションでは、コード化されたコマンドの例が少なくとも 1 つ用意されています。パラメーター数が多く論理的な組み合わせがいくつかあるコマンドの場合には、必要に応じて複数のコーディング例を示しています。

明確にするために、例はキーワード形式でのみコーディングされています。1 つ以上の定位置形式パラメーターをサポートするコマンドのために、同じ例を、定位置形式またはキーワード形式と定位置形式との組み合わせを使用してコーディングすることもできます。

CL エラー・メッセージ*:*

『エラー・メッセージ』セクションでは、コマンドに対して出される可能性のあるエラー・メッセージをリストしています。

CL コマンド・ヘルプ**:**

すべての IBM CL コマンドについて、オンライン・ヘルプを使用できます。

オンライン・ヘルプには、コマンドのパラメーターおよび値についての説明が含まれています。コマンドのヘルプを表示するには、以下のいずれかの方法を実行します。

- IBM i のコマンド行からコマンド名 (例えば CRTUSRPRF) を入力し、F1 を押します。そのコマンドの一般ヘルプおよび各コマンドのパラメーターのヘルプが画面に表示されます。
- IBM i のコマンド行からコマンド名 (例えば CRTUSRPRF) を入力して F4 を押し、コマンド・プロンプト画面を表示します。プロンプト画面で先頭行までカーソルを移動し、F1 を押します。

システムでの **CL** コマンドの説明の印刷**:**

CL コマンドのパラメーターおよび値の説明を印刷する際には、以下の手順に従ってください。

あるコマンドに関する完全なヘルプを印刷するには、次のいずれかのステップを実行します。

- 任意のコマンド行からコマンド名 (例えば CRTUSRPRF) を入力し、F1 を押します。そのコマンドの一般ヘルプおよび各コマンドのパラメーターのヘルプが画面に表示されます。F14 を押すと、そのコマンド・ヘルプが印刷されます。
- 任意のコマンドのプロンプト画面で先頭行までカーソルを移動し、F1 を押します。その後 F14 を押します。

CL コマンドの特定のキーワード・パラメーターに関するヘルプを印刷するには、以下のステップを実行します。

- コマンド行から CL コマンド名を入力して F4 を押し、コマンドのプロンプト画面を表示します。ヘルプが必要なキーワード・パラメーターの行の任意の位置にカーソルを置きます。F1 を押すと、そのキーワード・パラメーターに関するヘルプが表示されます。F14 を押すと、そのヘルプが印刷されます。

関連資料:

2 ページの『制御言語の PDF ファイル』
この情報の PDF ファイルを表示または印刷できます。

CL コマンド・プロンプター**:**

コマンド・プロンプターを使用すれば、CL コマンド・パラメーターと値を求めるプロンプトを出すことができます。

コマンド・プロンプターは直接呼び出すか、アプリケーション・プログラムから呼び出すことができます。プロンプターを使用すると、プロンプターがユーザーに代わってパラメーター・キーワード名やパラメーター区切り文字 (単一引用符や括弧など) を挿入するので、構文的に正しい CL コマンド・ストリングを容易に構築できます。 CL プロンプターではまた、オンライン・コマンド・ヘルプにアクセスできます。これは、コマンド、パラメーターとパラメーター値、コマンド例、およびコマンドが出すエラー・メッセージを説明するために使用できます。

System i® ナビゲーター は、クライアント PC で使用できるグラフィカルな CL コマンド・プロンプターを備えています。 IBM System i Access for Web は、Web ブラウザーで使用できる HTML フォーム・ベースの CL コマンド・プロンプターを備えています。 IBM Rational® Development Studio for i ライセンス・プログラムの機能であるリモート・システム・エクスプローラー (RSE) 機能も、グラフィカルな CL コマンド・プロンプターを備えています。

IBM i オペレーティング・システムは、コマンド行から F4 を押して使用できる CL コマンド・プロンプターを備えています。さらに、コマンド行ウィンドウの表示 (QUSCMDLN) API を使用して、アプリケーション内からコマンド行を表示することができます。

IBM i オブジェクトに対して機能する CL コマンド

IBM i の各オブジェクト・タイプには、そのオブジェクト・タイプを処理する 1 組のコマンドがあります。

ほとんどの IBM i オブジェクト・タイプには、次の作業を実行するコマンドがあります。

- 作成 (CRT): オブジェクトを作成し、その属性を指定します。
- 削除 (DLT): システムからオブジェクトを削除します。
- 変更 (CHG): オブジェクトの属性または内容、またはその両方を変更します。
- 表示 (DSP): オブジェクトの内容を表示します。表示コマンドは、オブジェクトの処理には使用できません。
- 処理 (WRK): オブジェクトの属性または内容、またはその両方を処理します。表示コマンドの場合と異なり、処理コマンドではオブジェクトを操作しアプリケーションを変更することができます。

関連資料:

『複数のオブジェクトを処理する CL コマンド』
単一オブジェクト・タイプに対して機能するコマンドに加えて、複数のオブジェクト・タイプに対して機能するコマンドもあります。これらのコマンドは、異なるタイプの複数のオブジェクトを同時に扱えるので、より強力です。

複数のオブジェクトを処理する CL コマンド

単一オブジェクト・タイプに対して機能するコマンドに加えて、複数のオブジェクト・タイプに対して機能するコマンドもあります。これらのコマンドは、異なるタイプの複数のオブジェクトを同時に扱えるので、より強力です。

例を以下に示します。

- オブジェクト記述表示 (DSPOBJD または DSPLNK) は、オブジェクトの共通属性を表示します。
- オブジェクト保管 (SAVOBJ または SAV) は、オブジェクトとその内容を、テープ、光メディア、または保管ファイルに保管します。
- オブジェクト復元 (RSTOBJ または RST) は、保管バージョンのオブジェクトを、テープ、光メディア、または保管ファイルから復元します。

一部のコマンドでは、一度に処理できるオブジェクトは 1 つのみですが、そのオブジェクトは任意のタイプの IBM i オブジェクトとすることが可能です。例を以下に示します。

- オブジェクト移動 (MOVOBJ または MOV) は、オブジェクトをあるライブラリーまたはディレクトリーから別のライブラリーまたはディレクトリーへ移します。
- オブジェクト名変更 (RNMOBJ または RNM) は、オブジェクトの新しい名前を指定します。

以下の表に、多くのオブジェクト・タイプ上で操作を実行するコマンドをリストします。

表 *1.* 複数のオブジェクト・タイプを処理するコマンド *(*オブジェクトがオブジェクト名、ライブラリー、およびタイプによって認識される場合*)*

対象	処置	識別コード
オブジェクト	ALC、CHK、CPR、CRTDUP、DCP、DLC、DLT、DMP、MOV、 RNM、RST、SAV、SAVCHG、WRK、	OBJ
オブジェクト・アクセス	SET	OBJACC
オブジェクト監査	CHG	OBJAUD
オブジェクト権限	DSP、EDT、GRT、RVK	OBJAUT
オブジェクト記述	CHG、DSP、RTV	OBJD
オブジェクト・ジャーナル処理	CHG、END、STR	JRNOBJ
オブジェクト・ロック	WRK	OBJLCK
オブジェクト所有者	CHG、WRK	OBJOWN
オブジェクト基本グループ	CHG、WRK	OBJPGP

表 *2.* 複数のオブジェクト・タイプを処理するコマンド *(*オブジェクトがパス名によって識別される場合*)*

対象	処置	識別コード
オブジェクト	CPY、MOV、RNM、RST、SAV	適用されない
オブジェクト監査	CHG	AUD
オブジェクト権限	CHG、DSP、WRK	AUT
オブジェクト記述	DSP、WRK	LNK
オブジェクト整合	CHK	OBJITG
オブジェクト・ジャーナル処理	END、STR	JRN
オブジェクト所有者	CHG	OWN
オブジェクト基本グループ	CHG	PGP

関連概念:

53 ページの『IBM i オブジェクトに対して機能する CL コマンド』
IBM i の各オブジェクト・タイプには、そのオブジェクト・タイプを処理する 1 組のコマンドがあります。

関連資料:

89 ページの『OBJTYPE パラメーター』
オブジェクト・タイプ (OBJTYPE) パラメーターは、オブジェクトが指定されたコマンドにより操作できる IBM i オブジェクトのタイプを指定します。

CL プログラムおよびプロシージャー

CL プログラムおよびプロシージャーは、CL コマンドのみで構成されるソース・ステートメントから作成されます。

CL ソース・プログラム は、オリジナル・プログラム・モデル (OPM) プログラムまたは統合言語環境 (ILE) モジュールのいずれかにコンパイルできる一連の CL ソース・ステートメントであり、さらにそれを、CL または他の言語で作成されたモジュールから構成されるプログラムにバインドできます。

CL プログラムとプロシージャーを使用する利点には、次のようなものがあります。

- CL プログラムとプロシージャーを使用すると、コマンドを 1 つずつ入力して実行するよりも処理が速くなります。
- CL プログラムとプロシージャーは、同じセットのコマンドとロジックにより、一貫した処理を提供します。
- 機能によっては、個別に入力することができず、CL プログラムまたはプロシージャーの一部として入力しなければならない CL コマンドが必要な場合があります。
- CL プログラムとプロシージャーは、他の高水準言語 (HLL) プログラムやプロシージャーと同様に、テストしたりデバッグしたりすることができます。
- CL プログラムおよびプロシージャーには、プログラムまたはプロシージャーによって実行される操作を特定の使用条件に合わせるためのパラメーターを渡すことができます。
- 他の ILE 高水準言語で作成されたモジュールと CL モジュールをプログラムにバインドすることもできます。

CL プログラムとプロシージャーは、種々のアプリケーションに使用することができます。例えば、次のような CL プロシージャーの使用方法が可能です。

- 対話式アプリケーションのユーザーに対するインターフェースとして使用することにより、ユーザーは、プログラムまたはプロシージャーで使用されているコマンドの意味を理解していなくても、アプリケーションの機能を要求することができます。これにより、ワークステーション・ユーザーの作業が簡素化されるだけでなく、コマンドの入力時に生じるエラーを最小限に抑えることができます。
- アプリケーションの中で使用する変数 (日付、時刻、外部標識など) を設定し、またアプリケーションに使用するライブラリー・リストを指定することによって、アプリケーションの操作を制御することができます。これにより、アプリケーションがいつ実行されても、所定の操作の実行が保証されます。
- あらかじめ定められている手順をシステム・オペレーターが実行できるようになります。このような手順としては、例えば、サブシステムの開始、ファイルのバックアップ・コピーの作成、あるいは他の操作の実行などがあります。このような手順に CL プログラムとプロシージャーを使用すれば、オペレーターが日常の作業で使用するコマンドの数が少なくなり、また一貫したシステム操作を確実に行うことができます。

システムにより提供される CL コマンドのほとんどは、CL プログラムとプロシージャーの中で使用することができます。また、CL プログラムとプロシージャーによる使用を前提として設計されているコマンドもあり、1 つずつ入力した場合には使用できないコマンドもあります。このようなコマンドには以下のものがあります。

- ロジック制御コマンド。これは、プログラムまたはプロシージャーの実行時点での条件に応じて、プログラムまたはプロシージャーが行う処理を制御するために使用します。例えば、ある条件が生じている場合にはある処理を行い、生じていなければ 別の処理を行う機能がこれに当てはまります。このようなロジック操作には、CL プログラムまたはプロシージャー内での条件付き分岐と無条件分岐の 2 つの機能があります。

- データ操作。これにより、ワークステーション・ユーザーとプログラムまたはプロシージャーとの間の通信が可能になります。このデータ操作により、プログラムまたはプロシージャーとワークステーションとの間での定様式データのやりとりを行うことができ、またデータベースへの限定されたアクセスも可能になります。
- プログラムまたはプロシージャーからディスプレイ装置のユーザーにメッセージを送るコマンド。
- 他のプログラムから送られてくるメッセージを受け取るコマンド。このようなメッセージには、プログラムとプロシージャーとの間の通常の通信に使用されるものや、エラーまたはその他の例外状態を示すものなどがあります。
- 変数およびパラメーターは、プログラムまたはプロシージャー内のコマンド間、またはプログラムとプロシージャーとの間で情報をやりとりするために使用されます。
- 他のプロシージャーを呼び出すコマンド。 (コマンド行またはバッチ・ジョブ・ストリームからは、プロシージャーを呼び出すことができません。)

CL プログラムおよびプロシージャーを使用すれば、各機能にそれぞれ独立したプログラムまたはプロシージャーを設計し、かつプログラムまたはプロシージャーの実行を制御する CL プログラムまたはプロシージャーを備えたアプリケーションを設計することができます。また、1 つのアプリケーションの中に CL プログラムまたはプロシージャーと他の HLL プログラムまたはプロシージャーの両方を含めることもできます。このようなアプリケーションでは、CL プログラムは次のような目的に使用されます。

- アプリケーション内でどのプログラムまたはプロシージャーを実行するかの判別。
- 他の HLL 言語では使用できないシステム機能の実行。
- ユーザーとアプリケーションとの間の対話。

CL プログラムおよびプロシージャーは、アプリケーションのユーザーが、どのような操作を行うかを選択し、それに必要なプロシージャーを実行することができるようにする柔軟性を備えています。

CL プログラムおよびプロシージャーには、プロシージャー、モジュール、プログラム、およびサービス・プログラムという 4 つのタイプがあります。

CL プロシージャー

プロシージャーとは、特定のタスクを実行した後に呼び出し元へ戻る、一組の自己完結型高水準言語ステートメントです。

CL において、プロシージャーは通常、PGM ステートメントで始まり、ENDPGM ステートメントで終わります。

CL モジュール

モジュール とは、統合言語環境 (ILE) コンパイラーを使用して高水準言語ソース・ステートメントをコンパイルした結果得られるオブジェクトです。

CL モジュールは、**制御言語モジュール作成 (CRTCLMOD)** コマンドを使用して CL ソースをコンパイルすることによって作成されます。モジュールは、実行するプログラムにバインドされなければなりません。

CL モジュールは、ユーザー作成のプロシージャーと、CL コンパイラーにより生成されたプログラム入り口プロシージャーの、2 つの部分から成り立っています。他の高水準言語 (HLL) (例えば C) では、複数のユーザーが作成したプロシージャーを 1 つのモジュールに含むことができます。

CL プログラム

制御言語 *(CL)* プログラム は、制御言語コマンドのみで構成されているソース・ステートメントから作成されるプログラムです。オペレーティング・システムでは、ILE プログラムと OPM CL プログラムの 2 つのタイプのプログラムがサポートされます。

ILE プログラムは、ILE (Integrated Language Environment®: 統合言語環境) に準拠した高水準言語で作成されたプログラムです。ILE プログラムは、1 つ以上のモジュールを含む IBM i オブジェクトです。モジュールは、プログラムにバインドされるまでは実行できません。これらのプログラムには、プログラム入り口プロシージャーが必要です。 CL コンパイラーは、それが作成する各モジュールでプログラム入り口プロシージャーを生成します。単一モジュールの ILE プログラムは、バインド **制御言語プログラム作成 (CRTBNDCL)** コマンドを使用して作成できます。 **プログラム作成 (CRTPGM)** コマンドを使用すれば、 ILE CL を含むさまざまな ILE コンパイラーによって生成されたモジュール・オブジェクトを含む ILE プログラムを作成できます。

OPM CL プログラムは、オリジナル・プログラム・モデル (OPM) に準拠したプログラムです。 OPM CL プログラムは、**CL プログラム作成 (CRTCLPGM)** コマンドを使用してソースをコンパイルした結果として生じるオブジェクトです。

関連情報:

ILE の概念

サービス・プログラム

サービス・プログラムとは、1 つ以上のモジュールを含む IBM i オブジェクトのことです。

サービス・プログラムのプロシージャーをまったく必要としないプログラムについては、サービス・プログラムに結合せずに実行することができます。しかし、サービス・プログラムのプロシージャーは、サービス・プログラムをプログラムに結合しない限り実行できません。サービス・プログラム内のプロシージャーを呼び出すために、プロシージャー名をエクスポートしなければなりません。サービス・プログラムは、**サービス・プログラムの作成 (CRTSRVPGM)** コマンドを使用して作成します。

プログラムには入り口点を 1 つだけしか設けられませんが、サービス・プログラムには複数の入り口点を設けることができます。サービス・プログラムは直接呼び出すことはできません。サービス・プログラム内のプロシージャーは、プログラムまたはサービス・プログラム内の他のプロシージャーから呼び出すことができます。

CL パラメーター

パラメーター とは、ユーザー入力を提供したり、コマンドまたはプログラムのアクションを制御するためにコマンドまたはプログラムに渡される値のことです。

パラメーター値

パラメーター値は、コマンドの実行中に使用されるユーザー提供の情報です。

個別の値は、定数値、変数名、式、または値のリストで指定できます。パラメーターには、コマンドにおけるそのパラメーターの定義に従って、上記の値を 1 つ以上指定することができます。複数の値を指定できるパラメーターは、値のリストを含むことができるので、リスト・パラメーター と呼ばれます。

キー・パラメーターおよび定位置パラメーターを持つコマンドでは、値はキーワード形式、定位置形式、または両形式の組み合わせで指定することができます。次のいずれかの条件が当てはまる場合には、パラメーター値を括弧で囲む必要があります。

- 値の前にキーワードがある場合。
- 値が式である場合。
- 値のリストを指定する場合。

注: リストに値を 1 つしか指定しない場合には、括弧は不要です。

関連概念:

45 ページの『CL コマンド・コーディング規則』
ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

関連資料:

41 ページの『CL コマンド・パラメーター』
ほとんどの CL コマンドは、そのコマンドを実行するために使用するオブジェクトおよび値を指定する、1 つ以上のパラメーター を持っています。

定数値:

定数値は、実際の値です。定数のタイプは、文字ストリング (名前、日付、および 16 進数値を含む)、10 進数、および論理です。

定数値は、実際の数値または特定の文字ストリングであり、その値は常に不変です。制御言語では、文字ストリング (引用符付き、または引用符のない文字ストリング)、10 進定数、および論理定数の 3 つのタイプの定数を使用することができます。

文字ストリング:

文字ストリングとは、任意の EBCDIC 文字 (英数字および特殊文字) からなる文字ストリングであり、1 つの値として使用されるものです。

これらの EBCDIC 値には、日付値および 16 進値を含めることができます。文字ストリングには、引用符付きのストリングと引用符のないストリングの 2 つの形式があります。どちらの形式の文字ストリングにも、最大 5000 文字まで使用することができます。

引用符付き 文字ストリングは、英数字と特殊文字からなるストリングを単一引用符で囲んだものです。例えば、'CREDIT LIMIT HAS BEEN EXCEEDED' は、引用符付き文字ストリングです。引用符付き文字ストリングは、引用符のない文字ストリングの中で使用すると無効になる文字データを使用したい場合に使用します。例えば、コマンドには、コマンドの機能を説明するためにユーザー指定のテキストを入れることができます。引用符のないストリングではブランクは使用できないので、その記述の中で複数の語を使用する場合には、記述を単一引用符で囲まなければなりません。

引用符のない 文字ストリングは、英数字と、以下の表の引用符のないストリング の欄に示されている特殊文字のみで構成されるストリングです。この表は、引用符付き文字ストリングおよび引用符のない文字ストリング値で有効な、主な EBCDIC 文字を要約したものです。表の最後の欄の X は、左側に示されている文字が有効であることを意味します。表の後の注を参照すると、その文字が有効である理由が示されています。特殊文字を使用することによって、次のものを引用符のない文字ストリング値にすることができます。

- 事前定義値 (* で始まる)
- 修飾オブジェクト名 (/)
- 総称名 (* で終わる)
- 10 進定数 (+、-、.、および ,)

これらの引用符のないストリングは、いずれも、文字ストリングを受け入れることができるように定義されているパラメーターに指定することができます。また、パラメーターの中には、事前定義値、名前、または 10 進数値 (あるいはこれらを組み合わせたもの) を受け入れるように定義されているものもあります。

表 3. 引用符付き文字ストリングおよび引用符のない文字ストリング

文字の名前	文字	引用符のないストリング	引用符付きストリング
アンパーサンド	&	注 5 を参照	X
単一引用符	'	注 7 を参照	-
アスタリスク (*)	*	注 5 および 6 を参照	X
アットマーク	@	X	X
ブランク			X
コロン	:		X
コンマ	,	注 1 を参照	X
数字	0 から 9	注 1 を参照	X
通貨記号	$	X	X
等しい	=	注 5 および 8 を参照	X
より大きい	>	注 5 および 8 を参照	X
左括弧	(	注 4 を参照	X
より小さい	<	注 5 および 8 を参照	X
英字 (小文字)	a から z	注 2 を参照	X
英字 (大文字)	A から Z	X	X
負符号	-	注 1 および 5 を参照	X
否定	¬	注 5 および 8 を参照	X
数値記号	#	X	X
パーセント	%		X
ピリオド	.	注 1 および 11 を参照	X
プラス	+	注 1 および 5 を参照	X
疑問符	?		X
引用符	" "	注 10 を参照	X
右括弧	)	注 4 を参照	X
セミコロン	;		X
斜線	/	注 3 および 5 を参照	X
下線	_	注 9 を参照	X
縦線	|	注 5 および 8 を参照	X

注:

1. 数字、1 個の小数点 (. または ,) (オプション)、および 1 個の先行符号 (+ または -) (オプション) からなる引用符のない文字ストリングは、引用符のない有効な文字ストリングです。このようなストリングは、コマンド定義におけるパラメーターの属性に応じて、数値として、または文字値として扱われます。CALL コマンドまたは式の中では、このような引用符のないストリングは数値として扱われるため、文字表現が必要な場合には、引用符付きのストリングを使用する必要があります。引用符のないストリングの中では、数字と英数字とを任意に組み合わせて使用することもできます。

2. 引用符のないストリングでは、ストリングが CASE(*MIXED) 属性を持つパラメーターに指定されている場合を除いて、小文字が大文字に変換されます。
3. 斜線は修飾名とパス名の結合子として使用できます。
4. 引用符のないストリングでは、括弧は、キーワードの値およびリストを区切る場合や、式の中で評価の順序を指示するために使用する場合に有効です。
5. 引用符のないストリングでは、文字 +、-、*、/、&、|、¬、<、>、および = は、単独で使用できます。これらの文字は、コマンド定義で EXPR(*NO) 属性が定義されているパラメーターで指定された場合、文字値として扱われます。コマンド定義で EXPR(*YES) 属性が定義されているパラメーターで指定された場合、これらの文字は式の演算子として扱われます。

   CL プロンプターで、パラメーター値の最初のカラムに文字 >、<、または + を入力すると、プロンプターに対して特別な意味を示します。

   - >：パラメーター・リストに項目を挿入します
   - <：パラメーター・リストから項目を除去します
   - +：パラメーター・リストを拡張して値を追加できます

   そのため、コマンドのプロンプトで、コマンド定義に EXPR(*NO) 属性が定義されているパラメーターで >、<、または + が指定されると、パラメーター値の最初にカラム内でない場合に限り、文字値として扱われます。
6. 引用符のないストリングでは、アスタリスクは、その直後に名前がある場合 (事前定義値など) や、直前に名前がある場合 (総称名など) に有効です。
7. 引用符付きのストリングの中で単一引用符を使用すると、開始単一引用符 (区切り文字) と対をなす終了区切り文字と解釈されてしまうので、引用符付きストリング内で区切り文字でない単一引用符を使用したい場合は、単一引用符を 2 重 ('') にしなければなりません。 引用符付き文字ストリング内の文字を数える場合、このような隣接した 2 重の単一引用符は 1 文字として数えられます。
8. 引用符のないストリングでは、<、>、=、¬、および | は、この中の他の文字と組み合わせて使用することができます。有効な組み合わせは、<=、>=、¬=、¬>、¬<、||、|<、および |> です。コマンド定義で EXPR(*NO) 属性が定義されているパラメーターで、このような組み合わせを指定した場合、これは文字の値として扱われます。また、コマンド定義で EXPR(*YES) 属性が定義されているパラメーターで指定した場合には、式の演算子として扱われます。
9. 引用符のないストリングでは、下線は、最初の文字として使用した場合、および単独で使用した場合には無効です。
10. 引用符は、引用符付きの名前を区切るために使用されます。
11. ピリオドは基本名の中で使用できますが、最初の文字には使用できません。

次の表は、引用符付きストリング定数の例を示しています。

表 4. 引用符付きストリング定数

定数	値
'1,2,'	1,2,
'DON''T'	DON'T
'24 12 20'	24 12 20

次の表は、引用符のないストリング定数の例を示しています。

表 5. 引用符のないストリング定数

定数	意味
CHICAGO	CHICAGO
FILE1	FILE1
*LIBL	ライブラリー・リスト
LIBX/PGMA	ライブラリー LIBX 中のプログラム PGMA
1.2	1.2

関連資料:

『日付値』
日付値は、日付を表す文字ストリングです。

『16 進数値』
16 進数値とは、16 進数字 A から F および 0 から 9 を組み合わせて表した定数です。

64 ページの『式』
式 とは、いくつかの定数、変数、または組み込み関数を演算子によって区切ったものであり、1 つの値を生むものです。

109 ページの『文字ストリング式』
文字ストリング式のオペランドは、引用符付きもしくは引用符なしの文字ストリング、CL 文字変数、サブストリング (%SUBSTRING または %SST) 組み込み関数、トリム (%TRIM、%TRIML、または %TRIMR) 組み込み関数、変換 (%CHAR、%LOWER、または %UPPER) 組み込み関数のいずれかでなければなりません。

16 **進数値*:***

16 進数値とは、16 進数字 A から F および 0 から 9 を組み合わせて表した定数です。

名前、日付、および時刻を除くすべての文字ストリングは、16 進形式で指定できます。16 進数値を指定するには、X の後に、長さが偶数の数値を単一引用符で囲んで指定します。例えば、X'F6' および X'A3FE' のようにします。

注**:** 00 から 3F の範囲の 16 進数値、または 16 進数値 FF を入力する場合には、注意が必要です。これらの文字が表示または印刷された場合、装置制御文字として処理され、予測できない結果が生じることがあります。

関連資料:

58 ページの『文字ストリング』
文字ストリングとは、任意の EBCDIC 文字 (英数字および特殊文字) からなる文字ストリングであり、1 つの値として使用されるものです。

日付値*:*

日付値は、日付を表す文字ストリングです。

日付値の形式は、システム値 QDATFMT によって指定します。日付の値の長さは、使用する形式と、区切り文字を使用するかどうかによって異なります。例えば、区切り文字を使用しない場合には、年間通算日形式の日付の長さは 5 文字であり、年間通算日以外の形式では 6 文字です。区切り文字を使用した場合には、これより長くなります。

システム値 QDATSEP は、日付の入力の際に使用できるオプションの区切り文字を指定します。区切り文字を使用する場合には、日付を単一引用符で囲まなければなりません。

日付値は、タイプ *DATE のパラメーターで使用できます。40 以上の年の値は、1940 年から 1999 年を表します。40 未満の年の値は、2000 年から 2039 年を表します。

関連資料:

58 ページの『文字ストリング』
文字ストリングとは、任意の EBCDIC 文字 (英数字および特殊文字) からなる文字ストリングであり、1 つの値として使用されるものです。

関連情報:

システム値

日付区切り記号 (QDATSEP) システム値

10 進数値***:***

10 進数値は、1 つ以上の数字からなる数字ストリングであり、必要に応じて、その前に正符号 (+) または負符号 (-) を付けることができます。

10 進数値は最高 15 桁まで可能であり、小数点 (コンマまたはピリオド) 以下は 9 桁まで使用することができます。したがって、10 進数値は、正符号または負符号と小数点 (ある場合) を含めて、17 桁を超えることはできません。10 進数値の例を以下に示します。

論理値***:***

論理値とは、シングル文字 (1 または 0) を単一引用符で囲んだものです。

論理値は、オンまたはオフ、YES または NO、および真または偽などの状態を表すスイッチとしてよく使われます。論理値を式の中で使用する場合には、必要に応じて、論理値の前に *NOT または ¬ を付けることができます。次に示すのは論理値の例です。

表 *6.* 論理値

定数	値	意味
'0'	0	オフ、NO、または偽
'1'	1	オン、YES、または真

浮動小数点定数***:***

浮動小数点定数は、数値定数の表記であり、オプションの符号として (末尾の) 1 つ以上の桁および小数点の前に表示されます。

この表記は次のものから構成されます。

- 仮数符号: 仮数有効符号としては、+ または - を使用できます。仮数符号はオプションであり、符号を指定しなかった場合は + であると見なされます。
- 仮数: 仮数には小数点が含まれていなければなりません。仮数として指定できる最大長は 253 桁ですが、演算に使用されるのは最初の 17 桁分の有効数字のみです。

- 指数文字: 指数文字は E でなければなりません。
- 指数符号: 指数符号は + または - でなければなりません。仮数符号はオプションであり、符号を指定しなかった場合は + であると見なされます。
- 指数: 指数は整数でなければなりません。使用できる数字は 0 から 9 までです。指定できる数字は最大で 3 桁です。

浮動小数点定数は、すべて倍精度の値として保管されます。浮動小数点定数の中で、各構成要素の間にブランクを含めることはできません。また、各構成要素の順序は上記のとおりでなければなりません。

浮動小数点定数を指定することのできるパラメーターを持つコマンドは、次のとおりです。

- **プログラム呼び出し (CALL)** コマンドまたは**プロシージャー呼び出し (CALLPRC)** コマンド: PARM パラメーターを使用して、呼び出されるプログラムに浮動小数点定数を渡すことができます。呼び出されるプログラムは、浮動小数点定数を倍精度で受け取る必要があります。
- **プログラム変数変更 (CHGPGMVAR)** コマンド: VALUE パラメーターを使用して、プログラムの中の浮動小数点変数を変更できます。
- ファイル・コピー (CPYF) コマンド: データベース・ファイルからコピーするレコードを選択するために、FROMKEY、TOKEY、および INCREL パラメーターで浮動小数点構造を使用できます。

関連情報:

DDS

変数名**:**

変数には、プログラムの実行中に変更できるデータの値が含まれています。 変数名は、値を含む変数の名前です。

変数は、コマンドの実行時点でその変数に含まれている値を渡すために、コマンドの中で使用されます。値が変化するのは、データ域、表示装置ファイル・フィールド、またはメッセージから値を受け取った場合、値がパラメーターとして渡された場合、プログラムの中で**変数変更 (CHGVAR)** コマンドが実行された場合、あるいは呼び出された別のプログラムが戻る前にその値を変更した場合です。

変数には、文字ストリング変数 (名前を含む)、10 進変数、論理変数、および整数の 4 つのタイプがあります。10 進変数および論理変数の値は、パラメーターとして求められている値のタイプに一致していなければなりません。文字変数にはどのようなタイプの値でも指定することができます。例えば、10 進数値が求められている場合には、10 進変数と同様に、文字変数によってもその値を指定することができます。

変数名は使用される値を識別するものであり、実際のデータがどこにあるのかを示します。CL 変数は CL プログラムの中でのみ有効なものなので、*CL* プログラム変数、あるいは単に CL 変数と呼ばれます。CL 変数名はアンパーサンド (&) で始めなければなりません。

CL コマンドのほとんどすべてのパラメーターは、CL 変数によって指定できます。パラメーターに CL 変数を指定した場合、そのパラメーターを含むコマンドが実行される際には、その時点におけるその変数の値が使用されます。つまり、変数の値は、ユーザーがその値を定数として指定した場合と同じように扱われます。

一般に CL プログラムの中のコマンドのほとんどのパラメーターに、CL 変数を使用できるので、たいていの場合、コマンドの説明の中では CL 変数については触れていません。定数のみに限定されるパラメーター (DCL コマンドのパラメーター)、CL 変数のみに限定されるパラメーター (**ジョブ属性検索 (RTVJOBA)** コマンドのすべてのパラメーター)、または特定のタイプの変数のみに限定されるパラメーター (例えば、RTVJOBA または **メッセージ検索 (RTVMSG)** コマンドのパラメーター) については、該当のパラ

メーターの説明の中でそのつどその制約が示されています。このような場合以外は、CL プログラムの中で使用できるコマンドでは、事前定義値のみを取るパラメーターも含めて、パラメーター値の代わりに CL 変数を使用することができます。例えば、*YES または *NO という事前定義値のみを取る KEEP パラメーターに、*YES または *NO の代わりに CL 変数を指定することができます。この場合、このパラメーターの値は、コマンドの実行時点での CL 変数の値に応じて、*YES または *NO になります。

CL 変数には 1 つしか値を入れることができません。すなわち、ブランクで区切った値のリストを入れることはできません。

CL プログラム変数の値は、次のタイプのいずれかとして定義することができます。

- 文字: 最高で 9999 文字を含むことができる文字ストリング。文字ストリングは、引用符付きまたは引用符なしのいずれの形式でも指定することができますが、変数に保管されるのはストリング自体に入っている文字のみです。
- 10 進数値: 最高で 15 桁 (小数部分は最高 9 桁) を持つことができるパック 10 進数。
- 論理値: オン/オフ、真/偽、または YES/NO を表す '1' または '0' の論理値。
- 整数値: 符号付き (正または負の値) または符号なし (常に正の値) のいずれかの 2 バイトまたは 4 バイトの 2 進整数値。

表 7. CL プログラム変数

値	宣言可能な **CL** 変数
名前	文字
日付または時刻	文字
文字ストリング	文字
数字	10 進数、整数値、または文字
論理値	論理値または文字

関連概念:

136 ページの『固有名に関するその他の規則』
オブジェクトの命名については、特殊文字 (付加文字) に関して以下のような追加の規則があります。

式:

式 とは、いくつかの定数、変数、または組み込み関数を演算子によって区切ったものであり、1 つの値を生むものです。

演算子は、値をどのように組み合わせて単一の値すなわち、結果を導き出すかを指定します。式のタイプには、算術、文字ストリング、比較、および論理があります。式は、CL ソース・プログラム中でのみ、CL コマンド・パラメーターの値として使用できます。

演算子には、算術演算子、文字ストリング演算子、関係演算子、および論理演算子があります。定数または変数は、文字、10 進数、整数、または論理にすることができます。例えば、式 (&A + 1) は、変数 &A の値に 1 を加えた結果を式の代わりに使用することを指定します。

| 文字ストリング式は、CL プログラムの中で EXPR(*YES) が定義されている特定のコマンド・パラメーターの中で使用することができます。式には、%BINARY (または %BIN)、 %CHAR、%CHECK、%CHECKR、%DEC、%INT、%LEN、%LOWER、%PARMS、%SCAN、%SIZE、%SUBSTRING (または %SST)、 %SWITCH、%TRIM、%TRIML、%TRIMR、%UINT (または %UNS)、および %UPPER という組み込み関数を含めることができます。 108 ページの『CL コマンドで使用する式』には、式のタイプとそれぞれの例を記載しています。

関連概念:

106 ページの『記号演算子』
さまざまな文字が、CL コマンドで記号演算子として使用できます。

関連資料:

58 ページの『文字ストリング』
文字ストリングとは、任意の EBCDIC 文字 (英数字および特殊文字) からなる文字ストリングであり、1 つの値として使用されるものです。

108 ページの『CL コマンドで使用する式』
文字ストリング式は、コマンド定義オブジェクトで EXPR(*YES) が定義されているどのようなパラメーター、要素、または修飾子にも使用することができます。

値のリスト:

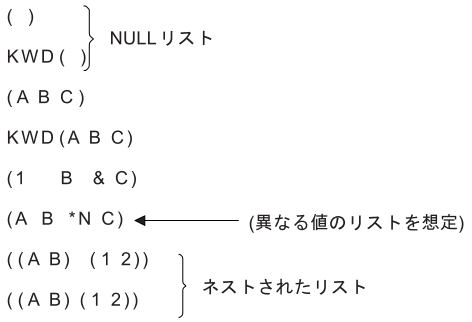
値のリストとは、1 つのパラメーターで指定できる 1 つ以上の値のことです。

すべてのパラメーターが、このようなリストを受け入れるわけではありません。リスト・パラメーター を定義することで、1 つ以上のタイプからなる複数の値の具体的な組み合わせを受け入れることができます。リスト中の値は、1 つ以上のブランクで区切らなければなりません。値のリストはそれぞれ、1 つのパラメーターとして扱うことを指示するために、全体を括弧で囲みます。定位置形式でパラメーターを指定する場合にも括弧を使用します。あるパラメーターについてリストを指定することができるかどうか、およびどのようなリストを指定することができるかについては、該当するコマンドの項の該当するパラメーターの説明を参照してください。

リスト・パラメーターは、複数個の同種の値のリスト (単純リスト)、または複数個の異種の値のリスト (混合リスト) を受け入れるように定義することができます。どちらのタイプのリストの場合も、リスト中の各値をリスト要素 と呼びます。リスト要素には、定数、変数、または他のリストを使用できますが、式は使用できません。

- 単純リスト・パラメーターは、パラメーターで受け入れられるタイプの 1 つ以上の値を受け入れます。例えば、(RSMITH BJONES TBROWN) は 3 つのユーザー名の単純リストです。
- 混合リスト・パラメーターは、別個に定義された一定の値のセット (特定の順序になっているもの) を受け入れます。 各値は、タイプや範囲など、特定の特性を持つものとして定義することができます。例えば、LEN(5 2) は混合リストであり、最初の要素 (5) はフィールドの長さを示し、2 番目の要素 (2) はそのフィールドの小数部分の長さを示しています。
- リストを受け入れるように定義された多くのパラメーターには、値のリストの代わりにいくつかの事前に定義された単一値を指定することができます。これらの単一値の 1 つはデフォルト値とすることができます。デフォルト値は、単純リストまたは混合リストにおいて、リストを指定しない場合に指定できる (または想定される) 値です。特定のリスト・パラメーターについてどのようなデフォルト値が受け入れられるかについては、そのパラメーターが定義され使用されるコマンドの項の該当するパラメーターの説明を参照してください。

  注: *N は単純リストには指定できませんが、混合リストには指定することができます。また、CALL コマンドおよび CALLPRC コマンドで渡される個々のパラメーターにリストを指定することはできません。
- リスト内リストのネストのレベルは、最初のレベルも含めて最高 3 レベルです。これは 3 つの括弧のネスト・レベルで示されます。



以下に、リストの例を示します。

最後の 2 つの例には、1 つのリストの中にネストされた 2 つのリストが含まれています。最初のリストには A および B という値、2 番目のリストには 1 および 2 という値が含まれています。ネストされた 2 つのリストの間にはスペースは不要です。ネストされた各リストの値の区切り文字にはブランクが使用されます。また、1 組の括弧を使用してネストされた値をグループ化することで、より大きなリストとして示しています。

関連資料:

41 ページの『CL コマンド・パラメーター』
ほとんどの CL コマンドは、そのコマンドを実行するために使用するオブジェクトおよび値を指定する、1 つ以上のパラメーター を持っています。

キーワード形式および定位置形式のパラメーター

CL におけるパラメーターは、キーワード形式、定位置形式、あるいはこの両方の組み合わせのいずれかを使用して指定できます。

キーワード形式のパラメーター

キーワード形式 のパラメーターの場合には、キーワードのすぐ後に、1 つの値 (またはブランクで区切った複数の値) を括弧で囲んで入力します。 キーワードと、パラメーター値の前の左括弧の間にブランクを使用することはできません。括弧とパラメーター値との間にはブランクがあっても構いません。例えば、LIB(MYLIB) は、ライブラリー名に MYLIB を指定するキーワード・パラメーターです。どのように使用されるかは、この LIB パラメーターを使用するコマンドによって異なります。

1 つのコマンドのパラメーターをすべてキーワード形式で指定する場合には、パラメーターはどのような順序で指定しても構いません。例えば、次の 2 つのコマンドは同じものであると見なされます。

```
CRTLIB   LIB(MYLIB)  TYPE(*TEST)
CRTLIB   TYPE(*TEST)  LIB(MYLIB)
```

定位置形式のパラメーター

定位置形式 のパラメーターの場合は、キーワードをコーディングせず、値 (またはリストの場合は複数の値) のみを指定します。指定した値の働きは、そのコマンドに指定するパラメーター・セットでのその値の位置によって決まります。コマンド名とパラメーター値との間、およびパラメーター値とパラメーター値との間は、1 つ以上のブランクで区切ります。コーディングできるパラメーターの定位置順序は 1 つのみであり、上記の CRTLIB の例の定位置形式は、以下のとおりです。

```
CRTLIB   MYLIB  *TEST
```

あるパラメーターの値を入力したくない場合は、そのパラメーターの位置に事前定義値 *N (ヌル値) を入力できます。システムは、*N をパラメーターの省略と解釈し、それに対してデフォルト値を割り当てるか、あるいはそのままヌルにしておきます。前記の CRTLIB コマンドの 2 番目の例で、TYPE パラメー

ターに *TEST ではなく *N をコーディングしたとすれば、コマンドの実行時にはデフォルト値である *PROD が使用され、MYLIB という名前の実働ライブラリーが作成されます。各パラメーターの説明は、CRTLIB コマンドの項を参照してください。

注:

- 定位置形式のコーディング限度を超えて、パラメーターを定位置形式でコーディングすることはできません。この限界点を超えて、定位置形式でパラメーターをコーディングしようとすると、システムはエラー・メッセージを戻します。
- CL プログラム・ソースで定位置形式を使用すると、プログラムの作成時に時間を節約できますが、ユーザーや他のユーザーによる保守がより困難になります。キーワード形式を使用して作成したコマンドは、通常理解しやすく、簡単に拡張できます。

キーワード形式と定位置形式の組み合わせ

1 つの CL コマンドで、キーワード形式と定位置形式の両方を用いてパラメーターをコーディングすることもできます。次の例では、CL 変数宣言 (DCL) コマンドの 3 通りのコーディング方法を示します。

キーワード形式:

```
DCL   VAR(&QTY)  TYPE(*DEC)  LEN(5)  VALUE(0)
```

定位置形式:

```
DCL   &QTY  *DEC  5  0
```

定位置形式とキーワード形式の組み合わせ:

```
DCL   &QTY  *DEC  VALUE(0)
```

最後の例では、オプションの LEN パラメーターをコーディングしなかったため、VALUE パラメーターはキーワード形式でコーディングしなければなりません。

注: キーワード形式で指定したパラメーターより後に、定位置形式でパラメーターを指定することはできません。

関連概念:

45 ページの『CL コマンド・コーディング規則』
ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

関連資料:

41 ページの『CL コマンド・パラメーター』
ほとんどの CL コマンドは、そのコマンドを実行するために使用するオブジェクトおよび値を指定する、1 つ以上のパラメーター を持っています。

必須パラメーター、オプション・パラメーター、およびキー・パラメーター

CL コマンドのパラメーターには、必ず指定しなければならないパラメーター (必須パラメーター) と、必ずしも指定を必要としないパラメーター (オプション・パラメーター) があります。

オプション・パラメーターに対してコマンドの入力時にパラメーター値を指定しなかった場合、通常は、システム定義のデフォルト値が割り当てられます。

コマンドには、キー・パラメーター が存在することもあります。ユーザーがコマンドのプロンプト画面を表示した場合に画面に表示されるのは、キー・パラメーターのみです。キー・パラメーターに値を入力する

と、残りのパラメーターが表示されます。このとき、残りのパラメーターには、デフォルト値 (*SAME、*PRV など) の代わりに実際の値が入っています。

関連資料:

41 ページの『CL コマンド・パラメーター』
ほとんどの CL コマンドは、そのコマンドを実行するために使用するオブジェクトおよび値を指定する、1 つ以上のパラメーター を持っています。

共通に使用されるパラメーター

一部のパラメーターは、多数の CL コマンドで共通して使用されます。

このような共通に使用されるパラメーターは、以下の基準の 1 つまたは両方を満たします。

- 使用形態について広範な情報を持つパラメーター。
- 多くの CL コマンドで使用されるパラメーター (例えば AUT パラメーター) で、個々のコマンドのパラメーターの説明では基本的な情報のみが簡潔に示されているもの。

ここでは、上記に該当するコマンド・パラメーターについて補足的な説明を行います。この理由は次のとおりです。

- 個々のコマンドの項におけるパラメーターの説明を簡潔にするため。パラメーターの主要機能をよく知っているプログラマーの方には、通常、詳細な説明は不要と考えられます。
- 場合によってはプログラマーの方にとって役に立つと思われる補足的な情報を提供するため。

この情報は、簡単に参照できる形式になっており、各パラメーターの全般的説明には、その機能についての説明、使用上の規則、その他の役立つ情報が記載されています。各パラメーターに指定できる値も列記してあります。各値については、その意味に加えて、それがどのコマンドで使用できるかも示しています。個々のコマンドでのすべての値が示されているわけではありません。個々のコマンドのパラメーターに指定する値の特定の用法については、該当するコマンドの項を参照してください。

関連概念:

45 ページの『CL コマンド・コーディング規則』
ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

AUT パラメーター**:**

権限 (AUT) パラメーターは、作成コマンド、認可コマンド、および取り消しコマンドで使用します。このパラメーターは、あるオブジェクトのすべてのユーザーに認可する権限を指定します。

AUT パラメーターは、オブジェクトの保護のために使用する権限リストも指定します。AUT パラメーターによって権限リストを組み込むことができるオブジェクト・タイプは、LIB、PGM、DTADCT、および FILE の 4 つです。共通権限は、IBM i オブジェクト属性の 1 つであり、システムへのアクセス権を備えたすべてのユーザーの、そのオブジェクトに対する基本的な権限の集合を制御するものです。これらの権限は、特定のユーザーについて拡大することも縮小することもできます。権限リストを指定した場合には、権限リストの共通権限が、そのオブジェクトについての共通権限になります。オブジェクトの所有者は、作成時にはそのオブジェクトに対するすべての権限を持っています。

オブジェクトを専用オブジェクトとして作成した場合、または全ユーザーに与える権限を制限して作成した場合には、所有者は、**オブジェクト権限認可 (GRTOBJAUT)** コマンドに、特定のユーザーの名前および権限を指定することによって、特定のユーザーに認可する権限を拡大したり縮小したりすることができます。また、所有者は、 **オブジェクト権限取り消し (RVKOBJAUT)** コマンド、または**オブジェクト権限編集**

(EDTOBJAUT) コマンドを使用することにより、特定のユーザーまたは (共通権限または明示指定された権限、あるいはその両方を持つ) 全ユーザーについて、特定の権限を取り消すこともできます。

指定可能な値

***LIBCRTAUT**
オブジェクトの共通権限は、ターゲット・ライブラリー (オブジェクトを含むライブラリー) の CRTAUT パラメーターの値から取得されます。共通権限は、オブジェクトの作成時に決まります。オブジェクトの作成後にライブラリーの CRTAUT 値を変更しても、その新しい値は既存のオブジェクトには影響しません。

***USE** ユーザーは、オブジェクトに対して、プログラムの実行やファイルの読み取りなどの基本操作を実行することができます。ユーザーは、オブジェクトを変更することはできません。*USE 権限には、オブジェクト操作権、読み取り権限、および実行権限があります。

***CHANGE**
ユーザーは、所有者限定の操作、またはオブジェクト存在権限とオブジェクト管理権によって制御される操作を除き、すべての操作をオブジェクトに対して実行できます。ユーザーは、オブジェクトの基本機能を変更および実行できます。変更権限には、オブジェクト操作権とすべてのデータ権限があります。

***ALL** ユーザーはすべての操作を実行できます。ただし、所有者限定の操作や、権限リスト管理権によって制御されている操作は実行できません。ユーザーは、オブジェクトの存在の制御、オブジェクトのセキュリティーの指定、オブジェクトの変更、およびオブジェクトに対する基本機能の実行を行うことができます。オブジェクトの所有権を変更することもできます。

***EXCLUDE**
ユーザーは、オブジェクトにアクセスすることができません。

***EXECUTE**
ユーザーは、プログラムまたはプロシージャーの実行、あるいはライブラリーまたはディレクトリーの検索を行うことができます。

authorization-list-name
使用する権限が入っている権限リストの名前を指定してください。

関連情報:

セキュリティー参照

CLS パラメーター**:**

クラス (CLS) パラメーターは、ジョブの実行時環境を定義する属性を識別します。

各クラスでは、次の属性が定義されます。

- 実行優先順位: クラスを使用して実行するすべてのジョブに割り当てる優先順位レベルを指定する数値。優先順位レベルは、システム資源を争奪するすべてのジョブのうちで、次に実行すべきジョブを決定するために使用されます。指定可能な値は 1 から 99 までであり、1 が最高優先順位です (優先順位 1 のすべてのジョブが最初に実行されます)。
- タイム・スライス: ジョブが開始可能になったときに、そのジョブを実行するのにシステムが認める最大プロセッサー時間。このタイム・スライスには、このジョブが意味ある量の作業を完了するのに必要とする時間を指定します (システムが補助記憶域にアクセスするために要する時間は、タイム・スライスには含まれません)。タイム・スライスが経過すると、そのジョブは待機状態になり、待ち行列内の同

じ優先順位または高い優先順位を持つ他のジョブが実行されている間、(それらのジョブのタイム・スライスに指定された時間が経過するまで) 待機します。そのあとで、待機していたそのジョブに別のタイム・スライスが与えられます。

- 除去値: ジョブが、処理の続行ができずに資源の割り当てを待っている間、またはタイム・スライスを使い果たし、待機中のほかの同位または高位のジョブに処理を譲らなければならない場合に、そのジョブ・ステップを主記憶域から補助記憶域に移すことができるかどうかを示します。
- デフォルトの待ち時間: 待ち状態を引き起こした命令が完了するのをシステムが待つ場合の、デフォルト設定の待ち時間。この待ち時間は、命令がシステム活動を待っている時間に適用されるもので、命令がユーザーからの応答を待っている時間には適用されません。通常、待ち時間とは、ユーザーが要求を取り消さずにシステムの処置を待っている時間のことです。待ち時間を超過すると、該当のジョブに対してエラー・メッセージが渡されます。このデフォルトの待ち時間は、待ち状態を生じさせる CL コマンドに待ち時間が指定されていない場合にのみ、適用されます。

  ファイル資源の割り振りに使用される待ち時間は、ファイル記述に指定しますが、指定変更コマンドを使用して指定変更することができます。指定変更コマンドは、クラス・オブジェクトに指定された待ち時間を使用することを指定します。ファイルがオープンされたとき、ファイル資源が使用可能でなかった場合は、システムは、待ち時間が経過するまではファイル資源の割り振りを待ちます。

  注**:** クラス属性は、ジョブの各経路指定ステップに適用されます。ほとんどのジョブでは経路指定ステップは 1 つのみですが、(**リモート・ジョブ**・コマンドまたは **ジョブ転送**コマンドなどのために) ジョブの経路が変更された場合には、クラス属性はリセットされます。
- 最大 CPU 時間: ジョブの経路指定ステップの実行を完了するために使用できる、プロセッサー時間の最大許容限度 (個々のタイム・スライスが異なる場合は全タイム・スライスの合計、各タイム・スライスが同じである場合はタイム・スライス時間×タイム・スライス数)。この時間内にジョブの経路指定ステップが完了しなかった場合には、その経路指定ステップは打ち切られ、メッセージがジョブ・ログに書き込まれます。
- 最大一時記憶域: ジョブの経路指定ステップで使用できる一時記憶域の最大量。一時記憶域は、ジョブで実行されるプログラム、ジョブをサポートするために使用されるシステム・オブジェクト、およびジョブが作成した一時オブジェクトを入れるために使用されます。

システムは、いくつかのジョブ処理環境の属性を定義する 1 組のクラスとともに出荷されます。ユーザーは、**クラス作成 (CRTCLS)** コマンドを使用して独自のクラスを作成することができます。また、どのクラスも、**クラス表示 (DSPCLS)** コマンドを用いて表示し、 **クラス削除 (DLTCLS)** コマンドを用いて削除することができます。

指定可能な値

修飾クラス名
: そのクラスを保管するライブラリーの名前です。クラス名は、必要に応じて修飾されます。クラスの名前が修飾されておらず、CRTCLS コマンドに CLS パラメーターが指定されている場合には、クラス・オブジェクトは *CURLIB に保管されます。そうでない場合には、ライブラリー・リスト (*LIBL) を使用してクラス名が探されます。

クラス

システムで提供されるクラス (名前別) は次のとおりです。

QGPL/QBATCH
: バッチ・ジョブ用

QSYS/QCTL
制御サブシステム用

QGPL/QINTER
対話式ジョブ用

QGPL/QPGMR
プログラミング・サブシステム用

QGPL/QSPL
スプーリング・サブシステムの印刷装置書出プログラム用

QGPL/QSPL2
基本システム・プールでの一般スプーリング用

COUNTRY パラメーター**:**

COUNTRY パラメーターは、国コードまたは地域コードを指定します。

国コードまたは地域コードは、X.400 O/R 名の構成要素です。ISO 3166 Alpha-2 コードまたは ITU-T の国または地域のコードを指定することができます。(ITU-T の国または地域のコードは、ITU-T (正式には CCITT) 勧告 X.121 (09/92)、「公衆データ網のための国際番号計画」に記載されている、データ国別または地域別コードあるいは市外局番です。次の表に、指定可能な国コードまたは地域コードのリストを示します。

指定可能な値

***NONE**
国コードまたは地域コードを指定しません。

国コード
次の表を見て、ISO 3166 Alpha-2 コード、または CCITT (ITU-2 とも呼ばれる) 国コードを指定してください。

表 *8. ISO X.400* 国コードまたは地域コード

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
アフガニスタン	AF	412
アルバニア	AL	276
アルジェリア	DZ	603
米領サモア	AS	544
アンドラ	AD	
アンゴラ	AO	631
アンギラ島	AI	
南極大陸	AQ	
アンティグア・バーブーダ	AG	344
アルゼンチン	AR	722
アルメニア	AM	283
アルバ島	AW	362
オーストラリア	AU	505
オーストリア	AT	232
アゼルバイジャン	AZ	400

表 *8. ISO X.400* 国コードまたは地域コード *(*続き*)*

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
バハマ	BS	364
バーレーン	BH	426
バングラデシュ	BD	470
バルバドス	BB	342
ベラルーシ	BY	257
ベルギー	BE	206
ベリーズ	BZ	702
ベナン	BJ	616
バーミューダ	BM	350
ブータン	BT	
ボリビア	BO	736
ボスニア・ヘルツェゴビナ	BA	
ボツワナ	BW	652
ブーベ島	BV	
ブラジル	BR	724
英領インド洋植民地	IO	
ブルネイ・ダルサラーム	BN	528
ブルガリア	BG	284
ブルキナファソ	BF	613
ブルンジ	BI	642
カンボジア	KH	456
カメルーン	CM	624
カナダ	CA	302、303
カーボベルデ	CV	625
ケイマン諸島	KY	346
中央アフリカ共和国	CF	623
チャド	TD	622
チリ	CL	730
中国	CN	460
クリスマス島	CX	
ココス (キーリング) 諸島	CC	
コロンビア	CO	732
コモロ諸島	KM	654
コンゴ	CG	629
クック諸島	CK	548
コスタリカ	CR	712
コートジボアール	CI	612
クロアチア	HR	
キューバ	CU	368
キプロス	CY	280

表 *8. ISO X.400* 国コードまたは地域コード (続き)

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
チェコ共和国	CZ	230
デンマーク	DK	238
ジブチ	DJ	638
ドミニカ国	DM	366
ドミニカ共和国	DO	370
東チモール	TP	
エクアドル	EC	740
エジプト	EG	602
エルサルバドル	SV	706
赤道ギニア	GQ	627
エリトリア	ER	
エストニア	EE	248
エチオピア	ET	636
フォークランド (マルビナス) 諸島	FK	
フェロー諸島	FO	288
フィジー	FJ	542
フィンランド	FI	244
フランス	FR	208、209
フランス本国	FX	
仏領アンティル諸島		340
仏領ギアナ	GF	742
仏領ポリネシア	PF	547
仏領極南諸島	TF	
ガボン	GA	628
ガンビア	GM	607
グルジア	GE	282
ドイツ	DE	262 から 265
ガーナ	GH	620
ジブラルタル	GI	266
ギリシャ	GR	202
グリーンランド	GL	290
グレナダ	GD	352
グアドループ島	GP	
グアム島	GU	535
グアテマラ	GT	704
ギニア	GN	611
ギニアビサオ	GW	632
ガイアナ	GY	738
ハイチ	HT	372

表 *8. ISO X.400* 国コードまたは地域コード *(*続き*)*

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
ハード・アンド・マクドナルド・アイランズ	HM	
ホンジュラス	HN	708
中国 (香港特別行政区)	HK	453、454
ハンガリー	HU	216
アイスランド	IS	274
インド	IN	404
インドネシア	ID	510
イラン	IR	432
イラク	IQ	418
アイルランド	IE	272
イスラエル	IL	425
イタリア	IT	222
ジャマイカ	JM	338
日本	JP	440 から 443
ヨルダン	JO	416
カザフスタン	KZ	401
ケニア	KE	639
キリバス	KI	545
大韓民国、朝鮮民主主義人民共和国	KP	467
大韓民国	KR	450、480、481
クウェート	KW	419
キルギスタン	KG	437
ラオス	LA	457
ラトビア	LV	247
レバノン	LB	415
レソト	LS	651
リベリア	LR	618
リビア・アラブ国	LY	606
リヒテンシュタイン	LI	
リトアニア	LT	246
ルクセンブルグ	LU	270
中国 (マカオ特別行政区)	MO	455
マケドニア [2]	MK [2]	
マダガスカル	MG	646
マラウィ	MW	650
マレーシア	MY	502
モルジブ	MV	472
マリ	ML	610
マルタ	MT	278

表 *8. ISO X.400* 国コードまたは地域コード *(*続き*)*

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
マーシャル諸島	MH	
マルチニーク島	MQ	
モーリタニア	MR	609
モーリシャス	MU	617
マヨット島	YT	
メキシコ	MX	334
ミクロネシア	FM	550
モルドバ	MD	259
モナコ	MC	212
モンゴル	MN	428
モンテネグロ [2]	ME [2]	
モントセラト島	MS	354
モロッコ	MA	604
モザンビーク	MZ	643
ミャンマー	MM	414
ナミビア	NA	649
ナウル	NR	536
ネパール	NP	429
オランダ	NL	204、205
オランダ領アンティル諸島	AN	362
ニューカレドニア	NC	546
ニュージーランド	NZ	530
ニカラグア	NI	710
ニジェール	NE	614
ナイジェリア	NG	621
ニウエ島	NU	
ノーフォーク島	NF	
北マリアナ諸島	MP	534
ノルウェー	NO	242
オマーン	OM	422
パキスタン	PK	410
パラオ	PW	
パナマ	PA	714
パプアニューギニア	PG	537
パラグアイ	PY	744
ペルー	PE	716
フィリピン	PH	515
ピトケアン島	PN	
ポーランド	PL	260
ポルトガル	PT	268

表 *8. ISO X.400* 国コードまたは地域コード *(*続き*)*

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
プエルトリコ	PR	330
カタール	QA	427
レユニオン島	RE	647
ルーマニア	RO	226
ロシア連邦	RU	250、251
ルワンダ	RW	635
セントヘレナ島	SH	
セントクリストファー・ネイビス	KN	356
セントルシア	LC	358
サンピエール・エ・ミクロン島	PM	308
セントビンセントおよびグレナディーンズ諸島	VC	360
サモア (西)	WS	549
サンマリノ	SM	292
サントメ・プリンシペ	ST	626
サウジアラビア	SA	420
セネガル	SN	608
セルビア [2]	SP [2]	
セイシェル	SC	633
シエラレオネ	SL	619
シンガポール	SG	525
スロバキア	SK	
スロベニア	SI	
ソロモン諸島	SB	540
ソマリア	SO	637
南アフリカ	ZA	655
南ジョージア島・南サンドイッチ諸島	GS	
スペイン	ES	214
スリランカ	LK	413
スーダン	SD	634
スリナム	SR	746
スバールバル諸島・ヤンマイエン島	SJ	
スワジランド	SZ	653
スウェーデン	SE	240
スイス	CH	228
シリア・アラブ共和国	SY	417
台湾	TW	466
タジキスタン	TJ	436
タンザニア連合共和国	TZ	640
タイ	TH	520

表 *8. ISO X.400* 国コードまたは地域コード (続き)

国または地域	**ISO 3166 Alpha-2** コード	**ITU-T** [1] 国コードまたは地域コード
トーゴ	TG	615
トケラウ諸島	TK	
トンガ	TO	539
トリニダード・トバゴ	TT	374
チュニジア	TN	605
トルコ	TR	286
トルクメニスタン	TM	438
タークス諸島・カイコス諸島	TC	376
ツバル	TV	
ウガンダ	UG	641
ウクライナ	UA	255
アラブ首長国連邦	AE	424、430、431
英国	GB	234、235、236、237
アメリカ合衆国	US	310 から 316
アメリカ合衆国統治領諸島	UM	
ウルグアイ	UY	748
ウズベキスタン	UZ	434
バヌアツ	VU	541
バチカン市国	VA	225
ベネズエラ	VE	734
ベトナム	VN	452
バージン諸島 (英領)	VG	348
バージン諸島 (米領)	VI	332
ウォリス・フトーナ諸島	WF	543
西サハラ	EH	
イエメン	YE	421、423
旧ユーゴスラビア領	YU	220
ザイール	ZR	630
ザンビア	ZM	645
ジンバブエ	ZW	648

注:

1. 国際電気通信連合 (ITU) 委員会の前身は、国際電信電話諮問委員会 (CCITT) です。
2. 出版時に、これらの国または地域の ISO 3166 Alpha-2 コードを確認することができませんでした。 このコードを使用する前に、最新の ISO 3166 標準を確認してください。

FILETYPE パラメーター**:**

FILETYPE パラメーターは、データベース・ファイル記述が、データ・レコードとソース・レコードのどちらを記述するのかを指定します。

さらに、FILETYPE パラメーターは、作成するデータベース・ファイルの各メンバーにデータ・レコードを入れるのか、ソース・レコード (ステートメント) を入れるのかも指定します。例えば、ファイルには、RPG プログラム用の RPG ソース・ステートメント、または装置ファイルかデータベース・ファイル用のデータ記述仕様ソース・ステートメント (DDS ステートメント) を入れることができます。

注**:** ソース・タイプの物理 データベース・ファイルを作成している場合に、(データ記述仕様 (DDS) によって) そのファイルに関するフィールド・レベルの記述を作成していないのであれば、 **物理ファイル作成 (CRTPF)** コマンドまたは**ソース物理ファイル作成 (CRTSRCPF)** コマンドのどちらでも使用することができます。ただし、CRTSRCPF コマンドは、ソース物理ファイルを作成するためのコマンドとして設計されているために、通常この方が便利であり効率的です。ソース・タイプのデータベース・ファイルの作成時に DDS が指定される場合には、CRTPF コマンドまたは**論理ファイル作成 (CRTLF)** コマンドを使用しなければなりません。この 2 つのコマンドには、ソース入力を指定するための SRCFILE パラメーターおよび、SRCMBR パラメーターが含まれています。

ソース・ファイルの中のレコードには、少なくとも 3 つのフィールドが必要です。最初の 2 つはソース順序番号フィールドおよびデータ・フィールドであり、3 番目はソース・ステートメントが入るフィールドです。DDS を使用せずにソース・ファイルを作成する場合には、これらの 3 つのフィールドは IBM i オペレーティング・システムにより自動的に作成されます。追加のソース・フィールドは DDS で定義することができます。 順序番号フィールドの長さは、ゾーン形式の 6 桁で、そのうち 2 桁は小数部でなければなりません。日付フィールドの長さは、ゾーン形式の 6 桁でなければならず、小数部が含まれてはなりません。

ソースの順序番号フィールドおよび日付フィールドは、次の時点でソース・レコードに付加されます。

- レコードがシステムに読み取られたとき
- 原始ステートメント入力ユーティリティー (ライセンス交付を受けた WebSphere® Development Studio プログラムの一部) によってレコードが作成されたとき

インライン・データ・ファイル (標準ソース・ファイル様式として指定されたもの) が、装置から読み取られると、これらのフィールドが付加されます。スプール読み取りプログラムは、ソース順序番号フィールドに順序番号を入れ、全桁ゼロの日付フィールドをセットします。

装置から読み取ったレコードにこれらのフィールドが既に入っている場合には、その既存フィールドは変更されません。データベース・ファイルのレコードがソースの様式で、しかもデータ形式のインライン・データ・ファイルとして読み取られる場合には、ソース順序番号フィールドおよび日付フィールドは除去されます。

指定可能な値

***DATA**
作成されたファイルは、データ・レコードを含むか、またはデータ・レコードを記述します。

***SRC** 作成されたファイルは、ソース・レコードを含むか、または記述します。キー順ファイルを作成する場合には、6 桁のソース・レコード順序番号フィールドをキー・フィールドとして使用しなければなりません。

関連情報:

データベース・プログラミング

FRCRATIO パラメーター**:**

強制書き込み率 (FRCRATIO) パラメーターは、挿入、更新、または削除したレコードを補助 (永久) 記憶装置に強制的に書き込む前に、挿入、更新、または削除できるレコードの最大数を指定します。

このパラメーターに強制書き込み率を指定すると、挿入、更新、または削除されたレコードはすべて、少なくともこの書き込み率で示される頻度で、確実に補助記憶装置に書き込まれるようになります。システム障害が起こった場合にも、失われる可能性のあるレコードは、最後の強制書き込み操作が行われた後で挿入、更新、または削除されたレコードのみです。

強制書き込み率は、それが適用されるファイルのオープン・データ・パス (ODP) を使用して、挿入、更新、または削除されたすべてのレコードに対して、適用されます。2 つのプログラムが (SHARE(*YES) の指定により) ファイルを共用している場合は、強制書き込み率が、各プログラムにより挿入、更新、または削除された一連のレコードに、別々に適用されるわけではありません。これは、任意の組み合わせのレコード (両方のプログラムからの) が指定の強制書き込み率パラメーター値に一致すると、適用されます。例えば、ファイルに対して強制書き込み率 5 を指定すると、2 つのプログラムからのレコードがどのように組み合わされていても (一方のプログラムから 4 つ、他方のプログラムから 1 つなど)、処理レコードの合計が 5 つになったら、補助記憶装置へのレコード強制書き込みが行われます。2 つ以上のプログラムがそれぞれ別の ODP を介して同じファイルを使用している場合には、各 ODP について別々の強制書き込み率を指定することができます。各プログラムからのレコードの挿入、更新、および削除は、各 ODP ごとに個別に累計されます。

各データベース・ファイルには、それぞれ、強制書き込み率を割り当てることができます。複数の物理ファイルのデータにアクセスできる論理ファイルの場合には、基礎となる物理ファイルに指定したよりもさらに限定的な強制書き込み率 (少ないレコード数) を指定することができます。ただし、論理ファイルには、物理ファイルより低い強制書き込み率を指定することはできません。 どの物理ファイルに指定した強制書き込み率よりも緩やかな強制書き込み率を論理ファイルに指定した場合には、各物理ファイルに指定した強制書き込み率のうち最も限定的なものが、論理ファイルに使用されます。例えば、3 つの物理ファイルの強制書き込み率がそれぞれ 2、6、8 であった場合は、これら 3 つの物理ファイルを基礎とする論理ファイルの強制書き込み率には、2 より大きい値を指定することはできません。その論理ファイルに強制書き込み率が指定されていない場合は、2 であると見なされます。したがって、(どの物理ファイルが影響を受けるかに関係なく) プログラムが、論理ファイル中の 2 つのレコードを挿入、更新、または削除するたびに、それらのレコードが補助記憶装置に書き込まれます。

FRCRATIO の指定値は、SEQONLY の指定値を指定変更します。例えば、次のように指定したとします。

```
OVRDBF ... SEQONLY(*YES 20) FRCRATIO(5)
```

この場合、値 20 は指定変更され、5 レコード分のバッファーが使用されます。FRCRATIO(1) を指定した場合にもバッファーは使用されますが、このバッファーにはレコードが 1 つしか入りません。

挿入、更新、および削除されたレコードに関連したアクセス・パスは、そのアクセス・パスにかかわるレコードがすべて補助記憶装置に書き込まれた後で、初めて補助記憶装置に書き込まれます。ファイルの ODP が 1 つしかない場合には、強制書き込みが起こった時点で必ずアクセス・パスが補助記憶装置に書き込まれます。ファイルに 2 つ以上の ODP がある場合には、すべての ODP に関連した挿入済み、更新済み、および削除済みのすべてのレコードの強制書き込みが行われるたびに、アクセス・パスも補助記憶装置に書き込まれます。

注:

1. これらの規則は、2 以上の強制書き込み率を指定した場合に限り適用されます。強制書き込み率として 1 を指定した場合は、すべての ODP がクローズされるまで、アクセス・パスは補助記憶装置に書き込まれません。
2. ファイルをジャーナルに記録している場合は、FRCRATIO(*NONE) を指定してください。

指定可能な値

***NONE**
強制書き込み率は指定されません。レコードをいつ補助記憶域に書き込むかは、システムによって決定されます。

number-of-records-before-force
補助記憶域への強制書き込みが明示的に行われる前に処理される、更新、挿入、または削除されるレコードの数を指定してください。

関連情報:

ジャーナル管理とシステム・パフォーマンス

IGCFEAT パラメーター:

IGCFEAT パラメーターは、装置および言語に応じて、どの 2 バイト文字セット (DBCS) を使用するのかを指定します。

構成する 2 バイト文字セットに対応する IGCFEAT パラメーター、および DBCS フォント・テーブルについては、次の表を参照してください。

表 *9. IGCFEAT* パラメーターで構成可能な *DBCS* 機能

言語/装置	物理 **DBCS** ワークステーションのタイプ	構成されるタイプ - 型	構成に組み込まれる **DBCS** 機能
日本語表示装置	5295-001 表示装置、5295-002 表示装置、InfoWindow 3477-J 表示装置、5250PC 付き PS/55、グラフィック 5250PC 付き PS/55、グラフィック 5250PC 付き PS/55、3270 表示装置、IBM System i Access 付き PS/55	5555-B01 5555-B01 5555-B01, C01 5555-B01 5555-G01 5555-G02 5555-E01 3279-0 5555-B01	((2424J4 55FE)) ((2424J4 68FE)) ((2424J4 68FE)) ((2424J4 68FE)) ((2424J4 68FE)) ((2424J4 68FE)) ((2424J0 (1))) ((2424J0 (1))) ((2424J0 (1)))
日本語 24x24 印刷装置	5295-001 表示装置に接続 5295-002 表示装置に接続 PS/55 に接続 5227-001 印刷装置 5327-001 印刷装置	5553-B01 5553-B01 5553-B01 5553-B01 5553-B01	((2424J1 55FE)) ((2424J1 68FE)) ((2424J1 68FE)) ((2424J2 55FE)) ((2424J2 68FE))
日本語 32x32 印刷装置	5337-001 印刷装置 5383-200 印刷装置	5553-B01 5583-200	((3232J0 (1))) ((3232J0 (1)))
韓国語表示装置	5250 表示装置 3270 表示装置	5555-B01 3279-0	((2424K0 (1))) ((2424K0 (1)))
韓国語 24x24 印刷装置	5295 表示装置に接続 PS/55 に接続 5227-002 印刷装置	5553-B01 5553-B01 5553-B01	((2424K0 (1))) ((2424K0 (1))) ((2424K2 52FE))
中国語 (繁体字) 表示装置	5250 表示装置 3270 表示装置	5555-B01 3279-0	((2424C0)) ((2424C0))
中国語 (繁体字) 24x24 印刷装置	5295 表示装置に接続 PS/55 に接続 5227-003 印刷装置	5553-B01 5553-B01 5553-B01	((2424C0)) ((2424C0)) ((2424C2 5CFE))
中国語 (簡体字) 表示装置	5250 表示装置 3270 表示装置	5555-B01 3279-0	((2424S0)) ((2424S0))

表 9. *IGCFEAT* パラメーターで構成可能な *DBCS* 機能 (続き)

言語/装置	物理 **DBCS** ワークステーションのタイプ	構成されるタイプ - 型	構成に組み込まれる **DBCS** 機能
中国語 (簡体字) 24x24 印刷装置	PS/55 に接続 5227-005 印刷装置	5553-B01 5553-B01	((2424S0)) ((2424S2 6FFE))

JOB パラメーター**:**

JOB パラメーターは、コマンドを適用する対象となるジョブの名前を指定します。

ジョブ名は、システム上のすべてのタイプのジョブを識別します。各ジョブは、次の形式を持つ修飾ジョブ名により識別されます。

ジョブ番号/ユーザー名/ジョブ名

注**:** 構文は同様ですが、ジョブ名は、IBM i オブジェクト名とは異なる形式で修飾されます。

修飾ジョブ名の部分は次のようになります。

- ジョブ番号: ジョブ番号は、システムにより各ジョブに割り当てられた 6 桁の固有の番号です。ジョブ番号は、それ以外にジョブ名を他のジョブ名と区別する方法がない場合は、固有の修飾子となります。ジョブ番号は、 **ジョブ表示 (DSPJOB)** コマンドを用いて調べることができます。ジョブ番号を指定する場合には、長さは必ず 6 桁にしなければなりません。
- ユーザー名: ユーザー名により、実行するジョブのもとのユーザー・プロファイルが分かります。ユーザー名は、ユーザー・プロファイルの名前と同じであり、最高 10 文字の英数字からなっています。ユーザー名は、ジョブのタイプに応じて、次のように決まります。
  - バッチ・ジョブ: ユーザー名は、SBMJOB コマンドに指定した名前か、BCHJOB または SBMJOB コマンドで参照するジョブ記述に指定した名前です。
  - 対話式ジョブ: ユーザー名は、サインオン時に指定した名前か、あるいはワークステーションのジョブ項目で参照されている、ジョブ記述のデフォルト値から取られた名前です。
  - 自動開始ジョブ: ユーザー名は、自動開始ジョブのジョブ項目で参照されるジョブ記述に指定されています。
- ジョブ名: ジョブ名には、最高 10 文字の英数字を指定できます。最初の文字は英字でなければなりません。ジョブ名は、ジョブの 3 つのタイプに応じて、次のように決まります。
  - バッチ・ジョブ: ジョブ名は、**バッチ・ジョブ (BCHJOB)** コマンドまたは、 **ジョブ投入 (SBMJOB)** コマンドに指定した名前です。これらのコマンドに指定しなかった場合は、ジョブ記述の非修飾名が使用されます。
  - 対話式ジョブ: ジョブ名は、サインオンを行った装置 (ワークステーション) の名前と同じ名前になります。
  - 自動開始ジョブ: ジョブ名は、ジョブの実行時点で使用するサブシステム記述の、自動開始ジョブ項目に指定されます。このジョブ名は、 **自動開始ジョブ項目追加 (ADDAJE)** コマンドで指定されたものです。

各コマンドに必要となるのは、ジョブを識別するために単純名を使用することのみです。ただし、単純名が固有の名前ではない場合には、さらに修飾を加える必要があります。

重複ジョブ名

対話式 ジョブにおいて、重複したジョブ名をコマンドで指定した場合には、システムは、指定されたジョブ名で重複するものをすべて、修飾された形式でユーザーに示します。ジョブ名は、ユーザー名とジョブ番号によって修飾された形で表示されるため、ユーザーは、コマンドに指定したいジョブを、さらに明確に識別できます。これにより、正しい修飾ジョブ名を入力することができます。

バッチ・ジョブでコマンドに重複ジョブ名を指定すると、そのコマンドは処理されません。この場合には、ジョブ・ログにエラー・メッセージが書き込まれます。

JOB パラメーターには、使用するコマンドに応じて、次に示す値の 1 つ以上を表示できます。

指定可能な値

***** このジョブはコマンドが入力されたジョブ、すなわち JOB(*) の指定を持つコマンドが入力されたジョブです。

***JOBD**
単純ジョブ名として、ジョブ記述の修飾されていない名前が使用されます。

***NONE**
ログ表示 (DSPLOG) コマンドの場合、ジョブ名は指定されません。

job-name
単純ジョブ名が指定されます。

qualified-job-name
修飾ジョブ名を指定する必要があります。ジョブ修飾子 (ユーザー名およびジョブ番号) を指定しなかった場合は、該当のジョブ名を見つけるために、現在システム上に存在しているすべてのジョブが検索されます。指定した名前が重複していることが検出された場合は、修飾ジョブ名を指定しなければなりません。

関連概念:

127 ページの『単純オブジェクト名および修飾オブジェクト名』
ライブラリー内の特定のオブジェクトの名前は、単純名としても修飾名としても指定できます。

LABEL パラメーター:

LABEL パラメーターは、入出力操作で使用するテープ上のデータ・ファイルの、データ・ファイル識別コードを指定します。

データ・ファイルは、交換形式または保管/復元形式のいずれでも構いません。

注**:** 装置ファイル・コマンドは、交換形式のテープに対してのみ使用され、保管/復元形式の場合には使用されません。また、ユーザー定義の装置ファイルは、保管/復元操作では使用されません。

テープの各データ・ファイルには、それ自体のファイル・ラベルに保管されたデータ・ファイル識別コードがあります。各データ・ファイルのデータ・ファイル・ラベル (または見出しラベル) は、ファイルのデータの直前にあるテープに保管されます。すなわち、テープでは、ファイルごとに、その見出しラベルとデータ・レコードとで 1 つの単位が形成され、1 つのファイルの後に別のファイルが続いています。各ラベルには、データ・ファイル識別コードのほかに、ファイルに関するその他の情報 (ファイル順序番号、レコード属性、ブロック属性、マルチボリューム・データ・ファイルかどうかなど) も入っています。

通常、データ・ファイル識別コードは 8 文字以下の英数字文字ストリングです。しかし、実際の最大文字数は、ファイルでどのようなデータ形式が使用されるか、およびどの CL コマンドに識別コードが指定されるかなどの、いくつかの条件によって決まります。ファイル ID フィールドの未使用の部分は、空白のままになります。

データ・ファイル識別コードの最初の文字は、英字 (A から Z、$、#、または @) である必要があります。2 番目以降の文字は、英数字 (A から Z、0 から 9、$、#、_、.、および @) である必要があります。識別コードを単一引用符で囲めば、特殊文字も使用できます。ただし、そのテープを IBM i 以外のオペレーティング・システムでも使用する場合は、識別コードの指定に関する要件も考慮しなければなりません。

テープのデータ・ファイル識別コード

テープのデータ・ファイル識別コードには 17 文字まで指定できます。ただし、テープを IBM i 以外のオペレーティング・システムで使用する場合には、8 文字以下の識別コード (または 17 文字以下の修飾識別コード) を使用しなければなりません。8 文字を超える場合には、識別コードを修飾した上で単一引用符で囲まなければなりません。修飾識別コードの 2 つの部分はピリオドで区切り、どちらの部分も、8 文字以下になるようにしてください。例えば、LABEL('TAXES.JAN1980') のように指定します。この制限が適用されるのは、次のコマンドです。**テープ・ファイル作成 (CRTTAPF)** コマンド、 **テープ・ファイル変更 (CHGTAPF)** コマンド、**テープ・ファイル指定変更 (OVRTAPF)** コマンド、および**テープ表示 (DSPTAP)** コマンド。

データ・ファイルがボリュームに書き込まれる時点で、そのデータ・ファイルの識別コードが同じボリュームに記録されます。入出力操作の場合には、テープ装置ファイル・コマンドの 1 つに、識別コードを指定したり、あるいは、高水準言語プログラムにより、該当の装置ファイルをオープンする際、パラメーターとして識別コードを渡すことができます。

保管/復元形式

保管/復元形式のテープの場合には、識別コードに最高 17 文字を指定できます。 ライブラリー名を用いてラベルを生成する場合、識別コードは 10 文字を超えることはできません。ライブラリー名以外のラベルを指定することもできます。

指定可能な値

LABEL パラメーターには、使用するコマンドに応じて、次の値のいずれか 1 つを指定することができます。

***ALL** 指定したテープ・ボリュームの、すべてのデータ・ファイル識別コードについて、ラベルが画面に表示されます。

***NONE**
データ・ファイル識別コードは、識別されません。テープ操作で使用するためには、装置ファイルまたはデータベース・ファイル、あるいはその両方がオープンされる前に、データ・ファイル識別コードを指定しなければなりません。

***SAME**
テープ装置ファイルに入っている既存のデータ・ファイル識別コードは、変更されません。

data-file-identifier
装置ファイル記述で使用するため、または表示するためにデータ・ファイル識別コードを指定します。

***LIB** ファイル・ラベルはシステムによって作成され、LIB パラメーターに指定したライブラリーの名前がファイル名の修飾子として使用されます。

***SAVLIB**
ファイル・ラベルはシステムによって作成され、SAVLIB パラメーターに指定したライブラリーの名前がファイル名の修飾子として使用されます。

LICOPT パラメーター**:**

ライセンス内部コード・オプション (LICOPT) パラメーターを使用して、コンパイル時間オプションを個別に指定します。

このパラメーターは、選択した各タイプのコンパイラー・オプションの考えられる利点と欠点について理解している上級プログラマーを対象としています。

次の表は、ライセンス内部コード・オプション (LICOPT) パラメーターによって認識されるストリングを示します。これらのストリングでは大文字と小文字が区別されませんが、読みやすくするために大文字と小文字を混ぜて示しています。

表 *10. LICOPT* パラメーター・ストリング

ストリング	説明
AllFieldsVolatile	設定されると、すべてのフィールドを揮発性として扱います。
NoAllFieldsVolatile	設定されると、どのフィールドも揮発性として扱いません。
AllowBindingToLoadedClasses	実行中の Java 仮想マシン内で defineClass 呼び出しの結果として作成された一時的なクラス表現が、同じ Java 仮想マシン内の他のクラス表現と密結合できることを示します。
NoAllowBindingToLoadedClasses	実行中の Java 仮想マシン内で defineClass 呼び出しの結果として作成された一時的なクラス表現が、同じ Java 仮想マシン内の他のクラス表現と密結合できないことを示します。
AllowClassCloning	複数の Java プログラムが JAR ファイル用に生成されるときに、1 つのプログラムのクラスのコピーを、別のプログラムの生成コードに組み込めるようにします。積極的なインライン化が可能になります。
NoAllowClassCloning	1 つのプログラムのクラスのコピーを、別のプログラムの生成コードに組み込めないようにします。
AllowInterJarBinding	コンパイル中のクラスまたは JAR ファイルの外部にあるクラスとの密結合が可能になります。積極的な最適化が可能になります。
NoAllowInterJarBinding	コンパイル中のクラスまたは JAR ファイルの外部にあるクラスとの密結合を禁止します。これは、CRTJVAPGM の CLASSPATH パラメーターと JDKVER パラメーターの指定を上書きします。
AllowMultiThreadedCreate	CRTJVAPGM は、作成中に複数のスレッドを使用します (使用可能な場合)。マルチプロセッサー・システムでは、これによって一度に複数のプロセッサーを使用して、長時間の CRTJVAPGM 操作で必要な全体の時間を削減することができます。ただし、CRTJVAPGM はより多くのシステム資源を使用するため、他のアプリケーションで使用可能な資源が少なくなります。
NoAllowMultiThreadedCreate	CRTJVAPGM が 1 つのスレッドのみを使用して、通常どおり実行することを示します。

表 10. LICOPT パラメーター・ストリング (続き)

ストリング	説明
AnalyzeObjectLifetimes	可視クラスを使用して分析を実行し、存在期間の短いオブジェクトを判別します。このような存在期間の短いオブジェクトは、それが割り振られたメソッド内でしか存在しないので、より積極的な最適化の対象となります。
NoAnalyzeObjectLifetimes	存在期間の短いオブジェクトの分析を実行しません。
AllowBindingWithinJar	ZIP ファイルまたは JAR ファイル内のクラス表現が、同じ ZIP ファイルまたは JAR ファイル内の他のクラス表現と密結合できることを示します。
NoAllowBindingWithinJar	ZIP ファイルまたは JAR ファイル内のクラス表現が、同じ ZIP ファイルまたは JAR ファイル内の他のクラス表現と密結合できないことを示します。
AllowInlining	変換プログラムに、ローカル・メソッドのインライン化が許可されることを伝えます。これは、最適化レベル 30 および 40 のデフォルトです。
NoAllowInlining	変換プログラムに、ローカル・メソッドのインライン化が許可されることを伝えません。
AssumeUnknownFieldsNonvolatile	外部クラス内のフィールドの属性を判別できないときに、このパラメーターは、フィールドが不揮発性であると想定することによってコードを生成します。
NoAssumeUnknownFieldsNonvolatile	外部クラス内のフィールドの属性を判別できないときに、このパラメーターは、フィールドが揮発性であると想定することによってコードを生成します。
BindErrorHandling	AssumeUnknownFieldsNonvolatile、PreresolveExtRef、または PreLoadExtRef ライセンス内部コード・オプションを受け入れた結果として、クラス表現に現行のコンテキストでは使用できないメソッド表現が含まれていることが Java 仮想マシンのクラス・ローダーによって検出された場合に、実行しなければならない処置を指定します。
BindInit	ローカル初期メソッドに対するバインド済み呼び出しを使用します。
NoBindInit	ローカル初期メソッドに対するバインド済み呼び出しを使用しません。
BindSpecial	ローカル特殊メソッドに対するバインド済み呼び出しを使用します。
NoBindSpecial	ローカル特殊メソッドに対するバインド済み呼び出しを使用しません。
BindStatic	ローカル静的メソッドに対するバインド済み呼び出しを使用します。
NoBindStatic	ローカル静的メソッドに対するバインド済み呼び出しを使用しません。
BindTrivialFields	プログラムの作成中に単純なフィールド参照をバインドします。
NoBindTrivialFields	最初のアクセス時にフィールド参照を解決します。
BindVirtual	ローカル最終仮想メソッドに対するバインド済み呼び出しを使用します。
NoBindVirtual	ローカル最終仮想メソッドに対するバインド済み呼び出しを使用しません。

表 *10. LICOPT* パラメーター・ストリング *(*続き*)*

ストリング	説明
DeferResolveOnClass	クラスの名前であると見なされるストリング・パラメーター (例えば、java.lang.Integer) を採用します。PreresolveExtRef を最適化レベル 40 に設定すると、DeferResolveOnClass で指定されたクラスは事前解決操作に使用されません。これは、コード内の未使用のパスにおける一部のクラスが CLASSPATH にない場合に役立ちます。欠落した各クラスに「DeferResolveOnClass='somepath.someclass'」を指定すると、このことに関係なく最適化レベル 40 を使用することができます。複数の DeferResolveOnClass 項目が使用可能です。
DevirtualizeFinalJDK	CRTJVAPGM が標準 JDK の知識を使用して、最終メソッドまたは最終クラスのメンバーとして認識されている JDK メソッドへの呼び出しを非仮想化することを許可します。これは、最適化レベル 30 および 40 でのデフォルトです。
NoDevirtualizeFinalJDK	CRTJVAPGM が標準 JDK の知識を使用して、最終メソッドまたは最終クラスのメンバーとして認識されている JDK メソッドへの呼び出しを非仮想化することを許可しません。
DevirtualizeRecursive	一部の再帰的メソッドのケースで特殊なコードが生成されるようにし、再帰的メソッド呼び出しのオーバーヘッドの大部分を除去します。ただし、再帰的メソッドの最初の入り口で追加の検査論理が生成されるため、再帰が浅いケースではパフォーマンスが向上しない可能性があります。
NoDevirtualizeRecursive	一部の再帰的メソッドのケースで特殊なコードが生成されるようにしません。
DisableIntCse	特定のタイプの整数式のコードを生成する際に、特定の共通副次式の最適化が使用不可になるようにします。これにより、最適化変換プログラムに他の最適化を行う機会が与えられるため、最適化全体が向上する可能性があります。
NoDisableIntCse	特定のタイプの整数式のコードを生成する際に、特定の共通副次式の最適化を使用不可にならないようにします。これにより、通常、最適化レベルが低いコードではパフォーマンスが向上します。
DoExtBlockCSE	拡張基本ブロック共通副次式の除去を実行します。
NoDoExtBlockCSE	拡張基本ブロック共通副次式の除去を実行しません。
DoLocalCSE	ローカル共通副次式の除去を実行します。
NoDoLocalCSE	ローカル共通副次式の除去を実行しません。
EnableCseForCastCheck	設定されると、以前のインスタンスに DAG できるキャスト・チェック用のコードを生成します。
NoEnableCseForCastCheck	設定されません。以前のインスタンスに DAG できるキャスト・チェック用のコードを生成しません。
ErrorReporting	実行時エラー報告フィールド**: 検査またはクラス形式エラーが検出されたときに、コンパイルを失敗させるためのオプションを提供します。0 = すべてのエラーを即時に報告します。1 = バイトコード検査エラーの報告を遅らせます。2 = バイトコード検査エラーとクラス形式エラーの報告を実行時まで遅らせます。
HideInternalMethods	複製したクラスのメソッドを内部メソッドとします。それによって、それらのメソッドに参照がない場合、またはすべての参照がインライン化される場合に、そのメソッドを省略できるようにします。デフォルトは、最適化 40 の場合は HideInternalMethods で、最適化 0 から 30 の場合は NoHideInternalMethods です。

表 10. LICOPT パラメーター・ストリング (続き)

ストリング	説明
InlineArrayCopy	スカラー配列の一部のケースで System.arraycopy メソッドのインライン化が行われるようにします。
NoInlineArrayCopy	System.arraycopy メソッドのインライン化が行われないようにします。
InlineInit	java.lang クラスの初期メソッドをインライン化します。
NoInlineInit	初期メソッドをインライン化しません。
InlineMiscFloat	java.lang.Math からの各種 float/double メソッドをインライン化します。
NoInlineMiscFloat	各種 float/double メソッドをインライン化しません。
InlineMiscInt	java.lang.Math からの各種 int/long メソッドをインライン化します。
NoInlineMiscInt	各種 int/long メソッドをインライン化しません。
InlineStringMethods	java/lang/String からの特定のメソッドのインライン化を許可します。
NoInlineStringMethods	java/lang/String からの特定のメソッドのインライン化を禁止します。
InlineTransFloat	java.lang.Math からの超越 float/double メソッドをインライン化します。
NoInlineTransFloat	超越 float/double メソッドをインライン化しません。
OptimizeJsr	単一のターゲットを持つ "jsr" バイトコードについて、より良いコードを生成します。
NoOptimizeJsr	単一のターゲットを持つ "jsr" バイトコードについて、より良いコードの生成を抑制します。
PreloadExtRef	参照されたクラスがメソッドの入り口で (クラスの初期設定を行わずに) プリロードできることを示します。
NoPreloadExtRef	参照されたクラスがメソッドの入り口でプリロードできないことを示します。ただし、PreresolveExtRef パラメーターはこの設定を指定変更し、参照されたクラスがプリロードおよび初期設定されるようにします。
PreresolveExtRef	参照されたメソッドをメソッドの入り口で事前に解決します。
NoPreresolveExtRef	メソッド参照を最初のアクセス時に解決します。これは、他のマシンで実行されるプログラムで「クラス未検出」例外を解決するために使用します。
ProgramSizeFactor	JAR ファイルが大きいため、複数の Java プログラムが必要な場合、この数値 (デフォルトは 100) を使用して、各プログラムがどれくらいまで大きくなれるかを決定します。
ShortCktAthrow	設定されると、athrow の回避を試みます。
NoShortCktAthrow	設定されないと、athrow の回避を試みません。
ShortCktExSubclasses	設定されると、例外のサブクラスの一部を認識し、それらを直接回避します。
NoShortCktExSubclasses	設定されないと、例外のサブクラスの一部を認識せず、それらを直接回避しません。
StrictFloat	Java 仕様に厳密に準拠しない浮動小数点の最適化を禁止します。
NoStrictFloat	Java 仕様に厳密に準拠しない浮動小数点の最適化を許可します。

二重のアスタリスク (**) は、これらのストリングが stringname=number (間にはスペースなし) という構文の数値の入力を必要とすることを示します。

MAXACT パラメーター**:**

並行して開始し、活動状態を維持することができるジョブの最大数を指定するには、最大活動レベル (MAXACT) パラメーターを使用します。この指定は、ジョブ待ち行列項目、通信項目、経路指定項目、またはワークステーション項目を介して行うことができます。

ジョブは、実行が開始された時点から実行が完了する時点まで、活動状態であると見なされます。これには、次に示す時間が含まれます。

- ジョブが実際に処理されている時間。
- ジョブがワークステーション・ユーザーからの応答を待っている時間。
- ジョブが開始されていて処理可能ではあるが、実際には処理装置を使用していない時間。例えば、ジョブが指定のタイム・スライスを使い終えて、別のタイム・スライスが割り当てられるのを待っている時間。
- ジョブが開始されたが、処理のために使用できる状態になっていない時間。ジョブがそのメッセージ待ち行列にメッセージが送られてくるのを待っている時間。

指定可能な値

***NOMAX**
同時に活動状態になれるジョブ数の限度はありません。

maximum-active-jobs
この項目を介して、同時に活動状態にできるジョブの最大数を表す値を指定してください。

関連情報:

実行管理機能

OBJ パラメーター**:**

オブジェクト (OBJ) パラメーターには、このパラメーターを使用したコマンドにより処理される 1 つ以上のオブジェクトの名前を指定します。

IBM i ライブラリーに存在する必要のあるオブジェクトが OBJ パラメーターによって識別される場合は、すべてのオブジェクトが、使用するコマンドに応じて、以下のいずれかである必要があります。

- LIB パラメーターで指定されるライブラリー
- SAVLIB パラメーター
- OBJ パラメーターのライブラリー修飾子
- OBJ パラメーターのパス名のライブラリー部分

コマンドによっては、一群のオブジェクトの総称名を指定することができるものもあります。総称名は、各オブジェクト名に共通する文字の後にアスタリスク (*) を付けたもの、例えば、ABC* などです。名前に * が付いていない場合には、システムはその名前を完全なオブジェクト名であると見なします。

指定可能な値

OBJ パラメーターには、使用するコマンドに応じて、次のタイプの値を指定できます。

- *ALL
- 単純オブジェクト名
- 修飾オブジェクト名
- 総称オブジェクト名

- 修飾総称オブジェクト名
- パス名

関連概念:

134 ページの『パス名 (*PNAME)』
パス名は、統合ファイル・システムの中でオブジェクトを位置指定するために使用できる文字ストリングです。

OBJTYPE パラメーター**:**

オブジェクト・タイプ (OBJTYPE) パラメーターは、オブジェクトが指定されたコマンドにより操作できる IBM i オブジェクトのタイプを指定します。

OBJTYPE パラメーターに指定できるオブジェクト・タイプは、コマンドによって異なります。

特定のコマンド・タイプに関連する特定のコマンドを知らなくても、この表で示すオブジェクト関連コマンドによって、ほとんどのオブジェクトに対して一般的な操作を行うことができます。例えば、**ファイル・コピー (CPYF)** または **ライブラリー・コピー (CPYLIB)** などの特定のコマンドを使用しなくても、**複製オブジェクト作成 (CRTDUPOBJ)** コマンドを使用して、ファイルまたはライブラリーのコピーを作成できます。

オブジェクト関連コマンド

このセクションには、OBJTYPE パラメーターを含むコマンドをリストします。コマンドを使用して操作可能なオブジェクト・タイプを見つけるには、リストされている各コマンドの情報を参照してください。

次に示すコマンドには OBJTYPE パラメーターがありますが、処理対象となるオブジェクト・タイプの数はわずかです。

- CHKDLO では *DOC および *FLR が処理対象になります。
- CPROBJ および DCPOBJ では *FILE、*MENU、*MODULE、*PGM、*PNLGRP、および *SRVPGM が処理対象になります。
- CRTSQLPKG では *PGM および *SRVPGM が処理対象になります。
- DSPPGMADP では *PGM、*SQLPKG および *SRVPGM が処理対象になります。
- DSPPGMREF では *PGM および *SQLPKG が処理対象になります。
- RSTCFG では *CFGL、*CNNL、*COSD、*CTLD、*DEVD、*LIND、*MODD および *NWID が処理対象になります。
- SAVLICPGM では *LNG および *PGM が処理対象になります。
- SETOBJACC では *FILE および *PGM が処理対象になります。

DSPLNK および WRKLNK では、すべてのオブジェクト・タイプが処理対象になります。

上記以外に、ALCOBJ コマンドおよび DLCOBJ コマンドでも、オブジェクト・タイプ値の指定が必要です。しかし、これらのコマンドの場合は、オブジェクト・タイプ値を必須パラメーター OBJ の 4 つの値 (値のリスト) の中の 1 つとして指定します。

以下のオブジェクト関連コマンドでは、数多くのオブジェクト・タイプが処理対象になります。

オブジェクト	オブジェクト権限	保管/復元	ジャーナル	その他
CHGOBJD CHKOBJ CRTDUPOBJ DLTOBJ DSPOBJDMOVOBJ RNMOBJ RTVOBJD WRKOBJ	CHGOBJAUD CHGOBJOWN CHGOBJPGP DSPOBJAUT EDTOBJAUT GRTOBJAUT RVKOBJAUT	RSTOBJ SAVCHGOBJ SAVOBJ	CHGJRNOBJ ENDJRNOBJ STRJRNOBJ	DMPOBJ DMPSYSOBJ PRTDSKINF WRKOBJLCK

関連概念:

124 ページの『外部オブジェクト・タイプ』
外部オブジェクトの多くのタイプは、ライブラリーに保管されます。一部のタイプの外部オブジェクトは、ディレクトリー内の統合ファイル・システムのみに保管できます。

関連資料:

5 ページの『CL コマンド名』
コマンド名は、コマンドの実行時に呼び出されるプログラムが実行する機能を識別します。コマンド名は、動詞 (動作) および動作の対象 (対象となるオブジェクト) を識別する名詞または句を組み合わせたものです (コマンド = 動詞 + 対象となるオブジェクト)。

97 ページの『TEXT パラメーター』
TEXT パラメーターは、作成または変更するオブジェクトを簡潔に説明するユーザー定義の記述を指定します。

53 ページの『複数のオブジェクトを処理する CL コマンド』
単一オブジェクト・タイプに対して機能するコマンドに加えて、複数のオブジェクト・タイプに対して機能するコマンドもあります。これらのコマンドは、異なるタイプの複数のオブジェクトを同時に扱えるので、より強力です。

OUTPUT パラメーター**:**

OUTPUT パラメーターは、表示コマンドからの出力を画面に表示するか、印刷するか、あるいは出力ファイルに書き出すかを指定します。

基本的には、出力が表示、印刷、または書き込みされるいずれの場合にも同じ情報が提供されますが、形式のみは、該当する装置に最も適した形式で情報が表示されるように変更されます。例えば、表示画面よりも印刷ページの方が行数が多いため、欄見出しの反復は印刷出力の方が少なくなります。

出力を画面に表示する場合、表示コマンドを出したワークステーションにその出力が送られます。この出力は、その表示コマンドで使用される表示装置ファイルに指定された形式で表示されます。各表示コマンドで作成される出力ごとに、別々の装置ファイルが使用されます。また、表示する場合、印刷する場合、および出力ファイルに書き出す場合によって、それぞれ装置ファイルが異なります。ほとんどの場合、コマンドの名前は、該当のタイプの装置ファイルのファイル名の一部になります。

出力を印刷する場合には、出力はスプールされ、項目がジョブの出力待ち行列に入れられます。この出力は、**印刷装置書出プログラム開始 (STRPRTWTR)** コマンドに指定されている印刷装置で印刷することができます。

注**:** IBM 提供の印刷装置ファイルは、出荷時点では SPOOL(*YES) が指定されていますが、**印刷装置ファイル指定変更 (OVRPRTF)** コマンド、または**印刷装置ファイル変更 (CHGPRTF)** コマンドを使用して、SPOOL(*NO) に変更することができます。

表示コマンドに OUTPUT パラメーターを指定しなかった場合は、デフォルト値である * を指定したものと見なされます。このデフォルト値の結果として生成される出力は、コマンドを入力したジョブのタイプによって決まります。次の表に、対話式ジョブおよびバッチ・ジョブの場合にどのように出力が生成されるかを示します。

表 *11.* 対話式およびバッチの出力

出力	対話式ジョブ	バッチ・ジョブ
*	表示	印刷
*PRINT	印刷	印刷

指定可能な値

* 対話式ジョブから要求した出力は画面に表示されます。 バッチ・ジョブによって要求された出力は、ジョブのスプール出力の印刷となります。

***PRINT**
出力は、ジョブのスプールされた出力により印刷されます。

***OUTFILE**
出力は、指定したデータベース・ファイルに書き込まれます。

PRTTXT パラメーター**:**

印刷テキスト (PRTTXT) パラメーターは、リストの最下部および区切りページに印刷するテキストを指定します。

印刷テキストは、ジョブがシステムに入った時点で、ジョブ属性からコピーされます。別のシステムで開始された印刷ファイルでは、ターゲット・システムの印刷テキストは使用されません。印刷テキストは、特定のジョブの印刷テキストを定義するためのジョブ属性 (PRTTXT) であり、また、*SYSVAL が指定されたジョブのデフォルト値となるシステム値 (QPRTTXT) です。 QPRTTXT は、システム全体にわたって、すべてのジョブのデフォルト値となります。

印刷テキストは、長さ 30 文字以内で指定できます。印刷テキストは、用紙幅の中央に位置合わせされ、オーバーフロー域に印刷されます。30 文字幅のテキスト・フィールド内でのテキストの中央そろえは、ユーザーが行わなければなりません。

印刷テキストがブランクでなければ、システムは、各ページの下に 30 文字のテキストを印刷します。このテキストは、通常、オーバーフロー行の次に置かれ、 (用紙長において可能であれば) テキストの前に 1 行のブランクが設けられます。ユーザーの印刷したテキストがオーバーフロー行を超えている場合は、ユーザー・テキストの最後の行の次に印刷テキストが置かれ、この場合も、可能であればその印刷テキストの前に 1 行のブランクが設けられます。オーバーフロー行が用紙の最終行の場合には、印刷テキストも用紙の最終行に印刷されるため、ユーザー・テキストの上に重ねて印刷テキストが印刷されることもあります。

ジョブ区切りおよびファイル区切り用の印刷テキストは、区切りページの 1 行目に置かれます。ジョブ区切りには、ファイルの印刷時点でその区切り記号を作成したジョブの印刷テキストが入ります。ファイル区切りには、その後に続くスプール・ファイルの印刷テキストと同じ印刷テキストが入ります。

印刷テキストは、あらゆるジョブ・タイプに指定できます。システムおよびサブシステムのモニター・ジョブでは、システム値が使用されます。読み取りプログラム・ジョブおよび書き出しプログラム・ジョブでは、システム値が使用されます。ただし、該当する読み取りプログラムまたは書き出しプログラムに関連した QSPLxxxx ジョブ記述内で印刷テキストを変更した場合は、別です。

印刷テキストは、次の階層順序に従って決定されます。ある階層で印刷テキストが指定されていなかった場合は、次の順序の階層のテキストが使用されます。

階層順序は、優先順位の高いものから示すと、次のようになります。

- 指定変更印刷ファイルの値
- 印刷ファイルの値
- **ジョブ変更 (CHGJOB)** コマンドにより変更されたジョブ属性
- **ジョブ投入 (SBMJOB)** コマンドまたは、 **バッチ・ジョブ (BCHJOB)** コマンドにより設定されたジョブ属性
- ジョブ記述
- システム値

指定可能な値

システム値 QPRTTXT には、*SYSVAL 以外の任意の文字ストリングを指定できます。*BLANK を指定した場合は、印刷テキストは印刷されません。PRTTXT には、使用するコマンドに応じて次に示す値のいくつかを選択することができます。

***SAME**
印刷テキストは変更されません。

***CURRENT**
印刷テキストは投入ジョブから取得されます。

***JOBD**
印刷テキストは、ジョブの実行を制御するジョブ記述から取得されます。

***SYSVAL**
印刷テキストは、システム値 QPRTTXT から取得されます。

***BLANK**
印刷されたテキストまたは空白はありません。

'print-text'
30 文字のテキストを指定してください。テキストにブランクが含まれている場合には、項目全体を単一引用符で囲まなければなりません。ページの中央にテキストを位置合わせして印刷するためには、その印刷テキストを、30 桁のフィールドの中央に位置合わせしなければなりません。

REPLACE パラメーター**:**

置き換え (REPLACE) パラメーターは作成コマンドで使用されます。このパラメーターは、既存のオブジェクト (存在する場合) を、同じ名前、ライブラリー、オブジェクト・タイプを持つ、今回作成するオブジェクトと置き換えるように指定します。

新しいオブジェクトのユーザーには、旧オブジェクトの場合と同じ権限が認可されます。置き換えられるオブジェクトが権限リストによって保護されている場合は、新しいオブジェクトも同じ権限リストによって保護されます。新しいオブジェクトの権限は、置き換えられるオブジェクトの共通権限と同じです。作成コマンドの AUT パラメーターは無視されます。置き換えられるオブジェクトの専用認可は、そのまま新しいオブジェクトに継承されます。置き換えられるオブジェクトの所有者は、新しいオブジェクトに継承されません。新しいオブジェクトの所有者は、そのオブジェクトの作成者、または作成者のグループ・プロファイルです。現行ジョブまたは別のジョブが使用中である場合、パネル・グループ、表示装置ファイル、およびメニューなどのいくつかのオブジェクトは、置き換えることができません。

作成されるオブジェクトがプログラムまたはサービス・プログラムの場合には、置き換えられるプログラムのユーザー・プロファイル (USRPRF パラメーター) の値が使用されます。 **プログラム作成 (CRTPGM)** コマンドまたは**サービス・プログラム作成 (CRTSRVPGM)** コマンドのユーザー・プロファイルの値 (USRPRF パラメーター) は、無視されます。置き換えられるプログラムまたはサービス・プログラムのユーザー・プロファイル (USRPRF パラメーター) の値が *OWNER の場合は、置き換えられるプログラムまたはサービス・プログラムの現在の所有者のみが、既存のプログラムまたはサービス・プログラムと置き換わる新しいプログラムまたはサービス・プログラムを作成できます。既存のオブジェクトと作成されるオブジェクトの所有者が一致しない場合、オブジェクトは作成されず、メッセージ CPF2146 が送られます。

作成されるオブジェクトがプログラムまたはサービス・プログラムの場合は、オブジェクトを作成するユーザーが、USEADPAUT(*YES) 属性によって、プログラムまたはサービス・プログラムを作成できる限り、置換されたプログラムまたはサービス・プログラムの使用借用権限 (USEADPAUT) 値が使用されます。ユーザーが借用権限を使用するプログラムまたはサービス・プログラムを作成できるかは、QUSEADPAUT システム値によって決まります。例えば、置換される既存のオブジェクトが USEADPAUT(*YES) を持ち、ユーザーが、借用権限を使用するプログラムまたはサービス・プログラムを作成する権限を持たない場合、作成されるプログラムまたはサービス・プログラムは USEADPAUT(*NO) を持ちます。この場合、USEADPAUT 値はコピーされません。ユーザーが、借用権限を使用するプログラムまたはサービス・プログラムを作成する権限を持つ場合、作成されるプログラムまたはサービス・プログラムは、置換されるプログラムまたはサービス・プログラムと同じ USEADPAUT 値を持ちます。USEADPAUT 値が置換されるオブジェクトにコピーされたかどうかを示す通知メッセージが送られます。

作成中のオブジェクトがファイルであり、REPLACE パラメーターでデフォルトの値を取るか *YES を指定した場合は、保管ファイルおよび DDM ファイルを除く既存の装置ファイルのうちで、同じ修飾名を持つファイルが、新しいファイルで置き換えられます。例えば、既存の表示装置ファイルを、新しい印刷装置ファイルまたはテープ・ファイルなどで置き換えることができます。

既存のオブジェクトを新しいオブジェクトで置き換えるには、その既存のオブジェクトに対するオブジェクト管理権 (*OBJMGT)、オブジェクト存在権 (*OBJEXIST)、および読み取り権 (*READ) が必要です。

新しいオブジェクトの作成が正常に完了すると、既存のオブジェクトは名前が変更され、ライブラリー QRPLOBJ に移されるか、あるいはそのオブジェクトが独立 ASP にある場合はライブラリー QRPLxxxxx に移されます (この「xxxxx」は、ASP グループの基本 ASP の番号です)。古いオブジェクトは、文字 Q の後にタイム・スタンプを付加した名前に変更され、ライブラリー QRPLOBJ に移されるか、あるいはそのオブジェクトが独立 ASP にあった場合はライブラリー QRPLxxxxx に移されます。既存のオブジェクトを QRPLOBJ ライブラリーに移せない場合 (例えば、QRPLOBJ が別のジョブによりロックされている場合)、既存のオブジェクトは、ジョブの QTEMP ライブラリーに移されます。既存のオブジェクトをジョブの QTEMP ライブラリーに移せない場合、既存のオブジェクトは削除されます。既存のオブジェクトを QRPLxxxxx ライブラリーに移せない場合 (例えば、QRPLxxxxx が他のジョブによりロックされている場合)、既存のオブジェクトは削除されます。

制限

プログラムは、実行中でも置き換えることができます。ただし、置き換えられたプログラムの名前が Q タイム・スタンプ名に変更された後で、そのプログラムがプログラム・メッセージ待ち行列を参照すると、プログラムは実行不能になり、該当するプログラム・メッセージ待ち行列が見つからないことを示すエラー・メッセージが表示されます。

物理データベース・ファイル、論理データベース・ファイル、および保管ファイルは、いかなるファイルによっても置き換えることはできません。

ライブラリー QRPLOBJ は、システムの初期プログラム・ロード (IPL) が完了した時点で消去されます。ライブラリー QRPLxxxxx は、ASP グループの基本 ASP がオンに変更になった時点で消去されます。

指定可能な値

***YES** 作成する新しいオブジェクトと同じ名前を持ち、しかも同じオブジェクト・タイプの既存のオブジェクトが同じライブラリーにある場合に、システムはその既存のオブジェクトを新しいオブジェクトで置き換えます。

***NO** 作成する新しいオブジェクトと同じ名前を持ち、しかも同じオブジェクト・タイプの既存のオブジェクトが同じライブラリーにあっても、システムはその既存のオブジェクトを新しいオブジェクトで置き換えません。

JOBPTY、**OUTPTY**、および **PTYLMT** スケジューリング優先順位パラメーター**:**

システムが処理すべきジョブとスプール・ファイルを選択する順序を決定するときには、スケジューリング優先順位パラメーターの指定が使用されます。

各ジョブには、ジョブ選択およびスプール・ファイル出力の両方に使用するスケジューリング優先順位を指定します。ジョブのスケジューリング優先順位は、 **バッチ・ジョブ (BCHJOB)**、**ジョブ投入 (SBMJOB)**、**ジョブ記述作成 (CRTJOBD)**、および**ジョブ記述変更 (CHGJOBD)** などのコマンドの JOBPTY パラメーターにより指定します。ジョブからのスプール出力を生成する場合の優先順位は、これらのコマンドの OUTPTY パラメーターにより指定します。

さらに、どのジョブも特定のユーザー・プロファイルのもとで処理されるので、各ジョブの優先順位には、**ユーザー・プロファイルの作成 (CRTUSRPRF)** コマンド、および**ユーザー・プロファイル変更 (CHGUSRPRF)** コマンドに指定する PTYLMT パラメーターの値によって、制約を付けることができます。このパラメーター値は、当該ユーザー・プロファイルのもとで実行されるすべてのジョブが持ち得る最高ジョブ・スケジューリング優先順位および最高出力優先順位を制御します。すなわち、ジョブ・コマンドの JOBPTY パラメーターおよび OUTPTY パラメーターに指定する優先順位は、そのジョブに関連したユーザー・プロファイルの PTYLMT パラメーターに指定された優先順位を超えることはできません。スケジューリング優先順位は、ジョブ実行のために選択する順序を決めるために使用されるもので、クラス・オブジェクトに指定する実行優先順位とは関係ありません。

3 つのスケジューリング優先順位パラメーターで指定する項目は次のとおりです。

- PTYLMT パラメーターは、投入するすべてのジョブに対し、最高のスケジューリング優先順位を指定します。ユーザー・プロファイルに影響を及ぼすコマンドでは、PTYLMT パラメーターの値が、それぞれの特定のジョブに関連したコマンドの JOBPTY パラメーターに指定可能な最高優先順位となります。ジョブの投入に使用するコマンドには、これより低い優先順位をジョブの優先順位として指定することができます。BCHJOB コマンドまたは SBMJOB コマンドの JOBPTY に、関連するユーザー・プロファイルの PTYLMT に指定した優先順位より高い優先順位を指定した場合には、エラー・メッセージが画面に表示され、PTYLMT に指定した最高優先順位を指定したものと見なされます。CHGJOB コマンドまたは CHGJOBD コマンドに PTYLMT 値より高い優先順位を指定した場合は、エラー・メッセージが表示され、ジョブの属性は変更されません。
- JOBPTY パラメーターは、投入する特定のジョブに使用される優先順位の値を指定します。 投入する特定ジョブに関連したコマンドでは、JOBPTY パラメーターに、そのジョブの実際のスケジューリング優先順位を指定します。

- OUTPTY パラメーターでは、ジョブで作成されるすべてのスプール出力ファイルからの出力を生成するための優先順位を指定します。OUTPTY パラメーターに指定した優先順位の値は、各スプール・ファイルが出力のために処理する順序を決定します。ジョブで作成されるすべてのスプール・ファイルに対して、同じ値が適用されます。

スケジューリング優先順位として指定できるパラメーターの値は、0 から 9 までであり、1 が最高の優先順位、9 が最低の優先順位です。優先順位 0 が指定されているジョブはすべて、待機中の 1 から 9 までの優先順位を持つほかのすべてのジョブより先に、処理のためにスケジューリングされます。

優先順位パラメーターは、次のコマンドに指定できます。

表 12. 優先順位パラメーター

優先順位パラメーター	優先順位パラメーターを指定できるコマンド
JOBPTY	ADDJOBJS、BCHJOB、CHGJOB、CHGJOBD、CHGJOBJS、CRTJOBD、SBMJOB、SBMJOBJS
OUTPTY	ADDJOBJS、BCHJOB、CHGDKTF、CHGJOBD、CHGJOBJS、CHGJOB、CHGJOBD、CHGJOBJS、CHGPJ、CHGPRTF、CHGSPLFA、CRTDKTF、CRTJOBD、CRTPRTF、OVRDKTF、OVRPRTF、SBMJOB、SBMJOBJS
PTYLMT	CHGUSRPRF、CRTUSRPRF、RTVUSRPRF

指定可能な値

優先順位パラメーターには、使用するコマンドに応じて、次の値のうちの 1 つ以上が適用されます。

5
CRTUSRPRF コマンドに値を指定しないと、5 がユーザー・プロファイルの優先順位の上限として使用されるデフォルトの値になります。 その場合は、実行のために投入するどのジョブについても、ユーザーが指定できる最高優先順位は 5 となります。CRTJOBD コマンドでほかに優先順位が指定されていなければ、5 がジョブ・スケジューリング優先順位および出力優先順位の両方について、デフォルト値となります。

***SAME**
割り当てられた優先順位、または割り当て可能な最高優先順位は変更されません。

***JOBD**
ジョブのスケジューリング優先順位は、そのジョブの実行に使用するジョブ記述から得られます。

scheduling-priority:
0 から 9 までの値の 1 つを優先順位として指定してください。0 は最高の優先順位を、9 は最低の優先順位を表します。 優先順位 0 は、CHGJOB でのみ指定することができます。

SEV パラメーター**:**

重大度 (SEV) パラメーターは、重大度コードを指定します。

重大度 (SEV) パラメーターは、次のような重大度コードを指定します。
- エラー・メッセージに関連した重大度のレベルを記述する。
- ユーザーまたはプログラムにメッセージが戻されるようにするための最小重大度レベルを示す。
- バッチ・ジョブを打ち切る。
- 重大な構文エラーがあった場合にコマンドの処理を打ち切る。

注**:** 一部のコマンドの LOG パラメーターでも、ロギングのため (どのジョブ活動メッセージ、およびエラー・メッセージをジョブ・ログに記録するかを制御するため) に、この重大度コードを使用します。

重大度コードは、00 から 99 までの範囲内の 2 桁の値です。値が大きくなるほど、状態の重大度または重要度が高くなります。ユーザーに送られるメッセージの重大度コードは、そのメッセージが示している状態の重大度を示します。複数のメッセージが同じ重大度コードを持てます。事前定義済みのメッセージに対して重大度コードが指定されていない場合は、00 (通知のみ) を指定したものと見なされます。

重大度コードは、**メッセージをメッセージ記述追加 (ADDMSGD)** コマンドで定義する際、どのようなメッセージに対しても指定することができます。メッセージの重大度コードを変更するには、**メッセージ記述変更 (CHGMSGD)** コマンドを使用します。

システムとともに出荷されるすべての IBM 提供のメッセージには、IBM 定義の重大度コードが使用されています。

00 - 通知*:*
: この重大度のメッセージは、情報の提供のみを行うものです。エラーが検出されたわけではないので、応答の必要はありません。この種のメッセージは、ある機能が実行中であることや、実行が正常に完了したことなどを示します。

10 - 警告*:*
: この重大度のメッセージは、潜在的なエラー条件を示します。プログラムは、欠落している入力を補うなど、デフォルトの処置を取っている場合もあります。この操作の結果は、意図されたとおりのものと見なされます。

20 - エラー*:*
: エラーが検出されましたが、それに対しては自動回復プロシージャーが適用されたものと判断されるので、処理は続行されます。エラーの起きた入力の代わりにデフォルトの値が使用された可能性もあります。操作の結果は、必ずしも正しいとは限りません。実行された機能は、部分的にしか完了していない場合があります。例えば、リストのうちのある項目は正しく処理されたのに、別の項目は正しく処理されていない場合もあります。

30 - 重大エラー*:*
: 検出されたエラーが重大すぎて自動リカバリーが行えず、可能なデフォルト値がありません。このエラーがソース・データの中にある場合には、その入力レコード全体がスキップされます。プログラムの実行中にこの種のエラーが起きた場合は、プログラムの異常終了 (重大度 40) が生じます。操作結果は有効ではありません。

40 - プログラムまたは機能の異常終了
: 操作が終了しました。多くの場合、データが正しくないために操作が打ち切られたか、あるいは、ユーザーが操作を打ち切ったためです。

50 - ジョブの異常終了*:*
: ジョブが終了したか、開始されていませんでした。経路指定ステップが異常終了したか、その開始に失敗したか、ジョブ・レベルの機能が正しく実行されなかったか、あるいはジョブが打ち切られたなどの理由が考えられます。

60 - システム状況*:*
: この重大度のメッセージは、システム・オペレーターに対してのみ出されます。この種のメッセージは、入出力装置、サブシステム、またはシステム全体についての状況あるいは注意を示します。

70 - 装置保全性*:*
: この重大度のメッセージは、システム・オペレーターに対してのみ出されます。このメッセージ

は、ある装置が誤動作状態にあるか、または何らかの動作不能状態になっていることを示します。ユーザーがシステム操作を復元できる場合も、サービス技術員の援助が必要な場合もあります。

80 - システム警報:
この重大度のメッセージは、システム・オペレーターに対してのみ出されます。このメッセージは、システムを直ちに停止しなければならないほど深刻ではありませんが、何らかの予防措置を取らない限り、事態がさらに悪化する可能性があるということを警告するものです。

90 - システム保全性:
この重大度のメッセージは、システム・オペレーターに対してのみ出されます。このメッセージは、サブシステムまたはシステム全体が操作不能になった状態を示しています。

99 - 処置:
この重大度のメッセージは、応答の入力、または印刷用紙の変更などの、手操作による何らかの処置が必要であることを示しています。

関連タスク:

546 ページの『重大度コードの割り当て』
メッセージ記述追加 (ADDMSGD) コマンドでメッセージに割り当てる重大度コードは、そのメッセージの重大度を示します。

SPLNBR パラメーター**:**

1 つのジョブで複数のスプール・ファイルを作成し、すべてのスプール・ファイルが同じ名前である場合には、スプール・ファイル番号 (SPLNBR) パラメーターを使用します。ファイルには、1 から始まり、ジョブがファイルをオープンする順序に従って番号が付けられます。

ジョブ・ログは常に、ジョブの最後のファイルになります。

ファイル番号は、ジョブ内で各ファイルをオープンする (出力レコードを作成する) 際に生成され、システムにそのジョブまたはファイル (あるいはその両方) が存在している間は、システムにより使用されます。ファイルが 2 回以上オープンされたためにそれぞれに固有の名前が付いていない場合は、このファイル番号を使用して、CL コマンドの処理するファイル (または、ファイルがまだ完成していない場合にはレコード群) を指定します。

TEXT パラメーター**:**

TEXT パラメーターは、作成または変更するオブジェクトを簡潔に説明するユーザー定義の記述を指定します。

この記述には、最高 50 文字まで入れることができます。引用符付きストリング (すなわち、単一引用符で囲まれているストリング) の場合には、256 個の EBCDIC 文字のどれでも指定できます。ストリングにブランクおよびその他の特殊文字が含まれていない場合は、単一引用符は不要です。50 の文字位置のうち指定した記述に満たない桁は、ブランクで埋められます。

この記述は、**オブジェクト記述表示 (DSPOBJD)** コマンドを用いて指定のオブジェクトを画面に表示する場合に、IBM i オブジェクトを説明するために使用されます。ユーザーが表示できるのは、オブジェクト操作権を持っているオブジェクトのみです。

データベース・ソース・ファイルを用いて、何らかのタイプのオブジェクトを作成するコマンドの場合は、(デフォルトでは) ソース・ファイル・メンバーのテキストを、新たに作成するオブジェクトのテキストとして使用できます。例えば、CL プログラムを作成するのに**制御言語プログラム作成 (CRTCLPGM)** コマンド

を使用する場合、TEXT パラメーターにテキストを指定しなければ、ソース・ファイル (SRCFILE パラメーター) のソース・ファイル・メンバー (SRCMBR パラメーター) に指定されているテキストが、CL プログラムの記述テキストとして使用されます。

指定可能な値

TEXT パラメーターには、使用するコマンドに応じて、次の値のうちの 1 つ以上を使用することができます。

***SRCMBRTXT**
データベース・ソース・ファイルに基づいてオブジェクトを作成するコマンドの場合に限り、指定したソース・メンバーのテキストが使用されます。 装置またはインライン・ファイルをソース入力として使用する場合、またはソース入力を使用しない場合には、テキストはブランクになります。

***BLANK**
作成または変更するオブジェクトのユーザー記述はブランクのままです。

***SAME**
ユーザー定義のテキストは変更されません。

'記述**'** 作成または変更されるオブジェクトについての説明を指定してください。オブジェクトを説明するために、最高 50 文字を単一引用符で囲んで (ブランクおよびその他の特殊文字が含まれている場合には必須) 指定できます。単一引用符を 50 文字のうちの 1 文字として使用する場合、単一引用符の文字を表すには 1 重でなく 2 重の単一引用符 (' ') を使用する必要があります。

関連資料:

89 ページの『OBJTYPE パラメーター』
オブジェクト・タイプ (OBJTYPE) パラメーターは、オブジェクトが指定されたコマンドにより操作できる IBM i オブジェクトのタイプを指定します。

VOL パラメーター**:**

ボリューム (VOL) パラメーターは、テープ操作または光ディスク操作で使用するボリュームのボリューム識別コードを指定します。

1 テープ・ボリュームは、1 テープ・カートリッジまたは 1 リールで構成されます。光ディスク・ボリュームは、光ディスク・カートリッジの片面か、1 枚の CD-ROM で構成されます。光ディスク・カートリッジには両面があり、それぞれの面は別個のボリュームになります。

ボリューム識別コードは、そのテープまたは光ディスクを識別する識別コードであり、各テープまたは光ディスク (ボリュームのラベル域) に保管されています。ボリューム識別コードが抜けている場合や、順序が正しくない場合は、システム・オペレーターに照会メッセージが送られます。

テープ・ボリュームは、テープ装置ファイル・コマンドの VOL パラメーターに指定した識別コードと同じ順序で、かつ DEV パラメーターに指定した装置名の順序に従って、テープ装置にセットしなければなりません。ただし、テープが逆方向に読み取られる (COBOL でサポートされている機能) 場合は、VOL パラメーターに指定した順序とは逆の順序でボリュームをセットします。この場合にも、DEV パラメーターに指定する装置名の順序は正方向読み取りの順序です。

一般に、テープ・ボリュームの識別コードを指定するための規則として、6 桁までの文字および数字を任意に組み合わせて使用することができます。識別コードを単一引用符で囲めば、特殊文字も使用することができます。ただし、そのテープを IBM i オペレーティング・システム以外のオペレーティング・システムでも使用する場合は、識別コードの指定に関する要件も考慮しなければなりません。

光ディスク・ボリューム識別コードには、32 文字まで指定することができ、数字と大文字を一緒に使用することができます。光ディスク・ボリューム識別コードは、固有のものを指定してください。同じ識別コードを持つ 2 つの光ディスク・ボリュームを、同時にシステムに提示することはできません。

ラベル付きテープについては、次の規則が適用されます。

- 文字: 各ボリューム識別コードを 6 文字まで指定することができます。英字および数字をどのような順序で指定しても差し支えありません。
- 固有性: 複数のボリュームに同じ識別コードを付けることができます。複数のボリュームに同じ識別コードを使用したファイルがある場合は、システムは各ボリュームに書き込まれた順序番号を用いて、内部的に順序を追跡します。ただし、各ボリュームには、できる限り固有の識別コードを付けてください。
- 順序: 1 つの操作で (識別コードの異なる) 複数のボリュームを使用する場合には、それらのボリュームは、VOL パラメーターに指定したボリューム識別コードと同じ順序になっていなければなりません。

マルチボリューム・ファイル

操作に使用する複数のボリューム (テープ) が、すべて同一のボリューム識別コードを持っている場合は、使用するボリュームごとに 1 回ずつ、その識別コードを VOL パラメーターに指定しなければなりません。例えば、QGPL という名前の 3 つのテープを保管操作に使用する場合は、VOL(QGPL QGPL QGPL) を指定する必要があります。

テープ のマルチボリューム・ファイルを処理する際に、複数のテープ装置を使用する場合は、VOL パラメーターに指定した順序と同じ順序で、テープ・ボリュームをテープ装置にセットしなければなりません。例えば、5 個のボリュームと 3 台のテープ装置を使用する場合には、VOL1 は装置 1 に、VOL2 は装置 2 に、VOL3 は 3 に、VOL4 は装置 1 に、そして VOL5 は装置 2 にそれぞれセットします。

指定可能な値

***MOUNTED**
現在デバイスに配置されているボリュームが使用されます。

***NONE**
ボリューム識別コードは指定されません。

***SAME**
以前に指定されたボリューム識別コードは、変更されません。

***SAVVOL**
システムは、保管/復元ヒストリーの情報を用いて、最新の保管バージョンが、どのテープ・ボリュームに入っているかを判断します。 復元コマンドの DEV パラメーターに指定された装置が、当該オブジェクトの最新バージョンの保管されている装置と一致しない場合は、ユーザーにエラー・メッセージが戻され、その機能は終了します。コマンドに指定されている装置に誤ったボリュームが取り付けられている場合は、復元操作を始める前にまずセットしなければならないボリュームを示すメッセージが、システム・オペレーターに戻されます。

volume-identifier
1 つ以上のボリュームの識別コードを、装置に装着して使用する順序で指定します。 各テープのボリューム識別コードには、最大 6 文字の英数字が入ります。各光ディスク・ボリューム識別コードには、最大 32 文字が使用されます。複数の識別コードをリストするときは、ブランクが区切り記号として使用されます。

関連情報:

光ディスク装置プログラミング

WAITFILE パラメーター**:**

WAITFILE パラメーターを使用すると、プログラムがリソースを待機するかどうか、および待機する時間について指定できます。

WAITFILE パラメーターを使用すると、プログラムが以下を待機する最大秒数を指定できます。

- ファイルが開かれる場合に割り振られるファイル・リソース
- APPC 装置に対して呼び出し機能が出されたときのセッション資源
- 取得操作を実行してファイルを読み取る際、割り振られる装置

プログラムが待機しなければならない場合には、該当する資源が使用可能になるまで、またはこの待ち時間が切れるまで、待機状態になります。2 つ以上のファイル資源が必要で、それらのファイル資源がほかのシステム・ユーザーにより使用されているために使用できない場合は、プログラムはそれぞれの資源の割り振りを個別に待たなければならないことになります。このパラメーターに指定した最大値は、個々の待機状態ごとに適用されます。

待ち時間の長さは、このパラメーターで指定することも、オブジェクトに適用されるクラスのデフォルトの待ち時間を使用することもできます。指定した秒数の時間内にファイル資源が割り振られなかった場合は、プログラムにエラー・メッセージが戻されます。

割り振りの必要なファイル資源は、オープンされるファイルのタイプによって決まります。ファイル資源は、以下のものから成ります。

- スプールされない (SPOOL(*NO) が指定された) 装置ファイルの場合、ファイル資源には、ファイル記述および装置記述が含まれます。装置記述の割り振りが必要なために、該当の装置自体も使用可能になっていなければなりません。
- スプールされる (SPOOL(*YES) が指定された) 装置ファイルの場合、ファイル資源には、ファイル記述、指定した出力待ち行列、および、システム内でスプール・データのために使用される記憶域が含まれます。データはスプールされるので、装置記述 (および装置自体) が使用可能である必要はありません。
- データベース・ファイルの場合、ファイル資源は、ファイル・データおよびメンバー・データにより構成されています。ファイルの関連メンバーのパスはアクセスされません。したがって、システムがその関連メンバーを待機することはありません。アクセス・パスが使用可能でない場合 (例えばアクセス・パスが再作成中の場合) には、 WAITFILE の指定時間が切れる前にファイルのオープン例外エラーが起きることがあります。

  オブジェクト割り振り (ALCOBJ) コマンドを用いて、ファイルがオープンされる前に、特定のファイル資源を割り振ることができます。

APPC 装置による会話のために割り振られたセッション資源は、アプリケーション・プログラムが切り離し機能を出すか切り離し指示を受け取ってから別の呼び出し機能が出るまでの間に失われることがあります。セッション資源が失われた場合には、このパラメーターの値に基づいて、別のセッション資源の割り振りをシステムが待つ時間が決まります。

指定可能な値

***IMMED**
: プログラムは待機しません。ファイルのオープン時に、ファイル資源の即時割り振りが必要になります。

***CLS** クラス記述に指定されているデフォルトの待ち時間が、ファイル資源の割り振りの待ち時間として使用されます。

1-32767
ファイル資源の割り振りをプログラムが待つ場合の、許容時間を秒単位で指定してください。

テストおよびデバッグに使用するパラメーター値

IBM i オペレーティング・システムには、プログラムの実行時に行われる処理をプログラマーが監視できる機能が組み込まれています。

テスト機能を使用することにより、意図した通りに実行されていないプログラム内の命令を突き止めることができます。この機能は、バッチ・ジョブでも、またワークステーションからの対話式ジョブでも使用することができます。どの場合も、監視対象のプログラムはデバッグ・モード と呼ばれるテスト環境下になければなりません。

CL ソース・ステートメントを調べてエラーを見つけ出すのは容易なことではありませんが、テスト機能を使用すれば、調べる必要のある範囲を限定することができます。予期したものとは異なる出力が生成されることによって、エラーがあることが判明する場合がよくあります。このようなエラーの個所を突き止めるには、プログラマーが特定の位置 (ブレークポイント (停止点) と呼びます) でプログラムを停止して、プログラム中の変数の値が正しいかどうかを調べることが必要です。またプログラマーは、プログラムの実行を進める前に、必要に応じてそれらの変数の値を変更することもできます。

プログラマーは機械語の命令を理解している必要はなく、またテスト機能を使用するためにプログラムに特殊な命令を組み込む必要もありません。 IBM i のテスト機能を使用することによって、プログラマーは次のことを行うことができます。

- プログラムのソース・ステートメント中の名前を持つステートメントでプログラムの実行を停止する。
- プログラムまたはプロシージャーの停止が可能な任意の位置で、プログラムまたはプロシージャー変数に関する情報を表示する。プログラマーはプログラムまたはプロシージャーの処理を進める前に、その変数の値を変更することもできます。

ここでは、プログラム変数、基底ポインター、添え字、および修飾名パラメーター値について補足説明を行います。これらの値は、 **ブレークポイント追加 (ADDBKP)** コマンド、**トレース追加 (ADDTRC)** コマンド、**高水準言語ポインター変更 (CHGHLLPTR)** コマンド、**プログラム変数変更 (CHGPGMVAR)** コマンド、および**プログラム変数表示 (DSPPGMVAR)** コマンドで指定することができます。

他の統合言語環境 (ILE) 言語による情報のデバッグについては、該当する ILE 資料を参照してください。

関連タスク:

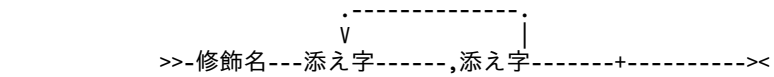
416 ページの『ILE プログラムのデバッグ』
統合言語環境 (ILE) プログラムをデバッグするには、ILE ソース・デバッガーを使用します。

441 ページの『オリジナル・プログラム・モデル・プログラムのデバッグ』
オリジナル・プログラム・モデル (OPM) プログラムをデバッグするには、テスト機能を使用します。これらの機能は、対話方式でもバッチ・ジョブの中でも使用できる 1 組のコマンドを介して実行することができます。

プログラム変数の記述**:**

プログラム変数は、特殊文字が含まれている場合には単一引用符で囲まなければなりません。プログラム変数名には最高 132 文字を使用できます。この文字数には、添え字、組み込みブランク、括弧、およびコンマも含まれます。ただし、特殊文字がある場合に名前を囲むために使用する単一引用符は、この文字数には含まれません。

プログラム変数



注: 最大 14 回まで反復可能です。

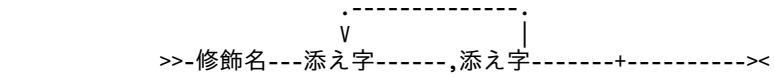
次に例を示します。

```
COUNTA
'VAR1(2,3)'
'A.VAR1(I,3,A.J,1)'
'VAR1 OF A(I,3,J OF A)'
'&LIBNAME'
```

基底ポインターの記述:

基底ポインターに特殊文字が含まれている場合には、その基底ポインターを単一引用符で囲まなければなりません。基底ポインター名には、最大 132 文字を指定することができます。この文字数には、添え字、組み込みブランク、括弧、およびコンマも含まれます。ただし、特殊文字がある場合に名前を囲むために使用する単一引用符は、この文字数には含まれません。

基底ポインター



注: 最大 14 回まで反復可能です。

次に例を示します。

```
PTRVAR1
'ABC.PGMPTR(5,B.I)'
```

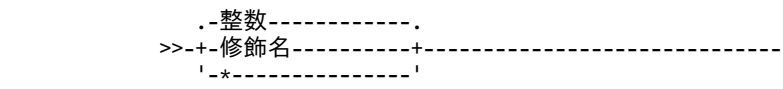
1 つの変数について複数の基底ポインターを指定する場合には、基底ポインターのリストを括弧で囲まなければなりません。複数の基底ポインターを指定する場合は、最初の基底ポインターから最後の基底ポインターまでを、変数の位置指定に使用する順序で指定しなければなりません。次の例では、基底ポインター PTR_1 が、変数の位置指定のため使用される最初の基底ポインターです。これは、基底ポインターとして宣言されている必要があるか、または、基底付き変数であってはなりません。ポインター A.PTR_2 (基底付きポインター変数として宣言されていなければなりません) の位置指定は、ポインター PTR_1 に含まれるアドレスを用いて行われます。ポインター配列 PTR_3 (これも基底付きとして宣言されていなければなりません) の位置指定は、ポインター A.PTR_2 の内容を用いて行われます。そして、最後のポインター配列内の指定された要素を用いて、該当の変数の位置が指定されます。例えば、次のようにします。

```
('PTR_1' 'A.PTR_2' 'PTR_3(1,B.J)')
```

添え字の記述:

整数には、必要に応じて先行符号 (プラスまたはマイナス) を付けた 1 から 15 桁までの数字が入ります。整数の添え字には小数点は使用できません。小数点を使用した場合は、システムが添え字の値を正しい数値として解釈せず、エラー・メッセージが出されます。

添え字



アスタリスク (*) は、配列プログラム変数の 1 つの次元のクロス・セクション表示を要求する場合に使用します。アスタリスクは、ブレークポイント追加 (ADDBKP) コマンド、トレース追加 (ADDTRC) コマンド、およびプログラム変数表示 (DSPPGMVAR) コマンドの PGMVAR キーワードの 1 次変数 (基底ポインターに基づいていない変数) でのみ、添え字として使用することができます。さらに、変数が複数の次元を持つ場合は、その添え字値の 1 つのみに対してアスタリスクを指定することができます。次に、配列のクロス・セクションの表示要求の例を示します。

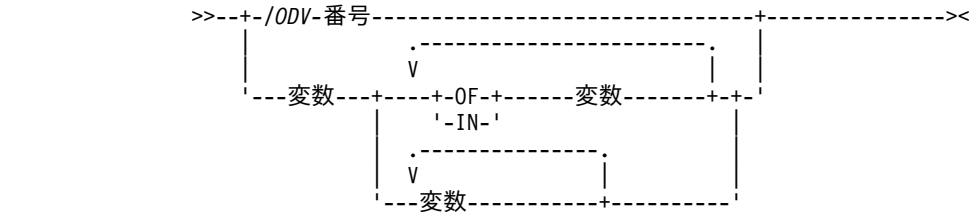
```
DSPPGMVAR  PGMVAR('X1(*,5,4)')
```

この例の場合、画面には、配列の要素のうち、2 番目の添え字が 5 に等しく、3 番目の添え字が 4 に等しいすべての要素が表示されます。

修飾名の記述:

高水準言語によっては、同じ完全修飾名を持つ複数の変数を宣言できる場合があります (ただし、通常は、宣言した後で、高水準言語プログラムの中でこれらの変数を参照することはできません)。IBM i のテスト機能コマンドを使用してこのような変数を参照しようとした場合には、システムは、それらの変数の 1 つを選択して処理に使用します。重複した完全修飾名を選択しても、エラーが通知されることはありません。

修飾名



注: 最大 19 回まで反復可能です。

修飾名の記述に関する規則

- ODV 番号は、斜線 (/) の後に、1 から 4 桁の 16 進数字 (0 から 9、および A から F) を付けたものです。
- 変数名は、プログラムの中の変数の名前でなければなりません。この名前は、高水準言語での指定方法と同じ方法で指定しなければなりません。高水準言語によっては、ユーザーがプログラムのソースで指定した変数名に修飾子を加えて、修飾変数名を生成する場合があります。変数名の詳細については、該当の高水準言語解説書を参照してください。
- 変数名と特殊語 OF および IN との間はブランクで区切らなければなりません。
- 名前を修飾するためにピリオドを使用する場合には、ピリオドと変数名との間にブランクを置いてはなりません。
  - 変数名の指定順序は次の規則に従っていなければなりません。
    - 途中にピリオドのない修飾名の場合は、変数名は、その構造の中でレベルの低いものから高いものへの順に指定されているものと見なされます。
    - 1 つ以上のピリオドを含む修飾名の場合は、変数名は、その構造の中でレベルの高いものから低いものの順に指定されているものと見なされます。
  - 修飾名に ODV 番号を使用しない場合は、変数がプログラムの中で固有のものとして識別されるようにするために、十分な数の修飾用変数名を指定しなければなりません。修飾名が単純名 (1 つの変数名のみが指定されたもの) であっても、複数の修飾用変数名を持つ名前であっても、次の条件のいずれかが満たされている場合は、その変数はプログラムの中で固有のものとして識別されます (これ

らの条件では、1 つのプログラム変数を固有のものとして選択するためには、IBM i のテスト機能コマンドを使用する方が、高水準言語プログラムで指定するよりも多くの修飾用変数名の指定を必要とすることがあります)。

- プログラムの中に、指定した修飾変数名に一致する修飾用変数名のセットを持つ変数がただ 1 つのみ存在する場合は、その変数は固有のものと見なされます。
- 変数が、指定した修飾用変数名とまったく同じ修飾用変数名のセットを持つ場合は、その変数はプログラムの中で固有のものとして識別されます。修飾子の完全なセットを指定した状態を、変数名の完全修飾 といいます。 修飾名については、完全修飾 が一致する変数が選択されます。修飾名と同じ名前をもつ変数があっても、指定されていない余分の修飾用変数が加えられていれば、それは選択されません。

制御言語要素

制御言語 (CL) には、文字セットと値、式、組み込み関数、およびコマンド内での命名などの要素が含まれます。

CL 文字セットおよび値

CL では、EBCDIC 文字セット、特殊文字、および事前定義値 と呼ばれる IBM 定義の固定値を使用できます。

文字セット:

制御言語 (CL) では、拡張 2 進化 10 進コード (EBCDIC) 文字セットを使用します。

制御言語で使用される文字と EBCDIC 文字セットの文字との関係をわかりやすくするために、制御言語では EBCDIC 文字を次のようなカテゴリーに分類するものとします。

カテゴリー	含まれる文字
英字[1]	26 個の英字 (A から Z)、$、#、および @[3]
数字	10 個の数字 (0 から 9)
英数字[1,2]	A から Z、0 から 9、$、#、@[3]、ピリオド (.)、および下線 (_)
特殊文字	その他のすべての EBCDIC 文字

注:

1. 小文字 (a から z) は使用できますが、通常、システムによって対応する大文字に変換されます。引用符で囲まれた文字ストリングまたは注記に含まれている文字には、小文字の変換が行われません。コマンド定義のパラメーターで、その TYPE パラメーターに文字 (*CHAR) 属性またはパス名 (*PNAME) 属性が指定されている場合、およびその CASE パラメーターに大/小文字混合 (*MIXED) 属性が指定されている場合には、そのパラメーターの値に指定された小文字について変換が行われません。
2. 下線 (_) は、英数字の結合子であり、IBM i CL では、語または英数字を連結して名前 (例えば PAYLIB_01) を作るために使用することができます。このような下線の用法は、他の高水準言語では無効な場合もあります。
3. 拡張英字の $、#、および @ は、EBCDIC CCSID によって変わる文字です。 CL は CL コマンドの処理で CCSID 37 を想定するため、$ は X'5B'、# は X'7B'、@ は X'7C' と想定します。 EBCDIC ジョブ CCSID が 37 以外の場合には、これら 3 つの 16 進コード・ポイントの 1 つにマップされる文字を使用する必要があります。

最初の 3 つのカテゴリーに含まれる文字は、引用符付き文字ストリングの中でも、引用符なしの文字ストリングの中でもまた、注記の中でも、CL 名 (コマンド、ラベル、キーワード、変数、および IBM i オブジェクトの名前など) の中でも使用できます。最後のカテゴリーに属する特殊文字は、引用符付き文字スト

リングおよび注記の中でのみ使用することができ、引用符なしの文字ストリングの中では使用することができません。ただし、一部の文字については、CL コマンドの中の所定の個所でコーディングすれば、構文上の特殊な意味に使用できます。

関連概念:

『特殊文字の使用法』
特殊文字は、このような特殊な用途で使用するか、あるいは引用符付き文字ストリングまたは注記内でしか使用できません。

CL* コマンドでの *2* バイト文字テキスト*:

記述テキストが使用できる CL コマンドの中なら、どこにでも 2 バイト文字データを使用することができます。

2 バイト文字テキストは次の方法で入力してください。

1. 2 バイト文字テキストの始めに、まず単一引用符 (') を入力します。
2. シフトアウト文字を入力します。
3. 2 バイト文字テキストを入力します。
4. シフトイン文字を入力します。
5. 2 バイト文字テキストの終わりに単一引用符 (') を入力します。

例えば、ABC という 2 バイト文字のリテラルを入力するには、以下を入力します。ここで、  はシフトアウト文字を表し、  はシフトイン文字を表します。

'  ABC  '

記述の表示および印刷が正しくできるようにするためには、 1 つのオブジェクトの 2 バイト文字テキスト記述の長さを、 2 バイト文字 14 文字にシフト制御文字の 2 文字分を加えた長さに限定してください。

特殊文字の使用法**:**

特殊文字は、このような特殊な用途で使用するか、あるいは引用符付き文字ストリングまたは注記内でしか使用できません。

EBCDIC 特殊文字は、CL の中でさまざまな用途に使用されます。それらの文字がもっとも頻繁に使用されるのは、区切り文字、および式の中での記号演算子としてです。 制御言語コマンドで使用する場合の各特殊文字の意味は、次の表に示すとおりです。

表 *13.* 区切り文字

名前	記号	意味
単一引用符	' '	単一引用符区切り文字は、引用符付き文字ストリング (定数) の始めと終わりを示します。
注記の始めと終わり	/* */	注記の始めと終わりを示します。
ブランク	b [1]	コマンドの構成要素 (ラベル、コマンド名、およびそのパラメーター) を区切るため、およびリスト中の値を区切るための基本的な区切り文字。
コロン	:	コマンド・ラベルの終わりを示す区切り文字。時刻の値の構成要素を区切るためにも使用されます。[3]

表 13. 区切り文字 (続き)

名前	記号	意味
コンマ	,	多くの国で数値の小数点として使用されます。日付の値の構成要素を区切ります。[2]
左右の括弧	( )	リストおよびパラメーター値のグループ化のための区切り文字、および式の順序を識別するための文字。
ピリオド	.	小数点。文書名およびフォルダー名の名前と拡張子を区切るために使用されます。また、日付の値の構成要素を区切るためにも使用されます。 [2]
引用符	" "	引用符付きオブジェクト名の始め。
斜線	/	修飾名またはパス名の各部分を連結します。
2 重斜線	//	ジョブ・ストリーム中の BCHJOB、ENDBCHJOB、および DATA コマンドの 1 桁目および 2 桁目で使用する識別文字。インライン・データ・ファイルでのデフォルトの区切り文字としても使用されます。

注:

1. [b] は、本書のオンライン・バージョンではブランク・スペースが表示されないため、他の方法で明確に示すことができない場合にのみ、ブランク・スペースを表すために使用されます。
2. ジョブ日付区切り文字の値として同じ文字が指定されている場合に限り有効です。
3. ジョブ時刻区切り文字の値として同じ文字が指定されている場合に限り有効です。

関連概念:

104 ページの『文字セット』
制御言語 (CL) では、拡張 2 進化 10 進コード (EBCDIC) 文字セットを使用します。

『記号演算子』
さまざまな文字が、CL コマンドで記号演算子として使用できます。

関連資料:

113 ページの『式の中の演算子』
式の中で使用する演算子は、その式の中のオペランドに対して行う演算、または、オペランドとオペランドとの関係を指定します。

記号演算子:

さまざまな文字が、CL コマンドで記号演算子として使用できます。

次の文字は、CL コマンドで記号演算子として使用されます。

表 14. CL コマンドで使用する記号演算子

名前	記号	意味
AND	&	AND を表す記号論理演算子。
アスタリスク	*	乗算演算子。名前の最後の文字として使用した場合には、総称名を表します。ストリングの最初の文字として使用した場合には、IBM i の予約値 (事前定義パラメーター値および式演算子) を表します。
連結	|>、|<、および || [3]	文字ストリング演算子 (2 つの値を結合することを示します)。
等しい	=	等しい ことを表す記号関係演算子。
より大きい	>	より大きい ことを表す記号関係演算子。

表 14. CL コマンドで使用する記号演算子 (続き)

名前	記号	意味
より小さい	<	より小さい ことを表す記号関係演算子。
負符号 (ハイフン)	-	減算演算子、コマンド継続文字、および負符号。日付の値の構成要素を区切ります。[1]
否定	¬ [2]	否定を表す記号関係演算子。
OR	| [3]	OR を表す記号論理演算子。
プラス	+	加算演算子、コマンド継続文字、および正符号。
斜線	/	除算演算子。日付の値の構成要素を区切ります。[1] 修飾名の各部分を区切るためにも使用されます。

注:

1. ジョブ日付区切り文字の値として同じ文字が指定されている場合に限り有効です。
2. 一部の文字セット (各国間共通文字セットを含む) では、¬ 文字の代わりに文字 ^ が使用されます。このような文字セットでは、論理 NOT 演算子として ^ または *NOT を使用することができます。
3. ある種の文字セット (各国共通文字セットを含む) では、| の代わりに ! が使用されます。このような文字セットでは、論理 OR 演算子として ! または *OR を使用することができ、また連結演算子として !! または *CAT を使用することができます。

注**:** また、記号演算子は、組み合わせて使用することもできます。

記号演算子は、次のように使用することもできます。

表 15. 記号演算子: その他の使用方法

名前	記号	意味
アンパーサンド	&	ストリングの最初の文字として使用した場合、そのストリングが CL 変数名であることを示します。
パーセント	%	ストリングの最初の文字として使用した場合、そのストリングが組み込みシステム関数であることを示します。
疑問符	?	コマンド名またはキーワード名の前に使用した場合、プロンプトを要求することを指定します。

関連概念:

105 ページの『特殊文字の使用法』
特殊文字は、このような特殊な用途で使用するか、あるいは引用符付き文字ストリングまたは注記内でしか使用できません。

関連資料:

64 ページの『式』
式 とは、いくつかの定数、変数、または組み込み関数を演算子によって区切ったものであり、1 つの値を生むものです。

113 ページの『式の中の演算子』
式の中で使用する演算子は、その式の中のオペランドに対して行う演算、または、オペランドとオペランドとの関係を指定します。

事前定義値**:**

事前定義値は、制御言語 (CL) の中での用法があらかじめ定められている IBM 定義の固定値であり、IBM i オペレーティング・システムにおける予約値と見なされます。

事前定義値は、アスタリスク (*) の後に 1 つの語または略語を付けたもので、例えば *ALL や *PGM などがこれにあたります。事前定義値で * を使用する目的は、オブジェクト名などのようなユーザー指定の値と混同されるのを防ぐためです。事前定義値には、それぞれ、1 つ以上のコマンド・パラメーター内での所定の用法があります。これについては、各コマンドの項で詳しく説明されています。

事前定義値には、*EQ および *AND などのように式の中で演算子として使用されるものもあります。事前定義値 *N はヌル値を指定するためのもので、オプション・パラメーターを表すために使用することができます。ヌル値 (*N) は、値の指定されていないパラメーターの位置を示します。これを使用することにより、後続のパラメーターを定位置形式で入力することができます。文字 *N を文字の値として (ヌル値としてではなく) 指定するためには、受け渡しするストリングを単一引用符で囲まなければなりません ('*N')。また、CL プログラムの実行時に CL プログラム変数の中に *N が現れた場合も、常にヌル値として扱われます。

CL コマンドで使用する式

文字ストリング式は、コマンド定義オブジェクトで EXPR(*YES) が定義されているどのようなパラメーター、要素、または修飾子にも使用することができます。

変数変更 (CHGVAR) コマンドおよび **IF** コマンドでは、どのような式でも単一のパラメーターとして使用できます。単純な式の形式としては、1 つの定数、1 つの変数、または 1 つの組み込み関数があります。通常の式は、2 つのオペランドと 1 つの演算子によって構成されます。演算子は、その式でどのような評価 (演算) を行うかを指定します。複数の式を組み合わせて複合式とすることもできます。

CL プログラムでは次のタイプの式を使用することができます。

- 算術式 (&VAR + 15)
- 文字ストリング (SIX || TEEN)
- 関係式 (&VAR > 15)
- 論理式 (&VAR & &TEST)

複合式は、複数のオペランド、それらのオペランドに対して行う演算を指定する演算子、および、オペランドと演算子とをまとめる括弧によって構成されます。オペランドとオペランドとの間には演算子を 1 つしか使用できません。ただし、符号として 10 進数値の前につく + および - 符号、ならびに論理式の中で使用される *NOT 演算子を除きます。複合式は、最外部 (必須) レベルを含む、5 を超える括弧のネスト・レベルを持つことはできません。

算術式と文字ストリング式は、関係演算子および論理演算子とともに使用すれば、複合式の中で同時に使用することができます。例えば、(A=B&(1+2)=3) となります。1 つの関係式の中で、1 組の算術式または 1 組の文字ストリング式を比較できます。また、1 つの論理式の中で、いくつかの関係式を使用することもできます。

関連資料:

64 ページの『式』
式 とは、いくつかの定数、変数、または組み込み関数を演算子によって区切ったものであり、1 つの値を生むものです。

算術式:

| 算術式中のオペランドは、10 進定数、CL 10 進変数、CL 整変数、あるいは数値の結果を返す CL 組み込
| み関数のいずれかでなければなりません。この CL 組み込み関数には、%BINARY、%CHECK、
| %CHECKR、%SCAN、%DEC、%INT、%UINT、%LEN、%SIZE、および %PARMS が含まれます。

オペランドの間には、算術演算子 (記号形式のみ) がなければなりません。算術式の演算結果はすべて 10 進数値であり、その結果は CL 変数に保管することができます。

注: 除算演算子 (/) の場合、その前のオペランドが変数名である場合には、演算子の前にブランクを 1 つ付けなければなりません。(例えば、`&A /2` とせずに、`&A/2` とします。) その他の演算子の場合は、前後のブランクの使用は任意です。

算術オペランドは、符号付きでも符号なしでも構いません。すなわち、各オペランドは (数値定数でも、CL 10 進変数または CL 整変数でも)、その直前に正符号 (+) または負符号 (-) を付けることができます。ただし、符号は必須ではありません。符号として使用する場合は、+ や - とその値の間にブランクは入りません。例えば、23.7 という 10 進定数は、+23.7 もしくは -23.7 (符号付き)、または 23.7 (符号なし) と表すことができます。

下記に算術式の例をいくつか挙げます。

```
| (&A + 1)    (&A + &B -15)
| (&A - &F)   (&A+&B-15)
| (&A + (-&B))
| (%SCAN('/' &STRING) + 1)
| (%DEC(&STRING) - 1)
| (%SIZE(&CHAR1) + %SIZE(&CHAR2))
| (%PARMS() - 1)
```

ある行のブランク以外の最後の文字が正符号 (+) または負符号 (-) である場合には、それは算術演算子ではなく継続文字として扱われます。

文字ストリング式:

文字ストリング式のオペランドは、引用符付きもしくは引用符なしの文字ストリング、CL 文字変数、サブストリング (%SUBSTRING または %SST) 組み込み関数、トリム (%TRIM、%TRIML、または %TRIMR) 組み込み関数、変換 (%CHAR、%LOWER、または %UPPER) 組み込み関数のいずれかでなければなりません。

各変数または組み込み関数に結び付いている値は、いずれも文字ストリングでなければなりません。連結の結果も文字ストリングになります。

文字ストリング式では、次の 3 つの演算子を使用できます。これらの演算子はいずれも文字ストリングを連結 (結合) するものですが、それぞれの機能に若干の相違があります。すなわち、

***CAT (**連結、記号 `||`**)** 演算子

*CAT 演算子は次の例のように 2 つの文字ストリングを連結します。例えば、`ABC *CAT DEF` は `ABCDEF` となります。

ブランクは、次の例のように連結の結果に含まれます。例えば、`'ABC ' *CAT 'DEF'` は `'ABC DEF '` となります。

***BCAT (**ブランク挿入連結、記号 `|>`**)** 演算子

*BCAT 演算子の場合には、最初の文字ストリングの末尾ブランクはすべて切り捨てられ、1 つのブランクが挿入された上で、2 つの文字ストリングが連結されます。2 番目のオペランドの先行ブランクは切り捨てられません。以下にその例を示します。

`ABC *BCAT DEF` は `ABC DEF` となります

`'ABC ' *BCAT DEF` は `'ABC DEF'` となります

***TCAT (**末尾ブランク切り捨て連結、記号 `|<`**)** 演算子

*TCAT 演算子の場合には、最初の文字ストリングのすべての末尾ブランクが切り捨てられた上で、2 つの文字ストリングが連結されます。2 番目のオペランドの先行ブランクは切り捨てられません。以下にその例を示します。

```
ABC *TCAT DEF は ABCDEF となります
'ABC ' *TCAT DEF は 'ABCDEF' となります
ABC *TCAT ' DEF' は 'ABC DEF' となります
 'ABC '*TCAT ' DEF' は 'ABC DEF' となります
```

連結演算子の両側のブランクはすべて無視されますが、事前定義値の演算子 (*CAT、*BCAT、または *TCAT) の前後には、それぞれ少なくとも 1 つはブランクがなければなりません。式の中で複数のブランクを使用する場合には、引用符付き文字ストリング (単一引用符で囲んだ文字ストリング) を使用しなければなりません。

関連資料:

58 ページの『文字ストリング』
文字ストリングとは、任意の EBCDIC 文字 (英数字および特殊文字) からなる文字ストリングであり、1 つの値として使用されるものです。

例*:* 文字ストリング式*:*

この例では、変数と、それに対応する文字ストリング式を示します。

各変数の値は以下のものであるとします。

変数	値
&AA	'GOOD '
&BB	'REPLACEMENT'
&CC	'ALSO GOOD'
&DD	'METHOD'
&EE	'nice'

次の表は、文字ストリング式とその結果を示しています。

式	結果
(&AA || &BB)	GOOD REPLACEMENT
(&AA||&BB)	GOOD REPLACEMENT
(&AA *CAT &BB)	GOOD REPLACEMENT
(&CC |> &DD)	ALSO GOOD METHOD
(&CC *BCAT &DD)	ALSO GOOD METHOD
(A *CAT MOUSE)	AMOUSE
('A ' *CAT MOUSE)	A MOUSE
(FAST *CAT MOUSE)	FASTMOUSE
('FAST ' *BCAT MOUSE)	FAST MOUSE

式	結果
('FAST ' *TCAT MOUSE)	FASTMOUSE
('AB' *CAT 'CD')	ABCD
('AB' *BCAT 'CD')	AB CD
('AB' *TCAT 'CD')	ABCD
(%SST(&AA 1 5)) *CAT (%SST(&BB 3 5))	GOOD PLACE
(%SST(&CC 1 9)) *BCAT (%SST(&BB 3 5))	ALSO GOOD PLACE
(&AA *CAT ' TIME')	GOOD TIME
(&CC *BCAT TIME)	ALSO GOOD TIME
(%TRIM(&CC 'ALOS ') *BCAT %TRIML(&BB 'ER'))	GOOD PLACEMENT
(%UPPER(&EE) *BCAT %LOWER(&BB))	NICE replacement

例*:* 文字ストリングおよび変数の使用*:*

この例は、いくつかの文字変数および文字ストリングを連結して、ワークステーション・オペレーターへのメッセージを作成する方法を示しています。

この例では、変数 &DAYS および &CUSNUM は、10 進変数ではなく文字変数として宣言されているものとします。

```
DCL     VAR(&MSG)TYPE(*CHAR)     LEN(100)
         *
         *
CHGVAR      &MSG   ('Customer' *BCAT &CUSNAMD  +
              *BCAT 'Account Number' *BCAT +
              &CUSNUM *BCAT 'is overdue by' +
              *BCAT &DAYS *BCAT 'days.')
```

各変数が該当の値で置き換えられた結果、次のようなメッセージができあがります。

```
Customer ABC COMPANY Account Number 12345
is overdue by 4 days.
```

変数 &DAYS および &CUSNUM が 10 進変数として宣言されていたとすれば、さらに別の 2 つの CHGVAR コマンドを使用して 10 進変数を文字変数に変更しない限り、連結を行うことはできません。例えば、プログラムの中で &DAYSALPH および &CUSNUMALPH という 2 つの文字変数も宣言されていたとすれば、CHGVAR コマンドは次のようになります。

```
CHGVAR  &DAYSALPH  &DAYS

CHGVAR  &CUSNUMALPH  &CUSNUM
```

この場合は、&MSG 用のすべての変数および文字ストリングを連結する CHGVAR コマンドでは、&DAYS および &CUSNUM の代わりに、新しい変数である &DAYSALPH および &CUSNUMALPH が指定されることになります。

もしくは、組み込み関数 %CHAR を使用して、&DAYS および &CUSNUM を以下のように連結式内の文字フォーマットに変換することもできます。

```
CHGVAR      &MSG   ('Customer' *BCAT &CUSNAMD  +
                 *BCAT 'Account Number' *BCAT +
                 %CHAR(&CUSNUM) *BCAT 'is overdue by' +
                 *BCAT %CHAR(&DAYS) *BCAT 'days.')
```

関係式:

関係式のオペランドは、算術式または文字ストリング式でもよく、また論理定数または論理変数であっても構いません。

関係式では、2 つのオペランドと 1 つの演算子しか使用できません。この 2 つのオペランドは、データ・タイプ (算術データ、文字ストリング・データ、または論理データ) が同じでなければなりません。関係式の演算結果は、'0' または '1' の論理値になります。

関係演算子は、記号 (=、>、<、>=、<=、 ¬ =、 ¬ >、 ¬ <) またはそれらに対応する予約値 (*EQ、*GT、*LT、*GE、*LE、*NE、*NG、*NL) で指定することができます。それぞれの意味については、『式の中の演算子』の表を参照してください。

演算において 2 つの文字フィールドの長さが同じでない場合には、短い方のフィールドの右側にブランクが追加されて、両者の長さが同じになるように調整されます。

算術フィールドは算術的に比較され、文字フィールドは EBCDIC 照合順序に従って比較されます。

論理フィールドを比較する場合、論理値 1 ('1') は論理値 0 ('0') より大きいと見なされます。これを記号で表せば ('1' > '0') となります。

次に、関係式の例を示します。

```
(&X *GT 25)
(&X > 25)
(&X>25)

(&NAME *EQ GSD)
(&NAME *EQ &GSD)
(&NAME *EQ 'GSD')
(&BLANK *EQ ' ')
```

論理式:

論理式のオペランドは、論理演算子で区切られた関係式、論理変数、または定数で構成されています。

これらのオペランドを 2 つ以上組み合わせて、論理式の中で 2 つ以上の式を作ることができます。この場合のネストのレベル数は最高 5 レベルまでです。論理式の演算結果は '0' または '1' のいずれかで、これは他の式の一部として使用することも、また論理変数に入れることもできます。

オペランド相互間の関係を指定するために使用する論理演算子は、*AND と *OR (予約値の場合) および & と | (記号の場合) です。AND 演算子は、両方のオペランド (演算子の両側のオペランド) が所定の値を持っている場合に、はじめて特定の結果が得られることを示します。OR 演算子は、いずれか一方のオペランドで結果が決まることを示します。

論理演算子 *NOT (または ¬) は、論理変数または論理定数を否定するために使用されます。*NOT 演算子は常に、*AND または *OR 演算子より先に評価されます。すなわち、*NOT 演算子の後に続くすべてのオペランドは、オペランド相互間の関係より先に評価されます。

以下に論理式の例を示します。

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C < 1) & (&TIME *GT 1430))
((&C<1)&(&TIME>1430))

(&A *OR *NOT &B)
(&TOWN *EQ CHICAGO *AND &ZIP *EQ 60605)
```

IF コマンドで使用される論理式の 2 つの例を下に示します。

```
IF &A CALL PROG1
IF (&A *OR &B) CALL PROG1
```

式の中の演算子:

式の中で使用する演算子は、その式の中のオペランドに対して行う演算、または、オペランドとオペランドとの関係を指定します。

演算子には 4 種類のタイプがあり、それぞれが、次の式のタイプに対応しています。

- 算術演算子 (+、 -、 *、 /)
- 文字演算子 (||、 |>、 |<)
- 論理演算子 (&、 |、 ¬)
- 関係演算子 (=、 >、 <、 >=、 <=、 ¬=、 ¬>、 ¬<)

各演算子は、その演算子を使用している式のオペランドとオペランドの間になければなりません。例えば、(&A + 4) のようになります。演算子は、事前定義値 (*EQ など) または記号 (= など) で指定することができます。

- 事前定義値の演算子の前後にはブランクが必要です。

  (&VAR *EQ 7)
- 除算演算子 (/) の場合を除いて、記号演算子の前後にはブランクはなくても構いません。例えば、(&VAR=7) と (&VAR= 7) はいずれも有効です。

  除算演算子を変数名の後に付ける場合には、その除算演算子の前にブランクがなければなりません。例えば、(&VAR / 5) と (&VAR /5) はいずれも有効ですが、(&VAR/5) は無効です。

次の表は、事前定義値とそれに対応する記号を示すとともに、それらが 4 種類の演算子のどれに属するものであるかを示しています。これらの値および記号は、引用符なしのストリングの中で、演算子以外の目的で使用してはなりません。

表 *16. 4* 種類の演算子を表す事前定義値および記号

事前定義値	事前定義記号	意味	タイプ
	+	加算	算術演算子
	-	減算	算術演算子
	*	乗算	算術演算子
	/	除算	算術演算子
*CAT	|| [1]	連結	文字ストリング演算子
*BCAT	|> [1]	ブランク挿入を伴う連結	文字ストリング演算子
*TCAT	|< [1]	ブランク切り捨てを伴う連結	文字ストリング演算子
*AND	&	AND	論理演算子

表 16. 4 種類の演算子を表す事前定義値および記号 (続き)

事前定義値	事前定義記号	意味	タイプ
*OR	|[1]	OR	論理演算子
*NOT	¬ [2]	NOT	論理演算子
*EQ	=	等しい	関係演算子
*GT	>	より大きい	関係演算子
*LT	<	より小さい	関係演算子
*GE	>=	より大きいまたは等しい	関係演算子
*LE	<=	より小さいまたは等しい	関係演算子
*NE	¬= [2]	等しくない	関係演算子
*NG	¬> [2]	より大きくない	関係演算子
*NL	¬< [2]	より小さくない	関係演算子

注:

1. 一部の国の文字セットおよび各国間共通文字セットでは、| (16 進 4F) の代わりに ! (感嘆符) が使用されます。このような文字セットでは、OR 演算子として ! または *OR を使用することができ、また連結に !! または *CAT、!> または *BCAT、!< または *TCAT を使用することができます。
2. 一部の国の文字セットおよび各国間共通文字セットでは、¬ (16 進 5F) の代わりに * が使用されます。このような文字セットでは、NOT 演算子として * または *NOT を使用することができます。

関連概念:

105 ページの『特殊文字の使用法』
特殊文字は、このような特殊な用途で使用するか、あるいは引用符付き文字ストリングまたは注記内でしか使用できません。

106 ページの『記号演算子』
さまざまな文字が、CL コマンドで記号演算子として使用できます。

関連タスク:

450 ページの『条件付きブレークポイントの追加』
デバッグ中のプログラムに条件付ブレークポイントを追加するには、**ブレークポイントの追加 (ADDBKP)** コマンドを使用してステートメントおよび条件を指定します。

式を評価する際の演算子の優先順位:

1 つの式に複数の演算子がある場合には、式の中の各演算子に応じて、一定の順序で式が評価されます。

式の評価の順序を変えるためには、括弧を使用することができます。次の表は、符号付きの 10 進数の値も含めて、式の中で使用されるすべての演算子の優先順位を示しています。

表 17. 演算子の優先順位

優先順位	演算子
1	符号 (+ および -) 付きの 10 進数値、*NOT、¬
2	* , /
3	+、- (2 つのオペランドの間で使用されている場合)
4	*CAT、||、*BCAT、|>、*TCAT、|<
5	*GT、*LT、*EQ、*GE、*LE、*NE、*NG、*NL、>、<、=、>=、<=、 ¬=, ¬>, ¬<
6	*AND、&

表 17. 演算子の優先順位 (続き)

優先順位	演算子
7	*OR、|

優先順位 1 は最高優先順位であり (符号付き数値が最初に評価されます)、優先順位 7 が最低優先順位です (OR 関係が最後に評価されます)。 1 つの式の中に優先順位の異なる複数の演算子がある場合には、演算子の優先順位に従って演算が行われます。

1 つの式の中に同じ優先順位を持つ複数の演算子がある場合には、演算はその式の中で左から右への順序で行われます。 括弧は、演算の実行順序を制御するためにいつでも使用することができます。括弧内の式の値は、内側の括弧から外側の括弧の順に評価され、また、一対の括弧の中では前に説明した優先順位に従って評価されます。

CL の組み込み関数

制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

CL は、次の組み込み関数を提供します。

- %ADDRESS
- %BINARY
- %CHAR
- %CHECK
- %CHECKR
- %DEC
- %INT
- %LEN
- %LOWER
- %OFFSET
- %PARMS
- %SCAN
- %SIZE
- %SUBSTRING
- %SWITCH
- %TRIM
- %TRIML
- %TRIMR
- %UINT
- %UPPER

関連概念:

136 ページの『固有名に関するその他の規則』
オブジェクトの命名については、特殊文字 (付加文字) に関して以下のような追加の規則があります。

関連資料:

243 ページの『%SUBSTRING 組み込み関数』
サブストリング組み込み関数 (%SUBSTRING または %SST) は、既存の文字ストリングのサブセットとしての文字ストリングを作成します。

246 ページの『%SWITCH 組み込み関数』
スイッチ組み込み関数 (%SWITCH) は、8 個のスイッチの 1 つまたは複数のスイッチを、該当のジョブで既に設定されている 8 つのスイッチの値と比較し、 '0' または '1' の論理値を返します。

248 ページの『%TRIM 組み込み関数』
1 つのパラメーターを指定してトリム組み込み関数 (%TRIM) を使用すると、先行ブランクおよび末尾ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定してトリム組み込み関数 (%TRIM) を使用すると、*characters-to-trim* パラメーター内にある先行文字と末尾文字がすべて除去された文字ストリングが生成されます。

249 ページの『%TRIML 組み込み関数』
1 つのパラメーターを指定して左方トリム組み込み関数 (%TRIML) を使用すると、先行ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定して左方トリム組み込み関数 (%TRIML) を使用すると、*characters-to-trim* パラメーター内にある先行文字がすべて除去された文字ストリングが生成されます。

250 ページの『%TRIMR 組み込み関数』
1 つのパラメーターを指定して右方トリム組み込み関数 (%TRIMR) を使用すると、末尾ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定して右方トリム組み込み関数 (%TRIMR) を使用すると、*characters-to-trim* パラメーター内にある末尾文字がすべて除去された文字ストリングが生成されます。

251 ページの『%UINT 組み込み関数』
%UINT は、文字、論理、10 進数、または整数データを符号なし整数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

253 ページの『%UPPER 組み込み関数』
%UPPER 組み込み関数は、指定された引数と同じ長さの文字ストリングを、それぞれの小文字を対応する大文字に置き換えて戻します。

コマンド内での命名

名前の指定に使用できる文字は、制御言語 (CL) で指定する名前のタイプによって決まります。

名前のタイプによっては、使用できる文字に制約があります。名前のタイプには、*NAME、*SNAME、および *CNAME があります。

注: コマンド定義を使用してコマンドを作成する際にこれらの名前を指定する方法の説明については、320 ページの『CL コマンドの定義』の PARM (パラメーター) ステートメントおよび ELEM (要素) ステートメントを参照してください。

以下の表で、*NAME、*SNAME、および *CNAME の名前で使用できる文字、およびそれらの名前を指定するために使用する規則について説明します。

表 *18. *NAME、*SNAME、および *CNAME* に使用可能な文字

名前のタイプ	最初の文字	その他の文字	最小の長さ	最大の長さ
*NAME [1]	A から Z、 $, #, @	A から Z、0 から 9、 $、 #、@、_、.	1	256

表 18. *NAME、*SNAME、および *CNAME に使用可能な文字 (続き)

名前のタイプ	最初の文字	その他の文字	最小の長さ	最大の長さ
*SNAME [1]	A から Z、 $, #, @	A から Z、 0 から 9、 $, #, @, _	1	256
*CNAME [1]	A から Z、 $, #, @	A から Z、 0 から 9、 $, #, @		
引用符付き名 [2]	"3	ブランク、 *、?、'、"、X'00' から X'3F'、および X'FF' を除く任意の文字	3	256

注:

1. このシステムでは、小文字が大文字に変換されます。
2. 引用符を使用できるのは基本名 (*NAME) のみです。
3. 最初と最後の文字は引用符 (") でなければなりません。

関連概念:

130 ページの『オブジェクトの命名規則』
制御言語 (CL) コマンドで使用されるすべての IBM i オブジェクトの命名には、以下の規則が適用されます。単純オブジェクト名、修飾名、または総称名が使用できるかどうかについては、各 CL コマンドのパラメーターの要約表に示しています。

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

40 ページの『CL コマンド・ラベル』
コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

フォルダー名および文書名:

フォルダー名および文書名には、フォルダーまたは文書の内容が分かるような名前を付けてください。

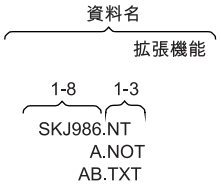
フォルダー名はそれぞれ固有のものでなければなりません。また、ユーザーが内容を判断しやすいということのほかに、できる限り入力しやすい名前にすることも必要です。システムの特定のフォルダーを見つけてその中の文書を変更するためには、そのフォルダー名を指定するか、あるいは、名前のリストから該当のフォルダー名を選択しなければなりません。

文書名は、フォルダーの中で固有のものでなければなりません。また、内容を判断しやすいということのほかに、できる限り入力しやすい名前にすることも必要です。命名の際には、後で文書を見つけやすくするための十分な配慮が必要です。

フォルダーまたは文書に使用する名前は、以下の規則に従ったものでなければなりません。

- 名前はフォルダーの中で固有のものでなければなりません。

- 文書名またはフォルダー名は、オプションの拡張子も含めて、長さが 1 から 12 文字でなければなりません。拡張子を使用しない場合の文書名またはフォルダー名は、長さが 8 文字以下でなければなりません。拡張子 (使用する場合) は、ピリオドで始まり、その後に最高 3 文字を付けることができます。文書名に拡張子を使用し、拡張子で文書を識別できるようにしておけば、システム内にある文書の選択リストを作成する際に役に立ちます。



- 文書名またはフォルダー名には任意の 1 バイト EBCDIC 文字を入れることができますが、次の特殊文字は使用できません。これらの文字はシステムが別の用途に使用します。

文字	特別な用途
アスタリスク (*)	乗算演算子、総称名、または IBM i の予約値を表します。
斜線 (/)	除算演算子、システム値の区切り文字、または修飾オブジェクト名の各部分を区切る文字として使用されます。
疑問符 (?)	システム・ヘルプに対する要求を開始するための文字です。

- あるフォルダーが別のフォルダーに保管されている場合には、両方のフォルダー名を斜線 (/) で区切って使用します。このように組み合わせた名前をフォルダー・パスと呼びます。例えば、FOLDR2 を FOLDR1 に保管させた場合、FOLDR2 のパスは FOLDR1/FOLDR2 となります。FOLDR1 が第 **1** レベル・フォルダーで、FOLDR2 が次レベル・フォルダーです。1 つのフォルダーの名前の長さは、オプションの拡張子も含めて 1 から 12 文字です。1 つのフォルダー・パスには、最高 63 文字を入れることができます。

  システム提供のフォルダー名は Q で始まっているため、ユーザーのフォルダー名の最初の文字には Q を使用することはできません。以下に、有効なフォルダー名およびフォルダー・パスの例を示します。

```
@LETTERS
FOLDER.PAY
PAYROLL/FOLDER.PAY
#TAX1/FOLD8.TAX/$1988/PAYROLL/FOLDER.PAY
```

注**:**

1. CL コマンドでは、システムが修飾名 (ライブラリー名/オブジェクト名) と混同することを避けるために、フォルダー・パス名は単一引用符で囲まなければなりません。名前の一部として単一引用符を含める場合には、2 つの連続した単一引用符を使用してください。
2. 多くの CL コマンドが、文書とフォルダーのいずれか一方に対して働きますが、両方に対して働くものもいくつかあります。文書またはフォルダーのいずれかを参照するには、省略語の DLO (文書ライブラリー・オブジェクト) を用います。
3. CL コマンドでは、CL の区切り文字を含むフォルダー名または文書名は、単一引用符で囲まなければなりません。
4. システムはグラフィック文字を認識することはできません。認識するのはコード・ポイントのみであり、さらに以下のことが前提となっています。

- フォルダー名および文書名はすべて、1 バイト EBCDIC コード・ページを使用してコード化されていること。これらのコード・ページの中で、グラフィック文字を表すのは、16 進数 41 から FE までのコード・ポイントのみであり、したがって、フォルダー名および文書名の中で使用できるのはこれらのコード・ポイントのみです。
- コード・ポイント 16 進数 5C はアスタリスク (*) を、61 はスラッシュ (/) を、6F は疑問符 (?) を表します。これらをフォルダーおよび文書の名前に使用することはできません。
- 英字の小文字に該当するコード・ポイントは、大文字のコード・ポイントに変換されます。16 進数 81 から 89 は C1 から C9 に、91 から 99 は D1 から D9 に、A2 から A9 は E2 から E9 に変換されます。

システム内部では、フォルダーおよび文書は、上述したフォルダー名および文書名に加え、システム・オブジェクト名によって類別されます。システム・オブジェクト名は、日付/タイム・スタンプから抽出された 10 文字の名前で、これは通常ユーザーには知らされませんが、一部の CL コマンドでは、フォルダー名または文書名として *SYSOBJNAM を指定し、別のパラメーターでシステム・オブジェクト名を指定することによって、この名前を指定できます。

関連概念:

130 ページの『オブジェクトの命名規則』
制御言語 (CL) コマンドで使用されるすべての IBM i オブジェクトの命名には、以下の規則が適用されます。単純オブジェクト名、修飾名、または総称名が使用できるかどうかについては、各 CL コマンドのパラメーターの要約表に示しています。

関連情報:


ローカル装置構成 PDF

装置記述の作成 (表示装置)(CRTDEVDSP) コマンド

IBM i オブジェクト

オブジェクト は、記憶域に存在する (スペースを占める) 名前付きの単位であり、オペレーティング・システムはこの単位に対して操作を実行します。IBM i オペレーティング・システムがすべてのデータ処理情報を保管して処理するための手段は、IBM i のオブジェクトによって提供されます。

オブジェクトは、コマンドによる操作の対象になる基本的な単位です。例えば、プログラムやファイルはオブジェクトです。ユーザーは、オブジェクトを介して、システムのデータの検索、保守、および処理を行うことができます。ユーザーは、使用するオブジェクトおよび機能 (コマンド) さえ知っていればよく、データの記憶域のアドレスを知っている必要はありません。

CL コマンドは、IBM i オブジェクト上で操作を実行します。いくつかのタイプの IBM i オブジェクトは、制御言語で作成され、使用されます。IBM i オブジェクトには、次の共通点があります。

- オブジェクトには、そのオブジェクトを記述する一連の属性が備わっていますが、これらの属性はオブジェクトの作成時にユーザーが定義します。
- システムが特定の機能の実行に使用しなければならないオブジェクトは、その機能を実行する CL コマンドで指定する必要があります。
- オブジェクトには、そのオブジェクトを記述する一連の属性があり、それらの属性にそれぞれ特定の値が割り当てられます。
- 通常、オブジェクトは、それぞれ他のオブジェクトから独立して存在します。ただし、オブジェクトによっては、他のオブジェクトに先立って作成しなければならないものもあります。例えば、基礎となる物理ファイルが存在していないと、論理ファイルを作成することはできません。

- オブジェクトは、そのオブジェクトを使用する操作の実行前に作成しておく必要があります。各作成コマンドで作成するオブジェクト・タイプについては、それぞれの作成 (CRT) コマンドの項で詳しく説明します。
- 制御言語で使用されるすべての IBM i オブジェクトに名前があります。CL コマンドで指定するオブジェクト名によって、コマンドの機能を実行するためにオペレーティング・システムがどのオブジェクトを使用するかが指示されます。
- オブジェクトは、単純名、修飾名、または総称名のいずれかを持ちます。

システムでは、各種のオブジェクトのタイプをサポートしています。タイプの中には、データ処理システムの多くに共通しているオブジェクトを指すものがあります。以下のタイプはその例です。

- ファイル
- プログラム
- コマンド
- ライブラリー
- 待ち行列
- モジュール
- サービス・プログラム

また次のように、必ずしも他のシステムでは使用されないタイプもあります。

- ユーザー・プロファイル
- ジョブ記述
- サブシステム記述
- 装置記述

オブジェクトのタイプが異なれば、その操作特性 (属性) も異なります。このような特性の相違が各オブジェクト・タイプをそれぞれ固有のものにしています。例えば、ファイルはデータが入っているオブジェクトであり、命令が入っているプログラムとは操作特性の点で異なっています。

オブジェクトにはそれぞれ名前があります。オブジェクトを識別するのに使用されるのは、オブジェクト名とオブジェクト・タイプです。オブジェクト名は、オブジェクトを作成するユーザーによって割り当てられます。オブジェクト・タイプは、そのオブジェクトの作成に使用されるコマンドによって決まります。例えば、あるプログラムを作成し、それに OEUPDT (受注項目更新 の略) という名前を付けた場合、そのプログラムは常にその名前で参照されることになります。システムは、オブジェクト名 (OEUPDT) およびオブジェクト・タイプ (プログラム) によって該当のオブジェクトを見つけ出し、そのオブジェクトに対する操作を行います。複数のオブジェクトに同じ名前を付けることもできますが、その場合、それらのオブジェクトはそれぞれオブジェクト・タイプが異なっているか、またはそれぞれ異なるライブラリーに保管されている必要があります。

システムは、オブジェクト・タイプに応じて特定の機能の誤った使用を防止することにより、保全性を維持します。例えば、CALL コマンドはプログラム・オブジェクトを実行します。 CALL コマンドにファイル名を指定した場合、そのファイルと同じ名前のプログラムが存在しなければ、その CALL コマンドは実行されません。

関連概念:

463 ページの『オブジェクトおよびライブラリー』
オブジェクトおよびライブラリーに固有のタスクおよび概念には、オブジェクトに対して実行される機能、ライブラリーの作成、およびオブジェクト権限の指定が含まれます。

ライブラリー・オブジェクト

ライブラリーとは、関連するオブジェクトをグループ化し、それらのオブジェクトの使用時に名前で見つけられるようにするために使用されるオブジェクトです。したがって、ライブラリーはオブジェクトのグループの登録簿のようなものです。

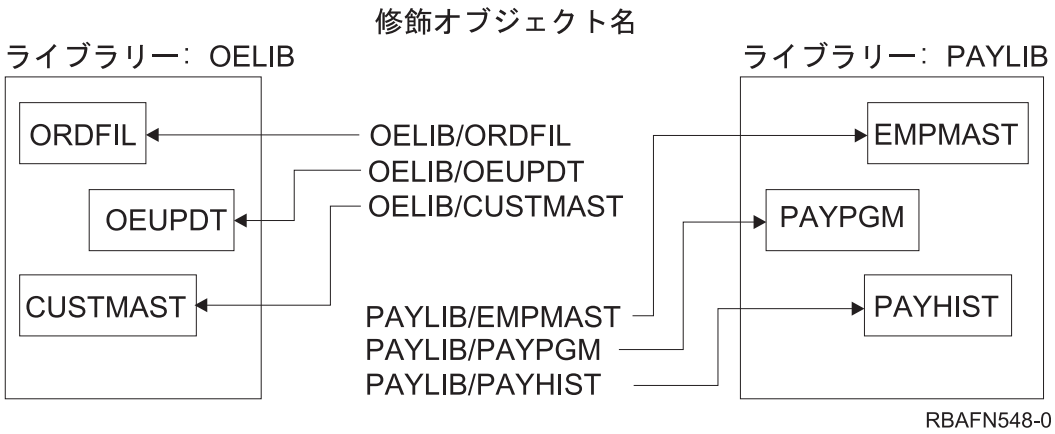
ライブラリーを使用することによって、オブジェクトを何らかの意味を持つグループにグループ化することができます。例えば、セキュリティー上の要件、バックアップ上の要件、または処理上の要件などによって類別し、それぞれのオブジェクトのグループを作ることができます。 1 つのライブラリーに入れることのできるオブジェクトの数、およびシステム上のライブラリーの数については、使用可能なディスク記憶域の量により制約されます。

ライブラリーによるオブジェクトのグループ化は論理的なグループ化です。ライブラリーを作成する時点で、そのライブラリーをどのユーザー補助記憶域プール (ASP) または独立補助記憶域プール (独立ディスク・プール) に作成するかを指定することができます。そのライブラリーに作成されるすべてのオブジェクトは、そのライブラリーと同じ ASP に作成されます。ライブラリー内のオブジェクトは必ずしも互いに物理的に隣接しているわけではありません。ライブラリーのサイズや、他のオブジェクトのサイズは、記憶域内の隣接する使用可能なスペースの大きさによって制限されることはありません。システムは、オブジェクトがシステムに保管されるときに、そのために必要なスペースを見つけ出します。

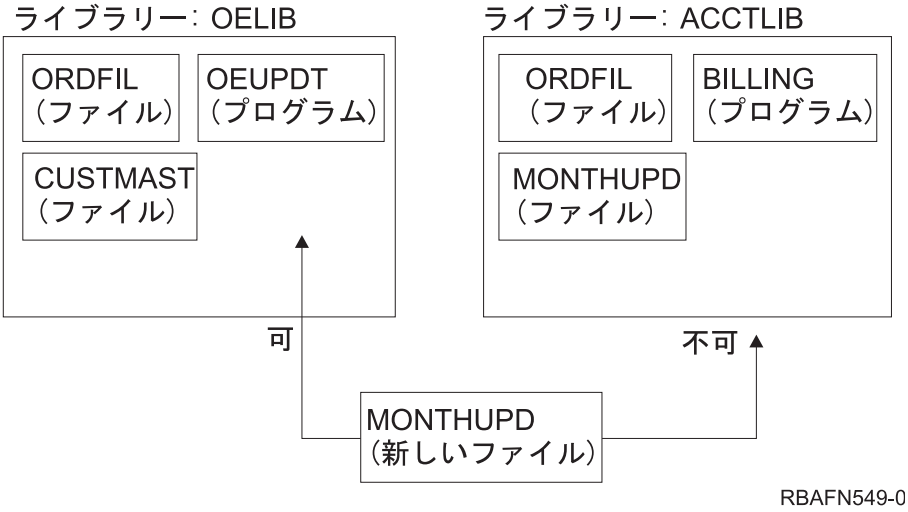
ほとんどのタイプのオブジェクトは、その作成時点でライブラリーに入れられます。**CRTLIB** の AUT パラメーターは、ライブラリーの共通権限を定義します。CRTAUT パラメーターは、ライブラリーに作成されるオブジェクトのデフォルト権限を指定します。オブジェクトを作成するコマンドが AUT パラメーターに ***LIBCRTAUT** を指定している場合、オブジェクトの共通認可はライブラリーに指定されていた作成権限になります。ほとんどのタイプのオブジェクトはライブラリーから別のライブラリーに移すことができますが、この場合、1 つのオブジェクトを同時に複数のライブラリーの中に存在させることはできません。オブジェクトを別のライブラリーに移しても、記憶域内でのオブジェクトの位置が変わるわけではありません。ライブラリー (つまり、どのライブラリーからオブジェクトを見つけ出すか) が変わるだけです。さらに、ほとんどのタイプのオブジェクトは、名前を変更することができ、またライブラリーから別のライブラリーにコピーすることもできます。

ライブラリー名は、オブジェクト名とは異なるレベルでオブジェクトを識別するための名前として使用することができます。既に述べたように、オブジェクトはその名前とタイプとによって識別されます。このオブジェクト名をライブラリー名で修飾することにより、オブジェクトをさらに限定して識別することができるようになります。オブジェクト名とライブラリー名とを組み合わせたものを、そのオブジェクトの修飾名と呼びます。修飾名は、オブジェクトの名前とそのオブジェクトが入っているライブラリーの名前をシステムに知らせます。

次の図は、2 つのライブラリーと、その中の各オブジェクトの修飾名を示しています。



異なるライブラリーには、同じ名前で同じタイプのオブジェクトを 2 つ存在させることができます。ただし、オブジェクト・タイプが異なる場合を除いて、同じライブラリーの中に同じ名前のオブジェクトが 2 つあってはなりません。この設計によって、オブジェクトを名前で参照するプログラムの場合に、そのプログラム自体を変更しなくても、いくつかの異なるオブジェクト (名前は同じで保管されているライブラリーが異なるもの) を、同一のプログラムの実行によって処理することができます。また、新しいオブジェクトを作成するユーザーは、他のライブラリーの中にどのような名前のオブジェクトがあるかについて注意を払う必要はありません。例えば、次の図で、MONTHUPD (月次更新) という名前の新しいファイルは、ライブラリー OELIB には追加することができますが、ライブラリー ACCTLIB に追加することはできません。ライブラリー ACCTLIB の中には、タイプがファイルである MONTHUPD という名前のオブジェクトが既に存在しているので、同じ名前のファイルを作成しようとするとエラーになります。



1 つのライブラリーの中では、オブジェクトはオブジェクト名およびタイプによって識別されます。CL コマンドは 1 つのオブジェクト・タイプに対してだけ使用されるものが多いので、そのようなコマンドではオブジェクト・タイプは必ずしも明示的に指定する必要はありません。ただし、複数のオブジェクト・タイプに使用されるコマンドの場合には、オブジェクト・タイプを明示的に指定しなければなりません。

統合ファイル・システムを使用するオブジェクトはディレクトリー内にあって、ライブラリーを検索する代わりにパス名パターンまたはオブジェクト名パターンを使用することによって検索することができます。また、オブジェクトを見つけるためにこれらのディレクトリーを使用することもできます。

関連概念:

統合ファイル・システム

関連タスク:

476 ページの『ライブラリーの使用』
ライブラリーは、関連するオブジェクトをグループ化し、名前によってオブジェクトを探索するために使用するオブジェクトです。したがって、ライブラリーはオブジェクトのグループの登録簿のようなものです。

関連情報:

独立ディスク・プールの例

外部オブジェクト・タイプ

外部オブジェクトの多くのタイプは、ライブラリーに保管されます。一部のタイプの外部オブジェクトは、ディレクトリー内の統合ファイル・システムのみに保管できます。

一部のタイプのオブジェクトは、特定のライブラリーのみに保管できます。このようなオブジェクト・タイプについて、オブジェクトが作成されるライブラリーを、表 19 の「デフォルトのユーザー・ライブラリー」欄に示します。

その他のタイプのオブジェクトでは、オブジェクトの作成時にライブラリー名を指定することができます。これらのタイプのオブジェクトを作成する CL コマンドのライブラリー名修飾子のデフォルト値は、特殊値である *CURLIB です。 *CURLIB 値では、現行ライブラリーの名前が使用されます。ジョブまたはスレッドの現行ライブラリーは、**現行ライブラリー変更 (CHGCURLIB)** コマンドを実行することで、ユーザー・プロファイル属性またはジョブ属性として設定できます。オブジェクトの作成時に有効な現行ライブラリーがない場合は、ライブラリー QGPL が使用されます。

「デフォルトのユーザー・ライブラリー」欄に「N/A」(適用外) と記されているその他のタイプのオブジェクトは、ライブラリーまたはディレクトリーに保管することはできません。

システム・オブジェクト・ダンプ (DMPSYSOBJ) コマンドの例外を使用して、「16 進形式」欄で示されている形式でオブジェクト・タイプを指定することはできません。

表 *19.* 外部 *IBM i* オブジェクト・タイプ別の事前定義値およびデフォルトのライブラリー・ロケーションまたはディレクトリー・ロケーション

値	オブジェクト・タイプ	**16** 進形式	デフォルトのユーザー・ライブラリーまたはディレクトリー
*ALRTBL	警報テーブル	0E09	*CURLIB
*AUTL	権限リスト	1B01	QSYS
*BLKSF	ブロック特殊ファイル	1E05	現行ディレクトリー
*BNDDIR	バインド・ディレクトリー	1937	*CURLIB
*CFGL	構成リスト	1918	QSYS
*CHRSF	文字特殊ファイル	1E06	現行ディレクトリー
*CHTFMT	図表形式	190D	*CURLIB
*CLD	C/400 ロケール記述	190B	*CURLIB
*CLS	クラス	1904	*CURLIB
*CMD	コマンド	1905	*CURLIB
*CNNL	接続リスト	1701	QSYS
*COSD	サービス・クラス記述	1401	QSYS
*CRG	クラスター資源グループ	192C	QUSRSYS
*CRQD	CRQ 記述	0E0F	*CURLIB

表 19. 外部 *IBM i* オブジェクト・タイプ別の事前定義値およびデフォルトのライブラリー・ロケーションまたはディレクトリー・ロケーション (続き)

値	オブジェクト・タイプ	**16** 進形式	デフォルトのユーザー・ライブラリーまたはディレクトリー
*CSI	通信サイド情報	1935	*CURLIB
*CSPMAP	システム間プロダクト・マップ	1922	*CURLIB
*CSPTBL	システム間プロダクト・テーブル	1923	*CURLIB
*CTLD	制御装置記述	1201	QSYS
*DDIR	分散ファイル・ディレクトリー	1F02	N/A
*DEVD	装置記述	1001	QSYS
*DIR	ディレクトリー	0C01	現行ディレクトリー
*DOC	文書	190E	QDOC
*DSTMF	分散ストリーム・ファイル	1F01	N/A
*DTAARA	データ域	190A	*CURLIB
*DTADCT	データ・ディクショナリー	1920	データ・ディクショナリーと同じ名前のライブラリー
*DTAQ	データ待ち行列	0A01	*CURLIB
*EDTD	編集記述	1908	QSYS
*EXITRG	出口登録情報	1913	QUSRSYS
*FCT	用紙制御テーブル	0E04	*CURLIB
*FIFO	先入れ先出し法特殊ファイル	1E07	現行ディレクトリー
*FILE	ファイル	1901	*CURLIB
*FLR	フォルダー	1912	QDOC
*FNTRSC	フォント資源	1926	*CURLIB
*FNTTBL	フォント・マッピング・テーブル	192B	*CURLIB
*FORMDF	書式定義	1928	*CURLIB
*FTR	フィルター	0E0B	*CURLIB
*GSS	グラフィック記号セット	190C	*CURLIB
*IGCDCT	2 バイト文字セット (DBCS) 変換辞書	0E06	*CURLIB
*IGCSRT	2 バイト文字セット (DBCS) 分類テーブル	191A	*CURLIB
*IGCTBL	2 バイト文字セット (DBCS) フォント・テーブル	1910	QSYS
*IMGCLG	イメージ・カタログ	192E	QUSRSYS
*IPXD	インターネットワーク・パケット交換機能記述	191E	QSYS
*JOBD	ジョブ記述	1903	*CURLIB
*JOBQ	ジョブ待ち行列	0E01	*CURLIB
*JOBSCD	ジョブ・スケジュール	0E0C	*CURLIB
*JRN	ジャーナル	0901	*CURLIB

表 19. 外部 *IBM i* オブジェクト・タイプ別の事前定義値およびデフォルトのライブラリー・ロケーションまたはディレクトリー・ロケーション (続き)

値	オブジェクト・タイプ	**16** 進形式	デフォルトのユーザー・ライブラリーまたはディレクトリー
*JRNRCV	ジャーナル・レシーバー	0701	*CURLIB
*LIB	ライブラリー	0401	QSYS
*LIND	回線記述	1101	QSYS
*LOCALE	ロケール	1921	*CURLIB
*MEDDFN	メディア定義	191C	*CURLIB
*MENU	メニュー記述	1916	*CURLIB
*MGTCOL	管理収集	192D	QYPSCSCA API を使用して指定されているライブラリーの場合は NA または QPFRDATA
*MODD	モード記述	1501	QSYS
*MODULE	コンパイラー単位	0301	*CURLIB
*MSGF	メッセージ・ファイル	0E03	*CURLIB
*MSGQ	メッセージ待ち行列	1902	*CURLIB
*M36	AS/400 アドバンスト 36 マシン	1E04	*CURLIB
*M36CFG	AS/400 アドバンスト 36 マシン構成	1924	*CURLIB
*NODGRP	ノード・グループ	192A	*CURLIB
*NODL	ノード・リスト	0E0E	*CURLIB
*NTBD	NetBIOS 記述	1914	QSYS
*NWID	ネットワーク・インターフェース記述	1601	QSYS
*NWSCFG	ネットワーク・サーバー構成	1939	QUSRSYS
*NWSD	ネットワーク・サーバー記述	1D01	QSYS
*OUTQ	出力待ち行列	0E02	*CURLIB
*OVL	オーバーレイ	1929	*CURLIB
*PAGDFN	ページ定義	1936	*CURLIB
*PAGSEG	ページ・セグメント	1927	*CURLIB
*PDFMAP	PDF マップ	0E11	*CURLIB
*PDG	印刷記述子グループ	1930	*CURLIB
*PGM	プログラム	0201	*CURLIB
*PNLGRP	パネル・グループ定義	1915	*CURLIB
*PRDAVL	プロダクト可用性	1933	QSYS
*PRDDFN	プロダクト定義	191B	QSYS
*PRDLOD	プロダクト・ロード	191D	QSYS
*PSFCFG	Print Services Facility™構成	1925	*CURLIB
*QMFORM	QUERY 管理機能書式	1932	*CURLIB
*QMQRY	QUERY 管理機能プログラム	1931	*CURLIB
*QRYDFN	QUERY 定義	1911	QGPL

表 19. 外部 *IBM i* オブジェクト・タイプ別の事前定義値およびデフォルトのライブラリー・ロケーションまたはディレクトリー・ロケーション (続き)

値	オブジェクト・タイプ	**16** 進形式	デフォルトのユーザー・ライブラリーまたはディレクトリー
*RCT	参照コード変換テーブル	0E08	QGPL
*SBSD	サブシステム記述	1909	*CURLIB
*SCHIDX	検索索引	0E07	QGPL
*SOCKET	ローカル・ソケット	1E03	N/A
*SPADCT	スペル援助ディクショナリー	1C01	QGPL
*SQLPKG	SQL パッケージ	0202	*CURLIB
*SQLUDT	ユーザー定義の SQL 型	191F	*CURLIB
*SQLXSR	SQL XML スキーマ・リポジトリー	1949	*CURLIB
*SRVPGM	サービス・プログラム	0203	*CURLIB
*SSND	セッション記述	0E05	QGPL
*STMF	バイトストリーム・ファイル	1E01	現行ディレクトリー
*SVRSTG	サーバー記憶域	1917	QUSRSYS
*SYMLNK	記号リンク	1E02	現行ディレクトリー
*S36	システム/36 マシン記述	1919	QGPL
*TBL	表	1906	*CURLIB
*TIMZON	タイム・ゾーン記述	192F	QSYS
*USRIDX	ユーザー索引	0E0A	*CURLIB
*USRPRF	ユーザー・プロファイル	0801	QSYS
*USRQ	ユーザー待ち行列	0A02	*CURLIB
*USRSPC	ユーザー・スペース	1934	*CURLIB
*VLDL	妥当性検査リスト	0E10	*CURLIB
*WSCST	ワークステーション・ユーザー・カスタマイズ・オブジェクト	1938	*CURLIB

関連概念:

463 ページの『オブジェクト・タイプおよび共通属性』
システム上の各オブジェクト・タイプは、システム内に固有の目的を持ち、そのオブジェクトを記述する共通の属性セットを持っています。

関連資料:

89 ページの『OBJTYPE パラメーター』
オブジェクト・タイプ (OBJTYPE) パラメーターは、オブジェクトが指定されたコマンドにより操作できる IBM i オブジェクトのタイプを指定します。

関連情報:

Change System Collector Attributes (QYPSCSCA, QypsChgSysCollectorAttributes) API

単純オブジェクト名および修飾オブジェクト名

ライブラリー内の特定のオブジェクトの名前は、単純名としても修飾名としても指定できます。

単純オブジェクト名 は、オブジェクトの名前のみで構成される名前です。 修飾オブジェクト名 は、オブジェクトが入っているライブラリーの名前の後に、そのオブジェクトの名前を付けたものです。 修飾オブジェクト名では、ライブラリー名とオブジェクト名とは斜線 (/) で連結します。

対象のオブジェクトが、ジョブのライブラリー・リストに含まれているライブラリーのいずれかに存在する場合には、そのオブジェクトの単純名および修飾名のいずれでも指定できます。すなわち、この場合、ライブラリー修飾子の指定はオプションです。指定するオブジェクトがライブラリー・リスト上のライブラリーのいずれにも含まれていない場合は、修飾名を指定する必要があります。

注**:** ジョブ名も修飾形式を取ることができますが、ジョブは IBM i オブジェクトではないため、修飾オブジェクト名としては扱われません。ジョブ名は、ライブラリー名ではなくユーザー名およびジョブ番号で修飾します。

単純オブジェクト名および修飾オブジェクト名の形式を、次の表に示します。

表 *20.* 単純オブジェクト名および修飾オブジェクト名

名前のタイプ	名前の構文	例
単純オブジェクト名	オブジェクト名	OBJA
修飾オブジェクト名	ライブラリー名/オブジェクト名	LIB1/OBJB

関連概念:

41 ページの『CL コマンドの区切り文字』
コマンド区切り文字は、コマンドの中で、文字のグループの始まりと終わりを示す特殊文字またはスペースです。

45 ページの『CL コマンド定義』
コマンド定義機能を使用することにより、ユーザーはアプリケーションの特殊な要件に対応する新たな制御言語コマンドを作成することができます。ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。

関連資料:

5 ページの『CL コマンド名』
コマンド名は、コマンドの実行時に呼び出されるプログラムが実行する機能を識別します。コマンド名は、動詞 (動作) および動作の対象 (対象となるオブジェクト) を識別する名詞または句を組み合わせたものです (コマンド = 動詞 + 対象となるオブジェクト)。

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

81 ページの『JOB パラメーター』
JOB パラメーターは、コマンドを適用する対象となるジョブの名前を指定します。

総称オブジェクト名

総称オブジェクト名は、同様の名前を持つ複数のオブジェクトを参照する場合に使用できます。

総称オブジェクト名 を用いて、複数のオブジェクトを示すことができます。 すなわち、総称名は、いくつかのオブジェクト名に共通する最初の 1 つ以上の文字で構成されます。システムは、ライブラリー・リスト内のライブラリーにあるオブジェクトのうち、指定された文字で始まる名前を持つオブジェクトをすべて見つけだします。名前の最後の文字がアスタリスク (*) であれば、それは総称名と見なされます。

引用符付き総称名は、引用符で囲んだ総称名です。通常の引用符付きの名前の場合と異なり、総称名の場合は、引用符と引用符との間に特殊文字がなくても、引用符が除去されません。例えば、総称名が "ABC*" であれば、引用符で始まってその後に ABC という 3 文字が続く名前のオブジェクトをすべてシステムが検索します。

総称名は、ライブラリー名によって修飾することもできます。総称名を修飾した場合、システムは、指定されたライブラリーのみを検索して、指定された総称名で始まる名前を持つオブジェクトを見付け出します。

注: 総称名がパス名である場合には、1 つ以上のディレクトリーによって修飾することもできます。パス名では、アスタリスク (*) の前後に文字を指定することができます。

総称名を指定すると、システムは、名前が指定の文字列で始まるすべてのオブジェクトに対して、所要の機能を実行します。ユーザーは、総称名によって識別される各オブジェクトに対して機能を実行するために必要な権限を持っている必要があります。ユーザーがオブジェクトに対して必要な権限を持っていないと、この機能は実行されず、その総称機能が失敗した個々のインスタンスについて診断メッセージが出されます。総称機能が正常に実行されたオブジェクトについては、それぞれ完了メッセージが出されます。完了メッセージを見るには、オンライン低レベル・メッセージを表示する必要があります。総称機能の実行が完了すると、すべてのオブジェクトに対して操作が正常に実行されたことを示す完了メッセージが出されます。1 つ以上のオブジェクトが正常に操作できなかった場合は、エスケープ・メッセージが出されます。ある装置ファイルについて指定変更の指定が有効な場合には、総称名ではなく、その指定変更で指定されている単一のオブジェクト名が、操作の実行に使用されます。

処理対象のオブジェクトが入っているライブラリーが既にロックされている場合は、削除、移動、または名前変更のためのコマンドで総称名を使用してはなりません。総称オブジェクト名を見つけるための検索では、オブジェクトを含むライブラリーに対して、完全オブジェクト名を見つけるための検索よりも拘束度の高いロックが必要になります。これは、削除、移動、または名前変更のためのコマンドの実行中に、他のユーザーが同じライブラリーの中に総称ストリング検索に該当する名前のオブジェクトを作成するのを防ぐためです。この問題は、総称名の代わりに完全オブジェクト名を使用することによって回避できます。あるいは、ライブラリーをロックしているジョブまたはサブシステムを終了させる方法もあります。

注: どのジョブまたはサブシステムがライブラリーをロックしているのかを判別するには、**オブジェクト・ロック処理 (WRKOBJLCK)** コマンドを使用してください。

コマンドによっては、総称名とともにライブラリー修飾子を指定して、操作の範囲を限定できる場合があります。例えば、**印刷装置ファイル変更 (CHGPRTF)** コマンドで FILE(LIB1/PRT*) を指定すると、ライブラリー LIB1 に存在し、かつ名前が PRT で始まっている印刷装置ファイルのみが操作の対象となります。その他のライブラリーにある印刷装置ファイルは操作の対象外となります。

ライブラリー修飾子によって、操作範囲は次のように限定されます。

- ライブラリー名: 指定したライブラリーに含まれ、かつ総称名に該当するオブジェクトのみが、操作の対象になります。
- *LIBL: 総称操作を要求したジョブに関連付けられたライブラリー・リストに含まれ、かつ総称名に該当するオブジェクトが、操作の対象になります。
- *CURLIB: 総称名に該当するオブジェクトであり、かつ現行ライブラリーに含まれているものが操作の対象となります。
- *ALL: ユーザーが権限を持つ、すべてのライブラリーの総称名に該当するオブジェクトが、操作の対象になります。
- *USRLIBL: ジョブのライブラリー・リストのユーザー部分に存在し、かつ総称名に該当するオブジェクトのみが、操作の対象になります。

- *ALLUSR: 一部の例外を除くシステム・ライブラリー以外のすべてのライブラリー (名前の頭文字が Q でないもの) が、操作の対象になります。

  注**:** ユーザーは、IBM がサポートする各リリースに対して、QUSRVxRxMx という形式の異なるライブラリー名を作成することができます。ここで、VxRxMx は、ライブラリーのバージョン、リリース、およびモディフィケーション・レベルを示します。

表 *21.* 総称オブジェクト名

名前のタイプ	名前の構文	例
単純総称名	総称名*	OBJ*
修飾総称名	ライブラリー名/総称名*	LIB1/OBJ*
引用符付き総称名	"generic-name*"	"ABC*"

関連概念:

統合ファイル・システム

総称ライブラリー名

『オブジェクトの命名規則』

制御言語 (CL) コマンドで使用されるすべての IBM i オブジェクトの命名には、以下の規則が適用されます。単純オブジェクト名、修飾名、または総称名が使用できるかどうかについては、各 CL コマンドのパラメーターの要約表に示しています。

133 ページの『総称名 (*GENERIC)』

総称名は、複数のオブジェクトに共通する少なくとも 1 つの先頭文字の後に、アスタリスクを付けたものです。

オブジェクトの命名規則

制御言語 (CL) コマンドで使用されるすべての IBM i オブジェクトの命名には、以下の規則が適用されます。単純オブジェクト名、修飾名、または総称名が使用できるかどうかについては、各 CL コマンドのパラメーターの要約表に示しています。

単一オブジェクトの名前の指定

単一オブジェクトの名前には、名前の各部分 (単純名およびライブラリー修飾名) にそれぞれ最高 10 文字を使用することができます。

ユーザー作成のオブジェクトの名前の指定

ユーザー作成のオブジェクトの名前の指定: IBM 提供のオブジェクト (コマンドを除く) には、すべて Q で始まる名前が付いていることから、ユーザー作成のオブジェクトと IBM 提供のオブジェクトを区別するために、ユーザー作成オブジェクトには Q で始まる名前を付けないようにしてください。 CL オブジェクト名には最大 10 文字を使用できますが、一緒に使用している他の高水準言語の命名規則との整合性を保つために、文字数を少なくしなければならないこともあります。また、高水準言語によっては、名前に下線の使用が認められていないこともあります。例えば、RPG では、名前が 8 文字以下に限定されており、下線も使用できません。

総称オブジェクトの名前の指定

総称名は、末尾のアスタリスク (*) を除いて、最高 9 文字の英数字を使用できます。

例えば、総称名が使用できる場合には、INV と INV* はどちらも有効な値です。INV を指定した場合には、INV という名前のオブジェクトのみが参照されます。総称名 INV* を指定した場合には、INV、INVOICE、INVENTORY、および INVENPGM1 など、INV で始まるすべてのオブジェクトが参照されます。引用符付きの総称名 "INV*" を指定した場合には、"INV%1" や "INV>" など、"INV" で始まるオブジェクトが参照されます。

オブジェクトのライブラリー修飾子に関する制約

作成するオブジェクトが、ライブラリー、ユーザー・プロファイル、回線記述、制御装置記述、装置記述、モード記述、サービス・クラス記述、または構成リストの場合は、オブジェクト名はライブラリー名で修飾することはできません。 ライブラリーをライブラリー内に入れることはできないため、ライブラリー名を修飾することはできません。その他のオブジェクト・タイプ (*USRPRF、*LIND、*CTLD、*DEVD、*MODD、*COSD、および *CFGL) は、QSYS ライブラリーのみに存在するタイプとして扱われます。これらのオブジェクト・タイプのオブジェクトの名前のみが受け入れられる場合には、オブジェクト名をライブラリー名で修飾することはできません。**オブジェクト記述表示 (DSPOBJD)** コマンドでは、どのようなオブジェクト名も受け入れられるため、QSYS を指定できます。

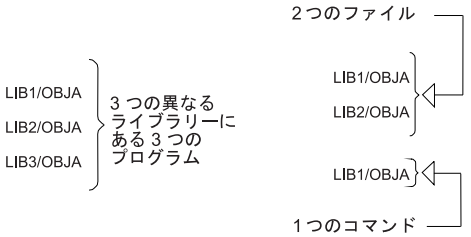
ライブラリー・リスト修飾子

ほとんどのコマンドでは、ライブラリーの代わりに事前定義値 *LIBL (および、*CURLIB などの値) を使用できます。*LIBL は、修飾名の 2 番目の部分で指定されているオブジェクトを見つけるために、ジョブのライブラリー・リスト上にあるライブラリーを使用するよう指定します。

重複オブジェクト名

同じライブラリーに、同じタイプで同じ名前を持つオブジェクトが 2 つ以上あってはなりません。

オブジェクト・タイプが異なる場合を除いて、同じ名前を持つ 2 つのオブジェクトを同じライブラリーに保管することはできません。 OBJA という名前を持つオブジェクトが 2 つある場合、その 2 つを LIBx というライブラリーに保管できるのは、例えば、一方がプログラムでもう一方がファイルである場合に限られます。次に示す名前とオブジェクト・タイプの組み合わせは、すべて同時にシステムに存在することができます。



複数のライブラリーに同じ名前のオブジェクトが含まれており (しかも、それらのライブラリーが同じライブラリー・リスト上にあり)、オブジェクト名とともにライブラリー修飾子を指定しなかった場合には、その名前を持つ最初に見つかったオブジェクトが使用されます。したがって、同じ名前のオブジェクトが 2 つ以上ある場合には、オブジェクト名を所定のライブラリー名で修飾するか、あるいは、所定のライブラリーがライブラリー・リストの最初にくるようにしてください。例えば、テストやデバッグを行う時点で修飾名を使用しない場合には、ライブラリー・リストの中で、テスト・ライブラリーを、実働ライブラリーより前に置いてください。

デフォルト・ライブラリー

修飾オブジェクト名において、ライブラリー名は通常オプションです。オプションのライブラリー修飾子を指定しなかった場合には、コマンドの記述の中に示されている、デフォルトのライブラリー (通常は *CURLIB または *LIBL) が使用されます。指定のオブジェクトを作成する場合は、現行ライブラリーがデフォルト値となります。すなわち、作成されたオブジェクトは、現行ライブラリーに入れられます。現行ライブラリーが定義されていない場合は、QGPL (汎用ライブラリー) に入れられます。既存のオブジェクトの場合には、ほとんどのコマンドで、 *LIBL がデフォルトのライブラリーです。したがって、指定されたオブジェクトを見つけるために、ジョブのライブラリー・リストが使用されます。システムは、指定された

オブジェクトが見つかるまで、現行ライブラリー・リストにリストされているすべてのライブラリーを検索します。

関連概念:

117 ページの『コマンド内での命名』
名前の指定に使用できる文字は、制御言語 (CL) で指定する名前のタイプによって決まります。

128 ページの『総称オブジェクト名』
総称オブジェクト名は、同様の名前を持つ複数のオブジェクトを参照する場合に使用できます。

118 ページの『フォルダー名および文書名』
フォルダー名および文書名には、フォルダーまたは文書の内容が分かるような名前を付けてください。

関連資料:

5 ページの『CL コマンド名』
コマンド名は、コマンドの実行時に呼び出されるプログラムが実行する機能を識別します。コマンド名は、動詞 (動作) および動作の対象 (対象となるオブジェクト) を識別する名詞または句を組み合わせたものです (コマンド = 動詞 + 対象となるオブジェクト)。

通信名 **(*CNAME):**

通信名の名前構文には制限があります。通常は、名前の長さと有効な文字セットが 1 つ以上の通信アーキテクチャーによって制限されている、装置構成オブジェクトを参照するために使用します。

通信名は、引用符のない基本名と同じですが、いくつか例外があります。

1. ピリオド (.) および下線 (_) は使用できません。
2. IBM コマンドの場合、*CNAME は 8 文字に制限されています。

通信名の例を次に示します。

```
APPN3@@
```

他の IBM システムでは制約のある文字セットを使用している場合があるため、特殊文字 #、$、および @ を名前に使用する場合には特に注意してください。リモート・システムのキーボードに、これらの特殊文字がない場合があります。リモート・システムとの間で交換する可能性のある名前には次のものがあります。

- ネットワーク識別コード
- ロケーション名
- モード名
- サービスのクラス名
- 制御点名

関連概念:

133 ページの『名前 (*NAME)』
基本名および引用符付きの基本名を作成する際は、以下の規則に従ってください。

関連資料:

40 ページの『CL コマンド・ラベル』
コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

総称名 **(*GENERIC):**

総称名は、複数のオブジェクトに共通する少なくとも 1 つの先頭文字の後に、アスタリスクを付けたものです。

アスタリスクは、その前の共通文字列が総称名であることを示します。アスタリスクがなければ、システムはその文字列を特定のオブジェクトの名前であると解釈します。

関連概念:

128 ページの『総称オブジェクト名』
総称オブジェクト名は、同様の名前を持つ複数のオブジェクトを参照する場合に使用できます。

名前 **(*NAME):**

基本名および引用符付きの基本名を作成する際は、以下の規則に従ってください。

***NAME (**引用符のない基本名**)**

基本名は、A から Z、$、#、または @ のいずれかの文字で始めることができ、その後に、9 文字以下の文字が指定できます。 2 文字目以降には、最初の文字と同じ文字のほか、数字 0 から 9、下線 (_)、およびピリオド (.) も使用できます。小文字はこのシステムによって大文字に変換されます。IBM 提供のコマンドで使用する基本名は 10 文字以下でなければなりません。しかし、ユーザー定義のコマンドでは、タイプ *NAME のパラメーター (PARM ステートメント、または ELEM ステートメントの TYPE パラメーターで指定される) を、最大 256 文字で定義することができます。

基本名の例を次に示します。

```
A987@.442#    ONE_NAME    LIB_0690    $LIBX
```

名前は引用符付きの形式でも引用符のない形式でも入力することができます。引用符付きの形式を使用する場合には、次の規則および考慮事項にも従ってください。

***NAME (**引用符付きの基本名**)**

引用符付きの名前は、引用符 (") で始まり、引用符で終っていなければなりません。2 つの引用符の間には、、*、?、'、"、16 進数 00 から 3F および 16 進数 FF を除く任意の文字を使用することができ、区切り文字には斜線を使用します。引用符付きの名前には、グラフィック文字を使用することもできます。IBM 提供のコマンドで使用される引用符付きの基本名は、両端の引用符を除いて 8 文字以下でなければなりません。ユーザー定義のコマンドでは、引用符付きのタイプ *NAME のパラメーターとして、両端の引用符を除いて最大 254 文字の名前を定義することができます。

注**:** 基本名のみが引用符を使用できます。

引用符付きの名前の例を次に示します。

```
"A"        "AA%abc"        "ABC%%abc"
```

引用符付きの名前を使用する場合には、以下に示す制約に注意してください。

- 名前の中のコード・ポイントは、すべてのキーボードからアドレス指定できるとは限りません。
- 引用符付きの名前の中の文字は、高水準言語では無効となるものもあります。
- システム/38 環境は、シンプル (*SNAME) 名のみをサポートします。他の文字を使用した場合は、このオブジェクトをシステム/38 環境オブジェクトとしてアクセスすることはできません。

- システム/36 環境では、制御言語の指定変更を使用しない限り、8 文字を超える名前にアクセスすることができません。
- ピリオドを含む構造化照会言語 (SQL) 名は、SQL ステートメントの中で引用符で囲んで指定しなければなりません。

引用符で囲んだ名前が、引用符を外した状態で基本名として有効な名前である場合には、引用符が除去されます。例えば、"ABC" は ABC と同じであると見なされます。引用符は除去されているため、名前の長さには含まれません。したがって、"ABCDEFGHIJ" は、引用符を含めれば 10 文字を超えていますが、IBM のコマンドでも有効な名前と見なされます。

関連概念:

132 ページの『通信名 (*CNAME)』
通信名の名前構文には制限があります。通常は、名前の長さと有効な文字セットが 1 つ以上の通信アーキテクチャーによって制限されている、装置構成オブジェクトを参照するために使用します。

136 ページの『単純名 (*SNAME)』
制御言語 (CL) の構文を簡素化するために、CL 変数、ラベル、およびキーワードには、単純名が使用されます。単純名は、引用符の付いていない基本名と同じですが、相違点が 1 つあります。単純名ではピリオド (.) は使用できません。

関連資料:

40 ページの『CL コマンド・ラベル』
コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

パス名 (*PNAME):

パス名は、統合ファイル・システムの中でオブジェクトを位置指定するために使用できる文字ストリングです。

このストリングは、1 つ以上の要素から構成できます。各要素は、斜線 (/) で区切ります。各構成要素は一般的にディレクトリーまたはそれに相当するものですが、最後の構成要素にはディレクトリー、ファイルのようなほかのオブジェクト、または置かれる予定のオブジェクト (複数も可) の総称を使用することができます。

注: また、一部の CL コマンドでは、円記号 (¥) をスラッシュ (/) に自動的に変換することにより、円記号 (¥) を区切り記号として使用できます。ただし、その他のいくつかの CL コマンドは、他の文字と同様に円記号 (¥) を取り扱います。そのため、円記号 (¥) 区切り記号を使用するには、注意が必要です。

/ と ¥ 文字が区切り記号として使用される場合、パス名の各構成要素に / 文字、¥ 文字、およびヌルを使用することはできません。オブジェクトを含むファイル・システムで大文字と小文字が区別されるかどうか、およびオブジェクトが作成されるか検索されるかに応じて、名前が大文字に変換される場合も変換されない場合もあります。このパラメーターに CASE(*MONO) (デフォルト) と指定すると、単一引用符で囲まれていない値が、コマンド分析プログラムによってすべて大文字に変更されます。

パス名の先頭にある区切り文字 (例えば、/) は、最上層のディレクトリーであるルート (/) ディレクトリーからパスが始まることを示します。パス名の先頭に区切り文字がない場合は、コマンドを入力したユーザーの現行ディレクトリーからパスが始まると見なされます。

パス名は、ジョブに対して現在有効な CCSID で表さなければなりません。ジョブの CCSID が 65535 である場合は、パス名はそのジョブのデフォルトの CCSID で表さなければなりません。プログラム内のハードコーディングされたパス名は、CCSID 37 でコード化されます。したがって、パス名は、ジョブ CCSID に変換してからコマンドに渡さなければなりません。CL コマンドでのパス名文字ストリングの最大長は 5000 文字です。

QSYS.LIB ファイル・システム内のオブジェクトを操作する場合、構成要素名の形式は「名前.オブジェクト・タイプ」としなければなりません。以下に例を示します。

```
'/QSYS.LIB/PAY.LIB/TAX.FILE'
```

オブジェクトが独立 ASP QSYS.LIB ファイル・システムにある場合は、名前の形式は以下のようにする必要があります。

```
'/asp-name/QSYS.LIB/PAY.LIB/TAX.FILE'
```

ここで、asp-name は独立 ASP の名前です。

特殊文字を含むパス名をコマンド行で入力する際には、単一引用符 (') 記号で前後を囲まなければなりません。パス名を画面上で入力する場合は、これらの記号はオプションです。しかし、引用符付きストリングまたは特殊文字をパス名に入れる場合は、それらを囲む '' 記号も入れなければなりません。以下に、特殊文字の使用に関する規則を示します。

- パス名の先頭の区切り文字 (例えば /) の前に波形記号 (~) 文字がある場合、パスは、コマンドを入力したユーザーのホーム・ディレクトリーから始まります。
- パス名の先頭に、波形記号 (~) 文字に続いてユーザー名、その次に区切り文字 (例えば /) がある場合、パスは、そのユーザー名によって識別されるユーザーのホーム・ディレクトリーから始まります。
- コマンドによっては、パス名の最後の構成要素にアスタリスク (*) または疑問符 (?) を使用して、名前のパターンを検索できるものもあります。* は、* 文字の位置にさまざまな文字がある名前を検索するようにシステムに要求します。? は、? 文字の位置に単一の文字がある名前を検索するようにシステムに要求します。
- IBM i の特殊値との混同を避けるため、パス名を単一のアスタリスク (*) 文字で始めることはできません。 パス名の先頭でパターン照合を行うには、2 つのアスタリスク (**) を使用します。

  注: これは、アスタリスクの前に他の文字がない相対パス名にのみ適用されます。
- 次の文字のいずれかを構成要素名で使用する場合は、パス名を単一引用符 (') および引用符 (") で囲まなければなりません。
  - アスタリスク (*)

    注: IBM i の特殊値との混同を避けるため、パス名を単一のアスタリスク (*) 文字で始めないでください。
  - 疑問符 (?)
  - 単一引用符 (')
  - 引用符 (")
  - 波形記号 (~)。ただし、パス名の最初の構成要素名の最初の文字として使用する場合 (他の位置で使用する場合は、波形記号は単なる文字として解釈されます)。

  注: 上記の文字を、パス名の最初のコンポーネント名の最初の文字として使用することは、コマンド・ストリングの文字の意味が混乱したり、入力で誤りが起こりやすいため、お勧めできません。
- パス名では、コロン (:) を使用しないでください。コロンは、システム内で特別な意味を持ちます。

- コマンドおよび関連するユーザー画面に対する処理サポートは、コマンド・ストリングまたは画面で使用できる文字として、16 進数の 40 未満のコード・ポイントを認識しません。これらのコード・ポイントを使用する場合は、次の例のように 16 進数表記として入力しなければなりません。

  ```
  crtdir dir(X'02')
  ```

  したがって、16 進数で 40 未満のコード・ポイントは、パス名に使用しないでください。この制約条件は、コマンドおよび関連する表示にのみ適用され、API には適用されません。さらに、16 進数 00 の値をパス名で使用することはできません。

関連タスク:

装置名の指定

統合ファイル・システムへのアクセス

関連資料:

88 ページの『OBJ パラメーター』
オブジェクト (OBJ) パラメーターには、このパラメーターを使用したコマンドにより処理される 1 つ以上のオブジェクトの名前を指定します。

40 ページの『CL コマンド・ラベル』
コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

単純名 **(*SNAME):**

制御言語 (CL) の構文を簡素化するために、CL 変数、ラベル、およびキーワードには、単純名が使用されます。単純名は、引用符の付いていない基本名と同じですが、相違点が 1 つあります。単純名ではピリオド (.) は使用できません。

単純名の例を次に示します。

```
NEWCMD    LIB_2
```

関連概念:

133 ページの『名前 (*NAME)』
基本名および引用符付きの基本名を作成する際は、以下の規則に従ってください。

関連資料:

40 ページの『CL コマンド・ラベル』
コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

固有名に関するその他の規則**:**

オブジェクトの命名については、特殊文字 (付加文字) に関して以下のような追加の規則があります。

- コマンド・ラベルの後にはコロン (:) を付ける必要があります。コロンの後にはブランクを置くことができますが、コロンの前にはブランクがあってはなりません。コマンド・ラベル名は引用符付きの名前であってはなりません。
- CL 変数名の前にはアンパーサンド記号 (&) を付けて、CL プログラムの中で使用される CL 変数であることを示す必要があります。

- CL 用組み込み関数の前にはパーセント記号 (%) を付けて、式の中で使用できる IBM 提供の組み込み関数であることを示す必要があります。組み込み関数名は引用符付きの名前とすることはできません。

これらの特殊文字は名前の一部とは見なされません。これらの文字は、その名前が何を示しているかをシステムに知らせるために、名前に付ける付加文字です。したがって、名前は付加文字を含めれば、最大で 11 文字になることがあります。

IBM i オブジェクト、CL プログラム変数、システム値、および組み込み関数の名前は、個々のコマンドのパラメーターの中で指定できます。CL プログラムのほとんどのパラメーターでは、定数値の代わりに CL 変数名を使用することによって、プログラムの実行中に変化する値を指定できます。コマンドの実行時に使用されるオブジェクトおよび変数は、変数の内容で指定します。

関連資料:

40 ページの『CL コマンド・ラベル』
コマンド・ラベルは、CL プログラム内でブランチを行うために特定のコマンドを識別します。また、デバッグ中の CL プログラム内のステートメントを識別するためにも、ラベルが使用されます。ラベルは、ブレークポイントとして使用するステートメントや、トレースのための開始ステートメントおよび終了ステートメントを識別することができます。

63 ページの『変数名』
変数には、プログラムの実行中に変更できるデータの値が含まれています。 変数名は、値を含む変数の名前です。

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

CL コマンドで使用されるデータベース・ファイルおよび装置ファイル

IBM 提供の CL コマンドの多くは、処理時にデータベース・ファイルおよび装置ファイルを使用します。

基準を満たすすべてのライセンス・プログラムのすべてのコマンドおよびファイルが含まれています。

- 記載されているファイルのタイプは、以下のとおりです。
  - データベース・ファイル: データが入っているファイルや、モデル・ファイルとして使用されるファイル (データなし) を含む、物理ファイル (PF) および論理ファイル (LF)。
  - 装置ファイル: テープ (TAPF)、および印刷装置 (PRTF)
- プログラムで宣言して別のファイルで指定変更できるようにするなど、ユーザーが使用する可能性のあるファイルのみが記載してあります。使用例を 2 つ示します。
  - 使用するフォント (FONT パラメーター) や印刷される 1 インチあたりの行数 (LPI パラメーター) など、印刷装置ファイルの属性のいくつかを変更する
  - IBM 提供のファイルをユーザー自身の出力ファイルで指定変更する (可能な場合)

この項に記載されていないファイル・タイプは、以下のとおりです。

- IBM 提供の表示 (DSPF) 装置ファイル。これらのファイルを変更したり、指定変更したりしてはならないためです。
- ディレクトリー・コマンド、文書ライブラリー・オブジェクト・コマンド、および光ディスク索引データベース・ファイルによって使用される、IBM 提供のデータベース・ファイルの大部分。これらのファイルは指定変更できないためです。

上述のように、特定のコマンドのモデル・ファイルとして使用される IBM 提供の物理 (PF) ファイルまたは論理 (LF) ファイルが、次の表にまとめてあります。 DSPFD コマンド、DSPJRN コマンド、および STRPFRMON コマンドのところにリストされているモデル・ファイルがその例です。ほとんどの場合、これらのモデル・ファイルにはデータは含まれません。代わりに、コマンドを使用した結果得られる実際の出力データを保管するファイルの定義 (すなわちレコード様式) を含んでいます。これらのファイルのレコード様式については、オンラインでファイル記述を表示するか、そのコマンドについて記載しているトピック・コレクションを参照してください。

さらに、IBM 提供のファイルであると見なされるファイルの中には、そのファイルを必要とする機能が使用されるまでは実際にはシステムに存在せず、必要になった時点で作成されるものもあります。

次の注では、コマンドおよびファイルの表の分類方法と、 174 ページの表 22 で使われている肩文字の意味について説明します。

注:

1. 表の最初の欄には、IBM 提供のファイル (3 番目の欄に示されています) を使用する CL コマンドをリストします。表の中では、まず、コマンド名 をアルファベット順に項目を並べています。1 つのコマンドに対して複数のライブラリーが存在する場合、ファイルのライブラリー名 順に項目を並べており、さらに、各ライブラリー内のファイル名 ごとに並べられています。
2. ファイルの説明に肩文字 **1 (1)** が付いているファイルは、コマンドからの出力がそのコマンドの出力関連のパラメーターによって出力される形式 に送信される場合にのみ使用されます。
   - 番号[1] が印刷装置ファイル (PRTF) 記述の末尾に付いている場合は、印刷装置ファイルが、ジョブ環境、およびコマンドに指定 (想定) された印刷関連の値に応じて使用されることを示します。コマンド (主に DSPxxx コマンドおよび WRKxxx コマンド) に OUTPUT パラメーターを指定する場合、バッチ・ジョブで OUTPUT(*) を指定するか、バッチ・ジョブまたは対話式ジョブで OUTPUT(*PRINT) を指定すると、出力が印刷されます。その他のコマンドの場合は、それぞれのパラメーターで *PRINT、*LIST、*SRC の値を指定すると、出力が印刷されます。
   - 番号[1] がデータベース・ファイル (PF または LF) 記述の末尾に付いている場合は、データベース・ファイルがコマンドに指定 (想定) された印刷関連の値に応じて使用されることを示します。 コマンド (主に DSPxxx コマンドおよび SAVxxx コマンド) に OUTPUT パラメーターを指定する場合、OUTPUT(*OUTFILE) を指定すると、出力がデータベース・ファイルに送信されます。*OUTFILE に類似する値をパラメーターの 1 つに指定するその他のコマンドについても、同じことが言えます。
3. ファイルの説明に肩文字 **2 (2)** が付いているファイルは、出力ファイルではなくモデル・ファイル です。モデル・ファイルでは、実際の出力を入れるために作成されるファイルのレコード様式を定義します。
4. ファイル・ライブラリーとして「ユーザー・ライブラリー」と示されているファイルは、ユーザーがこれらのファイルを作成しない限り、システムには存在しません。このコマンドを使用すると、ユーザーの指定したライブラリーに、示されたファイル名でファイルが作成されます。

表 *22. CL* コマンドで使用されるファイル *(*パート *1)*

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
ADDDSTQ	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
ADDDSTRTE	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
	QUSRSYS	QASNADSR	PF	SNADS 経路指定テーブル。
ADDDSTSYSN	QUSRSYS	QASNADSA	PF	SNADS 2 次ノード ID テーブル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
ADDNETJOBE	QUSRSYS	QANFNJE	PF	ネットワーク・ジョブ項目データベース・ファイル。
ADDSOCE	QUSRSYS	QAALSOC	PF	制御範囲ファイル。
ADDTAPCTG	QUSRSYS	QATAMID	PF	カートリッジ ID DB ファイル。
	QUSRSYS	QLTAMID	LF	カートリッジ ID 論理ファイル。
	QUSRSYS	QATACGY	PF	カテゴリー DB ファイル。
	QUSRSYS	QLTACGY	LF	カテゴリー論理ファイル。
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
ANSQST	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
ANZDBF	QPFR	QAPTAZDR	PF	パフォーマンス・データ収集ファイル: アプリケーション・データベース・ファイル分析データ。
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTANZD	PRTF	物理から論理へのデータベース・ファイルの関係と論理から物理へのデータベース・ファイルの関係を示すパフォーマンス印刷装置ファイル。
ANZDBFKEY	QPFR	QAPTAZDR	PF	論理ファイルのキー構造を示す、アプリケーション・データベース・ファイル分析データのパフォーマンス・入力ファイル。
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTANKM	PRTF	論理ファイル・キー構造データを含むパフォーマンス印刷装置ファイル。
	QPFR	QPPTANZK	PRTF	アクセス・パスおよびレコード選択データを含むパフォーマンス印刷装置ファイル。
ANZPFRDTA	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
	QPFR	QPAVPRT	PRTF	アドバイザーのレポートを含む、パフォーマンス印刷装置ファイル。 [1]
ANZPGM	QPFR	QAPTAZPD	PF	パフォーマンス・データ収集ファイル: アプリケーション・プログラム分析データ。
	QPFR	QPPTANZP	PRTF	プログラムからファイルへの関係とファイルからプログラムへの関係を示すパフォーマンス印刷装置ファイル。
APYJRNCHG	QSYS	QAJRNCHG	PF	ジャーナル処理済み変更適用のモデル出力ファイル。
ASKQST	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
CFGDSTSRV	QUSRSYS	QASNADSA	PF	SNADS 2 次ノード ID テーブル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QUSRSYS	QASNADSQ	PF	SNADS 宛先待ち行列テーブル。
	QUSRSYS	QASNADSR	PF	SNADS 宛先システム経路指定テーブル。
	QUSRSYS	QATOCIFC	PF	TCP/IP インターフェース・ファイル。
	QUSRSYS	QATOCPORT	PF	TCP/IP ポート制約条件ファイル。
	QUSRSYS	QATOCPS	PF	TCP/IP サービス・ファイル。
	QUSRSYS	QATOCRSI	PF	TCP/IP RSI ファイル。
	QUSRSYS	QATOCRTE	PF	TCP/IP 経路ファイル。
	QUSRSYS	QATOCTCPIP	PF	TCP/IP 属性ファイル。
CFGTCPSMTP	QUSRSYS	QATMSMTP	PF	TCP/IP SMTP ファイル。
	QUSRSYS	QATMSMTPA	PF	TCP/IP SMTP ファイル。
CHGDTA	QIDU	QDTALOG	PRTF	監査制御ログ印刷装置ファイル。
	QIDU	QDTAPRT	PRTF	レコードおよびアキュムレーターの合計印刷装置ファイル。
	QSYS	QPDZDTALOG	PRTF	DFU 実行時監査ログ。
	QSYS	QPDZDTAPRT	PRTF	DFU 実行時印刷装置データ・ファイル。
CHGDSTQ	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
CHGDSTRTE	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
	QUSRSYS	QASNADSR	PF	SNADS 経路指定テーブル。
CHGFTPA	QUSRSYS	QATMFTP	PF	TCP/IP FTP 構成ファイル。
CHGHTTPA	QUSRSYS	QATMHTTP	PF	TCP/IP HTTP ファイル。
CHGLPDA	QUSRSYS	QATMLPD	PF	TCP/IP LPD 構成ファイル。
CHGPOPA	QUSRSYS	QATMPOPA	PF	POP サーバー構成ファイル。
CHGPRB	QUSRSYS	QASXNOTE	PF	問題ログのユーザー注記ファイル。
	QUSRSYS	QASXPROB	PF	問題ログの問題ファイル。
CHGQSTDB	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
CHGSMTPA	QUSRSYS	QATMSMTP	PF	TCP/IP SMTP 構成ファイル。
CHGTAPCTG	QUSRSYS	QATAMID	PF	カートリッジ ID データベース・ファイル。
	QUSRSYS	QLTAMID	LF	カートリッジ ID 論理ファイル。
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
CHGTELNA	QUSRSYS	QATMTELN	PF	TCP/IP TELNET 構成ファイル。
CHKTAP	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
CMPJRNIMG	QSYS	QPCMPIMG	PRTF	ジャーナル・イメージ比較印刷装置ファイル。
CPYF	QSYS	QSYSPRT	PRTF	コピー・ファイル印刷装置ファイル。[1]
CPYFRMQRYF	QSYS	QSYSPRT	PRTF	コピー・ファイル印刷装置ファイル。[1]
CPYFRMTAP	QSYS	QSYSPRT	PRTF	コピー・ファイル印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	入出力に使用するテープ装置ファイル。
CPYPFRCOL	QPFR	QAPGSUMD	PF	グラフィック・データのパフォーマンス・データ収集ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYDWxxxx	PF	ディスク・ウォッチャーのパフォーマンス・データ・モデル・ファイル。モデル・ファイルの名前と説明については、『Disk Watcher データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYJWxxxx	PF	ジョブ・ウォッチャーのパフォーマンス・データ・モデル・ファイル。 [2]
	QSYS	QAYPExxxx	PF	Performance Explorer (PEX) のデータ・モデル・ファイル。[2]
	QPFR	QAPTLCKD	PF	パフォーマンス・データ収集ファイル: ロックと競合捕そくデータ。
CPYSRCF	QSYS	QSYSPRT	PRTF	コピー・ファイル印刷装置ファイル。[1]
CRTBNDC	QGPL	QCSRC	PF	ILE C ソース・デフォルト入力ファイル。
CRTBNDCBL	QGPL	QCBLLESRC	PF	ILE COBOL ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PF	ILE COBOL ソース・リスト印刷装置ファイル。
CRTBNDCL	QGPL	QCLSRC	PF	CL ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	CL ソース・リスト印刷装置ファイル。[1]
CRTBNDRPG	QGPL	QRPGLESRC	PF	ILE RPG ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	ILE RPG ソース・リスト印刷装置ファイル。
CRTCBLMOD	QGPL	QCBLLESRC	PF	ILE COBOL ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PF	ILE COBOL ソース・リスト印刷装置ファイル。
CRTCBLPGM	QGPL	QLBLSRC	PF	OPM COBOL ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	OPM COBOL ソース・リスト印刷装置ファイル。[1]
CRTCLD	QGPL	QCLDSRC	PF	C ロケール記述デフォルト・ソース入力ファイル。
	QSYS	QSYSPRT	PRTF	C ロケール記述ソース・リスト印刷装置ファイル。 [1]
CRTCLMOD	QGPL	QCLSRC	PF	CL ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	CL ソース・リスト印刷装置ファイル。[1]
CRTCLPGM	QGPL	QCLSRC	PF	CL ソース・デフォルト入力ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QSYSPRT	PRTF	CL ソース・リスト印刷装置ファイル。[1]
CRTCMD	QGPL	QCMDSRC	PF	コマンド定義ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	コマンド定義ソース・リスト印刷装置ファイル。
CRTCMOD	QGPL	QCSRC	PF	ILE C ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	ILE C ソース・リスト印刷装置ファイル。[1]
CRTDSPF	QGPL	QDDSSRC	PF	DDS ソース・デフォルト入力ファイル。
	QSYS	QPDDSSRC	PRTF	DDS ソース・リスト印刷装置ファイル。[1]
CRTICFF	QGPL	QDDSSRC	PF	DDS ソース・デフォルト入力ファイル。
	QSYS	QPDDSSRC	PRTF	DDS ソース・リスト印刷装置ファイル。[1]
CRTLF	QGPL	QDDSSRC	PF	DDS ソース・デフォルト入力ファイル。
	QSYS	QPDDSSRC	PRTF	DDS ソース・リスト印刷装置ファイル。[1]
CRTMNU	QGPL	QMNUSRC	PF	デフォルトのメニュー・ソース入力ファイル。
	QSYS	QSYSPRT	PRTF	メニュー・ソース・リスト印刷装置ファイル。[1]
CRTMSGFMNU	QGPL	QDDSSRC	PF	$BMENU によってメニュー用に作成された DDS ソース。
	QGPL	QS36DDSSRC	PF	$BMENU によってメニュー用に作成された DDS ソース。
	QSSP	QPUTMENU	PRTF	$BMENU ソース・リスト印刷装置ファイル。
	QSYS	QSYSPRT	PRTF	Pascal ソース・リスト印刷装置ファイル。[1]
CRTPF	QGPL	QDDSSRC	PF	DDS ソース・デフォルト入力ファイル。
	QSYS	QPDDSSRC	PRTF	DDS ソース・リスト印刷装置ファイル。[1]
CRTPFRDTA	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。[2]
CRTPGM	QSYS	QSYSPRT	PRTF	プログラム・ソース・リスト印刷装置ファイル。[1]
CRTPNLGRP	QGPL	QPNLSRC	PF	デフォルトのパネル・グループ・ソース入力ファイル。
	QSYS	QSYSPRT	PRTF	パネル・グループ・ソース・リスト印刷装置ファイル。[1]
CRTPRTF	QGPL	QDDSSRC	PF	DDS ソース・デフォルト入力ファイル。
	QSYS	QPDDSSRC	PRTF	DDS ソース・リスト印刷装置ファイル。[1]
CRTQSTDB	QSYS	QAQA00xxxx	LF	質問回答データベース・モデル・ファイル。[2]
	QSYS	QAQA00xxxx	PF	質問回答データベース・モデル・ファイル。[2]
CRTQSTLOD	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
CRTRJEBSCF	QRJE	QRJESRC	PF	RJE BSC ファイルの作成に使用する DDS ソース・ファイル。
CRTRJECFG	QRJE	QRJESRC	PF	RJE BSC または RJE 通信ファイルの作成に使用する DDS ソース・ファイル。
CRTRJECMNF	QRJE	QRJESRC	PF	RJE 通信ファイルの作成に使用する DDS ソース・ファイル。
CRTRPGMOD	QGPL	QRPGLESRC	PF	ILE RPG ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	ILE RPG ソース・リスト印刷装置ファイル。[1]
CRTRPGPGM	QGPL	QRPGSRC	PF	RPG ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	RPG ソース・リスト印刷装置ファイル。[1]
CRTRPTPGM	QGPL	QRPGSRC	PF	RPG ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	RPG ソース・リスト印刷装置ファイル。[1]
CRTSQLC	QGPL	QCSRC	PF	SQL C ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL C 印刷装置ファイル。[1]
CRTSQLCI	QGPL	QCSRC	PF	SQL C ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL C 印刷装置ファイル。[1]
CRTSQLCBL	QGPL	QLBLSRC	PF	SQL COBOL ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL COBOL 印刷装置ファイル。[1]
CRTSQLCBLI	QGPL	QCBLLESRC	PF	SQL COBOL ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL COBOL 印刷装置ファイル。[1]
CRTSQLPLI	QGPL	QPLISRC	PF	SQL PLI ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL PLI 印刷装置ファイル。[1]
CRTSQLRPG	QGPL	QRPGSRC	PF	SQL RPG ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL RPG 印刷装置ファイル。[1]
CRTSQLRPGI	QGPL	QRPGLESRC	PF	SQL RPG ソース・ファイル。
	QSYS	QSYSPRT	PRTF	SQL RPG 印刷装置ファイル。[1]
CRTSRVPGM	QSYS	QSYSPRT	PRTF	サービス・プログラム・ソース・リスト印刷装置ファイル。[1]
CRTS36CBL	#LIBRARY	QS36SRC	PF	S/36 互換 COBOL ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	S/36 互換 COBOL ソース・リスト印刷装置ファイル。[1]
CRTS36DSPF	QGPL	QDDSSRC	PF	$SFGR によって表示装置ファイル用に作成された DDS ソース。
	QGPL	QS36DDSSRC	PF	$SFGR によって表示装置ファイル用に作成された DDS ソース。
	QSSP	QPUTSFGR	PRTF	$SFGR ソース・リスト印刷装置ファイル。
CRTS36MNU	QGPL	QDDSSRC	PF	$BMENU によってメニュー用に作成された DDS ソース。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QGPL	QS36DDSSRC	PF	$BMENU によってメニュー用に作成された DDS ソース。
	QSSP	QPUTMENU	PRTF	$BMENU ソース・リスト印刷装置ファイル。
CRTS36RPG	#LIBRARY	QS36SRC	PF	システム/36 RPG II ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	システム/36 RPG II ソース・リスト印刷装置ファイル。[1]
CRTS36RPGR	#LIBRARY	QS36SRC	PF	システム/36 RPG II ソース・デフォルト入力ファイル。
CRTS36RPT	#LIBRARY	QS36SRC	PF	システム/36 RPG II 報告書簡易作成機能ソース・デフォルト入力ファイル。
	QSYS	QSYSPRT	PRTF	システム/36 RPG II 報告書簡易作成機能のソース・リスト印刷装置ファイル。[1]
CRTTAPCGY	QUSRSYS	QATACGY	PF	ライブラリー装置データベース・ファイル。
	QUSRSYS	QLTACGY	LF	ライブラリー装置論理データベース・ファイル。
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
CRTTBL	QGPL	QTBLSRC	PF	テーブル・ソース・デフォルト入力ファイル。
CVTPFRCOL	QPFR	QAPGSUMD	PF	グラフィック・データのパフォーマンス・データ収集ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。[2]
	QSYS	QAPYDWxxxx	PF	ディスク・ウォッチャーのパフォーマンス・データ・モデル・ファイル。モデル・ファイルの名前と説明については、『Disk Watcher データ・ファイル』のトピックを参照してください。[2]
	QSYS	QAPYJWxxxx	PF	ジョブ・ウォッチャーのパフォーマンス・データ・モデル・ファイル。[2]
	QSYS	QAYPExxxx	PF	Performance Explorer (PEX) のデータ・モデル・ファイル。[2]
	QPFR	QAPTLCKD	PF	パフォーマンス・データ収集ファイル: ロックと競合捕そくデータ。
CVTRPGSRC	QSYS	QSYSPRT	PRTF	ILE RPG リスト印刷装置ファイル。
	ユーザー・ライブラリー	QRNCVTLG	PF	ILE RPG 変換ログ・ファイル。
	ユーザー・ライブラリー	QRPGLESRC	PF	ILE RPG ソース・ファイル (宛先ファイル)。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	ユーザー・ライブラリー	QRPGSRC	PF	RPG/400® ソース・ファイル (元ファイル)。
DLTALR	QUSRSYS	QAALERT	PF	警報データベース・ファイル。
DLTPFRCOL	QPFR	QAPGSUMD	PF	グラフィック・データのパフォーマンス・データ収集ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYDWxxxx	PF	ディスク・ウォッチャーのパフォーマンス・データ・モデル・ファイル。モデル・ファイルの名前と説明については、『Disk Watcher データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYJWxxxx	PF	ジョブ・ウォッチャーのパフォーマンス・データ・モデル・ファイル。 [2]
	QSYS	QAYPExxxx	PF	Performance Explorer (PEX) のデータ・モデル・ファイル。[2]
	QPFR	QAPTLCKD	PF	パフォーマンス・データ収集ファイル: ロックと競合捕そくデータ。
DLTPRB	QUSRSYS	QASXxxxx	PF	DLTPRB コマンド用の 8 つの QASXxxxx ファイルは、すべて、DSPPRB コマンドの QUSRSYS ライブラリーに示したファイルのサブセットと同じです。
DLTQST	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
DLTQSTDB	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
DLTTAPCGY	QUSRSYS	QATACGY	PF	ライブラリー装置データベース・ファイル。
	QUSRSYS	QLTACGY	LF	ライブラリー装置論理データベース・ファイル。
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
DMPCLPGM	QSYS	QPPGMDMP	PRTF	CL プログラム・ダンプ印刷装置ファイル。
DMPJOB	QSYS	QPSRVDMP	PRTF	サービス・ダンプ印刷装置ファイル。
DMPOBJ	QSYS	QPSRVDMP	PRTF	サービス・ダンプ印刷装置ファイル。
DMPSYSOBJ	QSYS	QPSRVDMP	PRTF	サービス・ダンプ印刷装置ファイル。
DMPTAP	QSYS	QPTAPDMP	PRTF	テープ・ダンプ印刷装置ファイル。
	QSYS	QSYSTAP	TAPF	入出力に使用するテープ装置ファイル。
DMPTRC	QSYS	QAPMDMPT	PF	パフォーマンス・トレース・ファイル。
	QSYS	QSYSPRT	PRTF	SDA ソース印刷装置ファイル。
DSPACC	QSYS	QSYSPRT	PRTF	アクセス・コード表示印刷装置ファイル。[1]
DSPACCAUT	QSYS	QSYSPRT	PRTF	アクセス・コード表示権限印刷装置ファイル。 [1]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPACTPJ	QSYS	QSYSPRT	PRTF	活動事前開始ジョブ表示印刷装置ファイル。[1]
DSPAPPNINF	QUSRSYS	QALSxxx	PF	APPN 情報の保管に使用するレコード様式を含む 4 つの QALSxxx モデル・データベース・ファイルのセット。ここで、xxx は、DIR、END、INM、および TDB です。[2]
	QSYS	QSYSPRT	PRTF	APPN 情報表示印刷装置ファイル。[1]
DSPAUTHLR	QSYS	QADSHLR	PF	権限ホルダー・オブジェクト項目のレコード様式を含むモデル・データベース・ファイル。[2]
	QSYS	QPSYDSHL	PRTF	権限ホルダー表示印刷装置ファイル。[1]
DSPAUTL	QSYS	QAOBJAUT	PF	権限リスト項目のレコード様式を含むモデル・データベース・ファイル。[2]
	QSYS	QPOBJAUT	PRTF	権限リスト項目印刷装置ファイル。[1]
DSPAUTLDLO	QSYS	QSYSPRT	PRTF	権限リスト印刷装置ファイル。[1]
DSPAUTLOBJ	QSYS	QADALO	PF	権限リスト・オブジェクト項目のレコード様式を含むモデル・データベース・ファイル。[2]
	QSYS	QPSYDALO	PRTF	権限リスト・オブジェクト表示印刷装置ファイル。[1]
DSPAUTUSR	QSYS	QPAUTUSR	PRTF	許可ユーザー印刷装置ファイル。[1]
DSPBCKSTS	QSYS	QSYSPRT	PRTF	バックアップ状況表示印刷装置ファイル。[1]
DSPBCKUP	QSYS	QSYSPRT	PRTF	バックアップ・オプション表示印刷装置ファイル。[1]
DSPBCKUPL	QSYS	QSYSPRT	PRTF	バックアップ・リスト表示印刷装置ファイル。[1]
DSPBKP	QSYS	QPDBGDSP	PRTF	ブレークポイント (デバッグ・モード) 印刷装置ファイル。[1]
DSPBNDDIR	QSYS	QABNDBND	PF	バインド・ディレクトリー項目のレコード様式を含むモデル・データベース・ファイル。[2]
	QSYS	QSYSPRT	PRTF	バインド・ディレクトリー内容表示印刷装置ファイル。[1]
DSPCFGL	QSYS	QPDCCFGL	PRTF	構成リスト印刷装置ファイル。[1]
DSPCHT	QSYS	QPGDDM	PRTF	BGU 定義の図表出力印刷装置ファイル。[1]
DSPCLS	QSYS	QPDSPCLS	PRTF	クラス印刷装置ファイル。[1]
DSPCMD	QSYS	QPCMD	PRTF	コマンド値印刷装置ファイル。[1]
DSPCNNL	QSYS	QPDCCNNL	PRTF	接続リスト印刷装置ファイル。[1]
DSPCNNSTS	QSYS	QSYSPRT	PRTF	接続状況印刷装置ファイル。[1]
DSPCOSD	QSYS	QPDCCOS	PRTF	サービス・クラス記述印刷装置ファイル。[1]
DSPCSI	QSYS	QSYSPRT	PRTF	通信サイド情報印刷装置ファイル。[1]
DSPCTLD	QSYS	QPDCCTL	PRTF	制御装置記述印刷装置ファイル。[1]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPDBG	QSYS	QPDBGDSP	PRTF	デバッグ表示 (デバッグ・モード) 印刷装置ファイル。 [1]
DSPDBR	QSYS	QADSPDBR	PF	データベース・ファイルの関係に関する情報を保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QPDSPDBR	PRTF	データベース・ファイルの関係に関する情報を含む印刷装置ファイル。 [1]
DSPDDMF	QSYS	QPDSPDDM	PRTF	分散データ管理 (DDM) ファイル・リスト印刷装置ファイル。 [1]
DSPDEVD	QSYS	QPDCDEV	PRTF	装置記述印刷装置ファイル。[1]
DSPDIRE	QSYS	QAOSDIRO	PF	ディレクトリー表示出力ファイル: OUTFILFMT(*TYPE1)
	QSYS	QAOSDIRB	PF	ディレクトリー表示出力ファイル: OUTFILFMT(*TYPE2) DETAIL(*BASIC)
	QSYS	QAOSDIRF	PF	ディレクトリー表示出力ファイル: OUTFILFMT(*TYPE2) DETAIL(*FULL)
	QSYS	QAOSDIRX	PF	ディレクトリー表示出力ファイル: OUTFILFMT(*TYPE3) DETAIL(*FULL)
	QSYS	QPDSPDDL	PRTF	表示されるディレクトリー項目の全 詳細用の印刷装置ファイル。[1]
	QSYS	QPDSPDSM	PRTF	表示されるディレクトリー項目の基本 詳細用の印刷装置ファイル。[1]
DSPDLOAUT	QSYS	QSYSPRT	PRTF	文書ライブラリー・オブジェクト権限表示印刷装置ファイル。 [1]
DSPDLONAM	QSYS	QSYSPRT	PRTF	文書ライブラリー・オブジェクト表示印刷装置ファイル。 [1]
DSPDSTL	QSYS	QAOSDSTO	PF	配布リスト出力ファイル。
	QSYS	QPDSPLDL	PRTF	配布リストの詳細 印刷装置ファイル。[1]
	QSYS	QPDSPLSM	PRTF	配布リストの要約 印刷装置ファイル。[1]
DSPDSTCLGE	QSVMSS	QACQFVOF	PF	MSS/400 配布カタログ項目表示コマンドの出力ファイル・モデル。 [2]
DSPDSTLOG	QSYS	QPDSTDLG	PRTF	分散ログ表示印刷装置ファイル。 [1]
DSPDSTSRV	QSYS	QPDSTSRV	PRTF	配布サービス印刷装置ファイル。[1]
	QUSRSYS	QASNADSA	PF	SNADS 2 次ノード ID テーブル。
	QUSRSYS	QASNADSQ	PF	SNADS 宛先待ち行列テーブル。
	QUSRSYS	QASNADSR	PF	経路指定テーブル・データベース・ファイル。
DSPDTA	QIDU	QDTAPRT	PRTF	DFU 監査制御印刷装置ファイル。
	QSYS	QPDZDTALOG	PRTF	DFU 実行時監査ログ。
	QSYS	QPDZDTAPRT	PRTF	DFU 実行時印刷装置データ・ファイル。
DSPDTAARA	QSYS	QPDSPDTA	PRTF	データ域印刷装置ファイル。[1]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPDTADCT	QSYS	QPDSPFFD	PRTF	データ・ディクショナリー印刷装置ファイル。[1]
DSPEDTD	QSYS	QPDCEDSP	PRTF	編集記述印刷装置ファイル。[1]
DSPFD	DSPFD コマンドの場合、次の項目のうちファイル・タイプが PF であるものはすべて物理ファイル (実際の出力ファイルではなく、モデル・ファイル) であり、ファイルのタイプ (またはグループ) に関する特定のタイプの情報を保管するために作成されるファイルのレコード様式を定義します。すなわち、各モデル・ファイル記述の共通部分は、下記の太字部分で始まります。続く各記述の固有の部分は、ファイルの用途の欄に記載してあります。				

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QAFDACCP	PF	アクセス・パス・ファイル情報...[2]
	QSYS	QAFDBASI	PF	すべてのファイルに共通の基本的なファイル情報... [2]
	QSYS	QAFDBSC	PF	BSC ファイルおよび混合ファイルの装置属性情報... [2]
	QSYS	QAFDCMN	PF	通信ファイルおよび混合ファイルの装置属性情報... [2]
	QSYS	QAFDCSEQ	PF	照合順序情報...[2]
	QSYS	QAFDCST	PF	制約関係情報... [2]
	QSYS	QAFDDDM	PF	分散データ管理 (DDM) ファイル属性情報... [2]
	QSYS	QAFDDSP	PF	表示装置ファイルおよび混合ファイルの表示装置属性情報... [2]
	QSYS	QAFDICF	PF	ICF ファイル属性情報...[2]
	QSYS	QAFDJOIN	PF	結合論理ファイル情報...[2]
	QSYS	QAFDLGL	PF	論理ファイル属性情報... [2]
	QSYS	QAFDMBR	PF	データベース・メンバー情報...[2]
	QSYS	QAFDMBRL	PF	データベース・メンバー・リスト情報...[2]
	QSYS	QAFDNGP	PF	ノード・グループ情報...[2]
	QSYS	QAFDPHY	PF	物理ファイル属性情報... [2]
	QSYS	QAFDPRT	PF	印刷装置ファイル属性情報...[2]
	QSYS	QAFDRFMT	PF	レコード様式情報...[2]
	QSYS	QAFDSAV	PF	保管ファイル情報...[2]
	QSYS	QAFDSELO	PF	選択/省略情報...[2]
	QSYS	QAFDSPOL	PF	装置ファイルのスプール情報...[2]
	QSYS	QAFDTAP	PF	テープ・ファイル属性情報... [2]
	QSYS	QAFDTRG	PF	トリガー情報...[2]
	QSYS	QPDSPFD	PRTF	ファイル記述印刷装置ファイル。[1]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPFFD	QSYS	QADSPFFD	PF	ファイル・フィールド記述を保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。[2]
	QSYS	QPDSPFFD	PRTF	ファイル・フィールド記述印刷装置ファイル。[1]
DSPFLR	QSYS	QADSPDOC	PF	文書リスト出力データベース・ファイル。
	QSYS	QADSPFLR	PF	フォルダー・リスト出力データベース・ファイル。
	QSYS	QPDSPFLR	PRTF	フォルダー表示印刷装置ファイル。[1]
DSPFNTRSCA	QSYS	QPDSPFNT	PRTF	フォント資源属性印刷装置ファイル。[1]
DSPGDF	QSYS	QPGDDM	PRTF	BGU 定義のグラフィック・データ印刷装置ファイル。[1]
DSPHDWRSC	DSPHDWRSC コマンドの場合、次の項目のうちファイル・タイプが PF であるものはすべて物理ファイル (実際の出力ファイルではなく、モデル・ファイル) であり、特定のタイプのハードウェア資源情報を保管するために作成されるファイルのレコード様式を定義します。すなわち、各モデル・ファイル記述の共通部分は、下記の太字部分で始まります。続く各記述の固有の部分は、ファイルの用途の欄に記載してあります。			

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QARZDCMN	PF	通信資源...[2]
	QSYS	QARZDLWS	PF	ローカル・ワークステーションの資源...[2]
	QSYS	QARZDPRC	PF	プロセッサー資源...[2]
	QSYS	QARZDTRA	PF	トークンリング LAN (TRLAN) アダプター資源...[2]
	QSYS	QARZDSTG	PF	記憶装置資源...[2]
	QSYS	QSYSPRT	PRTF	ハードウェア資源印刷装置ファイル。[1]
DSPHFS	QSYS	QSYSPRT	PRTF	階層ファイル・システム表示印刷装置ファイル。[1]
DSPHSTGPH	QPFR	QPPGGPH	PRTF	履歴グラフ表示印刷装置ファイル。[1]
DSPIGCDCT	QSYS	QPDSPDCT	PRTF	DBCS 印刷装置ファイル。[1]
DSPJOB	QSYS	QPDSPJOB	PRTF	ジョブ表示印刷装置ファイル。[1]
DSPJOBD	QSYS	QPRTJOBD	PRTF	ジョブ記述印刷装置ファイル。[1]
DSPJOBLOG	QSYS	QPJOBLOG	PRTF	ジョブ・ログ印刷装置ファイル。[1]
	QSYS	QPJOBLOGO	PRTF	バージョン 2 リリース 3 より前の AS/400 のジョブ用のジョブ・ログ印刷装置ファイル。[1]
	QSYS	QAMHJLPR	PF	1 次ジョブ・ログ・モデル・ファイル。[2]
	QSYS	QAMHJLSC	PF	2 次ジョブ・ログ・モデル・ファイル。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPJRN	DSPJRN コマンドの場合、以下のファイルのうち、ファイル・タイプが PF のものは、すべて物理ファイル (実際の出力ファイルではなく、モデル・ファイル) であり、ジャーナル・レシーバーから取得および変換されたジャーナル項目のグループを保管するために作成されるファイルのレコード様式を定義します。検索する項目のグループは、ジャーナル処理されていれば、特定のタイプの情報でもすべてのタイプの情報でも構いません。作成された各ファイルには、検出されたジャーナル項目が、5 つの基本的な形式 (*TYPE1、*TYPE2、*TYPE3、*TYPE4、または *TYPE5) のいずれか、あるいは検出対象の特定のタイプのデータに定義された形式に変換されて保管されます。 • *TYPE1:『ジャーナル管理』のトピック・コレクションで説明される基本ファイル様式。 • *TYPE2: *TYPE1 のすべてとユーザー・プロファイル・フィールド。 • *TYPE3: *TYPE2 のすべてとヌル値標識。 • *TYPE4: *TYPE3 のすべてと JID (参照保全およびトリガー情報)。 • *TYPE5: *TYPE4 のすべておよびその他の情報。 • タイプ依存形式 - 検索される情報の特定のタイプ (4 番目以降のファイルについては以下で説明しています) に関連する形式。例えば、モデル・ファイル QASY**AF**JE は、システムにおける権限障害 (**AF**) に関連するすべての検索されたジャーナル項目を保管するために、固有の形式を持ちます。 以下にリストされている DSPJRN PF ファイルについては、すべてのモデル・ファイル記述に、以下の太字で示すような共通部分があります。各ファイル記述の固有の部分は、ファイルの用途欄に記載されています。			

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QADSPJRN	PF	特定のタイプ (またはすべてのタイプ) の情報 (*TYPE1 の形式で保管されます)... [2]
	QSYS	QADSPJR2	PF	特定のタイプ (またはすべてのタイプ) の情報 (*TYPE2 の形式で保管されます)... [2]
	QSYS	QADSPJR3	PF	特定のタイプ (またはすべてのタイプ) の情報 (*TYPE3 の形式で保管されます)... [2]
	QSYS	QADSPJR4	PF	特定のタイプ (またはすべてのタイプ) の情報 (*TYPE4 の形式で保管されます)... [2]
	QSYS	QADSPJR5	PF	特定のタイプ (またはすべてのタイプ) の情報 (*TYPE5 の形式で保管されます)... [2]
	QSYS	QADXERLG	PF	DSNX ログ・エラー...[2]
	QSYS	QADXERL4	PF	DSNX ログ・エラー...; *TYPE4 の形式で保管されます。[2]
	QSYS	QADXJRNL	PF	DSNX ログ・データ...[2]
	QSYS	QADXJRN4	PF	DSNX ログ・データ...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAJBACG	PF	ジョブ会計...[2]
	QSYS	QAJBACG4	PF	ジョブ会計...; *TYPE4 の形式で保管されます。[2]
	QSYS	QALZALK	PF	無効なライセンス・キー...[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QALZALK4	PF	無効なライセンス・キー...; *TYPE4 の形式で保管されます。[2]
	QSYS	QALZALL	PF	使用限度の拡大...[2]
	QSYS	QALZALL4	PF	使用限度の拡大...; *TYPE4 の形式で保管されます。[2]
	QSYS	QALZALU	PF	使用限度を超過したライセンス・ユーザー...[2]
	QSYS	QALZALU4	PF	使用限度を超過したライセンス・ユーザー...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAPTACG	PF	印刷ジョブ会計...[2]
	QSYS	QAPTACG4	PF	印刷ジョブ会計...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAPTACG5	PF	印刷ジョブ会計...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYADJE	PF	属性の監査に対する変更...[2]
	QSYS	QASYADJ4	PF	属性の監査に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYADJ5	PF	属性の監査に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYAFJE	PF	権限の障害...[2]
	QSYS	QASYAFJ4	PF	権限の障害...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYAFJ5	PF	権限の障害...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYAPJE	PF	借用権限の使用...[2]
	QSYS	QASYAPJ4	PF	借用権限の使用...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYAPJ5	PF	借用権限の使用...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYAUJ5	PF	セキュリティー属性の変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCAJE	PF	オブジェクト権限 (権限リストまたは権限オブジェクト) に対する変更... [2]
	QSYS	QASYCAJ4	PF	オブジェクト権限 (権限リストまたは権限オブジェクト) に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCAJ5	PF	オブジェクト権限 (権限リストまたは権限オブジェクト) に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCDJE	PF	コマンド・ストリング...[2]
	QSYS	QASYCDJ4	PF	コマンド・ストリング...; *TYPE4 の形式で保管されます。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYCDJ5	PF	コマンド・ストリング...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCOJE	PF	システムで作成されたオブジェクト...[2]
	QSYS	QASYCOJ4	PF	システムで作成されたオブジェクト...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCOJ5	PF	システムで作成されたオブジェクト...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCPJE	PF	ユーザー・プロファイルの作成操作、変更操作、および復元操作... [2]
	QSYS	QASYCPJ4	PF	ユーザー・プロファイルの作成操作、変更操作、および復元操作...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCPJ5	PF	ユーザー・プロファイルの作成操作、変更操作、および復元操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCQJE	PF	*CRQD オブジェクトに対する変更...[2]
	QSYS	QASYCQJ4	PF	*CRQD オブジェクトに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCQJ5	PF	*CRQD オブジェクトに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCUJ4	PF	クラスター操作...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCUJ5	PF	クラスター操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCVJ4	PF	接続検査...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCVJ5	PF	接続検査...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYCYJ4	PF	暗号構成...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYCYJ5	PF	暗号構成...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYDIJ4	PF	ディレクトリー・サービス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYDIJ5	PF	ディレクトリー・サービス...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYDOJE	PF	システムから削除されたオブジェクト...[2]
	QSYS	QASYDOJ4	PF	システムから削除されたオブジェクト...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYDOJ5	PF	システムから削除されたオブジェクト...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYDSJE	PF	DST 機密保護担当者のパスワードのリセット... [2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYDSJ4	PF	DST 機密保護担当者のパスワードのリセット...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYDSJ5	PF	DST 機密保護担当者のパスワードのリセット...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYEVJ4	PF	環境変数...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYEVJ5	PF	環境変数...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYGRJ4	PF	汎用監査レコード...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYGRJ5	PF	汎用監査レコード...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYGSJE	PF	記述子の提供...[2]
	QSYS	QASYGSJ4	PF	記述子の提供...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYGSJ5	PF	記述子の提供...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYIMJ5	PF	...侵入モニター; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYIPJE	PF	プロセス間通信...[2]
	QSYS	QASYIPJ4	PF	プロセス間通信...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYIPJ5	PF	プロセス間通信...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYIRJ4	PF	IP 規則処置...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYIRJ5	PF	IP 規則処置...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYISJ4	PF	インターネット・セキュリティー管理...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYISJ5	PF	インターネット・セキュリティー管理...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYJDJE	PF	ジョブ記述の USER パラメーターに対する変更... [2]
	QSYS	QASYJDJ4	PF	ジョブ記述の USER パラメーターに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYJDJ5	PF	ジョブ記述の USER パラメーターに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYJSJE	PF	ジョブの変更...[2]
	QSYS	QASYJSJ4	PF	ジョブの変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYJSJ5	PF	ジョブの変更...; *TYPE5 の形式で保管されます。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYKFJ4	PF	キー・リング・ファイル...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYKFJ5	PF	キー・リング・ファイル...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYLDJE	PF	ディレクトリーのリンク/リンク解除/探索...[2]
	QSYS	QASYLDJ4	PF	ディレクトリーのリンク/リンク解除/探索...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYLDJ5	PF	ディレクトリーのリンク/リンク解除/探索...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYMLJE	PF	メール処置...[2]
	QSYS	QASYMLJ4	PF	メール処置...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYMLJ5	PF	メール処置...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYNAJE	PF	ネットワーク属性に対する変更...[2]
	QSYS	QASYNAJ4	PF	ネットワーク属性に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYNAJ5	PF	ネットワーク属性に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYNDJE	PF	ディレクトリーの検索違反...[2]
	QSYS	QASYNDJ4	PF	ディレクトリーの検索違反...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYNDJ5	PF	ディレクトリーの検索違反...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYNEJE	PF	末端地点違反...[2]
	QSYS	QASYNEJ4	PF	末端地点違反...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYNEJ5	PF	末端地点違反...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYOMJE	PF	オブジェクトの移動操作および名前変更操作... [2]
	QSYS	QASYOMJ4	PF	オブジェクトの移動操作および名前変更操作...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYOMJ5	PF	オブジェクトの移動操作および名前変更操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYORJE	PF	オブジェクト復元操作...[2]
	QSYS	QASYORJ4	PF	オブジェクト復元操作...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYORJ5	PF	オブジェクト復元操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYOWJE	PF	オブジェクトの所有権に対する変更...[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYOWJ4	PF	オブジェクトの所有権に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYOWJ5	PF	オブジェクトの所有権に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYO1JE	PF	シングル光ディスク・オブジェクト・アクセス...[2]
	QSYS	QASYO1J4	PF	シングル光ディスク・オブジェクト・アクセス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYO1J5	PF	シングル光ディスク・オブジェクト・アクセス...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYO2JE	PF	デュアル光ディスク・オブジェクト・アクセス...[2]
	QSYS	QASYO2J4	PF	デュアル光ディスク・オブジェクト・アクセス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYO2J5	PF	デュアル・光ディスク・オブジェクト・アクセス...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYO3JE	PF	光ディスク・ボリューム・アクセス...[2]
	QSYS	QASYO3J4	PF	光ディスク・ボリューム・アクセス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYO3J5	PF	光ディスク・ボリューム・アクセス...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYPAJE	PF	所有者の権限を借用するプログラムに対する変更 (CHGPGM)... [2]
	QSYS	QASYPAJ4	PF	所有者の権限を借用するプログラムに対する変更 (CHGPGM)...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYPAJ5	PF	所有者の権限を借用するプログラムに対する変更 (CHGPGM)...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYPGJE	PF	オブジェクト基本グループに対する変更...[2]
	QSYS	QASYPGJ4	PF	オブジェクト基本グループに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYPGJ5	PF	オブジェクト基本グループに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYPOJE	PF	印刷装置出力処置...[2]
	QSYS	QASYPOJ4	PF	印刷装置出力処置...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYPOJ5	PF	印刷装置出力処置...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYPSJE	PF	プロファイルのスワッピング...[2]
	QSYS	QASYPSJ4	PF	プロファイルのスワッピング...; *TYPE4 の形式で保管されます。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYPSJ5	PF	プロファイルのスワッピング...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYPWJE	PF	無効なパスワードまたはユーザー・プロファイル名の使用試行... [2]
	QSYS	QASYPWJ4	PF	無効なパスワードまたはユーザー・プロファイル名の使用試行...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYPWJ5	PF	無効なパスワードまたはユーザー・プロファイル名の使用試行...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYRAJE	PF	権限変更時のオブジェクトの復元... [2]
	QSYS	QASYRAJ4	PF	権限変更時のオブジェクトの復元...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYRAJ5	PF	権限変更時のオブジェクトの復元...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYRJJE	PF	ユーザー・プロファイル名を含むジョブ記述の復元... [2]
	QSYS	QASYRJJ4	PF	ユーザー・プロファイル名を含むジョブ記述の復元...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYRJJ5	PF	ユーザー・プロファイル名を含むジョブ記述の復元...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYROJE	PF	所有権が QDFTOWN に変更されたときのオブジェクトの復元... [2]
	QSYS	QASYROJ4	PF	所有権が QDFTOWN に変更されたときのオブジェクトの復元...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYROJ5	PF	所有権が QDFTOWN に変更されたときのオブジェクトの復元...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYRPJE	PF	所有者の権限を借用するプログラムの復元... [2]
	QSYS	QASYRPJ4	PF	所有者の権限を借用するプログラムの復元...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYRPJ5	PF	所有者の権限を借用するプログラムの復元...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYRQJE	PF	*CRQD オブジェクトの復元...[2]
	QSYS	QASYRQJ4	PF	*CRQD オブジェクトの復元...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYRQJ5	PF	*CRQD オブジェクトの復元...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYRUJE	PF	RSTAUT コマンドの使用による、ユーザー・プロファイルに対する権限復元操作... [2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYRUJ4	PF	RSTAUT コマンドの使用による、ユーザー・プロファイルに対する権限復元操作...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYRUJ5	PF	RSTAUT コマンドの使用による、ユーザー・プロファイルに対する権限復元操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYRZJE	PF	基本グループの復元時の変更 ... [2]
	QSYS	QASYRZJ4	PF	基本グループの復元時の変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYRZJ5	PF	基本グループの復元時の変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSDJE	PF	システム配布ディレクトリーに対する変更... [2]
	QSYS	QASYSDJ4	PF	システム配布ディレクトリーに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSDJ5	PF	システム配布ディレクトリーに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSEJE	PF	サブシステムの経路指定に対する変更...[2]
	QSYS	QASYSEJ4	PF	サブシステムの経路指定に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSEJ5	PF	サブシステムの経路指定に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSFJE	PF	スプール・ファイルでの処置...[2]
	QSYS	QASYSFJ4	PF	スプール・ファイルでの処置...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSFJ5	PF	スプール・ファイルでの処置...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSGJ4	PF	非同期シグナル...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSGJ5	PF	非同期シグナル...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSKJ4	PF	ソケット接続の保護...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSKJ5	PF	ソケット接続の保護...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSMJE	PF	システム管理の変更...[2]
	QSYS	QASYSMJ4	PF	システム管理の変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSMJ5	PF	システム管理の変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSOJE	PF	サーバー・セキュリティーの変更...[2]
	QSYS	QASYSOJ4	PF	サーバー・セキュリティーの変更...; *TYPE4 の形式で保管されます。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYSOJ5	PF	サーバー・セキュリティーの変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSTJE	PF	システム保守ツールの使用...[2]
	QSYS	QASYSTJ4	PF	システム保守ツールの使用...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSTJ5	PF	システム保守ツールの使用...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYSVJE	PF	システム値に対する変更...[2]
	QSYS	QASYSVJ4	PF	システム値に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYSVJ5	PF	システム値に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVAJE	PF	アクセス制御リストに対する変更...[2]
	QSYS	QASYVAJ4	PF	アクセス制御リストに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVAJ5	PF	アクセス制御リストに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVCJE	PF	接続の開始および終了...[2]
	QSYS	QASYVCJ4	PF	接続の開始および終了...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVCJ5	PF	接続の開始および終了...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVFJE	PF	サーバー・ファイルのクローズ...[2]
	QSYS	QASYVFJ4	PF	サーバー・ファイルのクローズ...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVFJ5	PF	サーバー・ファイルのクローズ...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVLJE	PF	アカウント限界の超過...[2]
	QSYS	QASYVLJ4	PF	アカウント限界の超過...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVLJ5	PF	アカウント限界の超過...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVNJE	PF	ネットワークのログオンおよびログオフ...[2]
	QSYS	QASYVNJ4	PF	ネットワークのログオンおよびログオフ...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVNJ5	PF	ネットワークのログオンおよびログオフ...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVOJ4	PF	妥当性検査リストに対する処置...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVOJ5	PF	妥当性検査リストに対する処置...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVPJE	PF	ネットワーク・パスワード・エラー...[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYVPJ4	PF	ネットワーク・パスワード・エラー...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVPJ5	PF	ネットワーク・パスワード・エラー...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVRJE	PF	ネットワーク資源へのアクセス...[2]
	QSYS	QASYVRJ4	PF	ネットワーク資源へのアクセス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVRJ5	PF	ネットワーク資源へのアクセス...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVSJE	PF	サーバー・セッションの開始および終了...[2]
	QSYS	QASYVSJ4	PF	サーバー・セッションの開始および終了...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVSJ5	PF	サーバー・セッションの開始および終了...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVUJE	PF	ネットワーク・プロファイルに対する変更...[2]
	QSYS	QASYVUJ4	PF	ネットワーク・プロファイルに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVUJ5	PF	ネットワーク・プロファイルに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYVVJE	PF	サービス状況に対する変更...[2]
	QSYS	QASYVVJ4	PF	サービス状況に対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYVVJ5	PF	サービス状況に対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYXDJ5	PF	...ディレクトリー・サービス拡張; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYX0J4	PF	ネットワーク認証...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYX0J5	PF	ネットワーク認証...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYX1J5	PF	...ID トークン; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYYCJE	PF	文書ライブラリー・オブジェクトに対する変更... [2]
	QSYS	QASYYCJ4	PF	文書ライブラリー・オブジェクトに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYYCJ5	PF	文書ライブラリー・オブジェクトに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYYRJE	PF	文書ライブラリー・オブジェクトの読み取り操作... [2]
	QSYS	QASYYRJ4	PF	文書ライブラリー・オブジェクトの読み取り操作...; *TYPE4 の形式で保管されます。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASYYRJ5	PF	文書ライブラリー・オブジェクトの読み取り操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYZCJE	PF	オブジェクトに対する変更...[2]
	QSYS	QASYZCJ4	PF	オブジェクトに対する変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYZCJ5	PF	オブジェクトに対する変更...; *TYPE5 の形式で保管されます。[2]
	QSYS	QASYZMJE	PF	オブジェクト・メソッド・アクセス...[2]
	QSYS	QASYZMJ4	PF	オブジェクト・メソッド・アクセス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYZRJE	PF	オブジェクトの読み取り操作...[2]
	QSYS	QASYZRJ4	PF	オブジェクトの読み取り操作...; *TYPE4 の形式で保管されます。[2]
	QSYS	QASYZRJ5	PF	オブジェクトの読み取り操作...; *TYPE5 の形式で保管されます。[2]
	QSYS	QATOFIPF	PF	IP フィルター・ルール処置...[2]
	QSYS	QATOFNAT	PF	IP NAT ルール処置...[2]
	QSYS	QATOQQOS	PF	QoS ポリシーの変更...[2]
	QSYS	QATOSLOG	PF	SNMP ログ項目...[2]
	QSYS	QATOSLO4	PF	SNMP ログ項目...; *TYPE4 の形式で保管されます。[2]
	QSYS	QATOVSOF	PF	VPN 情報...[2]
	QSYS	QAWCTPJE	PF	パフォーマンスの調整...[2]
	QSYS	QAWCTPJ4	PF	パフォーマンスの調整...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZDALLG	PF	SNADS 警報ロギング。[2]
	QSYS	QAZDCFLG	PF	SNADS 配布待ち行列テーブルに対する構成の変更... [2]
	QSYS	QAZDCFL4	PF	SNADS 配布待ち行列テーブルに対する構成の変更...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZDERLG	PF	SNADS ログ・エラー...[2]
	QSYS	QAZDERL4	PF	SNADS ログ・エラー...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZDJRNL	PF	SNADS ログ・データ...[2]
	QSYS	QAZDJRN4	PF	SNADS ログ・データ...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZDRTLG	PF	SNADS の経路指定テーブルおよび 2 次システム名テーブルに対する変更... [2]
	QSYS	QAZDRTL4	PF	SNADS の経路指定テーブルおよび 2 次システム名テーブルに対する変更...; *TYPE4 の形式で保管されます。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QAZDSYLG	PF	SNADS のその他のロギング対象システム・レベル・オカレンス... [2]
	QSYS	QAZDSYL4	PF	SNADS のその他のロギング対象システム・レベル・オカレンス...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZMFCF	PF	メール・サーバー・フレームワーク (MSF) 構成変更ロギング... [2]
	QSYS	QAZMFCF4	PF	メール・サーバー・フレームワーク (MSF) 構成変更ロギング...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZMFER	PF	メール・サーバー・フレームワーク (MSF) エラー・ロギング... [2]
	QSYS	QAZMFER4	PF	メール・サーバー・フレームワーク (MSF) エラー・ロギング...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZMFLG	PF	メール・サーバー・フレームワーク (MSF) データ・ロギング... [2]
	QSYS	QAZMFLG4	PF	メール・サーバー・フレームワーク (MSF) データ・ロギング...; *TYPE4 の形式で保管されます。[2]
	QSYS	QAZMFSY	PF	メール・サーバー・フレームワーク (MSF) システム情報ロギング... [2]
	QSYS	QAZMFSY4	PF	メール・サーバー・フレームワーク (MSF) システム情報ロギング...; *TYPE4 の形式で保管されます。[2]
	QSYS	QPDSPJRN	PRTF	ジャーナル表示印刷装置ファイル。[1]

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPJRNRCVA	QSYS	QPDSPRCV	PRTF	ジャーナル・レシーバー属性印刷装置ファイル。 [1]
DSPLIB	QSYS	QPDSPLIB	PRTF	ライブラリー印刷装置ファイル。[1]
DSPLIBD	QSYS	QPRTLIBD	PRTF	ライブラリー記述印刷装置ファイル。[1]
DSPLIBL	QSYS	QPRTLIBL	PRTF	ライブラリー・リスト印刷装置ファイル。[1]
DSPLIND	QSYS	QPDCLINE	PRTF	回線記述印刷装置ファイル。[1]
DSPLOG	QSYS	QPDSPLOG	PRTF	ログ表示印刷装置ファイル。[1]
DSPMNUA	QSYS	QPDSPMNU	PRTF	メニュー属性印刷装置ファイル。[1]
DSPMOD	DSPMOD コマンドの場合、次のすべての項目は、ファイルのタイプ (またはグループ) に関する特定のタイプの情報を保管するために使用されるファイルのファイル・タイプを持ちます。すなわち、各モデル・ファイル記述の共通部分は、下記の太字部分で始まります。続く各記述の固有の部分は、ファイルの用途の欄に記載してあります。			

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSVMSS	QACQSRC	PRTF	セキュリティー出口プログラム例のソース... [1]
	QSYS	QABNDMBA	PF	基本情報と互換性のセクション... [1]
	QSYS	QABNDMSI	PF	解凍したサイズおよびサイズの限界... [1]
	QSYS	QABNDMEX	PF	このモジュールで定義され、他のモジュールにエクスポートされる記号... [1]
	QSYS	QABNDMIM	PF	このモジュールの外部の定義済みの記号... [1]
	QSYS	QABNDMPR	PF	プロシージャー名とタイプのリスト... [1]
	QSYS	QABNDMRE	PF	プログラムまたはサービス・プログラムにモジュールをバインドしたときにモジュールによって参照されていたシステム・オブジェクトのリスト... [1]
	QSYS	QABNDMCO	PF	モジュールの著作権情報...
	QSYS	QSYSPRT	PRTF	モジュール印刷装置ファイル...[1]
DSPMODD	QSYS	QPDCMOD	PRTF	モード記述印刷装置ファイル。[1]
DSPMODSTS	QSYS	QPDCMOD	PRTF	モード状況印刷装置ファイル。[1]
DSPMSG	QSYS	QPDSPMSG	PRTF	メッセージ表示印刷装置ファイル。[1]
DSPMSGD	QSYS	QPMSGD	PRTF	メッセージ記述印刷装置ファイル。[1]
DSPNETA	QSYS	QANFDNTF	PF	ネットワーク・ファイル項目のレコード様式を定義するために使用するモデル・データベース・ファイル。 [2]
	QSYS	QPDSPNET	PRTF	ネットワーク属性表示印刷装置ファイル。 [1]
	QSYS	QPNFNJE	PRTF	ネットワーク・ジョブ項目表示印刷装置ファイル。 [1]
DSPNWID	QSYS	QPDCNWID	PRTF	ネットワーク・インターフェース記述印刷装置ファイル。 [1]
DSPOBJAUT	QSYS	QAOBJAUT	PF	オブジェクト権限項目のレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QPOBJAUT	PRTF	オブジェクト権限印刷装置ファイル。[1]
DSPOBJD	QSYS	QADSPOBJ	PF	オブジェクト記述項目のレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QPRTOBJD	PRTF	オブジェクト記述印刷装置ファイル。[1]
DSPOPT	QSYS	QAMODFA	PF	ファイル属性のモデル出力ファイル。[2]
	QSYS	QAMODPA	PF	ディレクトリー属性のモデル出力ファイル。[2]
	QSYS	QAMODVA	PF	ボリューム属性のモデル出力ファイル。[2]
	QSYS	QPSRODSP	PRTF	保管/復元情報用の印刷装置ファイル。 [1]
	QSYS	QSYSPRT	PRTF	光情報の印刷装置ファイル。[1]
DSPOVR	QSYS	QPDSPOVR	PRTF	指定変更表示印刷装置ファイル。[1]
DSPPDGPRF	QGPL	QPCJPDGPRF	PRTF	印刷記述子グループ・プロファイル用の印刷装置ファイル。 [1]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPPFRGPH	QPFRDATA	QAPGGPHF	PF	パフォーマンス・データベース・ファイル: グラフ様式データ。
	QPFRDATA	QAPGPKGF	PF	パフォーマンス・データベース・ファイル: グラフ・パッケージ・データ。
	QPFR	QPPGGPH	PRTF	パフォーマンス・グラフ印刷装置ファイル。[1]
DSPPGM	QSYS	QPDPGM	PRTF	プログラム表示印刷装置ファイル。[1]
DSPPGMADP	QSYS	QADPGMAD	PF	指定のプロファイルを採用するプログラム名の保管用に作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QPPGMADP	PRTF	指定のプロファイルを適用するプログラムをリストする印刷装置ファイル。 [1]
DSPPGMREF	QSYS	QADSPPGM	PF	プログラム参照を保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。[2]
	QSYS	QPDSPPGM	PRTF	プログラム参照を含む印刷装置ファイル。 [1]
DSPPGMVAR	QSYS	QPDBGDSP	PRTF	プログラム変数 (デバッグ・モード) 印刷装置ファイル。 [1]
DSPPRB	下記の QUSRSYS ライブラリー内の 8 つの QASXxxxx ファイルは、すべて、DLTPRB コマンドおよび WRKPRB コマンドによっても使用されます。以下のそれ以外のファイル (QSYS 内) は、これらのコマンドでは使用されません。			

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QASXxxxx	PF	問題出力ファイルのレイアウトを含む 5 つの QASXxxxx モデル・データベース・ファイルのセット。ここで、xxxx は、CAOF、FXOF、PBOF、 SDOF、および TXOF です。 [2]
	QSYS	QSXPRTD	PRTF	問題ログの詳細 印刷装置ファイル。[1]
	QSYS	QSXPRTL	PRTF	問題ログの要約 印刷装置ファイル。[1]
[3] 以下の 8 ファイルは、DLTPRB コマンドおよび WRKPRB コマンドによっても使用されます。				

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QUSRSYS	QASXCALL [3]	PF	問題ログの呼び出し指定変更ファイル。
	QUSRSYS	QASXDTA [3]	PF	問題ログのデータ識別コード・ファイル。
	QUSRSYS	QASXEVT [3]	PF	問題ログの事象ログ・ファイル。
	QUSRSYS	QASXFRU [3]	PF	問題ログの考えられる原因のファイル。
	QUSRSYS	QASXNOTE [3]	PF	問題ログのユーザー注記ファイル。
	QUSRSYS	QASXPROB [3]	PF	問題ログの問題ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QUSRSYS	QASXPTF [3]	PF	問題ログの PTF ファイル。
	QUSRSYS	QASXSYMP [3]	PF	問題ログの徴候ストリング・ファイル。
DSPPTF	QSYS	QADSPPTF	PF	プログラム一時修正 (PTF) 情報を保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。[2]
	QSYS	QSYSPRT	PRTF	プログラム一時修正 (PTF) 表示印刷装置ファイル。[1]
DSPPWRSCD	QSYS	QSYSPRT	PRTF	電源スケジュール表示印刷装置ファイル。[1]
DSPRCDLCK	QSYS	QPDSPRLK	PRTF	レコード・ロック表示印刷装置ファイル。[1]
DSPRCYAP	QSYS	QSYSPRT	PRTF	アクセス・パス印刷装置ファイルの回復表示。
DSPRDBDIRE	QSYS	QSYSPRT	PRTF	分散リレーショナル・データベース・ディレクトリー印刷装置ファイル。[1]
	QSYS	QADSPDE	PF	RDB ディレクトリー項目のレコード様式を定義するモデル・データベース・ファイル。
DSPRJECFG	QRJE	QPRTCFGC	PRTF	RJE 構成印刷装置ファイル。[1]
DSPSAVF	QSYS	QPSRODSP	PRTF	保管ファイルの保管/復元情報用の印刷装置ファイル。[1]
DSPSBSD	QSYS	QPRTSBSD	PRTF	サブシステム記述印刷装置ファイル。[1]
DSPSFWRSC	QSYS	QARZLCOF	PF	IBM ライセンス・プログラムおよび SystemView パッケージ・アプリケーションに関する情報を保管するために作成されるファイルのモデル・データベース・ファイル。[2]
	QSYS	QSYSPRT	PRTF	ソフトウェア資源印刷装置ファイル。[1]
DSPSOCSTS	QSYS	QSYSPRT	PRTF	制御範囲状況印刷装置ファイル。[1]
	QUSRSYS	QAALSOC	PF	制御範囲データベース・ファイル。
DSPSRVPGM	QSYS	QSYSPRT	PRTF	サービス・プログラム印刷装置ファイル。[1]
DSPSYSSTS	QSYS	QPDSPSTS	PRTF	システム状況表示印刷装置ファイル。[1]
DSPSYSVAL	QSYS	QPDSPSVL	PRTF	システム値印刷装置ファイル。[1]
DSPTAP	QSYS	QATADOF	PF	テープ出力用のモデル出力ファイル。[2]
	QSYS	QPTAPDSP	PRTF	テープ出力用の印刷装置ファイル。[1]
	QSYS	QPSRODSP	PRTF	保管/復元形式のテープ用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	入力用のテープ装置ファイル。
DSPTAPCGY	QSYS	QTAPCGY	PRTF	テープ・カテゴリー 1 用の印刷装置ファイル。
	QSYS	QATACOF	PF	テープ・カテゴリー 2 のモデル出力ファイル。
	QUSRSYS	QATACGY	PF	ライブラリー装置データベース・ファイル。
	QUSRSYS	QLTACGY	LF	ライブラリー装置論理データベース・ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
DSPTAPCTG	QSYS	QPTACTG	PRTF	テープ・カートリッジ識別コード 1 用の印刷装置ファイル。
	QSYS	QATAVOF	PF	テープ・カートリッジ識別コード 2 のモデル出力ファイル。
	QUSRSYS	QATAMID	PF	ライブラリー装置データベース・ファイル。
	QUSRSYS	QLTAMID	LF	ライブラリー装置論理データベース・ファイル。
DSPTAPSTS	QSYS	QPTAPSTS	PRTF	テープ・ライブラリー用の印刷装置ファイル。
	QSYS	QATAIOF	PF	テープのモデル出力ファイル。
	QSYS	QSYSTAP	TAPF	入力用のテープ装置ファイル。
DSPTRC	QSYS	QPDBGDSP	PRTF	トレース (デバッグ・モード) 印刷装置ファイル。[1]
DSPTRCDTA	QSYS	QPDBGDSP	PRTF	トレース・データ (デバッグ・モード) 印刷装置ファイル。 [1]
DSPUSRPMN	QSYS	QSYSPRT	PRTF	文書権限表示印刷装置ファイル。 [1]
DSPUSRPRF	QSYS	QADSPUPA	PF	TYPE(*OBJAUT) の場合のユーザー・プロファイルのレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QADSPUPB	PF	TYPE(*BASIC) の場合のユーザー・プロファイルのレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QADSPUPO	PF	TYPE(*OBJOWN) の場合のユーザー・プロファイルのレコード様式を定義するモデル・データベース・ファイル。 [2]
	QSYS	QPUSRPRF	PRTF	ユーザー・プロファイル印刷装置ファイル。[1]
DSPWSUSR	QSYS	QSYSPRT	PRTF	ワークステーション・ユーザー印刷装置ファイル。[1]
DUPTAP	QSYS	QSYSTAP	TAPF	入出力に使用するテープ装置ファイル。
EDTIGCDCT	QSYS	QPDSPDCT	PRTF	DBCS 印刷装置ファイル。
EDTQST	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
EJTEMLOUT	QSYS	QPEMPRTF	PRTF	エミュレーション印刷装置ファイル。
ENDJOBTRC	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QAPTTRCJ	PF	パフォーマンス・データ収集ファイル: ジョブ・トレース・データ。
	QPFR	QPPTTRCD	PRTF	ジョブ・トレース分析詳細 データを含むパフォーマンス印刷装置ファイル。
	QPFR	QPPTTRC1	PRTF	物理ディスク活動のジョブ・トレース分析要約 データを含むパフォーマンス印刷装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QPFR	QPPTTRC2	PRTF	ジョブ・トレース分析入出力要約 データを含むパフォーマンス印刷装置ファイル。
ENDPRTEML	QSYS	QPEMPRTF	PRTF	エミュレーション印刷装置ファイル。
	QPFR	QAPTSAMH	PF	パフォーマンス・データ収集ファイル: 高水準のサンプル・アドレス・モニター (SAM) データ。
	QPFR	QAPTSAMV	PF	パフォーマンス・データ収集ファイル: 低水準のサンプル・アドレス・モニター (SAM) データ。
FMTDTA	QSYS	QSYSPRT	PRTF	データ形式印刷装置ファイル。
	QGPL	QFMTSRC	PF	分類ソース・デフォルト入力ファイル。
FNDSTRPDM	QPDA	QPUOPRTF	PRTF	ユーザーのストリング検索要求の PDM 印刷装置出力ファイル。
HLDDSTQ	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
INZTAP	QSYS	QSYSTAP	TAPF	入出力に使用するテープ装置ファイル。
LODQSTDB	QQALIB	QAQAxxxx00	PF	質問回答が提供するモデル・データベース・ファイル。[2]
	QSYS	QAQA00xxxx	LF	質問回答データベース・モデル・ファイル。[2]
	QSYS	QAQA00xxxx	PF	質問回答データベース・モデル・ファイル。[2]
	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
MRGFORMD	QPDA	QPAPFPRT	PRTF	組み合わせ形式記述印刷装置ファイル。
PRTACTRPT	ユーザー・ライブラリー	QAITMON	PF	パフォーマンス・データ収集ファイル: ジョブおよびライセンス内部コードのタスク・データ。
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPITACTR	PRTF	ジョブおよびライセンス内部コードのタスク・データを含むパフォーマンス印刷装置ファイル。
PRTAFPDTA	QSYS	QSYSPRT	PRTF	Advanced Function Printing (AFP) データ印刷装置ファイル。
PRTCADMRE	QHASM	QAHAPMRSC	PF	モニター・リソース項目のレコード様式を含むモデル・データベース・ファイル。[2]
	QHASM	QAHAPMATTR	PF	モニター・リソース属性項目のレコード様式を含むモデル・データベース・ファイル。[2]
	QHASM	QPHAPRTMRE	PRTF	クラスター管理ドメイン内のモニター対象リソースの印刷装置ファイル。
PRTCMDUSG	QSYS	QSYSPRT	PRTF	コマンド使用法印刷装置ファイル。
PRTCMNTRC	QSYS	QASCCMNT	PF	通信トレース・レコードを保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QPCSMPRT	PRTF	通信トレース印刷装置ファイル (並行保守モニター)。 [1]
PRTCPTRPT	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTCPTR	PRTF	構成要素レベルの活動データを含むパフォーマンス印刷装置ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
PRTDEVADR	QSYS	QPDDEVA	PRTF	装置アドレス印刷印刷装置ファイル。
PRTDOC	QSYS	QAOPOUFL	PF	文書出力データベース・ファイル。[1]
	QSYS	QSYSPRT	PRTF	文書印刷印刷装置ファイル。[1]
PRTDSKINF	QSYS	QAEZDISK	PF	ディスク・スペース情報のモデル出力ファイル。
	QUSRSYS	QAEZDISK	PF	ディスク・スペース情報のデータベース入力ファイル。
	QSYS	QPEZDISK	PRTF	ディスク・スペース報告印刷装置ファイル。
	QSYS	QSYSPRT	PRTF	ディスク・スペース報告印刷装置ファイル。OVRPRTF を使用する場合は、このファイルを指定しなければなりません。
PRTERRLOG	QSYS	QAPRTELG	PF	エラー・ログ・レコードを保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。[2]
	QSYS	QAVOLSTA	PF	ボリューム統計を保管するために作成されるファイルのレコード様式を定義するモデル・データベース・ファイル。[2]
	QSYS	QPCSMPRT	PRTF	エラー・ログ (並行保守モニター用) 印刷装置ファイル。 [1]
PRTINTDTA	QSYS	QPCSMPRT	PRTF	内部データ (並行保守モニター用) 印刷装置ファイル。
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
PRTJOBRPT	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTITVJ	PRTF	ジョブ間隔収集データを含むパフォーマンス印刷装置ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
PRTJOBTRC	QPFR	QAJOBTRC	PF	パフォーマンス・データ収集ファイル: ジョブ・トレース・データ。
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QAPTTRCJ	PF	パフォーマンス・データ収集ファイル: ジョブ・トレース・データ。
	QPFR	QPPTTRCD	PRTF	ジョブ・トレース分析詳細 データを含むパフォーマンス印刷装置ファイル。
	QPFR	QPPTTRC1	PRTF	物理ディスク活動のジョブ・トレース分析要約 データを含むパフォーマンス印刷装置ファイル。
	QPFR	QPPTTRC2	PRTF	ジョブ・トレース分析入出力要約 データを含むパフォーマンス印刷装置ファイル。
PRTLCKRPT	ユーザー・ライブラリー	QAPTLCKD	PF	ロックと競合捕そくデータを含むパフォーマンス・データ収集出力 ファイル
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTLCK	PRTF	ロックと競合捕そくデータを含むパフォーマンス・データ収集印刷装置 ファイル
	QSYS	QAPMDMPT	PF	システムのロックと競合捕そくのトレース・データを含むパフォーマンス・データ収集入力 ファイル。 [2]
PRTOPCACT	QSOC	QAOPCACT	PF	OptiConnect 活動のモデル出力ファイル。[2]
	QSYS	QSYSPRT	PRTF	OptiConnect 活動の印刷装置ファイル。[1]
PRTOPCJOB	QSOC	QAOPCJOB PF	PF	OptiConnect ジョブのモデル出力ファイル。[2]
	QSYS	QSYSPRT	PRTF	OptiConnect ジョブの印刷装置ファイル。[1]
PRTPOLRPT	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTITVP	PRTF	サブシステムおよびプール活動の間隔データを含むパフォーマンス印刷装置ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
PRTRSCRPT	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTITVR	PRTF	ディスクおよび通信回線の活動間隔データを含むパフォーマンス印刷装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
	QPFR	QAPTSAMH	PF	パフォーマンス・データ収集ファイル: 高水準のサンプル・アドレス・モニター (SAM) データ。
	QPFR	QAPTSAMV	PF	パフォーマンス・データ収集ファイル: 低水準のサンプル・アドレス・モニター (SAM) データ。
	QPFR	QPPTSAM	PRTF	SAM パフォーマンス・データの印刷装置ファイル。
PRTSYSRPT	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPPTSYSR	PRTF	システムの作業負荷および資源の使用状況データを含むパフォーマンス印刷装置ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
PRTTNSRPT	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QPFR	QPSPDJS	PRTF	ジョブ要約 データを含むパフォーマンス印刷装置ファイル。
	QPFR	QPSPDTD	PRTF	ジョブ状態遷移 データを含むパフォーマンス印刷装置ファイル。
	QPFR	QPSPDTS	PRTF	ジョブ・トランザクション・データを含むパフォーマンス印刷装置ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
PRTTRC	QSYS	QPSRVTRC	PRTF	ジョブ・トレース印刷装置出力ファイル。[1]
PRTTRCRPT	ユーザー・ライブラリー	QTRTJOBT	PF	バッチ・ジョブ・トレース・データを含むパフォーマンス・データ収集入力 ファイル。
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
	QSYS	QAPMDMPT	PF	システム・トレース・データを含むパフォーマンス・データ収集入力ファイル。 [2]
QRYDOCLIB	QSYS	QAOSIQDL	PF	文書ライブラリー QUERY 出力ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
QRYDST	QSYS	QAOSILIN	PF	着信配布出力ファイル。
	QSYS	QAOSILOT	PF	発信配布出力ファイル。
RCLSTG	QSYS	QPRCLDMP	PRTF	ダンプ出力リスト再利用。
RCVDST	QSYS	QAOSIRCV	PF	着信メール配布受信モデル・データベース・ファイル。 [2]
RCVTIEF	QSYS	QPTIRCV	PRTF	受信ファイル要約印刷装置ファイル。[1]
RLSDSTQ	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
RMVDSTQ	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
	QUSRSYS	QASNADSR	PF	SNADS 経路指定テーブル。
RMVDSTRTE	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
	QUSRSYS	QASNADSR	PF	SNADS 経路指定テーブル。
RMVDSTSYSN	QUSRSYS	QASNADSA	PF	SNADS 2 次ノード ID テーブル。
RMVJRNCHG	QSYS	QAJRNCHG	PF	ジャーナル処理済み変更除去のモデル出力ファイル。
RMVNETJOBE	QUSRSYS	QANFNJE	PF	ネットワーク・ジョブ項目データベース・ファイル。
RMVSOCE	QUSRSYS	QAALSOC	PF	制御範囲ファイル。
RMVTAPCTG	QUSRSYS	QATAMID	PF	カートリッジ ID データベース・ファイル。
	QUSRSYS	QLTAMID	LF	カートリッジ ID 論理データベース・ファイル。
	QUSRSYS	QATACGY	PF	カテゴリー・データベース・ファイル。
	QUSRSYS	QLTACGY	LF	カテゴリー論理ファイル。
RNMTCPHTE	QUSRSYS	QATOCHOST	PF	TCP/IP ホスト・ファイル。
RST	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RSTCFG	QSYS	QASRRSTO	PF	構成のモデル出力ファイル。[2]
	QSYS	QPSRLDSP	PRTF	復元オブジェクト状況印刷装置ファイル。 [1]
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RSTDFROBJ	QSYS	QASRRSTO	PF	復元されるオブジェクトのモデル出力ファイル。 [2]
	QSYS	QPSRLDSP	PRTF	復元オブジェクト状況印刷装置ファイル。 [1]
RSTDLO	QSYS	QAOJRSTO	PF	復元される文書ライブラリー・オブジェクトのモデル出力ファイル。 [2]
	QSYS	QPRSTDLO	PRTF	復元される文書およびフォルダー用の印刷装置ファイル。 [1]
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RSTLIB	QSYS	QASRRSTO	PF	ライブラリーのモデル出力ファイル。[2]
	QSYS	QPSRLDSP	PRTF	復元オブジェクト状況印刷装置ファイル。 [1]
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RSTLICPGM	QSYS	QPSRLDSP	PRTF	復元オブジェクト状況印刷装置ファイル。 [1]
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
RSTOBJ	QSYS	QASRRSTO	PF	復元されるオブジェクトのモデル出力ファイル。 [2]
	QSYS	QPSRLDSP	PRTF	復元オブジェクト状況印刷装置ファイル。 [1]
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RSTPFRCOL	QPFR	QAPGSUMD	PF	グラフィック・データのパフォーマンス・データ収集ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYDWxxxx	PF	ディスク・ウォッチャーのパフォーマンス・データ・モデル・ファイル。モデル・ファイルの名前と説明については、『Disk Watcher データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYJWxxxx	PF	ジョブ・ウォッチャーのパフォーマンス・データ・モデル・ファイル。 [2]
	QSYS	QAYPExxxx	PF	Performance Explorer (PEX) のデータ・モデル・ファイル。[2]
	QPFR	QAPTLCKD	PF	パフォーマンス・データ収集ファイル: ロックと競合捕そくデータ。
RSTSYSINF	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RSTUSRPRF	QSYS	QASRRSTO	PF	ユーザー・プロファイルのモデル出力ファイル。[2]
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
RTVDOC	QSYS	QAOSIRTV	PF	文書ライブラリー出力ファイルからの文書の検索。
RTVDSKINF	QSYS	QAEZDISK	PF	ディスク情報のモデル・ファイル。
	QUSRSYS	QAEZDISK	PF	ディスク情報用のデータベース出力ファイル。
RUNQRY	QSYS	QPQUPRFIL	PRTF	QUERY 出力に使用する印刷装置ファイル。[1]
SAV	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVCFG	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVCHGOBJ	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
SAVDLO	QSYS	QAOJSAVO	PF	保管される文書およびフォルダーのモデル出力ファイル。 [2]
	QSYS	QPSAVDLO	PRTF	保管される文書およびフォルダー用の印刷装置ファイル。 [1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVLIB	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QUSRSYS	QSRPNTWK	PRTF	保管されるデータベース・ファイル・ネットワーク用の印刷装置ファイル。
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVOBJ	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVPFRCOL	QPFR	QAPGSUMD	PF	グラフィック・データのパフォーマンス・データ収集ファイル。
	QSYS	QAPMxxxx	PF	QAPMxxxx のパフォーマンス・データ収集モデル・ファイル。モデル・ファイルの名前と説明については、『収集サービス・データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYDWxxxx	PF	ディスク・ウォッチャーのパフォーマンス・データ・モデル・ファイル。モデル・ファイルの名前と説明については、『Disk Watcher データ・ファイル』のトピックを参照してください。 [2]
	QSYS	QAPYJWxxxx	PF	ジョブ・ウォッチャーのパフォーマンス・データ・モデル・ファイル。 [2]
	QSYS	QAYPExxxx	PF	Performance Explorer (PEX) のデータ・モデル・ファイル。[2]
	QPFR	QAPTLCKD	PF	パフォーマンス・データ収集ファイル: ロックと競合捕そくデータ。
SAVSAVFDTA	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVSECDTA	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVSYS	QSYS	QASAVOBJ	PF	保管されるオブジェクトのモデル出力ファイル。[2]
	QSYS	QPSAVOBJ	PRTF	保管されるオブジェクト用の印刷装置ファイル。[1]
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SAVSYSINF	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
SBMFNCJOB	QSYS	QDFNDATA	DSPF	非 ICF 金融機関表示装置ファイル。
	QUSRSYS	QFNDEVTBL	PF	装置テーブルに関するデータを含むファイル。
	QUSRSYS	QFNPGMTBL	PF	プログラム・テーブルに関するデータを含むファイル。
	QUSRSYS	QFNUSRTBL	PF	ユーザー・テーブルに関するデータを含むファイル。
SETTAPCGY	QUSRSYS	QATACGY	PF	カテゴリー・データベース・ファイル。
	QUSRSYS	QLTACGY	LF	カテゴリー論理ファイル。
	QSYS	QSYSTAP	TAPF	入力に使用するテープ装置ファイル。
SNDDSTQ	QUSRSYS	QASNADSQ	PF	SNADS 配布待ち行列テーブル。
SNDFNCIMG	QSYS	QCRFDWNL	ICFF	4701 制御装置との通信に使用する ICF ファイル。
SNDPTFORD	QGPL	Qnnnnnnn	SAVF	PTF 保管ファイル。ここで、nnnnnnn は、PTF 番号です。
	QSYS	QESPRTF	PRTF	PTF カバー・レター用の印刷装置ファイル。
SNDSRVRQS	QGPL	Qnnnnnnn	SAVF	PTF 保管ファイル。ここで、nnnnnnn は、PTF 番号です。
	QSYS	QESPRTF	PRTF	PTF カバー・レター用の印刷装置ファイル。
	QUSRSYS	QAEDCDBPF	PF	保守連絡先情報を含むファイル。
STRCODE	ユーザー・ライブラリー	EVFCICFF	ICFF	ワークステーションとの通信に使用する ICF ファイル。
STRCPYSCN	QSYS	QASCCPY	PF	画面コピー出力ファイルのパターン。
STRDFU	QSYS	QDFUPRT	PRTF	DFU 印刷装置ファイル。
	QSYS	QPDZDTALOG	PRTF	DFU 実行時監査ログ。
	QSYS	QPDZDTAPRT	PRTF	DFU 実行時印刷装置データ・ファイル。
	QUSRSYS	QABBxxxxx	PF	注 E を参照。
STRPDM	QPDA	QPUOPRTF	PRTF	PDM の表示対象リスト用の印刷装置ファイル。
STRPRTEML	QSYS	QPEMPRTF	PRTF	エミュレーション印刷装置ファイル。
STRPRTWTR	QSYS	QPSPLPRT	PRTF	すべての印刷装置の書き込みに使用する印刷装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
STRQMQRY	QSYS	QPQXPRTF	PRTF	QUERY CPI によって使用される印刷装置ファイル。[1]
STRQST	QQALIB	QAQAxxxx00	PF	質問回答が提供するモデル・データベース・ファイル。[2]
	QSYS	QAQA00xxxx	LF	質問回答データベース・モデル・ファイル。[2]
	QSYS	QAQA00xxxx	PF	質問回答データベース・モデル・ファイル。[2]
	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
	QUSRSYS	QABBxxxxx	PF	注 F を参照。
STRSDA	QGPL	QDDSSRC	PF	DDS ソース・デフォルト入力ファイル。
STRSEU	QGPL	QTXTSRC	PF	SEU ソース・デフォルト入力ファイル。
	QPDA	QPSUPRTF	PRTF	SEU ソース・メンバー印刷装置ファイル。
STRSST	QSYS	QPCSMPRT	PRTF	保守ツール印刷装置ファイル。[1]
	QTY	QATYALMF	PF	警報レコード様式の電話データベース・モデル・ファイル。
	QUSRSYS	QATYSWTE	PF	ユーザーが作成したスイッチ項目の電話データベース・ファイル。
	QTY	QATYCDRF	PF	警報レコード様式の電話データベース・モデル・ファイル。
	QUSRSYS	QATYSWTE	PF	ユーザーが作成したスイッチ項目の電話データベース・ファイル。
	QUSRSYS	QABBADMTB	PF	... 管理テーブル。
	QUSRSYS	QABBCAN	PF	... 候補ファイル。
	QUSRSYS	QABBCOX	PF	...コンテキストの索引。
	QUSRSYS	QABBDEX	PF	...外部文書索引識別コード。
	QUSRSYS	QABBDIC	PF	...ディクショナリー。
	QUSRSYS	QABBDIX	PF	...内部文書見出し識別コード。
	QUSRSYS	QABBDOX	PF	...文書索引テーブル。
	QUSRSYS	QABBFIX	PF	...部分構文索引。
	QUSRSYS	QABBIQTB	PF	...スケジューリング待ち行列。
	QUSRSYS	QABBLADN [4]	PF	...索引付き LADN (ライブラリー割り当て文書名) のリスト。
TRCCNN	QSYS	QSYSPRT	PRTF	接続トレース印刷装置ファイル。
[4] この QABBLADN ファイルは、STRRGZIDX コマンドでも STRIDXMON コマンドでも使用されません。				

表 22. CL コマンドで使用されるファイル (パート 1)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
TRCCPIC	QSYS	QACM0TRC	PF	CPI 通信データベース・トレース・モデル出力ファイル。 [2]
	QSYS	QSYSPRT	PRTF	CPI 通信トレース印刷装置ファイル。 [1]
TRCICF	QSYS	QAIFTRCF	PF	ICF データベース・トレース出力ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
	QSYS	QPIFTRCF	PRTF	ICF トレース印刷装置ファイル。[1]
TRCINT	QSYS	QPCSMPRT	PRTF	内部トレース (並行保守モニター用) 印刷装置ファイル。
	QSYS	QSYSTAP	TAPF	出力に使用するテープ装置ファイル。
TRCJOB	QSYS	QATRCJOB	PF	トレース・レコードを保管するために作成されたファイルのレコード様式を定義するデータベース・ファイル。 [2]
	QSYS	QPSRVTRC	PRTF	ジョブ・トレース印刷装置出力ファイル。[1]
UPDDTA	QSYS	QPDZDTALOG	PRTF	DFU 実行時監査ログ。
	QSYS	QPDZDTAPRT	PRTF	DFU 実行時印刷装置データ・ファイル。
VFYLNKLPDA	QSYS	QSYSPRT	PRTF	LPDA をサポートしているリンク -2 印刷装置ファイルの検査。[1]
WRKACTJOB	QSYS	QPDSPAJB	PRTF	活動ジョブ表示印刷装置ファイル。[1]
WRKALR	QUSRSYS	QAALERT	PF	警報データベース・ファイル。
	QSYS	QSYSPRT	PRTF	警報印刷装置ファイル。[1]
WRKCFGSTS	QSYS	QSYSPRT	PRTF	構成状況印刷装置ファイル。[1]
WRKCMTDFN	QSYS	QPDSPJOB	PRTF	コミットメント定義リスト印刷装置ファイル。[1] スプール・ファイルの名前は QPCMTCTL です。
WRKCNTINF	QUSRSYS	QAEDCDBPF	PF	接続データを含むデータベース・ファイル。
WRKDDMF	QSYS	QPWRKDDM	PRTF	分散データ管理 (DDM) ファイル属性印刷装置ファイル。 [1]
WRKDEVTBL	QUSRSYS	QFNDEVTBL	PF	装置テーブルに関するデータを含むファイル。
WRKDIR	QSYS	QPDSPDDL	PRTF	ディレクトリー項目の詳細 印刷装置ファイル。
	QSYS	QPDSPDSM	PRTF	ディレクトリー項目の要約 印刷装置ファイル。
WRKDOCCVN	QUSRSYS	QAO1CRL	LF	入出力用の文書変換論理 ファイル。
	QUSRSYS	QAO1CVNP	PF	入出力用の文書変換物理 ファイル。
	QUSRSYS	QAO1DCVN	PRTF	文書変換印刷装置ファイル。
WRKDPCQ	QSYS	QPDXWRKD	PRTF	DSNX/PC の待機配布要求用の印刷装置ファイル。 [1]
WRKDSKSTS	QSYS	QPWCDSKS	PRTF	ディスク状況印刷装置ファイル。[1]
WRKDSTL	QSYS	QPDSPLDL	PRTF	配布リストの詳細 印刷装置ファイル。
	QSYS	QPDSPLSM	PRTF	配布リストの要約 印刷装置ファイル。
WRKDSTQ	QSYS	QPDSTSTS	PRTF	配布状況印刷装置ファイル。[1]
	QUSRSYS	QASNADSQ	PF	SNADS 宛先待ち行列テーブル。
WRKFCT	QRJE	QPDSPFCT	PRTF	用紙制御テーブル印刷装置ファイル。[1]
WRKHDWRSC	QSYS	QASUPTEL	PF	ハードウェア資源ロック・データベース・ファイル。

表 22. *CL* コマンドで使用されるファイル *(*パート *1) (*続き*)*

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
WRKHTTPCFG	QUSRSYS	QATMHTTPC	PF	TCP/IP HTTP ファイル。
WRKJOB	QSYS	QPDSPJOB	PRTF	ジョブ表示印刷装置ファイル。[1]
WRKJOBQ	QSYS	QPRTSPLQ	PRTF	ジョブ待ち行列印刷装置ファイル (スプール待ち行列)。 [1]
WRKJOBSCDE	QSYS	QSYSPRT	PRTF	ジョブ・スケジュール項目印刷装置ファイル。[1]
WRKJRNA	QSYS	QPDSPJNA	PRTF	ジャーナル属性印刷装置ファイル。[1]
	QSYS	QAWRKJRNA	PF	ジャーナル属性モデル出力ファイル。[2]
WRKLIBPDM	QPDA	QPUOPRTF	PRTF	PDM の表示対象リスト用の印刷装置ファイル。[1]
WRKMBRPDM	QPDA	QPUOPRTF	PRTF	PDM の表示対象リスト用の印刷装置ファイル。[1]
WRKMSG	QSYS	QPDSPMSG	PRTF	メッセージ待ち行列メッセージ用の印刷装置ファイル。 [1]
WRKMSGD	QSYS	QPMSGD	PRTF	メッセージ記述印刷装置ファイル。
WRKNAMSMTP	QSYS	QATMSMTP	PF	TCP/IP SMTP パーソナル別名テーブル。
	QSYS	QATMSMTPA	PF	TCP/IP SMTP システム別名テーブル。
WRKNETF	QSYS	QANFDNTF	PF	ネットワーク・ファイル表示用のデータベース・ファイル。 [1]
	QSYS	QPNFDNTF	PRTF	ネットワーク・ファイル表示用の印刷装置ファイル。 [1]
WRKNETJOBE	QUSRSYS	QANFNJE	PF	ネットワーク・ジョブ項目のデータベース・ファイル。 [1]
	QSYS	QPNFNJE	PRTF	ネットワーク・ジョブ項目の印刷装置ファイル。 [1]
WRKOBJLCK	QSYS	QPDSPOLK	PRTF	オブジェクト・ロック表示印刷装置ファイル。[1]
WRKOBJPDM	QPDA	QPUOPRTF	PRTF	PDM の表示対象リスト用の印刷装置ファイル。[1]
WRKOUTQ	QSYS	QPRTSPLQ	PRTF	出力スプール待ち行列印刷装置ファイル。[1]
WRKOUTQD	QSYS	QPDSPSQD	PRTF	出力待ち行列記述。[1]
WRKPGMTBL	QUSRSYS	QFNPGMTBL	PF	プログラム・テーブルに関するデータを含むファイル。
WRKPRB	WRKPRB コマンドの QUSRSYS ライブラリーに示されている 8 つの QASXxxxx というファイルは、すべて、DSPPRB コマンドの QUSRSYS ライブラリーに示されているファイルのサブセットと同じです。			
	QSYS	QSXPRTD	PRTF	問題ログの詳細 印刷装置ファイル。
	QSYS	QSXPRTL	PRTF	問題ログの要約 印刷装置ファイル。
	QUSRSYS	QASXxxxx	PF	QUSRSYS ライブラリー内にあるこれら 8 つのファイルについては、 DSPPRB コマンドを参照してください。
WRKQRY	QSYS	QPQUPRFIL	PRTF	QUERY 出力に使用する印刷装置ファイル。

表 22. CL コマンドで使用されるファイル (パート 1) (続き)

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
WRKQST	QSYS	QPQAPRT	PRTF	質問回答印刷装置ファイル。
WRKRDBDIRE	QSYS	QSYSPRT	PRTF	分散リレーショナル・データベース・ディレクトリー印刷装置ファイル。 [1]
WRKRDR	QSYS	QPRTRDWT	PRTF	読み取りプログラム表示印刷装置ファイル。[1]
WRKRJESSN	QRJE	QPRJESTS	PRTF	RJE セッションの活動状況印刷装置ファイル。 [1]
WRKRPYLE	QSYS	QPRTRPYL	PRTF	システム応答リスト印刷装置ファイル。[1]
WRKSBMJOB	QSYS	QPDSPSBJ	PRTF	投入済みジョブ印刷装置ファイル。[1]
WRKSBS	QSYS	QPDSPSBS	PRTF	サブシステム表示印刷装置ファイル。[1]
WRKSBSJOB	QSYS	QPDSPSBJ	PRTF	サブシステム・ジョブ表示印刷装置ファイル。[1]
WRKSHRPOOL	QSYS	QSYSPRT	PRTF	共用記憶域プール印刷装置ファイル。[1]
WRKSOC	QUSRSYS	QAALSOC	PF	制御範囲データベース・ファイル。
WRKSPLF	QSYS	QPRTSPLF	PRTF	スプール・ファイル印刷装置ファイル。[1]
WRKSPLFA	QSYS	QPDSPSFA	PRTF	スプール・ファイル属性印刷装置ファイル。[1]
WRKSPTPRD	QSYS	QSYSPRT	PRTF	サポート・プロダクト印刷装置ファイル。[1]
WRKSRVPVD	QUSRSYS	QAEDSPI	PF	サービス提供者情報ファイル。
WRKSRVRQS	QUSRSYS	QANSSRI	PF	サービス要求装置ファイル。
WRKSSND	QRJE	QPRTSSND	PRTF	セッション記述印刷装置ファイル。[1]
WRKSYSACT	ユーザー・ライブラリー	QAITMON	PF	パフォーマンス・データ収集ファイル: ジョブおよびライセンス内部コードのタスク・データ。[1]
	QPFR	QAPTDDS	PF	パフォーマンス・データ DDS ソース・ファイル。
WRKSYSSTS	QSYS	QPDSPSTS	PRTF	システム状況印刷装置ファイル。[1]
WRKSYSVAL	QSYS	QSYSPRT	PRTF	システム値印刷装置ファイル。[1]
WRKTAPCTG	QUSRSYS	QATAMID	PF	カートリッジ ID データベース・ファイル。
	QUSRSYS	QLTAMID	LF	カートリッジ ID 論理データベース・ファイル。
	QUSRSYS	QATACGY	PF	カテゴリー・データベース・ファイル。
	QUSRSYS	QLTACGY	LF	カテゴリー論理ファイル。
	QSYS	QSYSTAP	TAPF	入力用のテープ装置ファイル。
WRKTCPPTP	QUSRSYS	QATOCPTP	PF	TCP/IP Point-to-Point プロファイル構成。
	QUSRSYS	QATOCMODEM	PF	TCP/IP Point-to-Point モデム構成。
WRKTIE	QSYS	QPTIRCV	PRTF	受信されたファイルの印刷装置ファイル要約。
WRKTIMZON	QSYS	QSYSPRT	PRTF	選択されたタイム・ゾーン記述の情報。
WRKTRA	QSYS	QSYSPRT	PRTF	トークンリング・ネットワーク・アダプターのリストを含む印刷装置ファイル。 [1]
WRKUSRJOB	QSYS	QPDSPSBJ	PRTF	ユーザー・ジョブ表示印刷装置ファイル。[1]

表 22. *CL* コマンドで使用されるファイル *(*パート *1) (*続き*)*

コマンド名	ファイル・ライブラリー	ファイル名	ファイル・タイプ	ファイルの用途
WRKUSRTBL	QUSRSYS	QFNUSRTBL	PF	ユーザー・テーブルに関するデータを含むファイル。
WRKWTR	QSYS	QPRTRDWT	PRTF	書き出しプログラム表示印刷装置ファイル。[1]

関連情報:

問題表示 (DSPPRB) コマンド

CL プログラミング

CL を使用してプログラムするには、CL プログラミングに固有なプロシージャーおよび概念を理解する必要があります。

CL ソース・プログラム とは、CL ソース・ステートメントのセットであり、オリジナル・プログラム・モデル (OPM) プログラム、または統合言語環境 (ILE) モジュールのいずれかにコンパイルすることができます。

CL プログラム または *CL* プロシージャーとは CL コマンドのグループのことで、これにより入力の入手先、その処理方法、およびその結果の出力先をシステムに指示します。プログラムまたはプロシージャーには名前が割り当てられ、その名前によって他のプロシージャーが呼び出しを行ったり、またはプログラムにバインドして実行したりすることができます。他の言語のプロシージャーと同様、CL ソース・ステートメントを入力、コンパイル、およびバインドすると、プログラムまたはプロシージャーを実行することができます。

CL コマンドを個別に (例えばコマンド入力画面から、または入力ストリーム中の個々のコマンドとして) 入力すると、各コマンドはそれぞれ個別に処理されます。 CL コマンドを CL プログラムまたはプロシージャーのソース・ステートメントとして入力した場合は、ソースはそのままの形で保管され、必要に応じて修正することができます。ILE コンパイラーを使用する場合は、コマンドは 1 つのモジュールとしてコンパイルされます。このモジュールは、他のプログラムにバインドして実行できる永続システム・オブジェクトとして保持されます。したがって CL は、実質的にはシステム機能用の高水準プログラミング言語と見なすことができます。 CL プログラムまたは CL プロシージャーを使用すると、コマンドのグループを一貫して確実に処理することができます。コマンドを個別に入力した場合には実行できない機能を CL プログラムまたはプロシージャーで実行することができるため、CL プログラムまたはプロシージャーは、いくつかのコマンドをそれぞれ単独で実行する場合よりも、実行時のパフォーマンスが向上します。

| CL プログラムまたは CL プロシージャーは、バッチまたは対話式で使用することができます。ただし、
| コマンドや機能によっては、バッチ・ジョブか対話式ジョブのどちらかでしか使用できないものがあります。 CL ソース・ステートメントは CL コマンドによって構成されます。すべての CL コマンドを CL
| ソース・ステートメントとして使用できるわけではありません。また、CL コマンドによっては、CL プロ
| シージャーまたはオリジナル・プログラム・モデル (OPM) プログラムの中でしか使用できないものもあ
| ります。 CL ソース・ステートメントは、ワークステーションから対話式で、または装置からのバッチ・
| ジョブ入力ストリームによって、データベース・ソース・メンバーまたは IFS ストリーム・ファイルに入
| 力することができます。 CL ソース・ステートメントを使用してプログラムを作成するには、ソース・ス
| テートメントをデータベース・ソース・メンバーまたは IFS ストリーム・ファイルに入力しなければなり
| ません。その後、ソース・メンバーをコンパイルしてモジュールにし、そのモジュールをプログラム・オブ
| ジェクトにバインドすることにより、統合言語環境 (ILE) プログラムを作成することができます。

CL プログラムまたは CL プロシージャーは、例えば次のようなさまざまな用途に使用することができます。

- あるプログラムやプロシージャー内での処理の順序、および他のプログラムやプロシージャーの呼び出し順序を制御する。
- メニューを表示し、そのメニューから選択されたオプションに基づいてコマンドを実行する。これにより、ワークステーションのユーザーの作業が簡素化されるだけでなく、エラーを最小限に抑えることができます。
- データベース・ファイルの読み取り。
- 特定のメッセージを監視することによって、コマンド、プログラムまたはプロシージャーによって示されたエラー状態に対処する。
- アプリケーションで使用される変数 (日付、時刻、外部標識など) を設定することにより、アプリケーションの機能を制御する。
- サブシステムの開始やファイルの保管など、あらかじめ定められているシステム・オペレーターによる手順を実行する。これにより、オペレーターが定期的に使用するコマンドの数が減少するとともに、一貫したシステム操作が確実に行われるようにすることができます。

アプリケーションに CL プログラムおよびプロシージャーを使用する際の利点は、次のとおりです。

- プログラムまたはプロシージャーが作成された時点でコマンドが実行可能な形で保管されるので、プログラムおよびプロシージャーを使用すると、コマンドを個別に入力して実行するよりも処理が速くなります。
- CL プログラムおよびプロシージャーには柔軟性があります。 CL プログラムおよびプロシージャーには、プログラムまたはプロシージャーによって実行される操作を特定の使用条件に合わせるためのパラメーターを渡すことができます。
- CL プログラムまたは CL プロシージャーは、他の高水準プログラムおよびプロシージャーと同様にテストおよびデバッグすることができます。
- CL プログラムおよびプロシージャーには、コマンドを個別に入力した場合には使用できない、条件ロジックや特殊機能を組み込むことができます。
- CL プロシージャーは、他の言語のプロシージャーとともにバインドすることができます。

CL プログラムおよびプロシージャーを次の目的のために使用することはできません。

- データベース・ファイルに対するレコードの追加や更新。
- 印刷装置ファイルまたは ICF ファイルの使用。
- ディスプレイ装置ファイル内のサブファイルの使用。
- プログラム記述表示装置ファイルの使用。

関連タスク:

341 ページの『混合リストの CL コマンド・パラメーターに対する CL またはその他の HLL の使用』
CL コマンドを実行する場合、混合リストの要素はこの形式でコマンド処理プログラムに渡されます。

CL プログラムまたは CL プロシージャー作成のプロセス

すべてのプログラムは、ソース作成、モジュール作成、およびプログラム作成のステップで作成されます。

1. ソースの作成。CL ソース・ステートメントは CL コマンドによって構成されます。ソース・ステートメントは、ユーザーのアプリケーションの設計に基づく論理的な順序に従ってデータベース・ファイルまたは IFS ストリーム・ファイルに入力されます。

2. モジュールの作成。**制御言語モジュール作成 (CRTCLMOD)** コマンドにより、このソースを使用してシステム・オブジェクトを作成します。作成された CL モジュールは、プログラムにバインドすることができます。 CL モジュールには 1 つの CL プロシージャーが含まれます。他の高水準言語 (HLL) では、各モジュールに対して複数のプロシージャーが含まれることがあります。
3. プログラムの作成。**プログラム作成 (CRTPGM)** コマンドにより、このモジュール (他のモジュールおよびサービス・プログラムとともに) を使用して、プログラムを作成します。

注**:** ただ 1 つの CL モジュールからなるプログラムを作成したい場合は、ステップ 2 と 3 を結合したバインド **制御言語プログラム作成 (CRTBNDCL)** コマンドを使用することができます。 CL ステートメントからオリジナル・プログラム・モデル (OPM) CL プログラムを作成したい場合は、 **CL プログラム作成 (CRTCLPGM)** コマンドを使用することができます。

関連情報:

プログラム作成 (CRTPGM) コマンド

制御言語プログラム作成 (CRTCLPGM) コマンド

バインド制御言語プログラム作成 (CRTCLPGM) コマンド

対話式入力

対話式入力を使用すると、コマンドを個別に入力することができます。

IBM i オペレーティング・システムは、プログラマー・メニュー、コマンド入力画面、コマンド・プロンプト画面、およびプログラム開発管理機能 (PDM) メニューなど、対話式入力を可能にするさまざまなメニューと画面を提供しています。「機密保護解説書」で説明されている IBM i のセキュリティー機能を使用する場合、ユーザーがこれらの画面を使用できるかどうかは、そのユーザー・プロファイルで指定されている権限によって決まります。ユーザー・プロファイルは通常、システムの機密保護担当者により作成され管理されます。

頻繁に使用されるソース入力方式は、原始ステートメント入力ユーティリティー (SEU) です。このユーティリティーは、WebSphere Development Studio の一部です。**ファイルの編集 (EDTF)** コマンドを使用して、データベース・ソース・ファイルで CL コマンドを入力または変更することもできます。ただし、EDTF は、SEU に組み込まれている統合 CL コマンド・プロンプト・サポートを備えていません。

関連情報:

ファイル編集 (EDTF) コマンド

バッチ入力

1 つのバッチ入力ストリームによって、CL ソースの作成、CL モジュールの作成、およびプログラムの作成を行うことができます。

次の例は、入力ストリームの基本的な部分を示しています。この入力ストリームは、 **データベース・ジョブ投入 (SBMDBJOB)** コマンドを使用してジョブ待ち行列に入れられます。入力ストリームは次のような形式になっていなければなりません。

```
// BCHJOB
CRTBNDCL PGM(QGPL/EDUPGM) SRCFILE(PERLIST)
// DATA FILE(PERLIST) FILETYPE(*SRC)
   .
   .              (CL プロシージャーのソース)
   .
//
/*
// ENDINP
```

このストリームにより、インライン・ソースをもとにしてプログラムが作成されます。ファイルにソース・コードを保持しておきたい場合には、 **ファイル・コピー (CPYF)** コマンドを使用して、ソースをデータベース・ファイル (ソース・ファイル) にコピーすることができます。そして、そのデータベース・ファイルを使用してプログラムを作成することもできます。

また、IBM 提供の装置ファイルを使用して、テープなどの外部媒体に入っている CL ソースから直接 CL モジュールを作成することもできます。 IBM 提供のテープ・ソース・ファイルは QTAPSRC です。例えば、CL ソース・ステートメントが PGMA という名前のテープのソース・ファイルに入っているとします。

最初のステップでは、次のような LABEL 属性の一時変更を指定した一時変更コマンドを使用して、テープ内のソースの位置を指定します。

```
OVRTAPF FILE(QTAPSRC) LABEL(PGMA)
```

これで、**制御言語モジュール作成 (CRTCLMOD)** コマンドで QTAPSRC ファイルをソース・ファイルとして使用できるようになります。テープ・ファイルからのソース入力を使用して CL モジュールを作成するには、次のようなコマンドを入力します。

```
CRTCLMOD MODULE(QGPL/PGMA) SRCFILE(QTAPSRC)
```

この CRTCLMOD コマンドの処理では、QTAPSRC のソース・ファイルがデータベース・ソース・ファイルと同様に処理されます。一時変更を使用して、ソースはテープに置かれます。 PGMA は QGPL に作成され、このモジュールのソースはテープ上にそのまま残されます。

関連情報:

データベース・ジョブ投入 (SBMDBJOB) コマンド

ファイル・コピー (CPYF) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

CL ソース・プログラムの構成要素

CL ソース・プログラムの構成要素として入力する各ソース・ステートメントが実際には CL コマンドであっても、このソースは、多数の典型的な CL ソース・プログラムで使用される基本的な構成要素に類別できます。

PGM コマンド

```
PGM PARM(&A)
```

PGM コマンドはオプションのコマンドで、ソース・プログラムの始めを示すとともに、プロシージャーに渡されるパラメーターを指定するためのものです。

宣言コマンド

(DCL、DCLF、COPYRIGHT、DCLPRCOPT)

プログラムまたはプロシージャー変数が使用された場合は、その変数の必須の宣言で、サブルーチン・スタックのサイズのオプショナル定義です。 DCLPRCOPT では、CL コンパイラーの起動に使用された CL コマンドに指定されたコンパイラー処理オプションを指定変更することもできます。宣言コマンドは、PGM コマンドを除くすべてのコマンドより前になければなりません。

INCLUDE コマンド

コンパイル時に追加の CL ソース・コマンドを組み込む CL コマンドです。

CL 処理コマンド

CHGVAR、SNDPGMMSG、OVRDBF、DLTF、

定数または変数を処理するソース・ステートメントとして使用される CL コマンドです。 (これは部分的なリストです。)

ロジック制御コマンド

IF、THEN、ELSE、DO、ENDDO、DOWHILE、DOUNTIL、DOFOR、LEAVE、ITERATE、GOTO、SELECT、ENDSELECT、WHEN、OTHERWISE、CALLSUBR、SUBR、RTNSUBR、ENDSUBR

CL プログラムまたは CL プロシージャー内での処理の論理的な流れの制御に使用されるコマンドです。

組み込み関数

%SUBSTRING (%SST)、%SWITCH、%BINARY (%BIN)、%ADDRESS (%ADDR)、%OFFSET (%OFS)、%CHECK、%CHECKR、%SCAN、%TRIM、%TRIML、%TRIMR、%CHAR、%DEC、%INT、%UINT (%UNS)、%LEN、%SIZE、 %LOWER、%UPPER、%PARMS

算術式、文字ストリング式、比較式、または論理式の中で使用される組み込み関数および演算子です。

プログラム制御コマンド

CALL、RETURN、TFRCTL

他のプログラムに制御権を渡すのに使用される CL コマンドです。

プロシージャー制御コマンド

CALLPRC

別のプロシージャーに制御権を渡すのに使用される CL コマンドです。

ENDPGM コマンド

ENDPGM

オプションのコマンドで、プログラムの終わりを示します。

上記の構成要素の順序、組み合わせ、および範囲は、アプリケーションのロジックと設計によって異なります。

CL プログラムまたは CL プロシージャーでは他のオブジェクトが参照されることがありますが、それらのオブジェクトは、プロシージャーの作成時に存在していなければならない場合と、コマンドの処理時に存在していなければならない場合と、その両方の時点で存在していなければならない場合とがあります。状況に応じて、プログラムまたはプロシージャーを正しく実行するには、次のオブジェクトが必要になる場合があります。

- 表示装置ファイル。これは、ディスプレイ装置に表示する情報の様式を設定するためのものです。画面を使用するプロシージャーの場合、そのプロシージャーの作成前に、 **表示装置ファイル作成 (CRTDSPF)** コマンドを使用して、表示装置ファイルとレコード様式を作成しておかなければなりません。さらに、**ファイル宣言 (DCLF)** コマンドを使用して、宣言セクションのプロシージャーに表示装置ファイルを宣言しなければなりません。
- データベース・ファイル。データベース・ファイルのレコードは CL プロシージャーで読み取ることができます。プロシージャーでデータベース・ファイルを使用する場合には、そのプロシージャーの作成の前に、 **物理ファイル作成 (CRTPF)** コマンドまたは**論理ファイル作成 (CRTLF)** コマンドを使用して、そのファイルを作成しておかなければなりません。ファイルのレコード様式を定義するには、データ記述仕様 (DDS)、構造化照会言語 (SQL)、または対話式データ定義ユーティリティー (IDDU) を使用することができます。また、**ファイル宣言 (DCLF)** コマンドを使用して、DCL セクションでプロシージャーに対してこのファイルの宣言をしておかなければなりません。

- 他のプログラム。 CALL コマンドを使用する場合、呼び出されるプログラムはその CALL コマンドを処理する時点で存在していなければなりません。呼び出し側 CL プログラムまたは CL プロシージャーのコンパイル時にはなくても構いません。
- 他のプロシージャー。 **CALLPRC** コマンドを使用する場合、呼び出されるプロシージャーは、バインド制御言語プログラム作成 (CRTBNDCL) またはプログラム作成 (CRTPGM) コマンドの実行時に存在していなければなりません。

関連タスク:

515 ページの『CL プログラム内のオブジェクトへのアクセス』
CL プログラムからオブジェクトにアクセスするには、そのオブジェクトを参照するコマンドを実行しているときに、オブジェクトは指定したライブラリーに存在する必要があります。

519 ページの『CL プログラムまたはプロシージャーでのファイルの処理』
CL プロシージャーおよびプログラムでは、表示装置ファイルとデータベース・ファイルの 2 つのタイプのファイルがサポートされています。

277 ページの『プログラムおよびプロシージャー相互間の制御の受け渡しと通信』
プログラム呼び出し (CALL)、**結合プロシージャー呼び出し (CALLPRC)**、および **戻り (RETURN)** コマンドを使用して、プログラムとプロシージャー相互間での制御の受け渡しを行うことができます。

関連情報:

物理ファイル作成 (CRTPF) コマンド

論理ファイル作成 (CRTLF) コマンド

表示装置ファイル作成 (CRTDSPF) コマンド

ファイル宣言 (DCLF) コマンド

プログラム呼び出し (CALL) コマンド

結合プロシージャー呼び出し (CALLPRC) コマンド

例: 単純な CL プログラム

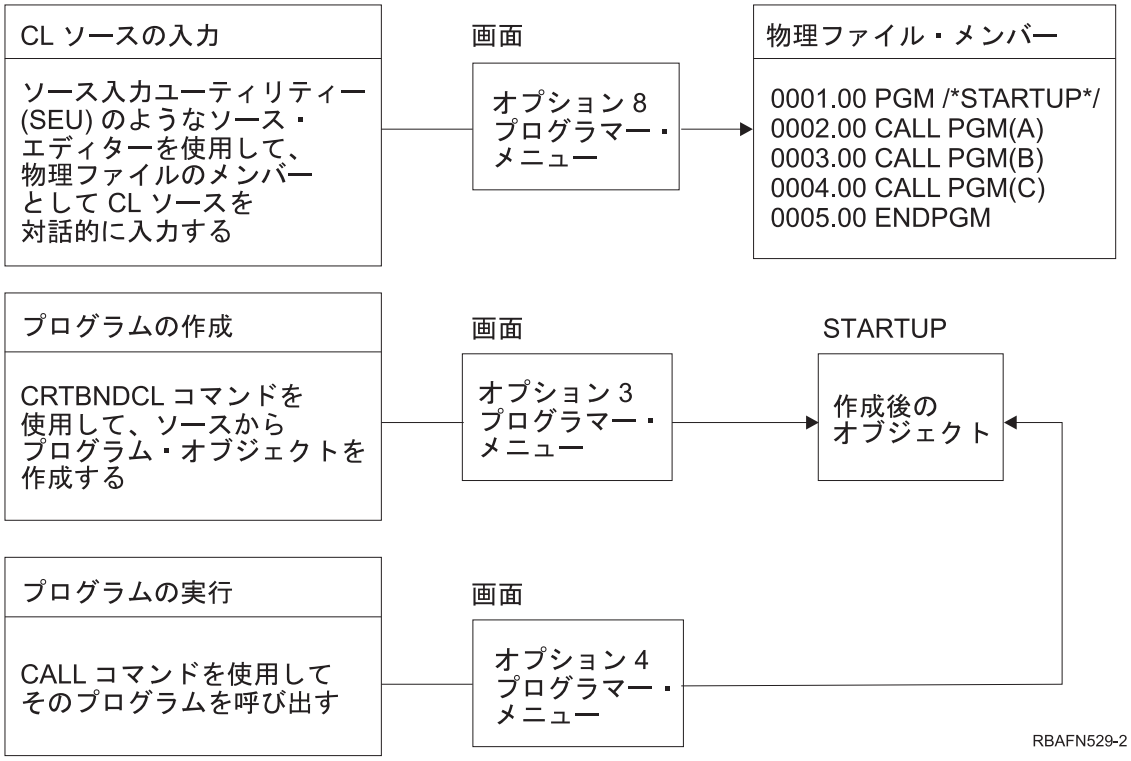
この例は、複数のプログラムを呼び出す CL プログラムです。

CL プログラムまたは CL プロシージャーは、必要に応じて単純なものにも複雑なものにもすることができます。例えば、一日の始めにシステム・オペレーターが必ず行う一連の作業 (プログラム A、B、および C を呼び出すなど) を 1 つにまとめて簡素化したい場合には、次のようにコーディングして、STARTUP という CL プロシージャーを作成することができます。

```
PGM  /*  STARTUP  */
CALL PGM(A)
CALL PGM(B)
CALL PGM(C)
ENDPGM
```

この例では、プログラムの作成にプログラマー・メニューが使用されます。また、WebSphere Development Studio for System i ライセンス・プログラムの機能である、プログラム開発管理機能 (PDM) またはリモート・システム・エクスプローラー (RSE) を使用することもできます。

このプログラムまたはプロシージャーを入力、作成、および使用するには、次のステップに従ってください。



CL ソースを入力するには、次のステップに従ってください。

1. プログラマー・メニューのオプション 8 (ソースの編集) を選択し、パラメーター・フィールドに STARTUP を指定してください。 (このオプションにより STARTUP という名前のソース・メンバーが作成され、これがプログラムの名前にもなります。)
2. 「タイプ」フィールドに CLLE を指定し，実行キーを押してください。
3. SEU 画面が表示されたら、I (挿入) 行コマンドを使用して、CL コマンドを入力してください (CALL も CL コマンドの 1 つです)。

```
桁 . . . . . :   1  71              編集                          QGPL/QCLSRC
SEC==> _____________________________________________                STARTUP
FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
      ***************** データの始め ***********************************
.......
.......
.......
.......
.......
.......
```

ソース・ステートメントの入力終了後、次のステップを行ってください。

1. F3 キーを押して SEU を終了する。
2. 終了画面ではデフォルトのオプション (オプション 2、終了してメンバーを更新) をそのまま使用し、実行キーを押してプログラマー・メニューに戻ります。
3. 入力したソース・ステートメントからプログラムを作成するために、オプション 3 (オブジェクトの作成) を選択する。画面上の他の情報は変更する必要はありません。

注: 参照されるプログラム (A、B、および C) は、プログラム STARTUP の作成時にはなくても構いません。

作成したプログラムは、プログラマー・メニューでオプション 4 (プログラムの呼び出し) を選択し、パラメーター・フィールドに STARTUP を指定することによって呼び出すことができます。ただし、このサンプル・プログラムを実行するには、CALL コマンドの処理の時点で、参照されるプログラムが存在する必要があります。

関連情報:

プログラム呼び出し (CALL) コマンド

CL プログラムまたは CL プロシージャーで使用されるコマンド

CL プログラムまたはプロシージャーは、CL コマンドのみを含みます。 IBM 提供のコマンド、またはユーザー定義のコマンドも使用することができます。

IBM 提供のコマンドの中には、CL プログラムまたは CL プロシージャーで使用できないものもあります。 CL コマンド・ファインダーまたはオンライン・ヘルプを使用すると、CL コマンドの説明を見つけたり、プログラムやプロシージャーでの適用度を判断したりするのに役立ちます。

RQSDTA および CMD パラメーターで入力されるコマンド

ジョブ転送 (TFRJOB) および**ジョブ投入 (SBMJOB)** コマンドなど、特定の CL コマンドには、**データまたはコマンドの要求** (RQSDTA) または **コマンド** (CMD) パラメーターがあり、これらのパラメーターには、他の CL コマンドをパラメーター値として指定することができます。この場合、CL プログラムまたは CL プロシージャーの中でだけ使用できるコマンドを、RQSDTA または CMD パラメーターの値として使用することはできません。

関連情報:

ジョブ転送 (TFRJOB) コマンド

ジョブ投入 (SBMJOB) コマンド

CL プログラムおよびプロシージャーでよく使用されるコマンド

リストには、CL プログラムおよびプロシージャーで頻繁に使用されるコマンドが含まれています。このリストを使用して、必要な機能を持つコマンドを選択することができます。

システムの機能	コマンド	コマンドの機能
プログラムまたはプロシージャー制御の変更	**呼び出し (CALL)**	プログラムを呼び出す。
	プロシージャーの呼び出し (CALLPRC)[1]	プロシージャーを呼び出す。
	戻り (RETURN)	プログラムまたはプロシージャーを実行するコマンドに続くコマンドに戻る。
CL プログラムまたはプロシージャーの制限	**プログラム終了 (ENDPGM)**[1]	CL ソース・プログラムの終わりを示す。
	プログラム (PGM)[1]	CL ソース・プログラムの始めを示す。

システムの機能	コマンド	コマンドの機能
CL プログラムまたはプロシージャーのロジック	サブルーチン呼び出し (CALLSUBR)[1]	同じプログラムまたはプロシージャー内で定義されたサブルーチンに制御を渡す。
	Do (DO)[1]	DO グループの始めを示す。
	Do For (DOFOR)[1]	指定された値に基づいてコマンドをゼロ回以上処理する DO グループの始めを示す。
	Do Until (DOUNTIL)[1]	論理式の値が真になるまでの間にコマンドのセットを処理する DO グループの始めを示す。
	Do While (DOWHILE)[1]	論理式の値が真である間にコマンドのセットを処理する DO グループの始めを示す。
	Else (ELSE)[1]	IF コマンドにおける ELSE (偽) 条件の場合に取る処置を定義する。
	End Do (ENDDO)[1]	DO グループの終わりを示す。
	End Select (ENDSELECT)[1]	SELECT グループの終わりを示す。
	サブルーチン終了 (ENDSUBR)[1]	サブルーチンを終了する。
	Go To (GOTO)[1]	別のコマンドに分岐する。
	仮定 (IF)[1]	論理式の値に基づいてコマンドを処理する。
	Iterate (ITERATE)[1]	Do While、Do Until、または Do For グループ内のコマンドの処理を終了し、グループの条件を再び評価する。
	Leave (LEAVE)[1]	Do While、Do Until、または Do For グループ内のコマンドの処理を終了する。
	Otherwise (OTHERWISE)[1]	SELECT グループ内の WHEN コマンドで真になる条件が存在しない場合に処理されるコマンドを定義する。
	サブルーチンからの戻り (RTNSUBR)[1]	サブルーチンを終了する。
	サブルーチン (SUBR)[1]	サブルーチンを定義するコマンドのグループを区切る。
	Select (SELECT)[1]	コマンド・グループの条件付き処理を可能にする SELECT グループの始めを示す。
	When (WHEN)[1]	論理式の値が真である場合に SELECT グループ内のコマンドを処理する。
CL プログラムまたはプロシージャーの変数	変数変更 (CHGVAR)[1]	CL 変数の値を変更する。
	宣言 (DCL) [1]	変数を宣言する。
変換	変数変更 (CHGVAR)[1]	CL 変数の値を変更する。
	日付形式変換 (CVTDAT)[1]	日付の形式を変更する。
データ域	データ域変更 (CHGDTAARA)	データ域を変更する。
	データ域作成 (CRTDTAARA)	データ域を作成する。
	データ域削除 (DLTDTAARA)	データ域を削除する。
	データ域表示 (DSPDTAARA)	データ域を表示する。
	データ域検索 (RTVDTAARA)[1]	データ域の内容を CL 変数にコピーする。

システムの機能	コマンド	コマンドの機能
ファイル	受信終了 (ENDRCV)[1]	RCVF、SNDF、または SNDRCVF コマンドによって表示装置ファイルに前に出された入力要求を取り消す。
	ファイル宣言 (DCLF)[1]	表示装置ファイルまたはデータベース・ファイルを宣言する。
	ファイル受信 (RCVF)[1]	表示装置ファイルまたはデータベース・ファイルからのレコードを読み取る。
	メンバー記述の検索 (RTVMBRD)[1]	データベース・ファイルの特定のメンバーの記述を検索する。
	ファイル送信 (SNDF)[1]	レコードを表示装置ファイルに書き込む。
	ファイル送信/受信 (SNDRCVF)[1]	レコードを表示装置ファイルに書き込み、ユーザーが応答してからそのレコードを読み取る。
	待機 (WAIT)[1]	表示装置ファイルに対して出された SNDF、 RCVF、または SNDRCVF からのデータを待つ。
メッセージ	メッセージ・モニター (MONMSG)[1]	プログラムのメッセージ待ち行列に送られてくるエスケープ・メッセージ、状況メッセージ、および通知メッセージを監視する。
	メッセージ受信 (RCVMSG)[1]	メッセージ待ち行列から CL プログラムまたはプロシージャー内の CL 変数にメッセージをコピーする。
	メッセージ除去 (RMVMSG)[1]	指定されたメッセージ待ち行列から指定されたメッセージを除去する。
	メッセージ検索 (RTVMSG)[1]	メッセージ・ファイルから CL プログラムまたはプロシージャー変数に事前定義メッセージをコピーする。
	プログラム・メッセージ送信 (SNDPGMMSG)[1]	プログラム・メッセージをメッセージ待ち行列に送る。
	応答送信 (SNDRPY)[1]	照会メッセージの送信元に応答メッセージを送る。
	ユーザー・メッセージ送信 (SNDUSRMSG)	通知メッセージまたは照会メッセージをディスプレイ装置またはシステム・オペレーターに送る。
その他のコマンド	オブジェクト検査 (CHKOBJ)	オブジェクトの存在の有無を検査し、必要があればそのオブジェクトの使用に必要な権限の有無を検査する。
	CL ソースの組み込み (INCLUDE)[1]	CL ソース・コマンドをコンパイル時に組み込む。
	コマンド使用状況印刷 (PRTCMDUSG)	指定した CL プログラムまたはプロシージャーのグループ内で使用されている、指定したコマンドのグループのリストの相互参照リストを作成する。
	構成ソースの検索 (RTVCFGSRC)	既存の構成オブジェクトを作成するための CL コマンド・ソースを生成し、そのソースをソース・ファイル・メンバーに入れる。
	構成状況検索 (RTVCFGSTS)[1]	アプリケーションが 3 つの構成オブジェクト (回線、制御装置、入出力装置) から構成状況を検索できるようにする。
	ジョブ属性検索 (RTVJOBA)[1]	1 つ以上のジョブ属性の値を検索し、CL 変数に入れる。
	システム値検索 (RTVSYSVAL)[1]	システム値を検索し CL 変数に入れる。
	ユーザー・プロファイル検索 (RTVUSRPRF)[1]	ユーザー・プロファイル属性を検索し、CL 変数に入れる。

システムの機能	コマンド	コマンドの機能
プログラム作成コマンド	**制御言語モジュール作成 (CRTCLMOD)**	統合言語環境 (ILE) CL モジュールを作成する。
	モジュール削除 (DLTMOD)	モジュールを削除する。
	プログラム削除 (DLTPGM)	プログラムを削除する。
	バインド CL プログラムの作成 (CRTBNDCL)	ILE CL プログラムを作成する。
	制御言語プログラム作成 (CRTCLPGM)	オリジナル・プログラム・モデル (OPM) CL プログラムを作成する。
	プログラム作成 (CRTPGM)	1 つ以上のモジュールから ILE プログラムを作成する。
	サービス・プログラムの作成 (CRTSRVPGM)	1 つ以上のモジュールから ILE サービス・プログラムを作成する。
[1] は、CL プログラムおよびプロシージャー内でしか使用できないコマンドを示しています。		

関連情報:

CL コマンド検索プログラム

CL プログラムまたはプロシージャーによって実行される操作

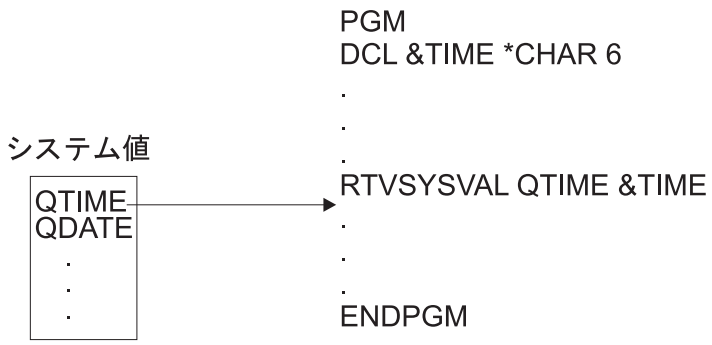
CL プログラムまたはプロシージャーを使用して実行できる操作のタイプの概説を説明します。

一般に、以下の操作を実行できます

- 変数、ロジック制御コマンド、式、および組み込み関数を使用して、CL プログラムまたはプロシージャーでデータを操作および処理する。

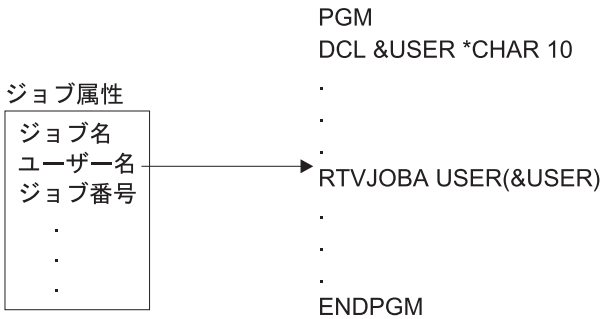
```
PGM
DCL &C *LGL
DCL &A *DEC VALUE(22)
DCL &B *CHAR VALUE(ABCDE)
•
•
•
CHGVAR &A (&A + 30)
•
•
•
IF (&A < 50) THEN(CHGVAR &C '1')
•
DSPLIB ('Q' vv &B)
•
IF (%SST(&B 5 1)=E) THEN(CHGVAR &A 12)
•
•
•
ENDPGM
```

- システム値を CL プログラムまたは CL プロシージャーの変数として使用する。



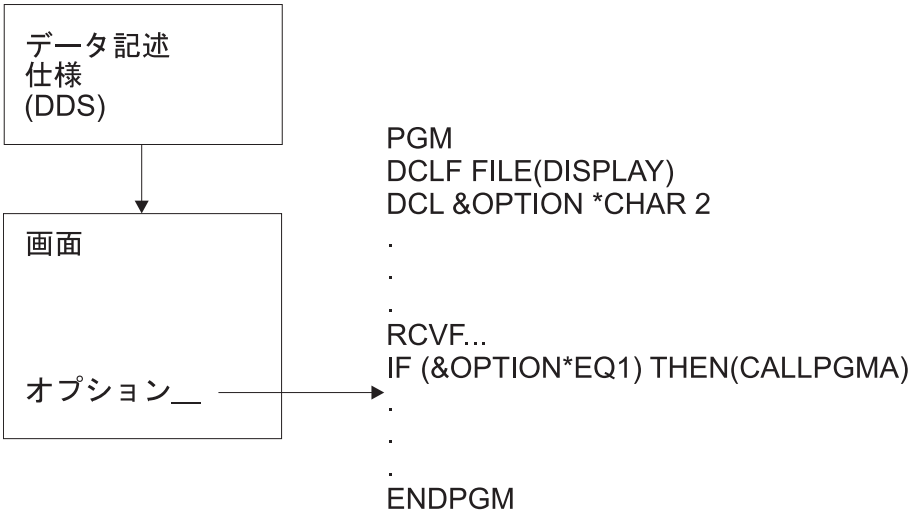
RBAFN551-0

- ジョブ属性を CL プログラムまたは CL プロシージャーの変数として使用する。



RBAFN552-0

- 表示装置ファイルと CL プログラムまたは CL プロシージャーとの間でデータのやりとりを行う。

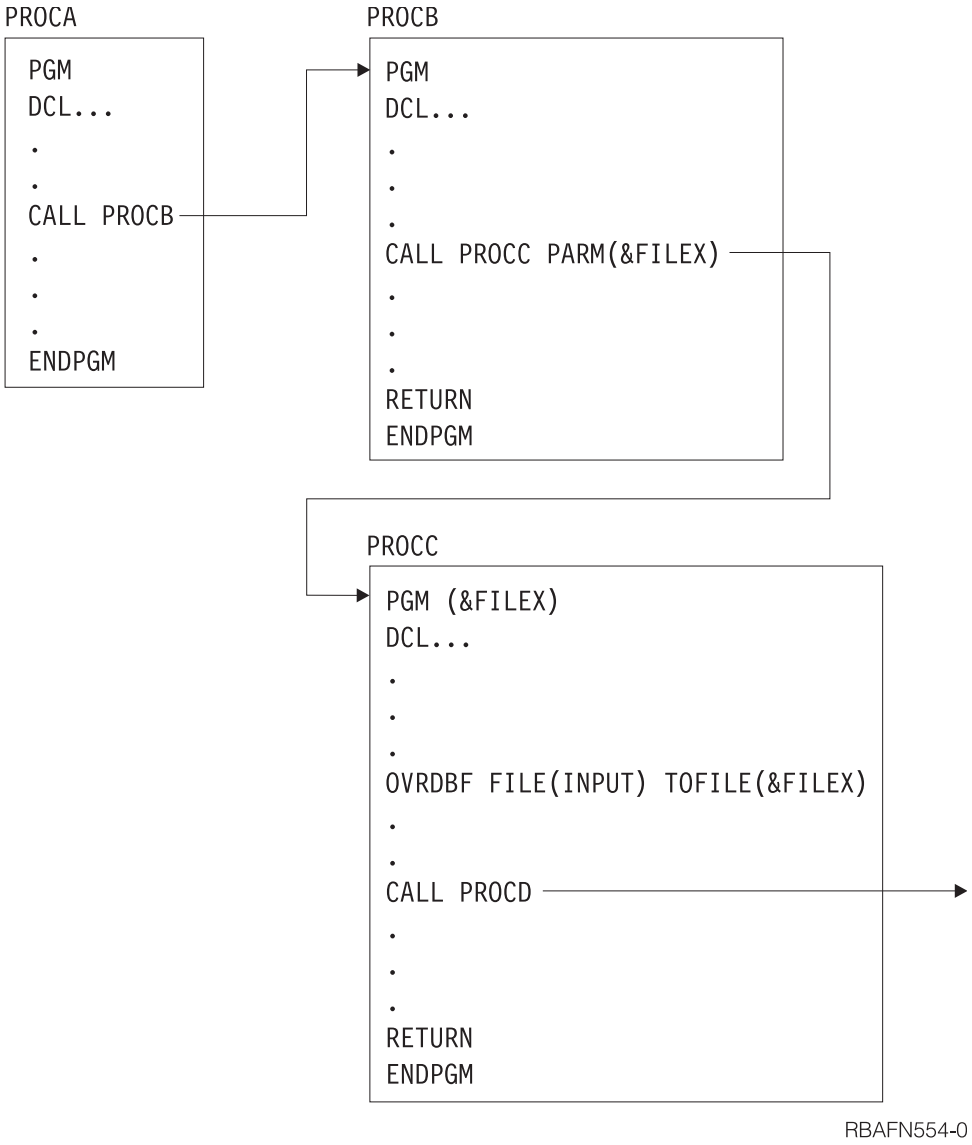


- ジョブのエラー・メッセージを監視する CL プログラムまたは CL プロシージャーを作成し、必要に応じて訂正処置をとる。

```
        PGM

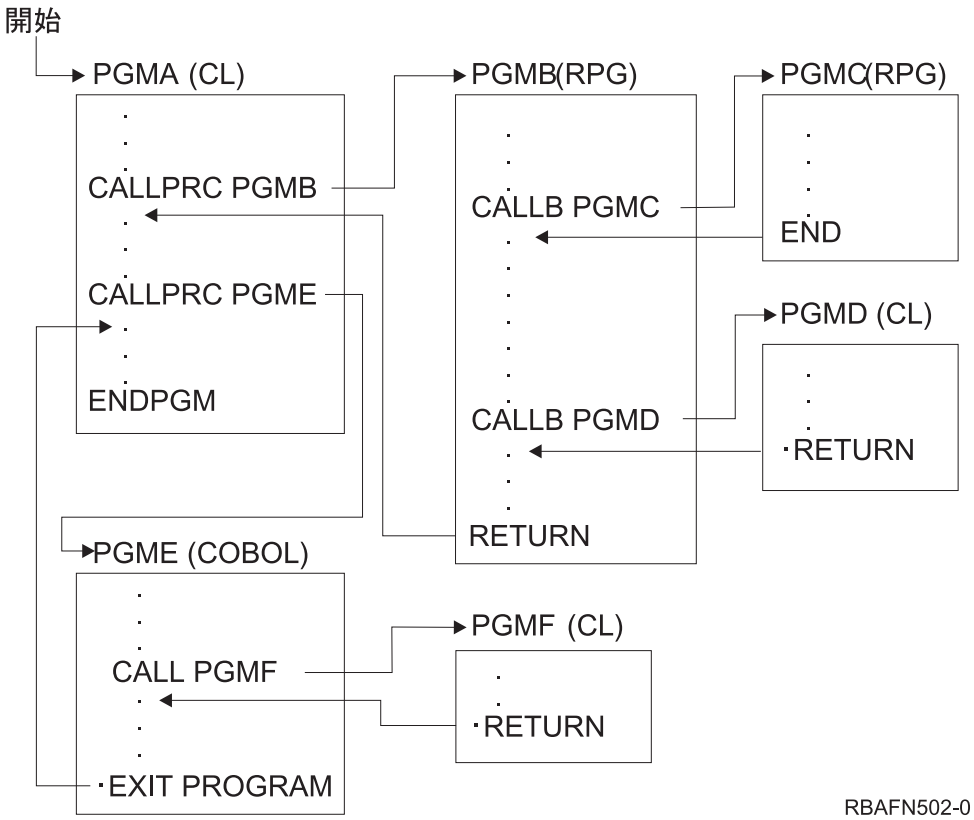
        MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
        CALL PROGA
        CALL PROGB
        RETURN
ERROR:  SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*ESCAPE)
        ENDPGM
```

- プロシージャーおよびプログラム間の処理を制御し、ファイルの一時変更のために、ある CL プロシージャーまたはプロシージャーから他のプログラムまたはプロシージャーにパラメーターを渡す。



CL プログラムまたはプロシージャーを制御プログラムまたはプロシージャーとして使用して、他の言語で作成されたプログラムまたはプロシージャーを呼び出すことができます。前の図は、1 つのアプリケーションにおける、CL プログラムまたはプロシージャー、RPG IV、および統合言語環境 (ILE) COBOL プロシージャーとの間での制御権が受け渡される方法を示しています。このアプリケーションを使用するには、アプリケーション全体を制御するプログラム A を要求します。この図で示される 1 つのバインド・プログラム (PGMA) は、PGMA を指定した CALL コマンドを使用して呼び出されます。 PGMA は次のプロシージャーから成ります。

- CL プロシージャー (PGMA) が RPG IV プロシージャー (PGMB) を呼び出す。
- RPG IV プロシージャー (PGMB) が他の RPG IV プロシージャー (PGMC) を呼び出す。
- RPG IV プロシージャー (PGMB) が CL プロシージャー (PGMD) を呼び出す。
- CL プロシージャー (PGMA) が ILE COBOL プロシージャー (PGME) を呼び出す。
- ILE COBOL プロシージャー (PGME) が CL プロシージャー (PGMF) を呼び出す。



プロシージャーは、次の例で示すように作成することができます。プロシージャーのソースは、別個のソース・メンバーに入力することができます。

```
CRTCLMOD PGMA
CRTRPGMOD PGMB
CRTRPGMOD PGMC
CRTCLMOD PGMD
CRTCBLMOD PGME
CRTCLMOD PGMF
CRTPGM PGM(PGMA) +
       MODULE(PGMA PGMB PGMC PGMD PGME PGMF) +
       ENTMOD(*FIRST)
```

CL コマンドで使用する変数

変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

CL ソース・プログラムは CL コマンドで構成されます。それぞれのコマンドは、コマンド名、パラメーター・キーワード名、およびパラメーター値で構成されます。パラメーターの値は、変数、定数、または式で表すことができます。 CL コマンドの多くのパラメーターには、その値として変数を使用することができます。 CL 変数をパラメーターの値として指定した場合、そのパラメーターを含むコマンドを実行する際に、その変数の値がパラメーターの値として使用されます。したがって、コマンドが実行されるたびに、変数の値としてそれぞれ異なる値を使用することができます。変数および式は、CL プロシージャーおよびプログラムの中でだけパラメーターの値として使用できます。

変数はライブラリーには保管されません。変数はオブジェクトではありません。変数を含むプログラムまたはプロシージャーが活動状態ではなくなると、その変数の値は破棄されます。さらに変数を値として使用すると、アプリケーションに応じて内容が変わるオブジェクトの高度な操作が可能になるため、CL プログラ

ミングは非常に柔軟性のあるものとなります。例えば、どのプログラムまたはワークステーションを制御するかを特定せずに、他のプログラムの処理またはいくつかのワークステーションの操作を制御する CL ソース・プログラムを作成することができます。これらのプログラムやワークステーションを、システムは CL プログラムまたは CL プロシージャーの中で変数として識別します。変数の値は、CL プログラムまたは CL プロシージャーの実行時に定義 (指定) することができます。

このトピックで述べた用法に加えて、変数は次の作業を実行するために使用することもできます。

- プロシージャーおよびジョブ相互間の情報の受け渡し。
- プロシージャーとディスプレイ装置相互間の情報の受け渡し。
- 条件に基づくコマンドの処理。
- オブジェクトの作成。オブジェクト名またはライブラリー名、あるいはその両方を示す値として変数を使用することができます。次に示す 2 つの**物理ファイル作成 (CRTPF)** コマンドの例で、最初のコマンドではライブラリー名が指定されており、2 番目のコマンドではライブラリー名が変数で指定されています。

```
CRTPF  FILE(DSTPRODLB/&FILE)
CRTPF  FILE(&LIB/&FILE)
```

変数は、コマンド名またはキーワードを変更したり、CALLPRC コマンドのプロシージャー名を指定したりする目的で使用することはできません。ただし、コマンド・パラメーターは、CL プロシージャーの処理時点でプロンプト機能によって変更することができます。

コマンドのパラメーターおよびキーワードを組み立て、QCAPCMD API または QCMDEXC API を使用してそのコマンドを処理することもできます。

関連タスク:

277 ページの『プログラムおよびプロシージャー相互間の制御の受け渡しと通信』
プログラム呼び出し (CALL)、**結合プロシージャー呼び出し (CALLPRC)**、および **戻り (RETURN)** コマンドを使用して、プログラムとプロシージャー相互間での制御の受け渡しを行うことができます。

530 ページの『複数装置表示装置ファイルの処理』
複数ディスプレイ装置構成が使用されるのは、1 人の要求元により呼び出された 1 つのジョブが、1 つの表示装置ファイルを介して複数のディスプレイ装置と交信を行う場合です。 1 つの CL プログラムまたはプロシージャーで処理できる表示装置ファイルは 1 つだけですが、その表示装置ファイル、またはその表示装置ファイルの異なるレコード様式を、複数のディスプレイ装置に送ることができます。

396 ページの『実行時のユーザー入力のプロンプト』
大部分の CL プログラムおよびプロシージャーでは、ワークステーション・ユーザーがプログラムに渡されるコマンド・パラメーター値を指定するか、表示プロンプトの入力可能フィールドに入力することで、入力を行います。

390 ページの『QCAPCMD プログラム』
コマンド処理 (QCAPCMD) API は、コマンド・ストリングに対してコマンド分析プログラムの処理を実行します。

391 ページの『QCMDEXC プログラム』
コマンド実行 (QCMDEXC) API は、1 つのコマンドを実行する IBM 提供のプログラムです。

288 ページの『CALL コマンド使用時のデータ・タイプ・エラー』
プログラム呼び出し (CALL) コマンドの使用中に、データ・タイプ・エラーが生じる場合があります。

関連資料:

204 ページの『CL プログラムまたは CL プロシージャー内での処理の制御』
コマンドを使用して、CL プロシージャー内のロジックの流れを変更することができます。

CL プログラムまたはプロシージャーへの変数宣言

CL プログラムまたはプロシージャーで変数を使用できるようにするには、その前にその CL プログラムまたはプロシージャーに対してすべての変数を宣言 (定義) しておかなければなりません。

変数は、以下の 2 つの方法で宣言します。

- 変数を宣言する。変数の定義は **CL 変数宣言 (DCL)** コマンドを用いて行い、これにより変数の属性を定義します。これらの属性には、タイプ、長さ、および初期値があります。

  ```
  DCL  VAR(&AREA)  TYPE(*CHAR)  LEN(4)  VALUE(BOOK)
  ```

- ファイルを宣言する。CL プログラムまたは CL プロシージャーでファイルを使用する場合には、ファイル宣言 (DCLF) コマンドの FILE パラメーターにそのファイルの名前を指定しなければなりません。ファイルには、ファイル中のレコードおよびレコード内のフィールドの記述 (様式) が含まれています。コンパイルの過程で、ファイルで定義されているフィールドおよび標識に対応する CL 変数が、DCLF コマンドによって暗黙的に宣言されます。

  例えば、ファイルの DDS で、2 つのフィールド (F1 および F2) からなるレコードが 1 つ指定されているとすれば、 &F1 および &F2 という 2 つの変数が、プログラムの中で自動的に宣言されます。

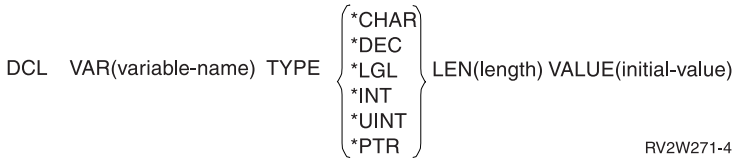
  ```
  DCLF  FILE(MCGANN/GUIDE)
  ```

  ファイルが DDS を使用せずに作成された物理ファイルの場合には、そのレコードに対応する 1 つの変数が宣言されます。この変数の名前はファイルの名前と同じであり、またその長さはファイルのレコード長と同じです。

宣言コマンドは、プログラムまたはプロシージャー内の他のすべてのコマンド (PGM コマンドを除く) よりも前になければなりませんが、宣言コマンド自体は任意の順序で使用することができます。

CL 変数宣言コマンドを使用する際の規則

最も単純な形式の **CL 変数宣言 (DCL)** コマンドには、次のようなパラメーターがあります。



DCL コマンドを使用する際には、次の規則に従ってください。

- CL 変数名はアンパーサンド (&) で始まっていなければならず、その後に 10 文字以内の文字を使用することができます。 & の後の最初の文字は英字、続く文字は英数字でなければなりません (&PART など)。
- CL 変数の値は、次のうちのいずれかでなければなりません。
  - 5000 文字までの文字ストリング。
  - 合計桁数が最大 15 桁で小数部分が 9 桁以下のパック 10 進数。
  - 論理値 '0' または '1'。 '0' はオフ、偽、または NO を意味し、'1' はオン、真、または YES を意味します。論理変数は '0' または '1' でなければなりません。
  - 2 バイト、4 バイトまたは 8 バイトの整数値。 *INT が TYPE パラメーターに指定されている場合は負の値にできます。TYPE パラメーターに *UINT が指定されている場合、値は正またはゼロでな

ければなりません。 **制御言語モジュール作成 (CRTCLMOD)** コマンドまたは**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドで CL ソースをコンパイルした場合にのみ、LEN(8) を指定できます。

  - 記憶域内でのデータのロケーションを保持することができるポインター値。
- 初期値を指定しなかった場合には、次の値がとられます。
  - 10 進変数の場合は '0'
  - 文字変数の場合はブランク
  - 論理変数の場合は '0'
  - 整変数の場合は '0'
  - ポインター変数の場合はヌル

  10 進数タイプおよび文字タイプの場合に、初期値を指定して LEN パラメーターを指定しなかった場合には、変数のデフォルトの長さはその初期値の長さと同じになります。タイプが *CHAR の場合には、LEN パラメーターを指定しなければ、ストリングは最大 5000 文字にすることができます。タイプが *INT または *UINT の場合には、LEN パラメーターを指定しなければ、デフォルトの長さは 4 です。
- パラメーターは、プログラムの DCL ステートメント内の変数として宣言します。

関連情報:

CL 変数宣言 (DCL) コマンド

基底付き変数の使用

基底付き変数は、プログラムに渡された変数をマップしたり、値の配列を操作するのに使用できます。

基本ポインターは、それを使用する前に宣言 (DCL) コマンドの ADDRESS キーワードまたは %ADDRESS 組み込み関数を使用して、設定する必要があります。基本ポインターを設定後は、その変数はローカル変数のように機能します。

以下の例で、基本ポインター &PTR は &AUTO のアドレスと等しいと宣言されます。 その後、変数 &BASED は、ポインター変数 &PTR でアドレス指定される最初の 10 バイトの値を持ちます。 この手順の後で、変数 &BASED の値は、変数 &AUTO の最初の 10 バイトに対して等しいかどうかチェックされます。 それらの値が同じ場合、つまり、ポインター &PTR が &AUTO の最初のバイトをアドレス指定する場合、ポインター・オフセットは 変数 &AUTO のバイト 11 をアドレス指定するように変更されます。 変数 &BASED は、変数 &AUTO のバイト 11-20 に等しい値を持つようになりました。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
DCL &AUTO  *CHAR 20
DCL &PTR   *PTR ADDRESS(&AUTO)
DCL &BASED *CHAR 10 STG(*BASED) BASPTR(&PTR)
:
IF COND(%SST(&AUTO 1 10) *EQ &BASED) +
   THEN(CHGVAR %OFS(&PTR) (%OFS(&PTR) + 10))
:
ENDPGM
```

関連情報:

CL 変数宣言 (DCL) コマンド

定義済み変数の使用

定義済み変数を使用すると、大きな変数から値をサブストリングとして取り出す必要がなくなるので、制御言語 (CL) の複雑なデータ構造を簡単に管理することができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

定義済み変数を使用すると、変数のさまざまな部分または特定の変数の同一の部分をさまざまな方法でマップすることができます。

以下の例で、変数 &OBJNAME は &OBJECT の最初の 10 バイトに、変数 &LIBNAME は &OBJECT の最後の 10 バイトに等しくなります。定義済み変数 &OBJNAME および &LIBNAME を使用すると、コードが読みやすくなり、操作しやすくなります。変数 &OBJECT は、&LIBNAME および &OBJNAME 変数両方の記憶域を提供します。

```
PGM
DCL &OBJECT  *CHAR 20
DCL &OBJNAME *CHAR 10 STG(*DEFINED) DEFVAR(&OBJECT)
DCL &LIBNAME *CHAR 10 STG(*DEFINED) DEFVAR(&OBJECT 11)
:
IF COND(&LIBNAME *EQ '*LIBL     ') +
   THEN(...))
:
ENDPGM
```

複数の定義を使用しても同じ記憶域を作成することができます。次の例では、変数 &BINLEN および &CHARLEN はどちらも、変数 &STRUCT の同じ 4 バイトを参照しています。したがって、次のプログラムは、その要件に最適な定義を使用することができます。

```
PGM
DCL &STRUCT  *CHAR 50
DCL &BINLEN  *INT   4 STG(*DEFINED) DEFVAR(&STRUCT)
DCL &CHARLEN *CHAR  4 STG(*DEFINED) DEFVAR(&STRUCT)
:
ENDPGM
```

次の例は、定義済み変数を使用して、変数内の値を変更する方法を示しています。この例はまた、%OFFSET 組み込み関数および基底付き変数を使用して、ライブラリー・リストをナビゲートします。これは、メッセージの置換を実行するための最適な方法ではありませんが、定義済み変数の機能の一部の説明にはなっています。

```
PGM
DCL &MESSAGE  *CHAR  25 VALUE('LIBRARY NNN IS XXXXXXXXXX')
DCL &SEQUENCE *CHAR   3 STG(*DEFINED) DEFVAR(&MESSAGE  9)
DCL &MSGLIBN  *CHAR  10 STG(*DEFINED) DEFVAR(&MESSAGE 16)
DCL &COUNTER  *INT    2
DCL &LIBL     *CHAR 165
DCL &PTR      *PTR      ADDRESS(&LIBL)
DCL &LIBLNAME *CHAR  10 STG(*BASED) BASPTR(&PTR)
:
RTVJOBA SYSLIBL(&LIBL)
CHGVAR &COUNTER 0
DOFOR &COUNTER FROM(1) TO(15)
   IF (&LIBLNAME *EQ '          ') THEN(LEAVE)
   CHGVAR &SEQUENCE &COUNTER
   CHGVAR &MSGLIBN &LIBLNAME
   SNDPGMMSG  MSGID(CPF9898) MSGF(QSYS/QCPFMSG) MSGDTA(&MESSAGE)
   CHGVAR %OFS(&PTR) (%OFS(&PTR) + 11)
ENDDO
:
ENDPGM
```

リストまたは修飾名を指定するために使用する変数

変数を使用してリストまたは修飾名を指定することができます。

パラメーターの値としてリストを指定できるものがあります。例えば、**ライブラリー・リスト変更 (CHGLIBL)** コマンドの LIBL パラメーターには、各ライブラリー間をブランクで区切ってライブラリーのリストを指定しなければなりません。このリストの個々の要素の値として変数を使用することができます。

```
CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

リストの要素を指定するために変数を使用する場合に、各要素は個別に宣言しなければなりません。

```
DCL VAR(&LIB1) TYPE(*CHAR) LEN(10) VALUE(QTEMP)
DCL VAR(&LIB2) TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL VAR(&LIB3) TYPE(*CHAR) LEN(10) VALUE(DISTLIB)
CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

リストの要素の変数を 1 つの文字ストリングとして指定することはできません。

誤り:

```
DCL   VAR(&LIBS) TYPE(*CHAR) LEN(20) +
      VALUE('QTEMP QGPL DISTLIB')
CHGLIBL LIBL(&LIBS)
```

リストとして 1 つの文字ストリングを指定した場合、システムはそのリストを個別の要素から成るリストとは見なさないので、エラーが起こります。

また、修飾名の指定に変数を使用することもできます。ただし、修飾子としての名前はそれぞれ別個の変数として宣言されていなければなりません。

```
DCL VAR(&PGM) TYPE(*CHAR) LEN(10)
DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
CHGVAR VAR(&PGM) VALUE(MYPGM)
CHGVAR VAR(&LIB) VALUE(MYLIB)
.
.
.
DLTPGM PGM(&LIB/&PGM)
ENDPGM
```

この例では、プログラム名とライブラリー名はそれぞれ別個に宣言されています。プログラム名とライブラリー名を、次のように 1 つの変数として指定することはできません。

誤り:

```
DCL VAR(&PGM) TYPE(*CHAR) LEN(11)
CHGVAR VAR(&PGM) VALUE('MYLIB/MYPGM')
DLTPGM PGM(&PGM)
```

この例の値も、2 つのオブジェクト (ライブラリーおよびオブジェクト) ではなく、1 つの文字ストリングとシステムは見なします。修飾名を、文字ストリング値をとる単一の変数として処理する必要がある場合には、組み込み関数 %SUBSTRING および *TCAT 連結機能を使用して、オブジェクト名およびライブラリー名を別々の変数に割り当てることができます。

関連タスク:

312 ページの『CL コマンドの定義』
CL コマンドを使用することによって、ユーザーは、広範囲にわたる機能をシステムに要求できます。IBM 提供のコマンドを使用して、コマンド・パラメーター用のデフォルト値を変更し、ユーザー独自のコマンドを定義することができます。

関連資料:

243 ページの『%SUBSTRING 組み込み関数』
サブストリング組み込み関数 (%SUBSTRING または %SST) は、既存の文字ストリングのサブセットとしての文字ストリングを作成します。

変数内の文字の大文字小文字

CL 変数内の文字に使用される大文字小文字には制限があります。

変数の値として使用できる予約値 (*LIBL など) は、特に単一引用符で囲んだ文字ストリングとして指定する場合には、大文字で表さなければなりません。例えば、コマンドでライブラリー名として変数を使用したい場合には、正しいコーディングは次のとおりです。

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*LIBL')
DLTPGM &LIB/MYPROG;
```

ここで、VALUE パラメーターを次のように指定するのは誤り です。

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*libl')
```

この VALUE パラメーターが単一引用符で囲まれていない場合には、その値は自動的に大文字に変換されるので、このパラメーターは正しい値として処理されます。このようなエラーがよく生じるのは、ディスプレイ装置からプロシージャーまたはプログラムに渡されるパラメーターが文字ストリングとして入力され、ディスプレイ装置での入力が小文字で行われる場合です。

注**:** 前の段落の記述では、大文字に変換されるかどうかが言語によって異なるという事実が考慮されていません。システムに依存して値を大文字に変換すると、予期しない結果が生じる場合があります。

予約されているパラメーター値または数値パラメーターの値を置換する変数

文字変数は、複数のコマンドに対して使用し、コマンド・パラメーターの値を示すことができます

CL コマンドによっては、そのパラメーターの値として数値または事前定義値 (予約値) の両方を指定できるものがあります。このような場合には、そのコマンド・パラメーターの値を文字変数を使用して指定することができます。

コマンドの各パラメーターは、それぞれ特定のタイプの値だけを受け入れます。パラメーターの値として使用できるのは、整数、文字ストリング、予約値、指定されたタイプの変数、またはこれらのいくつかを組み合わせたものです。パラメーターには必須の値のタイプがあるものもあります。パラメーターに数値が指定できる場合 (コマンドで値が *INT2、 *INT4、*UINT2、*UINT4、または *DEC として定義されている場合) であって、しかも予約値 (前にアスタリスクのついた文字ストリング) も指定できる場合には、そのパラメーターの値として変数を指定することができます。予約値を使用したい場合には、変数を TYPE(*CHAR) として宣言していなければなりません。

例えば、**出力待ち行列変更 (CHGOUTQ)** コマンドにはジョブ区切り (JOBSEP) パラメーターがありますが、このパラメーターは数値 (0 から 9) またはデフォルト値である *SAME のどちらかを値として指定することができます。数値と事前定義値のどちらでも受け入れられるため、JOBSEP の値として文字変数を使用する CL ソース・プログラムを作成することもできます。

```
     PGM
     DCL &NRESP *CHAR LEN(6)
     DCL &SEP *CHAR LEN(4)
     DCL &FILNAM *CHAR LEN(10)
     DCL &FILLIB *CHAR LEN(10)
     DCLF.....
     .
     .
```

```
      .
LOOP: SNDRCVF.....
      IF (&SEP *EQ IGNR) GOTO END
      ELSE IF (&SEP *EQ NONE) CHGVAR &NRESP '0'
      ELSE IF (&SEP *EQ NORM) CHGVAR &NRESP '1'
      ELSE IF (&SEP *EQ SAME) CHGVAR &NRESP '*SAME'
      CHGOUTQ OUTQ(&FILLIB/&FILNAM) JOBSEP(&NRESP)
      GOTO LOOP
END:  RETURN
      ENDPGM
```

上記の例では、ディスプレイ装置のユーザーは、指定した出力待ち行列に必要なジョブ区切りの数を画面に入力します。変数 &NRESP は、数値または事前定義値をとる文字変数です (単一引用符の用法に注意してください)。 **出力待ち行列変更 (CHGOUTQ)** コマンドの JOBSEP パラメーターは、これらの値が数値または事前定義値として入力されたものとして認識します。このプログラムで使用される表示装置ファイルの DDS では、ユーザーの応答を IGNR、NONE、NORM、または SAME に限定するために VALUES キーワードが使用されていなければなりません。

数値タイプの値を指定できるパラメーター (*INT2、*INT4、 *UINT2、*UINT4、または *DEC) で、しかも予約値 (*SAME など) を使用しない場合には、そのパラメーターで 10 進変数または整変数を使用することができます。

CL ソース・プログラムの中でプロンプターを使用することによって、これと同じ機能を得ることもできます。

関連タスク:

312 ページの『CL コマンドの定義』

CL コマンドを使用することによって、ユーザーは、広範囲にわたる機能をシステムに要求できます。IBM 提供のコマンドを使用して、コマンド・パラメーター用のデフォルト値を変更し、ユーザー独自のコマンドを定義することができます。

変数の値の変更

CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

変数の値は次のいずれかに変更することができます。

- 定数に対しては、以下の例で示します。

  ```
  CHGVAR   VAR(&INVCMPLT)   VALUE(0)
  ```

  変数 &INVCMPLT の値は 0 に設定されます。

  以下の表記も使用できます。

  ```
  CHGVAR   &INVCMPLT  0
  ```

- 別の変数の値に対しては、以下の例で示します。

  ```
  CHGVAR   VAR(&A)   VALUE(&B)
  ```

  変数 &A の値は変数 &B の値に設定されます。

  以下の表記も使用できます。

  ```
  CHGVAR   &A  &B
  ```

- 計算後の式の値に対しては、以下の例で示します。

  ```
  CHGVAR   VAR(&A)   VALUE(&A + 1)
  ```

変数 &A の値が 1 ずつ増加します。

以下の表記も使用できます。

```
CHGVAR   &A  (&A + 1)
```

- 組み込み関数 %SST により生成された値に対しては、以下の例で示します。

```
CHGVAR   VAR(&A)   VALUE(%SST(&B 1 5))
```

変数 &A の値は変数 &B の値の最初の 5 文字に設定されます。

- 組み込み関数 %SWITCH により生成された値に対しては、以下の例で示します。

```
CHGVAR  VAR(&A)  VALUE(%SWITCH(0XX111X0))
```

&A の値は、ジョブ・スイッチ 1 および 8 が 0 の場合、またジョブ・スイッチ 4、5 および 6 が 1 の場合は 1 に設定されます。それ以外の場合、&A の値は 0 に設定されます。

- 組み込み関数 %BIN により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%BIN((%B 1 4))
```

変数 &B の最初の 4 文字が等価 10 進数に変換され、変数 &A に保管されます。

- 組み込み関数 %CHECK により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%CHECK('0123456789' &B))
```

変数 &B 内の値が検査され、数字ではない左端の文字の位置が変数 &A 内に保管されます。変数 &B 内のすべての文字が数字である場合は、値ゼロが変数 &A 内に保管されます。

- 組み込み関数 %CHECKR により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%CHECKR('*' &B))
```

変数 &B 内の値が検査され、アスタリスク (*) ではない右端の文字の位置が変数 &A 内に保管されます。変数 &B 内のすべての文字がアスタリスクである場合は、値ゼロが変数 &A 内に保管されます。

- 組み込み関数 %SCAN により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%SCAN('.' &B))
```

変数 &B 内の値がスキャンされ、左端のピリオド (.) 文字の位置が変数 &A 内に保管されます。変数 &B 内にピリオド文字がない場合は、値ゼロが変数 &A 内に保管されます。

- 組み込み関数 %TRIM により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%TRIM(&B '* '))
```

変数 &B 内にある先行または末尾のアスタリスク (*) およびブランク文字が切り取られ、その結果のストリングが変数 &A 内に保管されます。

- 組み込み関数 %TRIML により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%TRIML(&B))
```

変数 &B 内にある先行ブランク文字が切り取られ、その結果のストリングが変数 &A 内に保管されます。

- 組み込み関数 %TRIMR により生成された値:

```
CHGVAR  VAR(&A)  VALUE(%TRIMR(&B '*'))
```

変数 &B 内にある末尾アスタリスク (*) 文字が切り取られ、その結果のストリングが変数 &A 内に保管されます。

- 組み込み関数 %CHAR により生成された値:

  ```
  CHGVAR  VAR(&A)  VALUE(%CHAR(&B))
  ```

  変数 &B が文字フォーマットに変換され、その結果のストリングが変数 &A 内に保管されます。
- 組み込み関数 %UPPER により生成された値:

  ```
  CHGVAR  VAR(&A)  VALUE(%UPPER(&B))
  ```

  変数 &B 内の小文字が大文字に変換され、その結果のストリングが変数 &A 内に保管されます。
- 組み込み関数 %SIZE により生成された値:

  ```
  CHGVAR  VAR(&A)  VALUE(%SIZE(&B))
  ```

  変数 &B によって占有されるバイト数が変数 &A 内に保管されます。
- 組み込み関数 %PARMS により生成された値:

  ```
  CHGVAR  VAR(&A)  VALUE(%PARMS())
  ```

  プログラムに渡されたパラメーターの数が変数 &A 内に保管されます。

また、CHGVAR コマンドを使用して、ローカル・データ域を検索したり、変更することもできます。例えば以下のコマンドは、ローカル・データ域の 10 バイトをブランクにし、ローカル・データ域の一部を検索します。

```
CHGVAR %SST(*LDA 1 10) ' '

CHGVAR &A %SST(*LDA 1 10)
```

次の表は、値 (リテラルまたは変数) から変数への有効な割り当てを示しています。

表 *23.* 値から変数への有効な割り当て

	論理値	文字値	**10** 進値	符号付き整数値	符号なし整数値
論理変数	X				
文字変数	X	X	X	X	X
10 進変数		X	X	X	X
符号付き整変数		X	X	X	X
符号なし整変数		X	X	X	X

注**:**

1. 文字変数に数値を指定する場合には、次の点に注意してください。
   - 文字変数の値は右寄せされ、必要があれば左側にゼロが埋められる。
   - 文字変数は、小数点および負符号 (-) が必要な場合には、それを入れるのに十分な長さでなければならない。
   - 負符号 (-) を使用する場合、値の左端の桁に置かれる。

   例えば、文字変数 &A を 10 進変数 &B の値に変更するものとします。 &A の長さが 6 で、&B の長さは 5 で、小数点以下の桁数は 2 です。この場合、&B の値が 123 であるとすれば、&A の結果の値は 123.00 になります。
2. 数値変数に文字値を指定する場合には、次の点に注意してください。
   - 小数点は、文字値における小数点の位置によって決まる。文字値に小数点が含まれていない場合には、値の右端の桁に小数点が置かれます。

- 文字値にはその値の左側に負符号 (-) または正符号 (+) を含めることができるが、中間にブランクがあってはならない。符号のない文字値は正と見なされます。
- 文字値の中の小数点の右側の桁数が、該当の数値変数で指定された長さよりも多い場合には、10 進変数の場合には文字の切り捨てが生じ、整変数の場合には丸めが生じます。小数点の左側の桁数に超過が生じた場合には、切り捨ては行われず、エラーが起こります。

  例えば、10 進変数 &C を文字変数 &D の値に変更するものとします。 &C の長さと小数部分の桁数がそれぞれ 5 と 2、 &D の長さが 10 で現在の値が +123.1bbbb (b= ブランク) とすると、&C の結果の値は 123.10 になります。

関連資料:

225 ページの『%ADDRESS 組み込み関数』
アドレス組み込み関数 (%ADDRESS または %ADDR) を使用して、CL ポインター変数に保管されているメモリー・アドレスを変更またはテストすることができます。

226 ページの『%BINARY 組み込み関数』
2 進数組み込み関数 (%BINARY または %BIN) は、指定された CL 文字変数の内容を、符号付き 2 進整数として解釈します。

228 ページの『%CHAR 組み込み関数』
%CHAR は、論理、10 進数、整数、または符号なし整数の各データを文字フォーマットに変換します。変換された値は、CL 変数に割り当てたり、文字定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

230 ページの『%CHECK 組み込み関数』
チェック組み込み関数 (%CHECK) は、コンパレーター・ストリング にはない文字を含む基本ストリングの 1 桁目を戻します。基本ストリングのすべての文字がコンパレーター・ストリングにも現れる場合、この関数の返す値は 0 です。

231 ページの『%CHECKR 組み込み関数』
リバース・チェック組み込み関数 (%CHECKR) は、コンパレーター・ストリング にはない文字を含む基本ストリング の最後の桁を返します。基本ストリングのすべての文字がコンパレーター・ストリングにも現れる場合、この関数の返す値は 0 です。

232 ページの『%DEC 組み込み関数』
%DEC は、文字、論理、10 進数、整数、または符号なし整数データをパック 10 進数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

234 ページの『%INT 組み込み関数』
%INT は、文字、論理、10 進数、または符号なし整数データを整数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

235 ページの『%LEN 組み込み関数』
%LEN 組み込み関数は、CL 数値変数または CL 文字変数の桁数または文字数を返します。

236 ページの『%LOWER 組み込み関数』
%LOWER 組み込み関数は、指定された引数と同じ長さの文字ストリングを、それぞれの大文字を対応する小文字に置き換えて戻します。

237 ページの『%OFFSET 組み込み関数』
オフセット組み込み関数 (%OFFSET または %OFS) を使用して、CL ポインター変数のオフセット部分を

保管または変更できます。

238 ページの『%PARMS 組み込み関数』
パラメーター数リターン組み込み関数 (%PARMS) は、%PARMS が使用されたプログラムに渡されたパラメーターの数を戻します。

241 ページの『%SCAN 組み込み関数』
スキャン組み込み関数 (%SCAN) は、ソース・ストリング の中の検索引数 の 1 桁目を戻し、またはそれが見つからない場合には 0 を戻します。

242 ページの『%SIZE 組み込み関数』
%SIZE 組み込み関数は、CL 変数によって占有されるバイト数を返します。

243 ページの『%SUBSTRING 組み込み関数』
サブストリング組み込み関数 (%SUBSTRING または %SST) は、既存の文字ストリングのサブセットとしての文字ストリングを作成します。

246 ページの『%SWITCH 組み込み関数』
スイッチ組み込み関数 (%SWITCH) は、8 個のスイッチの 1 つまたは複数のスイッチを、該当のジョブで既に設定されている 8 つのスイッチの値と比較し、 '0' または '1' の論理値を返します。

248 ページの『%TRIM 組み込み関数』
1 つのパラメーターを指定してトリム組み込み関数 (%TRIM) を使用すると、先行ブランクおよび末尾ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定してトリム組み込み関数 (%TRIM) を使用すると、*characters-to-trim* パラメーター内にある先行文字と末尾文字がすべて除去された文字ストリングが生成されます。

249 ページの『%TRIML 組み込み関数』
1 つのパラメーターを指定して左方トリム組み込み関数 (%TRIML) を使用すると、先行ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定して左方トリム組み込み関数 (%TRIML) を使用すると、*characters-to-trim* パラメーター内にある先行文字がすべて除去された文字ストリングが生成されます。

250 ページの『%TRIMR 組み込み関数』
1 つのパラメーターを指定して右方トリム組み込み関数 (%TRIMR) を使用すると、末尾ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定して右方トリム組み込み関数 (%TRIMR) を使用すると、*characters-to-trim* パラメーター内にある末尾文字がすべて除去された文字ストリングが生成されます。

251 ページの『%UINT 組み込み関数』
%UINT は、文字、論理、10 進数、または整数データを符号なし整数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

253 ページの『%UPPER 組み込み関数』
%UPPER 組み込み関数は、指定された引数と同じ長さの文字ストリングを、それぞれの小文字を対応する大文字に置き換えて戻します。

関連情報:

変数変更 (CHGVAR) コマンド

コマンド・パラメーターの末尾ブランク

特定のコマンド・パラメーターでは、末尾ブランクの処理方法を定義できます。

コマンドのパラメーターの一部は、VARY(*YES) というパラメーター値で定義されています。このパラメーター値を指定すると、単一引用符の間にある文字の数が、渡される値の長さになります。このように定義

されたパラメーターの値を指定するために CL 変数を使用した場合には、システムは、末尾ブランクを取り除いた長さを変数の長さとしてコマンド処理プログラムに渡します。したがって、パラメーターに末尾ブランクがあり、そのブランクに意味がある場合には、特別の処置を行って、渡される値の長さにこのようなブランクが含まれるようにする必要があります。多くのコマンド・パラメーターは、このような状態にならない形で定義され、使用されています。このような状態になる可能性の高いパラメーターの例として、**データベース・ファイル一時変更 (OVRDBF)** コマンドの POSITION パラメーターのキー値の要素があります。

このような状態が生じる場合には、パラメーターの値を単一引用符で区切ったコマンド・ストリングを構成し、そのストリングを QCMDEXC または QCAPCMD に渡して処理することによって、これらのパラメーターで意図したとおりの結果を得ることができます。

次の例は、**データベース・ファイル一時変更 (OVRDBF)** コマンドの実行の際に、末尾ブランクがキー値の一部として含まれるようにするためのプログラムを示しています。パラメーター VARY(*YES) により定義されたパラメーターを持ち、しかも末尾ブランクをパラメーターの一部として渡さなければならない他のコマンドに対しても、これと同じ方法を用いることができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
            PGM        PARM(&KEYVAL &LEN)
 /*   PROGRAM TO SHOW HOW TO SPECIFY A KEY VALUE WITH TRAILING        */
 /*      BLANKS AS PART OF THE POSITION PARAMETER ON THE OVRDBF       */
 /*      COMMAND IN A CL PROGRAM.                                     */
 /*   THE KEY VALUE ELEMENT OF THE POSITION PARAMETER OF THE OVRDBF   */
 /*      COMMAND IS DEFINED USING THE VARY(*YES) PARAMETER.           */
 /*      THE DESCRIPTION OF THIS PARAMETER ON THE ELEM COMMAND        */
 /*      DEFINITION STATEMENT SPECIFIES THAT IF A PARAMETER           */
 /*      DEFINED IN THIS WAY IS SPECIFIED AS A CL VARIABLE THE        */
 /*      LENGTH IS PASSED AS THE VARIABLE WITH TRAILING BLANKS        */
 /*      REMOVED. A CALL TO QCMDEXC USING APOSTROPHES TO DELIMIT      */
 /*      THE LENGTH OF THE KEY VALUE CAN BE USED TO CIRCUMVENT        */
 /*      THIS ACTION.                                                 */
 /*    PARAMETERS--                                                   */
            DCL        VAR(&KEYVAL) TYPE(*CHAR) LEN(32)  /*  THE VALUE +
                         OF THE REQUESTED KEY. NOTE IT IS DEFINED AS +
                         32 CHAR.  */
            DCL        VAR(&LEN) TYPE(*INT)              /*  THE LENGTH +
                         OF THE KEY VALUE TO BE USED. ANY VALUE OF  +
                         1 TO 32 CAN BE USED  */
 /*   THE STRING TO BE FINISHED FOR THE OVERRIDE COMMAND TO BE        */
 /*        PASSED TO QCMDEXC (NOTE 2 APOSTROPHES TO GET ONE).         */
            DCL        VAR(&STRING) TYPE(*CHAR) LEN(100) +
                 VALUE('OVRDBF FILE(X3) POSITION(*KEY 1 FMT1  '' ')
 /*   POSITION MARKER  123456789 123456789 123456789 123456789       */
            DCL        VAR(&END) TYPE(*DEC) LEN(15 5)  /*  A VARIABLE +
                         TO CALCULATE THE END OF THE KEY IN &STRING  */

            CHGVAR     VAR(%SST(&STRING 40 &LEN)) VALUE(&KEYVAL)  /*  +
                         PUT THE KEY VALUE INTO COMMAND STRING FOR +
                         QCMDEXC IMMEDIATELY AFTER THE APOSTROPHE.  */
            CHGVAR     VAR(&END) VALUE(&LEN + 40)  /*  POSITION AFTER +
                         LAST CHARACTER OF KEY VALUE  */
            CHGVAR     VAR(%SST(&STRING &END 2)) VALUE(''')')  /*  PUT +
                         A CLOSING APOSTROPHE & PAREN TO END  +
                         PARAMETER  */
            CALL       PGM(QCMDEXC) PARM(&STRING 100)  /*  CALL TO +
                         PROCESS THE COMMAND   */
            ENDPGM
```

注: VARY (*YES) および RTNVAL (*YES) を使用し、CL 変数を渡している場合、CL 変数のデータの長さではなく、変数の長さが渡されます。

関連情報:

データベース・ファイル指定変更 (OVRDBF) コマンド

CL プログラムおよびプロシージャーへの注記の書き込み

CL プログラムまたはプロシージャーに、注記を入れたり、CL プログラムまたはプロシージャー中のコマンドに注記を追加したりするには、/* と */ とを一対にして使用してください。 この 2 つの記号の間に注記を入れてください。

注記開始区切り文字である /* は 3 の長さを必要とします。ただし、/* がコマンド・ストリングの最初の 2 文字である場合、/* の後にブランクを入れる必要はありません。

3 文字の注記開始区切り文字は、次のいずれかの形式で入力することができます (b はブランクを表しています)。

```
/*b
b/*
/**
```

このため、注記開始区切り文字は、異なる方法で入力することができます。注記開始区切り文字である /* の入力には、次の 4 通りが考えられます。

- コマンド・ストリングの最初の 2 文字で使用する。
- 前にブランクを 1 つ付ける。
- 後にブランクを 1 つ付ける。
- 後にアスタリスクを 1 つ付ける (/**)

注: 注記の中に注記を組み込むことはできません。

次のプロシージャーの例では、注記はメニューからユーザーが選択できるオプションを簡潔に説明するために入れられています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
        PGM        /* ORD040C ORDER DEPT GENERAL MENU */
        DCLF       FILE(ORD040CD)
START:  SNDRCVF    RCDFMT(MENU)
        SELECT
          WHEN (&RESP=1) THEN(CALL CUS210)   /* CUSTOMER INQUIRY     */
          WHEN (&RESP=2) THEN(CALL ITM210)   /* ITEM INQUIRY         */
          WHEN (&RESP=3) THEN(CALL CUS210)   /* CUSTOMER NAME SEARCH */
          WHEN (&RESP=4) THEN(CALL ORD215)   /* ORDERS BY CUST       */
          WHEN (&RESP=5) THEN(CALL ORD220)   /* EXISTING ORDER       */
          WHEN (&RESP=6) THEN(CALL ORD410C)  /* ORDER ENTRY          */
          WHEN (&RESP=7) THEN(RETURN)
        ENDSELECT
        GOTO START
        ENDPGM
```

CL プログラムまたは CL プロシージャー内での処理の制御

コマンドを使用して、CL プロシージャー内のロジックの流れを変更することができます。

CL プロシージャー内のコマンドは順次処理されます。すなわち各コマンドは、プログラムの中に現れる順序で 1 つずつ処理されていきます。このような順次処理の流れは、プロシージャーのロジックの流れを変えるコマンドを使用して変更することができます。これらのコマンドは、条件付きまたは無条件にできます。

無条件分岐とは、分岐命令を出した時点での条件には関係なく、プロシージャー内の他の場所にあるコマンド (または 1 組のコマンド) へ分岐できることを意味します。無条件処理コマンドには以下のものがあります。

- GOTO
- ITERATE
- LEAVE
- CALLSUBR

条件付き分岐とは、指定された特定の条件が生じている場合に限り、プロシージャー内の別の場所にあるセクションまたはコマンドに分岐することを意味します。プロシージャー内のどのようなステートメントにも分岐することができます。これは、指定された条件が真である場合に限り分岐が行われるので、条件付き処理と呼ばれます。条件付き処理には、通常、IF コマンドが使用されます。また、ELSE コマンドを使用すれば、条件が真でない場合の代替処理を指定することができます。単純な DO コマンドを使用すれば、複数のコマンドを 1 つのグループにして、指定した条件が生じた場合にそれらのコマンドが常にグループとして処理されるようにすることができます。条件付き処理コマンドには以下のものがあります。

- IF および THEN
- SELECT、WHEN、および OTHERWISE
- DOFOR
- DOWHILE
- DOUNTIL

関連概念:

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

関連情報:

CL コマンド検索プログラム

CL プログラムまたはプロシージャー内の GOTO コマンドおよびコマンド・ラベル

GOTO (GOTO) コマンドは無条件分岐を指定するためのものです。

プロシージャーで **GOTO** コマンドを検出すると、その時点でプログラムまたはプロシージャー内の別の部分 (ラベルで指定された個所) に処理が移ります。この分岐は、何らかの式の評価の結果として生じるものではありません。指定されたラベルを持つステートメントへの分岐の後は、そのステートメントから始まってそれに続くステートメントへと順次処理が進められます。別の命令によって特に戻りを指定しない限り、再び **GOTO** コマンドの位置に戻ることはありません。この分岐は、正方向にも逆方向にも行うことができます。 **GOTO** コマンドを使用して、プロシージャー外のラベルにジャンプすることはできません。また、**GOTO** コマンドを使用して、プログラムまたはプロシージャー内に定義されているサブルーチンに分岐したり、サブルーチンから分岐することはできません。**GOTO** コマンドにはパラメーターが 1 つあり、このパラメーターにはどのステートメントに分岐をするかをラベルで指定します。

```
GOTO CMDLBL(ラベル)
```

ラベルは、**GOTO** コマンドによりプログラムまたはプロシージャー内のどのステートメントに処理を移すかを示します。したがって、**GOTO** コマンドを使用する場合には、分岐先のコマンドにラベルが付けられていなければなりません。

次の例ではラベルは START です。ラベルとして使用できる文字数は 10 文字で、その後にコロン (:) を付けなければなりません。ラベルとコマンド名との間にブランクがあっても構いません。

```
          PGM
          .
          .
          .
START:    SNDRCVF RCDFMT(MENU)
          IF (&RESP=1) THEN(CALL CUS210)
          .
          .
          .
          GOTO START
          .
          .
          .
          ENDPGM
```

関連情報:

CL コマンド検索プログラム

GOTO (GOTO) コマンド

CL プログラムまたはプロシージャー内の IF コマンド

IF (IF) コマンドは、条件を指定する場合に使用され、その条件が真の場合は、プログラムまたはプロシージャー内の実行するステートメントやステートメントのグループを指定します。

IF コマンドとともに **ELSE (ELSE)** コマンドを使用すれば、**IF** コマンドで指定した条件が偽であった場合に実行するステートメントやステートメントのグループを指定できます。

IF コマンドには、真偽をテストする式と、その式が真であった場合にとるべき処置を指定する THEN パラメーターが含まれます。すなわち、**IF** コマンドの形式は次のとおりです。

```
IF COND(論理式) THEN(CL コマンド)
```

COND パラメーターに指定する論理式は 1 つの論理変数または定数であるか、または 2 つ以上のオペランドの関係を示すものでなければなりません。この式について真か偽かが評価されます。

論理式により指定された条件が真であると評価された場合には、プログラムまたはプロシージャーは THEN パラメーターに指定されている CL コマンドを処理します。処理されるコマンドは、1 つの場合もあり、コマンドのグループである場合もあります。条件が真でなければ、プログラムまたはプロシージャーは次の順番のコマンドを実行します。

COND と THEN はどちらもこのコマンドのキーワードであり、定位置入力の場合には省略することができます。次の各例は、どちらもこのコマンドの構文的に正しい用例です。

```
IF COND(&RESP=1) THEN(CALL CUS210)
IF (&A *EQ &B) THEN(GOTO LABEL)
IF (&A=&B) GOTO LABEL
```

コマンド名 (IF) と、キーワード (COND) または値 (&A) の間にはブランクが必要です。キーワード (使用する場合) と、値を囲む左括弧との間には、ブランクを入れることはできません。

以下は、**IF** コマンドを使用した条件付き処理の例です。**IF** コマンドの条件を指定する論理式の評価の結果によって、分岐先が異なります。例えば、次のプログラムの開始時点で &A の値が 2 であり、 &C の値が 4 であったとします。

```
       IF (&A=2) THEN(GOTO FINAL)
       IF (&A=3) THEN(CHGVAR &C 5)
       .
       .
       .
FINAL: IF (&C=5) CALL PROGA
       ENDPGM
```

この場合、プログラムまたはプロシージャーは最初の **IF** コマンドを処理した後で、途中のコードを飛ばして FINAL に分岐します。 2 番目の **IF** コマンドには戻りません。 FINAL では、&C=5 のテストの結果は真にならないので、PROGA は呼び出されません。したがって、プログラムまたはプロシージャーはその次のコマンドである ENDPGM を処理します。このコマンドはプログラムまたはプロシージャーの終わりを示すものであり、これにより呼び出しプログラムまたはプロシージャーに制御権が戻されます。

これと同じプログラムを使用しても、変数の初期値が異なると処理のロジックが変わります。例えば、このプログラムの開始時点で &A の値が 3 であり、 &C の値が 4 であったとすれば、最初の IF ステートメントの評価の結果は偽となります。したがってプログラムまたはプロシージャーは、GOTO FINAL コマンドを処理せずに、最初の IF ステートメントを無視してその次のステートメントの処理に移ります。 2 番目の IF ステートメントの評価の結果は真となるので、 &C の値は 5 に変更されます。そしてその後に続くステートメント (例では示されていません) が順番に処理されます。処理が最後の IF ステートメントに達した時、&C=5 の条件の評価が真となるので、PROGA が呼び出されます。

次のような一連の連続した IF ステートメントがある場合、各 IF ステートメントはそれぞれ独立して実行されます。次の例をご覧ください。

```
      PGM /*  IFFY  */
      DCL &A..
      DCL &B..
      DCL &C..
      DCL &D..
      DCL &AREA *CHAR LEN(5) VALUE(YESNO)
      DCL &RESP..
      IF (&A=&B) THEN(GOTO END)   /*  IF #1  */
      IF (&C=&D) THEN(CALL PGMA)   /*  IF #2  */
      IF (&RESP=1) THEN(CHGVAR &C 2)  /*  IF #3  */
      IF (%SUBSTRING(&AREA 1 3) *EQ YES) THEN(CALL PGMB)  /*  IF #4  */
      CHGVAR &B &C
      .
      .
      .
END:  ENDPGM
```

この例で &A が &B に等しくない場合、次のステートメントが実行されます。 &C が &D に等しければ、PGMA が呼び出されます。 PGMA から制御権が戻ると、3 番目の IF ステートメントが評価されます (以降このようにして処理が続けられます)。組み込みコマンドとは、他のコマンドのパラメーターの値としてその全体が含まれているコマンドです。

例えば、次の例では**変数変更 (CHGVAR)** コマンドおよび DO コマンドが組み込みコマンドです。

```
IF (&A *EQ &B) THEN(CHGVAR &A (&A+1))


IF (&B *EQ &C) THEN(DO)
```

```
.
.
.
ENDDO
```

関連資料:

221 ページの『*AND、*OR、および *NOT 演算子』
論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

212 ページの『CL プログラムまたはプロシージャー内の DO コマンドおよび DO グループ』
DO グループ (DO) コマンドは、複数のコマンドをグループ化し、グループとして処理するためのものです。

『CL プログラムまたはプロシージャー内の ELSE コマンド』
ELSE (ELSE) コマンドは、対応する **IF (IF)** コマンドで指定した条件が偽であった場合に、代わりの処理を指定するための手段です。

210 ページの『CL プログラムまたはプロシージャー内の組み込み IF コマンド』
IF (IF) コマンドは他の **IF** コマンドに組み込むことができます。

関連情報:

CL コマンド検索プログラム

仮定 (IF) コマンド

CL プログラムまたはプロシージャー内の ELSE コマンド

ELSE (ELSE) コマンドは、対応する **IF (IF)** コマンドで指定した条件が偽であった場合に、代わりの処理を指定するための手段です。

対応する **ELSE** コマンドのない IF コマンドを使用することもできます。

```
IF (&A=&B) THEN(CALLPRC PROCA)
CALLPRC PROCB
```

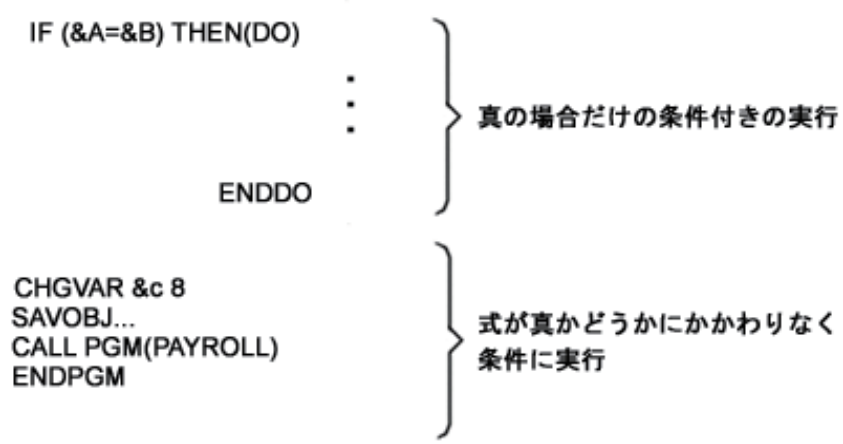
この例の場合、PROCA が呼び出されるのは &A=&B の場合だけですが、PROCB は常に呼び出されます。

ただし、このプロシージャーで **ELSE** コマンドを使用した場合には、処理のロジックが変わります。次の例では、&A=&B であれば PROCA が呼び出されますが、PROCB は呼び出されません。&A=&B の式が真でなければ、PROCB が呼び出されます。

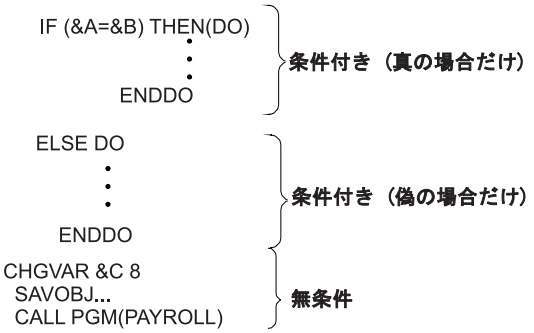
```
IF (&A=&B) THEN(CALLPRC PROCA)
ELSE CMD(CALLPRC PROCB)
CHGVAR &C 8
```

IF 式の評価の結果が偽となった場合に別の分岐 (すなわち二者択一的な処理) を行いたい場合には、**ELSE** コマンドを使用しなければなりません。

ELSE コマンドは、DO グループと組み合わせて使用した場合に実際に役立ちます。次の例では、IF 式の評価の結果によっては DO グループは実行されていないこともありますが、その他のコマンドは必ず処理されます。



ELSE コマンドを使用すれば、式が真でなかった場合に限り処理されるコマンド (またはコマンドのグループ) を指定して、二者択一的なロジックを設定することができます。



各 **ELSE** コマンドについて、その前にそれぞれ対応する IF コマンドがなければなりません。 IF コマンドのネストが存在する場合には、各 **ELSE** コマンドは、まだ他の **ELSE** コマンドと対になっていない最も内側の **IF** コマンドと対になります。

```
IF ... THEN ...
IF ...THEN(DO)
         IF ...THEN(DO)
                    .
                    .
                    .
         ENDDO
         ELSE DO
                    IF ...THEN(DO)
                        .
                        .
                        .
                        ENDDO
                    ELSE DO
                        .
                        .
                        .
                        ENDDO
         ENDDO
ELSE IF ... THEN ...
IF ... THEN ...
IF ... THEN ...
```

プロシージャー内の IF コマンドと **ELSE** コマンドとの対応を確認する際には、必ず最も内側の対応関係から調べていくようにしてください。

ELSE コマンドは、一連の二者択一的なオプションをテストすることができます。次の例では、最初に IF テストを通った後に、その埋め込みコマンドが処理され、次に**資源再利用 (RCLRSC)** コマンドが処理されます。

```
IF  COND(&OPTION=1) THEN(CALLPRC PRC(ADDREC))
 ELSE    CMD(IF COND(&OPTION=2) THEN(CALLPRC PRC(DSPFILE)))
  ELSE   CMD(IF COND(&OPTION=3) THEN(CALLPRC PRC(PRINTFILE)))
    ELSE CMD(IF COND(&OPTION=4) THEN(CALLPRC PRC(DUMP)))
RCLRSC
RETURN
```

関連資料:

206 ページの『CL プログラムまたはプロシージャー内の IF コマンド』
IF (IF) コマンドは、条件を指定する場合に使用され、その条件が真の場合は、プログラムまたはプロシージャー内の実行するステートメントやステートメントのグループを指定します。

221 ページの『*AND、*OR、および *NOT 演算子』
論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

関連情報:

CL コマンド検索プログラム

ELSE (ELSE) コマンド

CL プログラムまたはプロシージャー内の組み込み IF コマンド

IF (IF) コマンドは他の **IF** コマンドに組み込むことができます。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

IF コマンドが組み込まれるのは、評価の結果が真である場合に処理されるコマンド (THEN パラメーターに指定される CL コマンド) 自体が別の **IF** コマンドである場合です。

```
IF (&A=&B) THEN(IF (&C=&D) THEN(GOTO END))
GOTO START
```

この方法は、複数の条件が満たされた場合に限り、特定のコマンドまたはコマンドのグループを実行する際に使用すると便利です。上記の例では、最初の式が真であれば、システムは最初の THEN パラメーターを調べ、その結果 &C=&D の式が真であると評価されると、2 番目の THEN パラメーターに指定されているコマンド (GOTO END) を処理します。したがって、**GOTO END** コマンドが処理されるためには両方の式が真でなければなりません。いずれか一方が偽であれば、GOTO START コマンドが実行されます。式とコマンドを編成するための括弧の用法に注意してください。

CL プログラミングでは、このような組み込みが最高 25 レベルまで可能です。

組み込みのレベルが多くなるほどロジックも複雑になるので、相互の関係を明確にするために、次の例のように、自由な形式でコマンドを入力することができます。

```
PGM
DCL &A *DEC 1
DCL &B *CHAR 2
DCL &RESP *DEC 1
IF (&RESP=1) +
    IF (&A=5) +
```

```
         IF (&B=NO) THEN(DO)
                     .
                     .
                     .
                     ENDDO
CHGVAR &A VALUE(8)
CALL PGM(DAILY)
ENDPGM
```

上記の一連の IF コマンドは、それぞれ 1 つの組み込みコマンドとして処理されます。 IF 条件のいずれかが偽として評価されると、処理はプログラムの残りの部分 (**変数変更 (CHGVAR)** および後続のコマンド) に分岐します。この例の目的が、DO グループを処理するために真でなければならないすべての条件を集めることである場合には、複数の式を *AND を用いて 1 つのコマンドに指定することにより、コーディングを簡素化することができます。

ただし場合によっては、どの条件が偽であるかによって分岐先を変えなければならないこともあります。これは、個々の組み込み **IF** コマンドに対応する ELSE コマンドを追加することによって行うことができます。

```
PGM
DCL &A ...
DCL &B ...
DCL &RESP ...
IF (&RESP=1) +
      IF (&A=5) +
            IF (&B=NO) THEN(DO)
                        .
                        .
                        .
                        SNDPGMMSG ...
                        .
                        .
                        .
                       ENDDO
            ELSE CALLPRC PROCA
      ELSE CALLPRC PROCB
CHGVAR &A 8
CALLPRC PROC(DAILY)
ENDPGM
```

ここで、すべての条件が真の場合は、**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドが処理され、続いて**変数変更 (CHGVAR)** コマンドが処理されます。最初および 2 番目の条件 (&RESP=1 および &A=5) が真であり、3 番目 (&B=NO) が偽であれば、PROCA が呼び出されます。そして、PROCA から戻った時点で CHGVAR コマンドが処理されます。2 番目の条件が満たされなかった場合には、PROCB が呼び出され (&B=NO はテストされません)、次に CHGVAR コマンドが処理されます。最後に、&RESP が 1 でなければ、CHGVAR コマンドが直ちに処理されます。 ELSE コマンドは、各テストごとに異なる分岐を行うために使用されています。

注: 次の 3 つの例は、どれも構文的に正しく、しかも上記の例の組み込み **IF** コマンドと同じ意味を持っています。

```
IF (&RESP=1) THEN(IF (&A=5) THEN(IF (&B=NO) THEN(DO)))

IF (&RESP=1) THEN +
           (IF (&A=5) THEN +
                       (IF (&B=NO) THEN(DO)))

IF (&RESP=1) +
        (IF (&A=5) +
              (IF (&B=NO) THEN(DO)))
```

関連資料:

206 ページの『CL プログラムまたはプロシージャー内の IF コマンド』
IF (IF) コマンドは、条件を指定する場合に使用され、その条件が真の場合は、プログラムまたはプロシージャー内の実行するステートメントやステートメントのグループを指定します。

221 ページの『*AND、*OR、および *NOT 演算子』
論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

関連情報:

CL コマンド検索プログラム

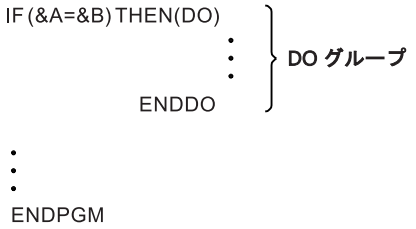
仮定 (IF) コマンド

CL プログラムまたはプロシージャー内の DO コマンドおよび DO グループ

DO グループ (DO) コマンドは、複数のコマンドをグループ化し、グループとして処理するためのものです。

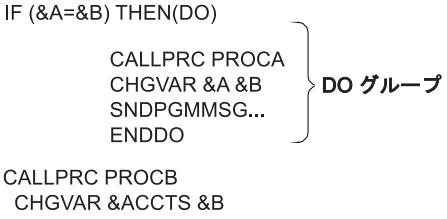
DO コマンドとそれに対応する **DO グループ終了 (ENDDO)** コマンドとの間のすべてのコマンドが、1 つのグループとして定義されます。

グループの処理は、通常関連するコマンドの評価の結果に基づいて条件付きで行われます。 DO グループは、**IF**、**ELSE**、または **MONMSG** コマンドと関連付けられるのが普通です。これは、DO グループの例です。



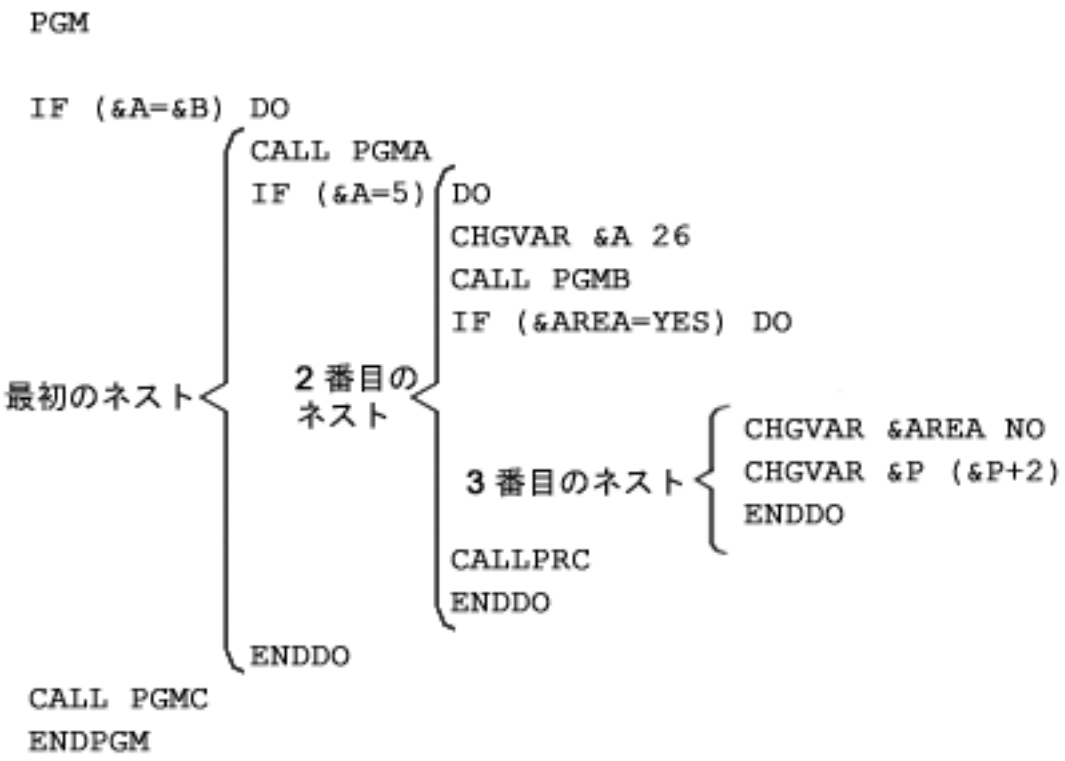
論理式 (&A=&B) が真であれば、DO グループが処理されます。この式が真でなければ、DO グループを飛ばして、**ENDDO** コマンドの次から処理が行われます。

次のプロシージャーでは、&A が &B に等しくなければ、システムは PROCB を呼び出します。したがって、PROCA は呼び出されず、DO グループ内のその他のコマンドも処理されません。



DO グループは他の DO グループ中にネストすることもできます。その場合のネストのレベルは最高 25 までです。

次の例では、3 つのレベルのネストがあります。各 DO グループがそれぞれ **ENDDO** コマンドによって完結している点に注意してください。



この例で、第 1 ネストの中の &A が 5 に等しくなければ、PGMC が呼び出されます。 &A が 5 に等しい場合には、2 番目の DO グループの中のステートメントが処理されます。 2 番目の DO グループの中の &AREA が YES でなければ、処理はその DO グループ内の次のコマンドに移るので、プロシージャー ACCTSPAY が呼び出されます。

CL コンパイラーは DO グループの始めも終わりも示しません。 CL コンパイラーが始めと終わりが対応していない条件を検出した場合、実際のエラーを見つけるのは簡単ではありません。

関連資料:

206 ページの『CL プログラムまたはプロシージャー内の IF コマンド』

IF (IF) コマンドは、条件を指定する場合に使用され、その条件が真の場合は、プログラムまたはプロシージャー内の実行するステートメントやステートメントのグループを指定します。

関連情報:

CL コマンド検索プログラム

Do グループ (DO) コマンド

DO および SELECT のネスト・レベルの表示

すべてのタイプの **DO** コマンドと **SELECT** コマンドのネスト・レベルを表示する CL コンパイラー・オプションを指定することができます。

CL ソース・プログラムには、任意の数のレベルにネストされた **DO** コマンドまたは **SELECT** コマンドが含まれることがあります。例えば、**DO** コマンドと対応する **ENDDO** コマンドの間に、**DOFOR** と別の **ENDDO** コマンドが入れることができます。 CL コンパイラーでは、 **DO** コマンドと **SELECT** コマンドについて、最大 25 レベルのネストをサポートしています。

DO コマンドまたは **SELECT** コマンドのネスト・レベルが高い CL ソース・コードを変更する作業は、難しくなります。それぞれのタイプの **DO** コマンドを該当の **ENDDO** コマンドと正しく対応させなければなりません。また、それぞれの **SELECT** コマンドを該当の **ENDSELECT** コマンドと対応させなければなりません。

CL コンパイラーには、**Do (DO)**、**Do For (DOFOR)**、**Do Until (DOUNTIL)**、**Do While (DOWHILE)**、および **Select (SELECT)** コマンドのネスト・レベルを CL コンパイラー・リストに表示するオプションがあります。コンパイラー・リストにネスト・レベルを表示するには、 **制御言語プログラム作成 (CRTCLPGM)** コマンド、 **制御言語モジュール作成 (CRTCLMOD)** コマンド、または**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドの OPTION パラメーターに *DOSLTLVL を指定します。このネスト・レベル情報を表示しない場合には、 OPTION パラメーターに *NODOSLTLVL を指定することができます。

関連情報:

制御言語プログラム作成 (CRTCLPGM) コマンド

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

CL プログラムまたはプロシージャー内の DOUNTIL コマンド

Do Until (DOUNTIL) コマンドは、CL コマンドのグループを 1 回以上処理します。

コマンドのグループは、**DOUNTIL** コマンドとそれに対応する **DO グループ終了 (ENDDO)** コマンドの間にあるコマンドとして定義されます。

コマンドのグループが処理された後、宣言されている条件が評価されます。条件が真であれば、DOUNTIL グループは終了し、関連する ENDDO に続くコマンドの処理が再開します。条件が偽であれば、グループ内の最初のコマンドから処理が継続します。

COND パラメーターに指定する論理式は 1 つの論理変数または定数であるか、または 2 つ以上のオペランドの関係を示すものでなければなりません。この式について真か偽かが評価されます。

以下は、**DOUNTIL** コマンドを使用した条件付き処理の例です。

```
DOUNTIL (&LGL)
  .
  .
  .
  CHGVAR &INT (&INT + 1)
  IF (&INT *GT 5) (CHGVAR &LGL '1')
ENDDO
```

DOUNTIL グループの本体は、最低でも 1 回実行されます。 &INT 変数の初期値が 5 以上であれば、&LGL は最初から真に設定され、グループの終わりの式が評価されたときに ENDDO に続く処理が行われます。初期値が 5 未満であれば、グループの本体は、&INT が 5 より大きくなって &LGL が真に変わるまで、繰り返されます。

Leave (LEAVE) コマンドを使用すると、DOUNTIL グループを終了して ENDDO の後の処理から再開することができます。 ITERATE コマンドを使用すれば、グループ内の残りのコマンドをスキップして、宣言されている条件を即時に評価できます。

関連資料:

221 ページの『*AND、*OR、および *NOT 演算子』
論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

関連情報:

CL コマンド検索プログラム

Do Until (DOUNTIL) コマンド

CL プログラムまたはプロシージャー内の DOWHILE コマンド

Do While (DOWHILE) コマンドは、複数のコマンドをグループ化し、論理式の値が真の間にゼロ回またはそれ以上処理するためのものです。

DOWHILE コマンドは、条件を指定する場合に使用され、その条件が真の場合は、プログラムまたはプロシージャー内の実行するコマンドやコマンドのグループを指定します。コマンドのグループは、**DOWHILE** コマンドとそれに対応する **DO グループ終了 (ENDDO)** コマンドの間にあるコマンドとして定義されます。

コマンドのグループが処理される前に、宣言されている条件が評価されます。宣言されている条件が最初に偽であれば、コマンドのグループは全く処理されません。条件が偽であれば、DOWHILE グループは終了し、関連する ENDDO に続くコマンドの処理が再開します。条件が真であれば、グループ内の最初のコマンドから処理が継続します。 ENDDO コマンドに達すると、制御は DOWHILE コマンドに戻り、再び条件が評価されます。

COND パラメーターに指定する論理式は 1 つの論理変数または定数であるか、または 2 つ以上のオペランドの関係を示すものでなければなりません。この式について真か偽かが評価されます。

以下は、**DOWHILE** コマンドを使用した条件付き処理の例です。

```
DOWHILE (&LGL)
  .
  .
  .
  IF (&INT *EQ 2) (CHGVAR &LGL '0')
ENDDO
```

DOWHILE グループが処理されると、宣言されている条件が評価されます。条件が真であれば、DOWHILE グループ内のコマンドのグループが処理されます。条件が偽であれば、関連する ENDDO コマンドに続くコマンドの処理が継続します。

&LGL の値が真であれば、DOWHILE グループ内のコマンドは、 &INT が 2 になって &LGL 変数値が偽に設定されるまで、実行されます。

Leave (LEAVE) コマンドを使用すると、 DOWHILE グループを終了して ENDDO の後の処理から再開することができます。 ITERATE コマンドを使用すれば、グループ内の残りのコマンドをスキップして、宣言されている条件を即時に評価できます。

関連資料:

221 ページの『*AND、*OR、および *NOT 演算子』

論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

関連情報:

CL コマンド検索プログラム

Do While (DOWHILE) コマンド

CL プログラムまたはプロシージャー内の DOFOR コマンド

Do For (DOFOR) コマンドは、複数のコマンドをグループ化し、指定した回数処理するためのものです。

DOFOR コマンドでは、変数、その初期値、増分または減少の量、および終了値条件を指定します。 **DOFOR** コマンドの形式は次のとおりです。

```
DOFOR VAR(integer-variable) FROM(initial-value) TO(end-value) BY(integer-constant)
```

DOFOR グループの処理が始まると、VAR パラメーターで指定した integer-variable が FROM パラメーターで指定した initial-value に初期化されます。 integer-variable の値は TO パラメーターで指定した end-value と比較されます。 BY パラメーターの integer-constant が正であれば、 integer-variable が end-value より大きいかどうかが比較されます。 BY パラメーターの integer-constant が負であれば、integer-variable が end-value より小さいかどうかが比較されます。

条件が真でなければ、DOFOR グループの本体が処理されます。 ENDDO に達すると、BY パラメーターの integer-constant が integer-value に追加され、条件が再び評価されます。

以下は、**DOFOR** コマンドを使用した条件付き処理の例です。

```
CHGVAR &INT2 0
DOFOR VAR(&INT) FROM(2) TO(4) BY(1)
  .
  .
  .
  CHGVAR &INT2 (&INT2 + &INT)
ENDDO
/* &INT2 = 9 after running the DOFOR group 3 times */
```

DOFOR グループが処理されると、&INT が 2 に初期化され、&INT の値が 4 より大きいかどうかが検査されます。そうでなければ、グループの本体が処理されます。グループの 2 回目の反復では、1 が &INT に加えられ、検査が繰り返されます。これは 4 より小さいので、DOFOR グループが再び処理されます。 ENDDO に 2 度目に達すると、 &INT は再び 1 つ増分されます。このとき &INT の値は 4 になります。 &INT は依然として 4 以下なので、DOFOR グループが再び処理されます。 ENDDO に 3 度目に達すると、 &INT は再び 1 つ増分されます。今回の値は 5 なので、 ENDDO に続くコマンドの処理が継続します。

LEAVE コマンドを使用すれば、DOFOR グループを終了して ENDDO に続く処理を再開できます。ITERATE コマンドを使用すれば、グループ内の残りのコマンドをスキップし、制御変数を増分して、end-value 条件を即時に評価できます。

関連情報:

CL コマンド検索プログラム

Do For (DOFOR) コマンド

CL プログラムまたはプロシージャー内の ITERATE コマンド

繰り返し (ITERATE) コマンドを使用すると、活動状態にある DOWHILE、DOUNTIL、または DOFOR グループ内の残りのコマンドをスキップできます。

ITERATE は単純な DO コマンド・グループでは無効です。

ラベルのない **ITERATE** コマンドは、最も内側の活動状態の DO グループの ENDDO にスキップします。ラベルを指定すると、そのラベルに関連した DO の ENDDO にスキップします。

次の例は、**ITERATE** コマンドの使い方を示しています。

```
DO_1:
DO_2:DOWHILE &LGL
DO_3:  DOFOR &INT FROM(0) TO(99)
         .
```

```
    .
    .
    IF (&A *EQ 12) THEN (ITERATE DO_1)
    .
    . /* Not processed if &A equals 12        */
    .
    IF (&A *GT 12) ITERATE
    .
    .  /* Not processed if &A greater than 12 */
    .
  ENDDO
    .
    .
    .
  IF (&A *LT 0) (ITERATE DO_1)
    .
    . /* Not processed if &A less than zero   */
    .
ENDDO
```

この例では、ラベル DO_1 および DO_2 が DOWHILE グループに関連付けられています。これらのラベルは、DOWHILE または DOFOR グループに現れる **ITERATE** コマンドで指定できます。 &A が 12 の場合は、ITERATE DO_1 コマンドが実行されます。処理は DOWHILE コマンドに関連した ENDDO から継続します。 &LGL の値が評価され、真であれば、DOWHILE に続く DOFOR から処理が継続します。 &LGL が偽であれば、2 番目の ENDDO に続く CL コマンドから処理が継続します。

&A が 12 でなく、12 より大きい場合は、DOFOR グループの ENDDO から処理が継続します。 &INT の値が増分され、終了値の 99 と比較されます。 &INT が 99 以下の場合は、**Do For (DOFOR)** コマンドに続く最初のコマンドから処理が継続します。 &INT が 99 より大きい場合は、最初の ENDDO に続くコマンドから処理が継続します。

3 番目の IF コマンドが処理され、&A がゼロより小さい場合は、2 番目の ENDDO から処理が継続します。 &LGL の値が評価され、偽であれば、ENDDO に続くコマンドに制御が渡されます。真の場合は、DOWHILE に続く **Do For (DOFOR)** コマンドから処理が再開します。

関連情報:

CL コマンド検索プログラム

繰り返し (ITERATE) コマンド

CL プログラムまたはプロシージャー内の LEAVE コマンド

Leave (LEAVE) コマンドを使用すると、活動状態にある DOWHILE、DOUNTIL、または DOFOR グループを終了することができます。

このコマンドは、**GOTO (GOTO)** コマンドを使用せずに活動状態のグループから抜け出す構造化された手段を提供します。 LEAVE は単純な **DO (DO)** コマンド・グループでは無効です。

ラベルのない **LEAVE** コマンドは、最も内側の活動状態の DO グループから抜け出します。ラベルを指定すると、1 つ以上の囲まれたグループから抜け出すことができます。

次の例は、**LEAVE** コマンドの使い方を示しています。

```
DO_1:
DO_2:DOWHILE &LGL
DO_3:  DOFOR &INT FROM(0) TO(99)
         .
         .
         .
         IF (&A *EQ 12) THEN(LEAVE DO_1)
```

```
    .
    . /* Not processed if &A equals 12        */
    .
    IF (&A *GT 12) LEAVE
    .
    .  /* Not processed if &A greater than 12 */
    .
  ENDDO
    .
    .
    .
  IF (&A *LT 0) (LEAVE DO_1)
    .
    . /* Not processed if &A less than zero   */
    .
ENDDO
```

この例では、ラベル DO_1 および DO_2 が DOWHILE グループに関連付けられています。これらのラベルは、DOWHILE または DOFOR グループに現れる LEAVE コマンドで指定できます。 &A が 12 の場合は、LEAVE DO_1 コマンドが実行され、 2 番目の ENDDO に続く CL コマンドから処理が継続します。

&A が 12 でなく、12 より大きい場合、DOFOR グループは終了され、最初の ENDDO に続くコマンドから処理が継続します。

3 番目の **If (IF)** コマンドが処理され、&A がゼロより小さい場合は、最初の ENDDO の後の次のコマンドから処理が継続します。

関連情報:

CL コマンド検索プログラム

Leave (LEAVE) コマンド

CL プログラムまたはプロシージャーの CALLSUBR コマンド

サブルーチン呼び出し (CALLSUBR) コマンドは、同じプログラムまたはプロシージャー内で定義されたサブルーチンに制御を渡すために CL プログラムまたは CL プロシージャーで使用されます。

CALLSUBR コマンドには、**サブルーチン (SUBR)** (制御が転送されるサブルーチンの名前を含む) と**戻り値 (RTNVAL)** (呼ばれたサブルーチンからの戻り値を含む変数を指定する) の 2 つのパラメーターがあります。 以下の例を参照してください。

```
CALLSUBR SUBR(mysubr) RTNVAL(&myrtnvar)
```

サブルーチン mysubr は、**SUBR** コマンドのサブルーチン (SUBR) パラメーターによってプログラムまたはプロシージャー内で定義する必要があります。変数 *&myrtnvar* は TYPE(*INT) LEN(4) として定義し、サブルーチン mysubr 内にある **サブルーチンからの戻り (RTNSUBR)** または **サブルーチン終了 (ENDSUBR)** コマンドのいずれかの戻り値 (RTNVAL) パラメーターからの値を含むようにする必要があります。RTNVAL パラメーターが定義されていない場合、サブルーチンからの戻り値は無視されます。

CALLSUBR コマンドは、他のサブルーチンを含む、プログラムまたはプロシージャー内の任意の場所に配置することができます。ただし、プログラム・レベルの**メッセージ・モニター (MONMSG)** コマンドは例外です。各 **CALLSUBR** コマンドは、実行時に、戻りアドレスをサブルーチン・スタックに配置し、そのスタックのサイズは、**処理オプションの宣言 (DCLPRCOPT)** コマンドのサブルーチン・スタック (SUBRSTACK) パラメーターを使用して変更することができます。グローバルにモニターされるメッセージによって GOTO コマンドが実行されるような場合には、サブルーチン・スタックは、実行されている次の **CALLSUBR** コマンドによってリセットされます。

次の例では、最初の **CALLSUBR** コマンドが制御をサブルーチン SUBR1 に渡し、制御が戻ったときに戻り値の 12 が変数 &myrtnvar に置かれます。 メッセージが **MONMSG** コマンドによってモニターされる場合、**Goto (GOTO)** コマンドが実行され、制御はラベル DUMP に分岐します。 CL プログラムまたはプロシージャーは、**DMPCLPGM** コマンドによってダンプされ、サブルーチン SUBR2 への次の **CALLSUBR** コマンドがサブルーチン・スタックをリセットします。

```
            PGM
            DCL         VAR(&myrtnvar) TYPE(*INT) LEN(4)
            MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(DUMP))
            :
            CALLSUBR    SUBR(SUBR1) RTNVAL(&myrtnvar)
            :
DUMP:       DMPCLPGM
            CALLSUBR    SUBR(SUBR2)
            :
            SUBR        SUBR(SUBR1)
            :
            ENDSUBR     RTNVAL(12)
            :
            SUBR        SUBR(SUBR2)
            :
            ENDSUBR
            ENDPGM
```

関連資料:

220 ページの『CL プログラムまたはプロシージャー内の SUBR コマンドおよびサブルーチン』

サブルーチン (SUBR) コマンドは、**サブルーチン終了 (ENDSUBR)** コマンドとともに CL プログラムまたは CL プロシージャーで使用され、サブルーチンを定義するコマンドのグループを区切ります。

関連情報:

処理オプション宣言 (DCLPRCOPT) コマンド

CL コマンド検索プログラム

サブルーチン呼び出し (CALLSUBR) コマンド

CL プログラムまたはプロシージャー内の SELECT コマンドおよび SELECT グループ

選択 (SELECT) コマンドを使用すると、1 つ以上の条件と、その条件が真の場合に処理されるコマンドの関連グループを識別できます。

宣言されている条件がすべて真でない場合に実行される、コマンドの特殊なグループを指定することもできます。 **When (WHEN)** または **他の場合 (OTHERWISE)** コマンドで識別されるコマンドのグループのうち、1 つだけがグループ内で処理されます。

SELECT コマンドの一般的な構造は以下のとおりです。

```
SELECT
  WHEN (condition-1) THEN(command-1)
  .
  .
  .
  WHEN (condition-n) THEN(command-n)
  OTHERWISE command-x
ENDSELECT
```

SELECT グループでは最低でも 1 つの **WHEN** コマンドを指定する必要があります。 **WHEN** コマンドには、真偽がテストされる式と、その式が真であった場合にとるべき処置を指定するオプションの THEN パラメーターが含まれます。

COND パラメーターに指定する論理式は 1 つの論理変数または定数であるか、または 2 つ以上のオペランドの関係を示すものでなければなりません。この式について真か偽かが評価されます。

論理式により指定された条件が真であると評価された場合には、プロシージャーは THEN パラメーターに指定されている CL コマンドを処理します。処理されるコマンドは、1 つの場合もあり、DO、DOWHILE、DOUNTIL、または DOFOR コマンドで指定されるコマンドのグループである場合もあります。条件が真でなければ、SELECT グループ内の次回の **WHEN** コマンドで指定される条件が評価されます。次回の **WHEN** コマンドが存在しない場合は、 **OTHERWISE** コマンドで識別されるコマンド (存在する場合) が処理されます。次回の WHEN コマンドと **OTHERWISE** コマンドがいずれも存在しない場合は、関連する ENDSELECT コマンドに続く次回のコマンドから処理が継続します。

```
SELECT
  WHEN (&LGL)
  WHEN (&INT *LT 0) THEN(CHGVAR &INT 0)
  WHEN (&INT *GT 0) (DOUNTIL (&INT *EQ 0))
                       CHGVAR &INT (&INT - 1)
                     ENDDO
  OTHERWISE (CHGVAR &LGL '1')
ENDSELECT
```

&LGL の初期値が真 ('1') の場合は、THEN パラメーターが存在しないので ENDSELECT に続くコマンドから処理が継続します。

&LGL の初期値が偽 ('0') の場合は、2 番目の WHEN の COND が評価されます。 &INT がゼロより小さい場合は、CHGVAR が処理され、 &INT の値がゼロに設定されます。その後、ENDSELECT に続くコマンドから処理が継続します。

最初の 2 つの条件が満たされない場合は、&INT の値がゼロより大きいかどうかが検査されます。この値がゼロより大きい場合は、DOUNTIL グループに入り、&INT がゼロに達するまで減分されます。 &INT がゼロに達すると、DOUNTIL グループは終了し、 ENDSELECT に続くコマンドから処理が継続します。

いずれの **WHEN** コマンドでも条件が真として評価されない場合は、 **OTHERWISE** コマンドの CMD パラメーターで指定された CHGVAR が処理されます。&LGL が真に設定されている間、&INT の値は変更されません。その後、ENDSELECT に続くコマンドから処理が継続します。

関連資料:

221 ページの『*AND、*OR、および *NOT 演算子』
論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

関連情報:

CL コマンド検索プログラム

SELECT グループ (SELECT) コマンド

CL プログラムまたはプロシージャー内の SUBR コマンドおよびサブルーチン

サブルーチン (SUBR) コマンドは、**サブルーチン終了 (ENDSUBR)** コマンドとともに CL プログラムまたは CL プロシージャーで使用され、サブルーチンを定義するコマンドのグループを区切ります。

CALLSUBR コマンドで使用される場合のサブルーチン名は、**SUBR** コマンドの SUBR パラメーターにより定義されます。

また、プログラムまたはプロシージャーで最初に発生する **SUBR** コマンドも、プログラムまたはプロシージャーの主部の終わりをマークします。この時点より後に発生するすべてのコマンド (ENDPGM コマンドを

除く) は、サブルーチンに含まれている必要があります。つまり、**SUBR** コマンドと **ENDSUBR** コマンドの間に配置される必要があります。**SUBR** コマンドと **ENDSUBR** コマンドは、一致するペアである必要があり、サブルーチンはネストすることができません。つまり、サブルーチンの SUBR SUBR - ENDSUBR ENDSUBR ネストはできません。

ENDSUBR および RTNSUBR コマンドの両方を使用して、サブルーチンを終了することができ、処理されると、サブルーチンを呼び出した**サブルーチン呼び出し (CALLSUBR)** コマンドの直後のコマンドに制御権が戻されます。両方のコマンドは、オプションの RTNVAL パラメーターを持ち、これは、TYPE(*INT) および LEN(4) の CL 変数に格納可能な任意のものにできます。**サブルーチンからの戻り (RTNSUBR)** コマンドまたは **ENDSUBR** コマンドで RTNVAL パラメーターが定義されていない場合、値ゼロが戻されます。

以下に、サブルーチンを含む CL プロシージャーの一般的構造の例を示します。

```
    PGM
    DCLPRCOPT  SUBRSTACK(25)
    DCL        VAR(&RTNVAR) TYPE(*INT) LEN(4)
:
    CALLSUBR   SUBR(SUBR1) RTNVAL(&RTNVAR)
:
    SUBR       SUBR(SUBR1)
:
    RTNSUBR    RTNVAL(-1)
:
    ENDSUBR
    ENDPGM
```

この例では、**処理オプション宣言 (DCLPRCOPT)** コマンドを使用して、サブルーチン・スタックのサイズを 25 に指定しています。変数 &RTNVAR は、サブルーチンからの戻り値を含むように使用されています。**CALLSUBR** コマンドは、SUBR コマンドにより定義されたように、制御権をサブルーチン SUBR1 に転送します。**RTNSUBR** コマンドが実行されている場合、&RTNVAR の値は -1 になり、**ENDSUBR** コマンドが実行されている場合、&RTNVAR は 0 になります。**CALLSUBR** コマンドで RTNVAL パラメーターが定義されていない場合、サブルーチンからの戻り値は無視されます。

関連資料:

218 ページの『CL プログラムまたはプロシージャーの CALLSUBR コマンド』
サブルーチン呼び出し (CALLSUBR) コマンドは、同じプログラムまたはプロシージャー内で定義されたサブルーチンに制御を渡すために CL プログラムまたは CL プロシージャーで使用されます。

関連情報:

CL コマンド検索プログラム

サブルーチン (SUBR) コマンド

サブルーチン終了 (ENDSUBR) コマンド

*AND、*OR、および *NOT 演算子

論理演算子 *AND および *OR は、論理式のオペランド相互間の関係を指定します。論理演算子 *NOT は、論理変数や定数を否定するために使用されます。

*AND および *OR は、論理式のオペランド相互間の関係を指定するために使用する予約値です。予約値 *AND の代わりにアンパーサンド (&) を、 *OR の代わりに縦線 (|) を使用することもできます。予約値の前後にはブランクがなければなりません。論理式の中のオペランドは、関係式で構成されるか、あるいは論理演算子で区切った論理変数または定数によって構成されます。 *AND 演算子は、結果が真であるためには、両方のオペランド (演算子の両側の) が真でなければならないことを意味します。 *OR 演算子は、結果が真であるためにはオペランドの一方または両方が真でなければならないことを意味します。

注**:** コード・ページによって記号とコード・ポイントの対応が異なるため、アンパーサンドや縦線 (|) を使用すると問題が生じる場合があります。この問題を回避するためには、記号の代わりに *AND や *OR を使用します。

論理演算子以外の演算子は、式の中のオペランドに実行される処置、またはオペランド相互間の関係を示すために式の中で使用します。論理演算子以外の演算子には次の 3 種類があります。

- 算術演算子 (+、-、*、/)
- 文字 (*CAT、||、*BCAT、|>、*TCAT、|<)
- 関係演算子 (*EQ、=、*GT、>、*LT、<、*GE、>=、*LE、<=、*NE、¬=、*NG、¬>、*NL、¬<)

以下の例は論理式について示しています。

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C < 1) & (&TIME>1430))
((&C< 1) & (&TIME>1430))
```

上記のどの場合も、論理式は 2 つのオペランドと 1 つの演算子 (*AND、*OR、またはそれを表す記号) の 3 つの部分で構成されています。式が論理式であるかどうかを決めるのはオペランドのタイプではなく、演算子のタイプ (*AND または *OR) です。論理式のオペランドとして使用できるのは、論理変数または他の式 (関係式など) です。 (関係式とは、>、<、または =、あるいはそれに対応する予約値が含まれている式のことです。) 例えば、次の例では、論理式全体が括弧で囲まれており、また両方のオペランドが関係式で、それぞれが別個に括弧で囲まれています。

```
((&C *LT 1) *AND (&TIME *GT 1430))
```

前の例の 2 番目の論理式のように、各オペランドをそれぞれ独立して括弧で囲む必要はありませんが、意味を明確にするために括弧で囲むことをお勧めします。また、*AND と *OR とは優先順位が異なるので括弧は不要です。 *AND は常に *OR より先に評価されます。同じ優先順位の演算子をいくつか使用する場合には、演算を行う順序を制御するために括弧を使用することができます。

コマンドの中の条件として単純な関係式を使用することができます。

```
IF (&A=&B) THEN(DO)
             .
             .
             .
             ENDDO
```

この関係式のオペランドは定数にすることもできます。

複数の条件を指定したい場合には、オペランドとして関係式を用いた論理式を使用することができます。

```
IF ((&A=&B) *AND (&C=&D)) THEN(DO)
                            .
                            .
                            .
                            ENDDO
```

210 ページの『CL プログラムまたはプロシージャー内の組み込み IF コマンド』の項で例として示した一連の IF コマンドは、次のようにコーディングすることができます。

```
PGM
DCL &RESP *DEC 1
DCL &A *DEC 1
DCL &B *CHAR 2
IF ((&RESP=1) *AND (&A=5) *AND (&B=NO)) THEN(DO)
                                          .
                                          .
```

```
                                        .
                                        ENDDO
CHGVAR &A VALUE(8)
CALLPRC PROC(DAILY)
ENDPGM
```

ここでも、関係式と関係式との間で論理演算子が使用されています。

論理式は他の論理式をオペランドとして使用することができるので、複雑なロジックを組み立てることも可能です。

```
IF (((&A=&B) *OR (&A=&C)) *AND ((&C=1) *OR (&D='0'))) THEN(DO)
```

この例では、&D は論理変数として定義されます。

関係式または論理式の評価の結果は、'1' または '0' (真または偽) のどちらかです。従属するコマンドが処理されるのは、式全体が真 ('1') として評価された場合だけです。次のコマンドは、真および偽の用語を用いて説明しています。

```
IF   ((&A = &B) *AND (&C = &D)) THEN(DO)

     ((真 '1') *AND (偽 '0'))
          (偽 '0')
```

この式は最終的には真でない ('0') と評価され、したがって DO は処理されません。どのようにしてこの評価に達するかの説明については、この項に示す行列を参照してください。

これと同じプロセスが、次の例のように、論理変数を使った論理式の評価に使用されます。

```
PGM
DCL &A *LGL
DCL &B *LGL
IF (&A *OR &B) THEN(CALL PGM(PGMA))
.
.
.
ENDPGM
```

この例では、&A または &B の値が '1' (真) であるかどうかを調べるために、条件式が評価されます。どちらかの値が真であれば、式全体が真となり、PGMA が呼び出されます。

これまでのすべての論理式の例において、最終的に到達した評価の結果は、 *OR または *AND 演算子での 2 つの値 (ここでは &A と &B) を比較する標準的な行列に基づいています。

論理変数または定数と *OR を使用する場合には、次の行列を使用します。

&A の値**:**
'0' '0' '1' '1'

&B の値**:**
'0' '1' '0' '1'

OR 式の結果**:**
'0' '1' '1' '1'

要約すると、論理変数または定数と複数の OR 演算子が存在する場合には、すべての値が偽 ('0') であればその式は偽になります。そして、値のいずれかが真 ('1') であればその式は真になります。

```
PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
```

```
DCL &C *LGL VALUE('1')
IF (&A *OR &B *OR &C) THEN(CALL PGMA)
.
.
.
ENDPGM
```

この例では、すべての値が偽ではないので式も真になり、したがって PGMA は呼び出されません。

論理変数または定数を *AND で結んだ論理式を評価する場合には、次の行列を使用します。

&A の値:
'0' '0' '1' '1'

&B の値:
'0' '1' '0' '1'

ANDed 式の結果:
'0' '0' '0' '1'

論理変数または定数と AND 演算子からなる論理式の評価は、値のいずれかが偽 ('0') であれば式は偽であり、すべての値が真であれば式も真になります。

```
PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *AND &B *AND &C) THEN(CALL PGMA)
.
.
.
ENDPGM
```

この例では、すべての値が真ではないので式は偽になり、したがって PGMA は呼び出されません。

これらの論理演算子を式の中で使用できるのは、これまでの例のように、オペランドが論理値を表している場合だけです。論理変数以外の変数に対して OR または AND を使用するのは誤りです。次の例をご覧ください。

```
PGM
DCL &A *CHAR 3
DCL &B *CHAR 3
DCL &C *CHAR 3

誤り:   IF (&A *OR &B *OR &C = YES) THEN...
```

この場合の正しいコーディングは、次のようになります。

```
IF ((&A=YES) *OR (&B=YES) *OR (&C=YES)) THEN...
```

このようにすれば、関係式間の OR 関係が生じます。

論理演算子 *NOT (または ¬) は、論理変数や定数を否定するために使用されます。*NOT 演算子は、*AND または *OR 演算子の評価の前に評価されます。また *NOT 演算子に続く値は、オペランド相互間の論理関係の評価より前に評価されます。

```
PGM
DCL &A *LGL '1'
DCL &B *LGL '0'
IF (&A *AND *NOT &B) THEN(CALL PGMA)
```

この例では、値がすべて真であるので式も真になり、したがって PGMA が呼び出されます。

```
PGM
DCL &A *LGL
DCL &B *CHAR 3 VALUE('ABC')
DCL &C *CHAR 3 VALUE('XYZ')
CHGVAR &A VALUE(&B *EQ &C)
IF (&A) THEN(CALLPRC PROCA)
```

この例では、値が偽であるので PROCA は呼び出されません。

関連資料:

206 ページの『CL プログラムまたはプロシージャー内の IF コマンド』
IF (IF) コマンドは、条件を指定する場合に使用され、その条件が真の場合は、プログラムまたはプロシージャー内の実行するステートメントやステートメントのグループを指定します。

208 ページの『CL プログラムまたはプロシージャー内の ELSE コマンド』
ELSE (ELSE) コマンドは、対応する **IF (IF)** コマンドで指定した条件が偽であった場合に、代わりの処理を指定するための手段です。

214 ページの『CL プログラムまたはプロシージャー内の DOUNTIL コマンド』
Do Until (DOUNTIL) コマンドは、CL コマンドのグループを 1 回以上処理します。

215 ページの『CL プログラムまたはプロシージャー内の DOWHILE コマンド』
Do While (DOWHILE) コマンドは、複数のコマンドをグループ化し、論理式の値が真の間にゼロ回またはそれ以上処理するためのものです。

219 ページの『CL プログラムまたはプロシージャー内の SELECT コマンドおよび SELECT グループ』
選択 (SELECT) コマンドを使用すると、1 つ以上の条件と、その条件が真の場合に処理されるコマンドの関連グループを識別できます。

%ADDRESS 組み込み関数

アドレス組み込み関数 (%ADDRESS または %ADDR) を使用して、CL ポインター変数に保管されているメモリー・アドレスを変更またはテストすることができます。

CHGVAR コマンドで %ADDRESS 関数を指定して、ポインター変数の値を変更することができます。IF コマンドの COND パラメーターで %ADDRESS 関数を指定して、ポインター変数に保管されている値をチェックすることができます。

アドレス組み込み関数の形式は、以下のとおりです。

```
%ADDRESS(変数名)
```

または

```
%ADDR(変数名)
```

以下の例では、ポインター変数 &P1 が文字変数 &C1 の最初のバイトのアドレスに初期設定されます。このプロシージャーの後半で、このポインターがまだ &C1 を指しているかどうかを確認するために %ADDRESS 関数を使用してこのポインターがチェックされ、指していない場合は %ADDRESS 関数を使用して &P1 が CL 変数 &C1 の最初のバイトにリセットされます。

```
PGM
DCL &C1  *CHAR  10
DCL &P1   *PTR  ADDRESS(&C1)
:
IF COND(&P1 *NE %ADDRESS(&C1))  +
   THEN(CHGVAR  &P1  %ADDRESS(&C1))
:
ENDPGM
```

注**:** %ADDRESS 関数を使用して、ポインター変数に保管されているアドレス・オフセットを保管することはできません。ポインター変数内のアドレス・オフセットを保管するには、%OFFSET 組み込み関数を使用する必要があります。

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%BINARY 組み込み関数

2 進数組み込み関数 (%BINARY または %BIN) は、指定された CL 文字変数の内容を、符号付き 2 進整数として解釈します。

開始位置は指定された位置で、2 文字または 4 文字の長さまで続きます。

次に 2 進組み込み関数の構文の例を示します。

```
%BINARY(文字変数名 開始位置 長さ)
```

この例のようにこれをコーディングすることもできます。

```
%BIN(文字変数名 開始位置 長さ)
```

開始位置および長さはオプションです。ただし、開始位置と長さが指定されていない場合、開始位置 1 および指定された文字変数の長さが使用されます。その場合には、文字変数の長さを 2 または 4 として宣言しなければなりません。

開始位置が指定されている場合には、同様に 2 または 4 の固定の長さを指定することが必要です。開始位置は、1 以上の正の数でなければなりません。開始位置および長さの合計が文字変数の長さよりも大きい場合、エラーが起こります。 (CL 10 進変数または整変数を開始位置として使用することもできます。)

2 進数組み込み関数は、**If (IF)** および**変数変更 (CHGVAR)** コマンドの両方で使用することができます。2 進数組み込み関数は単独で使用するか、または算術式や論理式の一部として使用することができます。また、EXPR(*YES) を指定した数字 (*DEC、*INT2、 *INT4、*UINT2、または *UINT4 の TYPE) として定義された任意のコマンド・パラメーターで、2 進数組み込み関数を使用することもできます。

2 進数組み込み関数が、IF コマンドの条件 (COND) パラメーター、または**変数変更 (CHGVAR)** コマンドの VALUE パラメーターで使用される場合、文字変数の内容は 2 進数から 10 進数への変換として解釈されます。

2 進数組み込み関数が **変数変更 (CHGVAR)** コマンドの VAR パラメーターで使用される場合、VALUE パラメーターの 10 進数は 2 バイトまたは 4 バイトの符号付き 2 進整数に変換され、指定された開始位置で文字変数に保管されます。 10 進数の小数部は切り捨てられます。

システムは 2 進数組み込み関数を**プロシージャーの呼び出し (CALLPRC)** コマンドの RTNVAL パラメーターで使用し、呼び出されたプロシージャーが符号付き 2 進整数を返すことを、呼び出し元プロシージャーが期待していることを示します。

2 バイト文字変数は、-32 768 から 32 767 までの符号付き 2 進整数値にすることができます。 4 バイト文字変数は、 -2 147 483 648 から 2 147 483 647 までの符号付き 2 進整数値にすることができます。

次は、2 進数組み込み関数の例です:

- ```
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)    VALUE(X'001C')
  DCL    VAR(&N)   TYPE(*DEC)   LEN(3 0)
  CHGVAR   &N   %BINARY(&B2)
  ```

  変数 &B2 の内容は、2 バイトの符号付き 2 進整数として扱われ、10 進数の等価値 28 に変換されます。その後、10 進変数 &N に割り当てられます。

- ```
  DCL    VAR(&N)   TYPE(*DEC)   LEN(5 0)  VALUE(107)
  DCL    VAR(&B4)  TYPE(*CHAR)  LEN(4)
  CHGVAR   %BIN(&B4)    &N
  ```

  10 進値 &N の値は、4 バイトの符号付き 2 進数に変換され、文字変数 &B4 に入れられます。変数 &B4 には X'0000006B' の値が入ります。

- ```
  DCL    VAR(&P)  TYPE(*CHAR)  LEN(100)
  DCL    VAR(&L)  TYPE(*DEC)   LEN(5 0)
  CHGVAR   &L  VALUE(%BIN(&P 1 2) * 5)
  ```

  変数 &P の最初の 2 文字は符号付き 2 進整数として扱われ、10 進数の等価値に変換され 5 倍にされます。その積は、10 進変数 &L に割り当てられます。

- ```
  DCL    VAR(&X)   TYPE(*CHAR)  LEN(50)
  CHGVAR   %BINARY(&X 15 2)  VALUE(122.56)
  ```

  数値 122.56 は切り捨てられて整数 122 になり、その後 2 バイトの符号付き 2 進整数に変換されて、文字変数 &X の 15 および 16 の位置に置かれます。変数 &X の位置 15 および 16 には、16 進数の等価値 X'007A' が入ります。

- ```
  DCL    VAR(&B4)  TYPE(*CHAR)  LEN(4)
  CHGVAR   %BIN(&B4)  VALUE(-57)
  ```

  値 -57 は 4 バイトの符号付き 2 進変数に変換され、文字変数 &B4 に割り当てられます。次に、変数 &B4 は、値 X'FFFFFFC7'を含みます。

- ```
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)   VALUE(X'FF1B')
  DCL    VAR(&C5)  TYPE(*CHAR)  LEN(5)
  CHGVAR   &C5  %BINARY(&B2)
  ```

  変数 &B2 の内容は、2 バイトの符号付き 2 進整数として扱われ、10 進数の等価値 -229 に変換されます。数値は文字形式に変換され、変数文字 &C5 に保管されます。次に、文字変数 &C5 は値 '-0229' を含みます。

- ```
  DCL    VAR(&C5)  TYPE(*CHAR)  LEN(5)   VALUE(' 1253')
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)
  CHGVAR   %BINARY(&B2)  VALUE(&C5)
  ```
  ```

文字変数 &C5 の文字数値 1253 は 10 進数に変換されます。 10 進数 1253 は 2 バイトの符号付き 2 進整数に変換され、変数 &B2 に保管されます。次に、変数 &B2 は、値 X'04E5' を持ちます。

-

```
DCL    VAR(&S)  TYPE(*CHAR)  LEN(100)
IF     (%BIN(&S 1 2) *GT 10)
       THEN( SNDPGMMSG MSG('Too many in list.') )
```

文字変数 &S の最初の 2 バイトは、数 10 と比較されるときに符号付き 2 進整数として扱われます。2 進数に 10 より大きい値がある場合、**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドが実行されます。

-

```
DCL    VAR(&RTNV) TYPE(*CHAR) LEN(4)
CALLPRC PRC(PROCA) RTNVAL(%BIN(&RTNV 1 4))
```

プロシージャー PROCA は変数 &RTNV に保管されている 4 バイトの整数を返します。

関連タスク:

198 ページの『変数の値の変更』

CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』

制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

関連情報:

CL コマンド検索プログラム

仮定 (IF) コマンド

変数変更 (CHGVAR) コマンド

結合プロシージャー呼び出し (CALLPRC) コマンド

プログラム・メッセージ送信 (SNDPGMMSG) コマンド

%CHAR 組み込み関数

%CHAR は、論理、10 進数、整数、または符号なし整数の各データを文字フォーマットに変換します。変換された値は、CL 変数に割り当てたり、文字定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

%CHAR 組み込み関数は、CL が文字式をサポートする場合はどこでも使用できます。%CHAR は、単独でも、より複雑な文字式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 文字変数に対して CL 数値変数を比較する際に、%CHAR を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ
*CHAR、*NAME、*SNAME、*CNAME、*PNAME、*GENERIC、*DATE、*TIME、または *X を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %CHAR を使用することができます。

文字データへの変換組み込み関数の形式は次のとおりです。

```
%CHAR(convert-argument)
```

convert-argument はタイプが *LGL、*DEC、*INT、または *UINT の CL 変数でなければなりません。

論理データの場合、結果は「0」と「1」のいずれかになります。

数値データの場合の結果は、左寄せにされ、値が負であれば先行負符号が付いた 10 進数形式になります。先行ゼロはありません。小数点は必要に応じて挿入されます。なお、小数点に使用される文字は、10 進数形式 (QDECFMT) のシステム値が示す文字 (デフォルト値は「.」) となります。例えば、TYPE(*DEC) および LEN(7,3) の CL 変数で宣言される CL 変数の %CHAR が返す値は、「-1.234」などになります。また、変換結果のゼロ抑制のタイプも、10 進数形式 (QDECFMT) のシステム値によって決まります。

以下は、%CHAR 組み込み関数の使用例です。

- パック 10 進変数の変換

```
DCL VAR(&MSG) TYPE(*CHAR) LEN(26)
DCL VAR(&ANSWER) TYPE(*DEC) LEN(10 5) VALUE(-54321.09876)

/* &MSG will have the value 'The answer is -54321.09876' */
CHGVAR VAR(&MSG) VALUE('The answer is' *BCAT %CHAR(&ANSWER))
```

- 整変数または符号なし整変数の変換

```
DCL VAR(&MSG) TYPE(*CHAR) LEN(60)
DCL VAR(&LENGTH) TYPE(*UINT) LEN(2) VALUE(50)
DCL VAR(&TEMP) TYPE(*INT) LEN(2) VALUE(-10)

/* &MSG will have the value 'The length of this box is 50 centi+
meters.                    ' */
CHGVAR VAR(&MSG) VALUE('The length of this box is' *BCAT %CHAR(&LENGTH) +
*BCAT 'centimeters.')
/* &MSG will have the value 'The temperature dropped to -10 degrees +
Centigrade.          ' */
CHGVAR VAR(&MSG) VALUE('The temperature dropped to' *BCAT %CHAR(&TEMP) +
*BCAT 'degrees Centigrade.')
```

- 論理変数の変換

```
DCL VAR(&ENDOFFILE) TYPE(*LGL) VALUE('1')
SNDPGMMSG MSG('End of file?' *BCAT %CHAR(&ENDOFFILE))
```

- 先行ゼロは抑止されますが、小数部の桁のゼロはすべて保持されます

```
DCL &AMOUNT TYPE(*DEC) LEN(7 4) VALUE(099.5)
/* &MSG will have the value "Your amount comes to 99.5000" */
SNDPGMMSG MSG('Your amount comes to' *BCAT %CHAR(&AMOUNT))
```

- 10 進数形式およびゼロ抑制

```
DCL VAR(&MSG) TYPE(*CHAR) LEN(20)
DCL VAR(&ANSWER) TYPE(*DEC) LEN(10 2) VALUE(-0.45)

/* &MSG will have the value 'The answer is -.45  ', while +
CHGJOB DECFMT(*BLANK) */
/* &MSG will have the value 'The answer is -0,45 ', while CHGJOB DECFMT(J) */
/* &MSG will have the value 'The answer is -,45  ', while CHGJOB DECFMT(I) */
CHGVAR VAR(&MSG) VALUE('The answer is' *BCAT %CHAR(&ANSWER))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログ

ラムまたはプロシージャーでのみ使用できます。

%CHECK 組み込み関数

チェック組み込み関数 (%CHECK) は、コンパレーター・ストリング にはない文字を含む基本ストリングの 1 桁目を戻します。基本ストリングのすべての文字がコンパレーター・ストリングにも現れる場合、この関数の返す値は 0 です。

%CHECK 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%CHECK は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 数値変数に対する比較を行う際に、%CHECK を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %CHECK を使用することができます。

チェック組み込み関数の形式は、以下のとおりです。

```
%CHECK(comparator-string base-string [starting-position])
```

comparator-string は、CL 文字変数または文字リテラルのいずれかでなければなりません。*base-string* には CL 文字変数または *LDA のいずれかを指定できます。*LDA が指定されていると、そのジョブのローカル・データ域のコンテンツに対してチェック関数が実行されます。*starting-position* は任意指定で、デフォルト値は 1 です。検査は開始位置 (start) から始められ、コンパレーター・ストリングに含まれていない文字が見つかるまで、右方へと続けられます。開始位置が指定されている場合でも、この結果は常にソース・ストリング内の最初のバイトに対する相対的な値となります。開始位置を指定する場合、その値は CL 整変数、小数点以下の桁数がゼロの CL 10 進変数、もしくは小数点以下の桁数がゼロの数値リテラルのいずれかです。開始位置をゼロまたは負の値にすることはできません。開始位置がベース変数またはローカル・データ域全体の長さを超えると、エラーが起こります。ローカル・データ域の長さは 1024 です。

以下は、チェック組み込み関数の例です。

- 数字 (0 から 9) ではない文字がないかどうか検査します。ここで、コンパレーター・ストリングは、数字 0 から 9 を含む文字リテラルです。CL 変数 &SN に数字ではない文字が含まれる場合、メッセージが送信されます。

  ```
  PGM PARM(&SN)
  DCL VAR(&SN) TYPE(*CHAR) LEN(10)
  IF COND(%CHECK('0123456789' &SN) *NE 0) +
     THEN(SNDPGMMSG ('INVALID CHARACTER FOUND!'))
  ```

- ドル記号でもアスタリスクでもない最初の文字を見つけます。変数 &PRICE の文字が左から右へ検査されます。ドル記号でもアスタリスクでもない最初の文字は 8 番目の文字なので、**CHGVAR** コマンドによって値 8 が CL 変数 &POS に割り当てられます。

  ```
  DCL VAR(&PRICE) TYPE(*CHAR) VALUE('$******5.27***      ')
  DCL VAR(&COMP) TYPE(*CHAR) LEN(2) VALUE('$*')
  DCL VAR(&POS) TYPE(*UINT) LEN(2)
  CHGVAR VAR(&POS) VALUE(%CHECK(&COMP &PRICE))
  ```

- ストリング中の特定の位置から開始して、数字でない文字がないか検査します。CL 変数 &SN 内の文字の検査が、5 番目のバイトから開始されます。&SN の 5 番目から 10 番目までのバイトの中に数字でない文字が含まれていれば、メッセージが送信されます。

  ```
  PGM PARM(&SN)
  DCL VAR(&SN) TYPE(*CHAR) LEN(10)
  DCL VAR(&POS) TYPE(*UINT) LEN(2) VALUE(5)
  IF COND(%CHECK('0123456789' &SN &POS) *NE 0) +
     THEN(SNDPGMMSG ('INVALID CHARACTER FOUND!'))
  ```

- ローカル・データ域 (*LDA) の文字を検査します。数字ではない、ローカル・データ域 (LDA) 内の左端のバイトの位置が、CL 変数 &POS に割り当てられます。LDA の 1024 文字すべてが数字である場合、値ゼロが変数 &POS に割り当てられます。

```
DCL VAR(&POS) TYPE(*UINT) LEN(2)
CHGVAR VAR(&POS) VALUE(%CHECK('0123456789' *LDA))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%CHECKR 組み込み関数

リバース・チェック組み込み関数 (%CHECKR) は、コンパレーター・ストリング にはない文字を含む基本ストリング の最後の桁を返します。基本ストリングのすべての文字がコンパレーター・ストリングにも現れる場合、この関数の返す値は 0 です。

%CHECKR 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%CHECKR は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 数値変数に対する比較を行う際に、%CHECKR を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %CHECKR を使用することができます。

リバース・チェック組み込み関数の形式は、以下のとおりです。

```
%CHECKR(comparator-string base-string [starting-position])
```

comparator-string は、CL 文字変数または文字リテラルのいずれかでなければなりません。*base-string* には CL 文字変数または *LDA のいずれかを指定できます。*LDA が指定されていると、そのジョブのローカル・データ域のコンテンツに対してリバース・チェック関数が実行されます。*starting-position* は任意指定で、デフォルトの値は基本ストリングの最後のバイトになります。検査は開始位置 (start) から始められ、コンパレーター・ストリングに含まれていない文字が見つかるまで、左方へと続けられます。開始位置が指定されている場合でも、この結果は常にソース・ストリング内の最初のバイトに対する相対的な値となります。 開始位置を指定する場合、その値は CL 整変数、小数点以下の桁数がゼロの CL 10 進変数、もしくは小数点以下の桁数がゼロの数値リテラルのいずれかです。開始位置をゼロまたは負の値にすることはできません。開始位置がベース変数またはローカル・データ域全体の長さを超えると、エラーが起こります。ローカル・データ域の長さは 1024 です。

以下は、リバース・チェック組み込み関数の例です。

- ストリングから数値を取り出します。最初の **CHGVAR** は、%CHECK 組み込み関数を使用して、ドル記号 ($)、アスタリスク (*)、ブランクのいずれでもない左端の文字を検索します。2 番目の **CHGVAR** は、%CHECKR 組み込み関数を使用して、ドル記号、アスタリスク、ブランクのいずれでもない右端の文字を検索します。3 番目の **CHGVAR** は、この 2 つの値の間にある文字数を計算し、最後の **CHGVAR** は、サブストリング (%SST) 組み込み関数を使用して、基本ストリングの一部を CL 10 進変数に変換します。

```
DCL VAR(&PRICE) TYPE(*CHAR) VALUE('$******5.27***      ')
DCL VAR(&COMP) TYPE(*CHAR) LEN(3) VALUE('$* ')
DCL VAR(&SPOS) TYPE(*UINT) LEN(2)
DCL VAR(&EPOS) TYPE(*UINT) LEN(2)
DCL VAR(&DEC) TYPE(*DEC) LEN(3 2)
DCL VAR(&LEN) TYPE(*UINT) LEN(2)
CHGVAR VAR(&SPOS) VALUE(%CHECK(&COMP &PRICE))
CHGVAR VAR(&EPOS) VALUE(%CHECKR(&COMP &PRICE))
CHGVAR VAR(&LEN) VALUE(&EPOS - &SPOS + 1)
CHGVAR VAR(&DEC) VALUE(%SST(&PRICE &SPOS &LEN))
```

- ストリング中の特定の位置から開始して、数字でない文字がないかリバース検査を行います。CL 変数 &SN 内の文字の検査が、9 番目のバイトから開始され、右から左への順序で行われます。&SN の 9 番目から 1 番目までのバイトの中に数字でない文字が含まれていれば、メッセージが送信されます。

```
PGM PARM(&SN)
DCL VAR(&SN) TYPE(*CHAR) LEN(10)
DCL VAR(&POS) TYPE(*UINT) LEN(2) VALUE(9)
IF COND(%CHECKR('0123456789' &SN &POS) *NE 0) +
   THEN(SNDPGMMSG ('INVALID CHARACTER FOUND!'))
```

- ローカル・データ域 (*LDA) の文字をリバース検査します。数字ではない、ローカル・データ域 (LDA) 内の右端のバイトの位置が、CL 変数 &POS に割り当てられます。LDA の 1024 文字すべてが数字である場合、値ゼロが変数 &POS に割り当てられます。

```
DCL VAR(&POS) TYPE(*UINT) LEN(2)
CHGVAR VAR(&POS) VALUE(%CHECKR('0123456789' *LDA))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%DEC 組み込み関数

%DEC は、文字、論理、10 進数、整数、または符号なし整数データをパック 10 進数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

%DEC 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%DEC は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 数値変数に対して CL 文字変数を比較する際に、%DEC を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %DEC を使用することができます。

パック 10 進データへの変換組み込み関数の形式は次のとおりです。

```
%DEC(convert-argument [total-digits decimal-places])
```

convert-argument はタイプが *CHAR、*LGL、*DEC、*INT、または *UINT の CL 変数でなければなりません。

パラメーター *total-digits* および *decimal-places* は省略可能です。これらのパラメーターが省略されている場合、文字データでは、合計桁数のデフォルト値は 15 となり、小数点以下の桁数のデフォルト値は 5 となります。論理データでは、合計桁数のデフォルト値は 1 となり、小数点以下の桁数のデフォルト値は 0 となります。数値データでは、合計桁数と小数点以下の桁数は数値データの属性から取得されます。合計桁数と小数点以下の桁数を指定する場合、これらの値は整数リテラルでなければなりません。*total-digits* には、1 から 15 までの値を指定できます。*decimal-places* には 0 から 9 までの値を指定できますが、その値が *total-digits* の値を超えてはいけません。

convert-argument パラメーターが文字変数である場合は、次の規則が適用されます。

- 符号は任意指定です。'+' または '-' を使用できます。数値データの前に付けることも、後ろに付けることもできます。
- 小数点は任意指定です。 ピリオドでもコンマでもかまいません。
- データ内では先行ブランクと末尾ブランクが使用できます。例えば、' + 3 ' は有効なパラメーターです。
- 値をすべてブランクにすると、値ゼロが返されます。
- 無効な数値データが検出された場合、CPF0818 の例外が発生します。

以下は、%DEC 組み込み関数の使用例です。

- 文字変数の変換

```
DCL VAR(&POINTS) TYPE(*CHAR) LEN(10) VALUE('-123.45')
DCL VAR(&ANSWER) TYPE(*DEC) LEN(10 5)

/* &ANSWER will have the value -00023.45000 */
CHGVAR VAR(&ANSWER) VALUE(100 + %DEC(&POINTS 5 2))
```

- 論理変数の変換

```
DCL VAR(&ANSWER1) TYPE(*LGL) VALUE('1')
DCL VAR(&ANSWER2) TYPE(*LGL) VALUE('1')
DCL VAR(&NUM) TYPE(*DEC) LEN(5 0)

/* &NUM will have the value 00002. */
CHGVAR VAR(&NUM) VALUE(%DEC(&ANSWER1 1 0) + %DEC(&ANSWER2))
SNDPGMMSG MSG('The number of YES answers is' *BCAT %CHAR(&NUM))
```

- パック 10 進変数の変換

```
DCL VAR(&POINTS1) TYPE(*DEC) LEN(5 2) VALUE(100.23)
DCL VAR(&POINTS2) TYPE(*DEC) LEN(5 2) VALUE(100.45)

IF (%DEC(&POINTS1 3 0) *EQ %DEC(&POINTS2 3 0)) +
THEN(SNDPGMMSG ('The scores are the same!'))
```

- 整変数または符号なし整変数の変換

```
DCL VAR(&P1) TYPE(*INT) LEN(2) VALUE(-1)
DCL VAR(&P2) TYPE(*UINT) LEN(2) VALUE(1)
DCL VAR(&NUM) TYPE(*DEC) LEN(5 0)

/* &NUM will have the value 00000. */
CHGVAR VAR(&NUM) VALUE(%DEC(&P1 10 0)+%DEC(&P2 5 2))
```

- 余分な小数部分の桁は、丸めなしで切り捨てられます

```
DCL VAR(&STRING) TYPE(*CHAR) LEN(10) VALUE('-123.567')
DCL VAR(&VALUE) TYPE(*DEC) LEN(7 2)

/* &VALUE will have the value -00123.56 */
CHGVAR VAR(&VALUE) VALUE(%DEC(&STRING 7 2))
```

- 値をすべてブランクにすると、値ゼロが返されます

```
DCL VAR(&STRING) TYPE(*CHAR) LEN(10) VALUE('          ')
DCL VAR(&VALUE) TYPE(*DEC) LEN(7 2)

/* &VALUE will have the value 00000.00 */
CHGVAR VAR(&VALUE) VALUE(%DEC(&STRING 7 2))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%INT 組み込み関数

%INT は、文字、論理、10 進数、または符号なし整数データを整数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

%INT 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%INT は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 数値変数に対して CL 文字変数を比較する際に、%INT を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %INT を使用することができます。

整数データへの変換組み込み関数の形式は次のとおりです。

```
%INT(convert-argument)
```

convert-argument はタイプが *CHAR、*LGL、*DEC、または *UINT の CL 変数でなければなりません。

convert-argument パラメーターが文字変数である場合は、次の規則が適用されます。

- 符号は任意指定です。'+' または '-' を使用できます。数値データの前に付けることも、後ろに付けることもできます。
- 小数点は任意指定です。ピリオドでもコンマでもかまいません。
- データ内では先行ブランクと末尾ブランクが使用できます。例えば、' + 3 ' は有効なパラメーターです。
- 値をすべてブランクにすると、値ゼロが返されます。
- 無効な数値データが検出された場合、CPF0818 の例外が発生します。

結果は整数フォーマットとなり、小数部分の桁は丸めなしで切り捨てられます。結果のサイズは常に 4 バイトです。

以下は、%INT 組み込み関数の使用例です。

- 文字変数の変換

```
DCL VAR(&POINTS) TYPE(*CHAR) LEN(10) VALUE('-123.45')
DCL VAR(&ANSWER) TYPE(*INT)

/* &ANSWER will have the value -23 */
CHGVAR VAR(&ANSWER) VALUE(100 + %INT(&POINTS))
```

- 論理変数の変換

```
DCL VAR(&ANSWER1) TYPE(*LGL) VALUE('1')
DCL VAR(&ANSWER2) TYPE(*LGL) VALUE('1')
DCL VAR(&NUM) TYPE(*INT)

/* &NUM will have the value 2. */
CHGVAR VAR(&NUM) VALUE(%INT(&ANSWER1) + %INT(&ANSWER2))
SNDPGMMSG MSG('The number of YES is' *BCAT %CHAR(&NUM))
```

- パック 10 進変数の変換

```
DCL VAR(&POINTS1) TYPE(*DEC) LEN(5 2) VALUE(100.23)
DCL VAR(&POINTS2) TYPE(*DEC) LEN(5 2) VALUE(100.45)

IF (%INT(&POINTS1) *EQ %INT(&POINTS2)) +
THEN(SNDPGMMSG ('The scores are the same!'))
```

- 符号なし整変数の変換

```
DCL VAR(&P1) TYPE(*INT) LEN(2)
DCL VAR(&P2) TYPE(*UINT) LEN(2) VALUE(1)

CHGVAR VAR(&P1) VALUE(%INT(&P2))
```

- 小数部分の桁は、丸めなしで切り捨てられます

```
DCL VAR(&STRING) TYPE(*CHAR) LEN(10) VALUE('-123.9')
DCL VAR(&ANSWER) TYPE(*INT)

/* &ANSWER will have the value -123 */
CHGVAR VAR(&ANSWER) VALUE(%INT(&STRING))
```

関連タスク:

198 ページの『変数の値の変更』

CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』

制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%LEN 組み込み関数

%LEN 組み込み関数は、CL 数値変数または CL 文字変数の桁数または文字数を返します。

%LEN 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%LEN は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 数値変数に対して CL 10 進変数の桁数を比較する際に、%LEN を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ
*DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %LEN を使用することができます。

長さ取得組み込み関数の形式は、以下のとおりです。

```
%LEN(variable-argument)
```

variable-argument はタイプが *CHAR、*DEC、*INT、または *UINT の CL 変数でなければなりません。

数値変数の場合、戻り値は変数の精度を表します。しかし、必ずしも実際の有効数字の桁数を表すとは限りません。2 バイトの *INT 変数または *UINT CL 変数の場合、戻り値は常に 5 です。4 バイトの *INT または *UINT CL 変数の場合、戻り値は常に 10 です。8 バイトの *INT CL 変数の場合、戻り値は常に 19 です。8 バイトの *UINT CL 変数の場合、戻り値は常に 20 です。

文字変数の場合、戻り値は文字数です。

以下は、%LEN 組み込み関数の使用例です。

- 数値変数の長さを取得します

```
DCL VAR(&NUM1) TYPE(*DEC)
DCL VAR(&NUM2) TYPE(*DEC) LEN(7 2)
DCL VAR(&NUM3) TYPE(*INT) LEN(4)
DCL VAR(&NUM4) TYPE(*UINT) LEN(2)
DCL VAR(&RTN) TYPE(*INT) LEN(2)

/* &RTN will have the value 15. */
CHGVAR VAR(&RTN) VALUE(%LEN(&NUM1))
/* &RTN will have the value 7. */
CHGVAR VAR(&RTN) VALUE(%LEN(&NUM2))
/* &RTN will have the value 10. */
CHGVAR VAR(&RTN) VALUE(%LEN(&NUM3))
/* &RTN will have the value 5. */
CHGVAR VAR(&RTN) VALUE(%LEN(&NUM4))
```

- 文字変数の長さを取得します

```
DCL VAR(&CHAR1) TYPE(*CHAR)
DCL VAR(&CHAR2) TYPE(*CHAR) LEN(20)
DCL VAR(&RTN) TYPE(*INT) LEN(2)

/* &RTN will have the value 32. */
CHGVAR VAR(&RTN) VALUE(%LEN(&CHAR1))
/* &RTN will have the value 20. */
CHGVAR VAR(&RTN) VALUE(%LEN(&CHAR2))
/* &RTN will have the value 52. */
CHGVAR VAR(&RTN) VALUE(%LEN(&CHAR1) + %LEN(&CHAR2))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%LOWER 組み込み関数

%LOWER 組み込み関数は、指定された引数と同じ長さの文字ストリングを、それぞれの大文字を対応する小文字に置き換えて戻します。

%LOWER 組み込み関数は、CL が文字式をサポートする場合はどこでも使用できます。%LOWER は、単独でも、より複雑な文字式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 文字変数を比較する前に、それらの CL 文字変数を小文字だけになるよう変換する際に、%LOWER を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *CHAR、*NAME、*SNAME、*CNAME、*PNAME、*GENERIC、*DATE、*TIME、または *X を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %LOWER を使用することができます。

小文字変換組み込み関数の形式は次のとおりです。

```
%LOWER(input-string [CCSID])
```

input-string はタイプが *CHAR の CL 変数でなければなりません。

CCSID パラメーターは任意指定で、デフォルトの値はジョブ CCSID です。*CCSID* は、入力ストリングのコード化文字セット ID (CCSID) が変換されるよう指定します。大/小文字の変換はこの *CCSID* に基づいて行われます。 *CCSID* を指定する場合、その値は CL 整変数、小数点以下の桁数がゼロの CL 10 進変数、もしくは小数点以下の桁数がゼロの数値リテラルのいずれかです。有効な値は次のとおりです。

- 0: ジョブの CCSID は、変換されるデータの CCSID を決定するために使用します。ジョブ CCSID が 65535 の場合には、デフォルト CCSID (DFTCCSID) ジョブ属性の CCSID が使用されます。
- 1-65533: この範囲内の有効な CCSID。

以下は、%LOWER 組み込み関数の使用例です。

- 小文字に変換

  ```
  DCL VAR(&STR) TYPE(*CHAR) LEN(12) VALUE('Hello World!')
  /* 'hello world!' is to be sent */
  SNDPGMMSG (%LOWER(&STR))
  ```

- CCSID に基づき小文字に変換

  ```
  /* define &STR as 'Hello World!' in CCSID 819 (ISO/ANSI Multilingual) */
  DCL VAR(&STR) TYPE(*CHAR) LEN(12) +
                VALUE(X'48656C6C6F20576F726C6421')
  /* &STR will have the value x'68656C6C6F20776F726C6421' ('hello world!') */
  CHGVAR VAR(&STR) VALUE(%LOWER(&STR 819))
  ```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%OFFSET 組み込み関数

オフセット組み込み関数 (%OFFSET または %OFS) を使用して、CL ポインター変数のオフセット部分を保管または変更できます。

CHGVAR コマンドでは、%OFFSET 関数は、以下の 2 つの方法で使用できます。

- 変数 (VAR パラメーター) の %OFFSET 関数を指定して、ポインター変数のオフセット部分を設定できます。
- 変数が変更される値 (VALUE パラメーター) の %OFFSET 関数を指定できます。

IF コマンドでは、式 (COND パラメーター) の中で %OFFSET 関数を指定することができ、4 バイト符号なし整数値のように取り扱われます。

オフセット組み込み関数の形式は次のとおりです。

```
%OFFSET(変数名)
```

または

%OFS(変数名)

以下の例では、ポインター変数 &P1 には、初期値が指定されておらず、NULL に初期設定されています。最初の CHGVAR コマンドは、整変数 &OFF1 の NULL ポインター値からのオフセットを保管します。コマンドは、エラーなしで実行され、NULL ポインターのオフセットが定義されていなくても &OFF1 の値はゼロになります。2 番目の CHGVAR コマンドは、&P1 をローカル文字変数 &C1 のアドレスに設定します。ポインター &P1 は、変数 &C1 のバイト 1 をポイントします。3 番目の CHGVAR コマンドは、&P1 のオフセット部分を、整変数 &OFF2 に保管します。オフセットは、現行プログラムの自動ストレージの開始または変数 &C1 の開始からではなく、現行スレッドのすべての自動変数のストレージの開始からです。4 番目の CHGVAR コマンドは、&OFF2 に保管された整変数数を取得し、100 を加算し、結果の数値を整変数 &OFF3 に保管します。5 番目の CHGVAR コマンドは、整変数 &OFF3 を使用してポインター &P1 のオフセット部分を変更します。ポインター &P1 は、変数 &C1 のバイト 101 をポイントします。6 番目の CHGVAR コマンドは、ポインター &P1 のオフセット部分を一時整変数に保管し、20 を減算して VALUE パラメーターの整数式を計算し、また、この計算値を使用して、ポインター &P1 のオフセット部分をリセットします。ポインター &P1 は、変数 &C1 のバイト 81 をポイントします。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
DCL  &C1  *CHAR 30000
DCL  &P1  *PTR
DCL  &P2  *PTR
DCL  &OFF1  *UINT 4
DCL  &OFF2  *UINT 4
DCL  &OFF3  *UINT 4
CHGVAR &OFF1 %OFFSET(&P1)               /* 1 */
CHGVAR &P1 %ADDRESS(&C1)                /* 2 */
CHGVAR &OFF2 %OFFSET(&P1)               /* 3 */
CHGVAR &OFF3 (&OFF2+100)                /* 4 */
CHGVAR %OFFSET(&P1) &OFF3               /* 5 */
CHGVAR %OFFSET(&P1) (%OFFSET(&P1)-20)   /* 6 */
ENDPGM
```

%OFFSET 組み込み関数は、テラスペース・ストレージをアドレス指定するポインター変数では使用できません。

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

関連情報:

CL コマンド検索プログラム

変数変更 (CHGVAR) コマンド

%PARMS 組み込み関数

パラメーター数リターン組み込み関数 (%PARMS) は、%PARMS が使用されたプログラムに渡されたパラメーターの数を戻します。

%PARMS 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。 %PARMS は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** コマンドまたは **WHEN** コマンドの COND パラメーターで CL 数値変数に対して、プログラムに渡されたパラメーターの数を比較する際に、%PARMS を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %PARMS を使用することができます。

パラメーター数リターン組み込み関数の形式は次のとおりです。

```
%PARMS()
```

バインド済み呼び出しで呼び出されたプログラム内で %PARMS が使用された場合、呼び出し側のプログラムまたはプロシージャーが最小限の操作記述子を渡さないと、%PARMS で戻される値は使用できません。 ILE CL コンパイラーまたは ILE RPG コンパイラーは常にそれを渡しますが、他の言語では渡しません。そのため、呼び出し側が別の ILE 言語で作成されている場合は、呼び出しで操作記述子を渡す必要があります。操作記述子が渡されない場合、%PARMS で戻された値は信頼できません。操作記述子が渡されなかったことをシステムが判別できる場合、%PARMS で戻される値は -1 になります。しかし、システムがこれを検出できないこともあり、その場合には、%PARMS で戻される値が、ゼロ以上の誤った値になる可能性があります。

以下に、パラメーター数リターン組み込み関数の例を示します。

- **CALL** によるプログラム呼び出しでプログラムが呼び出されたときに、パラメーター数を戻します。以下のプログラムの名前は、PARMSTESTS です。

```
                PGM        PARM(&PARM1 &PARM2 &PARM3)
                DCL        VAR(&PARM1) TYPE(*DEC) LEN(5 2)
                DCL        VAR(&PARM2) TYPE(*CHAR) LEN(10)
                DCL        VAR(&PARM3) TYPE(*INT)

                SELECT
                  WHEN       COND(%PARMS() *EQ 0) THEN(DO)
                    SNDPGMMSG MSG('0 parm was passed in')
                    GOTO     LP0
                  ENDDO
                  WHEN       COND(%PARMS() *EQ 1) THEN(DO)
                    SNDPGMMSG MSG('1 parm was passed in')
                    GOTO     LP1
                  ENDDO
                  WHEN       COND(%PARMS() *EQ 2) THEN(DO)
                    SNDPGMMSG MSG('2 parms were passed in')
                    GOTO     LP2
                  ENDDO
                  WHEN       COND(%PARMS() *EQ 3) THEN(DO)
                    SNDPGMMSG MSG('3 parms were passed in')
                    GOTO     LP3
                  ENDDO
                OTHERWISE  CMD(GOTO CMDLBL(LPE))
                ENDSELECT

     LP3:       SNDPGMMSG  MSG('parm3:' *BCAT %CHAR(&PARM3))
     LP2:       SNDPGMMSG  MSG('parm2:' *BCAT &PARM2)
     LP1:       SNDPGMMSG  MSG('parm1:' *BCAT %CHAR(&PARM1))
                RETURN
     LP0:       SNDPGMMSG  MSG('no parameter at all!')
                RETURN
     LPE:       SNDPGMMSG  MSG('error in %parm()')
                ENDPGM

                PGM
                DCL        VAR(&VARDEC) TYPE(*DEC) LEN(5 2) VALUE(123.45)
                DCL        VAR(&VARCHAR) TYPE(*CHAR) LEN(10) +
```

```
                    VALUE(OUTERPGM)
          DCL       VAR(&VARINT) TYPE(*INT) VALUE(-123)

          CALL      PGM(PARMSTESTS) PARM(&VARDEC &VARCHAR +
                    &VARINT)
          ENDPGM
```

上記プログラムの出力は、次のとおりです。

```
3 parms were passed in
parm3: -123
parm2: OUTERPGM
parm1: 123.45
```

- **CALLPRC** によるバインド済み呼び出しでプログラムが呼び出されたときに、パラメーター数を戻します。上記の例のプログラム PARMSTESTS を引き続き使用します。

```
          PGM
          DCL       VAR(&VARDEC) TYPE(*DEC) LEN(5 2) VALUE(123.45)
          DCL       VAR(&VARCHAR) TYPE(*CHAR) LEN(10) +
                    VALUE(OUTERPGM)
          DCL       VAR(&VARINT) TYPE(*INT) VALUE(-123)

          CALLPRC   PRC(PARMSTESTS) PARM((&VARDEC))
          ENDPGM
```

上記プログラムの出力は、次のとおりです。

```
1 parm was passed in
parm1: 123.45
```

- ILE C など、別の ILE 言語でプログラムが呼び出されたときに、パラメーター数を戻します。上記の例のプログラム PARMSTESTS を引き続き使用します。

```
#include <stdlib.h>
#include <stdio.h>
#include <decimal.h>

void PARMSTESTS (decimal(5,2)*, char*, int*);
#pragma descriptor (void PARMSTESTS (void, "", void))
int main(void)
{
  decimal(5,2) vardec = 123.45;
  char varchar[10] = "OUTERPGMC";
  int varint = -123;

  PARMSTESTS(&vardec, varchar, &varint);
  return 0;
}
```

上記プログラムの出力は、次のとおりです。

```
3 parms were passed in
parm3: -123
parm2: OUTERPGMC
parm1: 123.45
```

- **CALLPRC** によるバインド済み呼び出しでパラメーター・リストのどれかのパラメーターに特殊値 *OMIT を指定した場合、 *OMIT の数は、%PARMS によるパラメーター数リターンでカウントされます。PARMSTESTS の最初の実行ステートメントの前に **MONMSG MCH3601** を追加します。

```
          PGM
          DCL       VAR(&VARDEC) TYPE(*DEC) LEN(5 2) VALUE(123.45)
          DCL       VAR(&VARCHAR) TYPE(*CHAR) LEN(10) +
```

```
                    VALUE(OUTERPGM)
DCL         VAR(&VARINT) TYPE(*INT) VALUE(-123)

CALLPRC     PRC(PARMSTESTS) PARM((&VARDEC) (*OMIT) (*OMIT))
ENDPGM
```

上記プログラムの出力は、次のとおりです。

```
3 parms were passed in
Pointer not set for location referenced.
Pointer not set for location referenced.
parm1: 123.45
```

関連タスク:

198 ページの『変数の値の変更』

CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』

制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%SCAN 組み込み関数

スキャン組み込み関数 (%SCAN) は、ソース・ストリング の中の検索引数 の 1 桁目を戻し、またはそれが見つからない場合には 0 を戻します。

%SCAN 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%SCAN は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 数値変数に対する比較を行う際に、%SCAN を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %SCAN を使用することができます。

スキャン組み込み関数の形式は次のとおりです。

```
%SCAN(search-argument source-string [starting-position])
```

search-argument は、CL 文字変数または文字リテラルのいずれかでなければなりません。*source-string* には CL 文字変数または *LDA のいずれかを指定できます。*LDA が指定されていると、そのジョブのローカル・データ域のコンテンツに対してスキャン関数が実行されます。*starting-position* は任意指定で、デフォルト値は 1 です。検索はソース・ストリングの開始位置 (start) から始められます。開始位置が指定されている場合でも、この結果は常にソース・ストリング内の最初のバイトに対する相対的な値となります。開始位置を指定する場合、その値は CL 整変数、小数点以下の桁数がゼロの CL 10 進変数、もしくは小数点以下の桁数がゼロの数値リテラルのいずれかです。開始位置をゼロまたは負の値にすることはできません。開始位置がソース・ストリング変数またはローカル・データ域全体の長さを超えると、エラーが起こります。ローカル・データ域の長さは 1024 です。

以下に、スキャン組み込み関数の例を示します。

- 特定のストリングを検索します。ストリング「John」を検索対象として CL 変数 &FNAME がスキャンされます。変数 &FNAME の中にストリング「John」が見つからない場合、メッセージが送信されます。

注: スキャンには大/小文字の区別があります。このため、「John」を検索対象としてスキャンを行うと、&FNAME に値「JOHN」が含まれている場合は、正しい結果は戻されません。

```
PGM PARM(&FNAME)
DCL VAR(&FNAME) TYPE(*CHAR) LEN(10)
IF COND(%SCAN('John' &FNAME) *EQ 0) +
   THEN(SNDPGMMSG ('NOT FOUND!'))
```

- ストリング内の特定の位置から検索を行います。日付区切り記号を含む日付の値を取得し、月の値を検索して数値変数に入れるために、%SCAN 関数が複数回使用されます。最初の %SCAN で、日付と月の間の日付区切り記号の位置が検出されます。 2 番目の %SCAN は、最初の日付区切り記号の直後の文字から開始し、月と年の間にある日付区切り記号の位置を検出します。その後、サブストリング (%SST) 組み込み関数により、月の値が文字形式から 10 進変数に変換されます。

```
DCL VAR(&YYYYMMDD) TYPE(*CHAR) LEN(10) VALUE('2012/8/29')
DCL VAR(&DATSEP) TYPE(*CHAR) LEN(1) VALUE('/')
DCL VAR(&SPOS) TYPE(*UINT) LEN(2)
DCL VAR(&EPOS) TYPE(*UINT) LEN(2)
DCL VAR(&LEN) TYPE(*UINT) LEN(2)
DCL VAR(&MONTH) TYPE(*DEC) LEN(2)
CHGVAR VAR(&SPOS) VALUE(%SCAN(&DATSEP &YYYYMMDD))
CHGVAR VAR(&SPOS) VALUE(&SPOS + 1)
CHGVAR VAR(&EPOS) VALUE(%SCAN(&DATSEP &YYYYMMDD &SPOS))
CHGVAR VAR(&LEN) VALUE(&EPOS - &SPOS)
CHGVAR VAR(&MONTH) VALUE(%SST(&YYYYMMDD &SPOS &LEN))
```

- ローカル・データ域 (*LDA) の文字を検索します。ストリング「Escape」を検索対象文字として、ローカル・データ域 (LDA) のスキャンが LDA のバイト 1 から開始されます。このストリングが検出されると、LDA 内のストリングの位置が CL 変数 &POS に割り当てられます。LDA 全体を探してもストリング「Escape」が検出されなければ、値ゼロが変数 &POS に割り当てられます。

```
DCL VAR(&POS) TYPE(*UINT) LEN(2)
CHGVAR VAR(&POS) VALUE(%SCAN('Escape' *LDA))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%SIZE 組み込み関数

%SIZE 組み込み関数は、CL 変数によって占有されるバイト数を返します。

%SIZE 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%SIZE は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで 2 つの CL 変数が使用するバイト数を比較する際に、%SIZE を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %SIZE を使用することができます。

サイズ取得組み込み関数の形式は、以下のとおりです。

```
%SIZE(variable-argument)
```

variable-argument は CL 変数でなければなりません。

以下は、%SIZE 組み込み関数の使用例です。

- 数値変数のサイズを取得します

```
DCL VAR(&NUM1) TYPE(*DEC)
DCL VAR(&NUM2) TYPE(*DEC) LEN(7 2)
DCL VAR(&NUM3) TYPE(*INT)
DCL VAR(&NUM4) TYPE(*UINT) LEN(8)
DCL VAR(&RTN) TYPE(*INT) LEN(2)

/* &RTN will have the value 8. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&NUM1))
/* &RTN will have the value 4. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&NUM2))
/* &RTN will have the value 4. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&NUM3))
/* &RTN will have the value 8. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&NUM4))
```

- 文字変数のサイズを取得します

```
DCL VAR(&CHAR1) TYPE(*CHAR)
DCL VAR(&CHAR2) TYPE(*CHAR) LEN(20)
DCL VAR(&RTN) TYPE(*INT) LEN(2)

/* &RTN will have the value 32. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&CHAR1))
/* &RTN will have the value 20. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&CHAR2))
/* &RTN will have the value 52. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&CHAR1) + %SIZE(&CHAR2))
```

- 論理変数のサイズを取得します

```
DCL VAR(&LGL1) TYPE(*LGL)
DCL VAR(&RTN) TYPE(*INT) LEN(2)

/* &RTN will have the value 1. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&LGL1))
```

- ポインター変数のサイズを取得します

```
DCL VAR(&PTR1) TYPE(*PTR)
DCL VAR(&RTN) TYPE(*INT) LEN(2)

/* &RTN will have the value 16. */
CHGVAR VAR(&RTN) VALUE(%SIZE(&PTR1))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%SUBSTRING 組み込み関数

サブストリング組み込み関数 (%SUBSTRING または %SST) は、既存の文字ストリングのサブセットとしての文字ストリングを作成します。

変数変更 (CHGVAR) コマンドでは、変更したい変数 (VAR パラメーター)、または変数の変更後の新しい値 (VALUE パラメーター) の代わりに、 %SST 関数を指定することができます。また **If (IF)** コマンドでは、式の中で %SST 関数を指定することができます。

サブストリング組み込み関数の形式は次のとおりです。

```
%SUBSTRING(文字変数名 開始位置 長さ)
```

また、次の例に示すようにフォーマットすることもできます。

```
%SST(文字変数名 開始位置 長さ)
```

文字変数名の代わりに *LDA を指定して、ローカル・データ域の内容にサブストリング関数を実行するよう指示することができます。

サブストリング関数は、指定された CL 文字変数またはローカル・データ域の内容に基づいてサブストリングを作成します。サブストリングは指定された開始位置 (これは変数名であっても構いません) から始まり、指定された長さ (これも変数名であっても構いません) まで続けられます。開始位置および長さはどちらも、0 または負の値であってはなりません。開始位置とサブストリングの長さとの合計が、変数またはローカル・データ域全体の長さを超えると、エラーが起こります。ローカル・データ域の長さは 1024 です。

以下に、サブストリング組み込み関数の使用例を示します。

- 文字変数 &NAME の最初の 2 文字が IN であれば、プログラム INV210 が呼び出されます。そして &NAME の値全体が INV210 に渡され、 &ERRCODE の値は変わりません。 IN でない場合には、&ERRCODE の値は 99 に設定されます。

  ```
  DCL &NAME *CHAR VALUE(INVOICE)
  DCL &ERRCODE *DEC (2 0)
  IF (%SST(&NAME 1 2) *EQ 'IN') +
  THEN(CALL INV210 &NAME)
  ELSE CHGVAR &ERRCODE 99
  ```

- &A の最初の 2 文字が &B の最初の 2 文字に等しい場合には、プログラム CUS210 が呼び出されます。

  ```
  DCL &A *CHAR VALUE(ABC)
  DCL &B *CHAR VALUE(DEF)
  IF (%SST(&A 1 2) *EQ %SUBSTRING(&B 1 2)) +
  CALL CUS210
  ```

- 開始位置および長さの値は変数にすることができます。この例では、&X の中の &Y 文字目から始まる &Z の長さの部分が 123 に変更されます。

  ```
  CHGVAR %SST(&X &Y &Z) '123'
  ```

- 以下の CHGVAR コマンドを実行する前の &A の値が ABCDEFG である場合、&A の値は、

  ```
  CHGVAR %SST(&A 2 3) '123'
  ```

  コマンドの実行後に A123EFG になります。

- この例では、サブストリングの長さである 5 が、それと比較されるオペランド (YES) の長さを超えています。したがって、オペランドにはブランクが埋め込まれ、YESNO と YESbb (ただし、b はブランク) との間で比較が行われます。そして、この条件は偽となります。

  ```
  DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
  .
  .
  .
  IF (%SST (&NAME 1 5) *EQ YES) +
    THEN(CALL PROGA)
  ```

  サブストリングの方が他のオペランドより短い場合には、サブストリングにブランクが埋め込まれた上で比較が行われます。以下にその例を示します。

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
.
.
IF (%SST(&NAME 1 3 ) *EQ YESNO) THEN(CALL PROG)
```

YESbb (ただし、bb は 2 つのブランク) は YESNO に等しくないので、この条件は偽になります。

- 変数 &A の値が、ローカル・データ域の 1 から 10 文字目に入れられます。

  ```
  CHGVAR %SST(*LDA 1 10) &A
  ```

- ローカル・データ域の 1 から 3 文字目と定数 'XYZ' を連結した値が変数 &A の値に等しければ、PROCA が呼び出されます。例えば、ローカル・データ域の 1 から 3 文字目が 'ABC' で、変数 &A の値が ABCXYZ であった場合には、テストの結果は真になり、PROCA が呼び出されます。

  ```
  IF (((%SST*LDA 1 3) *CAT 'XYZ') *EQ &A) THEN(CALLPRC PROCA)
  ```

- このプロシージャーは文字変数 &NUMBER を走査して、先行ゼロがあればそれをすべてブランクに変更します。このプロシージャーは、メッセージを表示する前のフィールドの簡単な編集に使用することができます。

  ```
        DCL &NUMBER *CHAR LEN(5)
        DCL &X *DEC LEN(3 0) VALUE(1)
        .
        .
  LOOP:IF (%SST(&NUMBER &X 1) *EQ '0') DO
        CHGVAR (%SST(&NUMBER &X 1)) ' '  /* Blank out */
        CHGVAR &X (&X + 1)  /* Increment */
        IF (&X *NE 4) GOTO LOOP
        ENDDO
  ```

次のプロシージャーでは、50 文字のフィールド &INPUT から最初の文を見つけ出し、残りのテキストがあればそれをフィールド &REMAINDER に入れるために、サブストリング組み込み関数が使用されています。ただし、文は少なくとも 2 文字であり、かつ途中にピリオドが含まれていないものとします。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
      PGM (&INPUT &REMAINDER)   /*  SEARCH  */
      DCL &INPUT     *CHAR LEN(50)
      DCL &REMAINDER *CHAR LEN(50)
      DCL &X *INT  /* INDEX            */
      DCL &L *INT  /* REMAINING LENGTH */

      DOFORL:
      DOFOR &X 3 50
        IF (%SST(&INPUT &X 1) *EQ '.') THEN(DO)
           CHGVAR &L (50-&X)
           CHGVAR &X (&X+1)
           CHGVAR &REMAINDER %SST(&INPUT &X &L)
           LEAVE
           ENDDO
      ENDDO
ENDPGM
```

このプロシージャーは、3 文字目からピリオドの有無の検査を開始します。サブストリング関数は、&INPUT を 3 文字目から始めて 1 文字ずつ検査します。すなわち、最初は 3 文字目だけを検査するという点に注意してください (長さ 0 は指定できません)。 3 文字目がピリオドであれば、&INPUT の残りの長さが計算されます。そして、&X の値が残りの部分の先頭を示す値に変更され、 &INPUT の残りの部分が &REMAINDER に移されます。

3 文字目がピリオドでなければ、プロシージャーは 49 文字目に到達しているかどうかを調べます。 49 文字目に達している場合には、プロシージャーは 50 文字目がピリオドであると見なして戻ります。 49 文字目に到達していなければ、プロシージャーは &X を 4 文字目に進めて処理を繰り返します。

関連タスク:

196 ページの『リストまたは修飾名を指定するために使用する変数』
変数を使用してリストまたは修飾名を指定することができます。

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

関連情報:

CL コマンド検索プログラム

仮定 (IF) コマンド

変数変更 (CHGVAR) コマンド

%SWITCH 組み込み関数

スイッチ組み込み関数 (%SWITCH) は、8 個のスイッチの 1 つまたは複数のスイッチを、該当のジョブで既に設定されている 8 つのスイッチの値と比較し、 '0' または '1' の論理値を返します。

ジョブのスイッチの初期値は、**ジョブ記述作成 (CRTJOBD)** コマンドによって決定されます。デフォルト値は 00000000 です。スイッチの値は、必要があれば、**ジョブ投入 (SBMJOB)**、**ジョブ変更 (CHGJOB)**、または JOB コマンドの SWS パラメーターを使用して変更することができます。これらのコマンドでは、ジョブ記述でのスイッチ設定がデフォルト値になります。他の高水準言語でもジョブ・スイッチを設定することができます。

%SWITCH 値とジョブ値を比較して、すべてのスイッチが同じであった場合は、論理値 '1' が戻されます。テストしたスイッチのいずれかが指定した値と異なる場合には、結果は '0' (偽) になります。

%SWITCH 組み込み関数の構文は次のとおりです。

```
%SWITCH (8 文字のマスク)
```

8 文字のマスクには、どのジョブ・スイッチをテストするのか、および各スイッチのどのような値についてテストするのか指定します。マスクの各桁は、ジョブの 8 つのジョブ・スイッチの 1 つ 1 つに対応しています。すなわち、1 桁目はスイッチ 1 に、そして 2 桁目はスイッチ 2 に、というように対応しています。マスクの各桁には、3 つの値、0、1、または X のいずれかを指定できます。

0 対応するジョブ・スイッチが 0 (オフ) であるかどうかをテストします。

1 対応するジョブ・スイッチが 1 (オン) であるかどうかをテストします。

X 対応するジョブ・スイッチはテストしません。スイッチの値は、%SWITCH の結果に影響しません。

%SWITCH(0X111XX0) を指定した場合、ジョブ・スイッチ 1 および 8 が 0 であるかどうかがテストされ、スイッチ 3、4、および 5 が 1 であるかどうかがテストされますが、スイッチ 2、6、および 7 はテスト

されません。 テストの対象すべてのジョブ・スイッチが、マスクで指定した値 (1 または 0 のどちらか) と同じであれば、 %SWITCH の結果は '1'、すなわち真になります。

CL プログラムまたはプロシージャーの中では、スイッチのテストによりプログラムまたはプロシージャーのフローを制御することができます。この関数は、CL プログラムまたはプロシージャーの中で **If (IF)** および**変数変更 (CHGVAR)** コマンドとともに使用されます。また、CL プログラムまたはプロシージャーの中で**ジョブ変更 (CHGJOB)** コマンドを用いてスイッチを変更することができます。 CL プログラムまたはプロシージャーの場合、このような変更は直ちに有効になります。

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

関連情報:

CL コマンド検索プログラム

ジョブ変更 (CHGJOB) コマンド

ジョブ投入 (SBMJOB) コマンド

ジョブ記述変更 (CHGJOBD) コマンド

変数変更 (CHGVAR) コマンド

IF コマンドでの %SWITCH:

If (IF) コマンドでは、テストされる論理式として %SWITCH を COND パラメーターに指定することができます。

次の例では、0X111XX0 が事前設定されているジョブ・スイッチの値と比較されます。

```
IF  COND(%SWITCH(0X111XX0))  THEN(GOTO C)
```

ジョブ・スイッチ 1 に 0 が含まれ、ジョブ・スイッチ 3 に 1 が含まれ、ジョブ・スイッチ 4 に 1 が含まれ、ジョブ・スイッチ 5 に 1 が含まれ、ジョブ・スイッチ 8 に 0 が含まれる場合は、結果は真になり、プロシージャーは C というラベルを持つコマンドに分岐します。テストしたスイッチに、マスクで指定した値に一致しないスイッチが 1 つでもあれば結果は偽になり、分岐は行われません。

次の例では、2 つのプロシージャーのスイッチによって制御される条件付き処理を示しています。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
SBMJOB  JOB(APP502) JOBD(PAYROLL) CMD(CALL APP502)
    SWS(11000000)

PGM  /* CONTROL */
IF (%SWITCH(11XXXXXX)) CALLPRC PROCA
IF (%SWITCH(10XXXXXX)) CALLPRC PROCB
IF (%SWITCH(01XXXXXX)) CALLPRC PROCC
IF (%SWITCH(00XXXXXX)) CALLPRC PROCD
ENDPGM

PGM  /* PROCA */
```

```
CALLPRC TRANS
IF (%SWITCH(1XXXXXXX)) CALLPRC CUS520
ELSE CALLPRC CUS521
ENDPGM
```

関連情報:

仮定 (IF) コマンド

変数変更 コマンドでの **%SWITCH:**

変数変更 (CHGVAR) コマンドに %SWITCH を指定することにより、論理変数の値を変更することができます。

論理変数の値は、 %SWITCH の設定値とジョブ・スイッチの設定値との比較の結果により決まります。比較の結果が真であれば、論理変数は '1' に設定されます。比較の結果が偽であれば、変数は '0' に設定されます。例えば、ジョブ・スイッチの設定が 10000001 である場合に、次のプロシージャーが実行されたとします。すると変数 &A は '1' の値を持ちます。

```
PGM
DCL &A *LGL
CHGVAR VAR(&A) VALUE(%SWITCH(10000001))
.
.
.
ENDPGM
```

関連情報:

変数変更 (CHGVAR) コマンド

%TRIM 組み込み関数

1 つのパラメーターを指定してトリム組み込み関数 (%TRIM) を使用すると、先行ブランクおよび末尾ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定してトリム組み込み関数 (%TRIM) を使用すると、*characters-to-trim* パラメーター内にある先行文字と末尾文字がすべて除去された文字ストリングが生成されます。

%TRIM 組み込み関数は、CL が文字式をサポートする場合はどこでも使用できます。%TRIM は、単独でも、より複雑な文字式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 文字変数に対する比較を行う際に、%TRIM を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ
*CHAR、*NAME、*SNAME、*CNAME、*PNAME、*GENERIC、*DATE、*TIME、または *X を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %TRIM を使用することができます。

トリム組み込み関数の形式は、以下のとおりです。

```
%TRIM(character-variable-name [characters-to-trim])
```

トリム関数は、指定された CL 文字変数の内容に基づいてサブストリングを作成します。
characters-to-trim パラメーターが指定されている場合、その値は CL 文字変数または文字リテラルのいずれかでなければなりません。トリム後に何も文字が残らない場合、トリム関数によりブランク文字のストリングが生成されます。

以下に、トリム組み込み関数の例を示します。

- 先行ブランク文字と末尾ブランク文字を切り取ります。CL 変数 &FIRSTNAME および &LASTNAME から先行ブランクと末尾ブランクが切り取られ、結果のストリングは、*BCAT 演算子によりその 2 つの値の間の 1 つのブランクで連結されます。その後、連結されたストリングが CL 変数 &NAME に割り当てられます。

```
DCL VAR(&FIRSTNAME) TYPE(*CHAR) VALUE('   JOHN  ')
DCL VAR(&LASTNAME) TYPE(*CHAR) VALUE('   SMITH  ')
DCL VAR(&NAME) TYPE(*CHAR) LEN(10)
CHGVAR VAR(&NAME) VALUE(%TRIM(&FIRSTNAME) *BCAT %TRIM(&LASTNAME))
```

- リテラル・ストリング内の指定された文字を切り取ります。CL 変数 &NUM の先頭と末尾からアスタリスクとブランクがすべて切り取られ、残りの文字 (1.23) が CL 変数 &TRIMMED に割り当てられます。文字変数 &TRIMMED が数値に変換され、10 進変数 &DEC に割り当てられます。

```
DCL VAR(&NUM) TYPE(*CHAR) LEN(10) VALUE('* *1.23* *')
DCL VAR(&TRIMMED) TYPE(*CHAR) LEN(10)
DCL VAR(&DEC) TYPE(*DEC) LEN(3 2)
CHGVAR &TRIMMED %TRIM(&NUM '* ')
CHGVAR VAR(&DEC) VALUE(&TRIMMED)
```

- CL 変数内で指定された文字を切り取ります。変数 &NAME の最初の文字から開始して、変数 &TCHAR 内のいずれかの文字と一致する文字が切り取られます。また、変数 &NAME の最後の文字から開始して、変数 &TCHAR 内のいずれかの文字と一致する文字が切り取られます。この結果の &NAME のサブストリングがリテラル・ストリング '12345' と比較され、それらの値が等しければメッセージが送信されます。

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(10) VALUE('*+12345   ')
DCL VAR(&TCHAR) TYPE(*CHAR) LEN(3) VALUE('*+ ')
IF COND(%TRIM(&NAME &TCHAR) *EQ '12345') +
   THEN(SNDPGMMSG ('EQUAL!'))
```

関連タスク:

198 ページの『変数の値の変更』

CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』

制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%TRIML 組み込み関数

1 つのパラメーターを指定して左方トリム組み込み関数 (%TRIML) を使用すると、先行ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定して左方トリム組み込み関数 (%TRIML) を使用すると、*characters-to-trim* パラメーター内にある先行文字がすべて除去された文字ストリングが生成されます。

%TRIML 組み込み関数は、CL が文字式をサポートする場合はどこでも使用できます。%TRIML は、単独でも、より複雑な文字式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 文字変数に対する比較を行う際に、%TRIML を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ
*CHAR、*NAME、*SNAME、*CNAME、*PNAME、*GENERIC、*DATE、*TIME、または *X を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %TRIML を使用することができます。

左方トリム組み込み関数の形式は、以下のとおりです。

```
%TRIML(character-variable-name [characters-to-trim])
```

左方トリム関数は、指定された CL 文字変数の内容に基づいてサブストリングを生成します。
characters-to-trim パラメーターが指定されている場合、その値は CL 文字変数または文字リテラルのいずれかでなければなりません。トリム後に何も文字が残らない場合、左方トリム関数によりブランク文字のストリングが生成されます。

以下に、左方トリム組み込み関数の例を示します。

- 先行ブランク文字を切り取ります。CL 変数 &NUMCHAR から先行ブランクが切り取られ、結果のストリングは数値に変換されて CL 変数 &NUMDEC に割り当てられます。

```
DCL VAR(&NUMCHAR) TYPE(*CHAR) VALUE('     00001')
DCL VAR(&NUMDEC) TYPE(*DEC) LEN(5)
CHGVAR VAR(&NUMDEC) VALUE(%TRIML(&NUMCHAR))
```

- リテラル・ストリング内の指定された先行文字を切り取ります。CL 変数 &PRICE の先頭からドル記号とブランクがすべて切り取られ、残りの文字 (5.27) が CL 変数 &TRIMMED に割り当てられます。文字変数 &TRIMMED が数値に変換され、10 進変数 &DEC に割り当てられます。

```
DCL VAR(&PRICE) TYPE(*CHAR) VALUE('     $5.27')
DCL VAR(&TRIMMED) TYPE(*CHAR) LEN(5)
DCL VAR(&DEC) TYPE(*DEC) LEN(3 2)
CHGVAR VAR(&TRIMMED) VALUE(%TRIML(&PRICE '$ '))
CHGVAR VAR(&DEC) VALUE(&TRIMMED)
```

- CL 変数内で指定された先行文字を切り取ります。変数 &NAME の最初の文字から開始して、変数 &TCHAR 内のいずれかの文字と一致する文字が切り取られます。この結果の &NAME のサブストリングがリテラル・ストリング'12345'と比較され、それらの値が等しければメッセージが送信されます。

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(10) VALUE('    +12345')
DCL VAR(&TCHAR) TYPE(*CHAR) LEN(2) VALUE('+ ')
IF COND(%TRIML(&NAME &TCHAR) *EQ '12345') +
   THEN(SNDPGMMSG ('EQUAL!'))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%TRIMR 組み込み関数

1 つのパラメーターを指定して右方トリム組み込み関数 (%TRIMR) を使用すると、末尾ブランクがすべて除去された文字ストリングが生成されます。 2 つのパラメーターを指定して右方トリム組み込み関数 (%TRIMR) を使用すると、*characters-to-trim* パラメーター内にある末尾文字がすべて除去された文字ストリングが生成されます。

%TRIMR 組み込み関数は、CL が文字式をサポートする場合はどこでも使用できます。%TRIMR は、単独でも、より複雑な文字式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 文字変数に対する比較を行う際に、%TRIMR を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ
*CHAR、*NAME、*SNAME、*CNAME、*PNAME、*GENERIC、*DATE、*TIME、または *X を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %TRIMR を使用することができます。

右方トリム組み込み関数の形式は、以下のとおりです。

```
%TRIMR(character-variable-name [characters-to-trim])
```

右方トリム関数は、指定された CL 文字変数の内容に基づいてサブストリングを生成します。characters-to-trim パラメーターが指定されている場合、その値は CL 文字変数または文字リテラルのいずれかでなければなりません。トリム後に何も文字が残らない場合、右方トリム関数によりブランク文字のストリングが生成されます。

以下に、右方トリム組み込み関数の例を示します。

- 末尾ブランク文字を切り取ります。CL 変数 &FIRSTNAME および &LASTNAME から末尾ブランク文字が切り取られ、結果のストリングは *CAT 演算子により連結されます。その後、連結されたストリングが CL 変数 &NAME に割り当てられます。

  ```
  DCL VAR(&FIRSTNAME) TYPE(*CHAR) LEN(10) VALUE('JOHN      ')
  DCL VAR(&LASTNAME) TYPE(*CHAR) LEN(10) VALUE('SMITH     ')
  DCL VAR(&NAME) TYPE(*CHAR) LEN(10)
  CHGVAR VAR(&NAME) VALUE(%TRIM(&FIRSTNAME) *CAT %TRIM(&LASTNAME))
  ```

- リテラル・ストリング内の指定された末尾文字を切り取ります。CL 変数 &PRICE から末尾のゼロと正符号文字がすべて切り取られ、残りの文字 (5.27) が CL 変数 &TRIMMED に割り当てられます。文字変数 &TRIMMED が数値に変換され、10 進変数 &DEC に割り当てられます。

  ```
  DCL VAR(&PRICE) TYPE(*CHAR) LEN(10) VALUE('5.2700000+')
  DCL VAR(&TRIMMED) TYPE(*CHAR) LEN(5)
  DCL VAR(&DEC) TYPE(*DEC) LEN(3 2)
  CHGVAR VAR(&TRIMMED) VALUE(%TRIMR(&PRICE '0+'))
  CHGVAR VAR(&DEC) VALUE(&TRIMMED)
  ```

- CL 変数内で指定された末尾文字を切り取ります。変数 &NAME の最後の文字から開始して、変数 &TCHAR 内のいずれかの文字と一致する文字が切り取られます。この結果の &NAME のサブストリングがリテラル・ストリング '12345' と比較され、それらの値が等しければメッセージが送信されます。

  ```
  DCL VAR(&NAME) TYPE(*CHAR) LEN(10) VALUE('12345***  ')
  DCL VAR(&TCHAR) TYPE(*CHAR) LEN(2) VALUE('* ')
  IF COND(%TRIMR(&NAME &TCHAR) *EQ '12345') +
     THEN(SNDPGMMSG ('EQUAL!'))
  ```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%UINT 組み込み関数

%UINT は、文字、論理、10 進数、または整数データを符号なし整数フォーマットに変換します。変換された値は、CL 変数に割り当てたり、数値定数として別のプログラムまたはプロシージャーに渡したり、コンパイル済み CL から実行される CL コマンドのコマンド・パラメーターに対する値として指定したりすることができます。

%UINT 組み込み関数は、CL が算術式をサポートする場合はどこでも使用できます。%UINT は、単独でも、より複雑な算術式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで符号なし CL 整数変数に対して CL 10 進変数を比較する際に、%UINT を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ

*DEC、*INT2、*INT4、*UINT2、または *UINT4 を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %UINT を使用することができます。

符号なし整数データへの変換組み込み関数の形式は次のとおりです。

```
%UINT(convert-argument)
```

また、次のようにフォーマットすることもできます。

```
%UNS(convert-argument)
```

convert-argument はタイプが *CHAR、*LGL、*DEC、または *INT の CL 変数でなければなりません。

convert-argument パラメーターが文字変数である場合は、次の規則が適用されます。

- 符号は任意指定です。使用できるのは '+' のみです。数値データの前に付けることも、後ろに付けることもできます。
- 小数点は任意指定です。ピリオドでもコンマでもかまいません。
- データ内では先行ブランクと末尾ブランクが使用できます。例えば、' + 3 ' は有効なパラメーターです。
- 値をすべてブランクにすると、値ゼロが返されます。
- 無効な数値データが検出された場合、CPF0818 の例外が発生します。

結果は符号なし整数フォーマットとなり、小数部分の桁はすべて丸めなしで切り捨てられます。結果のサイズは常に 4 バイトです。

以下は、%UINT 組み込み関数の使用例です。

注: *%UNS は RPG との互換性のために使用できますが、コーディング例は %UINT を使用します。

- 文字変数の変換

```
DCL VAR(&POINTS) TYPE(*CHAR) LEN(10) VALUE('+123.45')
DCL VAR(&ANSWER) TYPE(*UINT)

/* &ANSWER will have the value 223 */
CHGVAR VAR(&ANSWER) VALUE(100 + %UINT(&POINTS))
```

- 論理変数の変換

```
DCL VAR(&ANSWER1) TYPE(*LGL) VALUE('1')
DCL VAR(&ANSWER2) TYPE(*LGL) VALUE('1')
DCL VAR(&NUM) TYPE(*UINT)

/* &NUM will have the value 2. */
CHGVAR VAR(&NUM) VALUE(%UINT(&ANSWER1) + %UINT(&ANSWER2))
SNDPGMMSG MSG('The number of YES answers is' *BCAT %CHAR(&NUM))
```

- パック 10 進変数の変換

```
DCL VAR(&POINTS1) TYPE(*DEC) LEN(5 2) VALUE(100.23)
DCL VAR(&POINTS2) TYPE(*DEC) LEN(5 2) VALUE(100.45)

IF (%UINT(&POINTS1) *EQ %UINT(&POINTS2)) +
THEN(SNDPGMMSG ('The scores are the same!'))
```

- 整変数の変換

```
DCL VAR(&P1) TYPE(*UINT) LEN(2)
DCL VAR(&P2) TYPE(*INT) LEN(2) VALUE(1)

CHGVAR VAR(&P1) VALUE(%UINT(&P2))
```

- 小数部分の桁は、丸めなしで切り捨てられます

```
DCL VAR(&STRING) TYPE(*CHAR) LEN(10) VALUE('+123.9')
DCL VAR(&ANSWER) TYPE(*UINT)

/* &ANSWER will have the value 123 */
CHGVAR VAR(&ANSWER) VALUE(%UINT(&STRING))
```

関連タスク:

198 ページの『変数の値の変更』

CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』

制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

%UPPER 組み込み関数

%UPPER 組み込み関数は、指定された引数と同じ長さの文字ストリングを、それぞれの小文字を対応する大文字に置き換えて戻します。

%UPPER 組み込み関数は、CL が文字式をサポートする場合はどこでも使用できます。%UPPER は、単独でも、より複雑な文字式の一部としても使用可能です。例えば、**IF** または **WHEN** コマンドの COND パラメーターで CL 文字変数を比較する前に、それらの CL 文字変数を大文字だけになるよう変換する際に、%UPPER を使用することができます。また、関連するコマンド・オブジェクトが EXPR(*YES) およびタイプ *CHAR、*NAME、*SNAME、*CNAME、*PNAME、*GENERIC、*DATE、*TIME、または *X を指定してパラメーターを定義する場合、CL コマンド・パラメーターの値を設定する際にも %UPPER を使用することができます。

大文字変換組み込み関数の形式は次のとおりです。

```
%UPPER(input-string [CCSID])
```

input-string はタイプが *CHAR の CL 変数でなければなりません。

CCSID パラメーターは任意指定で、デフォルトの値はジョブ CCSID です。*CCSID* は、入力ストリングのコード化文字セット ID (CCSID) が変換されるよう指定します。大/小文字の変換はこの *CCSID* に基づいて行われます。 *CCSID* を指定する場合、その値は CL 整変数、小数点以下の桁数がゼロの CL 10 進変数、もしくは小数点以下の桁数がゼロの数値リテラルのいずれかです。有効な値は次のとおりです。

- 0: ジョブの CCSID は、変換されるデータの CCSID を決定するために使用します。ジョブ CCSID が 65535 の場合には、デフォルト CCSID (DFTCCSID) ジョブ属性の CCSID が使用されます。
- 1-65533: この範囲内の有効な CCSID。

以下は、%UPPER 組み込み関数の使用例です。

- 大文字に変換

```
DCL VAR(&STR) TYPE(*CHAR) LEN(12) VALUE('Hello World!')
/* 'HELLO WORLD!' is to be sent */
SNDPGMMSG (%UPPER(&STR))
```

- CCSID に基づき大文字に変換

```
/* define &STR as 'Hello World!' in CCSID 819 (ISO/ANSI Multilingual) */
DCL VAR(&STR) TYPE(*CHAR) LEN(12) +
              VALUE(X'48656C6C6F20576F726C6421')
/* &STR will have the value x'48454C4C4F20574F524C4421' ('HELLO WORLD!') */
CHGVAR VAR(&STR) VALUE(%UPPER(&STR 819))
```

関連タスク:

198 ページの『変数の値の変更』
CL 変数の値は、**変数変更 (CHGVAR)** コマンドによって変更することができます。

関連資料:

115 ページの『CL の組み込み関数』
制御言語 (CL) は、いくつかの組み込み関数を備えています。組み込み関数は、算術式、文字ストリング式、比較式、または論理式で使用されます。これらの組み込み関数はすべて、コンパイル済み CL プログラムまたはプロシージャーでのみ使用できます。

メッセージ・モニター コマンド

メッセージ・モニター (MONMSG) コマンドは、**MONMSG** コマンドが使用されている CL プログラムまたはプロシージャーの呼び出しスタックに送られてくる、エスケープ・メッセージ、通知メッセージ、または状況メッセージの監視に使用されます。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

エスケープ・メッセージは、CL プログラムまたはプロシージャーの中のコマンド、および CL プログラムまたはプロシージャーによって呼び出されたプログラムやプロシージャーから、CL プログラムまたはプロシージャーに送られます。エスケープ・メッセージは、エラーが検出されたために要求された機能を実行できなかったことをプログラムまたはプロシージャーに伝えるためのものです。 CL プログラムまたは CL プロシージャーはエスケープ・メッセージの到着を監視することができ、ユーザーはそのようなメッセージに対してどのような処置をとるかをコマンドによって指定することができます。例えば、ある CL プログラムまたは CL プロシージャーが既に削除されているデータ域を移動しようとしたとすると、 **オブジェクト移動 (MOVOBJ)** コマンドからプログラムまたはプロシージャーに、オブジェクトが見つからないことを示すエスケープ・メッセージが送られます。

MONMSG コマンドを使用すると、その直前のコマンドの処理によって特定のエラーが生じた場合にとるべき処置を、あらかじめプログラムまたはプロシージャーに指定しておくことができます。 **MONMSG** コマンドには次のパラメーターがあります。

```
MONMSG   MSGID(メッセージ識別コード) CMPDTA(比較データ) +
         EXEC(CL-command)
```

特定のエラーについて送られる各メッセージには、それぞれ固有の識別コードが付いています。 MSGID パラメーターには、最高 50 個のメッセージ識別コードを指定することができます (メッセージおよび ID については、オンライン・ヘルプを参照してください)。 CMPDTA パラメーターを使用すれば、メッセージの MSGDTA 部分に特定の文字ストリングがあるかどうかを調べることが可能で、具体的で高度なエラー・メッセージの指定ができます。 EXEC パラメーターには、プログラムまたはプロシージャーでエラーからの回復処置を行うための CL コマンド (プログラム呼び出し (CALL)、実行 (DO)、または指定先に進む (GOTO) など) を指定することができます。

次の例では、**MONMSG** コマンドは**ファイル受信 (RCVF)** コマンドの次にあり、したがって RCVF コマンドから送られるメッセージだけを監視します。

```
READLOOP:  RCVF                          /* Read a file record  */
           MONMSG  MSGID(CPF0864)  EXEC(GOTO CMDLBL(EOF))
           /*  Process the file record  */
           GOTO  CMDLBL(READLOOP)       /*  Get another record  */
EOF:       /*  End of file processing  */
```

エスケープ・メッセージ CPF0864 は、読み取るレコードがファイルにもう存在しない場合に、プログラムまたはプロシージャーの呼び出し待ち行列に送られます。上記の例では MSGID (CPF0864) を指定しているため、MONMSG はこの条件を監視します。メッセージを受け取ると、**GOTO CMDLBL(EOF)** コマンドが実行されます。

MONMSG コマンドを使用して、CL プログラムまたはプロシージャー内のすべてのコマンドによって送られるメッセージを監視することもできます。次の例には 2 つの **MONMSG** コマンドが含まれています。最初の **MONMSG** コマンドは、メッセージ CPF0001 および CPF1999 を監視します。これらのメッセージは、プログラムまたはプロシージャー内の、このコマンド以降に実行されるいずれかのコマンドから送られる可能性があります。プログラムまたはプロシージャーの中で実行されるコマンドのいずれかから、このどちらかのメッセージを受け取ると、制御はラベル EXIT2 を持つコマンドに分岐します。

2 番目の **MONMSG** コマンドは、メッセージ CPF2105 および MCH1211 を監視します。EXEC パラメーターにはコマンドがコーディングされていないので、これらのメッセージはすべて無視されます。

```
PGM
DCL
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
MONMSG MSGID(CPF2105 MCH1211)
.
.
.
ENDPGM
```

メッセージ CPF0001 は、そのメッセージで示されているコマンドでエラーが検出されたことを知らせるものです。メッセージ CPF1999 は、例えば**プログラム変数変更 (CHGPGMVAR)** などのような多くのデバッグ用のコマンドから送られるものであり、そのコマンドにエラーが起こったことを示すものですが、メッセージでは該当のコマンドは識別されません。

EXEC パラメーターが指定された **MONMSG** コマンドにより監視されるすべてのエラー状態 (CPF0001 または CPF1999) は、EXIT2 で同じ方法により処理され、エラー後の次の順番のステートメントに戻ることはできません。これを避けるには、各コマンドの後でそれぞれ特定の条件に関する監視を行い、それぞれ特定のエラー訂正プログラムまたはプロシージャーに分岐する方法を取ることができます。

EXEC パラメーターの指定がない **MONMSG** コマンドにより監視されるエラー状態 (CPF2105 または MCH1211) はすべて無視され、次のコマンドについてプログラムまたはプロシージャー処理が続けられます。

IF コマンドでエラーが起こった場合には、その条件は偽であると見なされます。次の例では、IF コマンドで MCH1211 (ゼロ除算) が生じる可能性があります。その場合には条件は偽であると見なされ、PROCA が呼び出されます。

```
IF(&A / &B *EQ 5) THEN(DLTF ABC)
ELSE CALLPRC PROCA
```

CL プログラムまたはプロシージャーの最初の部分に **MONMSG** コマンドを指定した場合は、指定したメッセージは、どのコマンドから出されたかに関係なく、プログラム全体にわたり監視されます。 EXEC パラメーターを使用した場合、指定することができるのは GOTO コマンドだけです。 GOTO コマンドがユーザー・プログラム内で実行された場合、サブルーチン・スタックがリセットされます。

同じメッセージ ID を、プロシージャー・レベルまたはコマンド・レベルの **MONMSG** コマンドに指定することもできます。その場合は、コマンド・レベルの **MONMSG** コマンドが、プロシージャー・レベルの MONMSG コマンドより優先されます。次の例で、CMDB の実行でメッセージ CPF0001 を受け取ったとすれば、CMDC が実行されます。プログラムまたはプロシージャー内のその他のコマンドの実行によって

メッセージ CPF0001 を受け取った場合は、プログラムまたはプロシージャーは EXIT2 に分岐します。CMDB も含めて、いずれかのコマンドの実行によってメッセージ CPF1999 を受け取った場合は、プログラムまたはプロシージャーは EXIT2 に分岐します。

```
PGM
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
CMDA
CMDB
MONMSG MSGID(CPF0001) EXEC(CMDC)
CMDD
EXIT2:  ENDPGM
```

プログラムまたはプロシージャーにはさまざまなエスケープ・メッセージが送られる可能性があるので、それらのどれを監視し処理するかを決める必要があります。これらのメッセージのほとんどは、プログラムまたはプロシージャーでエラーが生じた場合に限り、プログラムまたはプロシージャーに送られますが、プログラムまたはプロシージャーの外部で生じた状態が原因で送られるものもあります。一般に、CL プログラムまたはプロシージャーでは、その基本的な機能に関連し、処理できるメッセージについて、監視を行う必要があります。その他のメッセージについては、IBM i オペレーティング・システムは、エラーが起こったものと見なしてデフォルトの処置を取ります。

関連タスク:

537 ページの『メッセージ記述の定義』
事前定義メッセージは、メッセージ・ファイルに保管されます。

536 ページの『メッセージ』
メッセージは、ユーザーとプログラム間の通信に使用されます。

518 ページの『オブジェクトの存在の検査』
プログラムでオブジェクトを使用する前に、そのオブジェクトが存在しているかどうか、およびそれを使用するために必要な権限があるかどうかを判別するための検査を行ってください。

関連情報:

CL コマンド検索プログラム

メッセージ・モニター (MONMSG) コマンド

変数として使用可能な値の検索

システム値およびジョブ属性などの値を検索して、プログラムまたはプロシージャー内で変数として使用することができます。すべての検索可能な値が含まれているわけではないことに注意してください。

関連情報:

CL コマンド検索プログラム

システム値検索

システム値には、システムの特定機能の操作に関する制御情報が入っています。

IBM 提供のシステム値には、いくつかのタイプがあります。例えば、IBM i オペレーティング・システムの始動の時点でユーザーが設定する日付と時刻のシステム値として、QDATE と QTIME があります。

システム値検索 (RTVSYSVAL) コマンドを使用してシステム値をプログラムまたはプロシージャーで検索し、変数としてシステム値を取り扱うことができます。

```
RTVSYSVAL  SYSVAL(システム値名)  RTNVAR(CL 変数名)
```

戻り値 (RTNVAR) パラメーターの CL 変数には、指定のシステム値の値を検索して入れる CL プログラムまたはプロシージャーの変数の名前を指定します。

変数のタイプはシステム値のタイプに一致していなければなりません。また、文字システム値および論理システム値の場合には、CL 変数の長さはそのシステム値の長さと同じでなければなりません。 10 進数値の場合には、変数の長さはシステム値の長さに等しいかそれより大きくなければなりません。

関連情報:

システム値

例: **QTIME** システム値の検索:

次の例では、QTIME の値を受け取りその値を変数に入れ、それをさらに別の変数の値と比較しています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
DCL  VAR(&PWRDNTME)  TYPE(*CHAR)  LEN(6)  VALUE('162500')
DCL  VAR(&TIME)  TYPE(*CHAR)  LEN(6)
RTVSYSVAL  SYSVAL(QTIME)  RTNVAR(&TIME)
IF  (&TIME *GT &PWRDNTME)  THEN(DO)
SNDBRKMSG('Powering down in 5 minutes.  Please sign off.')
PWRDWNSYS  OPTION(*CNTRLD)  DELAY(300)  RESTART(*NO)  +
          IPLSRC(*PANEL)

ENDDO
ENDPGM
```

関連情報:

システム値

QDATE システム値を検索して **CL** 変数に入れる:

多くのアプリケーションの場合、システム値 QDATE を検索し、それを変数に入れることにより、プログラムまたはプロシージャーで当日の日付として使用したい場合があります。また、その日付の形式を変えて使用したいこともあります。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

CL プログラムまたはプロシージャーで日付の形式を変更するには、 **日付形式変換 (CVTDAT)** コマンドを使用します。

システム日付の形式はシステム値 QDATFMT の値によって決まります。含まれる QDATFMT の値は、国や地域によって異なります。例えば、062488 は、1988 年 6 月 24 日を表す MDY (月日年) 形式です。この形式は、YMD (年月日) 形式、DMY (日月年) 形式、または JUL (年間通算日) 形式に変更することができます。年間通算日形式の場合には、QDAY の値が 001 から 366 までの範囲の 3 文字の値になります。この形式は、2 つの日付の間の日数を算出する場合に使用します。さらに日付区切り文字を取り除いたり、日付区切り文字として使用する文字を変更することも、**日付形式変換 (CVTDAT)** コマンドによって行うことができます。

日付形式変換 (CVTDAT) コマンドの形式は、以下のとおりです。

```
CVTDAT    DATE(変換したい日付) TOVAR(CL 変数) +
          FROMFMT(元の形式) TOFMT(新しい形式) +
          TOSEP(新しい区切り文字)
```

DATE パラメーターには、変換したい定数または変数を指定することができます。変換された日付は、TOVAR パラメーターに指定する変数に入れられます。次の例では、変数 &DATE の日付が、MDY の形式から DMY の形式に変更され、変数 &CVTDAT に入れられます。

```
CVTDAT    DATE(&DATE) TOVAR(&CVTDAT) FROMFMT(*MDY) TOFMT(*DMY)
          TOSEP(*SYSVAL)
```

日付区切り文字については、システム値 QDATSEP の指定がそのまま使用されます。

日付形式変換 (CVTDAT) コマンドは、その名前の一部として日付を使用するオブジェクトの作成やメンバーの追加を行う場合などに使用すると便利です。例えば、現在のシステム日付を使用してファイルにメンバーを追加しなければならないとします。また、現在の日付形式は MDY 形式で、これを年間通算日付形式に変換したいものとします。

```
PGM
DCL &DATE6 *CHAR  LEN(6)
DCL &DATE5  *CHAR  LEN(5)
RTVSYSVAL  QDATE  RTNVAR(&DATE6)
CVTDAT  DATE(&DATE6)  TOVAR(&DATE5)  TOFMT(*JUL)  TOSEP(*NONE)
ADDPFM  LIB1/FILEX  MBR('MBR'  *CAT  &DATE5)
.
.
.
ENDPGM
```

現在の日付が 1988 年 1 月 5 日であるとすれば、追加されるメンバーの名前は MBR88005 になります。

日付を変換する場合には、次の点に注意してください。

- DATE パラメーターの値の長さおよび TOVAR パラメーターの変数の長さは、日付の形式に対応するものでなければなりません。 TOVAR パラメーターに指定する変数の必要最小限の長さは次のとおりです。
-
  1. 年間通算日形式以外の形式で、年を 2 桁で表す場合
     a. 区切り文字を使用しない場合は 6 文字を使用する。

        1978 年 7 月 28 日は 072878 と表記される。

     b. 区切り文字を使用する場合は 8 文字を使用する。

        1978 年 7 月 28 日は 07-28-78 と表記される。
  2. 年間通算日形式以外の形式で、年を 4 桁で表す場合
     a. 区切り文字を使用しない場合は 8 文字を使用する。

        1978 年 7 月 28 日は 07281978 と表記される。

     b. 区切り文字を使用する場合は 10 文字を使用する。

        1978 年 7 月 28 日は 07-28-1978 と表記される。
  3. 年間通算日形式で、年を 2 桁で表す場合
     a. 区切り文字を使用しない場合は 5 文字を使用する。

        1996 年 12 月 31 日は 96365 と表記される。

     b. 区切り文字を使用する場合は 6 文字を使用する。

        1996 年 12 月 31 日は 96-365 と表記される。
  4. 年間通算日形式で、年を 4 桁で表す場合
     a. 区切り文字を使用しない場合は 7 文字が必要。

        1997 年 2 月 4 日は 1997035 と表記される。

b. 区切り文字を使用する場合は 8 文字が必要。

1997 年 2 月 4 日は 1997-035 と表記される。

変換後の文字数が変数に収まらないとエラー・メッセージが出されます。変換後の日付が変数より短い場合には、右側にブランクが埋め込まれます。

- 年間通算日付形式以外の日付形式の場合、月および日は、実際にどのような値が入るかに関係なくどれも 2 バイトのフィールドです。年は、2 バイトまたは 4 バイトのフィールドになります。変換後の値はすべて右寄せされ、必要に応じて先行ゼロが付けられます。
- 年間通算日付形式の場合には、日は 3 バイトのフィールドで、年は 2 バイトまたは 4 バイトのフィールドです。変換後の値はすべて右寄せされ、必要に応じて先行ゼロが付けられます。

次の例は、ILE バインド可能 API、現行ローカル時間取得 (CEELOCT) を使用する代替プログラムであり、年間通算日付形式にデータを変換します。このプログラムを作成するには、**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドを単独で使用するか、 **制御言語モジュール作成 (CRTCLMOD)** コマンドと**プログラム作成 (CRTPGM)** コマンドを共に使用する必要があります。

```
PGM
DCL  &LILDATE  *INT   LEN(4)
DCL  &PICTSTR  *CHAR  LEN(5)  VALUE(YYDDD)
DCL  &JULDATE  *CHAR  LEN(5)
DCL  &SECONDS  *CHAR  8       /*  Seconds from CEELOCT            */
DCL  &GREG     *CHAR  23      /*  Gregorian date from CEELOCT     */
                              /*                                  */
CALLPRC  PRC(CEELOCT)         /* Get current date and time        */ +
         PARM(&LILDATE        /* Date in Lilian format            */ +
              &SECONDS        /* Seconds field will not be used   */ +
              &GREG           /* Gregorian field will not be used */ +
              *OMIT)          /* Omit feedback parameter          */ +
                              /*  so exceptions are signalled     */

CALLPRC  PRC(CEEDATE) +
         PARM(&LILDATE        /* Today's date     */ +
              &PICTSTR        /* How to format    */ +
              &JULDATE        /* Julian date      */ +
              *OMIT)

ADDPFM   LIB1/FILEX  MBR('MBR'  *CAT   &JULDATE)

ENDPGM
```

関連情報:

現行ローカル時間取得 (CEELOCT) API

アプリケーション・プログラミング・インターフェース (API)

構成ソースの検索

構成ソースの検索 (RTVCFGSRC) コマンドを使用することによって、既存の構成オブジェクトを作成するための CL コマンド・ソースを生成し、そのソースをソース・ファイル・メンバーに入れることができます。

生成された CL コマンド・ソースは次の目的に使用することができます。

- システム相互間での構成の移送
- システムでの構成の保守
- 構成の保管 (SAVSYS を使用しない)

関連情報:

構成ソース検索 (RTVCFGSRC) コマンド

構成状況の検索

構成状況検索 (RTVCFGSTS) コマンドを使用することによって、アプリケーションは 3 つの構成オブジェクト (回線、制御装置、入出力装置) から構成状況を検索することができるようになります。

CL プログラムまたは CL プロシージャーで **RTVCFGSTS** コマンドを使用して構成記述の状況を検査することができます。

関連情報:

構成状況検索 (RTVCFGSTS) コマンド

ネットワーク属性の検索

ネットワーク属性検索 (RTVNETA) コマンドを使用して、プロシージャーでシステムのネットワーク属性を検索することができます。

また、ネットワーク属性変更は、 **ネットワーク属性変更 (CHGNETA)** コマンドによって変更でき、また **ネットワーク属性表示 (DSPNETA)** コマンドによって表示することができます。

関連情報:

ネットワーク属性検索 (RTVNETA) コマンド

ネットワーク属性変更 (CHGNETA) コマンド

ネットワーク属性表示 (DSPNETA) コマンド

ネットワーク属性検索 (QWCRNETA) API

例**:** ネットワーク属性検索コマンドの使用**:**

この例では、**ネットワーク属性検索 (RTVNETA)** コマンドを使用したネットワーク属性の検索方法を示します。

次の例では、デフォルトのネットワーク出力待ち行列およびそれが入っているライブラリーが検索されて QGPL/QPRINT に変更され、そして後で再び前の値に戻されます。

```
PGM
DCL  VAR(&OUTQNAME) TYPE(*CHAR) LEN(10)
DCL  VAR(&OUTQLIB) TYPE(*CHAR) LEN(10)
RTVNETA  OUTQ(&OUTQNAME) OUTQLIB(&OUTQLIB)
CHGNETA  OUTQ(QGPL/QPRINT)
.
.
.
CHGNETA  OUTQ(&OUTQLIB/&OUTQNAME)
ENDPGM
```

ジョブ属性の検索

ジョブ属性を検索してその値を CL 変数に移し、それを使用してアプリケーションを制御するには、**ジョブ属性検索 (RTVJOBA)** コマンドを使用します。このコマンドを使用することにより、すべてのジョブ属性、またはジョブ属性の任意の組み合わせを検索することができます。

次の CL プロシージャーでは、そのプロシージャーを呼び出したユーザーの名前を **ジョブ属性検索 (RTVJOBA)** コマンドによって検索しています。

```
PGM
 /* ORD410C Order entry program  */
DCL &CLKNAM TYPE(*CHAR) LEN(10)
DCL &NXTPGM TYPE(*CHAR) LEN(3)
.
.
```

```
        .
        RTVJOBA USER(&CLKNAM)
BEGIN:  CALL ORD410S2 PARM(&NXTPGM &CLKNAM)
         /* Customer prompt */
        IF (&NXTPGM *EQ 'END') THEN(RETURN)
        .
        .
        .
```

ユーザー名が入れられる変数 &CLKNAM は、最初に DCL コマンドによって宣言されています。 **ジョブ属性検索 (RTVJOBA)** コマンドは、一連の宣言コマンドの後に続きます。プログラム ORD410S2 が呼び出されると、 &NXTPGM および &CLKNAM という 2 つの変数がそのプログラムに渡されます。 &NXTPGM はブランクとして渡されますが、これは ORD410S2 の中で変更することができます。

関連情報:

ジョブ属性検索 (RTVJOBA) コマンド

例: ジョブ属性検索コマンドの使用:

この例では、**ジョブ属性検索 (RTVJOBA)** コマンドを使用したジョブ属性の検索方法を示します。

次の CL プロシージャーでは、対話式ジョブで CL プロシージャーをバッチとして投入することを想定しています。ジョブの完了メッセージが送られるメッセージ待ち行列の名前が、 **ジョブ属性検索 (RTVJOBA)** コマンドによって検索され、ジョブを投入したユーザーとの通信にそのメッセージ待ち行列が使用されています。

```
PGM
DCL &MSGQ *CHAR 10
DCL &MSGQLIB *CHAR 10
DCL &MSGKEY *CHAR 4
DCL &REPLY *CHAR 1
DCL &ACCTNO *CHAR 6
.
.
.
RTVJOBA SBMMSGQ(&MSGQ) SBMMSGQLIB(&MSGQLIB)
IF (&MSGQ *EQ '*NONE') THEN(DO)
   CHGVAR &MSGQ 'QSYSOPR'
   CHGVAR &MSGQLIB 'QSYS'
ENDDO
.
.
.
IF (. . . ) THEN(DO)
 SNDMSG:SNDPGMMSG MSG('Account number ' *CAT &ACCTNO *CAT 'is +
                  not valid. Do you want to cancel the update +
                  (Y or N)?') TOMSGQ(&MSGQLIB/&MSGQ) MSGTYPE(*INQ) +
                  KEYVAR(&MSGKEY)
 RCVMSG MSGQ(*PGMQ) MSGTYPE(*RPY) MSGKEY(&MSGKEY) +
          MSG(&REPLY) WAIT(*MAX)
 IF (&REPLY *EQ 'Y') THEN(RETURN)
 ELSE IF (&REPLY *NE 'N') THEN(GOTO SNDMSG)
ENDDO
.
.
.
```

使用するメッセージ待ち行列の名前とライブラリー名を受け入れる変数として、 &MSGQ と &MSGQLIB という 2 つの変数が宣言されています。 **ジョブ属性検索 (RTVJOBA)** コマンドは、このメッセージ待ち行列名およびライブラリー名を検索するために使用されています。このジョブについてメッセージ待ち行列が指定されていない可能性もあるので、メッセージ待ち行列名 *NONE という値と比較されます。この比較

が一致すれば、メッセージ待ち行列の指定がないことを示しており、ライブラリー QSYS のメッセージ待ち行列 QSYSOPR を使用するように、変数が変更されます。これ以降にプロシージャー内でエラー状態が検出された場合には、指定されたメッセージ待ち行列に照会メッセージが送られ、それに対する応答に基づいて処理が行われます。 **ジョブ属性検索 (RTVJOBA)** コマンドには、次のような使用法もあります。

- 1 つまたは複数のジョブ属性 (出力待ち行列やライブラリー・リストなど) を検索して、それらを一時的に変更し、後でまた元の値に戻すことができます。
- 1 つまたは複数のジョブ属性を検索して **ジョブ投入 (SBMJOB)** コマンドの中で使用することにより、投入されるジョブの属性が、それを投入するジョブの属性と同じになるようにすることができます。

ユーザー・プロファイル属性検索

ユーザー・プロファイル検索 (RTVUSRPRF) コマンドを使用することによって、ユーザー・プロファイル (パスワードを除く) を検索し、その値を CL 変数に入れて、アプリケーションの制御に使用することができます。

このコマンドには、10 文字のユーザー・プロファイル名、または *CURRENT を指定することができます。 **ユーザー・プロファイル検索 (RTVUSRPRF)** コマンドの実行の後でエスケープ・メッセージのモニターを行うこともできます。

関連情報:

ユーザー・プロファイル検索 (RTVUSRPRF) コマンド

例: ユーザー・プロファイル検索コマンドの使用:

この例では、**ユーザー・プロファイル検索 (RTVUSRPRF)** コマンドを使用したユーザー・プロファイル情報の検索方法を示します。

次の CL プロシージャーでは、このプロシージャーを呼び出したユーザーの名前、およびそのユーザーに対するメッセージが送られるメッセージ待ち行列の名前が、**ユーザー・プロファイル検索 (RTVUSRPRF)** コマンドによって検索されます。

```
DCL  &USR  *CHAR  10
DCL  &USRMSGQ  *CHAR  10
DCL  &USRMSGQLIB  *CHAR  10
.
.
.
RTVUSRPRF  USRPRF(*CURRENT)  RTNUSRPRF(&USR)  +
           MGSQ(&USRMSGQ)  MSGQLIB(&USRMSGQLIB)
```

このプロシージャーでは、次の情報が検索されます。

- &USR には、このプログラムを呼び出したユーザーのユーザー・プロファイル名が入る。
- &USRMSGQ には、そのユーザー・プロファイルで指定されているメッセージ待ち行列の名前が入る。
- &USRMSGQLIB には、ユーザー・プロファイルに指定されているメッセージ待ち行列の入っているライブラリーの名前が入る。

メンバー記述情報の検索

メンバー記述の検索 (RTVMBRD) コマンドを使用すれば、データベース・ファイルのメンバーに関する情報を検索して、アプリケーション・プログラムで使用することができます。

関連情報:

メンバー記述検索 (RTVMBRD) コマンド

例**:** メンバー記述の検索コマンドの使用**:**

この例では、**メンバー記述の検索 (RTVMBRD)** コマンドを使用したメンバー記述情報の検索方法を示します。

次の CL プロシージャーでは、**ユーザー・プロファイル属性検索 (RTVUSRPRF)** コマンドによって特定のメンバーの記述が検索されます。 MFILE というデータベース・ファイルが現行ライブラリー (MYLIB) の中にあり、そのファイルに 3 つのメンバー (AMEMBER、BMEMBER、および CMEMBER) があるものと想定しています。

```
DCL  &LIB      TYPE(*CHAR) LEN(10)
DCL  &MBR      TYPE(*CHAR) LEN(10)
DCL  &SYS      TYPE(*CHAR) LEN(4)
DCL  &MTYPE    TYPE(*CHAR) LEN(5)
DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
DCL  &CHGDATE  TYPE(*CHAR) LEN(13)
DCL  &TEXT     TYPE(*CHAR) LEN(50)
DCL  &NBRRCD   TYPE(*DEC)  LEN(10 0)
DCL  &SIZE     TYPE(*DEC)  LEN(10 0)
DCL  &USEDATE  TYPE(*CHAR) LEN(13)
DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
DCL  &RESET    TYPE(*CHAR) LEN(13)
.
.
.
RTVMBRD    FILE(*CWeb siteIB/MYFILE) MBR(AMEMBER *NEXT) +
           RTNLIB(&LIB) RTNSYSTEM(&SYS) RTNMBR(&MBR) +
           FILEATR(&MTYPE) CRTDATE(&CRTDATE) TEXT(&TEXT) +
           NBRCURRCD(&NBRRCD) DTASPCSIZ(&SIZE) USEDATE(&USEDATE) +
           USECOUNT(&USECNT) RESETDATE(&RESET)
```

このプロシージャーでは、次の情報が検索されます。

- 現行ライブラリーの名前 (MYLIB) が &LIB という名前の CL 変数に入れられます。
- MYFILE が見つかったシステムが、CL 変数 &SYS に入れられる。 (値 *LCL は該当ファイルがローカル・システムで見つかったこと、そして *RMT は該当ファイルがリモート・システムで見つかったことを意味する。)
- メンバー名 (BMEMBER) が CL 変数 &MBR に入れられる。これは、名前順のメンバー・リスト (*NEXT) で AMEMBER のすぐ後のメンバーが BMEMBER であるためです。
- MYFILE のファイル属性が CL 変数 &MTYPE に入れられる。 (値 *DATA は該当メンバーがデータ・メンバーであること、そして *SRC は該当メンバーがソース・メンバーであることを意味する。)
- BMEMBER の作成日が CL 変数 &CRTDATE に入れられる。
- BMEMBER のテキスト記述が CL 変数 &TEXT に入れられる。
- BMEMBER の現在のレコード数が CL 変数 &NBRRCD に入れられる。
- BMEMBER のデータ・スペースのサイズ (バイト数) が CL 変数 &SIZE に入れられる。
- BMEMBER の最後の使用日が CL 変数 &USEDATE に入れられる。
- BMEMBER の使用日数が CL 変数 &USECNT に入れられる。この日数カウントの開始日は CL 変数 &RESET に入れられる値です。

CL ソース・プログラムのコンパイル

CL ソース・プログラムを実行するには、まずそれをコンパイルする必要があります。

CL プログラムを 1 つのステップで作成するには、**制御言語プログラム作成 (CRTCLPGM)** コマンドまたは**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドを使用します。これにより、1 つのモジュールによってバインドされたプログラムを作成することができます。

また、**制御言語モジュール作成 (CRTCLMOD)** コマンドによって CL モジュールを作成することもできます。その後モジュールは、 **プログラム作成 (CRTPGM)** または**サービス・プログラム作成 (CRTSRVPGM)** コマンドを使用して、プログラムやサービス・プログラムにバインドする必要があります。

CL ソースの組み込み (INCLUDE) コマンドを使用して、コンパイル中に他のソース・ファイルを組み込みます。

次の例は、モジュール ORD040C を作成して、それをライブラリー DSTPRODLB に入れます。

```
CRTCLMOD     MODULE(DSTPRODLB/ORD040C) SRCFILE(QCLSRC)
             TEXT('Order dept general menu program')
```

ORD040C のソース・コマンドはソース・ファイル QCLSRC に入っており、そのソース・メンバー名は ORD040C です。デフォルト値により、コンパイラー・リストが作成されます。

もう 1 つの例では、プログラム IFSTEST を作成し、IFS ストリーム・ソース・ファイルからライブラリー QGPL にそれを配置します。

```
CRTBNDCL PGM(QGPL/IFSTEST) SRCSTMF('/home/ifstest/ifstest.clp')
```

IFSTEST のソース・コマンドは、IFS ストリーム・ソース・ファイル /home/ifstest/ifstest.clp にあります。デフォルト値により、コンパイラー・リストが作成されます。

制御言語プログラム作成 (CRTCLPGM) コマンドまたは**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドでは、リスト・オプション、およびプログラムをその所有者のユーザー・プロファイルのもとで実行することが必要かどうかを指定できます。

プログラムは所有者のユーザー・プロファイルまたは使用者のユーザー・プロファイルのどちらを用いても実行することができます。

関連情報:

CL コマンド検索プログラム

プログラム作成 (CRTPGM) コマンド

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

サービス・プログラム作成 (CRTSRVPGM) コマンド

CL ソース組み込み (INCLUDE) コマンド

CL ソース検索 (RTVCLSRC) コマンド

CL ソース・プログラムでの作成オプションの設定

処理オプション宣言 (DCLPRCOPT) コマンドを使用して、 CL コンパイラー・コマンド上の多数の同じ作成オプション・パラメーターを設定できます。CL コンパイラー・コマンドには、**制御言語プログラム作成 (CRTCLPGM)**、**制御言語モジュール作成 (CRTCLMOD)**、および**バインド制御言語プログラム作成 (CRTBNDCL)** が含まれます。

DCLPRCOPT コマンドに指定される値は、CL コンパイラー・コマンドに指定されるどの値よりも優先されます。さらに、**DCLPRCOPT** コマンドは、 **CRTBNDCL** コマンドの実行中に使用できる、バインド・ディレクトリ

ー (BNDDIR) およびバインド・サービス・プログラム (BNDSRVPGM) パラメーターに対応します。 以下の例は、 **DCLPRCOPT** コマンドの使用法を示します。

例: CRTCLPGM を一時変更するコンパイラー・オプションの宣言

```
DCLPRCOPT    ALWRTVSRC(*NO)  USRPRF(*OWNER)
```

このコマンドは、**CRTCLPGM** コマンドに指定される、RTVCLSRC 許可 (ALWRTVSRC) およびユーザー・プロファイル (USRPRF) パラメーター値を一時変更します。結果の CL プログラムでは、*PGM オブジェクトから CL ソース・コードを取り出すことはできません。プログラム・オブジェクトは、呼び出されると、*PGM オブジェクトを所有するユーザー・プロファイルの権限を適用します。

例: CRTCLMOD を一時変更するコンパイラー・オプションの宣言

```
DCLPRCOPT    LOG(*NO)  AUT(*USE)
```

このコマンドは、**CRTCLMOD** コマンドに指定される、ログ・コマンド (LOG) および権限 (AUT) パラメーター値を一時変更します。結果の ILE CL モジュールが ILE プログラムまたはサービス・プログラムにバインドされ、ILE CL プロシージャーが呼び出されると、このプロシージャーから実行する CL コマンドがジョブ・ログに記録されません。**CRTCLMOD** コマンドによって作成される *MODULE オブジェクトの共通権限は *USE です。

例: CRTBNDCL を一時変更するコンパイラー・オプションの宣言

```
DCLPRCOPT    ALWRTVSRC(*NO)  DFTACTGRP(*NO)  ACTGRP(MYAPP)  +
             BNDDIR(MYAPPLIB/MYBNDDIR)
```

このコマンドは、**CRTBNDCL** コマンドに指定される、 RTVCLSRC 許可 (ALWRTVSRC)、デフォルト活動化グループ (DFTACTGRP)、活動化グループ (ACTGRP)、およびバインド・ディレクトリー (BNDDIR) のパラメーター値を一時変更します。結果の ILE CL プログラムに、 **RTVCLSRC** コマンドを使用して取得可能な CL ソースは含まれません。プログラムは、活動化グループ MYAPP で実行されます。**CRTPGM** コマンドを使用してバインド CL プログラムを作成するとき、**CRTBNDCL** コマンドは、ライブラリー MYAPPLIB 内のバインド・ディレクトリー MYBNDDIR を、バインド・ディレクトリー (BNDDIR) パラメーターに追加します。このように、バインド・ディレクトリーによって参照されるサービス・プログラムおよび ILE モジュールを使用して、ILE CL プログラムで使用される ILE プロシージャーを解決できます。

関連情報:

制御言語プログラム作成 (CRTCLPGM) コマンド

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

処理オプション宣言 (DCLPRCOPT) コマンド

別のソース・メンバーからの CL コマンドの組み込み

CL ソース組み込み (INCLUDE) コマンドを使用すると、CL ソース・コードを複数のソース・ファイル・メンバーにわたってコンパイルできるように、CL ソース・コードを分割することができます。

組み込み対象の CL ソースは、CL コンパイラー・コマンドのソース・ファイル (SRCFILE) パラメーターで識別された同じソース・ファイルの別のメンバーに配置することも、別のソース・ファイルに配置することもできます。 CL コンパイラー・コマンドには、**制御言語プログラム作成 (CRTCLPGM)**、**制御言語モジュール作成 (CRTCLMOD)**、および**バインド制御言語プログラム作成 (CRTBNDCL)** が含まれます。

| 組み込み対象の CL ソースは、IFS ストリーム・ファイルに配置することも可能です。メイン・ソース・
| ファイルも IFS ストリーム・ファイルである場合は、コンパイラーが使用する検索パスで IFS ストリーム
| の組み込みファイルを検索するように、 **制御言語モジュール作成 (CRTCLMOD)** コマンドまたは **バインド制
| 御言語プログラム作成 (CRTBNDCL)** コマンドの組み込みディレクトリー (INCDIR) パラメーターに 1 つ以
| 上のディレクトリーを指定することができます。

| メイン・ソース・ファイルがデータベース・ファイルであっても、IFS ソース・ファイルであっても、ソー
| ス・ステートメントに **CL ソース組み込み (INCLUDE)** コマンドで組み込むストリーム・ファイルのパス名
| を指定できます。

```
| INCLUDE   SRCSTMF('/dir/pay_data.cl')
```

| 後で **CL ソース検索 (RTVCLSRC)** コマンドを実行して、元の CL ソース (INCLUDE コマンドのみが含まれる) または拡張された CL ソース (組み込み CL ソース・コマンドが含まれる) のいずれかを検索することができます。

関連情報:

制御言語プログラム作成 (CRTCLPGM) コマンド

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

CL ソース組み込み (INCLUDE) コマンド

CL ソース検索 (RTVCLSRC) コマンド

CL プログラムまたはプロシージャーのコマンドのロギング

CL プログラムまたはプロシージャー内で実行されているほとんどの CL コマンドをジョブ・ログに書き込む (記録する) よう指定できます。

CL コマンドをログに記録するには、CL ソース・プログラムのコンパイル時に、 **制御言語モジュール作成 (CRTCLMOD)** コマンドまたは **バインド制御言語プログラム作成 (CRTBNDCL)** コマンドの LOG パラメーターに、以下の値の 1 つを指定します。

***JOB** これはデフォルト値であり、ジョブのロギング・オプションがオンである場合にロギングを行うことを示します。このオプションはロギングを行わない値に初期設定されていますが、これは**ジョブ変更 (CHGJOB)** コマンドの LOGCLPGM パラメーターによって変更することができます。したがって、この値でモジュールやプログラムを作成した場合には、各ジョブごとに、あるいは 1 つのジョブの中で複数回に渡って、ロギング・オプションを変更することができます。

***YES** この値は、CL プログラムまたはプロシージャーを実行するごとにロギングを行うことを示します。この値は CHGJOB コマンドによって変更することはできません。

***NO** この値はロギングを行わないことを示します。この値は CHGJOB コマンドによって変更することはできません。

これらの値は**制御言語モジュール作成 (CRTCLMOD)** または**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドの一部なので、値の変更が必要な場合にはモジュールまたはプログラムを再コンパイルしなければなりません。

ロギングを指定した場合には、記録されたコマンドがジョブ・ログから除去されてしまうことのないように、 **メッセージ除去 (RMVMSG)** コマンドの扱いには十分な注意を払う必要があります。 RMVMSG コマンドで CLEAR (*ALL) を指定すると、その RMVMSG コマンドの実行前に記録されたコマンドはすべて

ジョブ・ログから消去されてしまいます。この影響を受けるのは、その RMVMSG コマンドを含む CL プログラムまたはプロシージャーのみであり、それ以前または以後の反復レベルで記録されるコマンドには影響ありません。

すべてのコマンドがジョブ・ログに記録されるわけではありません。以下に、ログに記録されないコマンドをリストします。

CALLPRC	**CALLSUBR**	**CHGVAR**
DCL	**DCLF**	**DCLPRCOPT**
DO	**DOFOR**	**DOUNTIL**
DOWHILE	**ELSE**	**ENDDO**
ENDPGM	**ENDSELECT**	**ENDSUBR**
GOTO	**IF**	**INCLUDE**
ITERATE	**LEAVE**	**MONMSG**
OTHERWISE	**PGM**	**RTNSUBR**
SELECT	**SUBR**	**WHEN**

ロギング・オプションがオンである場合、ロギング・メッセージは CL プログラムまたはプロシージャーのメッセージ待ち行列に送られます。CL プログラムまたはプロシージャーが対話式に実行され、ジョブの LOG パラメーターに指定されたメッセージ・レベルが 4 に設定されている場合は、F10 (詳細メッセージの表示) キーを押すことにより、記録されているすべてのコマンドを表示できます。また、メッセージ・レベルが 4 の場合に、サインオフの時点で *PRINT を指定すれば、そのログを印刷することができます。

ログには、時刻、プログラム名とプロシージャー名、メッセージ・テキスト、およびコマンド名が記録されます。コマンド名は、元のソース・ステートメントと同じように修飾されます。コマンドのパラメーターも記録されます。パラメーター情報が CL 変数である場合には、その変数の内容が印刷されます (RTNVAL パラメーターは除く)。

コマンドのロギングはパフォーマンスに影響を与えます。

関連概念:

620 ページの『ジョブ・ログ』
各ジョブには、ジョブ・ログが関連付けられています。

関連情報:

CL コマンド検索プログラム

制御言語モジュール作成 (CRTCLMOD) コマンド

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

ジョブ変更 (CHGJOB) コマンド

メッセージ除去 (RMVMSG) コマンド

CL ソース・コードの検索

使用された CL コンパイラー・オプションによっては、 CL プログラムまたは CL モジュール・オブジェクトから CL ソース・コードを検索することができます。

CL プログラムまたは CL モジュールは、RTVCLSRC 許可 (ALWRTVSRC) パラメーターに *YES を指定して作成されていなければなりません。

CL プログラムまたは CL モジュールから CL ソースを検索できると、オリジナルの CL ソース・コードを入手できないシステムで、CL コードの問題の診断と修正を簡単にできるようになります。

ALWRTVSRC(*YES) を指定して CL プログラムまたは CL モジュールが作成されたかどうかを判別するための CL コマンドおよび呼び出し可能 API プログラムがあります。

- **制御言語プログラム作成 (CRTCLPGM)** コマンドを使用してプログラム (*PGM) オブジェクトが作成された場合には、 **プログラム表示 (DSPPGM)** コマンドまたは**プログラム情報検索 (QCLRPGMI)** API を使用して、プログラム作成時に ALWRTVSRC パラメーターに指定された値を確認できます。
- **制御言語モジュール作成 (CRTCLMOD)** コマンドを使用してモジュール (*MODULE) オブジェクトが作成された場合には、 **モジュール表示 (DSPMOD)** コマンドまたは**モジュール情報検索 (QBNRMODI)** API を使用して、モジュール作成時に ALWRTVSRC パラメーターに指定された値を確認できます。
- **バインド制御言語プログラムの作成 (CRTBNDCL)** コマンドまたは**プログラム作成 (CRTPGM)** コマンドを使用して CL モジュールが統合言語環境 (ILE) プログラムにバインドされた場合には、DETAIL(*MODULE) を指定した**プログラム表示 (DSPPGM)** コマンド、または**プログラム情報リスト (QBNLPGMI)** API を使用して、モジュール作成時に ALWRTVSRC パラメーターに指定された値を確認できます。
- **サービス・プログラム作成 (CRTSRVPGM)** コマンドを使用して CL モジュールが ILE サービス・プログラム (*SRVPGM) オブジェクトにバインドされた場合には、 DETAIL(*MODULE) を指定した**サービス・プログラム表示 (DSPSRVPGM)** コマンド、または**サービス・プログラム情報リスト (QBNLSPGM)** API を使用して、モジュール作成時に ALWRTVSRC パラメーターに指定された値を確認できます。

ALWRTVSRC パラメーターの出荷時デフォルト値は *YES です。 CL ソースを ILE CL モジュールと共に保存する機能は、オペレーティング・システムの IBM i 7.1 のリリースで追加されました。

ILE CL プログラム内のモジュールからの CL ソースの検索

```
RTVCLSRC   PGM(MYCLPGM)  MODULE(MOD1)  SRCFILE(MYLIB/QCLSRC)
```

このコマンドは、ILE プログラム MYCLPGM のモジュール MOD1 から CL ソースを検索します。検索された CL ソースは、ライブラリー MYLIB にあるソース物理ファイル QCLSRC のメンバー MOD1 に保管されます。

関連情報:

制御言語プログラム作成 (CRTCLPGM) コマンド

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

プログラム作成 (CRTPGM) コマンド

サービス・プログラム作成 (CRTSRVPGM) コマンド

プログラム表示 (DSPPGM) コマンド

サービス・プログラム表示 (DSPSRVPGM) コマンド

モジュール表示 (DSPMOD) コマンド

CL ソース検索 (RTVCLSRC) コマンド

プログラム情報検索 (QCLRPGMI) API

モジュール情報検索 (QBNRMODI) API

プログラム情報リスト (QBNLPGMI) API

サービス・プログラム情報リスト (QBNLSPGM) API

CL モジュール・コンパイラー・リスト

CL モジュールを作成する時点で、**制御言語モジュール作成 (CRTCLMOD)** コマンドの OPTION および OUTPUT パラメーターを使用して、種々のタイプのリストを作成することができます。

OPTION パラメーターの値とそれぞれの意味は次のとおりです。

- *GEN または *NOGEN

  モジュールを作成するかどうか (デフォルト値は *GEN)。
- *XREF または *NOXREF

  ソースの変数およびデータの参照に関する相互参照リストを作成するかどうか (デフォルト値は *XREF)。

OUTPUT パラメーターの値とそれぞれの意味は次のとおりです。

- *PRINT - 印刷リスト
- *NONE - コンパイラー・リストなし

OUTPUT パラメーターの指定に基づいて作成されるリストを、コンパイラー・リスト と呼びます。 次に示すのはコンパイラー・リストの例です。図中の番号は、リストの後の説明を示しています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
1                                                                                                  2                    3
5722SS1 V5R3M0  041231                制御言語                MYLIB/DUMPER         SYSNAME  05/06/00 11:12:55       ページ 1
モジュール  . . . . . . . . . . . . . . . . . . . . :   DUMPERR
  ライブラリー . . . . . . . . . . . . . . . . . . . :     MYLIB
ソース・ファイル . . . . . . . . . . . . . . . . . . :   QCLSRC
  ライブラリー . . . . . . . . . . . . . . . . . . . :     MYLIB
ソース・メンバー名 . . . . . . . . . . . . . . . . . :   DUMPERR   05/06/94 10:42:26 4
ソースの印刷オプション . . . . . . . . . . . . . . . :   *XREF  *NOSECLVL  *NOEVENTF
モジュールのロギング . . . . . . . . . . . . . . . . :   *JOB
モジュール・オブジェクトの置き換え . . . . . . . . . :   *YES
ターゲット・リリース . . . . . . . . . . . . . . . . :   V5R3M0
権限 . . . . . . . . . . . . . . . . . . . . . . . . :   *LIBCRTAUT
ソート順序 . . . . . . . . . . . . . . . . . . . . . :   *HEX
言語識別コード . . . . . . . . . . . . . . . . . . . :   *JOBRUN
テキスト  . . . . . . . . . . . . . . . . . . . . . :   テスト・プログラム
最適化 . . . . . . . . . . . . . . . . . . . . . . . :   *NONE
デバッグ・ビュー . . . . . . . . . . . . . . . . . . :   *STMT
パフォーマンス収集を使用可能にする . . . . . . . . . :   *PEP
コンパイラー . . . . . . . . . . . . . . . . . . . . :   IBM i5/OS 制御言語コンパイラー 5
6                          ソース制御言語
SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.  日付  8
   100- PGM                                                                                                  05/06/94
   200- DCL &ABC *CHAR 10 VALUE('THIS')                                                                      05/06/94
   300- DCL &XYZ *CHAR 10 VALUE('THAT') 7                                                                    05/06/94
   400- DCL &MNO *CHAR 10 VALUE('OTHER')                                                                     05/06/94
   500- CRTLIB LB(LARRY)                                                                                     05/06/94
* CPD0043 30  このコマンドにはキーワード LB は正しくない。 9
   600- DLTLIB LIB(MOE                                                                                       05/06/94
* CPD0013 30  対応する括弧が見つからない。
   700- MONMSG CPF0000 EXEC(GOTO ERR)                                                                        05/06/94
   800- ERROR:                                                                                               05/06/94
   900- CHGVAR &ABC 'ONE'                                                                                    05/06/94
  1000- CHGVAR &XYZ 'TWO'                                                                                    05/06/94
  1100- CHGVAR &MNO 'THREE'                                                                                  05/06/94
  1200- DMPCLPGM                                                                                             05/06/94
  1300- ENDPGM                                                                                               05/06/94
                          * * * * *  ソ ー ス の 終 わ り  * * * * *
5722SS1 V5R3M0  040201                制御言語                MYLIB/DUMPER         SYSNAME  05/06/00 11:12:55       ページ 2
                                       相互参照
宣言済み変数
名前                定義     タイプ          長さ          参照
&ABC                200      *CHAR            10          900
&MNO                400      *CHAR            10         1100
10
&XYZ                300      *CHAR            10         1000
定義済ラベル
ラベル           定義済み    参照 11
ERR              ******     700
* CPD0715 30  ラベル 'ERR    ' が存在していません。
ERROR               800
                    * * * * *  相 互 参 照 表 の 終 わ り  * * * * *
5722SS1 V5R3M0  040201                制御言語                MYLIB/DUMPER         SYSNAME  05/06/04 11:12:55       ページ 3
                                       メッセージの要約
              重大度
合計        0-9  10-19  20-29  30-39  40-49  50-59  60-69  70-79  80-89  90-99  12
   3          0      0      0      3      0      0      0      0      0      0
モジュール DUMPERR はライブラリー MYLIB に作成されなかった。最大エラー重大度 30。 13
                          * * * * *  メ ッ セ ー ジ の 要 約 の 終 わ り  * * * * *
                            * * * * *  コ ン パ イ ル の 終 わ り  * * * * *
```

タイトル:

1 IBM i オペレーティング・システムのプログラム番号、リリース、モディフィケーション・レベル、および日付。

2 コンパイルを実行した日付と時刻。

3 リストのページ番号。

プロローグ:

4 制御言語モジュール作成 (CRTCLMOD) コマンドに指定したパラメーター値 (指定のない場合はデフォルト値)。ソース・ファイルがデータベース・ファイルでない場合には、メンバー名、日付、および時刻は省略されます。

5 コンパイラーの名前。

ソース:

6 ソースの行 (レコード) の順序番号。順序番号の後のダッシュ (-) は、ソース・ステートメントがその順序番号から始まっていることを示します。ダッシュがない場合は、そのステートメントは前のステートメントの続きを示します。

ソース・ステートメント間の注記は他のソース・ステートメントと同様に扱われ、順序番号も付けられます。

7 ソース・ステートメント。

8 ソース・ステートメントが最後に変更または追加された日付。ソース・ステートメントがデータベース・ファイルの中にない場合、または RGZPFM を用いて日付がリセットされている場合には、日付は省略されます。

9 コンパイルの過程でエラーが検出され、エラーのソース・ステートメントを特定できる場合には、そのソース・ステートメントの直後にエラー・メッセージが印刷されます。アスタリスク (*) は、その行にエラー・メッセージが含まれていることを示します。その行には、メッセージ識別コード、重大度、およびメッセージ・テキストが印刷されます。

相互参照:

10 記号変数テーブルは、プログラムで宣言されている有効な変数の相互参照リストです。このテーブルには、変数、その変数を宣言しているステートメントの順序番号、その変数の属性、およびその変数を参照しているステートメントの順序番号がリストされます。

11 ラベル・テーブルは、プログラムで宣言されている有効なラベルの相互参照リストです。このテーブルには、ラベル、そのラベルが定義されているステートメントの順序番号、およびそのラベルを参照しているステートメントの順序番号がリストされます。

メッセージ:

このサンプル・モジュールの場合には、総括的なエラー・メッセージは出されないので、サンプル・リストにはこのセクションはありません。総括的なエラー・メッセージがあった場合には、その各メッセージについてメッセージ識別コード、重大度、およびメッセージ・テキストがこの部分に示されます。

メッセージの要約

12 コンパイルの過程で出されたメッセージ数の要約。総数とともに、重大度別の内訳も示されます。

13 メッセージの要約の後には完了メッセージが印刷されます。

*SOURCE オプションを選択した場合には、タイトル、前書き、およびメッセージの要約の各セクションは必ず印刷されます。相互参照セクションは、*XREF オプションを指定した場合に印刷されます。メッセージ・セクションが印刷されるのは、総括的なエラーが見つかった場合だけです。

関連概念:

『一般的なコンパイル・エラー』

コンパイルの過程で検出されるエラーのタイプとしては、構文エラー、未定義の変数やラベルに対する参照、およびステートメントの欠落などがあります。

関連情報:

制御言語モジュール作成 (CRTCLMOD) コマンド

一般的なコンパイル・エラー

コンパイルの過程で検出されるエラーのタイプとしては、構文エラー、未定義の変数やラベルに対する参照、およびステートメントの欠落などがあります。

CL コンパイラー・リストでは、特定のコマンドに直接関係のあるエラー条件は、そのコマンドの直後にリストされます。特定のコマンドだけに関連するのではなく、より一般的なエラー・メッセージは、個々のソース・ステートメントの直後ではなく、リストのメッセージ・セクションにリストされます。

次のタイプのエラーがある場合、プログラムまたはモジュールは作成されません (重大度コードは無視されます)。

- 値のエラー
- 構文エラー
- 1 つのコマンドの中でのパラメーター相互間の依存関係に関するエラー
- 妥当性検査で検出されたエラー

プログラムまたはプロシージャーの作成中止の原因となるエラーが検出された場合、コンパイラーはソースのエラーの検査を続行します。これによって、モジュールまたはプログラムを再び作成し直す前に、できるだけ多くのエラーを訂正することができます。

関連資料:

268 ページの『CL モジュール・コンパイラー・リスト』
CL モジュールを作成する時点で、**制御言語モジュール作成 (CRTCLMOD)** コマンドの OPTION および OUTPUT パラメーターを使用して、種々のタイプのリストを作成することができます。

CL ダンプの取得

CL プログラムまたは CL プロシージャーの実行中に、その CL プログラムまたは CL プロシージャーのダンプを取得することができます。

CL ダンプには、プログラムまたはプロシージャーのメッセージ待ち行列上にあるすべてのメッセージ、およびそのプログラムまたはプロシージャーで使用されているすべての変数の値がリストされます。この情報は、プログラムまたはプロシージャーの処理に影響を与えるような問題の原因を判別するのに役立ちます。

CL ダンプを取得するには、次のいずれかのタスクを実行してください。

- **CL プログラム・ダンプ (DMPCLPGM)** コマンドを実行する。このコマンドは CL プログラムまたは CL プロシージャーの中でのみ使用できるコマンドであり、またこのコマンドを使用してもその CL プログラムまたは CL プロシージャーは終了しません。
- 照会メッセージ CPA0701 または CPA0702 に対する応答として D を入力する。システムがこのメッセージを送るのは、監視対象外のエスケープ・メッセージを CL プログラムまたは CL プロシージャーから受け取った場合です。対話式ジョブでプログラムが実行されている場合、システムはこのメッセージをジョブの外部メッセージ待ち行列に送ります。プログラムがバッチ・ジョブとして実行されている場合は、システムはこのメッセージをシステム・オペレーター・メッセージ待ち行列 (QSYSOPR) に送ります。
- 該当のジョブについて INQMSGRPY (*SYSRPYL) を指定する。 IBM 提供のシステム応答リストでは、メッセージ CPA0702 または CPA0701 に対する応答として D が指定されています。これらの照会メッセージのどちらかを受け取ると、システムはダンプを印刷します。
- メッセージ CPA0701 または CPA0702 に対するデフォルトの応答を、C (プログラム取り消し) から D (プログラム・ダンプ) に変更する。このようにすれば、CL プログラムまたは CL プロシージャー内で機能チェックが発生するたびにプロシージャー・ダンプが印刷されます。デフォルト値を変更するには、次のコマンドを入力してください。

  ```
  CHGMSGD MSGID(CPA0702) MSGF(QCPFMSG) DFT(D)
  ```

  注**:** 機密保護担当者か、またはメッセージ・ファイル QCPFMSG に対する更新権限をもつユーザーが、**CHGMSGD** コマンドを入力しなければなりません。

メッセージに対するデフォルトの応答を変更した場合、次のいずれかの条件のもとにダンプが印刷されます。

- システム・オペレーター・メッセージ待ち行列がデフォルトのモードになっている場合に、バッチ・ジョブからこのメッセージが送られた場合。
- ディスプレイ装置のユーザーが、メッセージに対するデフォルトの応答を返すために、応答を入力せずに実行キーを押した場合。
- 該当のジョブに対して INQMSGRPY (*DFT) が指定されている場合。

```
        1
5722SS1 V5R3M0  040201               制御言語         MYLIB/DUMPER         SYSNAME  05/06/00  11:05:032 ページ 1

ジョブ名 . . . . . :  DSP043  ユーザー名 . . . . . :   SMITH 3  ジョブ番号 . . . . . : 01329 3
プログラム名 . . . . . :  DUMP4  ライブラリー . . . . . :   MYLIB 4  ステートメント . . . . . : 1200 5
モジュール名 . . . . . :  DUMP   プロシージャー名 . . . . :   DUMP

                                                メッセージ

          メッセージ 6                          メッセージ              送信元                          送信先
時刻      ID              重大度     タイプ     テキスト                プログラム      命令      プログラム      命令
110503    CPC2102           00         COMP     Library LARRY created.  QLICRLIB        *N        DUMP            *N
110503    CPF2110           40         ESC      Library MOE not found.  QLICLLIB        *N        DUMP            *N

                                                変数 7

変数            タイプ     長さ              値                          値 (16 進数)
                                             *...+....1....+....2....+   * . . . + . . . . 1 . . . . + . . . . 2 . . . . +
&ABC            *CHAR     10                'ONE       '                D6D5C5404040404040
&XYZ            *CHAR     10                'TWO       '                E3E6D640404040404040

                                    * * * * * ダ ン プ の 終 わ り * * * * *
```

1 IBM i オペレーティング・システムのプログラム番号、リリース、モディフィケーション・レベル、および日付。

2 ダンプが印刷された日付と時刻。

3 プロシージャーが実行されていたジョブの修飾名。

4 プログラムの名前とライブラリー。

5 ダンプがとられたときに実行されていたステートメントの番号。そのコマンドがネストされたコマンドである場合、この番号はその外側のステートメントの番号を示しています。

6 プロシージャーのメッセージ待ち行列にある各メッセージ。これには、メッセージが送られた時刻、メッセージ ID、重大度、タイプ、テキスト、メッセージを送ったプログラムとその命令番号、およびメッセージを受け取ったプログラムとその命令番号が含まれます。

7 プロシージャーで宣言されているすべての変数。これには、変数名、タイプ、長さ、値、および 16 進数による値が含まれます。

10 進変数に有効でない 10 進データが含まれていた場合には、文字値および 16 進数による値が *CHAR 変数として印刷されます。

変数の値が見つからなかった場合には、*NOT ADDRESSABLE (アドレス不能) が印刷されます。これが生じるのは、該当の CL プロシージャーが、パラメーターとして TYPE(*NULL) と PASSVAL(*NULL) のどちらかが指定されているコマンドのコマンド処理プログラムとして使用されている場合、あるいはパラメーターに RTNVAL(*YES) の指定があり、戻り変数がコマンドで指定されていない場合です。

変数が TYPE(*LGL) として宣言されている場合、その変数は長さ 1 のタイプ *CHAR としてダンプ上に示されます。

関連情報:

照会管理プログラムの照会メッセージに対するデフォルト応答の制御

CL プログラム・ダンプ (DMPCLPGM) コマンド

モジュール属性の表示

CL モジュールなどのモジュール・オブジェクトの属性を表示するには、**モジュールの表示 (DSPMOD)** コマンドを使用します。

これによって表示または印刷された情報によって、モジュールを作成するために使用されたコマンドで、どのようなオプションが指定されているかを調べることができます。

関連情報:

モジュール表示 (DSPMON) コマンド

プログラム属性の表示

CL プログラムなどのプログラム・オブジェクトの属性を表示するには、**プログラム表示 (DSPPGM)** コマンドを使用します。

これによって表示または印刷された情報によって、プログラムを作成するために使用されたコマンドでどのようなオプションが指定されているかを調べることができます。

プログラム表示 (DSPPGM) コマンドを使用して、プログラムの属性を表示することができます。 いくつかの属性 (プログラム・タイプ、ソース・メンバー、テキスト、作成日付など) を CL 変数に取り出すには、**オブジェクト記述表示 (DSPOBJD)** コマンドを使用して、出力ファイルを作成することができます。 出力ファイルは、CL プロシージャーまたはプログラム内で**ファイル宣言 (DCLF)** および**ファイル受信 (RCVF)** コマンドを使用して読み取ることができます。 DSPPGM コマンドの他の属性 (USRPRF など) にアクセスするには、プログラム情報検索 (QCLRPGMI) API を使用できます。

関連情報:

プログラム表示 (DSPPGM) コマンド

戻りコードの要約

戻りコードは、**ジョブ属性検索 (RTVJOBA)** コマンドの戻りコード (RTNCDE) パラメーターを使用して戻すことができます。

戻りコードは、5 桁の 10 進数で、小数点以下の桁数はありません (12345. など)。この 10 進値は、呼び出されるプログラムの状況を示します。 CL プログラム自体は戻りコードを設定しませんが、 CL プログラム内の他のプログラムが設定する戻りコードの現行値を検索することができます。これを行うには、**ジョブ属性検索 (RTVJOBA)** コマンドの RTNCDE パラメーターを使用します。

以下のリストは、IBM i オペレーティング・システムでサポートされている言語で使用される戻りコードを要約しています。

- RPG IV プログラム

  RPG IV コンパイラーから送られる戻りコードには、次のものがあります。

  0 プログラムが作成された場合

  2 プログラムが作成されなかった場合

  RPG IV プログラムの実行によって送られる戻りコードには、次のものがあります。

  0 プログラムが開始された場合、またはプログラム呼び出し前の CALL 操作の時点

  1 LR がオンに設定されてプログラムが終了した場合

  2 エラーが生じてプログラムが終了した場合 (照会メッセージに対する応答が C、D、F、または S の場合)

3 停止標識 (H1 から H9) によってプログラムが終了した場合

RPG IV の戻りコードは CALL の後でだけテストされます。

– 0 または 1 はエラーがないことを示す。
– 3 の場合は RPG IV の状況コード 231 に対応する。
– その他の値の場合には、RPG IV の状況コード 202 (エラーにより呼び出し終了) に対応する。

RPG IV プログラムの中で、ユーザーは直接戻りコードをテストすることはできません。

- ILE COBOL プログラムと OPM COBOL プログラム

  COBOL プログラムの実行によって送られる戻りコードには、次のものがあります。

  0 プログラムが呼び出される前に各 CALL ステートメントによって送られるもの

  2 プログラムが機能チェック (CPF9999) を受け取った場合、または総称入出力例外処理プログラムが制御権を得たが、適用できる USE プロシージャーがなかった場合

  COBOL プログラムはこれらの戻りコードを検索することはできません。 OPM COBOL では、戻りコード値 2 はメッセージ LBE9001 を送信します。 ILE COBOL では、戻りコード値 2 はメッセージ CEE9001 を送信します。

- ILE C プログラム

  整数戻りコードの現行値は、ILE C プログラムの最後の ILE C 戻りステートメントにより戻されます。

関連情報:

ジョブ属性検索 (RTVJOBA) コマンド

資源再利用 (RCLRSC) コマンド

前のリリース用のソース・プログラムのコンパイル

CL コンパイラー・コマンドの **ターゲット・リリース** (TGTRLS) パラメーターを使用すると、CL ソース・プログラムを前のリリースで使用するためにコンパイルできます。

TGTRLS パラメーターには、CL プログラムをどのリリースの IBM i オペレーティング・システム上で実行するのかを指定します。指定できるのは、*CURRENT、*PRV、または特定のリリース・レベルです。

TGTRLS(*CURRENT) を指定してコンパイルした CL ソース・プログラムは、現行または将来のリリースのオペレーティング・システムでしか実行できません。 TGTRLS に *CURRENT 以外の値を指定してコンパイルした CL ソース・プログラムは、指定した値以降のリリースで実行できます。

関連情報:

制御言語プログラム作成 (CRTCLPGM) コマンド

前のリリース (*PRV) ライブラリー

CL コンパイラーは、前のリリースのコマンドとファイルに関する情報を CL 前リリース (*PRV) ライブラリーから検索します。

前のリリースのサポートが入っているのは、システム・ライブラリーとユーザー・ライブラリーという 2 種類のライブラリーです。ライブラリーには QSYSVxRxMx と QUSRVxRxMx という名前が付いています (VxRxMx は、サポートされている前のリリースのバージョン、リリース、およびモディフィケーショ

ン・レベルを表します)。例えば、QUSRV7R1M0 ライブラリーは、IBM i ライセンス・プログラムのバージョン 7 リリース 1 モディフィケーション・レベル 0 を実行するシステムをサポートします。

CL コンパイラーは、サポートされている前のリリース用のコンパイルを行う際、前のリリースのコマンドとファイルがあるかどうかを最初に検査します。前リリース・ライブラリーでコマンドまたはファイルが見つからない場合、システムはライブラリー・リスト (*LIBL) または修飾されたライブラリーの検索を次に実行します。

QSYSVxRxMx ライブラリー: QSYSVxRxMx ライブラリーは、前のリリース用の CL コンパイラー・サポートを導入するのと同時に、一緒に導入されます。 QSYSVxRxMx ライブラリーには、以前の特定のリリースのライブラリー QSYS に存在するコマンド定義オブジェクトと出力ファイル (*OUTFILE) が含まれています。

QUSRVxRxMx ライブラリー: 自分で独自の QUSRVxRxMx ライブラリーを作成し、そこにコマンドとファイルのコピーを、サポート対象の前のリリースで存在していたとおりに保持できます。これは特に、コマンドまたはファイルが現行リリースで変更された場合に重要です。

コンパイラーは、前のリリースのコマンドとファイルを探すとき、QUSRVxRxMx ライブラリーが存在するならこれを最初に検査し、次に QSYSVxRxMx ライブラリーを検査します。

注**:** 前のリリースのユーザー・コマンドとユーザー・ファイルを保持するには、QSYSVxRxMx ライブラリーではなく、QUSRVxRxMx ライブラリーを使用してください。 CL コンパイラーの将来のリリースを導入するときには、前のリリースのサポートも導入されます。前のリリースのサポートを導入すれば、サポートされなくなるリリースの QUSRVxRxMx ライブラリーを削除できるようになります。

ライブラリー・リスト (*LIBL) には、前リリース・ライブラリーを追加しないでください。前リリース・ライブラリーに入っているのは、以前のリリースをサポートするコマンドやファイルであり、現行システムでは実行できません。 CL コンパイラーだけが、前リリース・ライブラリー内のコマンドとファイルを参照および使用します。前のリリースのために提供されているシステム・コマンドは、システムの 1 次言語用のものです。 2 次各国語バージョンは利用できません。

注**:** System/38 (システム/38) 環境でコンパイルした CL プログラムを前のリリース用に保管することはできません。

関連情報:

オブジェクト保管 (SAVOBJ) コマンド

変更済みオブジェクト保管 (SAVCHGOBJ) コマンド

ライブラリー保管 (SAVLIB) コマンド

前のリリース用の CL コンパイラー・サポートの導入

*PRV CL コンパイラー・サポートおよび QSYSVxRxMx ライブラリーを導入できます。

*PRV CL コンパイラー・サポートをインストールするには、次のステップに従ってください。

1. ライセンス・プログラム・メニューを表示するには、次のコマンドを入力します。
   ```
   GO LICPGM
   ```
2. オプション 11 (ライセンス・プログラムの導入) を選択します。
3. 「*PRV CL コンパイラー・サポート (*PRV CL Compiler Support)」という名前のオプションを選択します。すると、QSYSVxRxMx ライブラリーが導入されます。

前のリリース用の CL コンパイラー・サポートを使用しない場合は、次のコマンドを入力してこのサポートを削除します。

```
DLTLICPGM LICPGM(5761SS1) OPTION(9)
```

CL コンパイラー・サポートを削除すると、QSYSVxRxMx ライブラリーはシステムから削除されますが、QUSRVxRxMx ライブラリーは削除されません。 QUSRVxRxMx ライブラリーが必要ない場合は、**ライブラリー削除 (DLTLIB)** コマンドを使用してこれを明示的に削除する必要があります。

プログラムおよびプロシージャー相互間の制御の受け渡しと通信

プログラム呼び出し (CALL)、**結合プロシージャー呼び出し (CALLPRC)**、および **戻り (RETURN)** コマンドを使用して、プログラムとプロシージャー相互間での制御の受け渡しを行うことができます。

各コマンドはそれぞれ少しずつ異なった特性を持っています。呼び出されるプログラムおよびプロシージャーには、制御を渡す際に、情報をパラメーターとして渡すことができます。

CALL コマンドまたは CALLPRC コマンドを実行するプログラムに対して、その作成時に USRPRF(*OWNER) が指定されている場合には、特に注意が必要です。これらのコマンドが、所有者のユーザー・プロファイルのもとで実行されるプログラムの中で処理される場合には、各コマンドのセキュリティー特性が通常とは異なったものとなります。

関連概念:

181 ページの『CL ソース・プログラムの構成要素』
CL ソース・プログラムの構成要素として入力する各ソース・ステートメントが実際には CL コマンドであっても、このソースは、多数の典型的な CL ソース・プログラムで使用される基本的な構成要素に類別できます。

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

関連タスク:

443 ページの『呼び出しスタックの表示』
呼び出しスタックを表示するには、**デバッグ表示 (DSPDBG)** コマンドを使用します。

関連情報:

結合プロシージャー呼び出し (CALLPRC) コマンド

プログラム呼び出し (CALL) コマンド

戻り (RETURN) コマンド

セキュリティー参照

別のプログラムまたはプロシージャーに制御を渡す

別のプログラムまたはプロシージャーに制御を渡すために、異なるオプションを選択できます。

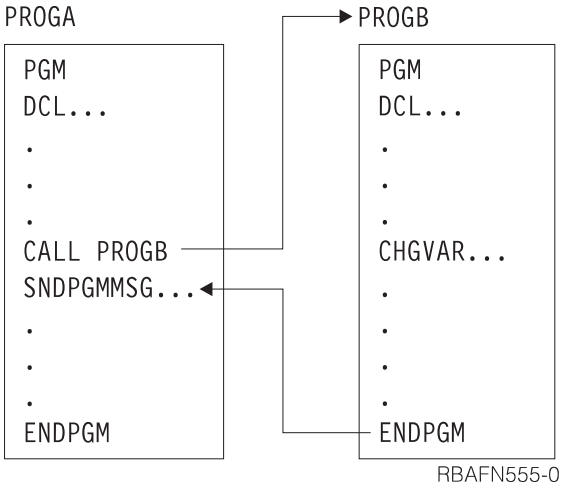
プログラム呼び出しコマンドを使用した制御の受け渡し:

プログラム呼び出し (CALL) コマンドは、指定したプログラムを呼び出し、そのプログラムに制御を渡す場合に使用します。

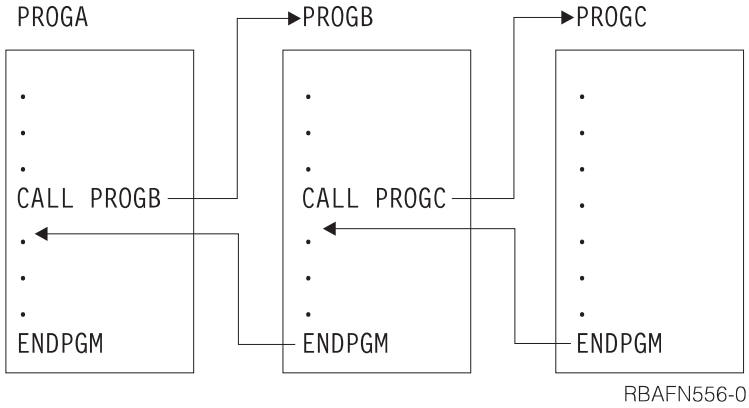
CALL コマンドの形式は次のとおりです。

```
CALL PGM(ライブラリー名 / プログラム名) PARM(パラメーター値)
```

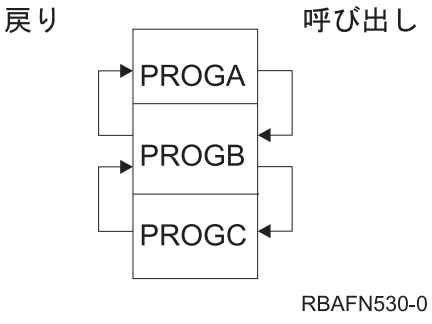
プログラム名およびライブラリー名は、どちらも変数にすることができます。呼び出されたプログラムが、ライブラリー・リスト上にないライブラリーの中にある場合には、PGM パラメーターにプログラムの修飾名を指定しなければなりません。呼び出されたプログラムが実行を完了すると、制御は呼び出し側プログラムの次のコマンドに戻されます。



相互に呼び出しを行う 1 組のプログラムにおける CALL コマンドの順序のことを、呼び出しスタックと呼びます。例えば、次のような一連の呼び出しがあるとします。



この一連の呼び出しにおいて、呼び出しスタックは、次のようになります。



PROGC の処理が終了すると、PROGB の中の、PROGC を呼び出したコマンドの次のコマンドに制御が戻ります。制御はこのようにして、呼び出しスタックの下から上へ戻ります。この戻りは、PROGC が RETURN コマンドまたは ENDPGM コマンドで終わっているかにかかわらず行われます。

CL プログラムはその CL プログラム自身を呼び出すこともできます。

関連タスク:

281 ページの『パラメーターの受け渡し』
他のプログラムまたはプロシージャーに制御を渡す時点で、受取側のプログラムまたはプロシージャー内での変更や使用のために情報を同時に渡すことができます。

285 ページの『プログラム呼び出しコマンドを使用して、呼び出されるプログラムに制御を渡す』
CL プロシージャーで**プログラム呼び出し (CALL)** コマンドを出す場合、呼び出されるプログラムに渡す各パラメーター値には、文字ストリング定数、数値定数、論理定数、または CL 変数のどれでも指定することができます。

関連情報:

プログラム呼び出し (CALL) コマンド

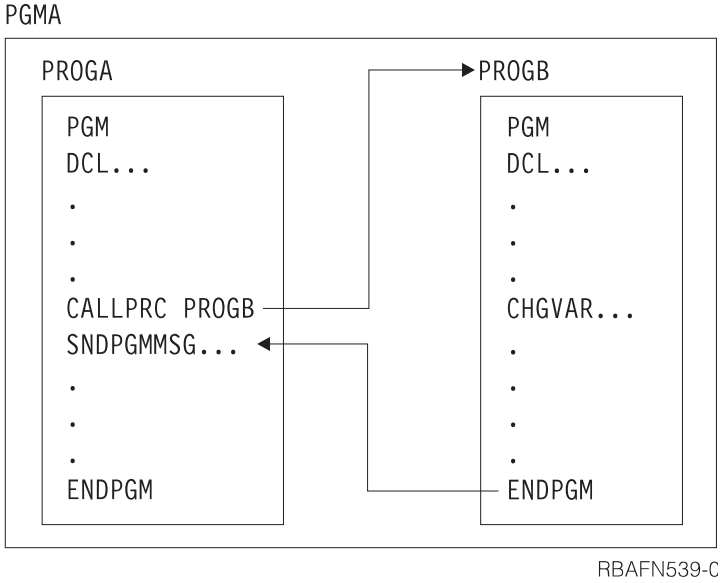
結合プロシージャー呼び出しコマンドを使用した制御の受け渡し:

プロシージャーの呼び出し (CALLPRC) コマンドは、コマンドで指定されたプロシージャーを呼び出し、そのプロシージャーに制御を渡します。

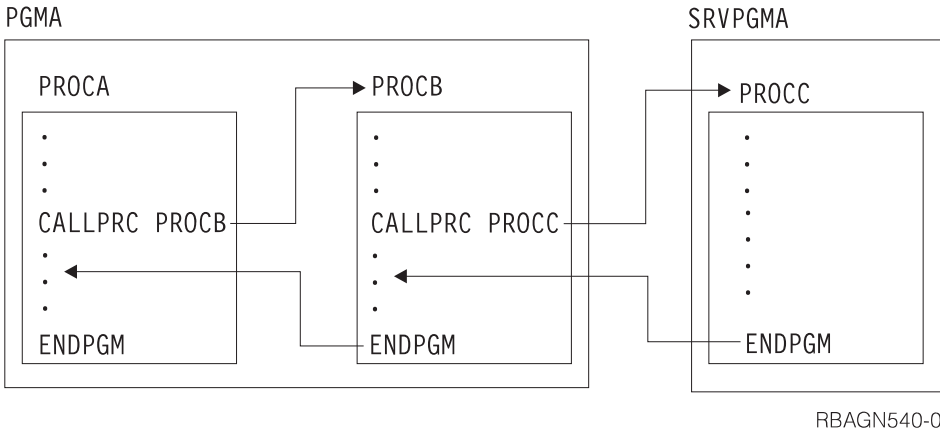
CALLPRC コマンドの形式は次のとおりです。

```
CALLPRC PRC(procedure-name) PARM(parameter-values)
RTNVAL(return-value-variable)
```

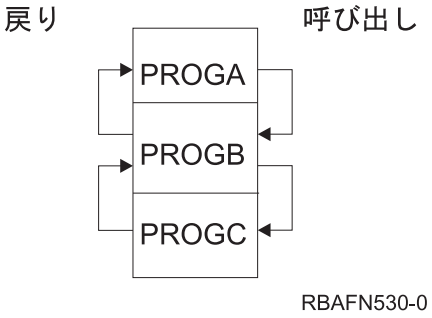
プロシージャー名は変数であってはなりません。呼び出されたプロシージャーが実行を完了すると、制御は呼び出し側プロシージャーの次のコマンドに戻されます。



相互に呼び出しを行う一組のプロシージャーの CALLPRC コマンドの順序が、呼び出しスタックです。例えば、以下のような一連の呼び出しがあるとします。



この一連の呼び出しでは、呼び出しスタックは次のようになります。



PROGC の処理が終了すると、PROGB の中の、PROGC を呼び出したコマンドの次のコマンドに制御が戻ります。制御はこのようにして、呼び出しスタックの下から上へ戻ります。この戻りは、PROGC が **戻り (RETURN)** または **プログラム終了 (ENDPGM)** コマンドで終わっているかにかかわらず行われます。

CL プロシージャーはそれ自身を呼び出すことができます。

関連タスク:

281 ページの『パラメーターの受け渡し』
他のプログラムまたはプロシージャーに制御を渡す時点で、受取側のプログラムまたはプロシージャー内での変更や使用のために情報を同時に渡すことができます。

関連情報:

結合プロシージャー呼び出し (CALLPRC) コマンド

戻りコマンドを使用した制御の受け渡し:

CL プロシージャーまたは OPM プログラム中に**戻り (RETURN)** コマンドがあると、そのプロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムは呼び出しスタックから除去されます。

RETURN コマンドを含むプロシージャーが **統合プロシージャー呼び出し (CALLPRC)** コマンドにより呼び出された場合には、制御は呼び出し側プログラム中のその **CALLPRC** コマンドの次の順番のステートメントに戻されます。

メッセージ・モニター (MONMSG) コマンドで、**RETURN** コマンドで終わる処理が指定されている場合には、制御はその **MONMSG** コマンドを含むプロシージャーまたはプログラムを呼び出したステートメントの、次の順番のステートメントに戻されます。

RETURN コマンドにはパラメーターはありません。

注: 初期プログラムに **RETURN** コマンドが含まれている場合には、コマンド入力画面が表示されます。セキュリティーの観点から、この画面の表示を避けたい場合もあります。

関連情報:

結合プロシージャー呼び出し (CALLPRC) コマンド

メッセージ・モニター (MONMSG) コマンド

戻り (RETURN) コマンド

パラメーターの受け渡し

他のプログラムまたはプロシージャーに制御を渡す時点で、受取側のプログラムまたはプロシージャー内での変更や使用のために情報を同時に渡すことができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

プログラムまたはプロシージャーに渡す情報は、**プログラム呼び出し (CALL)** コマンドまたは **結合プロシージャー呼び出し (CALLPRC)** コマンドの PARM パラメーターで指定することができます。この 2 つのコマンドは、その特性および要件が多少異なっています。

例えば、PROGA に次のようなコマンドが入っているとします。

```
CALL PROGB PARM(&AREA)
```

このコマンドは PROGB を呼び出して、&AREA の値をそのプログラムまたはプロシージャーに渡します。 PROGB は PGM コマンドで始まっていなければならず、またそのコマンドにも、そのプログラムまたはプロシージャーが受け取るパラメーターが指定されていなければなりません。

```
PGM PARM(&AREA) /* PROGB */
```

プログラム呼び出し (CALL) コマンドまたは**結合プロシージャー呼び出し (CALLPRC)** コマンドの場合、相手のプログラムまたはプロシージャーに渡すパラメーターを PARM パラメーターに指定するとともに、受取側のプログラムまたはプロシージャーの PGM コマンドにも、渡されるパラメーターを指定する必要があります。パラメーターは、名前ではなく位置に基づいて渡されるので、**プログラム呼び出し (CALL)** コマンドまたは**結合プロシージャー呼び出し (CALLPRC)** コマンドによって渡す値の位置は、受取側の PGM コマンド上での位置と対応していなければなりません。例えば、PROGA に次のようなコマンドが含まれているとします。

```
CALL PROGB PARM(&A &B &C ABC)
```

このコマンドは、3 つの変数と 1 つの文字ストリングを渡します。そして PROGB が次のコマンドで始まっているとします。

```
PGM PARM(&C &B &A &D) /*PROGB*/
```

この場合、PROGA の &A の値は PROGB の &C の値として使用され、以下順番に両者の値が対応していきます。PROGB の &D は ABC となります。PROGB での DCL ステートメントの順序は重要ではありません。どの変数が渡されるかは、パラメーターが PGM ステートメントに指定された順序だけに依存します。

パラメーターの位置 (指定順序) だけでなく、その長さおよびタイプにも注意する必要があります。受取側のプロシージャーまたはプログラムでリストされるパラメーターは、渡す側のプロシージャーまたはプログラムでのパラメーターと同じ長さおよびタイプで宣言されていなければなりません。 10 進定数は、常に (15 5) の長さで渡されます。

結合プロシージャー呼び出し (CALLPRC) コマンドを使用して文字ストリング定数を渡す場合は、正確なバイト数を指定し、正確にそのバイト数を渡す必要があります。呼び出されるプロシージャーは、操作記述子の情報を使用して、渡される正確なバイト数を判別することができます。操作記述子は、API CEEDOD を使用してアクセスすることができます。

CALL コマンドを使用すると、32 バイト以下の文字ストリング定数は、常に 32 バイトの長さで渡されます。ストリングが 32 バイトより長い場合には、正確なバイト数を指定し、正確にそのバイト数を渡さなければなりません。

次に &VAR1 の値を受け取るプロシージャーまたはプログラムの例を示します。

```
PGM PARM(&VAR1)  /*PGMA*/
DCL VAR1 *CHAR LEN(36)
.
.
.
ENDPGM
```

CALL コマンドまたは **結合プロシージャー呼び出し (CALLPRC)** コマンドには 36 文字を指定しなければなりません。

```
CALLPRC PGMA(ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJ)
```

次の例ではデフォルトの長さが指定されています。

```
PGM PARM(&P1 &P2)
DCL VAR(&P1) TYPE(*CHAR) LEN(32)
DCL VAR(&P2) TYPE(*DEC) LEN(15 5)
IF (&P1 *EQ DATA) THEN(CALL MYPROG &P2)
ENDPGM
```

このプログラムを呼び出すには、次のように指定します。

```
CALL PROG (DATA 136)
```

&P1 には DATA という文字ストリングが渡され、 &P2 には 136 という 10 進数値が渡されます。

ローカル側で定義された変数を参照すると、渡された変数を参照した場合よりもオーバーヘッドが少なく済みます。したがって、呼び出されたプロシージャーまたはプログラムが渡された変数を頻繁に参照する場合、渡された値をローカル変数にコピーし、渡された値ではなくローカル側で定義された値を参照することにより、パフォーマンスを改善できます。

オリジナル・プログラム・モデル (OPM) CL プログラムが呼び出される場合、このプログラムに渡されるパラメーターの数は、プログラムが予期している数と完全に一致していなければなりません。予期される数は、プログラムの作成時に決まります (オペレーティング・システムは、プログラムが予期しているのと異なる数のパラメーターをユーザーが呼び出すことがないようにします)。 ILE プログラムまたはプロシージャーを呼び出す場合、オペレーティング・システムは呼び出しで渡されるパラメーターの数を検査しません。さらに、オペレーティング・システムがパラメーターを保管するスペースは、プログラムまたはプロシージャー呼び出しごとに再初期化されません。 n 個のパラメーターを予期しているプロシージャーを n-1 個のパラメーターで呼び出すと、パラメーター・スペースに何が入っていようと、システムはこれを使用して n 番目の パラメーターにアクセスしようとします。この処置の結果は、まったく予想できません。これ

は、CL プログラムまたはプロシージャーを呼び出したり、CL プログラムまたはプロシージャーで呼び出されたりする、他の ILE 言語で作成されたプログラムまたはプロシージャーにも当てはまります。

これにより、可変長パラメーター・リストを持つプログラムまたはプロシージャーを作成できるので、ILE CL プログラムまたはプロシージャーの作成がさらに柔軟に行えます。例えば、あるパラメーターの値に応じて、リスト内の後方で指定されたパラメーターが不要になることがあります。オプショナル・パラメーターが未指定であるということを制御パラメーターが示した場合、呼び出されたプログラムまたはプロシージャーはそのオプショナル・パラメーターを参照しません。

結合プロシージャー呼び出し (CALLPRC) コマンドのパラメーター・リストから除外したいどのパラメーターについても、特殊値 *OMIT を指定することができます。あるパラメーターに *OMIT を指定すると、呼び出されるプログラムまたはプロシージャーはヌル・ポインターを渡します。呼び出されるプログラムまたはプロシージャーは、除外されているパラメーターを参照する場合、ヌル・ポインターを処理できるようになっていなければなりません。制御言語 (CL) では、除外可能パラメーターを初めて参照するときに MCH3601 を監視することにより、ヌル・ポインターがあるかどうかを検査できます。 MCH3601 を受け取ったら、プログラムまたはプロシージャーは適切な処置を実行しなければなりません。

注: 特殊値 *OMIT は、**結合プロシージャー呼び出し (CALLPRC)** コマンドの PARM パラメーターにのみ有効です。

プログラムまたはプロシージャーを呼び出す際、引数を参照または値によって渡すことができます。

次の例には 2 つの CL プロシージャーがあります。最初のプロシージャーは 1 つのパラメーターを予期しています。このパラメーターが指定されていないと、結果は予測できません。最初のプロシージャーは、別のプロシージャー PROC1 を呼び出します。 PROC1 は、1 つまたは 2 つのパラメーターを予期しています。最初のパラメーターの値が '1' であれば、2 番目のパラメーターは指定されていると見なされます。最初のパラメーターの値が '0' であれば、2 番目のパラメーターは指定されていないと見なされ、デフォルト値が使用されます。 PROC1 はまた、CEEDOD API を使用して、2 番目のパラメーターに渡された実際の長さを判別します。

```
MAIN: PGM   PARM(&TEXT)/* &TEXT must be specified.  Results will be +
                unpredictable if it is omitted.*/
      DCL   VAR(&TEXT) TYPE(*CHAR) LEN(10)
      CALLPRC   PRC(PROC1) PARM('0')
      CALLPRC   PRC(PROC1) PARM('1' &TEXT)
      CALLPRC   PRC(PROC1) PARM('1' 'Goodbye')
      ENDPGM

PROC1: PGM   PARM(&P1 &P2) /* PROC1 - Procedure with optional +
           parameter &P2 */
     DCL  VAR(&P1) TYPE(*LGL) /*Flag which indicates +
         whether or not &P2 will be specified. If +
         value is '1', then &P2 is specified */
     DCL   VAR(&P2) TYPE(*CHAR) LEN(10)
     DCL   VAR(&MSG) TYPE(*CHAR) LEN(10)
     DCL   VAR(&PARMPOS) TYPE(*CHAR) LEN(4) /* +
          Parameter position for CEEDOD*/
     DCL   VAR(&PARMDESC) TYPE(*CHAR) LEN(4) /* +
          Parameter description for CEEDOD*/
     DCL   VAR(&PARMTYPE) TYPE(*CHAR) LEN(4) /* +
          Parameter datatype from CEEDOD*/
     DCL   VAR(&PARMINFO1) TYPE(*CHAR) LEN(4) /* +
          Parameter information from CEEDOD */
     DCL   VAR(&PARMINFO2) TYPE(*CHAR) LEN(4) /* +
          Parameter information from CEEDOD */
     DCL   VAR(&PARMLEN) TYPE(*CHAR) LEN(4) /* +
          Parameter length from CEEDOD*/
     DCL   VAR(&PARMLEND) TYPE(*DEC) LEN(3 0) /* +
          Decimal form of parameter length*/
     IF   COND(&P1) THEN(DO) /* Parm 2 is+
          specified, so use the parm value for the +
          message text*/
     CHGVAR  VAR(%BIN(&PARMPOS 1 4)) VALUE(2) /* Tell +
          CEEDOD that we want the operational +
          descriptor for the second parameter*/
     CALLPRC  PRC(CEEDOD) PARM(&PARMPOS &PARMDESC +
          &PARMTYPE &PARMINFO1 &PARMINFO2 &PARMLEN) +
          /* Call CEEDOD to get the length of data +
          specified for &P2*/
      CHGVAR  VAR(&PARMLEND) VALUE(%BIN(&PARMLEN 1 4)) /* +
              Convert the length returned by CEEDOD to +
              decimal format*/
      CHGVAR  VAR(&MSG) VALUE(%SST(&P2 1 &PARMLEND)) /* +
            Copy the data passed in to a local variable*/
      ENDO
      ELSE   CMD(CHGVAR VAR(%MSG) VALUE('Hello')) /* Use +
             "Hello" for the message text*/
      SNDPGMMSG MSG(&MSG)
      ENDPGM
```

関連タスク:

277 ページの『プログラム呼び出しコマンドを使用した制御の受け渡し』

プログラム呼び出し (CALL) コマンドは、指定したプログラムを呼び出し、そのプログラムに制御を渡す場合に使用します。

279 ページの『結合プロシージャー呼び出しコマンドを使用した制御の受け渡し』

プロシージャーの呼び出し (CALLPRC) コマンドは、コマンドで指定されたプロシージャーを呼び出し、そのプロシージャーに制御を渡します。

410 ページの『例: 制御権転送コマンドの使用』

以下は、パフォーマンスを向上させる制御権転送の例です。

関連情報:

結合プロシージャー呼び出し (CALLPRC) コマンド

プログラム呼び出し (CALL) コマンド

Retrieve Operational Descriptor Information (CEEDOD) API

 ILE の概念

プログラム呼び出しコマンドを使用して、呼び出されるプログラムに制御を渡す:

CL プロシージャーで**プログラム呼び出し (CALL)** コマンドを出す場合、呼び出されるプログラムに渡す各パラメーター値には、文字ストリング定数、数値定数、論理定数、または CL 変数のどれでも指定することができます。

最高 255 個のパラメーターを、呼び出されるプログラムに渡すことができます。パラメーターの値は **CALL** コマンドで指定した順序に従って渡されます。この順序は、呼び出されるプログラムのパラメーター・リスト上でのパラメーターの指定順序に一致していなければなりません。ただし、渡される変数の名前は、受取側のパラメーター・リストの対応する変数名と同じである必要はありません。呼び出されるプログラムで値を受け取る変数の名前は、そのプログラムに対して宣言されていなければなりませんが、宣言コマンドの順序は任意です。

呼び出されるプログラム内の記憶域と、そのプログラムが受け取る変数の間に関係はありません。呼び出し側プログラムが変数を渡す場合、その変数の記憶域は、その変数が最初に宣言されているプログラムの中にあります。システムはアドレスによって変数を渡します。定数が渡される場合には、呼び出し側プログラムがその定数のコピーを作成し、呼び出されるプログラムにはそのコピーのアドレスが渡されます。

上記の結果として、変数を渡された場合に、呼び出されたプログラムがその変数の値を変更すると、その変更は呼び出し側プログラムにも反映されます。したがって、後で使用できるようにするために新しい値を呼び出し側プログラムに返す必要はありません。つまり、呼び出し側プログラムに返す変数について特別のコーディングの必要はありません。定数が渡された場合には、呼び出されたプログラムがその値を変更したとしても、変更後の値は呼び出し側プログラムには知らされません。したがって、呼び出し側プログラムが再び同じプログラムを呼び出した場合、定数の値は初期設定し直されますが、変数の値は初期設定し直されません。

上記の説明の例外として、**CALL** コマンドを使用して統合言語環境 (ILE) C プログラムを呼び出す場合があります。 **CALL** コマンドを使用して ILE C プログラムを呼び出し、文字や論理定数を渡す場合、システムはヌル文字 (x'00') を最後の非ブランク文字の直後に追加します。定数が、単一引用符で囲まれた文字ストリングか、16 進定数である場合、ヌル文字は指定された最後の文字の直後に追加されます。これにより、末尾ブランク (X'40' 文字) は保存されます。数値はヌルで終了することはありません。

コンパイルされていない **CALL** コマンドを使用して CL プログラムを呼び出す場合 (対話式 **CALL** コマンド、または SBMJOB コマンドを介して)、受取側のプログラムでは 10 進パラメーター (*DEC) を LEN(15 5)、文字パラメーター (*CHAR) を LEN(32) 以下で宣言してください。

CALL コマンドを CL プロシージャーまたはプログラム以外で実行する場合、変数を引数として渡すことができません。 **CALL** コマンドを TYPE(*CMDSTR) と定義したコマンド・パラメーターとして指定する場合は、注意が必要です。このように指定すると、PARM パラメーターに指定した変数の内容はすべて、定数に変換されます。 **ジョブ投入 (SBMJOB)** コマンド、**ジョブ・スケジュール項目追加 (ADDJOBSCDE)** コマンド、または**ジョブ・スケジュール項目変更 (CHGJOBSCDE)** コマンドにおける、コマンド (CMD) パラメーターがその例です。

パラメーターの受け渡しは次のように行われます。

- 32 バイト以下の文字ストリング定数は、常に 32 バイトとして (右側にブランクが埋め込まれて) 渡されます。文字定数が 32 バイトより長い場合には、その長さの定数全体が渡されます。パラメーターが 32 バイトより大きいものとして定義されている場合には、**CALL** コマンドは正確にそのバイト数分の定数を渡さなければなりません。 32 文字より長い定数については、受取側のプログラムで予定されている長さに合わせるための埋め込みは行われません。

  受取側のプログラムは、渡したバイト数より少ないバイト数しか受け取ることができません。例えば、プログラムで受け取ることのできるのは 4 文字であると指定されていて、ABCDEF が渡された場合 (26 文字のブランクが埋め込まれている)、プログラムが受け入れて使用するのは ABCD だけです。

  受取側のプログラムが、渡したバイト数より多いバイト数を受け取った場合、その結果は予期できません。文字として渡す数字は単一引用符で囲まなければなりません。
- 10 進定数は、長さ LEN(15 5) のパック 10 進形式で渡されます。15 は値全体の長さで、5 は小数部分の桁数です。したがって、12345 というパラメーターが渡されるとすれば、受取側のプログラムでは 10 進数フィールドが LEN(15 5) として宣言されていなければならず、そのパラメーターは 12345.00000 として受け取られます。

  呼び出されるプログラムに数値定数を渡す必要があり、かつそのプログラムが「15 5」以外の長さと精度でその値の受取が想定されている場合には、その定数を 16 進数形式でコーディングすることができます。次の **CALL** コマンドは、LEN(5 2) として宣言されているプログラム変数に値 25.5 を渡す方法を示しています。

  ```
  CALL PGMA PARM(X'02550F')
  ```
- 論理定数は 32 バイトの長さを持つものとして渡されます。論理値 0 または 1 は第 1 バイトにあり、その他のバイトはブランクです。論理値で動作するプログラムに 0 または 1 以外の値が渡されると、その結果は予期できないものになることがあります。
- 浮動小数点リテラルまたは浮動小数点特殊値 (*NAN、*INF、または *NEGINF) は、8 バイトを占める倍精度の値として渡されます。 CL プログラムは浮動小数点数を処理することはできませんが、浮動小数点数値を文字変数に受け入れて、その変数を浮動小数点数の処理が可能な高水準言語 (HLL) プログラムに渡すことはできます。
- CL プロシージャーまたはプログラムから呼び出しを行う場合には、システムは変数を渡すことができます。その場合には、受取側のプログラムでは、呼び出し側の CL プログラムで定義されている変数に対応するフィールドが宣言されていなければなりません。例えば、ある CL プロシージャーまたはプログラムで、&CHKNUM という名前の 10 進変数が LEN(5 0) として定義されているとすれば、受取側のプログラムでは、対応するフィールドとして長さが 5 文字で小数部分のないパック 10 進数のフィールドが宣言されていなければなりません。 CL プロシージャーまたはプログラムで SBMJOB コマンドを使用して、**CALL** コマンドをバッチ・モード実行する場合には、引数として渡される変数をシステムは定数として処理します。
- 呼び出されるプログラムに 10 進定数またはプログラム変数のどちらでも渡すことができる場合、パラメーターは LEN(15 5) として定義されていなければならず、また呼び出し側プログラムはすべてその定義に従っていなければなりません。呼び出し側プログラムと受取側のプログラムとの間で、パラメーターのタイプ、数、順序、または長さの相違 (上記で文字定数について述べた長さに関する例外を除く) があった場合には、予測できない結果になることがあります。
- 整数定数を整変数に渡すには、その定数を 16 進形式で指定する必要があります。 16 進定数と整変数のサイズは同じでなければなりません。
- ヌル値を他のプログラムに渡すことはできないので、 *N の値によってヌル値を指定することはできません。

次の例では、プログラム A は、1 個の論理定数、3 個の変数、1 個の文字定数および 1 個の数値定数の、6 つのパラメーターを渡します。

```
PGM /* PROGRAM A */
DCL VAR(&B) TYPE(*CHAR)
DCL VAR(&C) TYPE(*DEC) LEN(15 5) VALUE(13.529)
DCL VAR(&D) TYPE(*CHAR) VALUE('1234.56')
CHGVAR VAR(&B) VALUE(ABCDEF)
CALL PGM(B) PARM('1' &B &C &D XYZ 2)  /* Note blanks between parms */
.
.
.
ENDPGM
```

```
PGM PARM(&A &B &C &W &V &U) /* PROGRAM B */
DCL VAR(&A) TYPE(*LGL)
DCL VAR(&B) TYPE(*CHAR) LEN(4)
DCL VAR(&C) TYPE(*DEC)
            /* Default length (15 5) matches DCL LEN in program A */
DCL VAR(&W) TYPE(*CHAR)
DCL VAR(&V) TYPE(*CHAR)
DCL VAR(&U) TYPE(*DEC)
.
.
. ENDPGM
```

注: PGMB に渡される 5 番目のパラメーターが XYZ ではなく 456 で英数字データとして渡される場合、パラメーターには '456' として指定する必要があります。

論理定数 '1' は、呼び出し側プログラムで宣言する必要はありません。これは、プログラム B で、論理タイプの &A という名前で宣言されています。

&B に対する DCL コマンドでは長さは指定されていないので、デフォルトの長さである 32 文字が渡されます。 &B の 6 文字のみが指定されます (ABCDEF)。しかし、プログラム B では &B は 4 文字として宣言されているので、受け取られるのは 4 文字だけです。この値がプログラム B の中で変更された場合には、この呼び出しの効力が続いている間は、プログラム A でも &B の最初の 4 文字は変更されます。

プログラム A では、&C に対して長さ (LEN) パラメーターを指定しなければなりません。長さを指定しなかった場合には、デフォルト値により指定された値の長さになるので、プログラム B で想定されているデフォルトの長さと一致しなくなります。 &C の値は 13.52900 です。

プログラム B の &W (プログラム A から渡される &D) は、文字として宣言されているので、文字として受け取られます。 TYPE が *CHAR であれば、ストリングを指定する単一引用符は不要です。プログラム A ではデフォルト値により、長さは値の長さすなわち 7 になります (文字ストリングでは小数点も 1 文字として数えます)。プログラム B では 32 文字の長さが予定されています。したがって、最初の 7 文字は渡されますが、8 文字目以降の部分の内容は予測できません。

変数 &V は文字ストリング XYZ で、右側にブランクが埋め込まれます。変数 &U は数字データ 2.00000 です。

関連概念:

325 ページの『CL コマンド・パラメーター値の長さ』
長さ (LEN) パラメーターを使用して、パラメーターの長さを指定します。

関連タスク:

277 ページの『プログラム呼び出しコマンドを使用した制御の受け渡し』
プログラム呼び出し (CALL) コマンドは、指定したプログラムを呼び出し、そのプログラムに制御を渡す場合に使用します。

関連情報:

プログラム呼び出し (CALL) コマンド

CL コマンド検索プログラム

プログラムおよびプロシージャー呼び出し時の共通エラー:

値が、**プログラム呼び出し (CALL)** コマンドまたは **結合プロシージャー呼び出し (CALLPRC)** コマンドに渡される際に、エラーが生じる場合があります。これらのエラーには、デバッグが困難なものや、プログラムの機能に重大な影響を及ぼすものもあります。

CALL コマンド使用時のデータ・タイプ・エラー:

プログラム呼び出し (CALL) コマンドの使用中に、データ・タイプ・エラーが生じる場合があります。

コマンド・ストリングの全長には、コマンド名、スペース、パラメーター名、括弧、変数の内容、および使用されている単一引用符などが含まれています。大部分のコマンドでは、コマンド・ストリングがコマンド処理プログラムを開始します。ただし、コマンドによっては変数の一部を期待どおりには渡さない場合があります。

ジョブ投入 (SBMJOB) コマンドに CMD パラメーターを指定して、**プログラム呼び出し (CALL)** コマンドを使用すると、予期しない結果になる可能性があります。 CMD パラメーターに CALL コマンドを指定して使用するのは、CMD パラメーターを CALL コマンドへのコンパイラー指示として使用するのと構文的に同じです。 **プログラム呼び出し (CALL)** コマンドは、CMD パラメーターを指定して使用すると、コマンド・ストリングに変換されます。このコマンド・ストリングは、バッチ・サブシステムが後でコマンドを開始するときに実行されます。 **プログラム呼び出し (CALL)** コマンドを単独で使用すると、CL コンパイラーは呼び出しを実行するためのコードを生成します。

10 進定数および文字変数に共通する問題が生じることがしばしばあります。次のような場合、コマンド・ストリングは要求どおりには構成されません。

- 10 進数が 10 進定数に変換された場合。

  コマンド・ストリングの実行時に、この 10 進定数は長さ LEN(15 5) のパック形式で渡されます。 CL 変数で指定されている形式では渡されません。
- 宣言された文字変数が 32 文字より長い場合。

文字変数の内容は上記のように、通常は末尾ブランクを除去してある引用符付き文字定数として渡されます。その結果として、呼び出されるプログラムに十分なデータが渡されない場合があります。

以下に、コマンド・ストリングの構成における誤りの訂正に使用できる方法を示します。

- 投入する **プログラム呼び出し (CALL)** コマンド・ストリングを作成します。これは、コマンドの各種の部分を連結して 1 つの CL 変数にまとめることにより作成されます。 **ジョブ投入 (SBMJOB)** コマンドの要求データ (RQSDTA) パラメーターを使用して、コマンド・ストリングを投入します。
- CL 文字変数が 32 文字より多くて末尾ブランクが有効な場合、1 文字余分な変数を作成し、非ブランク文字を最後の位置にサブストリング化します。こうしておくと、有効なブランクは切り捨てられません。呼び出されるプログラムは、その余分の文字が予測している長さを超えているので、その文字を無視することになります。
- 呼び出されるプログラムを開始するコマンドを作成します。 **プログラム呼び出し (CALL)** コマンドを使用する代わりに、この新規のコマンドを投入します。コマンド定義は、パラメーターがコマンド処理プログラムに予測どおりに渡されていることを保証します。

関連概念:

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

関連情報:

プログラム呼び出し (CALL) コマンド

ジョブ投入 (SBMJOB) コマンド

CL コマンド検索プログラム

パラメーター引き渡し中のデータ・タイプ・エラー*:*

プログラム呼び出し (CALL) コマンドの使用中に生じるエラーの他に、その他のタイプのデータ・タイプ・エラーが生じる場合があります。

値を渡す場合、呼び出しプロシージャーまたはプログラムと、呼び出されるプロシージャーまたはプログラムとの間で、データ・タイプ (TYPE パラメーター) が同じ (*CHAR、*DEC、または *LGL) でなければなりません。これに関するエラーは、数値定数を渡そうとする場合によく起こります。数値定数を単一引用符で囲んだ場合には、文字ストリングとして渡されます。しかし定数を単一引用符で囲まない場合、定数は LEN(15 5) を指定したパック 10 進数フィールドとして渡されます。

次の例では、10 進数値の受取を予期しているプログラムに引用符付きの数字が渡されます。したがって、呼び出されるプログラム (PGMA) で変数 &A が参照された時点で、10 進データ・エラー (エスケープ・メッセージ MCH1202) が起こります。

```
CALL   PGMA PARM('123')  /* CALLING PROGRAM */
PGM    PARM(&A)  /* PGMA */
DCL    &A *DEC LEN(15 5)  /* DEFAULT LENGTH */
.
.
.
IF (&A *GT 0) THEN(...) /* MCH1202 OCCURS HERE */
```

次の例では、文字変数を定義しているプログラムに 10 進数値が渡されます。通常、このタイプのエラーは実行時の障害とはなりませんが、誤った結果になります。

```
CALL PGMB PARM(12345678)  /* CALLING PROG */

PGM PARM(&A)               /* PGMB */
DCL &A *CHAR 8
.
.
.
ENDPGM
```

PGMB 内の変数 &A は、001234567800000F という 16 進値を持ちます。

一般に、論理 (*LGL) 変数から文字 (*CHAR) 変数へ、またはその逆のデータの受け渡しでは、値が '0' または '1' のどちらかである限り、エラーは起こりません。

10 進数の桁数と精度のエラー*:*

10 進数値が間違った 10 進の桁数と精度で渡されると、エラーが生じる場合があります。

10 進数値が渡される際に、呼び出しプロシージャーまたはプログラムに 10 進数の長さや精度の間違いがあると、 (長過ぎる場合または短すぎる場合のどちらについても) 該当の変数が参照された時点で 10 進デ

ータ・エラー (MCH1202) が発生します。次の例では、数値定数は、LEN(15 5) として引き渡されますが、呼び出されるプロシージャーまたはプログラムでは LEN(5 2) と宣言されています。数値定数は、常にパック 10 進数 (15 5) として渡されます。

```
CALL PGMA PARM(123)     /* CALLING PROG */

PGM PARM(&A)                  /* PGMA */
DCL &A *DEC (5 2)
.
.
.
IF (&A *GT 0) THEN(...) /* MCH1202 OCCURS HERE */
```

呼び出し側プログラムまたはプロシージャーにおいて、10 進変数が LEN(5 2) と宣言され、値が定数ではなく変数として引き渡された場合は、エラーは発生しません。

プロシージャーまたはプログラムに数値定数を渡す必要があり、かつそのプロシージャーまたはプログラムが 15 5 以外の長さと精度での値の受取を想定している場合には、その定数を 16 進数形式でコーディングすることができます。次の CALL コマンドは、LEN(5 2) として宣言されているプログラム変数に値 25.5 を渡す方法を示しています。

```
CALL PGMA PARM(X'02550F')
```

渡された 10 進数値の長さが正しく、その精度 (小数部分の桁数) が正しくない場合には、受取側のプロシージャーまたはプログラムはその値を正しく解釈することができません。例えば、次の例では、プロシージャーに渡される数値定数の値 (長さ (15 5) は 25124.00) と解釈されてしまいます。

```
CALL PGMA PARM(25.124)  /* CALLING PGM */

PGM PARM(&A)            /* PGMA */
DCL &A *DEC (15 2)      /* LEN SHOULD BE 15 5*/
.
.
.
ENDPGM
```

このようなエラーは、変数の受け渡しまたは宣言の時点ではなく、その変数が参照される最初の時点で生じます。次の例では、呼び出されるプログラムは該当の変数を参照することはなく、値 (誤った長さの) を呼び出し側プログラムに返される変数に入れます。したがって、このエラーは、その変数が呼び出し側プログラムに返され、そこで最初に参照されるまで検出されません。このようなエラーは、最も検出しにくいエラーの 1 つです。

```
PGM           /* PGMA */
DCL &A *DEC (7 2)
CALL PGMB PARM(&A)  /* (7 2) PASSED TO PGMB */
IF (&A *NE 0)  THEN(...)  /* *MCH1202 OCCURS HERE */
.
.
.
ENDPGM

PGM PARM(&A) /* PGMB */
DCL &A *DEC (5 2)        /* WRONG LENGTH */
.
.
.
 CHGVAR &A (&B-&C)      /* VALUE PLACED in &A */
RETURN
```

制御がプログラム PGMA に戻り、&A が参照された時点で、エラーが起こります。

文字の長さエラー*:*

正しくない長さの文字値を渡すと、エラーが生じる場合があります。

受取側の変数として宣言されている文字数より長い文字値を渡す場合には、受取側のプロシージャーまたはプログラムはその超過した部分にアクセスすることができません。次の例では、PGMB は渡された変数をブランクに変更します。 PGMB では、その変数が LEN(5) として宣言されているので、5 文字分だけがブランクに変更されますが、PGMA でその値が参照される際には、残りの文字はそのまま値の一部として残っています。

```
PGM        /* PGMA */
DCL &A *CHAR 10
CHGVAR &A 'ABCDEFGHIJ'
CALL PGMB PARM(&A)           /* PASS to PGMB */
.
.
.
IF (&A *EQ '  ') THEN(...)  /* THIS TEST FAILS */
ENDPGM

PGM PARM(&A)     /* PGMB */
DCL &A *CHAR 5   /* THIS LEN ERROR*/
CHGVAR &A '  '   /* 5 POSITIONS ONLY; OTHERS UNAFFECTED */
RETURN
```

このようなエラーに対してはエスケープ・メッセージは出されませんが、このような変数の処理によって意図した結果とは異なる結果を招くことがあります。

プロシージャーまたはプログラムに渡される値が、受取側のプロシージャーまたはプログラムで宣言されている長さより短かった場合には、重大な結果を招く場合があります。この場合には、受取側のプロシージャーまたはプログラムにおける変数の値の最初の部分にはその渡された値が入りますが、残りの部分 (受取側のプログラムで宣言されている長さに達するまでの部分) には、記憶域でその値に続く部分の記憶域の値が入ることになります。この部分の記憶域の内容は予測不能な値です。渡された値がプログラム変数である場合には、その後に別のプログラム変数が続く場合や、プロシージャーまたはプログラムの内部制御構造が続いている場合もあります。渡された値が定数である場合には、その後には、CALL または CALLPRC コマンドによって渡された他の定数、または内部制御構造が続く場合も考えられます。

受取側のプロシージャーまたはプログラムが値を変更すると、元の値およびそれに続く記憶域にも操作を加えます。このような操作により直ちに現れる影響として、他の変数または定数の変更、あるいはプロシージャーまたはプログラムの実行不能の原因となるような内部構造の変更などがあります。記憶域に対する変更は直ちに有効になります。

次の例では、受取側のプログラムに 3 文字の定数が 2 つ渡されます。文字定数は **プログラム呼び出し (CALL)** コマンドに対して最低 32 文字として渡されます。(通常この 3 文字は左寄せにされ、その後に末尾ブランクが付加されます。) 受取側のプログラムで、値を受け取る変数として 32 文字を超える長さの変数が宣言されている場合には、32 文字を超える部分には、その値に対応する記憶域に続く記憶域にある未知の値が入ることになります。この例では、2 つの定数が隣接して記憶域に入っているものとします。

```
CALL PGMA ('ABC' 'DEF')  /* PASSING PROG */

PGM  PARM(&A &B) /* PGMA */
DCL  &A *CHAR 50  /* VALUE:ABC+29' '+DEF+15' ' */
DCL  &B *CHAR 10  /* VALUE:DEF+7' ' */
CHGVAR VAR(&A) (' ')  /* THIS ALSO BLANKS &B */
.
.
.
ENDPGM
```

変数として渡される値もこれとまったく同じです。

次の例では、受取側のプロシージャーに 3 文字の定数が 2 つ渡されます。 **プロシージャーの呼び出し (CALLPRC)** コマンドに対しては、指定された文字数のみが渡されます。受取側のプログラムで、値を受け取る変数として、渡される定数を超える長さの変数が宣言されている場合には、超過部分には、その値に対応する記憶域に続く記憶域にある未知の値が入ることになります。

この例では、2 つの定数が隣接して記憶域に入っているものとします。

```
CALLPRC PRCA ('ABC' 'DEF')  /* PASSING PROG */

PGM  PARM(&A &B)       /*  *PRCA  */
DCL  &A *CHAR 5        /*   VALUE:'ABC' + 'DE' */
DCL  &B *CHAR 3        /*   VALUE:'DEF' */
CHGVAR &A ' '          /*   This also blanks the first two bytes of &B */
.
.
.
ENDPGM
```

関連情報:

結合プロシージャー呼び出し (CALLPRC) コマンド

プログラム呼び出し (CALL) コマンド

プログラムおよびプロシージャー相互間の通信

プログラムおよびプロシージャーは、データ待ち行列およびデータ域を使用して通信することができます。

データ待ち行列の使用

データ待ち行列は、ユーザーが作成することのできるシステム・オブジェクトの 1 つのタイプであり、高水準言語 (HLL) プロシージャーまたはプログラムがこの待ち行列にデータを送り、別の HLL プロシージャーまたはプログラムがそこからそのデータを受け取ることができます。

受取側のプログラムは、データを待ち受けていて直ちに受け取ることも、後で受け取ることもできます。

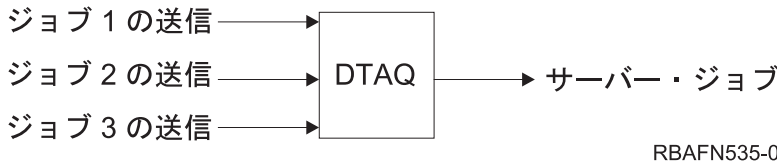
データ待ち行列の使用には次のような利点があります。

- データ待ち行列を利用すれば、ジョブによる作業の一部を軽減することができます。ジョブが対話式ジョブである場合には、これにより応答時間が短縮され、対話式プログラムおよびその処理アクセス・グループ (PAG) のサイズを縮小することができます。これは、システム全体のパフォーマンスの向上にもつながります。例えば、複数のワークステーション・ユーザーが、いくつかのファイルに対する更新操作または追加操作を含む同じトランザクションを入力するような場合に、対話式ジョブがそのトランザクションに対する要求を 1 つのバッチ処理ジョブに投入することによって、システムのパフォーマンスは大幅に向上します。
- データ待ち行列は、2 つのジョブ相互間の非同期的な通信の速度を最大限に高めるための手段として使用することができます。データベース・ファイル、メッセージ待ち行列、またはデータ域を用いてデータの送受信を行うよりも、データ待ち行列を用いてデータの送受信を行う方が、オーバーヘッドが少なく済みます。
- 任意の HLL プロシージャーまたはプログラムで、API QSNDDTAQ、QRCVDTAQ、QCLRDTAQ、QMHRDQM、QMHQRDQD、および QMHQCDQ を呼び出すことにより、データ待ち行列の内容または記述の送信、受信、消去、検索、およびデータ待ち行列の属性の一部を変更することができます。この場合、HLL プロシージャーまたはプログラムを終了したり、データ待ち行列操作を実行するための CL プロシージャーまたはプログラムを呼び出す必要はありません。

- データ待ち行列からデータを受け取る場合には、データ待ち行列に項目が届くまでジョブが待機する時間のタイムアウトを設定することができます。これは、OVRDBF コマンドの EOFDLY パラメーターを用いて、遅延時間が経過すると同時にジョブを活動化するのとは異なります。
- 複数のジョブが同じデータ待ち行列からデータを受け取ることができます。これは、所要のパフォーマンスの制限範囲内で、1 つのジョブでは処理し切れないほどの量の項目を扱わなければならないようなアプリケーションでは、特に役に立ちます。例えば、何台かの印刷装置が注文を印刷できる状態にある場合に、いくつかの対話式ジョブから 1 つのデータ待ち行列に要求を送ることができます。また、各印刷装置ごとの別個のジョブが、先入れ先出し (FIFO)、後入れ先出し (LIFO)、またはキー順のいずれかにより、データ待ち行列からデータを受け取ることができます。
- データ待ち行列には、待ち行列に入れられるメッセージのそれぞれに送信元 ID を付加する機能があります。送信元 ID は、その待ち行列が作成されるときに設定されるそのデータ待ち行列の属性で、修飾ジョブ名と現行ユーザー・プロファイルが入っています。

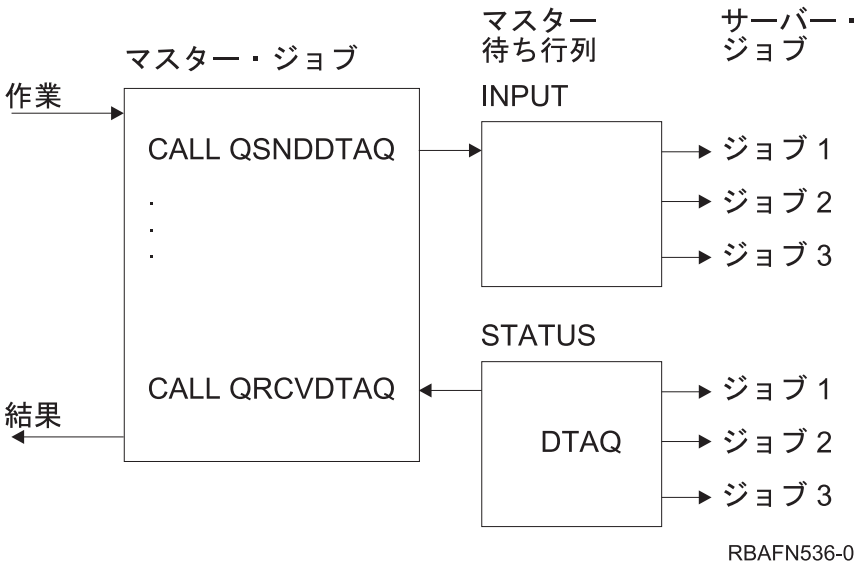
上記の利点に加え、データ待ち行列をジャーナルできます。これにより、異常な初期プログラム・ロード (IPL) または破損が起こったときにオブジェクトが変更アクションの途中であった場合であっても、オブジェクトを一貫性のある状態に回復することができます。また、ジャーナル処理により、リモート・システムのデータ待ち行列ジャーナルの複製も行えます (例えば、リモート・ジャーナルを使用する)。これにより、システムは、類似の環境にアクションを複製して、アプリケーション作業を複製することができます。

次の例は、データ待ち行列の動作を示しています。複数のジョブが 1 つのデータ待ち行列に項目を入れることができます。項目はサーバー・ジョブにより処理されます。この例は、複数のジョブから処理済みの命令を 1 つのジョブに送り、そのジョブに印刷を行わせるために使用することができます。同じ待ち行列にデータを送ることのできるジョブの数に制限はありません。



次に、データ待ち行列の使用例をもう 1 つ示します。この例では、マスター・ジョブが作業要求を受け取り、 (QSNDDTAQ プログラムを呼び出すことにより) 項目をデータ待ち行列に送ります。そして、サーバー・ジョブが、(QRCVDTAQ プログラムを呼び出すことにより) 項目をデータ待ち行列から受け取って、データを処理します。サーバー・ジョブは、別のデータ待ち行列を用いて、状況を伝える情報をマスター・ジョブに返すことができます。

データ待ち行列を用いることにより、マスター・ジョブはサーバー・ジョブにその作業を行わせることができます。これによりマスター・ジョブが解放され、次の作業要求を受け入れることができるようになります。同じデータ待ち行列から受け取ることのできるサーバー・ジョブの数に制限はありません。



データ待ち行列上に項目が 1 つもない場合、サーバー・ジョブとしては以下の選択肢があります。

- 待ち行列に項目が入るまで待つ。
- 一定の時間が経過するまで待つ。それでも項目が入らない場合は、処理を続ける。
- 待機せずに、直ちに制御を返す。

プログラムが、表示装置ファイル、ICF ファイル、およびデータ待ち行列からの入力を同時に待つ必要がある場合にも、データ待ち行列を使用することができます。この場合、次に示すコマンドに DTAQ パラメーターを指定してください。

- **表示装置ファイル作成 (CRTDSPF)** コマンド
- **表示装置ファイル変更 (CHGDSPF)** コマンド
- **表示装置ファイル一時変更 (OVRDSPF)** コマンド
- **ICF ファイル作成 (CRTICFF)** コマンド
- **ICF ファイル変更 (CHGICFF)** コマンド
- **ICF ファイル一時変更 (OVRICFF)** コマンド

次のどちらかが起こった場合に項目が入れられるデータ待ち行列を示すことができます。

- 送信勧誘された表示装置により、使用可能なコマンド・キーまたは実行キーが押された。
- データが、送信勧誘された ICF セッションで使用できるようになった。

システムで実行されているジョブは、QSNDDTAQ プログラムを使用することにより、DTAQ パラメーターに指定したのと同じデータ待ち行列に項目を入れることもできます。

アプリケーション・プログラムは、QRCVDTAQ プログラムを呼び出して、データ待ち行列に入っている各項目を受け取り、その項目を待ち行列に入れたのが表示装置ファイルであるか、ICF ファイルであるか、あるいは QSNDDTAQ プログラムであるかに応じて項目を処理します。

出力待ち行列作成 (CRTOUTQ) コマンドまたは**出力待ち行列変更 (CHGOUTQ)** コマンドを使用することにより、データ待ち行列を出力待ち行列と関連付けることもできます。スプール・ファイルが出力待ち行列で待機 (RDY) 状態にある場合、システムはデータ待ち行列内の項目を記録します。ユーザー・プログラムは、

データ待ち行列受信 (QRCVDTAQ) API を使用してデータ待ち行列からの情報を受信することにより、スプール・ファイルが出力待ち行列でいつ使用できるかを判別することができます。

関連概念:

印刷の基本

関連資料:

300 ページの『例: 表示装置ファイルおよび ICF ファイルからの入力を待機する』
この例は、データ待ち行列を使用して、表示装置ファイルおよび ICF ファイルからの入力を待っているプログラムを示しています。

302 ページの『例: 表示装置ファイルおよびデータ待ち行列からの入力を待機する』
この例は、表示装置ファイルからの入力を待っている、または別のジョブのデータ待ち行列で入力を待っている、ジョブのプログラムを示しています。

関連情報:

CL コマンド検索プログラム

ジャーナル管理

出力待ち行列変更 (CHGOUTQ) コマンド

出力待ち行列作成 (CRTOUTQ) コマンド

リモート・データ待ち行列:

リモート・データ待ち行列は、リモート・システムにあるデータ待ち行列です。

分散データ管理 (DDM) ファイルを使用して、リモート・データ待ち行列にアクセスすることができます。 DDM ファイルにより、1 つのシステムに存在するプログラムがリモート・システムにあるデータ待ち行列へアクセスし、次に示す関数を実行できるようになります。

- データ待ち行列へのデータの送信
- データ待ち行列からのデータの受信
- データ待ち行列からのデータの消去

標準のデータ待ち行列を現在使用しているアプリケーション・プログラムは、アプリケーションを再変更または再コンパイルすることなくリモート DDM データ待ち行列にアクセスすることもできます。正しいデータ待ち行列にアクセスしたかどうかを確かめるには、次のことを行う必要がある場合があります。

- 標準のデータ待ち行列を削除し、最初の標準のデータ待ち行列と同じ名前の DDM データ待ち行列を作成する。
- 標準のデータ待ち行列の名前を変更する。

次のコマンドで DDM データ待ち行列を作成できます。

```
CRTDTAQ DTAQ(LOCALLIB/DDMDTAQ) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/REMOTEDTAQ) RMTLOCNAME(SYSTEMB)
   TEXT('DDM data queue to access data queue on SYSTEMB')
```

上記の例 (「マスター・ジョブ / サーバー・ジョブ」) を応用して、リモート・データ待ち行列とともに使用する DDM データ待ち行列を作成することができます。マスター・ジョブは、SystemA に存在し、データ待ち行列とサーバー・ジョブは SystemB に移動されます。 DDM データ待ち行列 (INPUT および STATUS) を 2 つ作成した後、マスター・ジョブは SystemB に存在するサーバー・ジョブと非同期の通信を続行します。次にリモート・データ待ち行列を使用して DDM データ待ち行列を作成する例を示します。

```
CRTDTAQ DTAQ(LOCALLIB/INPUT) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/INPUT) RMTLOCNAME(SystemB)
   TEXT('DDM data queue to access INPUT on SYSTEMB')

CRTDTAQ DTAQ(LOCALLIB/STATUS) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/STATUS) RMTLOCNAME(SystemB)
   TEXT('DDM data queue to access STATUS on SYSTEMB')
```

マスター・ジョブは、QSNDDTAQ を呼び出し、LOCALLIB/INPUT というデータ待ち行列名を渡し、SystemB のリモート・データ待ち行列 (REMOTELIB/INPUT) にデータを送ります。リモート・データ待ち行列 (REMOTELIB/STATUS) からデータを受け取るために、マスター・ジョブは QRCVDTAQ への呼び出しに対して LOCALLIB/STATUS というデータ待ち行列名を渡します。

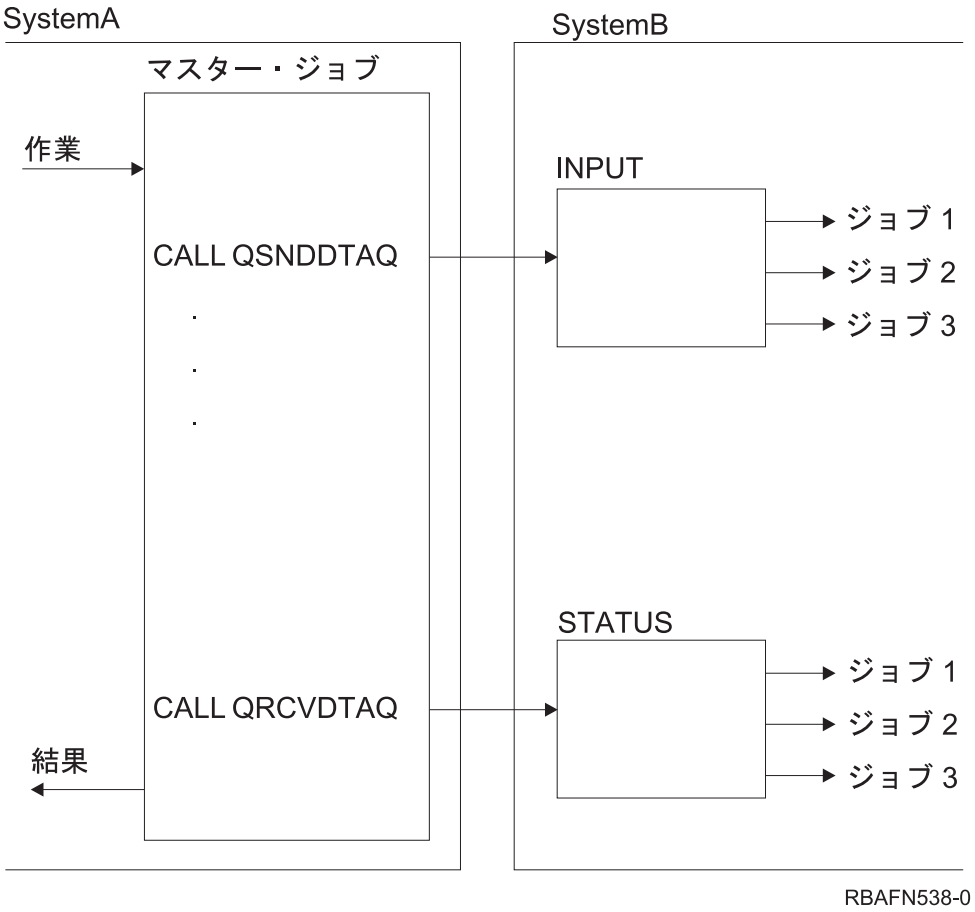


図 *1.* リモート・データ待ち行列へのアクセス例

関連情報:

データ待ち行列作成 (CRTDTAQ) コマンド

分散データベース・プログラミング

データベース・ファイルを待ち行列として使用した場合との比較**:**

データ待ち行列を使用した場合と、データベース・ファイルを使用した場合は、多くの点で異なります。

- データ待ち行列は、大量のデータまたは大量の項目を保管するためではなく、活動状態にあるプロシージャーまたはプログラム相互間の通信を効率よく行うことを目的としています。

- データ待ち行列は、長期間に渡るデータの保管には適していません。この目的のためにはデータベース・ファイルを使用してください。
- データ待ち行列を使用する際には、異常終了後のプロシージャーをプログラムに組み込んで、システムの終了前に処理が完了していなかった項目の回復ができるようにしておいてください。
- データ待ち行列を定期的に (1 日 1 回など) 支障のない時点で削除し、再作成するのは良い方法です。多くの項目が除去されずに残っていると、パフォーマンスに影響する場合があります。データ待ち行列を定期的に作成し直すことにより、データ待ち行列を最適なサイズに戻すことができます。より効率的なアプローチは、自動再利用機能を使用することです。

関連タスク:

『データ待ち行列が使用する記憶域の管理』
データ待ち行列が小さくなると、大きなデータ待ち行列よりもパフォーマンスがよくなるので、データ待ち行列に割り振られる記憶域のサイズの管理が重要になります。

メッセージ待ち行列との類似点:

データ待ち行列は、プロシージャーおよびプログラムが待ち行列にデータを送り、後で別のプロシージャーまたはプログラムがそのデータを受け取ることができるという点では、メッセージ待ち行列によく似ています。

しかし、メッセージ待ち行列では、保留されているデータを受け取ることのできるプログラムは一時点で 1 つだけであるのに対し、データ待ち行列の場合には、複数のプログラムが保留されているデータを同時に受け取ることができます。 (ただし、複数のプログラムが待機していても、データ待ち行列から 1 つの項目を受け取るのは 1 つのプログラムだけです。) データ待ち行列上の項目は、先入れ先出し、後入れ先出し、またはキー順のどの順序でも処理することができます。プログラムが受け取った項目は、データ待ち行列から除去されます。

データ待ち行列を使用するための前提条件:

データ待ち行列を使用する場合には、まず最初に**データ待ち行列作成 (CRTDTAQ)** コマンドを使用してデータ待ち行列を作成しておく必要があります。

次にこのコマンドの例を示します。

```
CRTDTAQ DTAQ(MYLIB/INPUT)  MAXLEN(128)
        TEXT('Sample data queue')
```

MAXLEN パラメーターは必須であり、データ待ち行列に送られる項目の最大長 (1 から 64 512 文字) を指定します。

関連情報:

データ待ち行列作成 (CRTDTAQ) コマンド

データ待ち行列が使用する記憶域の管理:

データ待ち行列が小さくなると、大きなデータ待ち行列よりもパフォーマンスがよくなるので、データ待ち行列に割り振られる記憶域のサイズの管理が重要になります。

各項目は、データ待ち行列に送られる際に、記憶域の割り振りを受け取ります。割り振られる記憶域は、**データ待ち行列作成 (CRTDTAQ)** コマンドで指定された、データ待ち行列の最大項目長として指定された値になります。データ待ち行列から項目を受け取ると、その項目はそのデータ待ち行列から除去されますが、その補助記憶域は解放されるわけではありません。データ待ち行列に新しい項目が送られてくると、システムは再び同じ補助記憶域を使用します。待ち行列に送られてきた項目を受け取らなければ、待ち行列は次第に

大きくなります。項目の初期数を超えて拡張されていない小さい待ち行列のほうが、パフォーマンスが優れています。データ待ち行列が大きくなりすぎた場合には、 **データ待ち行列削除 (DLTDTAQ)** コマンドを使用してデータ待ち行列を削除し、データ待ち行列の削除が完了したら、 **データ待ち行列作成 (CRTDTAQ)** コマンドを使用して作成し直してください。

V4R5M0 以降のリリースでは、別の方法でもデータ待ち行列のサイズを管理することができます。この方法では、**データ待ち行列作成 (CRTDTAQ)** コマンドで SIZE キーワードと AUTORCL キーワードを組み合わせて使用します。 SIZE キーワードを使用すれば、データ待ち行列に入れられる項目の最大数と、データ待ち行列に含まれる項目の初期数を指定することができます。拡張された待ち行列用に AUTORCL キーワードを使用すれば、データ待ち行列が空のときに、記憶域を自動的に再利用するかどうか指定できます。待ち行列に割り振られたままの記憶域の量は、その待ち行列の作成時に指定する項目の初期数と同じです。AUTORCL の値を *NO (デフォルト値) にすると、システムが未使用のスペースから記憶域を自動的に再利用することはありません。データ待ち行列が使用する記憶域を再利用するには、前の段落で説明したように、その記憶域をいったん削除してから再作成するか、データ待ち行列変更 (QMHQCDQ) API を使用して自動再利用属性を変更する必要があります。自動再利用は待ち行列のサイズによってはコスト高になることがあるので、 **データ待ち行列作成 (CRTDTAQ)** コマンドで指定する初期項目数を、そのデータ待ち行列で予期される通常の最大項目数に設定する必要があります。初期項目数を小さく設定しすぎると、自動再利用が実行される頻度が高くなります。

関連概念:

296 ページの『データベース・ファイルを待ち行列として使用した場合との比較』
データ待ち行列を使用した場合と、データベース・ファイルを使用した場合は、多くの点で異なります。

関連情報:

データ待ち行列作成 (CRTDTAQ) コマンド

データ待ち行列削除 (DLTDTAQ) コマンド

データ待ち行列の割り振り:

データ待ち行列のすべてのユーザーが、それを使用する前に割り振ります。これはデータ待ち行列が一度に複数のジョブからアクセスされないようにするのに役立ちます。

同時に複数のジョブからアプリケーションのデータ待ち行列にアクセスできないようにしたい場合は、データ待ち行列を使用する前に、**オブジェクト割り振り (ALCOBJ)** コマンドをアプリケーションに組み込んでおくようにしてください。この場合には、アプリケーションによるデータ待ち行列の使用が終了した時点で、**オブジェクト割り振り解除 (DLCOBJ)** コマンドによってデータ待ち行列の割り振りを解除する必要があります。

ALCOBJ コマンド自体には、他のジョブがデータ待ち行列との間でデータを送受したり、データ待ち行列を消去したりすることを制限する機能はありません。しかし、すべてのアプリケーションでデータ待ち行列の使用の前に ALCOBJ コマンドを組み込むようにコーディングされている場合、既に他のジョブに割り振られているデータ待ち行列の割り振りはできなくなります。そうすることで、同時に複数のジョブが同じデータ待ち行列を使用することを防止することができます。

データ待ち行列が既に別のジョブに割り振られていて、割り振りの要求が失敗した場合には、システムはエラー・メッセージ CPF1002 を出します。アプリケーション・プログラムまたはアプリケーション・プロシージャーで**メッセージ・モニター (MONMSG)** コマンドを使用して、このメッセージを監視し、エラー・メッセージに対する処置を取ることができます。可能な処置としては、ユーザーへのメッセージの表示、またはデータ待ち行列の割り振りの再試行などがあります。

データ待ち行列を変更して、IBM 提供の操作でデータ待ち行列にロックを実施するように指示することができます。V6R1 より前のバージョンでは、ALCOBJ 機能によるデータ待ち行列へのロックは、IBM 提供の操作で無視されます。データ待ち行列のパフォーマンスは、この属性を使用してデータ待ち行列のロックを実施することにより影響を受けることがあります。データ待ち行列を変更してデータ待ち行列にロックを実施する方法と、さまざまなデータ待ち行列操作で使用されるロックに関する詳細は、データ待ち行列の変更 (QMHQCDQ) API を参照してください。

関連タスク:

599 ページの『CL プログラムまたはプロシージャー内のメッセージの監視』

*ESCAPE メッセージ、*STATUS メッセージ、および *NOTIFY メッセージは、モニター可能なメッセージであり、プログラムまたはプロシージャーで使用される各 CL コマンドによって発行されます。

関連情報:

オブジェクト割り振り (ALCOBJ) コマンド

オブジェクト割り振り解除 (DLCOBJ) コマンド

メッセージ・モニター (MONMSG) コマンド

データ待ち行列変更 (QMHQCDQ) API

例: データ待ち行列の使用:

これらの例は、データ待ち行列を処理するさまざまな方法を説明しています。

例: データ待ち行列からのデータの受け取りに、最大 2 時間待機する:

この例は、データ待ち行列のある項目を受け取るのに、最大 2 時間待機するプログラムを示しています。

次の例では、プログラム B は、データ待ち行列から項目を受け取るのに最大で 2 時間 (7200 秒) 待機するよう指定しています。プログラム A は、ライブラリー QGPL にあるデータ待ち行列 DTAQ1 に項目を送ります。プログラム A が項目を 2 時間以内に送ると、プログラム B はその項目をこのデータ待ち行列から受け取り、すぐに処理を開始します。 2 時間経過してもプログラム A が項目を送らない場合は、戻されるフィールド長が 0 であるので、プログラム B はタイムアウト条件を処理します。プログラム B は、このタイムアウト条件が発生するまで、項目を受け取り続けます。これらのプログラムは CL で記述されていますが、どのような高水準言語でも作成することができます。

データ待ち行列は、以下のコマンドで作成されます。

```
CRTDTAQ DTAQ(QGPL/DTAQ1) MAXLEN(80)
```

この例では、データ待ち行列の項目の長さは 80 バイトです。

プログラム A では、次のステートメントによってデータ待ち行列との関係が設定されます。

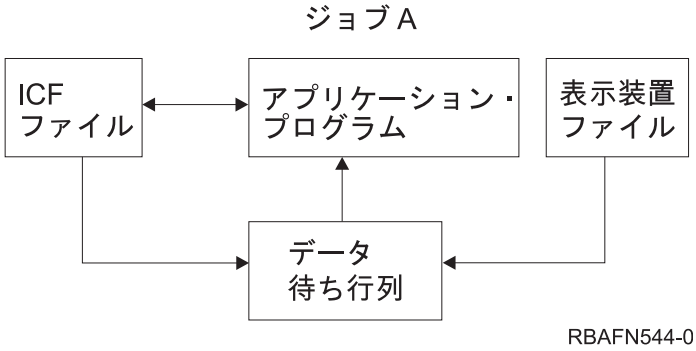
```
PGM
DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
DCL  &FIELD  *CHAR  LEN(80)
.
.(determine data to be sent to the queue)
.
CALL  QSNDDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD)
.
.
.
```

プログラム B では、次のステートメントによってデータ待ち行列との関係が設定されます。

```
        PGM
        DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
        DCL  &FIELD  *CHAR  LEN(80)
        DCL  &WAIT  *DEC  LEN(5 0)  VALUE(7200)   /* 2 hours */
        .
        .
        .
LOOP:   CALL QRCVDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD &WAIT)
        IF    (&FLDLEN *NE 0) DO   /* Entry received */
               .
               . (process data from data queue)
               .
               GOTO LOOP   /* Get next entry from data queue */
        ENDDO
        .
        . (no entries received for 2 hours; process time-out condition)
        .
```

例*:* 表示装置ファイルおよび *ICF* ファイルからの入力を待機する*:*

この例は、データ待ち行列を使用して、表示装置ファイルおよび ICF ファイルからの入力を待っているプログラムを示しています。



以下の例では、ジョブが 1 つしかないため、通常のデータ待ち行列とは使い方が異なっています。ここでのデータ待ち行列は、2 つのジョブ間ではなくジョブ内の通信オブジェクトとしての役割を持っています。

この例では、プログラムは表示装置ファイルおよび ICF ファイルからの入力を待機します。この両者からの入力を交互に待機する代わりに、プログラムはデータ待ち行列を使用して、1 つのオブジェクト (データ待ち行列) を待機するだけですみます。プログラムは QRCVDTAQ を呼び出し、表示装置ファイルおよび ICF ファイルに指定されているデータ待ち行列に項目が入れられるのを待機します。この 2 つのファイルは同じデータ待ち行列を指定しています。データがどちらかのファイルで使用可能になると、表示装置データ管理サポートまたは ICF データ管理サポートによって、2 つのタイプの項目が待ち行列に入れられます。 ICF ファイルの項目は *ICFF で始まり、表示装置ファイルの項目は *DSPF で始まります。

待ち行列に入れられる表示装置ファイルの項目または ICF ファイルの項目の長さは 80 文字であり、その中には以下に説明するフィールド属性が入っています。したがって、
CRTDSPF、CHGDSPF、OVRDSPF、CRTICFF、CHGICFF、および OVRICFF コマンドを使用して指定する待ち行列は最低 80 文字の長さが必要です。

桁 **(**およびデータ・タイプ**)**
: 説明

1 から **10** (文字)
データ待ち行列に項目を入れたファイルのタイプ。このフィールドには次の 2 つの値のどちらかが入っています。

- *ICFF、ICF ファイルの場合
- *DSPF、表示装置ファイルの場合

データ待ち行列からのデータを受け取るジョブが表示装置ファイルまたは ICF ファイルを 1 つだけオープンしている場合には、データ待ち行列からどのタイプの項目を受け取ったかを判別するのに必要なフィールドはこのフィールドだけです。

11 から **12** (2 進数)
ファイルの固有の識別コード。識別コードの値は、このファイルのオープン・フィードバック域の値と同じです。このフィールドは、同じ名前の複数のファイルがデータ待ち行列に項目を入れる場合にだけ、データ待ち行列からの項目を受け取るプログラムで使用しなければなりません。

13 から **22** (文字)
表示装置ファイルまたは ICF ファイルの名前。これは、一時変更の処理がすべて終了した後の、実際にオープンされたファイルの名前であり、このファイルのオープン・フィードバック域にあるファイル名と同じです。このフィールドは、データ待ち行列に項目を入れる表示装置ファイルまたは ICF ファイルが複数ある場合にだけ、データ待ち行列からの項目を受け取るプログラムで使用しなければなりません。

23 から **32** (文字)
該当のファイルが入っているライブラリー。これは、一時変更の処理がすべて終了した後のライブラリーの名前であり、このファイルのオープン・フィードバック域にあるライブラリー名と同じです。このフィールドは、データ待ち行列に項目を入れる表示装置ファイルまたは ICF ファイルが複数ある場合にだけ、データ待ち行列からの項目を受け取るプログラムで使用しなければなりません。

33 から **42** (文字)
一時変更の処理がすべて終了した後のプログラム装置名。この名前は、オープン・フィードバック域のプログラム装置定義リストにある名前と同じです。ファイル・タイプが *DSPF の場合、これはコマンド・キーまたは実行キーを押した表示装置の名前です。ファイル・タイプが *ICFF の場合、これはデータが使用可能なプログラム装置の名前です。データ待ち行列に項目を入れたファイルが、そのデータ待ち行列項目の受け取りに先立って送信勧誘された複数の装置またはセッションを持つ場合にだけ、データ待ち行列からの項目を受け取るプログラムで、このフィールドを使用しなければなりません。

43 から **80** (文字)
未使用

次の例は、前述のアプリケーション・プログラムが使用するコーディング・ロジックを示しています。

```
  .
  .
  .
  .
  OPEN DSPFILE ... /* Open the Display file. DTAQ parameter specified on*/
                   /* CRTDSPF, CHGDSPF, or OVRDSPF for the file.        */

  OPEN ICFFILE ... /* Open the ICF file. DTAQ parameter specified on    */
                   /* CRTICFF, CHGICFF, or OVRICFF for the file.        */

 .
 .
 DO
```

```
  WRITE DSPFILE   /* Write with Invite for the Display file           */
  WRITE ICFFILE   /* Write with Invite for the ICF file               */

  CALL QRCVDTAQ   /* Receive an entry from the data queue specified   */
                  /* on the DTAQ parameters for the files. Entries    */
                  /* are placed on the data queue when the data is    */
                  /* available from any invited device or session    */
                  /* on either file.                                  */
                  /* After the entry is received, determine which file */
                  /* has data available, read the data, process it,   */
                  /* invite the file again and return to process the  */
                  /* next entry on the data queue.                    */
  IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* Entry is from display */
    DO                                       /*  file. Since this entry*/
                                             /* does not contain the   */
                                             /* data received, the data*/
                                             /* must be read from the  */
                                             /* file before it can be  */
      READ DATA FROM DISPLAY FILE            /* processed.             */
      PROCESS INPUT DATA FROM DISPLAY FILE
      WRITE TO DISPLAY FILE                   /* Write with Invite    */
    END
  ELSE                                       /* Entry is from ICF     */
                                             /*  file. Since this entry*/
                                             /* does not contain the   */
                                             /* data received, the data*/
                                             /* must be read from the  */
                                             /* file before it can be  */
                                             /* processed.             */
      READ DATA FROM ICF  FILE
      PROCESS INPUT DATA FROM ICF FILE
      WRITE TO ICF FILE                       /* Write with Invite     */
  LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
.
.
.
 END
```

関連タスク:

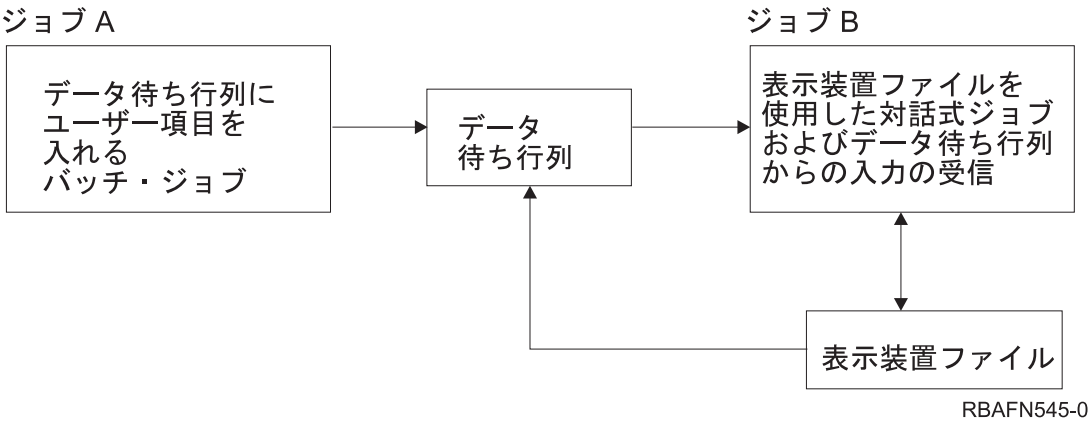
292 ページの『データ待ち行列の使用』

データ待ち行列は、ユーザーが作成することのできるシステム・オブジェクトの 1 つのタイプであり、高水準言語 (HLL) プロシージャーまたはプログラムがこの待ち行列にデータを送り、別の HLL プロシージャーまたはプログラムがそこからそのデータを受け取ることができます。

例*:* 表示装置ファイルおよびデータ待ち行列からの入力を待機する*:*

この例は、表示装置ファイルからの入力を待っている、または別のジョブのデータ待ち行列で入力を待っている、ジョブのプログラムを示しています。

ジョブ B にあるプログラムが、使用している表示装置ファイルからの入力、およびジョブ A からのデータ待ち行列に達する入力を待機しています。ここで、最初に表示装置ファイルを待ってからデータ待ち行列を待つ代わりに、プログラムは 1 つのオブジェクト、つまりデータ待ち行列を待機しています。



プログラムは QRCVDTAQ を呼び出し、表示装置ファイルに指定されているデータ待ち行列に項目が入れられるのを待ちます。また、ジョブ A も同じデータ待ち行列に項目を入れます。この待ち行列には 2 つのタイプの項目、すなわち表示装置ファイルの項目またはユーザー定義の項目が入ることになります。表示装置データ管理は、データが表示装置ファイルから使用できる場合に、その表示装置ファイルの項目をデータ待ち行列に入れます。ジョブ A は、ユーザー定義項目をデータ待ち行列に入れます。

表示装置ファイル項目の構成については、前の例で説明しています。

ジョブ A によって待ち行列に入れられる項目の構成は、アプリケーション・プログラマーが定義します。

次の例は、ジョブ B のアプリケーション・プログラムのコーディング・ロジックを示しています。

```
  .
  .
  .
  .
 OPEN DSPFILE ... /* Open the Display file. DTAQ parameter specified on*/
                  /* CRTDSPF, CHGDSPF, or OVRDSPF for the file.        */


.
.
DO
  WRITE DSPFILE   /* Write with Invite for the Display file           */

  CALL QRCVDTAQ   /* Receive an entry from the data queue specified    */
                  /* on the DTAQ parameter for the file. Entries       */
                  /* are placed on the data queue either by Job A or   */
                  /* by display data management when data is           */
                  /* available from any invited device on the display  */
                  /* file.                                             */
                  /* After the entry is received, determine what type  */
                  /* of entry it is, process it, and return to receive */
                  /* the next entry on the data queue.                 */
  IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* Entry is from display */
    DO                                       /*  file. Since this entry*/
                                             /* does not contain the   */
                                             /* data received, the data*/
                                             /* must be read from the  */
                                             /* file before it can be  */
     READ DATA FROM DISPLAY FILE             /* processed.             */
     PROCESS INPUT DATA FROM DISPLAY FILE
     WRITE TO DISPLAY FILE                    /* Write with Invite    */
    END
  ELSE                                       /* Entry is from Job A.  */
                                             /* This entry contains   */
```

```
                                          /* the data from Job A,  */
                                          /* so no read is required*/
                                          /* before processing the */
                                          /* data.                 */
     PROCESS DATA QUEUE ENTRY FROM JOB A
  LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
 .
 .
 .
 END
```

関連タスク:

292 ページの『データ待ち行列の使用』

データ待ち行列は、ユーザーが作成することのできるシステム・オブジェクトの 1 つのタイプであり、高水準言語 (HLL) プロシージャーまたはプログラムがこの待ち行列にデータを送り、別の HLL プロシージャーまたはプログラムがそこからそのデータを受け取ることができます。

出力待ち行列に関連付けられているデータ待ち行列の作成:

データ待ち行列と出力待ち行列を関連付けることで、出力待ち行列上のスプール・ファイルが待機 (RDY) 状態になると、1 つの項目がデータ待ち行列に送られるようにすることができます。

送られるデータ待ち行列項目のレイアウトまたは内容を表示するには、レコード・タイプ 01 データ待ち行列項目の形式を参照してください。

データ待ち行列と出力待ち行列を関連付けるには、**データ待ち行列作成 (CRTDTAQ)** コマンドを使用してデータ待ち行列を作成します。最大メッセージ長 (MAXLEN) パラメーターの値は最低でも 128 を指定してください。順序 (SEQ) パラメーターの値は *FIFO または *LIFO です。その後、**出力待ち行列作成 (CRTOUTQ)** コマンドまたは **出力待ち行列変更 (CHGOUTQ)** コマンドの DTAQ キーワードで、**CRTDTAQ** コマンドで作成するデータ待ち行列を指定します。

関連情報:

データ待ち行列作成 (CRTDTAQ) コマンド

データ待ち行列サポート

ジョブに関連付けられているデータ待ち行列の作成:

データ待ち行列を出口点 QIBM_QWT_JOBNOTIFY に関連付けて、特定のジョブ遷移についてデータ待ち行列に項目を入れることができます。

ジョブがジョブ待ち行列に入れられたとき、開始したとき、または終了したときに、項目をデータ待ち行列に入れることができます。データ待ち行列の作成方法、およびデータ待ち行列に入れる情報の形式について詳しくは、 ジョブ通知出口点 (QIBM_QWT_JOBNOTIFY) を参照してください。

データ域の使用

データ域は、システムで実行中の任意のジョブからアクセスできるデータを入れるために使用されるオブジェクトです。

データ域は、プロシージャーまたはファイルの有無に関係なく、限定されたサイズの情報を保管させておきたい場合に使用することができます。データ域の主な用法には次のものがあります。

- 1 つのジョブ内での情報を受け渡しするための区域 (多くの場合、各ジョブの QTEMP ライブラリーの中に) を設けるため。
- 次のような目的で、ジョブ内での参照の制御のために容易かつ頻繁に変更されるフィールドを設けるため。

  - 次に割り当てる順序番号を指示するため。
  - 次の検査番号を指示するため。
  - 次に使用する保管/復元媒体のボリュームを指示するため。
- 税率や配布リストなど、複数のジョブで使用する定数フィールドを設けるため。
- データ域を必要とする大規模な処理にだけアクセスを制限するため。データ域を 1 人のユーザーにだけロックすることにより、他のユーザーが同時に処理を行えないようにすることができます。

ローカル・データ域またはグループ・データ域以外のデータ域を作成する場合には、 **データ域作成 (CRTDTAARA)** コマンドを使用します。これを行うことにより、特定のライブラリーに独立したオブジェクトを作成して、特定の値に初期設定することができます。 CL プロシージャーまたはプログラムでその値を使用する場合には、 **データ域検索 (RTVDTAARA)** コマンドを使用して、その値をプロシージャーまたはプログラムの変数に入れます。その値を CL プロシージャーまたはプログラムの中で変更し、その新しい値をデータ域に戻したい場合には、 **データ域変更 (CHGDTAARA)** コマンドを使用します。

現行の値を表示したい場合には、**データ域表示 (DSPDTAARA)** コマンドを使用します。また、**データ域削除 (DLTDTAARA)** コマンドを使用すれば、データ域を削除することができます。

データ域をジャーナルすることができます。これにより、異常な IPL または破損が起こったときにオブジェクトが変更アクションの途中であった場合であっても、オブジェクトを一貫性のある状態に回復することができます。また、ジャーナル処理により、リモート・システムのデータ域ジャーナルの複製も行えます (例えば、リモート・ジャーナルを使用する)。これにより、システムは、類似の環境にアクションを複製して、アプリケーション作業を複製することができます。

関連情報:

データ域作成 (CRTDTAARA) コマンド

データ域検索 (RTVDTAARA) コマンド

データ域変更 (CHGDTAARA) コマンド

データ域表示 (DSPDTAARA) コマンド

データ域削除 (DLTDTAARA) コマンド

ジャーナル管理

ローカル・データ域:

ローカル・データ域は、自動開始ジョブ、システム読み取りプログラムにより開始されるジョブ、サブシステム監視ジョブなど、システム内のすべてのジョブごとに作成されます。

ローカル・データ域はシステムが作成するもので、その初期値は全桁ブランクであり、長さは 1024、タイプは *CHAR です。 SBMJOB コマンドを用いてジョブを投入した場合には、投入元のジョブのローカル・データ域の値が、投入されるジョブのローカル・データ域にコピーされます。ユーザーは、CHGDTAARA、RTVDTAARA、および DSPDTAARA コマンドの DTAARA キーワードに *LDA を指定するか、またはサブストリング組み込み関数 (%SST) に *LDA を指定することによって、ジョブのローカル・データ域を参照することができます。

ローカル・データ域については次のような条件があります。

- ローカル・データ域は他のジョブから参照することはできません。
- ユーザーがローカル・データ域を作成、削除、または割り振りすることはできません。
- ローカル・データ域を入れるためのライブラリーはありません。
- 2 次スレッド内のローカル・データ域に変更を加えることはできません。

- ILE CL コンパイラーは、初期スレッドで実行されているプロシージャーがローカル・データ域に変更を加えている間は、2 次スレッドで実行されているプロシージャーがローカル・データ域にアクセスできないようにするためのコードを生成します。

ローカル・データ域の内容は経路指定ステップの境界にまたがって存在します。したがって、**ジョブ転送 (TFRJOB)**、**バッチ・ジョブ転送 (TFRBCHJOB)**、 **ジョブ経路再指定 (RRTJOB)**、または**戻り (RETURN)** コマンドによって、ローカル・データ域の内容が影響を受けることはありません。

ローカル・データ域は次の目的で使用することができます。

- パラメーター・リストを使用せずにプロシージャーまたはプログラムに情報を渡すため。
- 情報をローカル・データ域に入れ、ジョブを投入することにより、その投入されたジョブに情報を渡すため。投入されたジョブでは、データにアクセスすることができます。
- CL プロシージャーまたはプログラムから他のタイプのデータ域へのアクセスよりも、パフォーマンスを向上させるため。
- データ域を作成および削除するためのユーザーの作業を必要とせずに、情報を保管するため。

ローカル・データ域は多くの高水準言語で使用することができます。 SBMxxxJOB コマンドおよび STRxxxRDR コマンドにより、ローカル・データ域がブランクに初期設定されたジョブが開始されます。投入元のジョブのローカル・データ域を新しいジョブに渡すことができるのは、SBMJOB コマンドだけです。

関連情報:

データ域作成 (CRTDTAARA) コマンド

データ域検索 (RTVDTAARA) コマンド

データ域変更 (CHGDTAARA) コマンド

データ域表示 (DSPDTAARA) コマンド

データ域削除 (DLTDTAARA) コマンド

グループ・データ域:

対話式ジョブが (**グループ属性変更 (CHGGRPA)** コマンドの使用により) グループ・ジョブになる時点で、システムはグループ・データ域を作成します。

グループ・データ域は 1 つのグループについて 1 つしかありません。グループ内の最後のジョブが、ENDJOB、SIGNOFF、または ENDGRPJOB コマンド、あるいは異常終了によって終了した場合、または GRPJOB(*NONE) が指定された CHGGRPA コマンドの使用によりジョブがグループ・ジョブの一部でなくなった場合には、グループ・データ域は削除されます。

グループ・データ域は、全桁ブランクに初期設定され、長さは 512、タイプは *CHAR です。ユーザーは、CHGDTAARA、RTVDTAARA、および DSPDTAARA の各コマンドの DTAARA パラメーターに *GDA を指定することによって、グループ・ジョブの中からグループ・データ域を使用することができます。グループ・データ域は、該当グループ内のすべてのジョブからアクセスすることができます。

グループ・データ域については次のような条件があります。

- サブストリング組み込み関数 (%SUBSTRING または %SST) で、文字変数としてグループ・データ域を指定することはできません。 (ただし、サブストリング関数が使用する 512 バイトの文字変数を、グループ・データ域との間で受け渡しすることはできます。)
- グループ・データ域は、該当のグループに属していないジョブで参照することはできません。
- ユーザーがグループ・データ域を作成、削除、または割り振りすることはできません。

- グループ・データ域のためのライブラリーはありません。

グループ・ジョブへの移行 (TFRGRPJOB) コマンドを使用しても、グループ・データ域の内容は変更されません。

グループ・データ域は、他のデータ域を使用する場合と同じように使用することができます。同一グループの各グループ・ジョブ相互間での情報の受け渡しのために使用することができます。例えば、**グループ属性変更 (CHGGRPA)** コマンドを出した後で、次のようなコマンドを使用してグループ・データ域の値をセットすることができます。

```
CHGDTAARA  DTAARA(*GDA)  VALUE('January1988')
```

このコマンドは、プログラムから実行することも、ワークステーションのユーザーが発行することもできます。

グループ内の他の CL プロシージャーまたはプログラムは、次のような CL コマンドによってそのグループ・データ域の値を検索することができます。

```
RTVDTAARA  DTAARA(*GDA)  RTNVAR(&GRPARA)
```

このコマンドは、グループ・データ域の値 (January1988) を CL 変数 &GRPARA に入れます。

関連情報:

CL コマンド検索プログラム

グループ属性変更 (CHGGRPA) コマンド

データ域作成 (CRTDTAARA) コマンド

データ域検索 (RTVDTAARA) コマンド

データ域変更 (CHGDTAARA) コマンド

データ域表示 (DSPDTAARA) コマンド

データ域削除 (DLTDTAARA) コマンド

プログラム初期設定パラメーター・データ域**:**

ジョブの開始の時点で、事前開始ジョブごとに、プログラム初期設定パラメーター (PIP) データ域 (PDA) が作成されます。

PDA のオブジェクト・サブタイプは通常のデータ域とは異なります。 PDA は、*PDA という特殊値名によってだけ参照できます。 PDA のサイズは 2000 バイトですが、そこに含まれるパラメーターの数には制限はありません。

データ域パラメーターとして特殊値 *PDA を使用できるのは、RTVDTAARA、 CHGDTAARA、および DSPDTAARA の各 CL コマンド、および RTVDTAARA、 CHGDTAARA の 2 つのマクロ命令です。

関連情報:

データ域作成 (CRTDTAARA) コマンド

データ域検索 (RTVDTAARA) コマンド

データ域変更 (CHGDTAARA) コマンド

データ域表示 (DSPDTAARA) コマンド

データ域削除 (DLTDTAARA) コマンド

リモート・データ域:

リモート・データ域は、リモート・システムのデータ域です。

分散データ管理機能 (DDM) を使用して、リモート・データ域にアクセスすることができます。あるシステムに存在するアプリケーション・プログラムがリモートのシステムに存在するデータを検索する場合、そのアプリケーション・プログラムを変更または再コンパイルする必要はありません。正しいデータ域にアクセスしたかどうかを確かめるには、次のいずれかの作業を行う必要がある場合があります。

- 標準のデータ域を削除し、最初の標準のデータ域と同じ名前を持つ DDM データ域を作成する。
- 標準のデータ域の名前を変更する。

次のことを行うことにより DDM データ域を作成できます。

```
CRTDTAARA DTAARA(LOCALLIB/DDMDTAARA) TYPE(*DDM)
   RMTDTAARA(REMOTELIB/RMTDTAARA) RMTLOCNAME(SYSTEMB)
   TEXT('DDM data area to access data area on SYSTEMB')
```

CL プログラムでリモート・システムのデータ域からの値を使用するには、 **データ域検索 (RTVDTAARA)** コマンドを使用します。プログラムの変数に現在の値を入れるには、DDM データ域の名前を指定します。その値を CL プログラムの中で変更し、その新しい値をリモートのデータ域に戻したい場合には、 **データ域変更 (CHGDTAARA)** コマンドを使用し、同じ DDM データ域を指定します。

データ域表示 (DSPDTAARA) コマンドの使用時に DDM データ域の名前を指定すると、リモートのデータ域の値ではなく、DDM データ域の値が表示されます。また、**データ域削除 (DLTDTAARA)** コマンドを使用すれば、DDM データ域を削除することができます。

関連情報:

分散データベース・プログラミング

データ域作成 (CRTDTAARA) コマンド

データ域検索 (RTVDTAARA) コマンド

データ域変更 (CHGDTAARA) コマンド

データ域表示 (DSPDTAARA) コマンド

データ域削除 (DLTDTAARA) コマンド

データ域の作成:

プログラム変数とは異なり、データ域はオブジェクトなので、プログラムまたはジョブで使用するためには、まずデータ域を作成しておかなければなりません。

データ域は次のように作成することができます。

- 2000 文字以下の文字ストリングとして。
- 10 進数として。この 10 進数値の属性は、CL プログラムまたはプロシージャーでだけ使用されるのか、それとも他の高水準言語プログラムでも使用されるのかによって異なります。 CL プロシージャーおよびプログラムの場合には、データ域は、小数点の左側に最高 15 文字、小数点の右側 (小数部分) に最高 9 文字をとることができますが、両方の合計桁数は最高 15 文字までです。その他の言語の場合には、データ域は、小数点の左側に最高 15 文字、小数点の右側 (小数部分) に最高 9 文字をとることができ、両方の合計桁数として 24 文字までが可能です。
- 論理値 '0' または '1'。 '0' はオフ、偽、または NO を意味し、'1' はオン、真、または YES を意味します。

データ域の作成の時点で、そのデータ域の初期値を指定することもできます。初期値を指定しなかった場合には、次の値がとられます。

- 10 進数の場合は 0。
- 文字の場合はブランク。
- 論理値の場合は '0'。

データ域を作成する場合には、**データ域作成 (CRTDTAARA)** コマンドを使用します。次の例では、あるプログラムから別のプログラムに得意先番号を渡すためのデータ域が作成されます。

```
CRTDTAARA  DTAARA(CUST) TYPE(*DEC) +
           LEN(5 0) TEXT('Next customer number')
```

関連情報:

データ域作成 (CRTDTAARA) コマンド

データ域のロックと割り振り:

データ域のロックと割り振りは、データ域が複数のジョブから一度にアクセスされないことを保証するのに役立ちます。

データ域変更 (CHGDTAARA) コマンドの場合、そのコマンドの処理の過程でデータ域に対して *SHRUPD (更新共用) ロックが適用されます。 **データ域検索 (RTVDTAARA)** コマンドおよび**データ域表示 (DSPDTAARA)** コマンドでは、コマンド処理の過程でデータ域に対して *SHRRD (読み取り共用) ロックが使用されます。あるデータ域に対して複数の操作を行う場合は、操作が完了するまで他のユーザーがそのデータ域にアクセスできないようにしたい場合があります。これは**オブジェクト割り振り (ALCOBJ)** コマンドを用いて行うことができます。例えば、同時に実行されるいくつかのジョブにより読み取られ、更新される値がデータ域に含まれている場合には、ALCOBJ コマンドを使用することにより、読み取り操作および更新操作の間その値を保護することができます。

他の (CL 以外の) 言語でのデータ域の取り扱いについては、使用する高水準言語 (HLL) の解説書を参照してください。

関連概念:

463 ページの『オブジェクトおよびライブラリー』
オブジェクトおよびライブラリーに固有のタスクおよび概念には、オブジェクトに対して実行される機能、ライブラリーの作成、およびオブジェクト権限の指定が含まれます。

関連情報:

データ域作成 (CRTDTAARA) コマンド

データ域検索 (RTVDTAARA) コマンド

データ域表示 (DSPDTAARA) コマンド

データ域の表示:

データ域表示 (DSPDTAARA) コマンドを使用して、属性 (名前、ライブラリー、タイプ、長さ、データ域のテキスト記述) およびデータ域の値を表示することができます。この表示では、先行ゼロを取り除いた 24 文字形式が使用されます。

関連情報:

データ域表示 (DSPDTAARA) コマンド

データ域の変更:

データ域の値を変更するには、**データ域変更 (CHGDTAARA)** コマンドを使用します。

CHGDTAARA コマンドを使用すれば、指定したデータ域の値の全体または一部を変更することができます。データ域のその他の属性は変更されません。新しい値には、定数または CL 変数を使用することができます。このコマンドを CL プログラムまたはプロシージャーに入れる場合は、プログラムまたはプロシージャーの作成時には、該当のデータ域は存在しなくてもかまいません。

関連情報:

データ域変更 (CHGDTAARA) コマンド

データ域の検索:

データ域検索 (RTVDTAARA) コマンドを使用して、データ域を検索して、変数にコピーすることができます。

データ域検索 (RTVDTAARA) コマンドを使用すれば、指定したデータ域の全体または一部を検索し、それを CL 変数にコピーすることができます。コンパイルの時点では該当のデータ域は存在している必要はなく、また、CL 変数はデータ域と同じ名前である必要はありません。このコマンドは、指定したデータ域の内容を検索するためのものであり、その内容を変更することはできないので注意してください。

関連情報:

データ域検索 (RTVDTAARA) コマンド

例: データ域の検索:

以下の例は、データ域のさまざまな検索方法を示しています。

例: データ域 ***ORDINFO*** の検索:

この例では、データ域を使用して受注ファイルの状況を追跡しています。

受注ファイルの状況の経過を保管しておくための ORDINFO という名前のデータ域を使用しているものとします。このデータ域は次のように設計されています。

- 1 文字目には、O (オープン)、P (処理中)、または C (完了) が入ります。
- 2 文字目には、I (在庫あり) または O (在庫切れ) が入ります。
- 3 から 5 文字目には、受注担当者の名前の頭文字が入ります。

上記の各フィールドは、プログラムまたはプロシージャーの中で次のように宣言することができます。

```
DCL VAR(&ORDSTAT) TYPE(*CHAR) LEN(1)
DCL VAR(&STOCKC) TYPE(*CHAR) LEN(1)
DCL VAR(&CLERK) TYPE(*CHAR) LEN(3)
```

受注状況を検索して &ORDSTAT に入れるには、次のように入力します。

```
RTVDTAARA DTAARA(ORDINFO (1  1))  RTNVAR(&ORDSTAT)
```

在庫状況を検索して &STOCK に入れるには、次のように入力します。

```
RTVDTAARA DTAARA(ORDINFO (2  1)) RTNVAR(&STOCKC)
```

担当者の頭文字を検索して &CLERK に入れるには、次のように入力します。

```
RTVDTAARA DTAARA(ORDINFO (3  3)) RTNVAR(&CLERK)
```

上記のどの RTVDTAARA コマンドの場合にも、データ域へのアクセスが必要になります。検索するサブフィールドの数が多い場合には、データ域全体を検索して変数に入れた上で、サブストリング組み込み関数を用いてサブフィールドを取り出す方が効率がよくなります。

例*:* データ域 ***DA1*** の検索*:*

この例は、データ域を検索し、それを変数にコピーする方法を示しています。

以下に示した**データ域検索 (RTVDTAARA)** コマンドの例では、5 文字のデータ域の指定された内容が 3 文字変数に入れられます。この例では次のことが行われます。

- DA1 という名前の 5 文字のデータ域が (ライブラリー MYLIB に) 作成され、その初期値として 'ABCDE' が入れられます。
- &CLVAR1 という名前の 3 文字の文字変数が宣言されます。
- DA1 の最後の 3 文字の内容が、&CLVAR1 にコピーされます。

上記のことを行うためには、次のようなコマンドを入力します。

```
CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(5) VALUE(ABCDE)
.
.
.
DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(3)
RTVDTAARA DTAARA(MYLIB/DA1 (3 3)) RTNVAR(&CLVAR1)
```

この結果、&CLVAR1 に 'CDE' が入ります。

関連情報:

データ域検索 (RTVDTAARA) コマンド

例*:* データ域 ***DA2*** の検索*:*

この例は、10 進データ域を検索し、それを 10 進変数にコピーする方法を示しています。

以下に示した**データ域検索 (RTVDTAARA)** コマンドの例では、5 桁の 10 進データ域の内容が 5 桁の 10 進変数に入れられます。この例では次のことが行われます。

- DA2 という名前で、小数部分が 2 桁、全体の長さが 5 桁のデータ域がライブラリー MYLIB に作成され、その初期値として 12.39 が入れられます。
- &CLVAR2 という名前で、小数部分の桁数が 1 文字、全体の桁数が 5 桁の変数が宣言されます。
- DA2 の内容が &CLVAR2 にコピーされます。

上記のことを行うためには、次のようなコマンドを入力します。

```
CRTDTAARA DTAARA(MYLIB/DA2) TYPE(*DEC) LEN(5 2) VALUE(12.39)
.
.
.
DCL  VAR(&CLVAR2) TYPE(*DEC) LEN(5 1)
RTVDTAARA DTAARA(MYLIB/DA2) RTNVAR(&CLVAR2)
```

この結果、&CLVAR2 に 0012.3 が含まれます (小数部は切り捨てられています)。

関連情報:

データ域検索 (RTVDTAARA) コマンド

例**:** データ域の変更および検索**:**

この例は、データ域を変更および検索する方法を示しています。

以下に、文字サブストリングを操作するための **データ域変更 (CHGDTAARA)** コマンドおよび**データ域検索 (RTVDTAARA)** コマンドの使用例を示します。

この例では次のことが行われます。

- DA1 という名前の 10 文字のデータが (MYLIB ライブラリーに) 作成され、その初期値として ABCD5678IJ が入れられます。
- &CLVAR1 という名前の 5 文字の文字変数が宣言されます。
- データ域 DA1 の内容 (5 文字目から始まって 4 文字分) が EFG という値 (G の後にブランクを 1 つ埋め込んだもの) に変更されます。
- データ域 DA1 の内容 (5 文字目から始まって 5 文字分) が検索され、CL 変数 &CLVAR1 に入れられます。

上記のことを行うためには、次のようなコマンドを入力します。

```
DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(5)
.
CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(10) +
     VALUE('ABCD5678IJ')
.
.
.
CHGDTAARA DTAARA((MYLIB/DA1) (5 4)) VALUE('EFG')
RTVDTAARA DTAARA((MYLIB/DA1) (5 5)) RTNVAR(&CLVAR1)
```

変数 &CLVAR1 に 'EFG I' が含まれるようになりました。

関連情報:

データ域検索 (RTVDTAARA) コマンド

データ域変更 (CHGDTAARA) コマンド

CL コマンドの定義および文書化

この情報は、独自の CL コマンドを定義して、それらに対する資料を用意するのに役立ちます。

CL コマンドの定義

CL コマンドを使用することによって、ユーザーは、広範囲にわたる機能をシステムに要求できます。 IBM 提供のコマンドを使用して、コマンド・パラメーター用のデフォルト値を変更し、ユーザー独自のコマンドを定義することができます。

CL コマンドは、システムに機能を実行するように要求するステートメントです。 コマンドを入力すると、その機能を実行するプログラムが入力開始します。 CL コマンドを使用することによって、ユーザーは、広範囲にわたる機能をシステムに要求できます。 CL コマンドは、 IBM 提供の形のままで使用するだけでなく、そのデフォルト値を変更したり、ユーザー独自のコマンドを定義して使用することもできます。

独自の CL コマンドを定義したり作成したりするときは、その資料も提供できます。

関連概念:

197 ページの『予約されているパラメーター値または数値パラメーターの値を置換する変数』
文字変数は、複数のコマンドに対して使用し、コマンド・パラメーターの値を示すことができます

関連タスク:

196 ページの『リストまたは修飾名を指定するために使用する変数』
変数を使用してリストまたは修飾名を指定することができます。

385 ページの『CL コマンドの文書化』
独自の CL コマンドを定義および作成する場合は、コマンドを説明するオンライン・コマンド・ヘルプを作成することもできます。

関連資料:

6 ページの『CL コマンドおよびキーワードで使用される省略形』
IBM i オペレーティング・システムおよびそのオペレーティング・システム上で実行されるその他のライセンス・プログラムの一部である大部分の CL コマンド名は、一貫した命名形式に従います。

CL コマンド定義ステートメント:

コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。定義済みのコマンドは、プログラムを呼び出して、ある機能を実行します。ユーザーは、コマンド定義ステートメントを使用してコマンドを定義できます。定義するコマンドには次のものを入れることができます。

- キーワード表記パラメーター。プログラムにデータを渡します。
- 省略されたパラメーターに代わるデフォルト値。
- パラメーターの妥当性検査。機能を実行するプログラムに対して正しい値が渡されるようにします。
- プロンプト・テキスト。対話式ユーザーに対して操作上の指示を与えます。

システム上のコマンドは、コマンド定義オブジェクトおよびコマンド処理プログラム (CPP) をそれぞれ 1 つ持っています。

CPP は、該当のコマンドが入力された時に呼び出されるプログラムです。コマンドの入力時の妥当性検査はシステムが行うため、CPP は必ずしも渡されたパラメーターを検査する必要はありません。

コマンド定義機能は次の目的で使用することができます。

- CL コマンドのユーザーに対するインターフェースの整合性を保持し、システムのユーザーが必要とする固有のコマンドを作成する。
- システムのユーザーの必要性に応じて、CL コマンドの代替バージョンを定義する。この機能によって、例えばパラメーターのデフォルト値の変更や、パラメーターをいくつか省いてコマンドを簡略化することもできます。また、パラメーターに定数値を定義することもできます。ただし、IBM 提供のコマンドは変更しないでください。

表 24. CL コマンドを定義するステートメント

ステートメント・タイプ	ステートメント名	関連コマンド	説明
コマンド	CMD	コマンド定義 (CMD) コマンド	作成されるコマンドのプロンプト・テキストを指定します。CMD ステートメントは、コマンド作成 (CRTCMD) コマンドにより参照されるソース・ファイルの任意の位置に入れることができます。作成するコマンドにプロンプト・テキストを指定しない場合でも、ソース・ファイルに CMD ステートメントを 1 つのみ使用できます。 CRTCMD コマンドの複数の作成オプション・パラメーターを、CMD ステートメントに指定することもできます。
パラメーター	PARM	パラメーター定義 (PARM) コマンド	作成されるコマンドのパラメーターを定義します。パラメーターは、値をコマンド処理プログラム (CPP) に渡す手段です。定義中のコマンドに指定する各パラメーターごとに、必ず 1 つの PARM ステートメントを使用しなければなりません。
要素	ELEM	要素定義 (ELEM) コマンド	このコマンド上の混合リスト (リスト要素) パラメーターの要素を定義するために使用されます。リスト・パラメーターは、複数の値を受け入れるパラメーターで、この複数の値は、1 つのキーワードによって連続の値としてまとめて渡されます。
修飾子	QUAL	修飾名定義 (QUAL) コマンド	修飾名の一部を記述します。 PARM ステートメントまたは ELEM ステートメントで定義されたパラメーターまたはリストの要素として指定できる値であれば、その名前は、名前の修飾に使用する各修飾子ごとに QUAL ステートメントを使用することにより、修飾名に変更することができます。
依存関係	DEP	従属定義 (DEP) コマンド	パラメーターとパラメーター値との間の関係を検査する必要がある場合に、両者の関係を定義するために用いられます。これには、パラメーターと、それに関連する特定の値との関係を検査する場合 (関係の検査) と、パラメーターが存在しなければならないことを検査する場合 (指定の検査) とがあります。
プロンプト制御	PMTCTL	プロンプト制御定義 (PMTCTL) コマンド	PARM ステートメントで PMTCTL ステートメントを参照しているパラメーターのためにプロンプトを出すかどうかを決める条件を指定します。

関連概念:

41 ページの『CL コマンドの区切り文字』
コマンド区切り文字は、コマンドの中で、文字のグループの始まりと終わりを示す特殊文字またはスペースです。

127 ページの『単純オブジェクト名および修飾オブジェクト名』
ライブラリー内の特定のオブジェクトの名前は、単純名としても修飾名としても指定できます。

45 ページの『CL コマンド・コーディング規則』
ここに記載するコマンド・コーディング規則についての一般情報の要約は、CL コマンドを正しくコーディングするために役に立ちます。

45 ページの『CL コマンド定義』
コマンド定義機能を使用することにより、ユーザーはアプリケーションの特殊な要件に対応する新たな制御言語コマンドを作成することができます。ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。

117 ページの『コマンド内での命名』
名前の指定に使用できる文字は、制御言語 (CL) で指定する名前のタイプによって決まります。

316 ページの『コマンド定義オブジェクト』
コマンド定義オブジェクトはシステム・プログラムが検査するオブジェクトで、そのコマンドが有効であること、また正しいパラメーターが指定されていることを確認できます。

関連タスク:

352 ページの『CL コマンド・パラメーターのプロンプト制御の指定』
プロンプト制御の指定を用いることにより、プロンプトの過程でコマンドのどのパラメーターを表示するかを制御できます。

385 ページの『CL コマンドの文書化』
独自の CL コマンドを定義および作成する場合は、コマンドを説明するオンライン・コマンド・ヘルプを作成することもできます。

関連情報:

統合ファイル・システム

ユーザー定義の *CL* コマンドの作成*:*

ソース・ファイルにコマンド定義ステートメントを入力し、そのソース・ファイルを入力として**コマンド作成 (CRTCMD)** コマンドを実行することによって、CL コマンドを定義することができます。

各コマンドのコマンド定義 は、コマンド定義ステートメント を 1 つ以上含みます。

ソース・ファイルには、CMD (コマンド) ステートメントが必ず 1 つのみ含まれていなければなりません。作成するコマンドに指定する各パラメーターについて、それぞれ PARM (パラメーター) ステートメントが必要です。複合パラメーターは、ELEM (要素) ステートメントおよび QUAL (修飾子) ステートメントを使用して、パラメーターの一部を定義することがあります。特定のキーワード関係を検査する必要がある場合には、DEP (従属) ステートメントを用いてその関係を定義します。DEP ステートメントは、既に定義されているパラメーターのみを参照できます。これらのステートメントは、任意の順序で指定できます。PMTCTL (プロンプト制御) ステートメントは、コマンド・パラメーターのプロンプトを選択して出す場合に使用します。

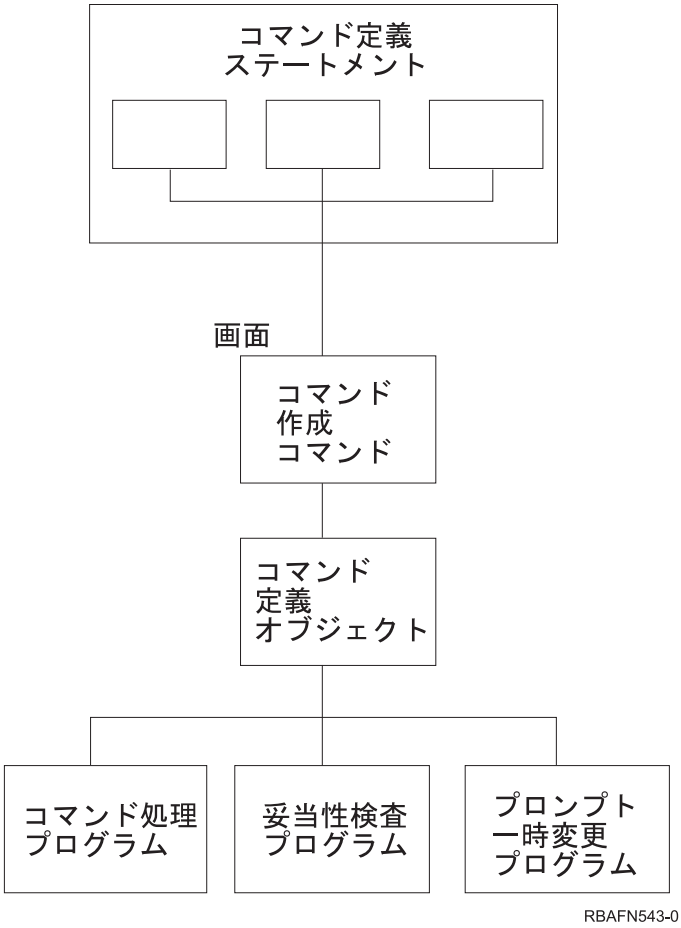
ソース・ファイルの 1 つのソース・メンバーによって定義できるコマンドは 1 つのみです。1 つのソース・ファイル・メンバー内のコマンド定義ステートメントから、コマンド定義オブジェクトを作成するには、**コマンド作成 (CRTCMD)** コマンドを実行します。他のユーザーには、**オブジェクト権限認可 (GRTOBJAUT)** コマンド、または**オブジェクト権限編集 (EDTOBJAUT)** コマンドにより、新しいコマンドの使用を許可できます。

| コマンド定義ステートメントは、以前のようにデータベース・ソース・ファイルに入力するか、または IFS
| ストリーム・ファイルに入力することができます。 IFS ストリーム・ファイルを使用するには、**CRTCMD** コ
| マンドのパラメーター SRCSTM にソース・ストリーム・ファイル名を指定する必要があります。

CL コマンド定義のプロセス*:*

独自の CL コマンドを定義するプロセス全体には、さまざまな段階があり、オプションの段階もあります。

以下の図は、コマンドを作成するプロセスを示しています。図の後に、プロセスの各段階について説明します。



ユーザー独自の妥当性検査プログラムとプロンプト一時変更プログラムを作成するかどうかは自由です。

コマンド定義ステートメントは、コマンドによる入力を促すプロンプト、その入力の妥当性検査、およびそのコマンドの実行時点で呼び出されるプログラムに渡す値の定義に必要な情報を指定します。

コマンド作成コマンド*:*

コマンド作成 (CRTCMD) コマンドは、コマンド定義ステートメントを処理してコマンド定義オブジェクトを作成します。

コマンド作成 (CRTCMD) コマンドは、対話式でもバッチ・ジョブでも実行できます。

関連情報:

コマンド作成 (CRTCMD) コマンド

コマンド定義オブジェクト*:*

コマンド定義オブジェクトはシステム・プログラムが検査するオブジェクトで、そのコマンドが有効であること、また正しいパラメーターが指定されていることを確認できます。

コマンド定義オブジェクトはそのコマンドを定義するもので、以下によって構成されています。

- コマンド名

- コマンド処理プログラム (CPP)
- そのコマンドで使用できるパラメーターおよび値
- コマンドが入力された時点でその妥当性を検査するためにシステムが使用する妥当性検査情報
- そのコマンドに関するプロンプトが要求された場合に表示するプロンプト・テキスト
- オンライン・ヘルプ情報

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

CL コマンドの妥当性検査*:*

システムは、コマンドの妥当性検査を行います。妥当性検査プログラムは必須ではありませんが、必要ならユーザー固有のプログラムを作成することもできます。

システムで行う妥当性検査では、以下のことが検査されます。

- 必須パラメーターに値が入力されていること。
- 各パラメーター値がデータ・タイプと長さの要件を満たしていること。
- 各パラメーター値が、コマンド定義で必要に応じて指定される以下のような要件を満たしていること。
  - 有効な値のリスト
  - 値の範囲
  - 値の関係比較
- 矛盾したパラメーターが入力されていないこと。

システムによる妥当性検査は、以下の場合に行われます。

- コマンドがディスプレイ装置から対話式で入力された場合。
- コマンドがスプーリングを使用して、バッチ入力ストリームから入力された場合。
- コマンドが、原始ステートメント入力ユーティリティー (SEU) によってデータベースに入力された場合。
- コマンドが高水準言語 (HLL) からの呼び出しによって、QCMDEXC、QCMDCHK、または QCAPCMD に渡された場合。
- CL モジュールまたはオリジナル・プログラム・モデル (OPM) プログラムが作成された場合。
- コマンドが CL プロシージャーか REXX プロシージャーで実行された場合。
- コマンドが C 言語システム機能を使用して実行された場合。

システムが実行する以上の妥当性検査が必要な場合、ユーザーは妥当性検査プログラム と呼ばれるプログラムを作成するか、またはコマンド処理プログラムに必要な検査機能を組み込むことができます。このような場合、CRTCMD コマンドにコマンド処理プログラムと妥当性検査プログラムの両方の名前を指定します。

コマンドに妥当性検査プログラムがある場合、システムはそのコマンド・パラメーター値を有効な妥当性検査プログラムに渡します。 このことは、システムがコマンド処理プログラムを呼び出す前に行われます。妥当性検査プログラムは、次の条件のときに、構文検査中に行われます。

- コマンドの実行中。

- 原始ステートメント入力ユーティリティー (SEU) を使用して、 CL のソース・メンバーにコマンドを入力し、さらにプログラマーが、コマンドで指定されるパラメーターに変数ではなく定数を使用している場合。
- コマンドで指定されるすべてのパラメーターに、変数ではなく定数を使用している CL ソース・プログラムをコンパイルしている場合。

プログラムがエラーを見つけると、ユーザーはメッセージを受け取り、エラーを即座に訂正できます。コマンド処理プログラムは、渡されたデータは正しいものであるという前提に基づいて処理を行うことができます。

関連概念:

390 ページの『コマンド関連の API』
一部のアプリケーション・プログラミング・インターフェース・プログラムは、コマンドで使用できます。

関連タスク:

377 ページの『CL コマンドの妥当性検査プログラム』
構文エラーを検出して、コマンドの診断メッセージを送信するには、妥当性検査プログラムを作成します。

CL コマンドのプロンプト一時変更プログラム*:*

プロンプト一時変更プログラムを作成して、コマンドのプロンプトでパラメーターのデフォルト値ではなく現在の値を表示できます。

例えば、変更コマンドでプロンプト一時変更プログラムを使用して、パラメーターの値を提供する (デフォルト値は *SAME) ことがよくあります。 プロンプト一時変更プログラムの使用はオプションです。

関連タスク:

355 ページの『CL コマンドのキー・パラメーターおよびプロンプト一時変更プログラム』
プロンプト一時変更プログラムを使用することによって、コマンドのプロンプトの表示時点で、デフォルト値ではなく現在の値を表示できます。キー・パラメーターは、例えばオブジェクトの名前などのようにオブジェクトを固有なものとして識別するパラメーターです。

コマンド処理プログラム*:*

コマンド処理プログラム (CPP) は、要求された機能を実行するためにコマンド分析プログラムが呼び出すプログラムです。

CPP は、CL プログラム、別の高水準言語 (HLL) プログラム、または REXX プロシージャーのいずれでも構いません。 例えば、ユーザー・コマンドが呼び出すアプリケーション・プログラムであっても構いませんし、1 つのシステム・コマンドか一連のコマンドが入っている CL プログラムか REXX プロシージャーであっても構いません。

CPP は、コマンド定義ステートメントによって定義されているパラメーターを受け入れなければなりません。

CL コマンド出口プログラムおよび独立 *ASP:*

CL コマンドによって必要とされるコマンド出口プログラムは、CL コマンドと異なる独立補助記憶域プール (ASP) に存在することはできません。

コマンド処理プログラム、妥当性検査プログラム、プロンプト一時変更プログラム、選択プログラム、またはプロンプト制御プログラムを含む、コマンドによって必要とされるすべての出口プログラムは、コマンドと同じ独立補助記憶域プール (ASP)、またはシステム ASP (ASP 1)、あるいは基本 ASP (ASP 2-32) 内になければなりません。コマンドと出口プログラムがそれぞれ別の独立 ASP 内にあってはなりません。これらの出口プログラムが常駐する独立 ASP が利用不能な場合 (例えば、独立 ASP 装置がオフに構成変更されている場合)、コマンドを実行する際に問題が生じる可能性があります。

CL コマンドを定義するために必要な権限**:**

CL コマンドを作成する際、プログラムおよびライブラリーに対して特定の権限が必要となります。

ユーザー作成のコマンドを使用するユーザーは、そのコマンドの操作権限と、そのコマンド処理プログラムおよび任意作成の妥当性検査プログラムのデータ権限を持っている必要があります。 さらに、該当のコマンドが入っているライブラリーの読み取り権限、コマンド処理プログラムの読み取り権限、および妥当性検査プログラムの読み取り権限も必要です。 コマンド処理プログラムまたは妥当性検査プログラムが、なんらかのサービス・プログラムを参照する場合、ユーザーはそのサービス・プログラムおよびサービス・プログラム・ライブラリーの実行権限を持っていなければなりません。 ユーザーには、以下に挙げるプログラムの実行権限が必要です。

- コマンド処理プログラム (CPP)。
- 妥当性検査プログラム (VCP)。
- CPP または VCP により使用されるサービス・プログラム。
- CPP、VCP、およびサービス・プログラムを含むライブラリー。

ユーザーは、コマンド処理プログラムで他のコマンドを実行するための適切な権限も必要です。ユーザーは、ファイルに対して、オープンのための権限もなければなりません。

例**:** ***CL*** コマンドの作成**:**

この例は、システム・オペレーターがシステムを開始するためのコマンドを呼び出すコマンドを作成する方法を示しています。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

次の例では、IBM 提供のソース・ファイルを使用することを想定しています。

S は新しいコマンドの名前です (CMD パラメーターによる指定)。 STARTUP は、コマンド処理プログラムの名前 (PGM パラメーターによる指定) であると同時に、コマンド定義ステートメントが入っているソース・メンバーの名前 (SRCMBR パラメーターによる指定) です。以上で、システム・オペレーターは、S を入力してコマンドを呼び出すか、CALL STARTUP を入力してコマンド処理プログラムを呼び出すことができます。

1. メンバー名として STARTUP を用いて、コマンド定義ソース・ステートメントをソース・ファイル QCMDSRC に入力します。
   ```
   CMD PROMPT('S Command for STARTUP')
   ```
2. 以下のコマンドを入力することにより、コマンドを作成します。
   ```
   CRTCMD CMD(S) PGM(STARTUP) SRCMBR(STARTUP)
   ```
3. STARTUP プログラム (コマンド処理プログラム) のソース・ステートメントを入力します。

```
PGM
STRSBS QINTER
STRSBS QBATCH
STRSBS QSPL
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(WTR)
STRPRTWTR DEV(WSPR2) OUTQ(WSPRINT) WTR(WTR2)
SNDPGMMSG MSG('STARTUP procedure completed') MSGTYPE(*COMP)
ENDPGM
```

4. バインド制御言語プログラム作成 (CRTBNDCL) コマンドを使用して、このプログラムを作成します。

```
CRTBNDCL STARTUP
```

関連情報:

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

CL コマンドの定義**:**

コマンドを作成するには、まずコマンド定義ステートメントによりそのコマンドを定義しなければなりません。

コマンド定義ステートメントの一般形式およびコーディング規則の概要は、以下のとおりです。

ステートメント	コーディング規則
CMD	CMD ステートメントは、1 つだけ必ず使用しなければなりません。この CMD ステートメントは、ソース・ファイルの任意の位置に入れることができます。
PARM	最高 99 個の PARM ステートメントを指定できます。 PARM ステートメントをソース・ファイルに入力した順序に基づいて、パラメーターがコマンド処理プログラム (CPP) と妥当性検査プログラムに渡される順序が決まります。コマンド処理プログラムに渡す個々のパラメーターごとに 1 つずつ、PARM ステートメントが必要です。パラメーターをキー・パラメーターとして指定する場合には、該当する PARM ステートメントに KEYPARM(*YES) を指定しなければなりません。 KEYPARM(*YES) を指定するパラメーターの数は、変更したいオブジェクトを固有に定義するのに必要な数に限定しなければなりません。キー・パラメーターを指定する場合、コマンドの作成時にプロンプト一時変更プログラムを指定する必要があります。キー・パラメーターは、PMTCTL(*PMTRQS) または PMTCTL(label) を使用して指定することはできません。
ELEM	1 つのリストには最高 300 個の ELEM ステートメントを指定できます。 ELEM ステートメントをソース・ファイルに入力した順序に基づいて、リスト内での要素の順序が決まります。最初の ELEM ステートメントには、そのリストに対応する PARM ステートメントか ELEM ステートメントの TYPE パラメーターに指定したステートメント・ラベルに一致するラベルを付けなければなりません。
QUAL	1 つの修飾名に対して最高 300 個の修飾子を指定できます。 QUAL ステートメントをソース・ファイルに入力した順序に基づいて、修飾子の指定順序と、修飾子を妥当性検査プログラムおよびコマンド処理プログラムへ渡す順序が決まります。
DEP	DEP ステートメントは、その DEP ステートメントで参照されるすべての PARM ステートメントの後に入れられなければなりません。したがって、DEP ステートメントは通常ソース・ファイルの終わりに指定されます。
PMTCTL	PMTCTL ステートメントは、その PMTCTL ステートメントで参照されるすべての PARM ステートメントの後に入れられなければなりません。したがって、PMTCTL ステートメントは通常ソース・ファイルの終わりに指定されます。

| ソース・ファイル中の ELEM ステートメントや QUAL ステートメントの前には、少なくとも 1 つの
| PARM ステートメントがなければなりません。コマンド定義ステートメントを入力したソース・ファイル
| は、そのコマンドの作成時に、CRTCMD コマンドにより使用されます。ソース・エディターを使用して、

| コマンド定義ステートメントをデータベース・ソース・ファイル・メンバーまたは **IFS** ストリーム・ファ
| イルに入力してください。コマンド定義ステートメントの場合も、CL コマンドと同様にプロンプトを出す
| ことが可能です。

関連概念:

487 ページの『制御言語の動的プロンプト・メッセージ』
コマンド (*CMD) が作成されたときに、そのコマンドのオブジェクトに保管されるメッセージ識別コードを使用して、メッセージ・ファイルからプロンプト・メッセージを動的に検索することができます。この機能により、単一のコマンドに複数の各国語で表記されたプロンプト・メッセージを含めることができます。

CMD ステートメントの使用*:*

CL コマンドを定義する場合には、一連のコマンド定義ステートメントとともに、CMD ステートメントを必ず 1 つだけ含める必要があります。

コマンドを定義する時点で、ユーザーに対するコマンドのプロンプト・テキストを指定できます。ユーザーが、コマンド全体に入力する代わりに、そのコマンドのプロンプトの表示を要求する場合があります。このような場合に、コマンドのプロンプト・テキストを指定しておけば、ユーザーがコマンド名を入力して、F4 (プロンプト) キーを押すと、コマンドのプロンプトが表示され、コマンド名および見出しのプロンプト・テキストが画面の 1 行目に表示されます。

コマンドのプロンプト・テキストを指定するには、CMD ステートメントの PROMPT パラメーターにプロンプトの見出しを指定します。次に、PARM、 ELEM、QUAL の各ステートメントの PROMPT パラメーターに、パラメーター、リストの要素、および修飾子についてのプロンプトをそれぞれ指定します。

CMD ステートメントの PROMPT パラメーターには、実際のプロンプト見出しテキストを 30 文字以内の文字ストリングとして指定できます。また、メッセージ記述のメッセージ識別コードを指定することもできます。次に示す例では、コマンド ORDENTRY に対して文字ストリングが指定されています。

```
CMD PROMPT('Order Entry')
```

ユーザーがコマンドを入力して F4 キーを押すと、プロンプトの 1 行目が次のように表示されます。

```
Order Entry (ORDENTRY)
```

定義するコマンドにプロンプト・テキストを指定しない場合には、CMD ステートメントには CMD という語だけが必要ですが、文書化の目的で PROMPT キーワードを使用することをお勧めします。

コマンド定義ソースでの作成オプション*:*

コマンド作成 (CRTCMD) コマンドの複数の作成オプション・パラメーターを、CMD ステートメントに指定することもできます。CMD ステートメントに指定される値は、**CRTCMD** コマンドに指定されるどの値よりも優先されます。

例**:** プロンプト・テキストおよびコマンド作成オプションの定義

```
CMD   PROMPT(UCD0002) PMTFILE(MYCMDPMT *DYNAMIC)+
      MSGF(MYCMDMSG)  TEXT(*CMDPMT)  MAXPOS(2)  +
      PRDLIB(MYAPPLIB)  HLPID(*CMD) +
      HLPPNLGRP(MYAPPHLP)
```

このステートメントは、ライブラリー・リストを使用して位置指定される、メッセージ・ファイル MYCMDPMT 内のメッセージ UCD0002 のコマンドについて、プロンプト・テキストを設定します。このコマンドについて定義されたプロンプト・テキスト・メッセージはすべて、コマンドのプロンプトが出され

るときに動的に取り出されます。コマンド定義の従属 (DEP) ステートメントから送信されるエラー・メッセージは、ライブラリー・リストを使用して位置指定される、メッセージ・ファイル MYCMDMSG 内にあります。コマンド・オブジェクトのテキストは、コマンド・プロンプト・テキストと同じです。コマンドの最初の 2 つのパラメーターのみ、関連付けられたパラメーター・キーワードを使用せずに、定位置形式で指定できます。ライブラリー MYAPPLIB は、コマンドが実行される前に自動的にライブラリー・リストに追加され、コマンドが完了するときにライブラリー・リストから除去されます。パネル・グループ MYAPPHLP のコマンド・ヘルプ・モジュールのヘルプ ID は、 **CRTCMD** コマンドが作成するコマンド・オブジェクトの名前で始まります。

CL コマンド・パラメーターの定義**:**

CL コマンド・パラメーターを定義するには、PARM ステートメントを使用しなければなりません。

各コマンドには、最高 99 個のパラメーターを定義できます。

PARM ステートメントには、次の事項を指定できます。

- パラメーターのキーワードの名前
- そのパラメーターがキー・パラメーターであるかどうか
- 渡すことができるパラメーター値のタイプ
- 値の長さ
- 必要に応じて、パラメーターのデフォルト値

さらに、パラメーターを指定するさいには、以下の事項を考慮に入れなければなりません。 (関連の PARM ステートメントのパラメーターは括弧に入れて示しています。)

- コマンド処理プログラムにより値が戻されるかどうか (RTNVAL)。 RTNVAL (*YES) を指定して、値が戻されるようにしたい場合には、コマンドを呼び出す時点でコマンドに戻り変数をコーディングしなければなりません。 RTNVAL(*YES) パラメーターに対して変数を指定しなかった場合、コマンド処理プログラムにはヌル・ポインターが渡されます。
- パラメーターが、プロンプトとしてユーザーには表示されず、定数としてコマンド処理プログラムに渡されるのかどうか (CONSTANT)。
- パラメーターが VALUES、SPCVAL、SNGVAL のいずれかのパラメーターで指定された特定の値に限定されているかどうか (RSTD)、あるいはパラメーター・タイプ、長さ、値の範囲、指定の関係比較条件に一致した値であればどのような値でも指定できるのかどうか。
- 特定の有効なパラメーター値の内容 (VALUES、SPCVAL、SNGVAL)。
- パラメーター値の妥当性を判別するために行うテスト (REL と RANGE)。
- オプションのパラメーターか必須パラメーターか (MIN)。
- 単純リストを必要とするパラメーターに対して指定できる値の数 (MIN と MAX)。
- 印刷不能データ (16 進数 00 から 3F、または 16 進数 FF の値に対応する文字) をパラメーターの値として指定できるかどうか (ALWUNPRT)。
- 変数名をパラメーターの値として指定できるかどうか (ALWVAR)。
- 値がプログラム名であるかどうか (PGM)。
- 値がデータ域名であるかどうか (DTAARA)。
- 値がファイル名であるかどうか (FILE)。
- 値の長さが指定の長さに正確に一致していなければならないかどうか (FULL)。
- 値とともに、値の長さを指定しなければならないかどうか (VARY)。

- 式をパラメーターの値として指定できるかどうか (EXPR)。
- パラメーターとして渡される値について属性情報を指定しなければならないかどうか (PASSATR)。
- 定義するパラメーターが指定されなかった場合に、コマンド処理プログラムまたは妥当性検査プログラムに値を渡すかどうか (PASSVAL)。
- 大文字小文字値を保持するか、または大文字小文字値を大文字に変換するかどうか (CASE)。
- リスト内リストの変位 (LISTDSPL) の値が、2 バイトの 2 進数か 4 バイトの 2 進数のどちらであるか。
- メッセージ識別コード、およびパラメーターのプロンプト・テキスト (PROMPT)。
- プロンプト画面の選択項目フィールドに表示する有効な値 (CHOICE)。
- 選択値をプログラムにより提供するかどうか (CHOICEPGM)。
- パラメーターのプロンプトを別のパラメーターで制御するかどうか (PMTCTL)。
- PMTCTL ステートメントの値をプログラムにより提供するかどうか (CTL キーワードで参照されるパラメーターに対して) (PMTCTLPGM)。
- ジョブ・ログ内、またはコマンド・プロンプトのさいに、値を非表示扱いにするかどうか (DSPINPUT)。

関連タスク:

336 ページの『単純リストの CL コマンド・パラメーターに対する CL またはその他の HLL の使用』
CL または高水準言語 (HLL) を使用するコマンドを実行する場合、単純リストの要素はこの形式でコマンド処理プログラムに渡されます。

341 ページの『混合リストの CL コマンド・パラメーターに対する CL またはその他の HLL の使用』
CL コマンドを実行する場合、混合リストの要素はこの形式でコマンド処理プログラムに渡されます。

347 ページの『修飾名 CL コマンド・パラメーターの定義』
修飾名とは、オブジェクト名の前に、そのオブジェクトが保管されているライブラリーの名前を付けたものです。

349 ページの『修飾名の CL コマンド・パラメーターに対する REXX の使用』
REXX を使用するコマンドを実行した場合、修飾名がコマンド処理プログラムへ渡される方法は、パラメーターに対して値を入力した場合と同じです。末尾ブランクは渡されません。

***CL** コマンド・パラメーターのキーワードの名前付け:*

CL コマンド・パラメーターには、そのパラメーターの値として要求する情報の内容を表すキーワード名を付けます。

例えば、ユーザー名には USER、比較値には CMPVAL、受注タイプには OETYPE などを使用します。キーワードは最高 10 文字の英数字とし、最初の文字は英字でなければなりません。

***CL** コマンド・パラメーターのタイプ:*

CL コマンドにはさまざまなタイプのパラメーターが使用されます。

基本的なパラメーター・タイプは以下のとおりです (TYPE パラメーターの値を括弧内に示しています)。

- 10 進数 (*DEC)。パラメーターの値は 10 進数であり、LEN パラメーターに指定された長さのパック 10 進数として、コマンド処理プログラムに渡されます。少数部分の桁数がパラメーターに定義された桁数よりも多い値を指定した場合には、切り捨てが行われます。
- 論理値 (*LGL)。パラメーターの値は、論理値 '1' または '0' であり、長さ 1 の文字ストリング (F1 または F0) としてコマンド処理プログラムに渡されます。

- 文字 (*CHAR)。パラメーターの値は、文字ストリングで単一引用符で囲んで指定することもでき、LEN パラメーターに指定された長さの文字ストリングとして、コマンド処理プログラムに渡されます。 値は、単一引用符を取り除いた形で渡され、左寄せとブランク埋め込みが行われます。
- 名前 (*NAME)。パラメーターの値は、基本名を表す文字ストリングです。 名前の最大長は 256 文字です。 最初の文字は英字 (A から Z)、$、#、または @ です。残りの文字は最初の文字と同じものに加えて 0 から 9 の数字、下線 (_)、およびピリオド (.) にすることができます。. 二重引用符 (") で囲まれた文字ストリングも名前として有効です。 システムは値を、LEN パラメーターに指定された長さの文字ストリングとして、コマンド処理プログラムに渡します。 値は左寄せにされ、ブランクが埋め込まれます。 通常は、オブジェクト名に *NAME タイプを使用します。 名前パラメーターの値として *LIBL や *NONE などの特殊値を入力できる場合には、その特殊値を SPCVAL パラメーターで記述しなければなりません。 そうすれば、パラメーターの値としてディスプレイ装置ユーザーが許可されている特殊値の 1 つを入力した場合、システムは名前検査の規則を迂回します。
- 単純名 (*SNAME)。パラメーターの値は、*NAME の場合と同じ命名規則に従う文字ストリングです。ただし、ピリオド (.) は使用できません。
- 通信名 (*CNAME)。パラメーターの値は、*NAME の場合と同じ命名規則に従う文字ストリングです。ただし、ピリオド (.) および下線 (_) は使用できません。
- パス名 (*PNAME)。パラメーターの値は、文字ストリングで単一引用符で囲んで指定することもでき、LEN パラメーターに指定された長さの文字ストリングとして、コマンド処理プログラムに渡されます。値は、単一引用符を取り除いた形で渡され、左寄せとブランク埋め込みが行われます。
- 総称名 (*GENERIC)。パラメーターの値は、アスタリスク (*) で終わる総称名です。名前がアスタリスクで終わっていない場合、その総称名は、1 つのオブジェクト名と見なされます。総称名は、アスタリスクの前に指定されている文字で始まる名前を持つすべてのオブジェクトを、1 つのグループとして識別するものです。例えば、INV* は、INV、INVOICE、INVENTORY など、名前が INV で始まるオブジェクトを識別します。総称名は、そこに指定された文字で始まるオブジェクト名を見つけるために、コマンド処理プログラムに渡されます。
- 日付 (*DATE)。パラメーターの値は文字ストリングであり、コマンド処理プログラムに渡されます。コマンド・ストリングは、cyymmdd (c = 世紀数字、y = 年、m = 月、d = 日) の形式を使用します。システムは、コマンドの日付パラメーターで指定された年に基づいて、世紀数字を設定します。 指定される年が 4 桁の場合には、システムは 19 で始まる年については、世紀数字を 0 に設定します。 システムは 20 で始まる年については、世紀数字を 1 に設定します。 2 桁で指定されている年については、yy が 40 から 99 の場合には、システムは世紀数字を 0 に設定します。 しかし、yy が 00 から 39 の場合には、システムは世紀数字を 1 に設定します。ユーザーは、コマンドの日付パラメーターに、日付形式 (DATFMT) ジョブ属性で指定されている形式で日付を入力しなければなりません。日付区切り記号 (DATSEP) ジョブ属性は、日付を入力するのに使用するオプションの区切り文字を決定します。DATFMT と DATSET ジョブ属性は、ジョブ変更 (CHGJOB) コマンドを使用して変更できます。 プログラムは、1940 年 1 月 1 日から 2039 年 12 月 31 日までの範囲の 2 桁の年の日付を読み取ります。 4 桁の年の日付は、1928 年 8 月 24 日から 2071 年 5 月 9 日までの範囲になければなりません。
- 時刻 (*TIME)。パラメーターの値は文字ストリングです。システムはこのストリングを hhmmss (ただし、h = 時、m = 分、s = 秒) の形式でコマンド処理プログラムに渡します。時間区切り記号 (TIMSEP) ジョブ属性は、時間を入力するのに使用するオプションの区切り文字を決定します。TIMSEP ジョブ属性は、ジョブ変更 (CHGJOB) コマンドを使用して変更できます。
- 16 進数 (*HEX)。パラメーターの値は 16 進数です。指定される文字は 0 から F でなければなりません。値は、16 進数 (EBCDIC) の文字 (1 バイト当たり 2 桁の 16 進数) としてコマンド処理プログラムに渡され、右寄せにされてゼロが埋め込まれます。値を単一引用符で囲む場合には、偶数個の桁が必要です。

- ゼロ要素 (*ZEROELEM)。パラメーターの値は、コマンドに値を指定しなくてよいゼロ要素のリストと見なされます。リスト形式の値を取るパラメーターでは、 (コマンド処理プログラムでは値を受け取るものと予期していても、) そのパラメーターに値が入力されないようにしたい場合に、このパラメーター・タイプを使用します。例えば、同じコマンド処理プログラムを使用する 2 つのコマンドを作成する場合に、一方のコマンドではパラメーター値として渡すことができるようにし、もう一方のコマンドでは値を渡さないようにしたいことがあります。このような場合に、この 2 番目のコマンドのパラメーターを TYPE(*ZEROELEM) として定義します。
- 整数 (*INT2 または *INT4)。パラメーターの値は整数であり、2 バイトまたは 4 バイトの符号付き 2 進数として渡されます。 2 進数は、CL プロシージャーまたはプログラムで TYPE(*INT) の変数として宣言できます。また、TYPE(*CHAR) を使用して、それらを %BINARY 組み込み関数を使用して処理することもできます。
- 符号なし整数 (*UINT2 または *UINT4)。パラメーターの値は整数であり、2 バイトまたは 4 バイトの符号なし 2 進数として渡されます。 2 進数は、CL プロシージャーまたはプログラムで TYPE(*UINT) の変数として宣言できます。また、TYPE(*CHAR) を使用して、それらを %BINARY 組み込み関数を使用して処理することもできます。
- ヌル (*NULL)。パラメーターの値はヌル・ポインターであり、常にプレース・ホルダー (場所を確保するもの) としてコマンド処理プログラムに渡されます。このパラメーター・タイプを指定できる PARM ステートメントのキーワードは、KWD、MIN、および MAX だけです。
- コマンド・ストリング (*CMDSTR)。パラメーターの値はコマンドです。 CL 変数を使用して、*CMDSTR パラメーターで指定されているコマンドにパラメーターを指定できます。 しかし、それらを使用して *CMDSTR パラメーター全体を指定することはできません。 例えば、制御言語プログラムまたはプロシージャーで、 "SBMJOB CMD(DSPLIB LIB(&LIBVAR))" は有効ですが、 "SBMJOB CMD(&CMDVAR)" は無効です。
- ステートメント・ラベル。ステートメント・ラベルは、この PARM ステートメントにより定義する修飾名や混合リストについてさらに詳しく記述するための、一連の QUAL ステートメントまたは ELEM ステートメントの先頭のステートメントを識別します。

以下のパラメーター・タイプは、IBM 提供のコマンドにだけ適用されます。

- 式 (*X)。パラメーターの値は、文字ストリング、変数名、または数値です。その値は、含まれているものが数字、正符号か負符号、小数点、またはこれら全部だけであった場合は、数値として渡されます。それ以外は、文字ストリングとして渡されます。
- 変数名 (*VARNAME)。パラメーター値は変数名であり、文字ストリングとしてコマンド処理プログラムに渡されます。値は左寄せにされ、ブランクが埋め込まれます。 変数とは、処理時に実際のデータ値を参照する名前のことです。変数名は 10 文字までの英数字 (最初の文字は英字) で、名前の前にアンパーサンド (&) を付けます (例えば、&PARM)。変数名が IBM i オペレーティング・システムで使用されている命名規則に従っていない場合は、その名前を単一引用符で囲む必要があります。
- コマンド (*CMD)。パラメーターの値はコマンドです。 例えば、CL コマンド IF はパラメーター THEN を伴いますが、この THEN パラメーターの値は、別のコマンドでなければなりません。

CL コマンド・パラメーター値の長さ*:*

長さ (LEN) パラメーターを使用して、パラメーターの長さを指定します。

*DATE または *TIME のパラメーター・タイプの場合には、 *DATE は常に 7 文字であり、*TIME は常に 6 文字です。次の表は、長さを指定できる各パラメーターのタイプについて、最大長とデフォルトの長さを示しています。

データ・タイプ	最大長	デフォルトの長さ
*DEC	24 (小数部 9 桁)	15 (小数部 5 桁)
*LGL	1	1
*CHAR	5000	32
*NAME	256	10
*SNAME	256	10
*CNAME	256	10
*GENERIC	256	10
*HEX	256	1
*X	(256 24 9)	(1 15 5)
*VARNAME	11	11
*CMDSTR	20000	256
*PNAME	5000	32

ここに示した最大長は、コマンドの実行時に各パラメーター・タイプで指定できる最大の長さです。ただし、コマンド定義ステートメントの文字定数に指定できる最大長は 32 文字です。この制約は、CONSTANT、DFT、VALUES、REL、RANGE、 SPCVAL、および SNGVAL の各パラメーターに適用されます。 CL コマンドのプロンプト画面で表示される入力フィールドには、一定の長さがあります。入力フィールドの長さは、1 から 12 文字および 17、25、32、50、 80、132、256、512 文字です。あるパラメーターの長さが上記以外の長さである場合、その入力フィールドは次に大きいフィールド長で表示されます。パラメーターの長さが 512 文字より長い場合でも、IBM i の CL コマンド・プロンプターが表示する入力フィールドは、最大で 512 文字です。

関連タスク:

285 ページの『プログラム呼び出しコマンドを使用して、呼び出されるプログラムに制御を渡す』
CL プロシージャーで**プログラム呼び出し (CALL)** コマンドを出す場合、呼び出されるプログラムに渡す各パラメーター値には、文字ストリング定数、数値定数、論理定数、または CL 変数のどれでも指定することができます。

デフォルトの *CL* コマンド・パラメーター値*:*

オプションのパラメーターを定義する場合には、ユーザーがコマンドにパラメーター値を入力しなかった場合に使用する値を DFT パラメーターで指定できます。

この値は、デフォルト値 と呼ばれます。デフォルト値は、該当するパラメーターの値に関するすべての要件 (タイプ、長さ、特殊値など) を満たしていなければなりません。オプションのパラメーターにデフォルト値を指定しなかった場合には、次に示すデフォルト値が使用されます。

データ・タイプ	デフォルト値
*DEC	0
*INT2	0
*INT4	0
*UINT2	0
*UINT4	0
*LGL	'0'
*CHAR	ブランク

データ・タイプ	デフォルト値
*NAME	ブランク
*SNAME	ブランク
*CNAME	ブランク
*GENERIC	ブランク
*DATE	ゼロ ('F0')
*TIME	ゼロ ('F0')
*ZEROELEM	0
*HEX	ゼロ ('00')
*NULL	ヌル
*CMDSTR	ブランク
*PNAME	ブランク

例: *CL* コマンド・パラメーターの定義:

この例は、CL コマンドによるアプリケーションの呼び出しで使用されるパラメーターの定義方法を示しています。

以下の例では、受注アプリケーションを呼び出す CL コマンドのパラメーター OETYPE を定義しています。

```
PARM   KWD(OETYPE)  TYPE(*CHAR)  RSTD(*YES) +
       VALUES(DAILY WEEKLY MONTHLY)  MIN(1) +
       PROMPT('Type of order entry')
```

OETYPE パラメーターは、(MIN パラメーターに 1 が指定されているので) 必須パラメーターであり、その値は DAILY、WEEKLY、または MONTHLY に限定されています。 (RSTD パラメーターに *YES が指定されている)。 PROMPT パラメーターには、このパラメーターのプロンプト・テキストが指定されています。 LEN キーワードの指定がなく、TYPE(*CHAR) が定義されているので、デフォルト値を取って 32 がデフォルトの長さになります。

関連タスク:

363 ページの『CL コマンドの作成』
コマンド定義ステートメントによるユーザー・コマンドの定義が完了した時点で、**コマンド作成 (CRTCMD)** コマンドを使用して、そのコマンドを作成できます。

データ・タイプおよびパラメーターの制約事項:

これらの表は、パラメーター・タイプに従った有効なパラメーターの組み合わせを示しています。

X は、その組み合わせが有効なことを示し、また数字は表の後に示されている制約事項についての注の参照番号です。

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
*DEC	X	2	X	X	X	X	X	X	3	1
*LGL	X	2	X	X	X	X			3	1
*CHAR	X	2	X	X	X	X	X	X	3	1
*NAME	X		X	X	X	X	X	X	3	1
*SNAME	X		X	X	X	X	X	X	3	1
*CNAME	X		X	X	X	X	X	X	3	1

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
*PNAME	X	2	X	X	X	X	X	X	3	1
*GENERIC	X		X	X	X	X	X	X	3	1
*DATE			X	X	X	X	X	X	3	1
*TIME			X	X	X	X	X	X	3	1
*HEX	X		X	X	X	X	X	X	3	1
*ZEROELEM										
*INT2		2	X	X	X	X	X	X	3	1
*INT4		2	X	X	X	X	X	X	3	1
*UINT2		2	X		X		X	X	3	1
*UINT4		2	X		X		X	X	3	1
*CMDSTR	X		X		X					
*NULL	X									
STMT LABEL			X		X					X

注:

1. MAX の値が 1 より大きい場合だけ有効です。また、受け取り置き換え値も、CPP が REXX プロシージャーの場合には無視されます。 REXX プロシージャーのパラメーターとして渡される値は入力された値か、または各パラメーターのデフォルト値です。
2. コマンド CPP が REXX プロシージャーの場合には無効です。
3. CPP が REXX プロシージャーの場合、受け取り置き換え値は無視されます。 REXX プロシージャーのパラメーターとして渡される値は入力された値か、各パラメーターのデフォルト値です。

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
*DEC	X	X		X		X				
*LGL	X	X		X		X	X	1		
*CHAR	X	X	X	X	X	X	X	X	X	1
*NAME	X	X		X	X	X	X	X	X	1
*SNAME	X	X		X	X	X	X	X	X	1
*CNAME	X	X		X	X	X	X	X	X	1
*PNAME	X	X	X	X	X	X	X	X	X	1
*GENERIC	X	X		X	X	X	X	X	X	1
*DATE	X	X		X		X				
*TIME	X	X		X					X	
*HEX	X	X		X				X	X	
*ZEROELEM	X	X								
*INT2	X	X		X					X	
*INT4	X	X		X					X	
*UINT2	X	X		X					X	
*UINT4	X	X		X					X	
*CMDSTR	2	3		4						1
*NULL	2	3								
STMT LABEL	X	X			X					

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY

注:

1. CPP が REXX プロシージャーの場合、パラメーターは無視されます。
2. TYPE(*NULL) の場合には、MIN の値が 1 を超えてはなりません。
3. TYPE(*NULL) または TYPE(*CMDSTR) の場合には、MAX の値が 1 を超えてはなりません。
4. このタイプのパラメーターでは ALWVAR の値は無視されます。パラメーター・タイプが *CMDSTR の場合には、CL 変数は使用できません。

	PASSATR	PASSVAL	CASE	LISTDSPL	DSPINPUT
*DEC	1	X		X	X
*LGL	1	X		X	X
*CHAR	1	X	3	X	X
*NAME	1	X		X	X
*SNAME	1	X		X	X
*CNAME	1	X		X	X
*PNAME	1	X	3	X	X
*GENERIC	1	X		X	X
*DATE	1	X		X	X
*TIME	1	X		X	X
*HEX	1	X		X	X
*ZEROELEM					
*INT2	1	X		X	X
*INT4	1	X		X	X
*UINT2	1	X		X	X
*UINT4	1	X		X	X
*CMDSTR	1			X	X
*NULL					
STMT LABEL		2			

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
*DEC	X	X	X	X	X	
*LGL	X	X	X	X	X	
*CHAR	X	X	X	X	X	4
*NAME	X	X	X	X	X	4
*SNAME	X	X	X	X	X	4
*CNAME	X	X	X	X	X	4
*PNAME	X	X	X	X	X	4
*GENERIC	X	X	X	X	X	4
*DATE	X	X	X	X	X	
*TIME	X	X	X	X	X	
*HEX	X	X	X	X	X	4
*ZEROELEM						
*INT2	X	X	X	X	X	
*INT4	X	X	X	X	X	
*UINT2	X	X	X	X	X	
*UINT4	X	X	X	X	X	

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
*CMDSTR	X	X	X	X	X	4
*NULL						
STMT LABEL	X	X	X	X	X	X

注:

1. CPP が REXX プロシージャーの場合、パラメーターは無視されます。
2. CPP が REXX プロシージャーの場合、PASSVAL が渡すキーワードには、括弧の中にブランクも他の文字も入っていません。
3. 大文字小文字混合 (*MIXED) は *CHAR と *PNAME のタイプに限り使用できます。
4. *CHAR、*NAME、*SNAME、*CNAME、および *PNAME でのみ INLPMTLEN(*PWD) を使用することができます。

次の表は、PARM、ELEM、QUAL の各ステートメントについての、パラメーターの有効な組み合わせと制約事項を示したものです。例えば、LEN の行と DFT の列が交わる欄はブランクになっています。したがって制約事項は何もなく、LEN(XX) と DFT(XX) の組み合わせは有効です。しかし、DFT の行と CONSTANT の列が交わる欄には 4 が入っています。この数字は、表の後に示している、制約事項を説明した注の参照番号です。

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
LEN										
RTNVAL			1	1	1	1	1	1	1	1
CONSTANT		1			4					16
RSTD		1				7	9	9	7	7
DFT		1	4							
VALUES		1		7						
REL		1		9				9		
RANGE		1		9			9			
SPCVAL		1		7						
SNGVAL		1	21	7						
MIN					8					
MAX		2	2							10
ALWUNPRT										
ALWVAR		12								
PGM		1								
DTAARA		1								
FILE		1								
FULL		1								
EXPR		1	5							
VARY		3								
PASSATR		3								
PASSVAL		13						11		
CASE										
LISTDSPL										
CHOICE			14							
CHOICEPGM										

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
PMTCTL			15							
PMTCTLPGM			15							
PROMPT			6							
INLPMTLEN		17	17	17						

注:

1. RTNVAL パラメーターは、以下のどのパラメーターとも併用することはできません。
   CONSTANT、RSTD、DFT、VALUES、REL、RANGE、SPCVAL、SNGVAL、PGM、DTAARA、FILE、FULL、EXPR。
   CPP として REXX プロシージャーを使用するコマンドでは、RTNVAL パラメーターを使用できません。
2. MAX の値は、1 を超えてはなりません。
3. RTNVAL(*YES) および PASSATR(*YES) を指定した場合には、VARY(*YES) も指定しなければなりません。 RTNVAL(*YES) および VARY(*YES) を指定した場合には、*INT2 または *INT4 を使用しなければなりません。 *INT2 と *INT4 を組み合わせると無効になります。
4. CONSTANT パラメーターと DFT パラメーターを同時に指定することはできません。
5. EXPR(*YES) パラメーターと CONSTANT パラメーターを同時に指定することはできません。
6. PROMPT パラメーターは指定できません。
7. RSTD パラメーターを指定した場合には、VALUES、SPCVAL、または SNGVAL の各パラメーターのいずれか 1 つを同時に指定しなければなりません。
8. MIN の値は 0 でなければなりません。
9. REL、RANGE、および RSTD(*YES) の各パラメーターを同時に指定することはできません。
10. MAX の値が 1 より大きいか、パラメーター・タイプがステートメント・ラベルであるか、あるいはこの 2 つの条件を同時に満たしていなければなりません。
11. このパラメーターは、パラメーター PASSVAL(*NULL) を指定して定義されたパラメーターを参照することはできません。
    PASSVAL(*NULL) として定義された PARM ステートメントでは、パラメーター間の範囲指定は無効です。
12. RTNVAL(*YES) で指定した場合、ALWVAR(*NO) を指定できません。
13. PASSVAL(*NULL) を指定した場合、または MIN の値が 0 より大きい場合には、RTNVAL(*YES) を指定することはできません。
14. CHOICE パラメーターと CONSTANT パラメーターを同時に指定することはできません。
15. CONSTANT は、PMTCTL パラメーターおよび PMTCTLPGM パラメーターと同時に指定することはできません。
16. PARM ステートメントに SNGVAL パラメーターが定義されている場合には、ELEM/QUAL ステートメントで CONSTANT パラメーターを定義することはできません。
17. CONSTANT とともに INLPMTLEN パラメーターを使用することはできません。 FULL(*YES)、RTNVAL(*YES)、または RSTD(*YES) を指定した場合には、INLPMTLEN(*CALC) を指定するか、これをデフォルト値として使用しなければなりません。

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
LEN										
RTNVAL		2		8	1	1	1	1	1	3
CONSTANT		2							4	
RSTD										
DFT	5									
VALUES										
REL										
RANGE										
SPCVAL										
SNGVAL		7								
MIN		6								

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
MAX	6									
ALWUNPRT										
ALWVAR										
PGM						9	9			
DTAARA					9		9			
FILE						9	9			
FULL										
EXPR										
VARY										
PASSATR										3
PASSVAL	10									
CASE										
LISTDSPL										
CHOICE										
CHOICEPGM										
PMTCTL	11									
PMTCTLPGM										
PROMPT										
INLPMTLEN								12		

注:

1. RTNVAL パラメーターは、以下のどのパラメーターとも併用することはできません。
   CONSTANT、RSTD、DFT、VALUES、REL、RANGE、SPCVAL、SNGVAL、PGM、DTAARA、FILE、FULL、EXPR。
   CPP として REXX プロシージャーを使用するコマンドでは、RTNVAL パラメーターを使用できません。
2. MAX の値は、1 を超えてはなりません。
3. RTNVAL(*YES) および PASSATR(*YES) を指定した場合には、VARY(*YES) も指定しなければなりません。 RTNVAL(*YES) および VARY(*YES) を指定した場合には、*INT2 または *INT4 を使用しなければなりません。 *INT2 と *INT4 を組み合わせると無効になります。
4. EXPR(*YES) パラメーターと CONSTANT パラメーターを同時に指定することはできません。
5. MIN の値は 0 でなければなりません。
6. MIN パラメーターに指定した値が、MAX パラメーターで指定した値を超えてはなりません。
7. MAX の値が 1 より大きいか、パラメーター・タイプがステートメント・ラベルであるか、あるいはこの 2 つの条件を同時に満たしていなければなりません。
8. RTNVAL(*YES) で指定した場合、ALWVAR(*NO) を指定できません。
9. PGM(*YES)、DTAARA(*YES)、および *NO 以外の FILE パラメーター値を、同時に指定することはできません。
10. PASSVAL(*NULL) を指定した場合、または MIN の値が 0 より大きい場合には、RTNVAL(*YES) を指定することはできません。
11. MIN の値が 0 より大きい場合には、PMTCTL を指定することはできません。
12. FULL(*YES)、RTNVAL(*YES)、または RSTD(*YES) を指定した場合には、INLPMTLEN(*CALC) を指定するか、これをデフォルト値として使用しなければなりません。

	PASSATR	PASSVAL	CASE	LISTDSPL	DSPINPUT
LEN					
RTNVAL	1	4			
CONSTANT				9	5
RSTD					

	PASSATR	PASSVAL	CASE	LISTDSPL	DSPINPUT
DFT					
VALUES					
REL					
RANGE		3			
SPCVAL					
SNGVAL					
MIN		4			
MAX					
ALWUNPRT					
ALWVAR					
PGM					
DTAARA					
FILE					
FULL					
EXPR					
VARY	1				
PASSATR					
PASSVAL					
CASE			10		
LISTDSPL				11	
CHOICE					
CHOICEPGM					
PMTCTL					
PMTCTLPGM					
PROMPT					
INLPMTLEN					

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
LEN						
RTNVAL						12
CONSTANT			7	7	2	12
RSTD						12
DFT						
VALUES						
REL						
RANGE						
SPCVAL						
SNGVAL						
MIN			8			
MAX						

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
ALWUNPRT						
ALWVAR						
PGM						
DTAARA						
FILE						
FULL						12
EXPR						
VARY						
PASSATR						
PASSVAL						
CASE						
LISTDSPL						
CHOICE		6				
CHOICEPGM	6					
PMTCTL						
PMTCTLPGM						
PROMPT						
INLPMTLEN						

注**:**

1. RTNVAL(*YES) および PASSATR(*YES) を指定した場合には、VARY(*YES) も指定しなければなりません。 RTNVAL(*YES) および VARY(*YES) を指定した場合には、*INT2 または *INT4 を使用しなければなりません。 *INT2 と *INT4 を組み合わせると無効になります。
2. PROMPT パラメーターは指定できません。
3. このパラメーターは、パラメーター PASSVAL(*NULL) を指定して定義されたパラメーターを参照することはできません。 PASSVAL(*NULL) として定義された PARM ステートメントでは、パラメーター間の範囲指定は無効です。
4. PASSVAL(*NULL) を指定した場合、または MIN の値が 0 より大きい場合には、RTNVAL(*YES) を指定することはできません。
5. CHOICE パラメーターと CONSTANT パラメーターを同時に指定することはできません。
6. CHOICE(*PGM) を指定した場合には、CHOICEPGM に名前を指定しなければなりません。
7. CONSTANT は、PMTCTL パラメーターおよび PMTCTLPGM パラメーターと同時に指定することはできません。
8. MIN の値が 0 より大きい場合には、PMTCTL を指定することはできません。
9. CONSTANT と、DSPINPUT(*NO) または DSPINPUT(*PROMPT) とを同時に指定することはできません。
10. CASE パラメーターは PARM ステートメントと ELEM ステートメントに限り有効です。 CASE は QUAL ステートメントでは無効です。
11. LISTDSPL パラメーターは PARM ステートメントに限り有効です。

12. CONSTANT とともに INLPMTLEN パラメーターを使用することはできません。FULL(*YES)、RTNVAL(*YES)、または RSTD(*YES) を指定した場合には、INLPMTLEN(*CALC) を指定するか、これをデフォルト値として使用しなければなりません。

CL コマンド・リスト・パラメーターの定義**:**

1 つの値だけでなく、値のリストを受け入れるパラメーターを定義できます。

定義できるリストのタイプは以下のとおりです。

- 単純リスト。1 つのパラメーターに対して、同じタイプの 1 つまたは複数の値を指定できます。
- 混合リスト。1 つのパラメーターに対して、別個に定義された 1 組の値を指定できます。
- リスト内リスト。1 つのパラメーターに対してリストを複数指定すること、あるいは混合リスト内の 1 つの値としてリストを指定できます。

以下のサンプル・コマンドのソースは、それぞれ異なるタイプのリストの例を示しています。

```
             CMD        PROMPT('リストの例のコマンド')


   /* THE FOLLOWING PARAMETER IS A SIMPLE LIST. IT WILL ACCEPT UP TO   */
   /* 5 NAMES.                                                         */
             PARM       KWD(SIMPLST) TYPE(*NAME) LEN(10) DFT(*ALL) +
                         SPCVAL((*ALL)) MAX(5) PROMPT('5 つまでの  +
                         名前の単純リスト')


   /* THE FOLLOWING PARAMETER IS A MIXED LIST OF 3 VALUES, EACH OF A   */
   /* DIFFERENT TYPE AND/OR LENGTH. EACH ELEMENT MAY NOT BE REPEATED.  */
             PARM       KWD(MXDLST) TYPE(MLSPEC) PROMPT('これは 3   +
                  つの変数の混合リスト')
MLSPEC:      ELEM       TYPE(*CHAR) LEN(4) PROMPT('要素 1')
             ELEM       TYPE(*DEC) LEN(3 0) PROMPT('2 番目')
             ELEM       TYPE(*CHAR) LEN(10) PROMPT('最後の +
                          要素')


/* THE FOLLOWING PARAMETER IS A LIST WITHIN A LIST. IT CONTAINS A   */
/* LIST OF UP TO 2 ELEMENTS, WHICH MAY BE REPEATED UP TO 3 TIMES.   */
             PARM       KWD(LWITHINL1) TYPE(LWLSPECA) MAX(3) +
                          PROMPT('2 つの要素の折り返しリスト')
LWLSPECA:    ELEM         TYPE(*CHAR) LEN(10) PROMPT('折り返しリス +
                            トの最初の部分')

             ELEM         TYPE(*DEC) LEN(5 0) PROMPT('折り返しリス +
                            トの 2 番目の部分')


/* THE FOLLOWING PARAMETER IS A LIST WITHIN A LIST. IT CONTAINS A   */
/* LIST OF UP TO 2 ELEMENTS, THE FIRST OF WHICH MAY BE REPEATED     */
/* UP TO 3 TIMES.                                                   */
             PARM       KWD(LWITHINL2) TYPE(LWLSPECB) MAX(1) +
                          PROMPT('混合内折り返し単純リスト')
LWLSPECB:    ELEM       TYPE(*CHAR) LEN(10) MAX(3) PROMPT('リスト内単+
                          純リスト')
             ELEM       TYPE(*DEC) LEN(5 0) PROMPT('リスト内単一パラ +
                          メーター')
```

以下の画面は、前述のサンプル・コマンドについてのプロンプト画面です。

```
                        リストの例のコマンド (LSTEXAMPLE)

選択項目を入力して，実行キーを押してください。

5 つまでの名前の単純リスト  . .  SIMPLST       *ALL

                          値の続きは+
3 つの変数の混合リスト           MXDLST
  要素 1  . . . . . . . . . . .
  2 番目  . . . . . . . . . . .
  最後の要素  . . . . . . . . .
2 つの要素の折り返しリスト       LWITHINL1
  折り返しリストの最初の部分  .
  折り返しリストの 2 番目の部分
                          値の続きは+
混合内折り返し単純リスト         LWITHINL2
  リスト内単純リスト  . . . . .
                          値の続きは＋
  リスト内単一パラメーター  . .









               終わり
F3=終了    F4=プロンプト    F5=最新表示    F12=取り消し     F13=この画面の使用法
F24=キーの続き
```

単純リスト・コマンド・パラメーターの定義*:*

単純リストは、パラメーターにより指定されたタイプの 1 つまたは複数の値を受け入れることができます。

例えば、パラメーターがユーザー名に対するものであれば、単純リストはそのパラメーターに複数のユーザー名を指定できることを意味します。

```
USER(JONES SMITH MILLER)
```

パラメーターの値が単純リストである場合には、PARM ステートメントの MAX パラメーターを使用して、そのリストに指定できる要素の最大数を指定します。単純リストの場合には、パラメーターの定義に PARM ステートメント以外のコマンド定義ステートメントを指定する必要はありません。

以下の例では、ディスプレイ装置ユーザーが最高 5 つのユーザー名 (単純リスト) を指定できるパラメーター USER を定義しています。

```
PARM     KWD(USER) TYPE(*NAME) LEN(10) MIN(0) MAX(5) +
         SPCVAL(*ALL) DFT(*ALL)
```

このパラメーターは、MIN(0) が指定されているのでオプションのパラメーターであり、また DFT(*ALL) が指定されているのでデフォルト値は *ALL です。

単純リストの要素をコマンド処理プログラムに渡す場合、その形式は、CL、高水準言語 (HLL)、または REXX のどれを使用しているかによって異なります。

単純リストの ***CL*** コマンド・パラメーターに対する ***CL*** またはその他の ***HLL*** の使用*:*

CL または高水準言語 (HLL) を使用するコマンドを実行する場合、単純リストの要素はこの形式でコマンド処理プログラムに渡されます。

RBAFN509-0

渡される値の数は、長さが 2 文字の 2 進数値で指定されます。この値は、実際に入力された値の数 (つまり、渡される値の数) を示すものであり、指定可能な値の数ではありません。値の受け渡しは、単一のパラメーター値の受け渡しの場合と同様、パラメーターのタイプに応じて行われます。例えば、2 つのユーザー名 (BJONES および TBROWN) が USER パラメーターに指定されている場合は、以下のように渡されます。

RBAFN510-0

ユーザー名は、左寄せにされてブランクが埋め込まれた 10 文字の値として渡されます。

単純リストが渡される場合には、コマンドに指定されている数の要素だけが渡されます。最後に渡された要素の直後の記憶域はリストの一部ではないので、リストの一部として参照してはなりません。したがって、コマンド処理プログラム (CPP) は単純リストを処理する場合に、渡された要素の数に従って処理すべき要素の数を判別します。

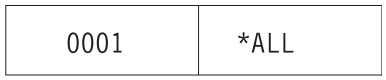
以下の例は、バイナリーの組み込み関数を使用して単純リストを処理する CL プロシージャーを示しています。

```
PGM PARM (...&USER..)
.
.
.
/* Declare space for a simple list of up to five */
/* 10-character values to be received            */
DCL VAR(&USER) TYPE(*CHAR) LEN(52)
.
DCL VAR(&CT)    TYPE(*DEC)  LEN(3 0)
DCL VAR(&USER1) TYPE(*CHAR) LEN(10)
DCL VAR(&USER2) TYPE(*CHAR) LEN(10)
DCL VAR(&USER3) TYPE(*CHAR) LEN(10)
DCL VAR(&USER4) TYPE(*CHAR) LEN(10)
DCL VAR(&USER5) TYPE(*CHAR) LEN(10)
.
.
.
CHGVAR  VAR(&CT) VALUE(%BINARY(&USER 1 2))
.
IF (&CT > 0) THEN(CHGVAR &USER1 %SST(&USER 3 10))
IF (&CT > 1) THEN(CHGVAR &USER2 %SST(&USER 13 10))
IF (&CT > 2) THEN(CHGVAR &USER3 %SST(&USER 23 10))
IF (&CT > 3) THEN(CHGVAR &USER4 %SST(&USER 33 10))
IF (&CT > 4) THEN(CHGVAR &USER5 %SST(&USER 43 10))
IF (&CT > 5) THEN(DO)
/*  If CT is greater than 5, the values passed     */
/*  is greater than the program expects, and error */
/*  logic should be performed                      */
.
.
.
ENDDO
ELSE DO
/* The correct number of values are passed */
/* and the program can continue processing */
.
.
.
ENDDO
ENDPGM
```

図 2. 単純リストの例

同じ手法を使用して、他のリストも CL プロシージャーまたは CL プログラムで処理できます。

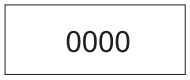
単純リストの場合、リストの代わりに *ALL または *NONE などの単一の値をコマンドに指定できます。単一の値は、個々の値として渡されます。同様に、パラメーターに対して値をまったく指定しなかった場合、デフォルト値が定義されていれば、そのデフォルト値がリストの唯一の値として渡されます。例えば、USER パラメーターでデフォルト値 *ALL を使用した場合には、以下のように渡されます。



RBAFN511-0

*ALL は、左寄せにされてブランクが埋め込まれた 10 文字の値として渡されます。

オプションの単純リスト・パラメーターに対してデフォルト値が指定されていない場合、以下のものが渡されます。



RBAFN512-0

関連タスク:

322 ページの『CL コマンド・パラメーターの定義』
CL コマンド・パラメーターを定義するには、PARM ステートメントを使用しなければなりません。

単純リストの ***CL*** コマンド・パラメーターに対する ***REXX*** の使用*:*

同じコマンドを実行した場合、単純リストの要素は、以下の形式の引数ストリングとして REXX プロシージャーに渡されます。ここで、valueN は、単純リストの最後の値です。

```
. . . USER(value1 value2
. . . valueN) . . .
```

例えば、2 つのユーザー名 (BJONES および TBROWN) が USER パラメーターに指定されている場合は、以下のように渡されます。

```
. . . USER(BJONES
TBROWN) . . .
```

単純リストが渡される場合には、コマンドに指定されている数の要素だけが渡されます。したがって、CPP は単純リストを処理する場合、渡された要素の数を調べて処理すべき要素の数を判別します。

以下の REXX の例は、319 ページの『例: CL コマンドの作成』 の CL プロシージャーと同じ結果を作成します。

```
.
.
.
PARSE ARG . 'USER(' user ')' .
.
.
CT = WORDS(user)
IF CT > 0 THEN user1 = WORD(user,1) else user1 = '
IF CT > 1 THEN user2 = WORD(user,2) else user2 = '
IF CT > 2 THEN user3 = WORD(user,3) else user3 = '
IF CT > 3 THEN user4 = WORD(user,4) else user4 = '
IF CT > 4 THEN user5 = WORD(user,5) else user5 = '
IF CT > 5 THEN
  DO
    /* If CT is greater than 5, the values passed
       is greater than the program expects, and error
       logic should be performed */
.
.
.
  END
ELSE
  DO
    /* The correct number of values are passed
       and the program can continue processing */
  END
EXIT
```

図 *3. REXX* 単純リストの例

REXX プログラムで他のリストを処理する場合にも、これと同様の手法を使用できます。

単純リストの場合、リストの代わりに *ALL または *NONE などの単一の値をコマンドに指定できます。単一の値は、個々の値として渡されます。同様に、パラメーターに対して値をまったく指定しなかった場合、デフォルト値が定義されていれば、そのデフォルト値がリストの唯一の値として渡されます。例えば、USER パラメーターでデフォルト値 *ALL を使用した場合には、以下のように渡されます。

```
. . . USER(*ALL) . . .
```

オプションの単純リスト・パラメーターに対してデフォルト値が指定されていない場合、以下のものが渡されます。

```
. . . USER() . . .
```

関連情報:


REXX/400 Programmer's Guide (PDF)


REXX/400 Reference (PDF)

混合リスト *CL* コマンド・パラメーターの定義*:*

混合リスト CL コマンド・パラメーターは、個別に定義された一連の値を受け入れます。

これらの値は通常、その意味もタイプも異なり、リスト内での位置も固定されています。例えば、LOG(4 0 *SECLVL) は混合リストの指定です。最初の値である 4 は、ログに記録するメッセージ・レベルを識別し、2 番目の値である 0 は、ログに記録するメッセージの重大度の最低レベルです。3 番目の値 *SECLVL は、ログに記録する情報の量 (1 次レベルおよび 2 次レベルのメッセージ) を指定するものです。パラメーターの値が混合リストである場合には、リストの要素は 1 つの要素について 1 つの要素 (ELEM) ステートメントを使用し、個々に定義しなければなりません。

関連の PARM ステートメントの TYPE パラメーターには、リストの最初の ELEM ステートメントを参照するラベルを指定しなければなりません。

```
        PARM    KWD(LOG)    TYPE(LOGLST) ...

LOGLST: ELEM    TYPE(*INT2)      ...
        ELEM    TYPE(*INT2)      ...
        ELEM    TYPE(*CHAR)      LEN(7)
```

最初の ELEM ステートメントは、ラベルを持つことのできる唯一の ELEM ステートメントです。 ELEM ステートメントは、リスト内における要素の順序と同じ順序で指定しなければなりません。

PARM ステートメントの MAX パラメーターの値が 1 より大きく、TYPE パラメーターで ELEM ステートメントが参照されている場合には、定義されるパラメーターはリスト内リストになります。

ELEM ステートメントに指定できるパラメーターは、
TYPE、LEN、CONSTANT、RSTD、DFT、VALUES、REL、RANGE、SPCVAL、SNGVAL、MIN、MAX、LWUNPRT、ALWVAR、PGM、DTAARA、FILE、FULL、EXPR、VARY、PASSATR、CHOICE、CHOICEPGM、および PROMPT です。

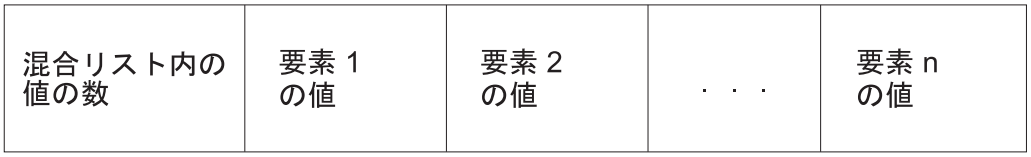
以下の例では、パラメーター CMPVAL を定義しています。ユーザーは、このパラメーターに対して比較値および比較のための開始位置 (混合リスト) を指定できます。

```
      PARM   KWD(CMPVAL) TYPE(CMP) SNGVAL(*ANY) DFT(*ANY) +
               MIN(0)
CMP:  ELEM   TYPE(*CHAR) LEN(80) MIN(1)
      ELEM   TYPE(*DEC) LEN(2 0) RANGE(1 80) DFT(1)
```

混合リストの要素をコマンド処理プログラムに渡す場合、その形式は、CL (または別の高水準言語)、あるいは REXX のいずれを使用しているかによって異なります。

混合リストの ***CL*** コマンド・パラメーターに対する ***CL*** またはその他の ***HLL*** の使用***:***

CL コマンドを実行する場合、混合リストの要素はこの形式でコマンド処理プログラムに渡されます。



RBAFN513-0

混合リスト内の値の数は、長さ 2 の 2 進数値として渡されます。この値が示すのは、常に、混合リストとして定義されている値の数であって、コマンドに実際に入力された値の数ではありません。 SNGVAL パラメーターが入力されたか、またはデフォルト値が渡された場合には、この値は 1 になります。要素に値を指定しなかった場合、デフォルト値が渡されます。要素の受け渡しは 1 つのパラメーター値の受け渡しの場合と同様に、タイプに応じて行われます。例えば、前の例で、ユーザーが CMPVAL パラメーターに比較値 QCMDI を入力し、開始位置の値 (そのデフォルト値は 1) を入力しなかった場合には、以下のものが渡されます。

RBAFN514-0

データ QCMDI は、左寄せにされ、ブランクが埋め込まれた 80 文字の値として渡されます。要素の数は、長さ 2 の 2 進数値として渡されます。

ディスプレイ装置ユーザーが単一の値を入力するか、混合リストのデフォルト値が単一の値であってそれが使用された場合には、その値はリストの最初の要素として渡されます。例えば、ディスプレイ装置ユーザーがパラメーターの単一の値として *ANY を入力した場合には、以下のものが渡されます。

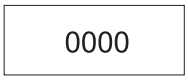
RBAFN515-0

*ANY は、左寄せにされ、ブランクが埋め込まれた 80 文字の値として渡されます。

混合リストは、CL プログラムで処理できます。単純リストの場合とは異なり、混合リストの場合には、リスト内の値の数を判別するために 2 進数をテストする必要はありません。この 2 進数値は、SNGVAL パラメーターがコマンド処理プログラムに渡された場合を除き、混合リストでは常に同じだからです。SNGVAL が渡された場合は、この値は 1 です。パラメーターに単一の値を指定してコマンドが入力された場合には、その単一の値だけが渡されます。 CL プログラムまたは CL プロシージャーで混合リストを処理するには、サブストリング組み込み関数を使用しなければなりません。

混合リストの値の数として 0000 の 2 進数値だけが渡される場合があります。それは、オプションのパラメーターの PARM ステートメントにデフォルト値が定義されておらず、しかもリストの最初の値が必須 (MIN(1)) の場合です。その結果、パラメーター自体は必須でないにもかかわらず、要素を 1 つでも指定す

る場合には、最初の要素は必ず指定しなければなりません。このようなパラメーターに対して値を指定しないでコマンドを入力すると、以下のものが渡されます。



RBAFN512-0

このようなパラメーターの例を次に示します。

```
     PARM  KWD(KWD1)    TYPE(E1) MIN(0)
E1:  ELEM  TYPE(*CHAR)  LEN(10)  MIN(1)
     ELEM  TYPE(*CHAR)  LEN(2)   MIN(0)
```

このパラメーターを CL プログラムまたは CL プロシージャーで処理する場合には、パラメーター値を 14 文字の CL 変数で受け取るようにすることができます。最初の 2 文字は、以下のどちらかと比較できます。

- %SUBSTRING 関数を使用して 16 進数 0000 に初期設定された 2 文字の変数
- %BINARY 組み込み関数を使用した 10 進数 0

関連概念:

178 ページの『CL プログラミング』
CL を使用してプログラムするには、CL プログラミングに固有なプロシージャーおよび概念を理解する必要があります。

関連タスク:

322 ページの『CL コマンド・パラメーターの定義』
CL コマンド・パラメーターを定義するには、PARM ステートメントを使用しなければなりません。

混合リストの *CL* コマンド・パラメーターに対する *REXX* の使用*:*

REXX を使用して CL コマンドを実行すると、混合リスト内の要素は以下の形式でコマンド処理プログラムに渡されます。valueN は混合リスト内の最後の値です。

```
. . . CMPVAL(value1 value2
. . . valueN) . . .
```

要素に値を指定しなかった場合、デフォルト値が渡されます。例えば前の例で、ユーザーが CMPVAL パラメーターに比較値 QCMDI を入力し、開始位置の値 (そのデフォルト値は 1) を入力しなかった場合には、以下のものが渡されます。

```
. . . CMPVAL(QCMDI 1)
. . .
```

末尾ブランクは REXX の値として渡されないことに注意してください。

ディスプレイ装置ユーザーが単一の値を入力するか、混合リストのデフォルト値が単一の値であり、それが使用された場合には、その値はリストの最初の要素として渡されます。例えば、ディスプレイ装置ユーザーがパラメーターの単一の値として *ANY を入力した場合には、以下のものが渡されます。

```
. . . CMPVAL(*ANY) . . .
```

ここでも、末尾ブランクは REXX 値と一緒に渡されないことに注意してください。

オプションのパラメーターの PARM ステートメントにデフォルト値が定義されておらず、しかもリストの最初の値が必須 (MIN(1)) の場合には、そのパラメーター自体は必須ではありません。ただし、いずれかの要素を指定する場合には、最初の要素は必須です。このようなパラメーターに対して値を指定しないでコマンドを入力すると、以下のものが渡されます。

```
. . . CMPVAL() . . .
```

CL コマンド・パラメーターのリスト内のリストの定義*:*

リスト内リストは、1 つの CL コマンド・パラメーターに複数回指定可能なリストか (単純リストまたは混合リスト)、混合リスト内のある値に指定可能なリストのいずれかです。

次に示すのは、リスト内リストの例です。

```
STMT((START RESPND) (ADDDSP CONFRM))
```

外側の括弧は、このパラメーターに対して指定できるリスト (外部リスト) を囲んでおり、内側の 2 組の括弧はリスト内リスト (内部リスト) を囲んでいます。

以下の例では、単純リストの中で混合リストを定義しています。混合リストが指定され、PARM ステートメントの MAX パラメーターには 1 より大きい値が入っています。したがって、混合リストは、MAX パラメーターで指定している回数まで指定できます。

```
       PARM KWD(PARM1) TYPE(LIST1) MAX(5)
LIST1: ELEM TYPE(*CHAR) LEN(10)
       ELEM TYPE(*DEC) LEN(3 0)
```

この例では、2 つの要素を最高 5 回まで指定できます。このパラメーターに対する値の指定は次のようになります。

```
PARM1((VAL1 1.0) (VAR2 2.0) (VAR3 3.0))
```

以下の例では、単純リストが混合リストの値として指定されています。この例では、ELEM ステートメントの MAX パラメーターの値が 1 より大きいので、要素は、MAX パラメーターで指定した回数まで、繰り返し指定できます。

```
       PARM KWD(PARM2) TYPE(LIST2)
LIST2: ELEM TYPE(*CHAR) LEN(10) MAX(5)
       ELEM TYPE(*DEC) LEN(3 0)
```

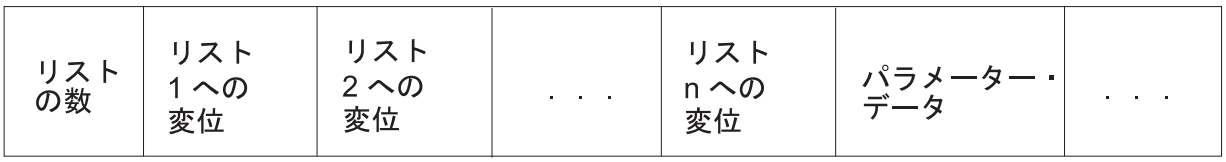
この例の場合、最初の要素は最高 5 回まで指定できますが、2 番目の要素は一度だけしか指定できません。このパラメーターに対する値の指定は次のようになります。

```
PARM2((NAME1 NAME2 NAME3) 123.0)
```

リスト内のリストをコマンド処理プログラムに渡す場合、その形式は、CL (または別の高水準言語)、あるいは REXX のいずれを使用しているかによって異なります。

リスト内のリストに対する ***CL*** またはその他の ***HLL*** の使用*:*

CL コマンドを実行すると、リスト内リストであるコマンド・パラメーターはこの形式でコマンド処理プログラムに渡されます。



RBAFN534-0

リストの数は、長さ 2 の 2 進数値として渡されます。リストの数に続いて、各リストへの変位を示す値が渡されます (リストに入力された値ではありません)。各変位は、LISTDSPL パラメーターの値に従って、長さ 4 または長さ 2 の 2 進数値として渡されます。

以下の例は、単純リスト内の混合リストであるパラメーター KWD2 の定義、ディスプレイ装置ユーザーによるパラメーターの指定、および渡される内容を示しています。パラメーターの定義は以下のとおりです。

```
      PARM    KWD(KWD2)    TYPE(LIST) MAX(20) MIN(0) +
              DFT(*NONE)  SNGVAL(*NONE)  LISTDSPL(*INT2)
LIST: ELEM    TYPE(*CHAR)  LEN(10) MIN(1)       /*From value*/
      ELEM    TYPE(*CHAR)  LEN(5) MIN(0)        /*To value*/
```

ディスプレイ装置ユーザーは、KWD2 パラメーターを次のように入力します。

```
KWD2((A B))
```

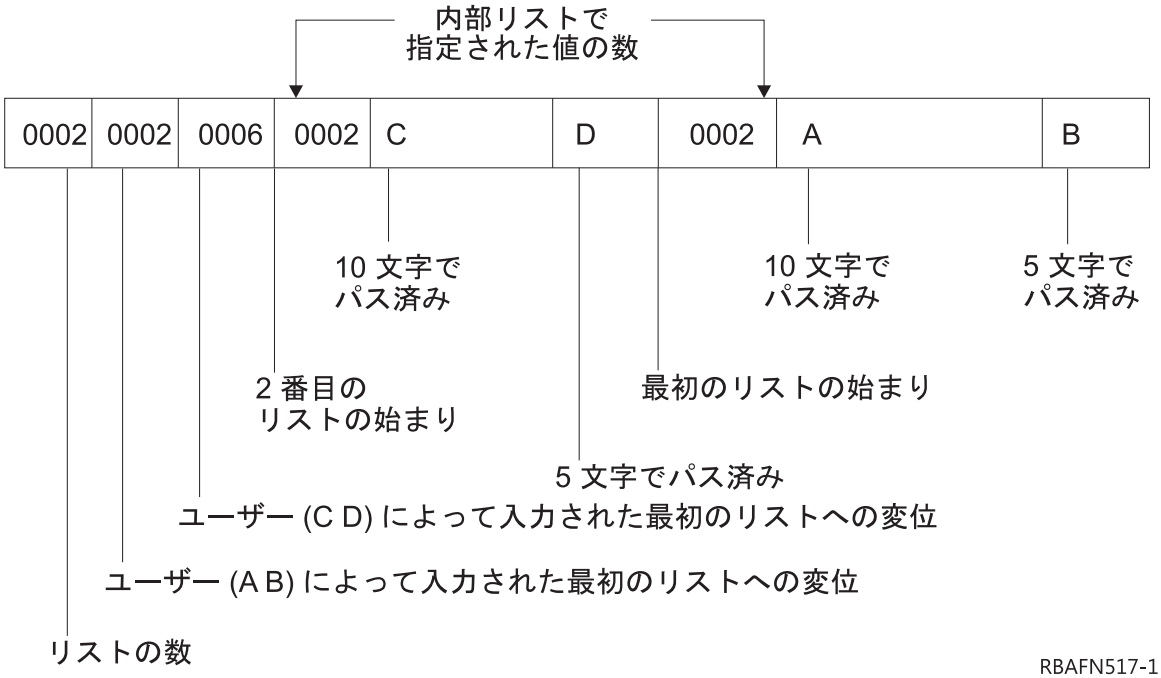
以下のものがコマンド処理プログラムに渡されます。

RBAFN516-0

ディスプレイ装置ユーザーが、上記の代わりに以下のものを入力したとします。

```
KWD2((A B) (C D))
```

以下のものがコマンド処理プログラムに渡されます。



リスト内リストは、n (ディスプレイ装置ユーザーが最後に入力したリスト) から 1 (ディスプレイ装置が最初に入力したリスト) への順序で、コマンド処理プログラムに渡されます。ただし、変位の値は 1 から n への順序で渡されます。

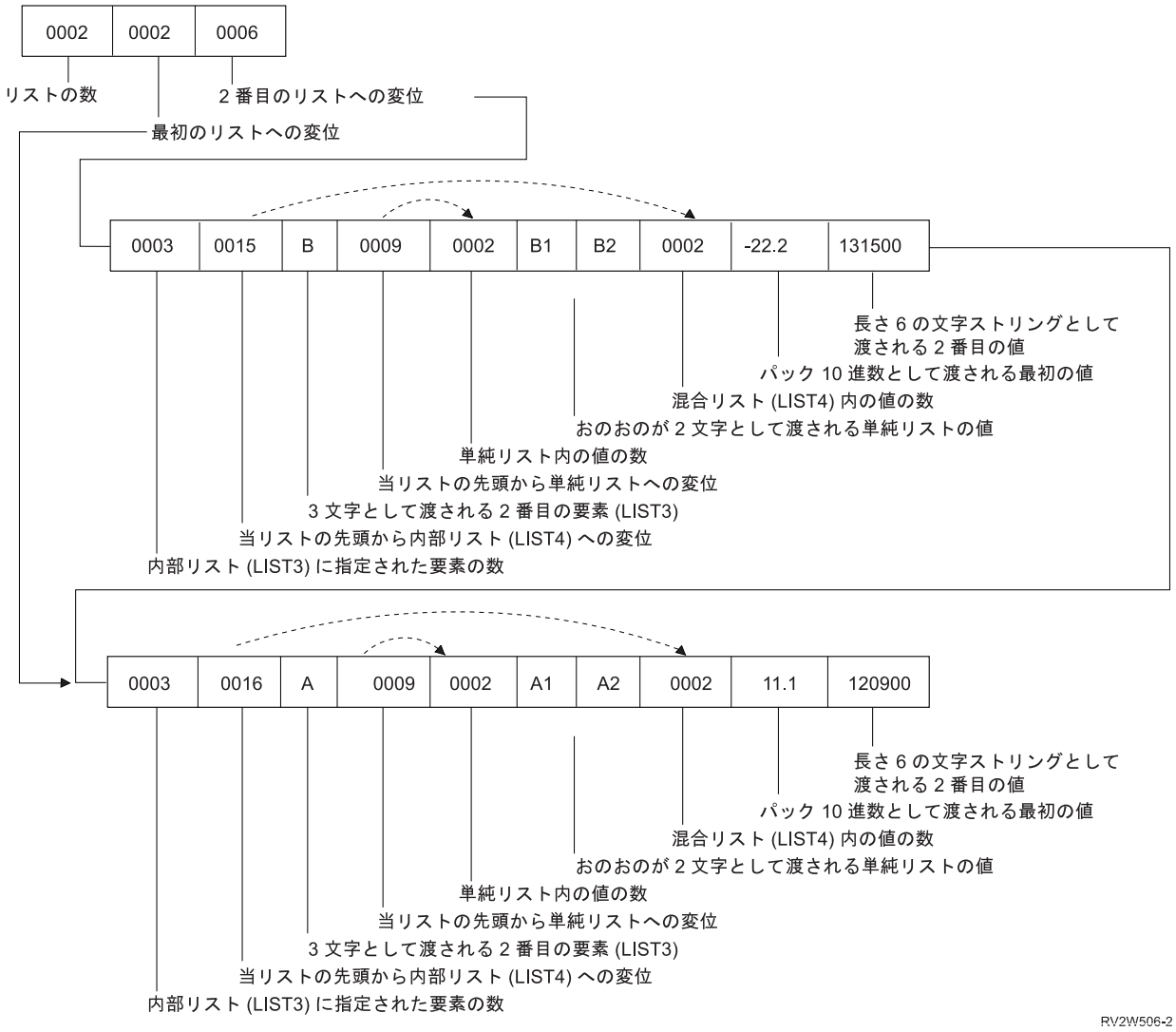
以下の例は、リスト内リストのさらに複雑な使用例です。パラメーターの定義は以下のとおりです。

```
        PARM    KWD(PARM1)  TYPE(LIST3)  MAX(25)
LIST3:  ELEM    TYPE(LIST4)
        ELEM    TYPE(*CHAR)  LEN(3)
        ELEM    TYPE(*NAME)  LEN(2)  MAX(5)
LIST4:  ELEM    TYPE(*DEC) LEN(7 2)
        ELEM    TYPE(*TIME)
```

ディスプレイ装置ユーザーが、以下の例に示しているように PARM1 パラメーターを入力したとします。

```
PARM1(((11.1  120900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

以下のものがコマンド処理プログラムに渡されます。



リスト内リストに対する ***REXX*** の使用*:*

REXX を使用する CL コマンドを実行した場合、リスト内リストであるコマンド・パラメーターは、パラメーターに対して値を入力した場合と同じ方法でコマンド処理プログラムへ渡されます。末尾ブランクは渡されません。

以下の例は、単純リスト内の混合リストであるパラメーター KWD2 の定義、ディスプレイ装置ユーザーによるパラメーターの指定、および渡される内容を示しています。パラメーターの定義は以下のとおりです。

```
      PARM    KWD(KWD2)    TYPE(LIST) MAX(20) MIN(0) +
              DFT(*NONE)  SNGVAL(*NONE)
LIST:  ELEM   TYPE(*CHAR)  LEN(10) MIN(1)       /*From value*/
       ELEM   TYPE(*CHAR)  LEN(5) MIN(0)        /*To value*/
```

ディスプレイ装置ユーザーは、KWD2 パラメーターを次のように入力します。

```
KWD2((A B))
```

以下のものがコマンド処理プログラムに渡されます。

```
KWD2(A B)
```

ディスプレイ装置ユーザーが、上記の代わりに以下のものを入力したとします。

```
KWD2((A B) (C D))
```

以下のものがコマンド処理プログラムに渡されます。

```
KWD2((A B) (C D))
```

以下の例は、リスト内リストのさらに複雑な使用例です。パラメーターの定義は以下のとおりです。

```
       PARM    KWD(PARM1)  TYPE(LIST3)  MAX(25)
LIST3: ELEM    TYPE(LIST4)
       ELEM    TYPE(*CHAR)  LEN(3)
       ELEM    TYPE(*NAME)  LEN(2)  MAX(5)
LIST4: ELEM    TYPE(*DEC) LEN(7 2)
       ELEM    TYPE(*TIME)
```

ディスプレイ装置ユーザーは、PARM1 パラメーターを次のように入力します。

```
PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

以下のものがコマンド処理プログラムに渡されます。

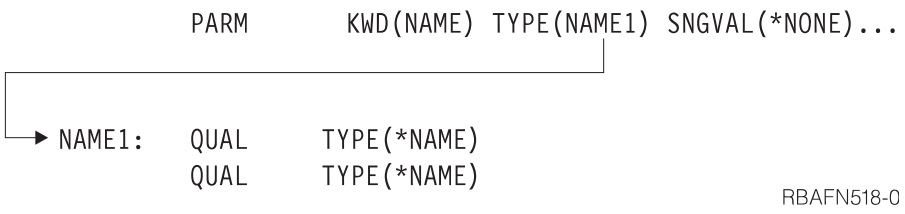
```
PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

修飾名 ***CL*** コマンド・パラメーターの定義**:**

修飾名とは、オブジェクト名の前に、そのオブジェクトが保管されているライブラリーの名前を付けたものです。

パラメーターの値またはリストの項目が修飾名である場合には、修飾子 (QUAL) ステートメントを使用してその名前を別個に定義しなければなりません。修飾名を構成するそれぞれの部分について、QUAL ステートメントを 1 つずつ使用して定義しなければなりません。また、修飾名の各部分は、修飾名における順序どおりに記述しなければなりません。最初の QUAL ステートメントには *NAME または *GENERIC を指定しなければなりません。関連の PARM ステートメントまたは ELEM ステートメントでは、修飾名の最初の QUAL ステートメントを参照するラベルを識別しなければなりません。

以下のコマンド定義ステートメントは、最も一般的な修飾名を定義するためのものです。修飾オブジェクト名は、オブジェクトが入っているライブラリーの名前の後にオブジェクト自体の名前を付けたものです。QUAL ステートメントは、修飾名において名前が指定される順序と同じ順序で指定しなければなりません。



RBAFN518-0

QUAL ステートメントに指定できるパラメーターの多くは、PARM ステートメントの場合と同じです。ただし、TYPE パラメーターには以下の値だけを指定できます。

- *NAME
- *GENERIC
- *CHAR
- *INT2

- *INT4

修飾名をコマンド処理プログラムに渡す場合、その形式は、CL (または別の高水準言語)、あるいは REXX のいずれを使用しているかによって異なります。

関連タスク:

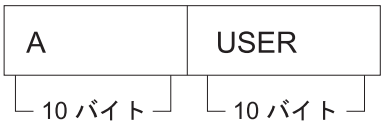
322 ページの『CL コマンド・パラメーターの定義』
CL コマンド・パラメーターを定義するには、PARM ステートメントを使用しなければなりません。

修飾名の ***CL*** コマンド・パラメーターに対する ***CL*** またはその他の ***HLL*** の使用*:*

CL またはその他の高水準言語 (HLL) を使用している場合、修飾名はこの形式でコマンド処理プログラムに渡されます。

RBAFN519-0

例えば、前ページで定義した QUAL ステートメントに対してディスプレイ装置ユーザーが NAME(USER/A) を入力した場合、この名前は以下の形式でコマンド処理プログラムに渡されます。



RBAFN520-0

修飾子はそのタイプと長さに基づき、連続してコマンド処理プログラムに渡されますが、渡す方法は単一パラメーター値の場合と同じです (322 ページの『CL コマンド・パラメーターの定義』を参照)。 区切り文字 (/) は渡されません。これは 1 つのパラメーター、混合リストの要素、または単純リストのどの形式の修飾名の受け渡しの場合にも当てはまります。

ディスプレイ装置ユーザーが修飾名に対して単一の値を入力した場合には、渡される値の長さは、修飾名の各部分の長さを合計した長さとなります。例えば、それぞれ長さ 10 の 2 つの値を用いて修飾名を定義した場合に、ディスプレイ装置ユーザーが単一の値を入力すると、その単一の値は左寄せされ、全体が 20 文字になるように右側にブランクが埋め込まれた上で渡されます。ディスプレイ装置ユーザーが単一値として *NONE を入力すると、以下の 20 文字の値が渡されます。

RBAFN521-0

CL 変数 &OBJ および &LIB は、CL コマンド処理プログラムに渡される修飾名パラメーターの 2 つの部分に対して定義されます。

```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
DCL &OBJ TYPE(*CHAR) LEN(10) STG(*DEFINED) DEFVAR(&QLFDNAM 1)
DCL &LIB TYPE(*CHAR) LEN(10) STG(*DEFINED) DEFVAR(&QLFDNAM 11)
```

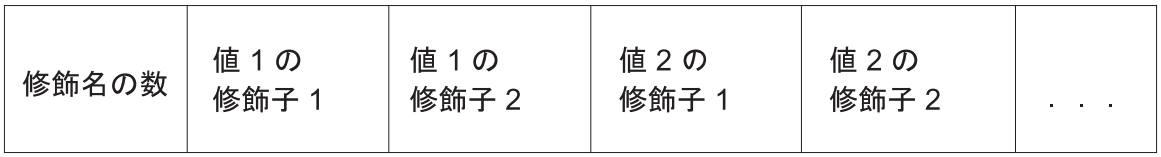
```
 .
 .
 .
ENDPGM
```

ユーザーは次に、適切な CL 構文でこの修飾名を指定できます。例えば、OBJ(&LIB/&OBJ) を指定します。

また、以下の方法を使用して、修飾名を 2 つの値に分けることもできます。

```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
CHKOBJ (%SST(&QLFDNAM 11 10)/%SST(&QLFDNAM 1 10)) *PGM
 .
 .
 .
ENDPGM
```

修飾名の単純リストは、以下の形式でコマンド処理プログラムに渡されます。



RBAFN522-0

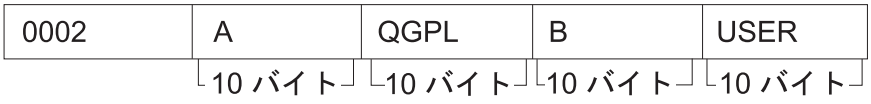
例えば、以下の例の NAME パラメーターの PARM ステートメントに、次のように MAX(3) を指定したとします。

```
        PARM   KWD(NAME) TYPE(NAME1) SNGVAL(*NONE) MAX(3)
NAME1:  QUAL   TYPE(*NAME)
        QUAL   TYPE(*NAME)
```

そして、ディスプレイ装置ユーザーが次のように入力したとします。

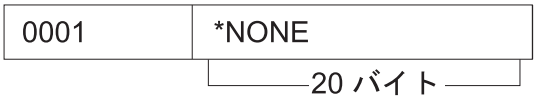
```
NAME(QGPL/A USER/B)
```

NAME パラメーターは、以下のようにコマンド処理プログラムに渡されます。



RBAFN523-0

ディスプレイ装置ユーザーが単一値 NAME(*NONE) を入力した場合には、パラメーターは以下の形で渡されます。



RBAFN524-0

修飾名の ***CL*** コマンド・パラメーターに対する ***REXX*** の使用*:*

REXX を使用するコマンドを実行した場合、修飾名がコマンド処理プログラムへ渡される方法は、パラメーターに対して値を入力した場合と同じです。末尾ブランクは渡されません。

例えば、ディスプレイ装置ユーザーが、この項で前に定義した QUAL ステートメントに対して以下のものを入力したとします。

```
NAME(USER/A)
```

この場合、修飾名は以下の形式でコマンド処理プログラムに渡されます。

```
NAME(USER/A)
```

修飾子はそのタイプと長さに基づき、連続してコマンド処理プログラムに渡されますが、渡す方法は単一パラメーター値の場合と同じです。

ディスプレイ装置ユーザーが単一値として *NONE を入力すると、以下の 20 文字の値が渡されます。

```
NAME(*NONE)
```

以下の例は、ディスプレイ装置ユーザーによる修飾名の単純リストの入力を示しています。

```
NAME(QGPL/A USER/B)
```

REXX を使用した場合、このパラメーターはコマンド処理プログラムに以下のように渡されます。

```
NAME(QGPL/A USER/B)
```

関連タスク:

322 ページの『CL コマンド・パラメーターの定義』

CL コマンド・パラメーターを定義するには、PARM ステートメントを使用しなければなりません。

CL **コマンド・パラメーター間の従属関係の定義*:***

従属関係とは、CL コマンドのパラメーター間に存在する必要のある関係のことです。

パラメーター相互間に特定の関係が存在していなければならず、コマンドの実行時にパラメーター値の検査が必要な場合には、従属 (DEP) ステートメントを使用してその関係を定義してください。 DEP ステートメントを使用することにより、以下の機能が実行可能になります。

- PARM パラメーターに定義されたパラメーター相互間の関係が真であるかをテストする前に、その前提条件として真にならなければならない制御条件を指定する (CTL)。
- CTL により定義された制御条件が真である場合にテストを必要とするパラメーター相互間の関係を指定する (PARM)。
- 関連する PARM ステートメントに定義されたパラメーター相互間の関係のうちで、制御条件が真である場合に真でなければならない関係の数を指定する (NBRTRUE)。
- パラメーターの従属条件が満たされていない場合にシステムがディスプレイ装置ユーザーに送る、メッセージ・ファイル内のエラー・メッセージのメッセージ識別コードを指定する。

以下の例では、ディスプレイ装置ユーザーが TYPE(LIST) パラメーターを指定した場合には ELEMLIST パラメーターも同時に指定しなければなりません。

```
DEP CTL(&TYPE *EQ LIST) PARM(ELEMLIST)
```

以下の例では、パラメーター &WRITER が常にパラメーター &NEWWTR と異なっている必要があります。この条件が真でなければ、メッセージ USR0001 がディスプレイ装置ユーザーに対して送られます。

```
DEP CTL(*ALWAYS) PARM((&WRITER *NE &NEWWTR)) MSGID(USR0001)
```

以下の例で、ディスプレイ装置ユーザーが FILE パラメーターを指定した場合には、VOL および LABEL の各パラメーターも同時に指定しなければなりません。

```
DEP CTL(FILE) PARM(VOL LABEL) NBRTRUE(*EQ 2)
```

CL コマンド・パラメーターにおいて選択可能な項目と指定可能な値*:*

CL コマンド・パラメーターの定義において選択可能な項目と指定可能な値を判別するには、この情報を参照してください。

プロンプターは、パラメーターに対する値として選択可能な項目をプロンプト画面の該当入力フィールドの右側に表示します。表示するテキストは、自動的に作成することも、コマンド定義ソースに指定することも、あるいは出口プログラム (出口) により動的に作成することもできます。選択可能な項目を記述するテキストは PARM ステートメント、ELEM ステートメント、または QUAL ステートメントについて定義できます。ただし、画面様式に制約があるため、テキストが表示されるのはフィールド長が 12 以下の値の場合で、グループ内の最初の修飾子を除き他のすべての修飾子についてはフィールド長が 10 以下の値の場合だけです。

選択可能な項目のテキストは、CHOICE パラメーターで定義します。このパラメーターのデフォルト値は *VALUES で、これは TYPE、 RANGE、 VALUES、 SPCVAL、および SNGVAL の各キーワードに指定された値に基づいてテキストを自動的に作成することを示します。テキストは最大 30 文字です。値が多すぎてこのサイズに収まらない場合には、値がまだ存在することを示すために、その末尾に省略符号 (...) が付加されます。

選択可能な項目を表示しないこと (*NONE) を指定することもできます。また、表示するテキスト・ストリングを指定することも、**CRTCMD** コマンドの PMTFILE パラメーターに指定されたメッセージ・ファイルから取り出すテキスト・メッセージの識別コードを指定することもできます。

また、選択可能な項目のテキストを得るために、プロンプトの際に、出口プログラムを実行することも指定できます。これは、例えば現在システムに存在しているオブジェクトのリストをユーザーに表示したい場合などに使用します。同じ出口プログラムを使用して、パラメーター値の指定画面に表示する値のリストを用意できます。出口プログラムを指定したい場合には、PARM ステートメント、ELEM ステートメント、または QUAL ステートメントの CHOICE パラメーターに *PGM を指定し、CHOICEPGM パラメーターに出口プログラムの修飾名を指定してください。

出口プログラムは、以下の 2 つのパラメーターを受け入れなければなりません。

- パラメーター **1**: プロンプターにより選択プログラムに渡され、以下の値を含む 21 バイトのフィールド。

桁	説明
1-10	コマンド名。プログラムを実行させる、処理中のコマンドの名前を指定します。
11-20	キーワード名。選択可能な項目のテキストまたは指定可能な値が要求されているキーワード名を指定します。
21	プロンプターにより要求されているデータのタイプを示す文字 C または P。英字 C は、選択可能な項目のテキストを返す 30 バイトのフィールドであることを示します。英字 P は、指定可能な値のリストを返す 10240 バイトのフィールドであることを示します。

- パラメーター **2**: 次のどちらかを返すための 30 バイトまたは 10240 バイトのフィールド。
  - 最初のパラメーターのバイト 21 が C の場合、選択可能な項目のテキストを返すことを示します。これは、プロンプト画面で入力フィールドの右にプログラムがそのテキストを入れる 30 バイトのフィールドです。
  - 最初のパラメーターのバイト 21 が P の場合 (指定可能な値のリストを返すことを示す)、これはプログラムがそのリストを入れる 10240 バイトのフィールドです。リストの最初の 2 バイトには、リ

スト中の項目の数が (2 進数で) 入っていなければなりません。この値の後に項目が続きます。各項目は、2 バイトの 2 進数で長さを表すフィールドと、その後に 1 から 34 バイトの長さの値が続きます。

最初の 2 バイトに 2 進数ゼロの値が返された場合には、指定可能な値は表示されません。

最初の 2 バイトに 2 進数の負の値が返された場合には、指定可能な値のリストはコマンドから取られます。

プログラムの呼び出し時に何らかの例外が起こった場合には、選択可能な項目のテキストはブランクのままで、指定可能な値のリストはコマンドから取られます。

CL コマンド・パラメーターのプロンプト制御の指定:

プロンプト制御の指定を用いることにより、プロンプトの過程でコマンドのどのパラメーターを表示するかを制御できます。

この制御を用いることにより、表示が必要なパラメーターだけを表示できるので、プロンプトを簡素化できます。

パラメーターが他のパラメーターに指定された値によって表示されるようにできます。この指定は、他のパラメーター (制御パラメーターと呼ぶ) に特定の値が指定された場合にだけ意味を持つパラメーターに対して使用すると便利です。

また、プロンプトの過程でユーザーが機能キーを押して追加のパラメーターの表示を要求した場合にだけ、特定のパラメーターが選択されるように指定することもできます。通常はデフォルト値が使用されるか、または使用頻度の低い機能の制御に使用され、ユーザーによって指定されることがまれなパラメーターに対してこの指定を使用できます。

プロンプト制御が指定されているコマンドに対して、すべてのパラメーターを表示したい場合には、プロンプトの過程で F9 キーを押すことにより、すべてのパラメーターの表示を要求できます。

関連概念:

41 ページの『CL コマンドの区切り文字』
コマンド区切り文字は、コマンドの中で、文字のグループの始まりと終わりを示す特殊文字またはスペースです。

45 ページの『CL コマンド定義』
コマンド定義機能を使用することにより、ユーザーはアプリケーションの特殊な要件に対応する新たな制御言語コマンドを作成することができます。ユーザー定義のコマンドも、使用法はシステム・コマンドと同様です。

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

***CL* コマンド・パラメーターの条件付きプロンプトの指定:**

いくつかの条件を満たした場合に限りプロンプトを出してパラメーターを要求するよう指定することができます。

コマンドのプロンプトの時点で、他のパラメーターにより条件付けられているパラメーターは以下の場合に表示されます。

- そのパラメーターが、制御パラメーターに指定された値に基づき選択された場合。
- 制御パラメーターに指定された値がエラーである場合。
- 条件付きパラメーターに値が指定されていた場合。
- プロンプトの過程で機能キーが押され、パラメーターのすべての表示が要求された場合。

条件付きのパラメーターに対してプロンプトが出され、その制御パラメーターにまだ値が指定されていない場合には、それまでに選択されていたすべてのパラメーターが表示されます。ユーザーが実行キーを押すと制御パラメーターがテストされ、条件付きパラメーターを表示すべきかどうかが判別されます。

条件付きプロンプトをコマンド定義ソースで指定するには、他のパラメーターに条件付けられた各パラメーターの PARM ステートメントの PMTCTL パラメーターにラベル名を指定してください。 指定するラベルは、制御パラメーターとプロンプトの対象としてパラメーターを選択するためのテスト条件を指定した PMTCTL ステートメントに定義されているラベルでなければなりません。複数の PARM ステートメントに同じラベルを指定できます。

PMTCTL ステートメントには、制御パラメーターの名前、1 つまたは複数のテスト条件、および条件付きパラメーターをプロンプトの対象として選択するには真でなければならないテスト条件の数を指定してください。制御パラメーターが特殊値のマッピングを持つものである場合には、PMTCTL ステートメントに入力する値は受け取り置き換え値でなければなりません。制御パラメーターがリストまたは修飾名である場合には、最初のリスト要素または修飾子だけが比較されます。

以下の例では、パラメーター OUTFILE および OUTMBR は、OUTPUT パラメーターに *OUTFILE が指定されている場合に限り選択され、パラメーター OUTQ は OUTPUT パラメーターに *PRINT が指定されている場合に限り選択されます。

```
     PARM OUTPUT TYPE(*CHAR) LEN(1) DFT(*) RSTD(*YES) +
           SPCVAL((*) (*PRINT P) (*OUTFILE F))
     PARM OUTFILE TYPE(Q1) PMTCTL(OUTFILE)
     PARM OUTMBR TYPE(*NAME) LEN(10) PMTCTL(OUTFILE)
     PARM OUTLINK TYPE(*CHAR) LEN(10)
     PARM OUTQ TYPE(Q1) PMTCTL(PRINT)
     Q1: QUAL TYPE(*NAME) LEN(10)
         QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
OUTFILE: PMTCTL CTL(OUTPUT) COND((*EQ F)) NBRTRUE(*EQ 1)
PRINT:   PMTCTL CTL(OUTPUT) COND((*EQ P)) NBRTRUE(*EQ 1)
```

この前の例で、OUTMBR パラメーターの条件のテストが終わった後に、OUTLINK パラメーターのプロンプトが表示されます。場合によっては、OUTMBR パラメーターのテストの前に OUTLINK パラメーターのプロンプトが必要なことがあります。プロンプトの順序を変えるには、コマンド定義のソースの中でパラメーターの順序を変えるか、あるいは OUTLINK パラメーターの PARM ステートメントに PROMPT キーワードを使用してください。

ラベルを使用して、PMTCTL ステートメントのグループを参照できます。これにより、1 つのパラメーターの条件付けに複数の制御パラメーターを使用できます。 PMTCTL ステートメントのグループを指定するには、グループの最初のステートメントにラベルを指定します。グループ内の PMTCTL ステートメントの間に他のステートメントを入れることはできません。

グループ内のステートメント相互間の論理関係を指定するには、LGLREL パラメーターを使用します。グループの最初の PMTCTL ステートメントには、LGLREL パラメーターを指定することはできません。後続の PMTCTL ステートメントについては、LGLREL パラメーターはその前の 1 つまたは複数の PMTCTL ステートメントに対する論理関係 (*AND または *OR) を指定します。グループ内のステートメントは、*AND 関係および *OR 関係を任意に組み合わせて論理的に関連付けることができます (*AND 関係が最初に検査され、次に *OR 関係が検査されます)。

以下の例は、論理関係を使用して複数の PMTCTL ステートメントをグループ化する方法を示しています。この例では、以下の条件のいずれか 1 つが存在していれば、パラメーター P3 が選択されます。

- P1 に対して *ALL が指定されている。
- P1 に対して *SOME が指定され、P2 に対して *ALL が指定されている。
- P1 に対して *NONE が指定され、P2 に対して *ALL が指定されていない。

```
PARM P1 TYPE(*CHAR) LEN(5) RSTD(*YES) VALUES(*ALL *SOME *NONE)
PARM P2 TYPE(*NAME) LEN(10) SPCVAL(*ALL)
PARM P3 TYPE(*CHAR) LEN(10) PMTCTL(PMTCTL1)
PMTCTL1:PMTCTL CTL(P1) COND((*EQ *ALL))
        PMTCTL CTL(P1) COND((*EQ *SOME)) LGLREL(*OR)
        PMTCTL CTL(P2) COND((*EQ *ALL)) LGLREL(*AND)
        PMTCTL CTL(P1) COND((*EQ *NONE)) LGLREL(*OR)
        PMTCTL CTL(P2) COND((*NE *ALL)) LGLREL(*AND)
```

制御パラメーターをテストする前にその制御パラメーターに対して何らかの処理を行いたい場合は、出口プログラムを指定できます。この出口プログラムを使用して、以下の内容に基づいてプロンプトの条件付けを行うことができます。

- オブジェクトのタイプまたはその他の属性
- 最初のリスト要素または修飾子以外の他のリスト要素または修飾子
- リスト全体または修飾名

出口プログラムを指定するには、該当の制御パラメーターに対して PARM ステートメントの PMTCTLPGM パラメーターに、プログラムの修飾名を指定してください。プロンプト表示中のパラメーターの検査時に、出口プログラムが実行されます。 PMTCTL ステートメントの条件は、制御パラメーターに対して指定された値とではなく、出口プログラムから戻された値と比較されます。

出口プログラムが見つからなかった場合、または正しく実行されなかった場合は、システムは戻り値を使用する条件が真であると想定します。

出口プログラムは、以下の 3 つのパラメーターを受け入れるように作成しなければなりません。

- 20 文字のフィールド。プロンプターは、前半の 10 文字にコマンドの名前、後半の 10 文字に制御パラメーターの名前を渡します。このフィールドは変更してはなりません。
- 制御パラメーターの値。このフィールドは、コマンド処理プログラムに渡されるときと同じ形式で、変更してはなりません。制御パラメーターが VARY(*YES) として定義されている場合、値の前には長さ値が付きます。制御パラメーターが PASSATR(*YES) の場合には、属性バイトが含まれます。
- 32 文字のフィールド。出口プログラムはこのフィールドに、PMTCTL ステートメントでテストする値を入れます。

  PMTCTL ステートメントでテストされる値は、宣言されたデータ・タイプと同じ形式で戻されます。

以下の例で OBJ は修飾名であり、コマンド、プログラム、またはファイルの名前です。出口プログラムはそのオブジェクト・タイプを判別し、タイプを変数 &RTNVAL に戻します。

```
CMD
PARM OBJ TYPE(Q1) PMTCTLPGM(CNVTYPE)
Q1: QUAL TYPE(*NAME) LEN(10)
    QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
PARM CMDPARM TYPE(*CHAR) LEN(10) PMTCTL(CMD)
PARM PGMPARM TYPE(*CHAR) LEN(10) PMTCTL(PGM)
PARM FILEPARM TYPE(*CHAR) LEN(10) PMTCTL(FILE)
CMD: PMTCTL CTL(OBJ) COND((*EQ *CMD) (*EQ *)) NBRTRUE(*EQ 1)
PGM: PMTCTL CTL(OBJ) COND((*EQ *PGM) (*EQ *)) NBRTRUE(*EQ 1)
FILE: PMTCTL CTL(OBJ) COND((*EQ *FILE) (*EQ *)) NBRTRUE(*EQ 1)
```

出口プログラムのソース・ステートメントは以下のとおりです。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM PARM(&CMD &PARMVAL &RTNVAL)
DCL &CMD *CHAR 20                /* Command and parameter name      */
DCL &PARMVAL *CHAR 20            /* Parameter value                 */
DCL &RTNVAL *CHAR 32             /* Return value                    */
DCL &OBJNAM *CHAR 10             /* Object name                     */
DCL &OBJLIB *CHAR 10             /* Object type                     */
CHGVAR &OBJNAM %SST(&PARMVAL 1 10)
CHGVAR &OBJLIB %SST(&PARMVAL 11 10)
CHGVAR &RTNVAL '*'               /* Initialize return value to error*/
CHKOBJ &OBJLIB/&OBJNAM *CMD      /* See if command exists           */
MONMSG CPF9801 EXEC(GOTO NOTCMD) /* Skip if no command              */
CHGVAR &RTNVAL '*CMD'            /* Indicate object is a command    */
RETURN                           /* Exit                            */
NOTCMD:
CHKOBJ &OBJLIB/&OBJNAM *PGM      /* See if program exists           */
MONMSG CPF9801 EXEC(GOTO NOTPGM) /* Skip if no program              */
CHGVAR &RTNVAL '*PGM'            /* Indicate object is a program    */
RETURN                           /* Exit                            */
NOTPGM:
CHKOBJ &OBJLIB/&OBJNAM *FILE     /* See if file exists              */
MONMSG CPF9801 EXEC(RETURN)      /* Exit if no file                 */
CHGVAR &RTNVAL '*FILE'           /* Indicate object is a file       */
ENDPGM
```

CL コマンドのプロンプト時に追加または拡張パラメーターを非表示にする*:*

ユーザーがプロンプトの過程で機能キーを押して追加のパラメーターを要求しない限り、使用頻度の低いパラメーターについてのプロンプトが表示されないようにするには、そのパラメーターに対する PARM ステートメントに PMTCTL(*PMTRQS) を指定します。

コマンドのプロンプトが表示される場合に、PMTCTL(*PMTRQS) の指定を持つパラメーターはそれに値が指定されているか、あるいはユーザーが F10 キーを押して追加のパラメーターを要求しない限りそのプロンプトは表示されません。

プロンプターは他のパラメーターと区別するため、PMTCTL(*PMTRQS) の指定を持つパラメーターの前には区切り線を表示します。デフォルト値により、 PMTCTL(*PMTRQS) が指定されたパラメーターについてのプロンプトは、コマンド定義ソースに定義されている順序にかかわらず、すべて最後に表示されます。これは、PROMPT キーワードに相対プロンプト番号を指定することによって変更できます。ただし、この変更を行うと、F10 キーを押した場合に追加されたのがどのパラメーターなのかを見分けにくくなります。

関連タスク:

『CL コマンドのキー・パラメーターおよびプロンプト一時変更プログラム』
プロンプト一時変更プログラムを使用することによって、コマンドのプロンプトの表示時点で、デフォルト値ではなく現在の値を表示できます。キー・パラメーターは、例えばオブジェクトの名前などのようにオブジェクトを固有なものとして識別するパラメーターです。

CL コマンドのキー・パラメーターおよびプロンプト一時変更プログラム*:*

プロンプト一時変更プログラムを使用することによって、コマンドのプロンプトの表示時点で、デフォルト値ではなく現在の値を表示できます。キー・パラメーターは、例えばオブジェクトの名前などのようにオブジェクトを固有なものとして識別するパラメーターです。

コマンドにプロンプト一時変更プログラムが定義されている場合、プロンプト一時変更プログラムの呼び出しの結果は以下の 2 とおりの方法で表示できます。

- パラメーターを指定せずにコマンド名だけを任意のコマンド行に入力して、F4 キー (プロンプト) を押したとします。次の画面に、そのコマンドのキー・パラメーターが表示されます。

  表示されたすべてのフィールドに入力し、実行キーを押してください。次の画面にそのコマンドのすべてのパラメーターが表示され、キー・パラメーターのフィールド以外のパラメーター・フィールドにはデフォルト値 (*SAME や *PRV など) ではなく、現在の値が表示されます。

  例えば、コマンド行に CHGLIB を入力して F4 キー (プロンプト) を押すと、ライブラリー・パラメーターだけが表示されます。次に *CURLIB を入力して実行キーを押すと、現行ライブラリーの現在の値が表示されます。
- コマンドの名前とすべてのキー・パラメーターの値を任意のコマンド行に入力し、そして F4 キー (プロンプト) を押したとします。次の画面にそのコマンドのすべてのパラメーターが表示され、キー・パラメーターのフィールド以外のパラメーター・フィールドにはデフォルト値 (*SAME や *PRV など) ではなく、現在の値が表示されます。

  例えば、コマンド行に CHGLIB LIB(*CURLIB) を入力して F4 キー (プロンプト) を押すと、ユーザーの現行ライブラリーの現在の値が表示されます。

F10 キー (追加のパラメーター) を押すと、PMTCTL(*PMTRQS) が定義されているすべてのパラメーターが現在の値とともに表示されます。

コマンド・プロンプトを終了するには、F3 キー (終了) を押してください。

関連概念:

318 ページの『CL コマンドのプロンプト一時変更プログラム』
プロンプト一時変更プログラムを作成して、コマンドのプロンプトでパラメーターのデフォルト値ではなく現在の値を表示できます。

355 ページの『CL コマンドのプロンプト時に追加または拡張パラメーターを非表示にする』
ユーザーがプロンプトの過程で機能キーを押して追加のパラメーターを要求しない限り、使用頻度の低いパラメーターについてのプロンプトが表示されないようにするには、そのパラメーターに対する PARM ステートメントに PMTCTL(*PMTRQS) を指定します。

CL コマンドのプロンプト一時変更プログラムの使用*:*

プロンプト一時変更プログラムを使用することによって、コマンドのプロンプトの表示時点で、デフォルト値ではなく現在の値を表示できます。

プロンプト一時変更プログラムを使用するには、以下の手順に従ってください。

1. キー・パラメーターとなるすべてのパラメーターをコマンド定義ソースの PARM ステートメントに指定します。
2. プロンプト一時変更プログラムを作成します。
3. コマンドの作成または変更時点で、プロンプト一時変更プログラムの名前を PMTOVRPGM パラメーターに指定します。

CL コマンドのキー・パラメーターの識別*:*

キー・パラメーターの数は、変更したいオブジェクトを固有なものとして定義するのに必要なパラメーターの数に限定しなければなりません。

コマンド定義ソースの中でキー・パラメーターを正しくコーディングするには、以下の要件について考慮してください。

- コマンド定義ソースの PARM ステートメントに KEYPARM(*YES) を指定する。
- KEYPARM(*YES) を指定するパラメーターはすべて、KEYPARM(*NO) を指定するすべてのパラメーターの前に定義する。

  注**:** KEYPARM(*YES) を指定した PARM ステートメントを、KEYPARM(*NO) を指定した PARM ステートメントの後に指定すると、そのパラメーターはキー・パラメーターとしては扱われず、警告メッセージが出されます。
- その PARM ステートメントの MAX 値に 1 より大きい値を指定してはなりません。
- キー・パラメーターに関連する ELEM ステートメントに対して、MAX 値に 1 より大きい値を指定してはなりません。
- その PARM ステートメントの PMTCTL キーワードに対して、*PMTRQS またはプロンプト制御ステートメントを指定しなければなりません。
- コマンド定義ソースの中のキー・パラメーターは、プロンプトの表示時点でパラメーターを表示したい順序で指定する。

CL コマンドのプロンプト一時変更プログラム*:*

プロンプト一時変更プログラムに対して、コマンドのプロンプトの表示時点での現在の値を戻すために必要な特定の情報を渡す必要があります。

それで、プログラム一時変更プログラムを作成する場合には、渡す値と戻される値の両方を考える必要があります。

関連概念:

361 ページの『例: プロンプト一時変更プログラムの使用』

この例は、あるコマンドのコマンド・ソースとそのプロンプト一時変更プログラムを示しています。

プロンプト一時変更プログラムに渡されるパラメーター*:*

プロンプト一時変更プログラムには、複数のパラメーターが渡されます。

- 20 文字のフィールド。このフィールドの先頭の 10 文字にはコマンドの名前を入れ、後半の 10 文字にはライブラリーの名前を入れます。
- キー・パラメーターがある場合、それぞれの値。複数のキー・パラメーターが定義されている場合、パラメーター値を渡す順序は、コマンド定義ソースの中で該当のキー・パラメーターが定義されている順序です。
- プロンプト一時変更プログラムによって作成されるコマンド・ストリングを入れるための 32676 バイト (32K) のスペース。 このフィールドの先頭の 2 バイトには、戻されるコマンド・ストリングの 16 進数の長さが入っていなければなりません。 実際のコマンド・ストリングは、その 2 バイトの後に続きます。

  例えば、1 つのコマンドに 2 つのキー・パラメーターが定義されている場合には、4 つのパラメーターが次のようにプロンプト一時変更プログラムに渡されます。
  - コマンドに 1 つのパラメーター
  - キー・パラメーターに 2 つのパラメーター
  - コマンド・ストリングのスペースに 1 つのパラメーター

プロンプト一時変更プログラムから戻される情報*:*

プロンプト一時変更プログラムは、渡された情報に基づいて、キー・パラメーター以外のパラメーターの現在の値を検索します。

検索された値はコマンド・ストリングに入れられ、ストリングの長さが判別され、その値が戻されます。

コマンド・ストリングを正しく定義するには、以下のことを守ってください。

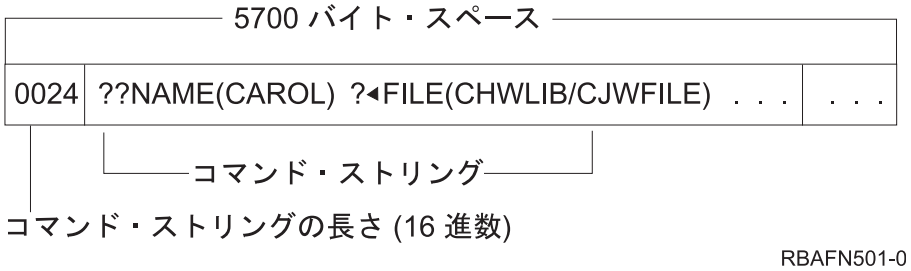
- コマンド・ストリングに、コマンド行の場合と同様のキーワード形式を使用する。
- コマンド・ストリングには、コマンド名およびキー・パラメーターを含めてはならない。
- 各キーワードの前に選択プロンプト文字を指定して、そのパラメーターの表示方法と、CPP にどの値を渡すかを定義する。

  選択プロンプトを使用する場合、以下のことを行ってください。
  - コマンド定義ソースでパラメーターを MIN(1) として定義している場合 (つまり、このパラメーターは必須)、プロンプト一時変更プログラムが生成するコマンド・ストリング中のそのキーワードに対しては、選択プロンプト文字 **??** を指定する必要があります。
  - プロンプト一時変更プログラムのコマンド・ストリングの中では、選択プロンプト文字 **?-** を指定してはなりません。

  以下の例は、プロンプト一時変更プログラムから戻されるコマンド・ストリングの例です。

  ```
  ??Number(123456) ?<Qualifier(CLIB/CFILE) ?<LIST(ITEM1
  ITEM2 ITEM3) ?<TEXT('Carol's file')
  ```

- プログラムが渡すスペースの先頭の 2 バイトに指定される値が、コマンド・ストリングの実際の 16 進数の長さであることを確認する。



RBAFN501-0

- コマンド・ストリングには、コマンドのプロンプトの表示時点で現在の値を表示したいパラメーターだけを指定する。コマンド・ストリングに指定されていないパラメーターは、デフォルト値が表示されます。
- コマンド・ストリングに含まれる数字に対しては文字形式を使用する。 2 進数形式またはパック形式を使用してはなりません。コマンド・ストリングの中に 16 進数を含めることもできません。
- ライブラリーと修飾子の間、または修飾子とオブジェクトの間にブランクを入れない。以下にその例を示します。

  ??KWD1(library /object)
  無効

  ??KWD1(library/ object)
  無効

  ??KWD1(library/object)
  有効

??KWD1(library/object)

有効

- 特殊値または単一値を使用する場合には、コマンド定義ソースに定義されている取り出し値に変換されるように指定する。

  例えば、コマンド定義ソースの中でキーワードに SPCVAL(*SPECIAL *) を指定して特殊値を定義したとします。この場合、*SPECIAL が取り出し値で * が受け取り置き換え値です。このキーワードに対して現在の値を検索するとき、* が検索される値ですが、*SPECIAL をプロンプト一時変更プログラムによって戻されるコマンド・ストリングに含まれなければなりません。複数の特殊値または複数の単一値の受け取り置き換え値が同じ場合があり得るので、コマンド・ストリングに正しい取り出し値を入れなければなりません。例えば、KWD1 SPCVAL((*SPC *) (*SPECIAL *)) の指定がある場合、* は *SPC に対応する受け取り置き換え値なのか、 *SPECIAL に対応する受け取り置き換え値なのかをプロンプト一時変更プログラムが判別しなければなりません。
- テキスト検索に使用するフィールド長を次のように定義する。

  (2*(コマンド定義ソースに定義されているフィールドの長さ)) + 2

  この長さには、テキスト・フィールドで使用できる引用符の最大数も含まれています。例えば、コマンド定義ソースで CHGxxx コマンドの TEXT パラメーターを LEN(50) として定義した場合、そのプロンプト一時変更プログラムでの該当するパラメーターは CHAR(102) として宣言してください。

  プロンプト一時変更プログラムの中でテキスト・フィールドのパラメーターが正しく指定されていない場合、プロンプト一時変更プログラムが検索したテキスト・ストリングに引用符が含まれているとそのコマンドのプロンプトは正しく行われません。
- 内部に含まれる単一引用符は必ずすべて二重にする。次に例を示します。

```
?<TEXT('Carol''s library')
```

コマンドによっては特定のモード (DEBUG など) または特定のジョブ状況 (*BATCH など) でだけ実行できるものがありますが、プロンプトは他のモードまたは他のジョブ状況からでも表示できます。コマンドのプロンプトを表示する時には、ユーザーの環境に関係なくプロンプト一時変更プログラムが呼び出されます。プロンプト一時変更プログラムを呼び出したモードまたは環境がそのコマンドに対して無効な場合、該当するコマンドのデフォルト値が表示され、長さの値として 0 が戻されます。このような状態の例として、デバッグ・モードでない時点でデバッグ・コマンドである**デバッグ変更 (CHGDBG)** コマンドまたは **プログラム追加 (ADDPGM)** コマンドを使用する場合などがあります。

関連概念:

361 ページの『例: プロンプト一時変更プログラムの使用』
この例は、あるコマンドのコマンド・ソースとそのプロンプト一時変更プログラムを示しています。

関連タスク:

398 ページの『CL コマンドの選択プロンプトの使用』
CL コマンドの選択プロンプトは、長い構文のコマンドを使用する際に、一部のパラメーターについてはプロンプトを表示したくない場合に、特に便利な機能です。

関連情報:

デバッグ変更 (CHGDBG) コマンド

プログラム追加 (ADDPGM) コマンド

プロンプト一時変更プログラムでのエラーの許容:

ユーザーのプロンプト一時変更プログラムには、エラー処理が含まれている必要があります。

プロンプト一時変更プログラムがエラーを検出した場合、プログラムは以下の手順に従ってエラーを処理します。

1. コマンド・ストリングの長さをゼロに設定する。これによりコマンドのプロンプトには、現在の値ではなくデフォルト値が表示されます。
2. 呼び出しスタックでの直前のプログラムに診断メッセージを送る。
3. エスケープ・メッセージ CPF0011 を送る。

例えば、ライブラリーが存在していないことを表すメッセージを出したい場合には、次のようなメッセージ記述を追加してください。

```
ADDMSGD      MSG('Library &2 does not exist') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 10))
```

注**:** 置換変数 &1 は、このメッセージでは使用されていませんが、FMT パラメーターで 4 文字として、定義されています。 &1 はシステムによる使用のために予約されており、必ず 4 文字でなければなりません。 置換変数 &1 がメッセージに定義された唯一の置換変数である場合には、メッセージの送信時点でメッセージ・データの 4 番目のバイトに決してブランクが入らないようにしなければなりません。 システムは 4 番目のバイトを使用して、コマンド処理時およびプロンプト表示時にメッセージを管理します。

プロンプト一時変更プログラムで以下のように指定して、このメッセージをプロンプト一時変更プログラムの呼び出し元のプログラムに送ることができます。

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' vv &libname) MSGTYPE(*DIAG)
```

プロンプト一時変更プログラムは、必要な診断メッセージをすべて送った後、メッセージ CPF0011 を送らなければなりません。メッセージ CPF0011 を送るには、プログラム・メッセージ送信 (SNDPGMMSG) コマンドを以下のように指定します。

```
SNDPGMMSG    MSGID(CPF0011) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

メッセージ CPF0011 が受け取られると、メッセージ CPD680A が呼び出し元のプログラムに送られ、エラーが検出されたことを示すメッセージとしてプロンプト画面に表示されます。診断メッセージはすべて、ユーザーのジョブ・ログに入れられます。

関連情報:

プログラム・メッセージ送信 (SNDPGMMSG) コマンド

CL コマンド作成または変更時のプロンプト一時変更プログラムの指定*:*

作成するコマンドに対してプロンプト一時変更プログラムを使用する場合には、**コマンド作成 (CRTCMD)** コマンドを使用するときに、そのプログラム名を指定してください。 **コマンド変更 (CHGCMD)** コマンドを使用してコマンドを変更する場合にも、プログラム名を指定できます。

CRTCMD と **CHGCMD** のどちらのコマンドの場合にも、プロンプト一時変更プログラムの名前を PMTOVRPGM パラメーターに指定してください。

コマンド定義ソースでキー・パラメーターが定義されている場合に、コマンド作成または変更時にプロンプト一時変更プログラムを指定しないと、警告メッセージ CPD029B が出されます。この場合、キー・パラメーターは無視され、コマンドのプロンプトにはコマンド定義ソースに指定されているデフォルトの値が表示されます。

コマンドの作成時にコマンド定義ソースにキー・パラメーターが定義されていない場合、プロンプト一時変更プログラムを指定することがあります。この場合、プロンプト一時変更プログラムは、コマンドのプロンプトが出される前に呼び出されます。コマンドを作成または変更する際に情報メッセージ CPD029A が出されます。

関連情報:

コマンド作成 (CRTCMD) コマンド

コマンド変更 (CHGCMD) コマンド

例: プロンプト一時変更プログラムの使用:

この例は、あるコマンドのコマンド・ソースとそのプロンプト一時変更プログラムを示しています。

次のコマンドにより、ライブラリーの所有者とテキスト記述を変更できます。このコマンドのプロンプト一時変更プログラムは、ライブラリー名を受け取り、ライブラリーの現在の所有者の値およびテキスト記述の値を検索し、検索した値をコマンド・ストリングに入れて呼び出し元に戻します。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

コマンド・ソース

```
CHGLIBATR: CMD   PROMPT('Change Library Attributes')
           PARM  KWD(LIB) +
                   TYPE(*CHAR) MIN(1) MAX(1) LEN(10) +
                   KEYPARM(*YES) +
                   PROMPT('Library to be changed')
           PARM  KWD(OWNER) +
                   TYPE(*CHAR) LEN(10) MIN(0) MAX(1) +
                   KEYPARM(*NO) +
                   PROMPT('Library owner')
           PARM  KWD(TEXT) +
                   TYPE(*CHAR) MIN(0) MAX(1) LEN(50) +
                   KEYPARM(*NO) +
                   PROMPT('Text description')
```

プロンプト一時変更プログラム

以下のプロンプト一時変更プログラムは、 "?^" 選択プロンプト文字を使用します。

```
  PGM PARM(&cmdname &keyparm1 &rtnstring)
 /*****************************************************************/
 /*                                                               */
 /*  Declarations of parameters passed to the prompt override program */
 /*                                                               */
 /*****************************************************************/
  DCL  VAR(&cmdname)   TYPE(*CHAR) LEN(20)
  DCL  VAR(&keyparm1)   TYPE(*CHAR) LEN(10)
  DCL  VAR(&rtnstring) TYPE(*CHAR) LEN(5700)
 /*****************************************************************/
 /*                                                               */
 /* Return command string structure declaration                   */
 /*                                                               */
 /*****************************************************************/

                      /* Length of command string generated        */
  DCL  VAR(&stringlen) TYPE(*DEC)  LEN(5 0) VALUE(131)
  DCL  VAR(&binlen)    TYPE(*CHAR) LEN(2)
                      /* OWNER keyword                             */
  DCL  VAR(&ownerkwd)  TYPE(*CHAR) LEN(8)  VALUE('?<OWNER(')
```

```
 DCL  VAR(&name)      TYPE(*CHAR) LEN(10)
                      /* TEXT keyword                                  */
 DCL  VAR(&textkwd)   TYPE(*CHAR) LEN(8)  VALUE(' ?<TEXT(')
 DCL  VAR(&descript)  TYPE(*CHAR) LEN(102)
/*****************************************************************/
/*                                                               */
/* Variables related to command string declarations              */
/*                                                               */
/*****************************************************************/
 DCL  VAR(&quote)      TYPE(*CHAR) LEN(1) VALUE('''')
 DCL  VAR(&closparen)  TYPE(*CHAR) LEN(1) VALUE(')')
/*****************************************************************/
/*                                                               */
/*                  Start of operable code                       */
/*                                                               */
/*****************************************************************/
/*****************************************************************/
/*                                                               */
/* Monitor for exceptions                                        */
/*                                                               */
/*****************************************************************/
    MONMSG  MSGID(CPF0000) +
            EXEC(GOTO CMDLBL(error))
/*****************************************************************/
/*                                                               */
/* Retrieve the owner and text description for the library specified*/
/* on the LIB parameter.  Note:  This program assumes there are     */
/* no apostrophes in the TEXT description, such as (Carol's)  */
/*                                                               */
/*****************************************************************/
    RTVOBJD OBJ(&keyparm1) OBJTYPE(*LIB) OWNER(&name) TEXT(&descript)

    CHGVAR VAR(%BIN(&binlen)) VALUE(&stringlen)
/*****************************************************************/
/*                                                               */
/* Build the command string                                      */
/*                                                               */
/*****************************************************************/
    CHGVAR VAR(&rtnstring) VALUE(&binlen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &ownerkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &name)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &textkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &descript)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)

    GOTO CMDLBL(pgmend)
    ERROR:
    VALUE(0)
    CHGVAR VAR(%BIN(&rtnstring 1 2)) VALUE(&stringlen)
    VALUE(&binlen)
/*****************************************************************/
/*                                                               */
/* Send error message(s)                                         */
/*                                                               */
/* NOTE: If you want to send a diagnostic message as well as CPF0011*/
/*       you will have to enter a valid error message ID in the     */
/*       MSGID parameter and a valid message file in the MSGF       */
/*       parameter for the first SNGPGMMSG command listed below.    */
/*       If you do not want to send a diagnostic message, do not    */
/*       include the first SNDPGMMSG in your program.  However, in  */
/*       error conditions, you must ALWAYS send CPF0011 so the      */
/*       second SNDPGMMSG command must be included in your program. */
/*                                                               */
```

```
/*****************************************************************/
    SNDPGMMSG MSGID(XXXXXXX) MSGF(MSGLIB/MSGFILE) MSGTYPE(*DIAG)
    SNDPGMMSG MSGID(CPF0011) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)

PGMEND:
ENDPGM
```

関連概念:

357 ページの『CL コマンドのプロンプト一時変更プログラム』
プロンプト一時変更プログラムに対して、コマンドのプロンプトの表示時点での現在の値を戻すために必要な特定の情報を渡す必要があります。

358 ページの『プロンプト一時変更プログラムから戻される情報』
プロンプト一時変更プログラムは、渡された情報に基づいて、キー・パラメーター以外のパラメーターの現在の値を検索します。

CL コマンドの作成**:**

コマンド定義ステートメントによるユーザー・コマンドの定義が完了した時点で、**コマンド作成 (CRTCMD)** コマンドを使用して、そのコマンドを作成できます。

コマンド名、ライブラリー名、および CL 言語や高水準言語 (HLL) のコマンド処理プログラム名、REXX の場合のソース・メンバー、ソース・ファイル、コマンド環境、および出口プログラムを指定する他に、次のようなコマンドの属性を定義できます。

- コマンドで使用する妥当性検査プログラム
- コマンドを実行できるモード
  - 実動
  - デバッグ
  - 保守
- コマンドを使用できる場所
  - バッチ・ジョブ
  - 対話式ジョブ
  - バッチ・ジョブの ILE CL モジュール
  - バッチ・ジョブの CL プログラム
  - 対話式ジョブの ILE CL モジュール
  - 対話式ジョブの CL プログラム
  - バッチ・ジョブの REXX プロシージャー
  - 対話式ジョブの REXX プロシージャー
  - QCMDEXC または QCAPCMD に対する呼び出しを介してシステムが解釈実行するコマンドとして。
- 定位置形式で指定できるパラメーターの最大数
- プロンプト・テキストが入っているメッセージ・ファイル
- プロンプト表示が可能なパラメーターのヘルプとして使用されるヘルプ・パネル・グループ
- このコマンドで使用される一般的なヘルプ・モジュールのヘルプ識別コード名
- DEP ステートメントで指定されたメッセージが入っているメッセージ・ファイル
- コマンド処理の過程で活動化する現行ライブラリー
- コマンド処理の過程で活動化する実行 (プロダクト) ライブラリー

- REPLACE(*YES) が指定された場合に、名前、タイプ、およびライブラリーが同じである既存のコマンドを置き換えるかどうか
- コマンドおよびその記述に対して共通認可として与える権限
- コマンドおよびその機能について簡潔に記述するテキスト

REXX の CPP を使用するコマンドの場合、以下のものも指定できます。

- プロシージャーの開始時にコマンドを処理するための初期コマンド環境
- プロシージャーの実行を制御する出口プログラム

以下の例では、受注アプリケーションを呼び出すための ORDENTRY という名前のコマンドを定義しています。 CRTCMD コマンドは、ORDENTRY について前述の属性を定義し、IBM 提供のソース・ファイル QCMDSRC のメンバー ORDENTRY に入っているパラメーター定義を用いてコマンドを作成します。 ORDENTRY には、 327 ページの『例: CL コマンド・パラメーターの定義』 の項に示した例で使用した PARM ステートメントが入っています。

```
CRTCMD     CMD(DSTPRODLB/ORDENTRY) +
           PGM(*LIBL/ORDENT) +
           TEXT('Calls order entry application')
```

結果のコマンドは、以下のとおりです。ここで、値には、DAILY、WEEKLY、MONTHLY のいずれかを指定できます。

```
ORDENTRY  OETYPE(value)
```

コマンドを作成すると、次のタスクを行うことができます。

- コマンド表示 (DSPCMD) コマンドを使用して、そのコマンドの属性を表示する。
- コマンド変更 (CHGCMD) コマンドを使用して、そのコマンドの属性を変更する。
- コマンド削除 (DLTCMD) コマンドを使用して、そのコマンドを削除する。

関連概念:

390 ページの『コマンド関連の API』
一部のアプリケーション・プログラミング・インターフェース・プログラムは、コマンドで使用できます。

関連タスク:

327 ページの『例: CL コマンド・パラメーターの定義』
この例は、CL コマンドによるアプリケーションの呼び出しで使用されるパラメーターの定義方法を示しています。

関連情報:

コマンド作成 (CRTCMD) コマンド

CL コマンド検索プログラム

CL コマンド定義ソース・リスト**:**

CL コマンドを作成する際には、ソース・リストが生成されます。

これは、ソース・リストの例です。リスト中の参照番号は、後述する説明の番号に対応しています。

```
5770SS1 V7R1M0  100423 1               Command Definition       DSTPRODLB/ORDENTRY   SYSNAME  11/20/12 14:53:32 2Page 13

コマンド名 . . . . . :   ORDENTRY
  ライブラリー . . . . . :     DSTPRODLB
コマンド処理プログラム . . . . . :   ORDENT     4
  ライブラリー . . . . . :     *LIBL
ソース・ファイル . . . . . :   QCMDSRC
  ライブラリー . . . . . :     QGPL
ソース・ファイル・メンバー . . . . . :   ORDENTRY     11/20/12  14:54:32
妥当性検査プログラム . . . . . :   *NONE
有効なモード . . . . . :   *PROD
                                  *DEBUG
                                  *SERVICE
許された環境 . . . . . :   *IREXX
                                  *BREXX
                                  *BPGM
                                  *IPGM
                                  *EXEC
                                  *INTERACT
                                  *BATCH
                                  *BMOD
                                  *IMOD
制限ユーザー可能 . . . . . :   *NO
定位置パラメーターの最大数 . . . . . :   *NOMAX
プロンプト・ファイル . . . . . :   *NONE
メッセージ・ファイル . . . . . :   QCPFMSG
  ライブラリー . . . . . :     *LIBL
権限 . . . . . :   *LIBCRTAUT
置き換えコマンド . . . . . :   *YES
グラフィカル・ユーザー・インターフェース使用可能 . :   *NO
スレッド・セーフ . . . . . :   *NO
マルチスレッド・ジョブの処置 . . . . . :   *SYSVAL
テキスト . . . . . :   受注アプリケーションの呼び出し
ヘルプ・ブック名 . . . . . :   *NONE
ヘルプ・ブックシェルフ . . . . . :   *NONE
ヘルプ・パネル・グループ . . . . . :   *NONE
ヘルプ ID. . . . . :   *NONE
ヘルプ検索見出し . . . . . :   *NONE
現行ライブラリー . . . . . :   *NOCHG
プロダクト・ライブラリー . . . . . :   *NOCHG
プロンプト一時変更プログラム . . . . . :   *NONE
コンパイラー . . . . . :   IBM コマンド定義コンパイラー 5

                              コマンド定義ソース
6
SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.  日付  8
   100-              CMD       PROMPT('Order entry command')                                    11/20/12
7
   200-              PARM      KWD(OETYPE) TYPE(*CHAR) RSTD(*YES) +                             11/20/12
   300                           VALUES(DAILY WEEKLY MONTHLY) MIN(1) +                         11/20/12
   400                           PROMPT('Type of order entry:')                                11/20/12
                                     * * * * *  ソ ー ス の 終 わ り  * * * * *

5770SS1 V7R1M0  100423               コマンド定義               DSTPRODLB/ORDENTRY     11/20/12 14:54:32         Page    2
                                       相互参照

定義済みキーワード 9
キーワード           番号             定義      参照
OETYPE                001             200
                          * * * * *  相 互 参 照 表 の 終 わ り  * * * * *

5770SS1 V7R1M0  100423               コマンド定義               DSTPRODLB/ORDENTRY     11/20/12 14:54:32         Page    3
                                       最終メッセージ

メッセージ             順序
ID                     番号         重大       テキスト  10

                                   メッセージの要約
合計         情報      エラー
    0            0          0   11
* CPC0202                          00       コマンド ORDENTRY がライブラリー DSTPRODLB に作成された。12
                             * * * * *  コ ン パ イ ル の 終 わ り  * * * * *
```

タイトル:

1 IBM i オペレーティング・システムのプログラム番号、バージョン、リリース、モディフィケーション・レベル、および日付。

2 実行の日付および時刻。

3 リストのページ番号。

プロローグ

4 **CRTCMD** コマンドに指定したパラメーター値 (指定のない場合はデフォルト値)。ソース・ファイルがデータベース・ファイルでない場合には、メンバー名、日付、および時刻は省略されます。

5 コマンド定義作成コンパイラーの名前。

ソース*:*

6 ソース・ファイルの行 (レコード) の順序番号。順序番号の後のダッシュ (-) は、ソース・ステートメントがその順序番号から始まっていることを示します。ダッシュがない場合は、そのステートメントは前のステートメントの続きを示します。例えば、PARM ソース・ステートメントは順序番号 200 から始まり、順序番号 300 および 400 まで継続しています。 (順序番号 200 および 300 の PARM ステートメントに継続文字 + があることに注意してください。)

注記ソース・ステートメントは他のソース・ステートメントと同様に扱われ、順序番号を持っています。

7 ソース・ステートメント。

8 ソース・ステートメントが最後に変更または追加された日付。変更または追加が行われたことのないソース・ステートメントについては、日付は示されません。ソースがデータベース・ファイルに入っていない場合、日付は省略されます。

コマンド定義ステートメントの処理でエラーが検出され、そのエラーをトレースして特定のソース・ステートメントに結び付けることができた場合には、そのソース・ステートメントの直後にエラー・メッセージが印刷されます。アスタリスク (*) は、その行がエラー・メッセージであることを示します。その行には、メッセージ識別コード、重大度、およびメッセージ・テキストが印刷されます。

相互参照*:*

9 キーワード表は、コマンド定義の中で正しく定義されているキーワードの相互参照リストです。この表には、キーワード、コマンド内でのキーワードの位置、キーワードを定義したステートメントの順序番号、およびキーワードを参照するステートメントの順序番号がリストされます。

コマンド定義に有効なラベルが定義されている場合は、ラベルの相互参照リスト (ラベル表) が提供されます。このテーブルには、ラベル、そのラベルが定義されているステートメントの順序番号、およびそのラベルを参照しているステートメントの順序番号がリストされます。

メッセージ*:*

10 コマンド定義ステートメントの処理で検出され、ソースの部分にリストされていない一般的なエラー・メッセージのリスト。この部分には、各メッセージごとにメッセージ識別コード、エラーが起こったステートメントの順序番号、重大度、およびメッセージ・テキストが示されます。

メッセージの要約*:*

11 コマンド定義ステートメントの処理で出されたメッセージの数についての要約。総数とともに、重大度別の内訳も示されます。

12 メッセージの要約の後には完了メッセージが印刷されます。

関連概念:

『CL コマンド定義ステートメント処理時の共通エラー』
コマンド定義ステートメントの処理時に検出されるエラーには、構文エラー、未定義のキーワードやラベルの参照、ステートメントの欠落などがあります。

CL コマンド定義ステートメント処理時の共通エラー*:*

コマンド定義ステートメントの処理時に検出されるエラーには、構文エラー、未定義のキーワードやラベルの参照、ステートメントの欠落などがあります。

以下のタイプのエラーがコマンド定義コンパイラーにより検出されると、コマンドの作成は打ち切られます (重大度コードは無視されます)。

- 値のエラー
- 構文エラー

コマンドの作成を中止するエラーが検出された後でも、コマンド定義コンパイラーは他にエラーがないかどうか調べるため、ソースの検査を続けます。構文エラーおよび固定値のエラーが起こると、ユーザー名およびユーザー指定値のエラー、およびキーワードまたはラベルの参照エラーを識別するための最終検査が行われなくなります。しかし、構文エラーおよび固定値エラーの検査は続行されます。したがってユーザーは、できるだけ多くのエラーを見つけて訂正した上で、コマンドを作成し直すことができます。ソース・ステートメントで生じたエラーの訂正には、 EDTF (ファイルの編集) コマンドを使用するか、原始ステートメント入力ユーティリティー (SEU) を使用できます。SEU は、WebSphere Development Studio の一部です。

コマンド定義のソース・リストでは、特定のソース・ステートメントに直接関係するエラー条件はそのコマンドの直後にリストされます。 特定のソース・ステートメントに結びつかない、より一般的な内容のメッセージは、ソース・ステートメントの間に挿入されず、リストのメッセージ部分にまとめて示されます。

関連資料:

364 ページの『CL コマンド定義ソース・リスト』
CL コマンドを作成する際には、ソース・リストが生成されます。

CL コマンド定義の表示**:**

コマンド作成 (CRTCMD) コマンドでパラメーターとして指定されている値を表示または印刷するには、**コマンド表示 (DSPCMD)** コマンドを使用します。

DSPCMD コマンドによって、ユーザー作成コマンドまたは IBM 提供のコマンドに関する以下の情報が表示されます。

- 修飾コマンド名。ライブラリー名は、表示するコマンドが入っているライブラリーの名前です。
- コマンド処理プログラムの修飾名。ライブラリー名は、**CRTCMD** コマンド、または **CHGCMD** コマンドにライブラリー名が指定されていた場合には、コマンド作成時にコマンド処理プログラムが入っていたライブラリーの名前です。ライブラリー名が指定されていなかった場合には、 *LIBL がライブラリー修飾子として表示されます。 CPP が REXX プロシージャーの場合、*REXX が表示されます。
- 修飾ソース・ファイル名 (ソース・ファイルがデータベース・ファイルの場合)。ライブラリー名は、**CRTCMD** コマンドの実行時にソース・ファイルが入っていたライブラリーの名前です。ソース・ファイルがデータベース・ファイルでない場合には、このフィールドはブランクになります。
- ソース・ファイル・メンバー名 (ソース・ファイルがデータベース・ファイルの場合)。
- CPP が REXX プロシージャーの場合、以下の情報が表示されます。
  - REXX プロシージャーのメンバー名
  - REXX プロシージャーが入っている REXX 修飾ソース・ファイル名
  - REXX コマンド環境
  - REXX 出口プログラム
- 妥当性検査プログラムの修飾名。ライブラリー名は、**CRTCMD** コマンド、または **CHGCMD** コマンドにライブラリー名が指定されていた場合には、コマンドの作成時にプログラムが入っていたライブラリーの名前です。ライブラリー名が指定されていなかった場合には、 *LIBL がライブラリー修飾子として表示されます。

- 有効な操作モード。
- コマンドが実行できる有効な環境。
- コマンドの最大定位置パラメーター数。そのコマンドに最大定位置パラメーターがない場合には、*NOMAX が表示されます。
- プロンプト・メッセージ・ファイルの修飾名。ライブラリー名は、**CRTCMD** コマンドの実行時に、メッセージ・ファイルが入っていたライブラリーの名前です。そのコマンドについてプロンプト・メッセージ・ファイルが存在していない場合には、*NONE が表示されます。
- DEP ステートメントのメッセージ・ファイルの修飾名。コマンドの作成時にこのメッセージ・ファイルについてライブラリー名が指定されている場合には、そのライブラリー名が表示されます。コマンドの作成時にライブラリー・リストが使用された場合には、*LIBL が表示されます。コマンドについて DEP メッセージ・ファイルが存在していない場合には、*NONE が表示されます。
- ヘルプ・パネル・グループの修飾名。
- コマンドのヘルプ識別コード名。
- プロンプト一時変更プログラムの修飾名。
- コマンドに関連するテキスト。コマンドについてテキストが存在していない場合には、ブランクが表示されます。
- コマンド・プロンプトが、グラフィカル・ユーザー・インターフェースへの変換に使用可能かどうかを示すための標識。
- スレッド・セーフ標識。
- コマンドがスレッド・セーフでない場合の、マルチスレッド・ジョブの処置。

コマンド情報の検索 (QCDRCMDI) API を使用すれば、 **CRTCMD** コマンドでそのコマンドの作成時に指定されたコマンド属性を戻せます。また、コマンド定義の検索 (QCDRCMDD) API を使用すれば、コマンド定義オブジェクトの構造を検索できます。これには、パラメーター情報、パラメーター内の依存関係の情報、および条件付きプロンプトに関する情報が含まれます。

関連情報:

コマンド作成 (CRTCMD) コマンド

コマンド変更 (CHGCMD) コマンド

コマンド表示 (DSPCMD) コマンド

Retrieve Command Information (QCDRCMDI) API

Retrieve Command Definition (QCDRCMDD) API

プロシージャーまたはプログラムでの **CL** コマンドのコマンド定義変更による影響**:**

さらにアクションを取らずにコマンドのコマンド定義に複数の変更を加えることができます。他の変更は、プログラムやプロシージャーの再作成を必要としたり、プログラムやプロシージャーが異なって機能する場合があります。

CL モジュールまたは CL プログラムの作成時には、そのプロシージャーまたはプログラム内のコマンドのコマンド定義を使用してモジュールまたはプログラムが生成されます。 CL プロシージャーまたは CL プログラムの実行時にも、コマンド定義が使用されます。 CL プロシージャーまたは CL プログラム中のコマンドにライブラリー名を指定した場合、そのコマンドはプロシージャー作成時にもプロシージャー実行時にも同じライブラリーに入っていなければなりません。 CL プロシージャーまたは CL プログラム中のコマンドに *LIBL を指定した場合には、プログラムの作成時にも実行時にもライブラリー・リスト (*LIBL) を用いてコマンドが探されます。

コマンドのコマンド定義ステートメントに対して、そのコマンドを使用するモジュールまたはプログラムを作成し直す必要なしに、以下の変更を行うことができます。これらの変更の中には、コマンド定義ステートメントのソースに対する変更であるためコマンドを作成し直さなければならないものもあります。またコマンド変更 (CHGCMD) コマンドを用いて行う変更もあります。

- オプショナル・パラメーターを任意に追加する。最大定位置パラメーターの前にオプショナル・パラメーターを追加すると、定位置形式で指定されたパラメーターを含むプロシージャー、プログラムおよびバッチ入力ストリームに影響が生じることがあります。
- REL および RANGE の検査を変更し、制約条件を少なくする。
- 新しい特殊値を追加する。ただし、該当の値が変更前に指定されている場合には、プロシージャーまたはプログラムの処置が変更することがあります。
- パラメーターの順序を変更する。ただし、最大定位置パラメーターより前にあるパラメーターの順序を変更すると、定位置形式で指定されたパラメーターを含むプロシージャー、プログラム、およびバッチ入力ストリームに影響が生じます。
- 単純リスト内のオプションの要素の数を増やす。
- デフォルト値を変更する。ただし、これはプロシージャーまたはプログラムの操作に影響を及ぼすことがあります。
- 単純リスト内の必須リスト要素の数を減らす。
- パラメーターを必須からオプションに変更する。
- RSTD を *YES から *NO に変更する。
- FULL(*NO) が指定されている場合に、長さを増やす。
- FULL を *YES から *NO に変更する。
- PROMPT のテキストを変更する。
- ALLOW の値を変更して制約を緩和する。
- コマンド処理プログラムの名前を変更する (新しいコマンド処理プログラムが正しい数とタイプのパラメーターを受け入れる場合)。
- 妥当性検査プログラムの名前を変更する (新しい妥当性検査プログラムが正しい数とタイプのパラメーターを受け入れる場合)。
- コマンドの実行が可能なモードを変更する (新しいモードが、CL プロシージャーまたは CL プログラムで使用されている同じコマンドの旧モードに影響を及ぼさない場合)。
- TYPE を変更し、互換性があり制約の少ない値にする。例えば、TYPE を *NAME から *CHAR に変更できます。
- MAX を 1 より大きい値に変更する。
- PASSATR および VARY の値を変更する。

以下の変更をコマンド定義ステートメントに対して行うことができるかどうかは、そのコマンドを使用する CL プロシージャーまたは CL プログラムの指定内容に応じて決まります。

- パラメーターを除去する。
- RANGE および REL の値を変更して制約を厳しくする。
- 特殊値を除去する。
- リストに指定可能な要素数を減らす。
- TYPE の値を変更して制約を厳しくするか、またはもとの TYPE の値との互換性がなくなるようにする。例えば、TYPE の値を *CHAR から *NAME に変更したり、*PNAME を *CHAR に変更するような場合です。

- 以前にはリスト要素であった SNGVAL パラメーターを追加する。
- オプショナル・パラメーターの名前を変更する。
- 値のリストから値を除去する。
- 必須リスト項目の数を増やす。
- SNGVAL パラメーターを SPCVAL パラメーターに変更する。
- 単純リストを類似要素の混合リストに変更する。
- オプショナル・パラメーターを定数に変更する。
- RTNVAL の値を *YES から *NO に、または *NO から *YES に変更する。
- ケース値を *MIXED から *MONO に変更する。

以下の変更は、コマンド定義ステートメントに対して行うことができますが、そのコマンドを使用するプロシージャーまたはプログラムの機能に影響を与えることがあります。

- 値の意味を変更する。
- デフォルト値を変更する。
- SNGVAL パラメーターを SPCVAL パラメーターに変更する。
- 値を SNGVAL パラメーターに変更する。
- リストをリスト内リストに変更する。
- ケース値を *MIXED から *MONO に変更する。

以下のコマンド定義ステートメントに対する変更には、そのコマンドを使用するプロシージャーまたはプログラムの再作成が必要になります。

- 新しい必須パラメーターを追加する。
- 必須パラメーターを除去する。
- 必須パラメーターの名前を変更する。
- 必須パラメーターを定数に変更する。
- コマンド処理プログラムを *REXX へ、または *REXX から変更する。

さらに、コマンドの作成時または変更時に、コマンド処理プログラムや妥当性検査プログラムの名前の修飾子として *LIBL を指定した場合は、コマンド定義ステートメントを変更することなくそのコマンド処理プログラムや妥当性検査プログラムを、ライブラリー・リスト内の別のライブラリーに移すことができます。

CL コマンドのデフォルト変更*:*

コマンド・パラメーターのデフォルト値を変更するには、**コマンドのデフォルト変更 (CHGCMDDFT)** コマンドを使用します。

コマンド・パラメーターには、新規デフォルト値に変更する既存のデフォルト値が備わっている必要があります。 変更するコマンドは、IBM 提供のコマンドとユーザー作成のコマンドのどちらでも構いません。IBM 提供のコマンドのデフォルト値を変更する場合には注意が必要です。デフォルトを変更する際の推奨事項は、次のとおりです。

1. 変更するコマンドが IBM 提供のコマンドの場合には、**複製オブジェクト作成 (CRTDUPOBJ)** コマンドを使用して、変更するコマンドの複製をユーザー・ライブラリーに作成してください。こうすることにより、システムの他のユーザーは、必要な場合には IBM 提供のデフォルト値を使用できます。

システム・ライブラリー・リスト変更 (**CHGSYSLIBL**) コマンドを用いて、ユーザー・ライブラリーを QSYS またはその他のすべてのシステム提供のライブラリーより前に移してください。 これによってユーザーは、ライブラリー修飾子を指定せずに、変更されたコマンドを使用できるようになります。

システム全体の規模で必要なコマンドに変更を加える場合は、ユーザー・ライブラリー内で変更を行います。 そして、そのユーザー・ライブラリー名が QSYS より前にくるように、QSYSLIBL システム値に追加しなければなりません。 変更されたコマンドはシステム全体で使用されます。 IBM 提供のデフォルト値を使用するアプリケーション・プログラムを実行する必要がある場合には、 **CHGSYSLIBL** コマンドを使用します。このことを行うと、影響を受けるコマンドの特殊ライブラリー修飾またはライブラリー修飾を除去することができます。

2. ライセンス・プログラムの新しいリリースを導入する場合には、そのライセンス・プログラムに対する IBM 提供のコマンドはすべて、システム内部で新しいリリースによって置き換えられます。 新しいリリースを導入した時点で CL プログラムを使用してコマンドに変更を加える必要があります。 このようにして、CL プログラムを実行して、新しいコマンドの複製を作成して新しいキーワードを組み込み、コマンドのデフォルト値を変更することができます。

   IBM 提供のコマンドに新しいキーワードが追加されている場合には、前のリリースのコマンドの複写は正しく実行できないことがあります。

   次に示すのは、古いバージョンを削除して新しい変更されたコマンドを作成するために使用する CL プログラムの例です。

   注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
DLTCMD USRQSYS/SIGNOFF
CRTDUPOBJ OBJ(SIGNOFF) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS) NEWOBJ(*SAME)
CHGCMDDFT CMD(USRQSYS/SIGNOFF) NEWDFT('LOG(*LIST)')
 .
 .
変更する各コマンドに対して DLTCMD、CRTDUPOBJ、および
CHGCMDDFT を繰り返します
 .

 .
ENDPGM
```

新しいリリースを導入するときに、使用する CL コマンドのデフォルト値に対して加える変更をトラックすることができます。変更をトラックするには、出口点 QIBM_QCA_RTV_COMMAND 用の出口プログラムを登録してください。出口プログラムは、**CHGCMDDFT** コマンドを実行するときに呼び出されます。出口プログラムに渡されるパラメーターの 1 つに、実行されているコマンド・ストリングがあります。コマンド・ストリングをソース・ファイルに保管してから、ソース・ファイルを CL プログラムにコンパイルすることができます。最後に、このプログラムを使用して、前のリリースの使用中にコマンドのデフォルト値に対して加えた変更を再作成します。

以下のステップを使用して、**CHGCMDDFT** コマンドの NEWDFT コマンド・ストリングを組み立てることができます。この例では USRQSYS/CRTCLPGM コマンドが使用されています。

1. 以下のコマンドによって、変更したいコマンドの複製をユーザー・ライブラリーに作成します。

```
CRTDUPOBJ OBJ(CRTCLPGM) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS)  NEWOBJ(*SAME)
```

2. 変更したいコマンドの名前を原始ステートメント入力ユーティリティー (SEU) で参照されるソース・ファイルに入れます。

3. F4 キーを押して、コマンドのプロンプターを呼び出します。
4. 変更したいキーワードの新しいデフォルト値を入力します。 この例では、AUT(*EXCLUDE) および TEXT('Isn''t this nice text') を入力します。
5. 必須キーワードにはデフォルト値を指定することはできません。ただし、コマンド・ストリングをソース・ファイルに入れるには、各必須キーワードに対して有効な値を指定しなければなりません。ここでは、PGM パラメーターに PGM1 を指定しています。
6. 実行キーを押して、コマンド・ストリングをソース・ファイルに入れます。 次のようなコマンド・ストリングが戻されます。

   ```
   USRQSYS/CRTCLPGM PGM(PGM1) AUT(*EXCLUDE) +
   TEXT('Isn''t this nice text')
   ```

7. コマンド・ストリングから必須キーワードを除去します。

   ```
   USRQSYS/CRTCLPGM  AUT(*EXCLUDE) +
   TEXT('Isn''t this nice text')
   ```

   変更できるのは、既存のデフォルト値を備えたパラメーター、要素、または修飾子だけである点に注意してください。既存のデフォルト値がないパラメーター、要素、または修飾子に値を指定しても、デフォルト値は変更されません。
8. 次の例で示すように、**CHGCMDDFT** コマンドを先頭に入れてください。

   ```
   CHGCMDDFT USRQSYS/CRTCLPGM AUT(*EXCLUDE) +
   TEXT('Isn''t this nice text')
   ```

9. NEWDFT キーワードに対する入力は、次の例で示すように引用符で囲まなければなりません。

   ```
   CHGCMDDFT USRQSYS/CRTCLPGM 'AUT(*EXCLUDE) +
   TEXT('Isn''t this nice text')'
   ```

10. NEWDFT の値の一部として単一引用符が使用されているので、正しく実行するには各引用符をさらに二重にしなければなりません。

    ```
    CHGCMDDFT USRQSYS/CRTCLPGM 'AUT(*EXCLUDE) +
    TEXT(''Isn''''t this nice text'')'
    ```

11. F4 キーを押してコマンド・プロンプターを呼び出し、次に F11 キーを押してキーワードのプロンプトを要求すると、次に示すような画面が表示されます。

    ```
    コマンド  . . . . . :   CMD         >   CRTCLPGM
      ライブラリー  . . . . . :                 USRQSYS
    新しい省略時パラメーター・ストリング:   NEWDFT      >   'AUT(*EXCLUDE)
    TEXT(''Isn''''t this nice text'')'
    ```

12. 実行キーを押すと、**CHGCMDDFT** コマンド・ストリングは次のようになります。

    ```
    CHGCMDDFT CMD(USRQSYS/CRTCLPGM) NEWDFT('AUT(*EXCLUDE) +
              TEXT(''Isn''''t  this nice text'')')
    ```

13. F3 キーを押して SEU を終了し、この CL プログラムまたは CL プロシージャーを作成し、実行してください。
14. USRQSYS/CRTCLPGM は、AUT のデフォルト値として *EXCLUDE および TEXT のデフォルト値として 'Isn''t this nice text' を持つことになります。

関連情報:

CL コマンド検索プログラム

アプリケーション・プログラミング・インターフェース

例: *CL* コマンドのデフォルト変更:

この例では、CL コマンドのデフォルト値を変更する方法を示します。

- コマンド**物理ファイル作成 (CRTPF)** の MAXMBRS キーワードのデフォルト値を *NOMAX にするには、以下のように指定してください。

```
CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(1)
  .
  .
CHGCMDDFT CMD(CRTPF) NEWDFT('MAXMBRS(*NOMAX)')
```

- コマンド**物理ファイル作成 (CRTPF)** の MAXMBRS キーワードのデフォルト値を 10 にするには、以下のように指定してください。

```
CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(*NOMAX)
  .
  .
CHGCMDDFT CMD(CRTPF) NEWDFT('MAXMBRS(10)')
```

- 以下の指定により、コマンド**制御言語プログラム作成 (CRTCLPGM)** の SRCFILE キーワードの最初の修飾子のデフォルト値は LIB001 になり、2 番目の修飾子のデフォルトの値は FILE001 になります。AUT キーワードの新しいデフォルト値は *EXCLUDE になります。

```
CRTCLPGM PGM(PROGRAM1) SRCFILE(*LIBL/QCMDSRC)
  .
  .
CHGCMDDFT CMD(CRTCLPGM) +
          NEWDFT('SRCFILE(LIB001/FILE001) AUT(*EXCLUDE)')
```

- 以下の指定により、コマンド**ジョブ変更 (CHGJOB)** の、PRTTXT キーワードのデフォルトの値は 'Isn't this print text' になります。NEWDFT キーワードには単一引用符が組み込まれているため、これらの単一引用符を二重にしないでください。二重にすると、処理は正常に実行されません。

```
 CHGJOB PRTTXT('Isn''t this print text')
  .
  .
CHGCMDDFT CMD(CHGJOB) +
          NEWDFT('PRTTXT(''Isn''''t this print text'')')
```

- 以下の指定により、コマンド**論理ファイル作成 (CRTLF)** DTAMBRS キーワードの最初のリスト項目の最初の修飾子 (ライブラリー名) のデフォルトの値は QGPL になります。DTAMBRS キーワードの 2 番目のリストの要素 (メンバー名) の新しいデフォルト値は MBR1 です。

```
CRTLF FILE(FILE1) DTAMBRS(*ALL)
  .
  .
CHGCMDDFT CMD(CRTLF) +
          NEWDFT('DTAMBRS((QGPL/*N (MBR1)))')
```

  *ALL は、DTAMBRS のリスト全体に対する SNGVAL (単一値) なので、ライブラリー名のデフォルト値 *CURRENT とメンバー名のデフォルト値 *NONE は、最初のコマンド・プロンプト表示画面には表示されません。ライブラリー名のデフォルト値 *CURRENT およびメンバー名のデフォルト値 *NONE は新しいデフォルト値に変更できますが、*ALL は DTAMBRS のリスト全体に対する単一値 (SNGVAL) なので、*CURRENT と *NONE は最初のプロンプト表示画面には表示されません。

- ジョブのスプール・ファイルを表示するコマンドを作成するには、以下のようにします。

```
CRTDUPOBJ OBJ(WRKJOB) FROMLIB(QSYS) +
          TOLIB(MYLIB) NEWOBJ(WRKJOBSPLF)
WRKJOBSPLF  OPTION(*SPLF)
  .
  .
CHGCMDDFT CMD(MYLIB/WRKJOBSPLF) +
          NEWDFT('OPTION(*SPLF)')
```

CL コマンドのコマンド処理プログラム**:**

コマンド処理プログラム (CPP) は、コマンドを処理するプログラムです。このプログラムは、複数の妥当性検査を実行し、要求された機能が実行されるようにコマンドを処理します。

コマンド処理プログラム (CPP) としては CL プログラム、高水準言語 (HLL) プログラム、または REXX プロシージャーを使用できます。 CL や HLL で書かれているプログラムは、**呼び出し (CALL)** コマンドで直接呼び出すこともできます。REXX プロシージャーは、**REXX プロシージャー開始 (STRREXPRC)** コマンドを使用して直接呼び出すことができます。 **コマンド作成 (CRTCMD)** コマンドの実行時にコマンド処理プログラムが存在している必要はありません。ライブラリー修飾子として *LIBL を使用している場合、作成されたコマンドの実行時のコマンド処理プログラムの探索にはライブラリー・リストが使用されます。

コマンド処理プログラムの実行結果として出されたメッセージは、ジョブ・メッセージ待ち行列に送って、自動的に表示または印刷できます。また、要求元のディスプレイ装置にその画面を表示することもできます。

注**:**

1. コマンドに定義されたパラメーターは、定義されている順序 (PARM ステートメントの順序) に従って個別に渡されます。
2. 10 進数値は、PARM ステートメントに定義された長さのパック 10 進数値として、HLL プログラムまたは CL プログラムに渡されます。
3. 文字、名前、および論理値は、PARM ステートメントに定義された長さの文字ストリングとして、HLL プログラムまたは CL プログラムに渡されます。

関連情報:

コマンド作成 (CRTCMD) コマンド

プログラム呼び出し (CALL) コマンド

REXX プロシージャー開始 (STRREXPRC) コマンド

CL または ***HLL*** コマンド処理プログラム*:*

ここでは、CL または別の高水準言語 (HLL) で作成されたコマンド処理プログラムについて説明します。

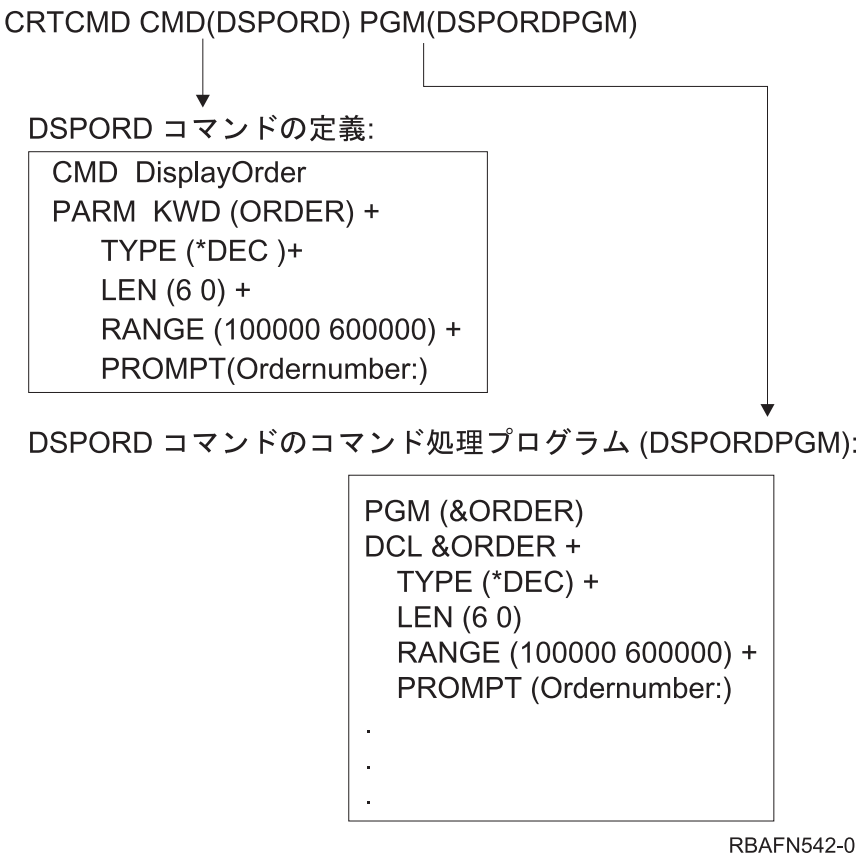


図 *4. CL* および *HLL* の場合のコマンドの相互関係

コマンド処理プログラムが CL で書かれたプログラムである場合、パラメーターの値を受け取る変数は、各 PARM ステートメントに指定されたタイプおよび長さに従って宣言しなければなりません。以下の表は、この対応関係を示しています。

PARM ステートメントのタイプ	**PARM** ステートメントの長さ	宣言された変数のタイプ	宣言された変数の長さ
*DEC	x y[1]	*DEC	x y[1]
*LGL	1	*LGL	1
*CHAR	n	*CHAR	≤n^2
*NAME	n	*CHAR	≤n^2
*CNAME	n	*CHAR	≤n^2
*SNAME	n	*CHAR	≤n^2
*GENERIC	n	*CHAR	≤n^2
*CMDSTR	n	*CHAR	≤n^2
*DATE	7	*CHAR	7
*TIME	6	*CHAR	6
*INT2	n	*INT または *CHAR	2
*INT4	n	*INT または *CHAR	4
*UINT2	n	*UINT または *CHAR	2
*UINT4	n	*UINT または *CHAR	4

PARM ステートメントのタイプ	**PARM** ステートメントの長さ	宣言された変数のタイプ	宣言された変数の長さ
注: [1] x は長さに等しく、y は小数点以下の桁数に等しい値です。 [2] 文字変数の場合には、渡される値の長さが宣言された長さより大きい場合、宣言された長さに合わせて値が切り捨てられます。 RTNVAL(*YES) を指定した場合には、宣言された長さと PARM ステートメントに定義された長さが等しくなければなりません。			

コマンド処理プログラムとして使用される、CL で作成されたプログラムでは、2 進数値 (*INT2 または *INT4 など) を処理できます。このプログラムでは、これらの値を文字フィールドとして受け取ることができます。その場合は、2 進数組み込み関数 (%BINARY) を使用して、その値を 10 進数に変換できます。それ以外の場合、CL プログラムはそれらを整変数として宣言できます。

*INT2 または *INT4 および *UINT2 または *UINT4 との間の相違は、 *INT2 タイプと *INT4 タイプは符号付き整数であり、 *UINT2 タイプと *UINT4 タイプは符号なし整数であることです。すべての整数タイプのデフォルト値は 0 です。*UINT2 タイプと *UINT4 タイプには、 *INT タイプと *INT4 タイプと同じ制約事項があります。

注**:** %BINARY 組み込み関数は、符号付き整数とともに使用するためのものです。符号なし整数に対応する関数はありません。

関連タスク:

378 ページの『例: CL コマンドの定義および作成』
これらの例は、CL コマンドを定義および作成する方法を示しています。

関連情報:

コマンド作成 (CRTCMD) コマンド

CL コマンドの ***REXX*** コマンド処理プロシージャー**:**

ここでは、CL コマンドの REXX コマンド処理プロシージャーについて説明します。

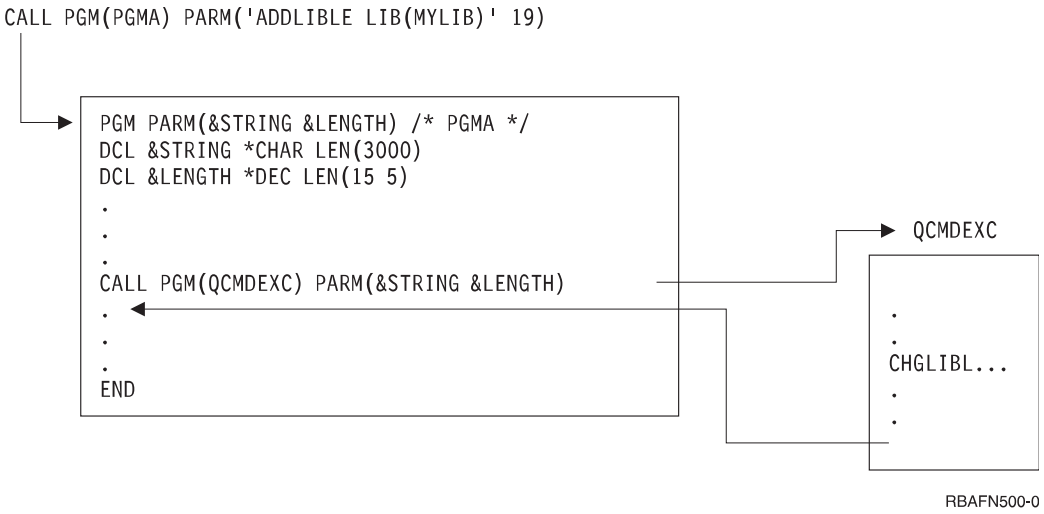


図 *5. REXX* の場合のコマンドの相互関係

関連情報:
コマンド作成 (CRTCMD) コマンド

CL コマンドの妥当性検査プログラム**:**

構文エラーを検出して、コマンドの診断メッセージを送信するには、妥当性検査プログラムを作成します。

ユーザー・コマンドに対する妥当性検査プログラムを作成するには、 **コマンド作成 (CRTCMD)** コマンドの VLDCKR パラメーターに、その妥当性検査プログラムの名前を指定してください。

妥当性検査プログラムは、**コマンド作成 (CRTCMD)** コマンドの実行時に存在している必要はありません。ライブラリー修飾子として *LIBL を指定している場合、作成されたコマンドの実行時の妥当性検査プログラムの探索にはライブラリー・リストが使用されます。

妥当性検査プログラムに関して以下の 2 つの考慮事項があります。

- 妥当性検査プログラムは、コマンドの構文が正しい場合にだけ呼び出されます。すべてのパラメーターは、コマンド処理プログラムへの受け渡しの場合と同様に妥当性検査プログラムに渡されます。
- 妥当性検査プログラムを使用してパラメーターを変更しないでください。これは、変更された値はコマンド処理プログラムに渡されない場合があるからです。

この項では、CL で作成された妥当性検査プログラムからのメッセージの送信方法について説明します。

妥当性検査プログラムはエラーを検出した場合、先行の呼び出しに対して診断メッセージを送り、次にエスケープ・メッセージ CPF0002 を送るようにコーディングしなければなりません。例えば、口座番号が正しくないことを示すメッセージが必要な場合には、メッセージ・ファイルに次のようなメッセージ記述を追加します。

```
ADDMSGD      MSG('Account number &2 no longer valid') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 6))
```

置換変数 &1 は、メッセージの中にはなく、FMT パラメーターに長さ 4 の文字として定義されている点に注意してください。 &1 はシステムによる使用のために予約されており、必ず 4 文字でなければなりません。 置換変数 &1 がメッセージに定義された唯一の置換変数である場合には、メッセージの送信時点でメッセージ・データの 4 番目のバイトに決してブランクが入らないようにしなければなりません。

このメッセージは、妥当性検査プログラムに次のように指定することによって、システムに送ることができます。

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' vv &ACCOUNT) MSGTYPE(*DIAG)
```

妥当性検査プログラムは、必要な診断メッセージをすべて送り出した後で、メッセージ CPF0002 を送らなければなりません。メッセージ CPF0002 を送るための**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドは、以下のようになります。

```
SNDPGMMSG    MSGID(CPF0002) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

システムはメッセージ CPF0002 を受け取ると、その呼び出し元のプログラムにメッセージ CPF0001 を送ってエラーが検出されたことを伝えます。

メッセージ CPD0006 は、ユーザー定義の妥当性検査プログラムで使用できるように定義されています。このメッセージ・データとして即時メッセージを送ることができます。以下の例では、メッセージの前に必ず 4 文字のゼロが置かれることに注意してください。

次に示すのは、妥当性検査プログラムの例です。

```
PGM PARM(&PARM01)
DCL VAR(&PARM01) TYPE(*CHAR) LEN(10)
IF COND(&PARM01 *EQ 'ERROR') THEN(DO)
SNDPGMMSG MSGID(CPD0006) MSGF(QCPFMSG) +
          MSGDTA('0000 DIAGNOSTIC MESSAGE FROM USER-DEFINED +
          VALIDITY CHECKER INDICATING THAT PARM01 IS IN ERROR.') +
          MSGTYPE(*DIAG)
SNDPGMMSG MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
ENDDO
ELSE
 .
 .
 .
ENDPGM
```

関連概念:

317 ページの『CL コマンドの妥当性検査』
システムは、コマンドの妥当性検査を行います。妥当性検査プログラムは必須ではありませんが、必要ならユーザー固有のプログラムを作成することもできます。

関連情報:

プログラム・メッセージ送信 (SNDPGMMSG) コマンド

コマンド作成 (CRTCMD) コマンド

例: **CL** コマンドの定義および作成:

これらの例は、CL コマンドを定義および作成する方法を示しています。

関連タスク:

374 ページの『CL または HLL コマンド処理プログラム』
ここでは、CL または別の高水準言語 (HLL) で作成されたコマンド処理プログラムについて説明します。

例: アプリケーション・プログラムの呼び出し:

この例では、アプリケーション・プログラムを呼び出す CL コマンドを定義および作成する方法を示します。

アプリケーション・プログラムを呼び出すためのコマンドを作成できます。アプリケーション・プログラムを呼び出すためのコマンドを作成した場合、そのプログラムに渡されるパラメーターの妥当性検査は IBM i オペレーティング・システムにより行われます。ただし、**プログラム呼び出し (CALL)** コマンドを使用してアプリケーション・プログラムを呼び出した場合には、妥当性検査はアプリケーション・プログラムで実行しなければなりません。

例えば、宛名ラベル作成プログラム (LBLWRT) は、1 部または 2 部複写のどちらかのタイプの用紙に、特定の顧客用の宛名ラベルを指定枚数だけ印刷するとします。 LBLWRT プログラムは実行時に 3 つのパラメーター、すなわち顧客番号、宛名ラベルの数、および用紙のタイプ (ONE または TWO) を必要とします。

このプログラムがディスプレイ装置から直接呼び出される場合は、2 番目のパラメーターがプログラムにとって正しくない形式になります。 CALL コマンドの数値定数は、常に小数部分が 5 桁で全体が 15 桁に

なりますが、LBLWRT プログラムでは少数部分のない 3 桁の数値を必要としているからです。そこで、プログラムで必要とする形式でデータを用意するコマンドを作成できます。

LBLWRT プログラムを呼び出すためのコマンドのコマンド定義ステートメントは以下のとおりです。

```
CMD PROMPT('Label Writing Program')
PARM  KWD(CUSNBR) TYPE(*CHAR) LEN(5) MIN(1) +
      PROMPT('Customer Number')
PARM  KWD(COUNT) TYPE(*DEC) LEN(3) DFT(20) RANGE(10 150) +
      PROMPT('Number of Labels')
PARM  KWD(FRMTYP) TYPE(*CHAR) LEN(3) DFT('TWO') RSTD(*YES) +
      SPCVAL(('ONE') ('TWO') ('1' 'ONE') ('2' 'TWO')) +
      PROMPT('Form Type')
```

2 番目のパラメーター COUNT はデフォルト値として 20 が指定されており、また RANGE パラメーターにラベル数として入力できる値が 10 から 150 までの範囲の値だけに限定されています。

3 番目のパラメーター FRMTYP には、ディスプレイ装置ユーザーが SPCVAL パラメーターの指定により 'ONE'、'TWO'、'1'、または '2' を入力できます。プログラムは 'ONE' または 'TWO' の値を受け取ることを想定していますが、ディスプレイ装置ユーザーが '1' または '2', を入力した場合には、FRMTYP パラメーターに対する必要な置き換えが、このコマンドにより行われます。

このコマンドの処理プログラムは、アプリケーション・プログラム LBLWRT です。アプリケーション・プログラムが RPG プログラムの場合には、以下のパラメーターを受け取るために以下の指定がプログラムに必要になります。

```
*ENTRY   PLIST
         PARM    CUST   5
         PARM    COUNT  30
         PARM    FORM   3
```

コマンド作成 (CRTCMD) コマンドは以下のとおりです。

```
CRTCMD CMD(LBLWRT) PGM(LBLWRT) SRCMBR(LBLWRT)
```

関連情報:

コマンド作成 (CRTCMD) コマンド

例: デフォルト値の置換:

この例では、IBM 提供の CL コマンドのデフォルト値を指定する CL コマンドを定義および作成する方法を示します。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

IBM 提供のコマンドにデフォルト値を用意し、ディスプレイ装置ユーザーにとって必要な入力操作を軽減するためのコマンドを作成できます。例えば、テープを初期設定し、ライブラリーをテープ装置 TAPE1 に保管するための、**テープへのライブラリーの保管 (SAVLIBTAP)** コマンドを作成できます。このコマンドは、標準の**ライブラリー保管 (SAVLIB)** コマンドのパラメーターにデフォルト値を使用するため、ディスプレイ装置ユーザーはライブラリー名の指定だけが必要になります。

テープへのライブラリーの保管 (SAVLIBTAP) コマンドのコマンド定義ステートメントは以下のとおりです。

```
CMD PROMPT('Save Library to Tape')
PARM KWD(LIB) TYPE(*NAME) LEN(10) MIN(1) +
  PROMPT('Library Name')
```

コマンド処理プログラムは以下のとおりです。

```
PGM PARM(&LIB)
DCL &LIB TYPE(*CHAR) LEN(10)
INZTAP DEV(TAPE1) CHECK(*NO)
SAVLIB LIB(&LIB) DEV(TAPE1)
ENDPGM
```

コマンド作成 (CRTCMD) コマンドは以下のとおりです。

```
CRTCMD CMD(SAVLIBTAP) PGM(SAVLIBTAP) SRCMBR(SAVLIBTAP)
```

関連情報:

コマンド作成 (CRTCMD) コマンド

例: 出力待ち行列の表示:

この例は、出力待ち行列を表示する CL コマンドを定義および作成する方法を示しています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

出力待ち行列 PGMR を表示することをデフォルト値とする出力待ち行列を表示するためのコマンドを作成できます。ディスプレイ装置ユーザーは、次の**出力待ち行列表示 (DSPOQ)** コマンドを使用して、ライブラリー・リストに含まれている任意の出力待ち行列を表示することができ、印刷オプションも提供されます。

出力待ち行列表示 (DSPOQ) コマンドのコマンド定義ステートメントは以下のとおりです。

```
CMD PROMPT('WRKOUTQ.-Default to PGMR')
PARM  KWD(OUTQ) TYPE(*NAME) LEN(10) DFT(PGMR) +
      PROMPT('Output queue')
PARM  KWD(OUTPUT) TYPE(*CHAR) LEN(6) DFT(*) RSTD(*YES)
      VALUES(* *PRINT) PROMPT('Output')
```

2 番目の PARM ステートメントの RSTD パラメーターは、指定する値が値のリストの中のいずれかの値でなければならないことを定義しています。

出力待ち行列表示 (DSPOQ) コマンドのコマンド処理プログラムは以下のとおりです。

```
PGM PARM(&OUTQ &OUTPUT)
DCL &OUTQ TYPE(*CHAR) LEN(10)
DCL &OUTPUT TYPE(*CHAR) LEN(6)
WRKOUTQ OUTQ(*LIBL/&OUTQ) OUTPUT(&OUTPUT)
ENDPGM
```

コマンド作成 (CRTCMD) コマンドは以下のとおりです。

```
CRTCMD CMD(DSPOQ) PGM(DSPOQ) SRCMBR(DSPOQ)
```

次に示すコマンド DSPOQ1 は、上述のコマンドを変形したものです。ワークステーション・ユーザーは DSPOQ1 コマンドを使用して、出力待ち行列名として修飾名を入力できます。また、ライブラリー名のデフォルト値として、このコマンドでは *LIBL が使用されています。

DSPOQ1 コマンドのコマンド定義ステートメントは以下のとおりです。

```
        CMD PROMPT('WRKOUTQ.-Default to PGMR')
        PARM    KWD(OUTQ) TYPE(QUAL1) +
                PROMPT('Output queue:')
        PARM    KWD(OUTPUT) TYPE(*CHAR) LEN(6) RSTD(*YES) +
                VALUES(* *PRINT) DFT(*) +
```

```
                PROMPT('Output')
QUAL1:   QUAL TYPE(*NAME) LEN(10) DFT(PGMR)
         QUAL TYPE(*NAME) LEN(10) DFT(*LIBL) +
         SPCVAL(*LIBL)
```

QUAL ステートメントは、ユーザーが OUTQ パラメーターに入力できる修飾名を定義するために使用されています。ユーザーが名前を入力しなかった場合には、*LIBL/PGMR が使用されます。 SPCVAL パラメーターが使用されているのは、すべてのライブラリー名は有効な名前としての規則 (例えば、A から Z で始まるなどの規則) に従っていなければなりませんが、値 *LIBL はこのような規則からはずれているからです。 SPCVAL パラメーターは IBM i オペレーティング・システムに、*LIBL が入力された場合には名前の妥当性に関する規則を無視するように指示します。

DSPOQ1 コマンドのコマンド処理プログラムは以下のとおりです。

```
PGM PARM(&OUTQ &OUTPUT)
DCL &OUTQ TYPE(*CHAR) LEN(20)
DCL &OBJNAM TYPE(*CHAR) LEN(10)
DCL &LIB TYPE(*CHAR) LEN(10)
DCL &OUTPUT TYPE(*CHAR) LEN(6)
CHGVAR &OBJNAM %SUBSTRING(&OUTQ 1 10)
CHGVAR &LIB %SUBSTRING(&OUTQ 11 10)
WRKOUTQ OUTQ(&LIB/&OBJNAM) OUTPUT(&OUTPUT)
ENDPGM
```

修飾名は、20 文字の変数としてコマンドから渡されているので、このプログラムでは、サブストリング組み込み関数 (%SUBSTRING または %SST) を使用して、修飾名を適切な CL 構文に組み立てています。

関連情報:

コマンド作成 (CRTCMD) コマンド

例: ***IBM*** 提供のコマンドからのメッセージの表示**:**

この例は、メッセージを再表示する CL コマンドを定義および作成する方法を示しています。

出力待ち行列消去 (CLROUTQ) コマンドは、完了メッセージ CPF3417 を発行します。このメッセージには、削除された項目の数、削除されなかった項目の数、および出力待ち行列の名前が示されます。 **出力待ち行列消去 (CLROUTQ)** コマンドをコマンド処理プログラム内で実行した場合にも同じメッセージが出されますが、コマンド処理プログラムが直接発行するメッセージではないので詳細メッセージとなります。例えば、ユーザー定義の**出力待ち行列消去 (CLROUTQ)** コマンドをプログラマー・メニューから発行した場合には、このメッセージは表示されません。ただし、IBM 提供のメッセージを受け取って、それをコマンド処理プログラムから発行し直すことができます。

例えば、出力待ち行列 QPRINT2 を消去するための、CQ2 という名前のコマンドを作成するとします。

CQ2 コマンドのコマンド定義ステートメントは以下のとおりです。

```
CMD PROMPT ('Clear QPRINT2 output queue')
```

コマンド作成 (CRTCMD) コマンドは以下のとおりです。

```
CRTCMD CMD(CQ2) PGM(CQ2)
```

コマンド処理プログラムは以下のとおりです。このプログラムは完了メッセージを受け取り、それを表示します。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM /* Clear QPRINT2 output queue CPP */
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(100)
CLROUTQ QPRINT2
RCVMSG MSGID(&MSGID) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
ENDPGM
```

メッセージ CPF3417 の MSGDTA の長さは 28 バイトです。ただし、使用しない桁はすべて無視されるので、変数 &MSGDTA を 100 バイトとして定義することにより、これと同じ方法を大部分のメッセージについて使用できます。

関連情報:

コマンド作成 (CRTCMD) コマンド

例*:* ジョブ変更簡略 ***CL*** コマンドの作成*:*

この例は、簡略コマンドを定義および作成する方法を示しています。

注*:* コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

ユーザー独自の簡略コマンドを作成することによって、IBM 提供のコマンドを簡素化したり、ユーザーが使用できるパラメーターを制限できます。例えば、ユーザー独自の**ジョブの変更 (CJ)** コマンドを作成して、印刷装置のパラメーターだけを変更できます。以下に、ユーザー独自の**ジョブの変更 (CJ)** コマンドを作成し、実装するためのステップを示します。

- ステップ 1: コマンド定義ソース・ステートメント

```
CMD   PROMPT('Change Job')

PARM  KWD(PRTDEV) +
      TYPE(*NAME) +
      LEN(10)  +
      SPCVAL(*SAME *USRPRF *SYSVAL *WRKSTN) +
      PROMPT('Printer Device')
```

- ステップ 2: 処理プログラム

```
PGM PARM(&PRTDEV)
DCL VAR(&PRTDEV)  TYPE(*CHAR)  LEN(10)
CHGJOB  PRTDEV(&PRTDEV)
ENDPGM
```

- ステップ 3: **コマンド作成 (CRTCMD)** コマンド

```
CRTCMD  CMD(CJ)  PGM(CJ)  SRCMBR(CJ)
```

関連情報:

コマンド作成 (CRTCMD) コマンド

例*:* 印刷装置書き出しプログラムの簡略 ***CL*** コマンドの作成*:*

この例は、簡略 CL コマンドを定義および作成する方法を示しています。

注*:* コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

印刷装置書き出しプログラム W1 を開始するための DW1 という簡略コマンドは、次のようにして作成できます。

コマンド定義ステートメントは以下のとおりです。

```
CMD /* Start printer writer command */
```

コマンド処理プログラムは以下のとおりです。

```
PGM
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(W1)
ENDPGM
```

コマンド作成 (CRTCMD) コマンドは以下のとおりです。

```
CRTCMD CMD(DW1) PGM(DW1) SRCMBR(DW1)
```

関連情報:

コマンド作成 (CRTCMD) コマンド

例: ファイルおよびソース・メンバーの削除のための *CL* コマンド:

この例は、ファイルおよびソース・メンバーを削除する CL コマンドの定義および作成方法を示しています。

ファイルとそれに対応する QDDSSRC 内のソース・メンバーを削除するための CL コマンドを作成できます。

ファイルおよびソースの削除 (DFS) コマンドのコマンド定義ステートメントは以下のとおりです。

```
CMD PROMPT('Delete File and Source')
PARM KWD(FILE) TYPE(*NAME) LEN(10) PROMPT('File Name')
```

コマンド処理プログラムは、削除したいファイルの名前とソース・ファイル・メンバーの名前が同じであることを前提として書かれています。また、このプログラムでは、削除したいファイルとソース・ファイルの両方がライブラリー・リストにあることが前提になっています。このプログラムでファイルを削除できなかった場合は通知メッセージが送られ、ソース・メンバーの除去がコマンドにより試みられます。ソース・メンバーが存在していない場合にはエスケープ・メッセージが送られます。

これは、コマンド処理プログラムです。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM PARM(&FILE)
DCL &FILE TYPE(*CHAR) LEN(10)
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(80)
DCL &SRCFILE TYPE(*CHAR) LEN(10)
MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* CATCH ALL */
DLTF &FILE
MONMSG MSGID(CPF2105) EXEC(DO) /* NOT FOUND */
RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
      MSGDTA(&MSGDTA)
GOTO TRYDDS
ENDDO
RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
    /* DELETE FILE COMPLETED */
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
          MSGDTA(&MSGDTA) /* TRY IN QDDSSRC FILE */
TRYDDS:   CHKOBJ QDDSSRC OBJTYPE(*FILE) MBR(&FILE)
          RMVM QDDSSRC MBR(&FILE)
          CHGVAR &SRCFILE 'QDDSSRC'
          GOTO END
```

```
END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
              /* REMOVE MEMBER COMPLETED */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
          MSGDTA(&MSGDTA)
          RETURN
ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
              /* ESCAPE MESSAGE */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
               MSGDTA(&MSGDTA)
ENDPGM
```

例*:* プログラム・オブジェクトの削除のための *CL* コマンド*:*

この例は、プログラム・オブジェクトおよびソース・メンバーを削除する CL コマンドを定義および作成する方法を示しています。

高水準言語 (HLL) のプログラムとそれに対応するソース・メンバーを削除するコマンドを作成できます。

DPS と名付けた、このようなコマンドのコマンド定義ステートメントは以下のとおりです。

```
CMD PROMPT ('Delete Program and Source')
PARM KWD(PGM) TYPE(*NAME) LEN(10) PROMPT('Program Name')
```

これに対応するコマンド処理プログラムは、削除したいプログラムの名前とそのソース・ファイル・メンバーの名前が同じであることを前提として書かれています。 さらに、IBM 提供のソース・ファイル (QCLSRC、QRPGSRC、および QCBLSRC) を使用しなければなりません。また、このコマンド処理プログラムでは、削除したいプログラムとソース・ファイルの両方がライブラリー・リストにあることが前提になっています。 プログラムをオープンできない場合は、システムは通知メッセージを送信し、ソース・メンバーの除去がコマンドにより試みられます。 ソース・メンバーが存在していない場合にはシステムはエスケープ・メッセージを送ります。 これは、コマンド処理プログラムです。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM PARM(&PGM)
DCL &PGM TYPE(*CHAR) LEN(10)
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(80)
DCL &SRCFILE TYPE(*CHAR) LEN(10)
MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* CATCH ALL */
DLTPGM &PGM
MONMSG MSGID(CPF2105) EXEC(DO) /* NOT FOUND*/
RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
      MSGDTA(&MSGDTA)
GOTO TRYCL  /* TRY TO DELETE SOURCE MEMBER */
ENDDO
RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
    /* DELETE PROGRAM COMPLETED */
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
     MSGDTA(&MSGDTA) /* TRY IN QCLSRC */
TRYCL:   CHKOBJ QCLSRC OBJTYPE(*FILE) MBR(&PGM)
         MONMSG MSGID(CPF9815) EXEC(GOTO TRYRPG) /* NO CL MEMBER */
         RMVM QCLSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QCLSRC'
         GOTO END
TRYRPG:   /* TRY IN QRPGSRC FILE */
         CHKOBJ QRPGSRC OBJTYPE(*FILE) MBR(&PGM)
         MONMSG MSGID(CPF9815) EXEC(GOTO TRYCBL) /* NO RPG MEMBER */
         RMVM QRPGSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QRPGSRC'
         GOTO END
```

```
TRYCBL:   /* TRY IN QCBLSRC FILE */
         CHKOBJ QCBLSRC OBJTYPE(*FILE) MBR(&PGM)
         /* ON LAST SOURCE FILE LET CPF0000 OCCUR FOR A NOT FOUND +
         CONDITION */
         RMVM QCBLSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QCBLSRC'
         GOTO END
TRYNXT:   /* INSERT ANY ADDITIONAL SOURCE FILES */
          /* ADD MONMSG AFTER CHKOBJ IN TRYCBL AS WAS +
            DONE IN TRYCL AND TRYRPG */
END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
               /*REMOVE MEMBER COMPLETED */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
             MSGDTA(&MSGDTA)
          RETURN
ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
              /* ESCAPE MESSAGE */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
               MSGDTA(&MSGDTA)
          ENDPGM
```

CL コマンドの文書化

独自の CL コマンドを定義および作成する場合は、コマンドを説明するオンライン・コマンド・ヘルプを作成することもできます。

関連概念:

48 ページの『CL コマンド情報および文書』
IBM は、CL コマンドに関する文書を提供しています。さらに、独自のコマンドについての文書を作成することもできます。

関連タスク:

312 ページの『CL コマンドの定義』
CL コマンドを使用することによって、ユーザーは、広範囲にわたる機能をシステムに要求できます。IBM 提供のコマンドを使用して、コマンド・パラメーター用のデフォルト値を変更し、ユーザー独自のコマンドを定義することができます。

関連資料:

313 ページの『CL コマンド定義ステートメント』
コマンド定義ステートメントを使用することにより、システムのユーザーはアプリケーションの特殊な要件に対応する新たな CL コマンドを作成することができます。

CL コマンドおよびコマンド・オンライン・ヘルプ:

コマンド・プロンプトとオンライン・コマンド・ヘルプは、CL コマンドの強力な機能です。

1 つ以上の独自の CL コマンドを開発した場合、IBM の CL コマンドで使用されるすべてのコマンド・プロンプト機能をそのコマンドに使用できます。同じことがコマンド・ヘルプに対しても当てはまります。IBM 提供のコマンドに対して用意されているヘルプ用の同じ機能も使用でき、オンライン・ヘルプを作成できます。

最初のステップは、コマンドとコマンド・ヘルプの間の接続の仕組みを理解することです。

- コマンド・ヘルプ情報はパネル・グループ・オブジェクトに保管されます。パネル・グループのシンボリック・オブジェクト・タイプは *PNLGRP です。ヘルプ・パネル・グループはヘルプ・モジュールで構成されています。各ヘルプ・モジュールにはヘルプ・モジュール名があります。

- **コマンド作成 (CRTCMD)** コマンドには、コマンド (*CMD) オブジェクトとオンライン・ヘルプ・パネル・グループを接続する 2 つのパラメーター、**HLPID** (ヘルプ ID) および **HLPPNLGRP** (ヘルプ・パネル・グループ) があります。
- オンライン・ヘルプ・パネル・グループには、以下の 4 つのタイプのコマンド・ヘルプ・モジュールがあります。
  1. コマンド・レベル・ヘルプ・モジュール
  2. パラメーター・レベル・ヘルプ・モジュール
  3. コマンド例ヘルプ・モジュール
  4. コマンド・エラー・メッセージ・ヘルプ・モジュール

CL コマンドのヘルプを表示しようとする (例えば、コマンド・プロンプトで F1 (ヘルプ) キーを押す) と、IBM i オペレーティング・システムはそのコマンドに関連したヘルプ・パネル・グループがあるかどうかを判別します。そのコマンドの作成時に HLPPNLGRP パラメーターでパネル・グループ名が指定されたか、**コマンド変更 (CHGCMD)** コマンドによってそのコマンドにヘルプ・パネル・グループが関連付けられた場合は、そのパネル・グループに保管されているヘルプが検索され、形式設定され、表示されます。オペレーティング・システムは、ヘルプ・パネル・グループ内の以下のヘルプ・モジュールから、ヘルプを検索しようとします。

- コマンドの作成または変更時に HLPID パラメーターで指定された値と同じ名前を持つコマンド・レベル・ヘルプ・モジュール。例えば、コマンド STRPAY が HLPID(STRPAY) を指定して作成された場合、オペレーティング・システムは STRPAY という名前のヘルプ・モジュールを探します。
- 定数パラメーターを除く各コマンド・パラメーターのパラメーター・レベル・ヘルプ・モジュール。ヘルプ・モジュール名は、コマンドの HLPID 値にスラッシュ文字とパラメーター・キーワード名を続けたものでなければなりません。例えば、コマンド STRPAY の HLPID 値が STRPAY で、TITLE という名前のパラメーターがある場合、オペレーティング・システムは、STRPAY/TITLE という名前のパラメーター・レベル・ヘルプ・モジュールを探します。
- そのコマンドの 1 つ以上の使用例を含むヘルプ・モジュール。ヘルプ・モジュール名は、コマンドの HLPID 値にスラッシュ文字と COMMAND/EXAMPLES を続けたものでなければなりません。例えば、コマンド STRPAY の HLPID 値が STRPAY の場合、オペレーティング・システムは、STRPAY/COMMAND/EXAMPLES という名前のコマンド例ヘルプ・モジュールを探します。
- そのコマンドから発信できるモニター可能メッセージのリストを含むヘルプ・モジュール。ヘルプ・モジュール名は、コマンドの HLPID 値にスラッシュ文字と ERROR/MESSAGES を続けたものでなければなりません。例えば、コマンド STRPAY の HLPID 値が STRPAY の場合、オペレーティング・システムは、STRPAY/ERROR/MESSAGES という名前のコマンド例ヘルプ・モジュールを探します。

拡張 CL コマンド・ヘルプを 5250 端末で (または、5250 エミュレーター・ソフトウェアを使用して) 表示する場合、コマンド・レベルおよびパラメーター・レベルのヘルプ・セクションは必須であり、例およびエラー・メッセージ・ヘルプ・セクションはオプションです。コマンド・レベル・ヘルプ・セクションまたはいずれかのパラメーター・レベル・ヘルプ・セクションがオンライン・ヘルプ・パネル・グループで見つからないと、診断メッセージ CPF6E01 (Help information is incomplete) がヘルプ画面の下部に表示されます。例またはエラー・メッセージ・ヘルプ・モジュールが見つからない場合、診断メッセージは表示されません。

関連情報:

コマンド作成 (CRTCMD) コマンド

CL コマンド・ヘルプの作成**:**

CL コマンド (*CMD) オブジェクトをオンライン・ヘルプ・パネル・グループ (*PNLGRP) オブジェクトに接続する方法が理解できたら、コマンドの 4 つのタイプのヘルプ・モジュールに接続されるヘルプ・テキストを作成できます。

CL コマンドのオンライン・ヘルプは、ユーザー・インターフェース・マネージャー (UIM) というタグ言語で作成します。 UIM ソースを **パネル・グループの作成 (CRTPNLGRP)** コマンドでコンパイルすると、*PNLGRP オブジェクトを作成できます。

関連情報:

パネル・グループ作成 (CRTPNLGRP) コマンド

Application Display Programming

CL コマンド・ヘルプの ***UIM*** ソースの生成**:**

UIM 構文を学ぶ代わりに、より単純に **コマンド資料の生成 (GENCMDDOC)** コマンドを使用して、CL コマンドのオンライン・ヘルプを生成できます。このコマンドを使用すると、UIM ソースを含むファイルを作成できます。このファイルは、オンラインのコマンド・ヘルプのテンプレートを提供します。

UIM ソースを作成するには、GENOPT (生成オプション) パラメーターに *UIM を指定する必要があります。テンプレートの作成に使用する情報は、指定したコマンド・オブジェクト (*CMD) から検索されます。 **コマンド資料の生成 (GENCMDDOC)** を使用して UIM ソースを作成すると、CL コマンドのオンライン・ヘルプの作成を単純化できます。

次に示すのは、**コマンド資料の生成 (GENCMDDOC)** コマンドによる UIM テンプレートの生成の例です。

```
GENCMDDOC   CMD(MYLIB/MYCMD)
            TODIR('/QSYS.LIB/MYLIB.LIB/QPNLSRC.FILE')
            TOSTMF(*CMD)  GENOPT(*UIM)
```

この例のコマンドは、ライブラリー MYLIB にある MYCMD という名前のコマンド・オブジェクトから情報を検索し、 UIM ソースをライブラリー MYLIB にあるソース・ファイル QPNLSRC のメンバー MYCMD に生成します。テンプレート UIM ソースが生成されたら、UIM ソースを以下のように編集する必要があります。

- コマンド・レベル・ヘルプ・モジュールがコマンドの目的を記述していることを確かめます。最初の文の先頭の後に続く **<...>** マーカーを、適切なテキストで置換する必要があります。コマンドの実行に適用される制限 (例えば、最初に実行しなければならないコマンドや、必要な特殊権限) が記述されていることを確かめてください。制限の例がいくつか提供されています。ご使用のコマンドの制限に合わせて編集してください。
- それぞれのパラメーター・レベル・ヘルプ・モジュールがパラメーターの目的を記述していることを確かめます。最初の文の先頭の後に続く **<...>** マーカーを、適切なテキストで置換する必要があります。パラメーター間の依存関係または制限も記述できます。パラメーター・レベルの制限には、:NT. (注記) および :ENT. (注記の終わり) の UIM タグを使用できます。

  パラメーター・レベル・ヘルプ・モジュールでは、パラメーターの可能な選択項目についても記述する必要があります。それぞれの特殊値または単一値の見出しが提供されますが、**<...>** マーカーをパラメーター値の記述で置換する必要があります。
- 例が必要に応じて提供されていることを確かめます。コマンド例のヘルプ・モジュールには、2 つの例の見出しが含まれています。コマンドにパラメーターがない場合は、このヘルプ・モジュールを編集し、例を 1 つだけにすることができます。コマンドに多くのパラメーターがある場合や、複数の特殊機能が備わっている場合は、見出しをコピーして追加のコマンド例を作成できます。コマンド例を編集し

て、独自のコマンドのパラメーター・キーワードとパラメーター値を挿入したり、**<...>** マーカーをコマンド例の処理の記述で置換したりする必要があります。

- エラー・メッセージが必要に応じて提供されていることを確かめます。コマンド・エラー・メッセージ・ヘルプ・モジュールには、 &MSG. 組み込み関数を使用して、コマンドから送られたエラー・メッセージのメッセージ・テキストを組み込む方法を示す見出しが含まれています。メッセージのリストを編集して、コマンドから出されるメッセージの実際のメッセージ ID を、メッセージ記述を含むメッセージ・ファイルと共に含める必要があります。

UIM ソースをコマンドに合わせて編集したら、**パネル・グループの作成 (CRTPNLGRP)** コマンドを使用してオンライン・ヘルプ・パネル・グループを作成できます。次の例は、**パネル・グループの作成 (CRTPNLGRP)** コマンドの使用方法を示しています。

```
CRTPNLGRP   PNLGRP(MYLIB/MYCMD)
            SRCFILE(MYLIB/QPNLSRC)  SRCMBR(MYCMD)
```

このコマンドは、ライブラリー MYLIB のソース物理ファイル QPNLSRC のメンバー MYCMD にある UIM ソースから、パネル・グループの作成を試行します。 UIM ソースのコンパイル時に重大なエラーが見つからなければ、 MYCMD という名前のパネル・グループ (*PNLGRP) オブジェクトがライブラリー MYCMD 内に作成されます。このコマンドが作成するスプール・ファイルを表示すれば、 UIM コンパイラーが検出した通知、警告、および重大エラーを参照できます。

関連情報:

パネル・グループ作成 (CRTPNLGRP) コマンド

コマンド資料の生成 (GENCMDDOC) コマンド

CL コマンド間での共通ヘルプの共用*:*

コマンド・オンライン・ヘルプとコマンドの接続に使用する命名体系により、多数のコマンドのヘルプ・モジュールを単一のパネル・グループに保管できます。複数の関連したコマンドのヘルプ・モジュールが単一のパネル・グループに入っている場合は、 :IMHELP. (ヘルプの組み込み) の UIM タグを使用して、共通ヘルプ情報を共用できます。

例えば、OUTPUT という名前のパラメーターを持つ複数のコマンドが存在し、各コマンドでこのパラメーターの一般的な記述が同じである場合があります。同じコマンド・ヘルプ情報を共用すれば、このパラメーターを持つすべてのコマンドで整合性が保証されます。

オンライン・ヘルプ・テキストを共用する別のオプションは、:IMPORT. (インポート) の UIM タグを使用することです。これにより、別のパネル・グループ内にある 1 つ以上のヘルプ・モジュールを、:IMHELP タグを使用して動的に組み込めるものとして定義できます。

関連情報:

Application Display Programming

ヘルプ・モジュールへの ***CL*** オンライン・ヘルプ・テキストの編成*:*

パネル・グループ内のヘルプ・モジュールに接続する CL コマンド・ヘルプ・テキストの編成方法を選択できます。

:IMHELP. タグを使用して 1 つのヘルプ・モジュールから 1 つ以上の他のヘルプ・モジュールにヘルプを組み込む UIM 機能を使用すれば、事前に定義した 4 つのタイプのコマンド・ヘルプ・モジュールのヘルプ・テキストを 2 つ以上のヘルプ・モジュールの間で分割できます。ヘルプ・テキストを小さめのヘルプ・モジュールに分割する最も一般的な理由は、共通ヘルプ・テキストの共用を容易にすることです。

他のヘルプ・モジュールに組み込まれるヘルプ・モジュールを定義するときは、必ず、4 つのタイプのオンライン・コマンド・ヘルプ・モジュールと混同しないですむヘルプ・モジュール名を選択してください。

CL コマンド資料用の **HTML** ソースの生成**:**

CL コマンドのオンライン・ヘルプからブラウザーを使用して表示されるハイパーテキスト・マークアップ言語 (HTML) 情報を生成するには、**コマンド資料の生成 (GENCMDDOC)** コマンドを使用して、GENOPT (生成オプション) パラメーターに *HTML を指定します。

コマンドのヘルプ情報の作成に加え、IBM i オペレーティング・システムを実行しているシステムに接続していないとき、またはヘルプ情報が印刷されるときに表示することができるコマンド資料も作成できます。このオペレーティング・システムでは、コマンド資料を HTML ソースとして生成し、それをインターネット・ブラウザーを使用して表示したり、多数のワード・プロセッシング・プログラムにインポートしたりできます。これを行うには、**コマンド資料の生成 (GENCMDDOC)** コマンドを使用して、 GENOPT (生成オプション) パラメーターに *HTML を指定します。 このコマンドは、指定したコマンド・オブジェクト (*CMD) と以前から存在しているコマンド用のコマンド・ヘルプ・パネル・グループ (*PNLGRP) オブジェクトから検索した情報を含むファイルを生成します。

HTML 出力の様子やオンライン・ヘルプとの対応性については、任意のコマンドのヘルプと、IBM i Information Center で提供されている CL コマンドの資料を比較してください。 IBM は、Information Center のコマンド資料を、オンライン・コマンド・ヘルプから構築しています。

次に示すのは、**コマンド資料の生成 (GENCMDDOC)** コマンドによる HTML ソースの生成の例です。

```
GENCMDDOC   CMD(MYLIB/MYCMD)
```

この例のコマンドは、ライブラリー MYLIB にある MYCMD という名前のコマンド・オブジェクトから情報を検索し、 HTML ソースをジョブの現行作業ディレクトリーにあるストリーム・ファイル MYLIB_MYCMD.HTML に生成します。生成されたストリーム・ファイルは、**ファイルの表示 (DSPF)** コマンドを使用して HTML ソース形式で表示するか、標準的なインターネット・ブラウザー・ソフトウェアを使用して HTML ブラウザー形式で表示するか、コマンドによって表示できます。

関連情報:

コマンド資料の生成 (GENCMDDOC) コマンド

プロキシー CL コマンド

プロキシー・コマンドは、ターゲット CL コマンドへのショートカットを作成するために使用します。

プロキシー・コマンドは、コマンド処理プログラムというよりはターゲット・コマンドを実行する能力において、一般的な IBM i コマンドとは異なっています。

プロキシー・コマンドを作成するには、**プロキシー・コマンド作成 (CRTPRXCMD)** を使用します。**プロキシー・コマンド作成 (CRTPRXCMD)** には、**コマンド名** (CMD) および**ターゲット・コマンド** (TGTCMD) が必要です。オプション・パラメーターには、**テキスト記述** (TEXT)、**権限** (AUT)、および**置換コマンド** (REPLACE) オプションが含まれます。例えば、ターゲット・コマンド QSYS/DSPJOB を持つプロキシー・コマンドを作成するには、以下のコマンドを入力します。

```
CRTPRXCMD CMD(MYLIB/PCMD1) TGTCMD(QSYS/DSPJOB) TEXT('dspjob proxy')
```

TGTCMD パラメーターにはプロキシー・コマンドを指定することができます。プロキシー・チェーンは、最大で 5 つのプロキシー・コマンドを連結し、最後に 6 番目としてプロキシー以外のコマンドを連結

して構成することができます。 5 つを超えるプロキシー・コマンドのチェーンを持つプロキシー・コマンドを実行すると、CPD0196 エラー「チェーニングされたプロキシー・コマンドの数が 5 を超えています。」が発生します。

プロキシー・コマンドを作成する場合は、コマンド定義ソースを指定しないでください。プロキシー・コマンドには、パラメーター情報、プロンプト制御情報、またはパラメーター間依存関係情報は含まれません。プロキシー・コマンドが使用されている場合、その情報はすべて、プロキシー・コマンドで定義されているターゲット・コマンドから継承されます。プロキシー・コマンドには、ターゲット・コマンドに指定するのと同じパラメーターを指定します。

プロキシー・コマンドを変更する場合は、**プロキシー・コマンド変更 (CHGPRXCMD)** コマンドを使用します。プロキシー・コマンドを削除する場合は、**コマンド削除 (DLTCMD)** コマンドを使用します。

プロキシー・チェーンを経由するように解決されるコマンドを実行または処理するには、ユーザーにそのコマンドおよびライブラリーに対する正しい権限が必要になります。また、プロキシー・コマンド・チェーンの各プロキシー・コマンドおよびライブラリーに対する *USE 権限もユーザーに必要です。

プロキシー・コマンドは他のコマンドを実行するので、プロキシー・コマンドに対して出口プログラムを登録することはできません。プロキシー・コマンドに対して出口プログラムを登録しようとすると、エラー・メッセージ CPF019A が発行されます。プロキシー・コマンドが実行され、そのプロキシー・コマンドの解決先となるプロキシーではないコマンドが出口プログラムを登録していた場合、その後のプロキシー・チェーンは、RTVC0100 または CHGC0100 が戻したフォーマット・データ内に置かれます。

関連情報:

プロキシー・コマンド作成 (CRTPRXCMD) コマンド

プロキシー・コマンド変更 (CHGPRXCMD) コマンド

コマンド削除 (DLTCMD) コマンド

コマンド関連の API

一部のアプリケーション・プログラミング・インターフェース・プログラムは、コマンドで使用できます。

関連概念:

317 ページの『CL コマンドの妥当性検査』
システムは、コマンドの妥当性検査を行います。妥当性検査プログラムは必須ではありませんが、必要ならユーザー固有のプログラムを作成することもできます。

関連タスク:

363 ページの『CL コマンドの作成』
コマンド定義ステートメントによるユーザー・コマンドの定義が完了した時点で、**コマンド作成 (CRTCMD)** コマンドを使用して、そのコマンドを作成できます。

QCAPCMD プログラム

コマンド処理 (QCAPCMD) API は、コマンド・ストリングに対してコマンド分析プログラムの処理を実行します。

この API を使用すると、次のことを行えます。

- コマンド・ストリングの構文を実行前に検査する。
- コマンドのプロンプトを出し、変更されたコマンド・ストリングを受信する。
- 高水準言語からコマンドを使用する。
- コマンドのヘルプを表示する。

関連概念:

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

関連情報:

Process Commands (QCAPCMD) API

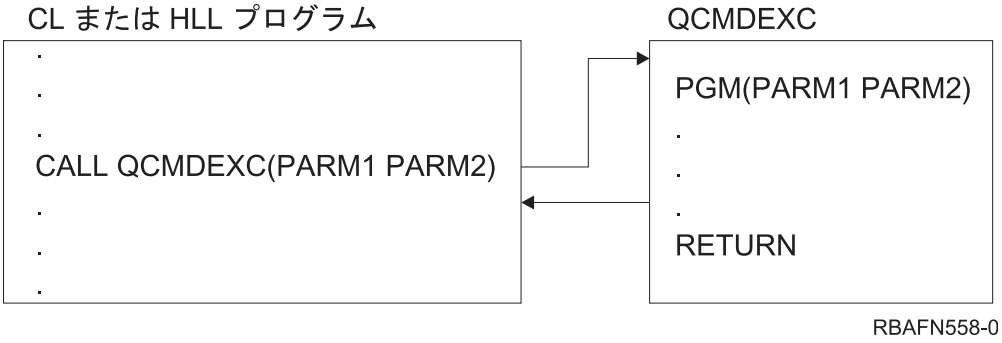
QCMDEXC プログラム

コマンド実行 (QCMDEXC) API は、1 つのコマンドを実行する IBM 提供のプログラムです。

この API を使用すると、次のものから別のコマンドを活動化できます。

- 高水準言語 (HLL) プログラム
- CL プロシージャー
- 実行されるコマンドまたは使用されるパラメーターが、コンパイル時には認識されていないプログラム

QCMDEXC プログラムは、HLL または CL プロシージャーまたはプログラム内から呼び出されます。実行されるコマンドは、CALL コマンド上のパラメーターとして渡されます。



コマンドの実行後、制御は HLL または CL プロシージャーまたはプログラムに返されます。

コマンドは、それがプログラムに入っていなかったかのように実行されます。したがって、コマンドで変数を使用することはできません (コマンドから CL 変数に値を戻すことができないため)。さらに、CL プロシージャーまたはプログラムでのみ使用できるコマンドを QCMDEXC プログラムによって実行することはできません。 QCMDEXC プログラムへの呼び出しの形式は次のとおりです。

```
CALL PGM(QCMDEXC) PARM(コマンド コマンド長)
```

実行したいコマンドを最初のパラメーターに文字ストリングとして入力してください。コマンド・ライブラリーを指定しなければなりません。

```
CALL PGM(QCMDEXC )  PARM('QSYS/CRTLIB LIB(TEST)' 22)
```

コマンドにブランクが含まれる場合は、コマンドを単一引用符で囲まなければなりません。 文字ストリングの最大長は 32,702 文字ですが、区切り文字 (単一引用符) はストリングの一部としてカウントしません。 PARM パラメーターの 2 番目の値として指定する長さは、コマンドとして渡される文字ストリングの長さです。これは、長さ 15 (小数点以下の桁数 5) のパック 10 進数値でなければなりません。

したがって、ライブラリー・リストを置換する場合、QCMDEXC プログラムへの呼び出しは次のようになります。

```
CALL PGM(QCMDEXC) PARM('CHGLIBL LIBL(QGPL NEWLIB QTEMP)' 31)
```

このステートメントを HLL または CL プログラムにコーディングして、プログラムの実行時にライブラリー・リストを置換することが可能です。ただし、QCMDEXC プログラムをこのような方法で使用する場合、実行時の柔軟性はありません。

実行時の柔軟性は、次の事柄によって可能になります。

1. パラメーター・リスト内の定数の代わりに変数を使用し、さらに
2. HLL または CL プログラムへの呼び出しで変数の値を指定する。

例えば、以下の図を検討してみましょう。

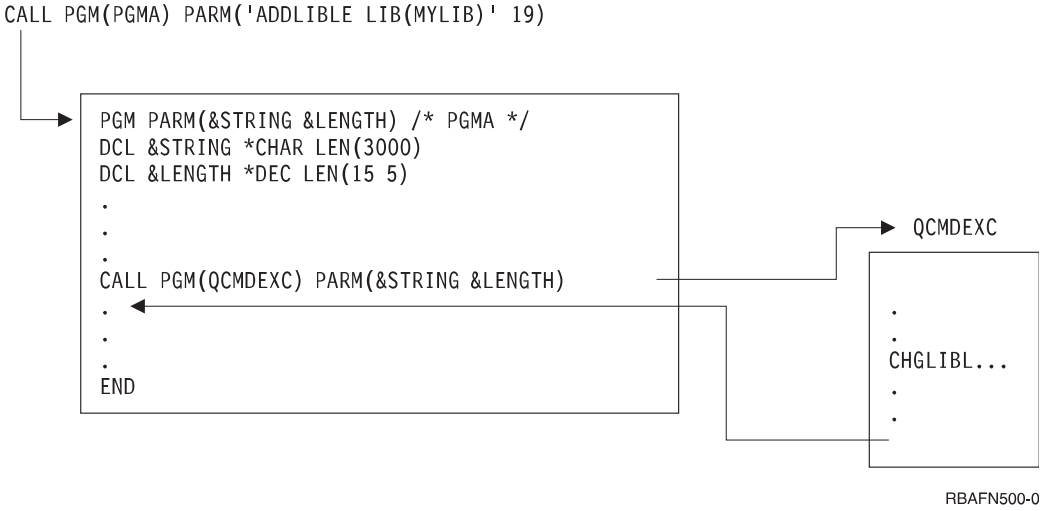


図 *6.* 呼び出し *PGM* の例

2 番目のパラメーターで QCMDEXC プログラムに渡されるコマンド長は、渡されるコマンド・ストリングの最大長です。コマンド・ストリングが引用符で囲まれたストリングとして渡される場合、コマンド長は、引用符の内側のストリングの長さになります。コマンド・ストリングが変数の形で渡される場合、コマンド長は、その CL 変数の長さです。コマンド長を変数内のコマンド・ストリングの実際の長さに合わせて短くしても支障はありませんが、その必要はありません。

すべてのコマンドが QCMDEXC プログラムを使用して実行できるわけではありません。 QCMDEXC プログラムへの呼び出しで渡すコマンドは、呼び出しが実行される時点の環境 (対話式またはバッチ) で有効なものでなければなりません。コマンドは、次のどちらかであってはなりません。

- 入力ストリーム制御コマンド (BCHJOB、ENDBCHJOB、および DATA)
- CL プログラムでのみ使用できるコマンド

対話式ジョブで QCMDEXC を呼び出す場合には、CL コマンドの前に疑問符 (?) を付けて指定することによりプロンプトを要求するか、あるいは、選択プロンプトを使用することができます。

QCMDEXC プログラムを介してコマンドを処理している過程でエラーが検出されると、エスケープ・メッセージが送られます。 **メッセージ・モニター (MONMSG)** コマンドを使用すると、CL プロシージャーまたはプログラム内でこのエスケープ・メッセージを監視することができます。

構文エラーが検出された場合には、メッセージ CPF0006 が送られます。コマンドの実行中にエラーが検出された場合には、そのコマンドから送られたエスケープ・メッセージはすべて QCMDEXC プログラムに

よりユーザー・プログラムに戻されます。 QCMDEXC プログラムを用いて実行するコマンドからのメッセージは、CL プロシージャーおよびプログラムに入っているコマンドからのメッセージを監視するのと同じ方法で監視します。

高水準言語プログラムが呼び出しでのエラーを処理する方法については、高水準言語に関する適切な参照書を参照してください。

関連概念:

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

関連タスク:

537 ページの『メッセージ記述の定義』
事前定義メッセージは、メッセージ・ファイルに保管されます。

398 ページの『CL コマンドの選択プロンプトの使用』
CL コマンドの選択プロンプトは、長い構文のコマンドを使用する際に、一部のパラメーターについてはプロンプトを表示したくない場合に、特に便利な機能です。

関連情報:

Execute Commands (QCMDEXC) API

DBCS データを用いる QCMDEXC プログラム:

コマンド実行 (QCMDEXC) API を使用して、あるコマンドに対して 2 バイト文字セット (DBCS) 入力データが入力されるよう要求することができます。

2 バイト・データの入力を要求する場合の QCMDEXC の形式は次のとおりです。

```
CALL QCMDEXC ('コマンド' コマンド長 IGC)
```

QCMDEXC プログラムの 3 番目のパラメーターである IGC は、2 バイト・データを受け入れるようシステムに指示します。例えば、次の CL プログラムは、ユーザーにメッセージとして 2 バイト・テキストを入力するよう要求するものです。システムは、入力されたメッセージを送ります。

```
PGM
   CALL QCMDEXC ('?SNDMSG' 7 IGC)
ENDPGM
```

この例の説明は次のとおりです。

- 文字 ? は、**メッセージ送信 (SNDMSG)** コマンドのコマンド・プロンプトを出すようシステムに指示します。
- 値 7 は、**メッセージ送信 (SNDMSG)** コマンドに疑問符を加えたストリングの長さです。
- 値 IGC を使用すれば、2 バイト・データを要求できます。

QCMDEXC プログラムの実行後に、次の画面が表示されます。この画面では 2 バイト変換を使用することができます。

```
                          メッセージ送信  (SNDMSG)

選択項目を入力して，実行キーを押してください。

メッセージ・テキスト. . . . . .
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
TO ユーザー・プロファイル . . . . _________  名前，*SYSOPR，*ALLACT...









                                                                    終わり
 F3= 終了  F4= プロンプト F5= 最新表示 F10= 追加のパラメーター
 F12= 取り消し
 F13= この画面の使用法    F24= キーの続き
```

関連情報:

Execute Commands (QCMDEXC) API

QCMDCHK プログラム

コマンドの構文検査 (QCMDCHK) API は、単一コマンドに対する構文検査を行い、オプションでそのコマンドのプロンプトを出す IBM 提供のプログラムです。

コマンドは実行されません。プロンプトが要求された場合は、プロンプトを通じて入力された、更新された値とともに、コマンド・ストリングが呼び出しプロシージャーまたはプログラムに返されます。
QCMDCHK プログラムは CL プロシージャーまたはプログラム、あるいは HLL プロシージャーまたはプログラムから呼び出せます。

QCMDCHK の代表的な用途は次のとおりです。

- ユーザーにコマンドのプロンプトを表示し、そのコマンドを後で処理するために保管する。
- ユーザーが指定したオプションを判別する。
- 処理されたコマンドを記録する。最初に QCMDCHK でプロンプトを表示し、QCMDEXC でコマンドを実行し、次に、処理されたコマンドを記録します。

QCMDCHK への呼び出しの形式は次のとおりです。

```
CALL PGM(QCMDCHK) PARM(コマンド  コマンド長)
```

QCMDCHK に渡される最初のパラメーターは、検査またはプロンプトの対象となるコマンドを示す文字ストリングです。最初のパラメーターが変数である場合にプロンプトが要求されると、ワークステーション・ユーザーによって入力されたコマンドがその変数に入れられます。

2 番目のパラメーターは、渡されるコマンド・ストリングの最大長です。コマンド・ストリングが引用符で囲まれたストリングとして渡される場合、コマンド長は、引用符の内側のストリングの長さになります。コマンド・ストリングが変数の形で渡される場合、コマンド長は、その CL 変数の長さです。 2 番目のパラメーターは、長さ 15 (小数点以下の桁数 5) のパック 10 進数値でなければなりません。

QCMDCHK プログラムは、渡されたコマンド・ストリングに対して構文検査を行います。このプログラムにより、必須のパラメーターがすべて指定されているかどうかと、すべてのパラメーターに許容値が指定されているかどうかが検査されます。ただし、処理環境の検査は行われません。コマンドがバッチ・ジョブでだけ使用可能か、対話式ジョブでだけ使用可能か、あるいはバッチまたは対話式の CL プログラムでだけ使用可能であるのかを検査することはできます。このプログラムでは、コマンド定義ステートメントは検査できません。

コマンドに構文エラーが見つかった場合には、メッセージ CPF0006 が送られます。このメッセージを監視することにより、コマンドにエラーがあるかどうかを判別することができます。メッセージ CPF0006 の前に、エラーを識別するための 1 つまたは複数の診断メッセージが出ます。次の例では、**制御言語プログラム作成 (CRTCLPGM)** コマンドの PGM パラメーターに指定された値 123 が有効ではないので、プログラム内のラベル ERROR に制御権が移ります。

```
CALL QCMDCHK ('CRTCLPGM PGM(QGPL/123)' 22)
MONMSG CPF0006 EXEC(GOTO ERROR)
```

コマンド名の前に疑問符を入れるか、またはコマンド・ストリングの 1 つまたは複数のキーワード名の前に選択プロンプト文字を入れることによって、コマンドについてのプロンプトを要求することができます。

コマンドの検査およびプロンプトの過程でエラーが検出されなかった場合には、更新されたコマンド・ストリングが、最初のパラメーターに指定された変数に入れられます。プロンプト要求文字は、コマンド・ストリングから除去されます。次にこの例を示します。

```
DCL &CMD *CHAR 2000
 .
 .
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 2000)
```

QCMDCHK プログラムへの呼び出しが実行された後、変数 &CMD には、プロンプターを通じて入力されたすべての値とともにコマンド・ストリングが入ります。これは、次のようになります。

```
CRTCLPGM PGM(PGMA) SRCFILE(TESTLIB/SOURCE) USRPRF(*OWNER)
```

コマンド名の前にあった疑問符が除去されていることに注意してください。

QCMDCHK プログラムによりプロンプトを要求する場合には、コマンド・ストリングは CL 変数の形で渡さなければなりません。そうしなければ、更新されたコマンド・ストリングがプロシージャーまたはプログラムに返されません。また、コマンド・ストリングを表す変数が、プロンプターから戻される更新済みのコマンド・ストリングを収容するのに十分な長さであることを確認しなければなりません。長さが足りないと、メッセージ CPF0005 が送られ、コマンド・ストリングを含む変数は変更されません。選択プロンプトを使用しない場合には、プロンプターからはユーザーが入力した項目だけが戻されます。

変数の長さは 2 番目のパラメーターの値によって決まりますが、それは変数の実際の長さではありません。次の例では、変数は十分な長さを用いて宣言されていますが、指定された長さが短すぎて更新されたコマンドを収容できないために、エスケープ・メッセージ CPF0005 が送られます。

```
DCL &CMD *CHAR 2000
 .
 .
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 9)
```

QCMDCHK の実行時に F3 または F12 を押してプロンプターを終了すると、メッセージ CPF6801 が、QCMDCHK を呼び出したプロシージャーまたはプログラムに送信され、コマンド・ストリングの入った変数は変更されません。

PARM、ELEM、または QUAL コマンド定義ステートメントで PASSATR(*YES) を指定し、**コマンド定義の変更 (CHGCMDDFT)** コマンドを用いてデフォルト値を変更すると、デフォルト値は、それがデフォルト値ではなくユーザー指定の値であるかのように強調表示されます。変更された PARM、ELEM、または QUAL コマンド定義ステートメントのデフォルト値をもとのデフォルト値に戻すと、デフォルト値は強調表示されなくなります。

関連タスク:

398 ページの『CL コマンドの選択プロンプトの使用』
CL コマンドの選択プロンプトは、長い構文のコマンドを使用する際に、一部のパラメーターについてはプロンプトを表示したくない場合に、特に便利な機能です。

関連情報:

Check Command Syntax (QCMDCHK) API

実行時のユーザー入力のプロンプト

大部分の CL プログラムおよびプロシージャーでは、ワークステーション・ユーザーがプログラムに渡されるコマンド・パラメーター値を指定するか、表示プロンプトの入力可能フィールドに入力することで、入力を行います。

次の方法で、ワークステーション・ユーザーに、CL プロシージャーまたはプログラムへの入力を促すプロンプトを出すことができます。

- CL ソース・プログラムの CL コマンドの前に **?** を入力すると、システムはその CL コマンドの入力を指示するプロンプトを表示します。ソース・プログラム内で既に指定しているパラメーター値は組み込まれるため、ワークステーション・ユーザーが変更することはできません。
- QCMDEXC プログラムを呼び出し、選択プロンプトを要求する場合、CL コマンドの入力を指示するプロンプトがシステムによって表示され、処理時に使用される CL コマンドを CL ソース・プログラムで指定する必要はありません。

関連概念:

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

CL プロシージャーまたはプログラム内での IBM i プロンプターの使用

CL プロシージャーまたはプログラムの対話式処理から、プロンプトを要求することができます。

例えば、次のプロシージャーはコンパイルして実行することができます。

```
PGM
.
.
.
?DSPLIB
.
.
.
ENDPGM
```

この場合には、プログラムの実行過程で**ライブラリー表示 (DSPLIB)** コマンドについてのプロンプトが表示されます。 **ライブラリー表示 (DSPLIB)** コマンドの処理は、ユーザーが必要なパラメーターに値を入力し、実行キーを押すまで開始しません。

ソース・プログラムに指定された値は、オペレーター (またはユーザー) が直接変更することはできません。以下にその例を示します。

```
PGM
.
.
.
?SNDMSG TOMSGQ(WS01 WS02)
.
.
.
ENDPGM
```

プロシージャーが呼び出され、 **メッセージ送信 (SNDMSG)** コマンドの入力を指示するプロンプトが表示されたら、オペレーター (またはユーザー) は、MSG、 MSGTYPE、および RPYMSGQ パラメーターに値を入力できますが、TOMSGQ パラメーターの値は変更できません。例えば、オペレーター (またはユーザー) は WS03 の追加や、WS02 の削除を行うことはできません。 CL プログラムまたはプロシージャー内での実行時のプロンプターの使用には、次の制約事項が適用されます。

- CL またはプログラムからプロンプターが呼び出されるときは、そのプロンプトのパラメーター値に変数名や式は入力できません。
- IF、ELSE、または MONMSG の各コマンドに組み込んだコマンドに対してプロンプトを要求することはできません。

  正

  ```
  IF (&A=5) THEN(DO)
              ?SNDMSG
              ENDDO
  ```

  誤り

  ```
  IF (&A=5) THEN(?SNDMSG)
  ```

- 実行時に以下のコマンドに対してプロンプトを使用することはできません。

CALL	**CALLPRC**	**CALLSUBR**	**CHGVAR**
COPYRIGHT	**DCL**	**DCLF**	**DCLR**
DO	**DOFOR**	**DOUNTIL**	**DOWHILE**
ELSE	**ENDDO**	**ENDPGM**	**ENDRCV**
ENDSELECT	**ENDSUBR**	**GOTO**	**IF**
ITERATE	**LEAVE**	**MONMSG**	**OTHERWISE**
PGM	**RCVF**	**RETURN**	**RTNSUBR**
SELECT	**SNDF**	**SNDRCVF**	**SUBR**
WAIT	**WHEN**		

- バッチ・ジョブでは、プロンプトを使用することはできません。

CL ソース・ファイル・メンバー内のコマンドにプロンプト要求 (?) を入力すると、コンパイルが正常に終了する前に、そのコマンドについての診断メッセージが表示されることがあります。この場合は、メッセージを慎重に検査して、プロシージャーまたはプログラムの実行時にプロンプト表示画面で入力した値によって、エラーが訂正されているかどうか調べなければなりません。

対話式環境では、上にリストされているコマンド (CL プロシージャーまたはプログラムの処理時にプロンプトを出すことができない) を除き、任意のモードで許可されているすべてのコマンドのプロンプトを出すことができます。このため、ワークステーションで任意のコマンドのプロンプトを出すことができ、各種のコマンドとそれらのパラメーターについて説明されている資料を参照する必要性が減ります。

プロンプトが出されたコマンドの実行時に F3 または F12 を押してそのコマンドを取り消すと、エスケープ・メッセージ (CPF6801) が CL プロシージャーまたはプログラムに送信されます。 CL プロシージャーまたはプログラムで **メッセージ・モニター (MONMSG)** コマンドを用いると、このメッセージを監視することができます。

コマンドのプロンプトを出すと、プロシージャーまたはプログラムは、入力されたコマンド・ストリングを受信しません。受け取るようにするためには、QCMDCHK を使用してプロンプトを表示し、次に QCMDEXC を使用してそのコマンドを実行してください。プロンプトを表示し、コマンドを実行するには、QCAPCMD も使用できます。

関連タスク:

401 ページの『CL プロシージャーおよびプログラム内のプロンプトを伴う QCMDEXC の使用』
コマンド実行 (QCMDEXC) プログラムを使用して、プロンプターを呼び出すことができます。

関連情報:

CL コマンド検索プログラム

CL コマンドの選択プロンプトの使用

CL コマンドの選択プロンプトは、長い構文のコマンドを使用する際に、一部のパラメーターについてはプロンプトを表示したくない場合に、特に便利な機能です。

選択プロンプトは対話式プロンプトの中で使用することもできますし、CL プロシージャーまたはプログラム内で使用するためにソースとして (SEU で) 入力することもできます。選択プロンプトを指定するソースは SEU を使用して入力することができますが、SEU でコマンドを入力している過程で選択プロンプトを使用することはできません。

選択プロンプトを使用して、次のことを行うことができます。

- プロンプトの必要なパラメーターの選択
- 保護したいパラメーターの判別
- プロンプトからのパラメーターの除外

選択プロンプトに適用される制約事項は次のとおりです。

- 次の場合は、コマンド名またはラベルの前に? (疑問符) を付ける必要があります。
  - 1 つまたは複数の選択プロンプト・オプションが ?- (疑問符、負符号) である場合。
  - CPF6805 メッセージ (コンパイルが成功したのに、コマンド上に診断問題があることを示すメッセージ) を受け取らないようにする場合。
- パラメーターは、位置により指定することができますが、その前に選択プロンプト文字を付けてはなりません。
- 選択プロンプトの対象とするパラメーターは、キーワード形式でなければなりません。
- 選択プロンプト文字とキーワードとの間にブランクを入れてはなりません。
- 選択プロンプトは、パラメーター・レベルでだけ使用することができます。したがって、値のリスト中の特定のキーワード値を指定することはできません。
- ?- は、プロンプト一時変更プログラムでは使用できません。
- パラメーターが必須の場合は、?? 選択プロンプトを使用する必要があります。

  コマンドのプロンプトが表示されるときに入力欄が強調表示されるため、パラメーターが必須であることを示すことができます。

ユーザーが指定した値には、選択プロンプトの場合にも通常のプロンプトの場合にも、その値の前に特殊記号 (>) が付けられます。パラメーター・プロンプトのユーザー指定の値の前にこの記号が付いていない場合には、そのコマンドのデフォルト値がコマンド処理プログラムに渡されます。

PARM、ELEM、または QUAL の各コマンド定義ステートメントに PASSATR(*YES) が指定され、しかも **コマンド定義変更 (CHGCMDDFT)** コマンドを使用してそのデフォルト値が変更されている場合には、変更後のデフォルト値はデフォルト値としてではなく、 (> 記号を用いて) ユーザー指定の値として表示されます。変更した PARM、ELEM、または QUAL の各コマンド定義ステートメントのデフォルト値を当初のデフォルト値に戻すと、> 記号が除かれます。

選択プロンプトの使用中に F5 キーを押せば、最初に画面に表示されていた値を再度表示することができます。

選択プロンプトで表示されるパラメーターの値を、CL 変数を使用して指定した場合にはプロンプトの値を変更することができ、コマンドの実行時には変更後の値が使用されます。プロシージャーまたはプログラム内の変数の値は変更されません。 CL プロシージャーに次のものが入っている場合、

```
OVRDBF ?*FILE(FILEA) ??TOFILE(&FILENAME) ??MBR(MBR1)
```

3 つのパラメーター、FILE、TOFILE、および MBR が、プロンプト画面に示されます。 FILE パラメーターに指定されている値は変更することはできませんが、TOFILE および MBR パラメーターの値は変更することができます。 CL 変数 &FILENAME の値が FILE1 であり、それを FILE2 に変更したいとします。コマンドの実行時には FILE2 の値が使用されますが、プロシージャー内の &FILENAME の値は変更されません。次の表は、各種の選択プロンプト文字と、その結果とられる処置を示しています。

入力	表示される値	保護	指定がない場合に **CPP** に渡される値	> 記号の表示
??KEYWORD()	デフォルト値	無	デフォルト値	無
??KEYWORD(VALUE)	値	無	値	有
?*KEYWORD()	デフォルト値	有	デフォルト値	無
?*KEYWORD(VALUE)	値	有	値	有
?<KEYWORD()	デフォルト値	無	デフォルト値	無
?<KEYWORD(VALUE)	値	無	デフォルト値	無
?/KEYWORD()	デフォルト値	有	デフォルト値	無
?/KEYWORD(VALUE)	値	有	デフォルト値	無
?-KEYWORD()	なし	N/A	デフォルト値	N/A
?-KEYWORD(VALUE)	なし	N/A	値	N/A
?&KEYWORD()	デフォルト値	無	デフォルト値	無
?&KEYWORD(VALUE)	値	無	デフォルト値	無
?%KEYWORD()	デフォルト値	有	デフォルト値	無
?%KEYWORD(VALUE)	値	有	デフォルト値	無

入力	**F5** を押すか消去した時に表示される値	説明
??KEYWORD()	デフォルト値	コマンドのデフォルト値を表示する通常のキーワード・プロンプト
??KEYWORD(VALUE)	値	プログラム指定のデフォルト値を表示する通常のキーワード・プロンプト

入力	**F5** を押すか消去した時に表示される値	説明
?*KEYWORD()	デフォルト値	コマンドのデフォルト値だけが使用される場合に、保護されたプロンプトが (情報として) 表示されます。
?*KEYWORD(VALUE)	値	プログラム指定の値だけが使用される場合に、保護されたプロンプトが (情報として) 表示されます。例えば、値を情報として表示するだけで変更できないようにしたい場合に指定します。
?<KEYWORD()	デフォルト値	コマンドのデフォルト値を表示する通常のキーワード・プロンプト
?<KEYWORD(VALUE)	値	プログラム指定のデフォルト値を表示する通常のキーワード・プロンプト
?/KEYWORD()	デフォルト値	IBM が使用するために予約済みです。
?/KEYWORD(VALUE)	値	IBM が使用するために予約済みです。
?&KEYWORD()	デフォルト値	コマンドのデフォルト値を表示する通常のキーワード・プロンプト
?&KEYWORD(VALUE)	値	プログラム指定のデフォルト値を表示する通常のキーワード・プロンプト
?%KEYWORD()	デフォルト値	コマンドのデフォルト値だけが使用される場合に、保護されたプロンプトが (情報として) 表示されます。
?%KEYWORD(VALUE)	値	プログラム指定の値だけが使用される場合に、保護されたプロンプトが (情報として) 表示されます。例えば、値を情報として表示するだけで変更できないようにしたい場合に指定します。

選択プロンプトは、QCMDEXC や QCMDCHK プログラムで使用することができます。呼び出しの形式は次のとおりです。

```
CALL PGM(QCMDEXC または QCMDCHK) PARM(コマンド コマンド長さ)
```

次の表に、選択プロンプト文字についての要約を示します。

選択プロンプト文字	説明
??	パラメーターが表示され、入力可能です。
?*	パラメーターは表示されますが、入力可能ではありません。ユーザー指定の値がすべて、コマンド処理プログラムに渡されます。
?<	パラメーターが表示され、入力可能ですが、表示されている値を変更しない場合、コマンドのデフォルト値がコマンド処理プログラムに渡されます。
?/	IBM が使用するために予約済みです。
?–	パラメーターは表示されません。ユーザー指定の値 (またはデフォルト値) がコマンド処理プログラムに渡されます。プロンプト一時変更プログラムでは使用できません。

選択プロンプト文字	説明
?&	パラメーターは、F9=すべてのパラメーターが押されるまで表示されません。表示されると、入力可能になります。パラメーター上に表示された値が変更されない限り、コマンドのデフォルト値が CPP に送られます。
?%	パラメーターは、F9=すべてのパラメーターが押されるまで表示されません。表示されても、入力は不可です。コマンドのデフォルト値が CPP に送られます。

関連概念:

358 ページの『プロンプト一時変更プログラムから戻される情報』
プロンプト一時変更プログラムは、渡された情報に基づいて、キー・パラメーター以外のパラメーターの現在の値を検索します。

関連タスク:

391 ページの『QCMDEXC プログラム』
コマンド実行 (QCMDEXC) API は、1 つのコマンドを実行する IBM 提供のプログラムです。

394 ページの『QCMDCHK プログラム』
コマンドの構文検査 (QCMDCHK) API は、単一コマンドに対する構文検査を行い、オプションでそのコマンドのプロンプトを出す IBM 提供のプログラムです。

CL プロシージャーおよびプログラム内のプロンプトを伴う QCMDEXC の使用

コマンド実行 (QCMDEXC) プログラムを使用して、プロンプターを呼び出すことができます。

CL プロシージャーおよびプログラム内でプロンプトを伴う QCMDEXC を使用すると、コマンド名自体を除いては、コマンド上のすべての値を変更することができます。これはプロンプターを直接使用するよりも柔軟性があります。プロンプターを直接使用する場合は、ソースで指定されていない値しか入力できません (前の項を参照してください)。次のようなコマンドでプロンプターを直接呼び出す場合は、

```
?OVRDBF FILE(FILEX)
```

FILE を除く任意のパラメーターに値を指定できます。ただし、次の例のように、QCMDEXC プログラムを使用したプログラムの処理中にこのコマンドが呼び出される場合は、FILE を始めとする任意のパラメーターに値を指定することができます。この例では、FILEX がデフォルト値です。

```
CALL QCMDEXC PARM('?OVRDBF FILE(FILEX)' 19)
```

前述の選択プロンプトを使用すれば、変更が可能な、指定された値の入ったプロンプトを実現することもできます。ただし、各キーワードを明示的に選択する必要があります。プロンプターは、次のようなコマンドで直接呼び出されます。

```
OVRDBF ??FILE(FILEX) ??TOFILE(*N) ??MBR(*N)
```

関連タスク:

396 ページの『CL プロシージャーまたはプログラム内での IBM i プロンプターの使用』
CL プロシージャーまたはプログラムの対話式処理から、プロンプトを要求することができます。

関連情報:

Execute Command (QCMDEXC) API

プログラム・ソースの入力

プログラマー・メニューを使用してプログラム・ソースを入力できます。

プログラマー・メニューは、QPGMMENU プログラムを呼び出すか、または**プログラマー・メニュー開始 (STRPGMMNU)** コマンドを使用して、直接呼び出すことができます。このコマンドを使用すると、プログラマー・メニューで使用するデフォルト値を事前に指定することができます。さらに、**STRPGMMNU** コマンドは、プログラマー・メニューの使用を調整するのに用いることができる他のオプションもサポートします。

関連情報:

プログラマー・メニュー開始 (STRPGMMNU) コマンド

プログラマー・メニュー開始コマンドの使用

プログラマー・メニュー開始 (STRPGMMNU) コマンドは、QPGMMENU の呼び出しと同じ機能を実行し、標準入力フィールドに入力します。

次のコマンド・パラメーターを使用して、メニューの下部にある標準入力フィールドに入力することができます。

- ソース・ファイル
- ソース・ライブラリー
- オブジェクト・ライブラリー
- ジョブ記述

このコマンドは、メニューの初期値を制御する 1 つまたは複数のパラメーターと併用できます。これを、サインオン目的で、またはユーザーが特定のユーザー作成機能を呼び出す目的で、初期プログラムの一部として作成することができます。次の例は、それぞれ異なる初期値を必要とする各アプリケーション領域ごとに独立した機能を持つ、この種のプログラムを示します。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
      PGM
      CHGLIBL   LIBL(PGMR1 QGPL QTEMP)
LOOP:
      STRPGMMNU SRCLIB(PGMR1) OBJLIB(PGMR1) JOBD(PGMR1)
      MONMSG    MSGID(CPF2320) EXEC(GOTO END) /* F3 or F12 to leave menu */
      GOTO LOOP
END:  ENDPGM
```

- プログラマー・メニュー・オプションの制御

  他のパラメーターは、メニューおよびその機能を制御するさいの手助けとなります。例えば、ALWUSRCHG(*NO) を指定すると、ユーザーがメニュー上に表示される値を変更するのを防ぐことができます。このパラメーターはセキュリティー機能とは見なされません。それは、メニューを使用するユーザーは **STRPGMMNU** コマンドを呼び出して、別個の呼び出しで値を変更できるためです。 (ユーザーはさらに、コマンド入力画面を呼び出すために F10 を用いて機能を開始することができます。) **STRPGMMNU** コマンドによってメニューが表示される場合は、ユーザーが (権限によって) QPGMMENU プログラムを直接呼び出さないようにすることができますが、ユーザーが新たに **STRPGMMNU** コマンド呼び出しを要求しないようにすることはできません。

- メニュー作成オプションの調整

  EXITPGM および DLTOPT パラメーターを用いると、メニュー作成オプション (オプション 3) に独自のサポートを提供することができます。オプション 3 を要求したときには、ユーザー・プログラムを呼ぶ出すことができます。 IBM は、ユーザー・プログラムに渡されるこれらのパラメーターおよびパラメーター・リストの詳細について説明するオンライン情報を提供しています。詳細は、 281 ページの『パラメーターの受け渡し』を参照してください。

関連情報:

プログラマー・メニュー開始 (STRPGMMNU) コマンド

プログラマー・メニュー開始コマンドの **EXITPGM** パラメーターの使用**:**

EXITPGM パラメーターは、さまざまな目的に使用することができます。

- オプション 3 で投入された作成コマンドに使用するデフォルト値を変更するため。

  例えば、F4 (プロンプト) が使用されない場合、EXITPGM パラメーターで 1 つまたは複数の作成コマンドを変更して、独自のデフォルト要件を指定することができます。 F4 を使用すると、EXITPGM パラメーターで、プログラマーが入力したコマンドを投入できます (パラメーターを変更せずに)。
- プログラマーによる F4 の使用に関係なく、パラメーターを変更するため。

  このためには、&RQSDTA512 パラメーターの値 (出口プログラムに渡される) を走査して、それが既に使用されており、必要な値を置換しているかどうかを調べなければなりません。
- **ジョブ投入 (SBMJOB)** コマンド上の他のパラメーターを変更するため。

  例えば、**SBMJOB** コマンドのユーザー・パラメーターを変更すると、 *CURRENT の値ではなく、ジョブ記述の値を指定できます。 **ジョブ属性検索 (RTVJOBA)** コマンドを使用し、特定の値として属性を入力して、1 つ以上のジョブ属性の値を検索することもできます。
- ローカル・プログラミング規則を実施するため。

  例えば、すべての物理ファイルには P で終わる 7 文字の名前を付けることが命名規則で要求されている場合、出口プログラムで、この規則に従わない名前を指定した**物理ファイル作成 (CRTPF)** コマンドを使用する試みをすべて拒否することができます。

コマンド分析プログラムの出口点

出口プログラムの登録機能は、システム上の制御言語 (CL) コマンドに対して 2 つの出口点を提供します。

- QIBM_QCA_CHG_COMMAND 出口点には、ある特定の CL コマンドの出口プログラムを 1 つだけ登録することができます。この出口点に対して指定されるプログラムは、プロンプターに制御を渡す前にコマンド分析プログラムによって呼び出されます。
- QIBM_QCA_RTV_COMMAND 出口点の場合、各 CL コマンドごとに最高 10 個の出口プログラムを登録することができます。コマンド分析プログラムは、妥当性検査プログラム (VCP) を実行した後、これらの出口プログラムを呼び出します。出口プログラムは、コマンドに対してコマンド処理プログラム (CPP) を実行する前でも後でも呼び出すことができます。

関連情報:

API 出口プログラム (API Exit Programs)

DBCS データ用のアプリケーション・プログラミングの設計

2 バイト・データを処理するアプリケーション・プログラムを設計したり、英数字アプリケーション・プログラムを 2 バイト・プログラムに変換したりする場合は、この情報を考慮してください。

DBCS アプリケーション・プログラムの設計

2 バイト・データを処理するアプリケーション・プログラムは、英数字データを処理するアプリケーション・プログラムを設計するのと同じ方法で設計できますが、付加的な考慮事項があります。

- データベース・ファイル内で使用される 2 バイト・データがある場合は、それを識別します。

- 2 バイト・データで使用できる表示および印刷装置様式を作成します。
- 必要であれば、対話式アプリケーション用のデータ入力手段として 2 バイト変換を提供します。表示装置ファイル内で DBCS 変換を指定するには、2 バイト変換のための DDS キーワード (IGCCNV) を使用してください。
- プログラムによって表示される 2 バイト・エラー・メッセージを作成します。
- 拡張文字処理を指定して、システムがすべての 2 バイト・データを印刷して表示するようにします。
- 定義しなければならない 2 バイト文字 (もしあれば) を判別します。

関連情報:

DBCS データの処理

DBCS データを処理するための英数字プログラムの変換

英数字アプリケーション・プログラムで外部記述の表示装置ファイルが使用されている場合は、ファイルを変更するだけで、そのアプリケーション・プログラムを 2 バイト・アプリケーション・プログラムに変更することができます。

アプリケーション・プログラムを 2 バイト・アプリケーション・プログラムに変換するには、次の手順を行います。

1. 変更したい英数字ファイルのソース・ステートメントの複製コピーを作成します。
2. 英数字定数およびリテラルを、2 バイト定数およびリテラルに変更します。
3. ファイル内のフィールドを、DBCS データを入力するために、次のいずれかのデータ・タイプに変更します。
   - DBCS 混用 (O) データ・タイプ
   - DBCS 専用 (J) データ・タイプ
   - DBCS 択一 (E) データ・タイプ

   フィールドの長さは変更する必要がありません。
4. 変換された表示装置ファイルを別個のライブラリーに保管します。ファイルに、その英数字バージョンと同じ名前を付けます。
5. ジョブ内で変換済みファイルを使用するには、ライブラリー・リスト変更 (CHGLIBL) コマンドを用いて、ファイルが使用されるジョブについてのライブラリー・リストを変更します。そうすることによって、英数字バージョンのファイルが保管されているライブラリーの前に、2 バイト表示装置ファイルが保管されているライブラリーが検査されるようにします。

CL プログラム内での DBCS データの使用

CL プログラム内で異なるキーボード・シフトを使用することができます。

以下のプログラムで、2 バイト・データがこのプログラムでどのようにテキスト値としてのみ使用されているのかに注意してください。ただし、コマンド自体は英数字です。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
            PGM
            DCLF       FILE(IGCTEST)
START:      CHGVAR     VAR(&OUTPUTA) VALUE('ABCDETGHIJ')
            CHGVAR     VAR(&OUTPUTJ) VALUE('ABCD')
            CHGVAR     VAR(&BOTHJ) VALUE('ABCD')
            CHGVAR     VAR(&OUTPUTE) VALUE('EFGH')
            CHGVAR     VAR(&OUTPUTO) VALUE('ABCDF')
```

```
LOOP:       SNDRCVF
            IF         COND(&INO1 *EQ '0') THEN(RETURN)
            CHGVAR     VAR(&OUTPUTA) VALUE(&INPUTA)
            CHGVAR     VAR(&OUTPUTJ) VALUE(&INPUTJ)
            CHGVAR     VAR(&OUTPUTE) VALUE(&INPUTE)
            CHGVAR     VAR(&BOTHE) VALUE(&INPUTE)
            CHGVAR     VAR(&OUTPUTO) VALUE(&INPUTO)
            GOTO       CMDLBL(LOOP)
            ENDPGM
```

実行時、このプログラムは、DDS 表示装置ファイルの異なるキーボード・シフトがどのように使用されるかを示します。

```
     ALPHANUMERIC SHIFT (A) . . . . . _________     ABCDETGHIJ
     DBCS-ONLY (J). . . . . . . . . . _______      ABCD
     DBCS-EITHER (E). . . . . . . . . _______      EFGH
     DBCS-OPEN (O). . . . . . . . . . _________    ABCDF
     DBCS-ONLY (J) INPUT-OUTPUT . . . . _______
     DBCS-EITHER (E) INPUT-OUTPUT . . . _______
     INPUT 0 TO END . . . . . . . . . . _
```

以下のコード・スニペットは、表示装置ファイル IGCTEST のデータ記述仕様 (DDS) を示しています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
00010A         R IGCTEST
00020A                                 04 07'ALPHANUMERIC SHIFT (A) . . . . . .'
00030A           INPUTA        10A  I 04 43
00040A           OUTPUTA       10A  O 04 58
00050A                                 05 07'DBCS-ONLY (J) . . . . . . . . . . '
00060A           INPUTJ        08J  I 05 43
00070A           OUTPUTJ       08J  O 05 58
00080A                                 06 07'DBCS-EITHER (E) . . . . . . . . . '
00090A           INPUTE        08E  I 06 43
00100A           OUTPUTE       08E  O 06 58
00110A                                 07 07'DBCS-OPEN (O) . . . . . . . . . . '
00120A           INPUTO        09O  I 07 43
00130A           OUTPUTO       09O  O 07 58
00140A                                 08 07'DBCS-ONLY (J) INPUT-OUTPUT . . . .'
00150A           BOTHJ         08J  B 08 43
00160A                                 09 07'DBCS-EITHER (E) INPUT-OUTPUT . . .'
00170A           BOTHE         08E  B 09 43
00180A                                 10 07'INPUT 0 TO END . . . . . . . . . .'
00190A           INO1          01A  B 10 43
```

注:

- 表示装置ファイル IGCTEST の作成時に、CRTDSPF コマンドで IGCDTA(*YES) を指定します。
- CL プログラム用のソース物理ファイルの作成時に、CRTSRCPF コマンドで IGCDTA(*YES) を指定します。

制御言語でのユニコード・サポート

制御言語 (CL) は、ユニコード (UTF-16) パラメーター値をサポートします。このサポートにより、プログラムは、ジョブの EBCDIC セットだけでなく、ユニコード文字の完全なセットを渡すことができるようになります。

ユニコードの概要

ユニコードは、文字セットおよびその文字セットの少数のエンコードを明確に定義する標準です。ユニコードを使用すると、任意の言語のテキストを効果的に処理できます。

ユニコードについて

ユニコードは、任意の言語のテキストを処理できるようにする標準です。ユニコードを使用すると、単一のアプリケーションが世界中のユーザーのために機能することができます。

ユニコードが開発される前に存在していたエンコード・システムでは、使用されている必要なすべての数値、文字、および記号に対応できませんでした。エンコード・システムが違うと、異なる文字に同じ番号が割り当てられていることがありました。誤ったエンコード・システムを使用すると、出力が予期したとおりに表示されないことがあります。

ユニコードは、プラットフォーム、言語、またはプログラムに関係なく、すべての文字に固有の番号を提供します。

ユニコードを使用することによって、さまざまなプラットフォーム、言語、および国で機能するソフトウェア製品を開発することができます。

ユニコードを使用する理由

オペレーティング・システムは、マルチリンガル・サポート機能を提供します。ユニコードは、単一のファイルで選択した各国語でデータを保管および検索する手段を提供します。そのため、ユニコードは、入力装置の言語に関係なく、すべてのテキストのニーズをサポートする 1 つのデータベース・ファイルを提供します。例えば、同一ファイルに、ギリシャ語、ロシア語、および英語のテキストを含めることができます。

関連情報:

ユニコードの処理

制御言語でのユニコードの設計

制御言語 (CL) でのユニコード・サポートにより、コマンド処理プログラム (CPP) は、データを CL に渡す方法にかかわらず、常に拡張 2 進化 10 進コード (EBCDIC) または UTF-16 のいずれかでデータを取得できるようになります。これは、ユニコード・データをアプリケーションに渡すのに役立つ 2 つの関連した部分を持っているといえます。

パラメーター・サポート

CL での PARM サポートにより、このパラメーターを EBCDIC として表示するかユニコードとして表示するかを指定できるようになります。EBCDIC がデフォルトです。ユニコードを指定すると、この値が EBCDIC として CL に提供される場合でも、CPP は常にこの値をユニコードで受け取ります。

パーサー・サポート

CL でのパーサー・サポートは、提供された入力を変換するため、CPP は常に、ユーザーが指定したタイプの値を EBCDIC またはユニコードのいずれかで取得します。これにより、ユニコードをジョブ CCSID に変換したり、ジョブ CCSID をユニコードに変換したりすることができます。

CL ランタイムは、提供されたパラメーター値を要求されたタイプに自動的に変換するため、CPP は、入力がユニコードとして提供されたかどうかを識別する必要はありません。

このサポートがあるため、1 つの CL コマンドは、ユニコード・データと非ユニコード・データの両方への呼び出しをサポートできます。

CPP へのユニコードの引き渡しを指定する方法

ユニコード・サポートは、パラメーター (PARM) コマンドの値 (CCSID) パラメーターの CCSID ではオプションです。この値は、TYPE パラメーター値が *CHAR または *PNAME の場合のみ指定できます。以下に、CCSID パラメーターで指定可能な値を示します。

***JOB** コマンド・ストリングが最初はユニコードだった場合は、値はジョブ CCSID に変換されます。元のコマンド・ストリングがユニコードではなかった場合は、ジョブ CCSID であると見なされ、変換は行われません。

***UTF16**
パラメーター値は UTF-16 に変換されます。元の入力がユニコードではなかった場合は、ジョブ CCSID であると見なされます。

例: EBCDIC およびユニコード値の引き渡し

この例では、拡張 2 進化 10 進コード (EBCDIC) およびユニコード値を渡すコマンドの指定方法を示します。

```
START:      CMD        PROMPT('EXAMPLE FOR UNICODE')
            PARM       KWD(STRING1) TYPE(*CHAR) LEN(40) DFT(ABC123) +
                         MIN(0) CCSID(*JOB) PROMPT('String one') +
                         /* Passed in job CCSID (EBCDIC) */

            PARM       KWD(STRING2) TYPE(*CHAR) LEN(40) DFT(ABC123) +
                         MIN(0) CCSID(*UTF16) PROMPT('String two') +
                         /* Passed in Unicode (UTF-16)*/
```

例では、STRING1 はジョブ CCSID でコマンド処理プログラム (CPP) に送信され、STRING2 はユニコードで送信されます。CL は、提供されるすべての値を変換して、指定された値と一致するようにします。ユニコードは、*PNAME パラメーターまたは *CHAR パラメーターでのみサポートされます。

*PNAME または *CHAR が TYPE パラメーターに指定され、*UTF16 が値 (CCSID) パラメーターの CCSID に指定される場合は、CPP に渡されるバイト数は、LEN 値に指定された数値の 2 倍になることがあります。 CPP に渡される値は、右側の UTF-16 ブランク文字で増加します。例えば、データ X'00410042' の長さが 2 文字のみでも、フィールドの長さが 4 文字の場合は、CPP に渡されるデータは x'0041004200200020' になります。

パラメーター定義 (PARM) コマンドの情報には、パラメーター値にユニコードを指定した場合の影響に関する詳細が含まれています。

関連情報:

パラメーター定義 (PARM)

ユニコードが使用可能なコマンドの呼び出し

ユニコードが使用可能な CL コマンドにユニコードを渡す方法を確認するには、この情報を参照してください。この情報は、ユニコードが使用可能なシステム提供のコマンド、およびユニコードが使用可能なサポートを活用するユーザー作成のコマンドに適用されます。

ユニコードの CL コマンド・サポート

いくつかのシステム提供の CL コマンドでは、一部のパラメーターでユニコードがサポートされます。このようなコマンドには、パラメーターでユニコードが使用可能なことを記した資料の注記があります。コマンドの呼び出し方法によっては、任意のユニコード文字を含めることができるパラメーターのユニコード値にこれらのコマンドを渡すことができます。

例えば、ディレクトリー作成 (MKDIR) コマンドのディレクトリー (DIR) パラメーターでは、ユニコードが使用可能です。

DIR パラメーターでユニコードが使用可能になる前には、MKDIR コマンドを発行するためにコマンド処理 (QCAPCMD) API を呼び出したプログラムがある場合は、ジョブの CCSID で DIR 名を渡す必要がありました。これによって、EBCDIC CCSID を選択しなければならなくなり、選択肢が限定されることがありました。例えば、この制限は、ギリシャ語文字とロシア語文字の両方を使用したディレクトリーを作成するような場合に発生します。 EBCDIC CCSID には両方の文字セットが含まれていないため、このようなディレクトリーを作成することはできませんでした。

現在は、DIR パラメーターでユニコードが使用可能であるため、QCAPCMD API を呼び出して、MKDIR コマンドでユニコードのディレクトリー・パス名に渡すようにプログラムを変更できます。名前はユニコード形式であるため、ギリシャ語文字とロシア語文字の両方を使用できます。

コマンドでユニコードが使用可能になっていても、コマンドの他の呼び出しには影響しません。例えば、QCMD プロンプト行からコマンドへの呼び出しに変更はありません。ただし、そのような呼び出しは、ユニコードの使用可能性を活用することはできません。

QCAPCMD API を呼び出して、ユニコードを CL コマンドに渡す方法

QCAPCMD API を呼び出す場合、ストリング全体を QCAPCMD API に EBCDIC として渡す方法と、ユニコード値として渡す方法のいずれかを選択することができます。

以下に、指定可能な CCSID 値を示します。

- 0。コマンドの入力ストリングはジョブ CCSID の形式です。
- 1208。コマンド・ストリングは UTF-8 形式です。
- 1200。コマンド・ストリングは UTF-16 形式です。

この値を 1208 または 1200 に設定する場合は、入力ストリング全体をユニコードで渡す必要があります。つまり、ユニコードに設定したパラメーターだけでなく、すべてのデータをユニコード値として渡す必要があります。

UTF-8 ソース・ファイルを使用して、ユニコードを CL コマンドに渡す方法

任意のユニコード文字 (UTF-8 でエンコードされた文字) を含むことができるソース・メンバーを作成して、これを CL に投入して処理することができます。例えば、これによってバッチ・ジョブは、名前の言語に関係なく、統合ファイル・システムのオブジェクト・セット全体をコピーすることができます。 CL でユニコードが使用可能になる前は、すべての情報が 1 つの EBCDIC ジョブ CCSID に変換され、ジョブ CCSID は複数の言語をサポートしていないため、これを単一のバッチ・ジョブで行うことはできませんでした。

以下に、UTF-8 でエンコードされたソース・メンバーおよびロシア語とギリシャ語のファイル名の例を示します。

```
//BCHJOB
MKDIR DIR('~/sample')
MKDIR DIR('~/sample/one')
MKDIR DIR('~/sample/K■ντρα πληροφορι■ν')
MKDIR DIR('~/sample/πο οποο')
MKDIR DIR('~/sample/my backup info')
//ENDBCHJOB
```

UTF-8 ファイルは、次のコマンドで作成できます。

```
CRTSRCPF FILE(MYLIB/UTF8TEST) MBR(TEST) TEXT('test of utf8 file') CCSID(1208)
```

この後で、UTF-8 が使用可能なメソッドを使用して、ファイルを更新する必要があります。例えば、Rational Development Studio for i ライセンス・プログラムを使用して、この UTF-8 ファイルを直接編集することができます。次に、データベース・ジョブ投入 (SBMDBJOB) コマンドを使用して、要求を発行することができます。

ユニコードの QCMD プロンプト行サポート

システムの QCMD プロンプト行は、ユニコードをサポートしていません。

関連情報:

Process Commands (QCAPCMD) API

テープまたは光メディアからのアプリケーションのロードおよび実行

媒体プログラムのロードおよび実行 (LODRUN) コマンドは、別のユーザーまたはソフトウェア・ベンダーによって作成されたアプリケーションを、他のユーザーによって提供されたテープまたは光メディアからロードし、実行します。

LODRUN コマンドが実行されると、次のようになります。

- QINSTAPP という名前のユーザー作成のプログラムを見つけるために媒体が検索されます。テープが使用されている場合、そのテープが最初に巻き戻されます。
- QINSTAPP プログラムが既にユーザーのシステムの QTEMP ライブラリー内に存在している場合、それは削除されます。
- **オブジェクト復元 (RSTOBJ)** コマンドを使用して、QINSTAPP プログラムが QTEMP ライブラリーに復元されます。
- システムの制御は QINSTAPP プログラムに渡されます。 QINSTAPP プログラムは、例えば、他のアプリケーションをユーザーのシステムに復元し、それらのアプリケーションを実行するのに使用できます。

関連情報:

ロード実行 (LODRUN) コマンド

例: QINSTAPP プログラム

これは、テープまたは光メディアに保存可能で、**メディア・プログラムのロードおよび実行 (LODRUN)** コマンドを使用してシステムにロード可能なプログラム例です。

LODRUN コマンドは、システムの制御をプログラムに渡し、次にプログラムに書き込まれたタスクを実行します。

この例のプログラムは、多くの異なるタスクを実施するように作成できます。例えば、このプログラムは次のことを行います。

- 他のプログラムまたはアプリケーションを復元し、実行する。
- ライブラリーを復元する。
- 別のプログラムまたはアプリケーションを削除する。
- 特定の環境を作成する。
- 既存のアプリケーションの問題を訂正する。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM         PARM(&DEV)  /* "Device" is only Parm allowed          */
DCL         VAR(&DEV)   TYPE(*CHAR) LEN(10)
DCL         VAR(&MODEL) TYPE(*CHAR) LEN(4)

/* Can check for appropriate model number, release level, and so on */
RTVSYSVAL   SYSVAL(QMODEL) RTNVAR(&MODEL)
IF          (&MODEL *EQ 'xxxxx') THEN...

/* Install a library for new application (programs, data):        */
RSTLIB      SAVLIB(NEWAPP) DEV(&DEV) ENDOPT(*LEAVE) +
              MBROPT(*ALL)
/* Install a command to start new application:                    */
            RSTOBJ OBJ(NEWAPP) SAVLIB(QGPL) DEV(&DEV) +
              MBROPT(*ALL)

END:        ENDPGM
```

図 *7. LODRUN* コマンドを使用するアプリケーションの例

関連情報:

ロード実行 (LODRUN) コマンド

制御権転送によるパフォーマンスの向上

制御権転送 (TFRCTL) コマンドは、コマンドで指定されたプログラムを呼び出し、そのプログラムに制御権を渡し、制御権を転送したプログラムを呼び出しスタックから除去します。

呼び出しスタックにあるプログラムの数を減らせば、それだけパフォーマンスが向上します。 **呼び出し (CALL)** コマンドを使用した場合には、呼び出されたプログラムはその**呼び出し (CALL)** コマンドの入っているプログラムに制御権を戻します。一方**制御権転送 (TFRCTL)** コマンドを使用した場合は、呼び出しスタックにある最初のプログラムに制御権が戻り、それからその最初のプログラムは**呼び出し (CALL)** コマンドの次の順次命令を開始します。

注: **制御権転送 (TFRCTL)** コマンドは統合言語環境 (ILE) CL プロシージャーでは無効です。

関連情報:

制御権転送 (TFRCTL) コマンド

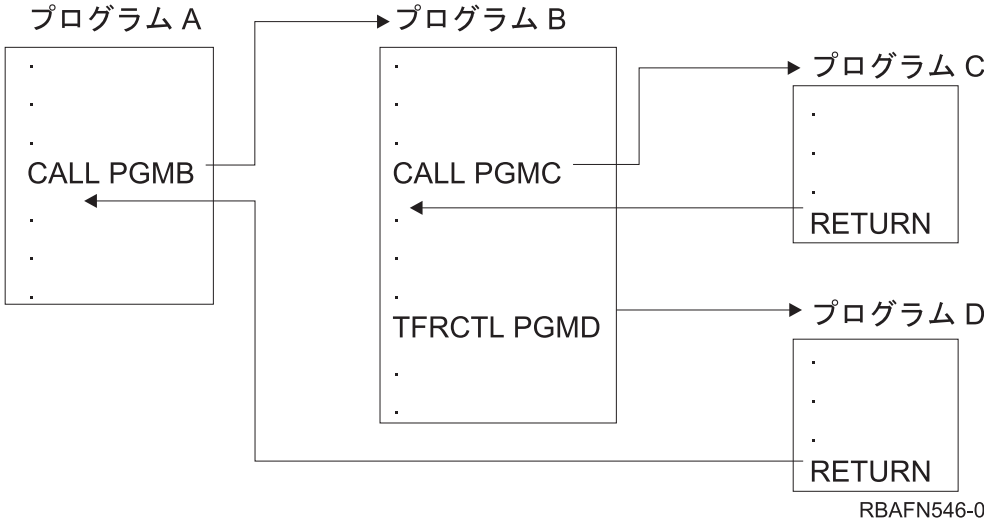
プログラム呼び出し (CALL) コマンド

例: 制御権転送コマンドの使用

以下は、パフォーマンスを向上させる制御権転送の例です。

次の図でプログラム A に対し USRPRF (*OWNER) が指定されていると、図に示されているすべてのプログラムに対してその所有者の権限が適用されます。プログラム B に対して USRPRF (*OWNER) が指定されている場合には、プログラム B およびプログラム C が活動状態にある間だけその所有者の権限が適

用されます。プログラム B がプログラム D に制御を転送すると、プログラム B はもはや呼び出しスタックに存在せず、プログラム B の所有者はプログラム D の実行中、権限を持つものとは見なされません。また、プログラムの処理が (制御権が戻るか転送されることにより) 完了すると、所有者の権限は効力を失います。プログラム B で一時変更を行った場合には、その効力はプログラム D の実行過程で存続し、プログラム D が制御権を戻した時点でその効力は失われます。



制御権転送 (TFRCTL) コマンドの形式は次のとおりです。

```
TFRCTL PGM (ライブラリー名 / プログラム名) PARM(CL 変数)
```

プログラム (およびライブラリー修飾子) は変数であっても構いません。

ここで注意しなければならないのは、このコマンドでパラメーター引数として使用できるのは変数だけであり、またそれらの変数は制御の転送を行うプログラムを呼び出したプログラムから引数リストのパラメーターとして受け取ったものでなければならないという点です。すなわち、**制御権転送 (TFRCTL)** コマンドを実行するプログラムに渡されたのではない変数は**制御権転送 (TFRCTL)** コマンドによって渡すことができないということです。

次の例では、最初の**制御権転送 (TFRCTL)** コマンドは有効です。 2 番目の**制御権転送 (TFRCTL)** コマンドは、&B がこのプログラムには渡されていないので無効です。3 番目の TFRCTL コマンドは、定数を値として指定することはできないので無効です。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM PARM(&A)
DCL &A *DEC 3
DCL &B *CHAR 10
IF (&A *GT 100) THEN (TFRCTL PGM(PGMA) PARM(&A)) /*   valid   */
IF (&A *GT 50) THEN (TFRCTL PGM(PGMB) PARM(&B))  /* not valid */
ELSE (TFRCTL PGM(PGMC) PARM('1'))                /* not valid */
ENDPGM
```

関連タスク:

281 ページの『パラメーターの受け渡し』
他のプログラムまたはプロシージャーに制御を渡す時点で、受取側のプログラムまたはプロシージャー内での変更や使用のために情報を同時に渡すことができます。

関連情報:

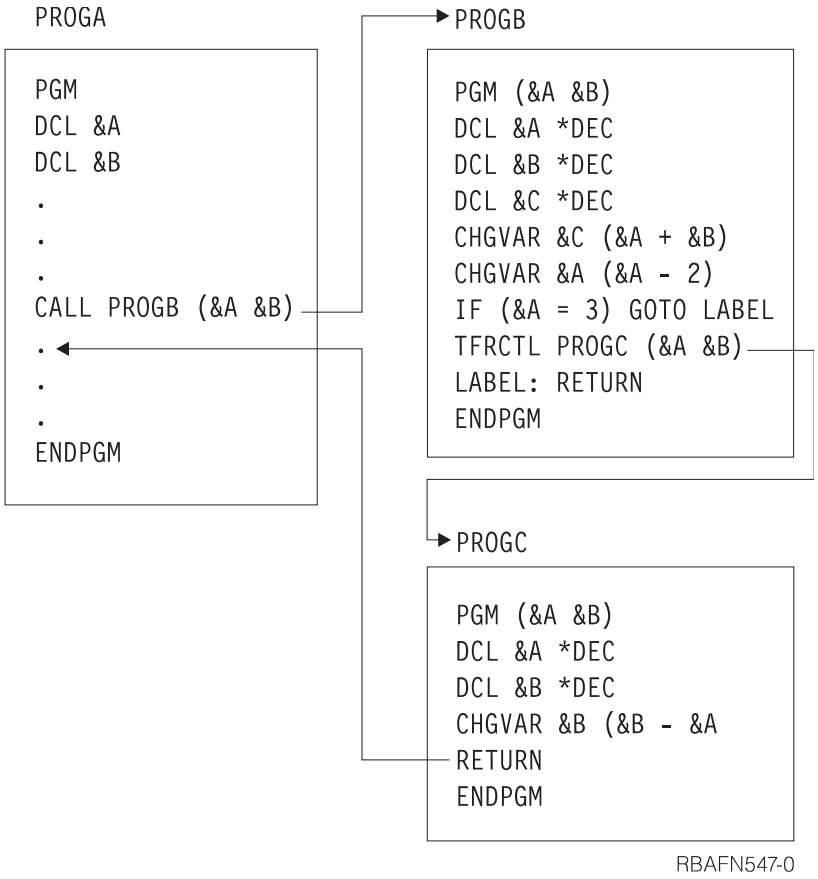
制御権転送コマンドを使用したパラメーターの引き渡し

制御権転送 (TFRCTL) コマンドは、呼び出されているプログラムにパラメーターを渡します。

TFRCTL コマンドは、CALL コマンドによってパラメーターを渡すのと同じ方法で呼び出されるプログラムにパラメーターを渡すことができますが、次のような制約があります。

- 渡されるパラメーターは CL 変数でなければならない。
- 制御の転送を行うプログラムが渡す CL 変数は、それ以前にパラメーターとしてそのプログラムが受け取った CL 変数でなければならない。
- このコマンドはオリジナル・プログラム・モデル (OPM) CL プログラム中に限り有効となる。

次の例では、PROGA は PROGB を呼び出し、&A と &B の 2 つの変数をそのプログラムに渡します。PROGB では、この 2 つの変数と、内部で宣言されている別の変数 &C が使用されます。 PROGB から PROGC に制御権を転送する場合に、PROGC に渡すことができる変数は &A と &B だけです。PROGC の処理が完了すると、制御権はこの 2 つの変数の生成元である PROGA に戻されます。



関連情報:

制御権転送 (TFRCTL) コマンド

例: CL のプログラミング

次の例は、CL プログラムの柔軟性、簡略性、および用途の広さを示します。

次の CL プログラムは、その機能および対象ユーザー別に説明します。

注: V4R3 以降のリリースで ILE CL コンパイラーが生成するコードはスレッド・セーフですが、 コマンドの多くはスレッド・セーフではありません。 したがって、CL プログラムまたは CL プロシージャーが使用するコマンドがすべてスレッド・セーフでない限りは、その CL プログラムまたは CL プロシージャーをスレッド・セーフと見なさないでください。 **コマンド表示 (DSPCMD)** コマンドを使用すれば、コマンドがスレッド・セーフであるかどうかを調べることができます。また、各コマンドの情報 (オンライン・ヘルプおよびコマンドの資料) は、コマンドがスレッド・セーフかどうかを示しています。

関連情報:

マルチスレッド・アプリケーション

例: セットアップ (プログラマー) のための初期プログラム

この例では、テスト・ライブラリーが最初にライブラリー・リストに置かれ、使用できる印刷装置用の出力待ち行列が選択され、プログラマー・メニューが表示されます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
CHGLIBL LIBL(TESTLIB QGPL QTEMP)
CHGJOB OUTQ(WSPRTR)
TFRCTL QPGMMENU
ENDPGM
```

例: 特定のオブジェクトをアプリケーションに保管する (システム・オペレーター)

このプログラム例により、定期的に繰り返されるプロシージャーへの整合したコマンド入力が確実に行えます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
SAVOBJ OBJ(FILE1 FILE2) LIB(LIBA) OBJTYPE(*FILE) DEV(TAP01) +
  CLEAR(*ALL)
SAVOBJ OBJ(DTAARA1) LIB(LIBA) OBJTYPE(*DTAARA) DEV(TAP01)
SNDPGMMSG MSG('Save of daily backup of LIBA completed') +
  MSGTYPE(*COMP)
ENDPGM
```

もちろん、この他に**オブジェクト保管 (SAVOBJ)** コマンドを追加することができます。ただし、このプログラムは、オペレーターが各アプリケーションの定期バックアップ用に正しいテープを選択することに依存しています。これは、それぞれの保管操作で各テープ・セットに固有の名前を割り当てることによって制御できます。例えば、給与計算ファイルを 4 週間、週別に保管する場合は、各テープに別々の名前を付け、テープの名前をその週の正しい名前と比較するプログラムを作成します。

例: 異常終了からのリカバリー (システム・オペレーター)

この例では、システム・オペレーターが異常終了からリカバリーします。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
DCL &SWITCH *CHAR LEN(1)
RTVSYSVAL SYSVAL(QABNORMSW) RTNVAR(&SWITCH)
IF (&SWITCH *EQ '1') THEN(DO)  /*CALL RECOVERY PROGRAMS*/
        SNDPGMMSG MSG('Recovery programs in process. +
```

```
          Do not start subsystems until notified') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
        CALL PGMA
        CALL PGMB
        SNDPGMMSG MSG('Recovery programs complete. +
          Startup subsystems') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
        RETURN
        ENDDO
ENDPGM
```

例: 表示装置ディスプレイからの入力待機時のタイムアウト

このプログラムは、表示装置ファイルを用いて、ユーザーによるオプションの入力を、指定の時間待機する CL プログラムを作成する方法を示します。時間内に入力しないと、そのユーザーはサインオフされます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
DCLF FILE(QGPL/MENU)
DOWHILE '1' /* DO FOREVER */
  SNDRCVF DEV(*FILE) RCDFMT(MENUFMT) WAIT(*NO)
  WAIT  MONMSG MSGID(CPF0889) EXEC(SIGNOFF)
  CHGVAR VAR(&IN99) VALUE('0')
  IF COND(&IN01) THEN(ITERATE)
  SELECT
    WHEN (&OPTION *EQ '1') (CALL ORDENT) /* OPTION 1-ORDER ENTRY      */
    WHEN (&OPTION *EQ '2') (CALL ORDDSP) /* OPTION 2-ORDER DISPLAY    */
    WHEN (&OPTION *EQ '3') (CALL ORDCHG) /* OPTION 3-ORDER CHANGE     */
    WHEN (&OPTION *EQ '4') (CALL ORDPRT) /* OPTION 4-ORDER PRINT      */
    WHEN (&OPTION *EQ '9') (SIGNOFF)     /* OPTION 9-SIGNOFF          */
    OTHERWISE DO                         /* OPTION SELECTED NOT VALID */
      CHGVAR VAR(&IN99) VALUE('1')
      ENDDO
  ENDSELECT
ENDDO
ENDPGM
```

表示装置ファイルは次のコマンドで作成されました。

```
CRTDSPF FILE(MENU) SRCFILE(QGPL/QDDSSRC) SRCMBR(MENU)  +
    DEV(*REQUESTER) WAITRCD(60)
```

表示装置ファイルは *REQUESTER 装置を使用します。 **待機 (WAIT)** コマンドを発行すると、WAITRCD キーワードで指定された秒数 (60) の間、待機状態になります。次に、表示装置ファイル用の DDS を示します。

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ...
6 ... ... 7 ... ... 8

 0100       A                                      PRINT CA01(01)
 0200       A          R MENUFMT                   BLINK
 0300       A                                      TEXT('Order
Entry Menu')
 0400       A                                    1 31'Order Entry Menu'
 0500       A                                    2  2'Select one
of the following:    '
 0600       A                                    3  4'1.  Enter Order'
 0700       A                                    4  4'2.  Display Order'
 0800       A                                    5  4'3.  Change Order'
 0900       A                                    6  4'4.  Print Order'
 1000       A                                    7  4'9.  Sign Off'
 1100       A                                   23  2'Option:'
 1200       A            OPTION         1   I 23 10
 1300       A  99                                  ERRMSG('Invalid
```

```
option selected.')
```

＊ ＊ ＊ ＊　ソース仕様の終わり　＊ ＊ ＊ ＊

このプログラムは、SNDRCVF WAIT(*NO) を実行してメニューを表示し、ユーザーからのオプションを要求します。次に、WAIT コマンドを発行して、ユーザーからのオプションを受諾します。ユーザーが 1 から 4 を入力すると、該当のプログラムが呼び出されます。ユーザーが 9 を入力すると、SIGNOFF コマンドが発行されます。ユーザーが無効なオプションを入力すると、メニューには、'OPTION SELECTED NOT VALID' メッセージが表示されます。ユーザーは新たに有効なオプションを入力できます。ユーザーが 60 秒以内に応答しないと、CPF0889 メッセージがプログラムに出され、**メッセージ監視 (MONMSG)** コマンドが SIGNOFF コマンドを発行します。

INVITE DDS キーワードが入ったレコード様式を使用する **ファイル送信 (SNDF)** コマンドが、SNDRCVF WAIT(*NO) の代わりに使用される場合もあります。機能は同じです。

例: 日付演算の実行

このプログラムは、現行のシステム日時に対して特定の日数を加算または減算する CL プログラムを作成する方法を示しています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
/* Calculate new date from current system date.  Pass negative     */
/* number to subtract, positive number to add                      */
/*                                                                 */
/* The first parameter is a character 8 date in YYYYMMDD format    */
/* or the special value *CURRENT                                   */
/*                                                                 */
/* The second parameter is a decimal value for the number of days  */
/* to adjust the first parameter by                                */
/*                                                                 */
/* Test cases:  CALL CALCDATE (*CURRENT -5)                        */
/*              CALL CALCDATE (*CURRENT  5)                        */
/*              CALL CALCDATE ('20030225' -90)                     */
/*              CALL CALCDATE ('30020228' 90)                      */
/*                                                                 */
/* There is no error handling in this sample, so make sure the     */
/* input dates are valid (that is, no 20031325).  The valid date   */
/* date range is Oct 14 1582 to Dec 31 9999                        */
/*                                                                 */
             PGM        PARM(&curdate &DAYSTOCHG)
             DCL        VAR(&CURDATE) TYPE(*CHAR) LEN(8)
             DCL        VAR(&DAYSTOCHG) TYPE(*DEC) LEN(15 5)
             DCL        VAR(&DATETIME) TYPE(*CHAR) LEN(17)
             DCL        VAR(&DATE) TYPE(*CHAR) LEN(8)
             DCL        VAR(&LILDATEINT) TYPE(*CHAR) LEN(4)
             DCL        VAR(&LILDATEDEC) TYPE(*DEC) LEN(10 0)
             DCL        VAR(&ERRCOD) TYPE(*CHAR) LEN(4) +
                          VALUE(X'00000000')
             DCL        VAR(&MSG) TYPE(*CHAR) LEN(50)
             IF         COND(&CURDATE = '*CURRENT') THEN(DO)
             CALL       PGM(QWCCVTDT) PARM('*CURRENT' ' ' '*YYMD' +
                          &DATETIME &ERRCOD) /* Get current system +
                          date and time in YYYYMMDD */
             CHGVAR     VAR(&DATE) VALUE(%SST(&DATETIME 1 8)) /* Get +
                           just the date portion */
             ENDDO
             ELSE       CMD(CHGVAR VAR(&DATE) VALUE(&CURDATE)) /* +
                           Use the date provided */
             CALLPRC    PRC(CEEDAYS) PARM(&DATE 'YYYYMMDD' +
                           &LILDATEINT *OMIT) /* Get Lilian date for +
                           current date */
```

```
CHGVAR     VAR(&LILDATEDEC) VALUE(%BIN(&LILDATEINT)) /* +
             Get Lilian date in decimal format */
CHGVAR     VAR(&LILDATEDEC) VALUE(&LILDATEDEC + +
             &DAYSTOCHG) /* Adjust specified number +
             of days */
CHGVAR     VAR(%BIN(&LILDATEINT)) VALUE(&LILDATEDEC) /* +
             Get Lilian date in integer format */
CALLPRC    PRC(CEEDATE) PARM(&LILDATEINT 'YYYYMMDD' +
             &DATE *OMIT) /* Return calculated date in +
             YYYYMMDD format */
CHGVAR     VAR(&MSG) VALUE('The new date is ' *CAT &DATE)
SNDPGMMSG  MSG(&MSG) TOPGMQ(*EXT)
ENDPGM
```

CL プログラムおよびプロシージャーのデバッグ

デバッグとは、統合言語環境 (ILE) またはオリジナル・プログラム・モデル (OPM) プログラムで、エラーを検出、診断、および除去することです。

ILE プログラムのデバッグ

統合言語環境 (ILE) プログラムをデバッグするには、ILE ソース・デバッガーを使用します。

次のタスクを実行することができます。

- ユーザーの ILE プログラムのデバッグを準備する。
- デバッグ・セッションを開始する。
- デバッグ・セッションでプログラムの追加や除去を行う。
- デバッグ・セッションのプログラム・ソースを表示する。
- 条件付きブレークポイントおよび無条件ブレークポイントを設定および除去する。
- プログラムをステップスルーする。
- 変数の値を表示する。
- 変数の値を変更する。
- 変数の属性を表示する。
- 省略名を変数、式、またはデバッグ・コマンドと等しくする。

ユーザー・プログラムのデバッグとテストを行う際には、ライブラリー・リストを変更して、テスト・データを含むテスト・ライブラリーにプログラムを入れ、既存のどの実データにも決して影響を与えないようにしてください。

次のどちらかのコマンドを使用すると、実動 (プロダクション) ライブラリー内のデータベース・ファイルが不慮に修正されないようにすることができます。

- **デバッグ開始 (STRDBG)** コマンドを使用して、UPDPROD パラメーターのデフォルト値 *NO をそのまま使用する。
- **デバッグ変更 (CHGDBG)** コマンドを使用する。

ILE ソース・デバッガーを使用して、プログラム・オブジェクトやサービス・プログラム中のエラーを検出し、除去します。ソース・デバッガーを使用すると、以下の操作を行えます。

- 任意の ILE CL アプリケーションまたは混合 ILE 言語アプリケーションをデバッグする。
- プログラムの実行中にデバッグ・コマンドを使用して、プログラムのフローをモニターする。
- プログラム・ソースを表示する。

- 条件付きブレークポイントおよび無条件ブレークポイントを設定および除去する。
- ステートメントを指定数だけステップスルーする。
- 変数の値の表示や変更を行う。
- 変数の属性を表示する。

ブレークポイントかステップ・コマンドのためにプログラムが停止すると、そのプログラムが停止した個所の該当するモジュール・オブジェクトのビューが表示されます。この個所でデバッグ・コマンドを追加入力できます。

制御言語モジュール作成 (CRTCLMOD) または **バインド CL PGM の作成 (CRTBNDCL)** を使用してモジュール・オブジェクトやプログラム・オブジェクトを作成する場合は、まずデバッグ・オプション (DBGVIEW) を使用してからソース・デバッガーを使用してください。ブレークポイントか他の ILE ソース・デバッガーを設定した後に、プログラムを呼び出すことができます。

関連タスク:

441 ページの『デバッグ・モードの開始』
デバッグ・モードは特殊な環境で、通常のシステム機能に加えてテスト機能を使用することができます。

関連資料:

101 ページの『テストおよびデバッグに使用するパラメーター値』
IBM i オペレーティング・システムには、プログラムの実行時に行われる処理をプログラマーが監視できる機能が組み込まれています。

関連情報:

デバッグ変更 (CHGDBG) コマンド

デバッグ開始 (STRDBG) コマンド

CL コマンド検索プログラム

ILE の概念

デバッグ・コマンド

統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

デバッグ・コマンドとそのパラメーターは、デバッグ・コマンド行に入力します。この行は、「モジュール・ソースの表示」画面と「評価式」画面の最下部に表示されます。これらのコマンドは、大文字、小文字、または混合文字のどれでも入力できます。

注: ソース・デバッガー・コマンド行に入力するデバッグ・コマンドは、CL コマンドではありません。

以下の表に、これらのデバッグ・コマンドを要約します。ILE ソース・デバッガーのオンライン・ヘルプには、デバッグ・コマンドが記載され、その有効な省略語が説明されています。

表 25. *ILE* ソース・デバッガー・コマンド

デバッグ・コマンド	説明
ATTR	変数の属性を表示できる。属性とは、デバッグ・シンボル・テーブルに記録されている変数のサイズとタイプのことです。
BREAK	テストされるプログラム中にある位置に無条件ブレークポイントか条件付きブレークポイントのどちらかを入力できる。条件付きブレークポイントを入力する場合は、BREAK 位置 WHEN 式 を使用します。

表 25. *ILE* ソース・デバッガー・コマンド *(*続き*)*

デバッグ・コマンド	説明
SBREAK	テストされるプログラム中にある位置にサービス・エントリー・ポイントを入力できる。サービス・エントリー・ポイントはプログラム内に設定されたブレークポイントのタイプの 1 つで、システム・デバッガーが発生したジョブの制御権を得るのを容易にします。ブレークポイントには、サービス・エントリー・ポイントにヒットしたジョブが現在デバッグ中でない場合にのみシグナルが送られます。
CLEAR	条件付きブレークポイントと無条件ブレークポイントを除去できる。
DISPLAY	EQUATE コマンドを使用して割り当てた名前と定義を表示できる。「モジュール・ソースの表示」画面に現在表示されていないソース・モジュールも表示できます。モジュール・オブジェクトは、現行のプログラム・オブジェクト中に存在している必要があります。
EQUATE	省略した名前に式、変数、またはデバッグ・コマンドを割り当てることができる。
EVAL	変数の値の表示や変更を行ったり、式の値の変更を行うことができる。
QUAL	後続の EVAL コマンドに表示される変数の有効範囲を定義できる。
STEP	デバッグされるプログラムの 1 つまたは複数のステートメントを実行できる。
FIND	現在表示されているモジュールを探索して、指定された行番号、またはテキストのストリングを見つける。
UP	表示されるソースのウィンドウを、入力量に応じてビューの先頭方向に移動する。
DOWN	表示されるソースのウィンドウを、入力量に応じてビューの末尾方向に移動する。
LEFT	表示されるソースのウィンドウを、入力する文字数に応じて左方に移動する。
RIGHT	表示されるソースのウィンドウを、入力する文字数に応じて右方に移動する。
TOP	ビューの位置を移動して、最初の行が見えるようにする。
BOTTOM	ビューの位置を移動して、最後の行が見えるようにする。
NEXT	現在表示されているソースの次のブレークポイントにビューの位置を移動する。
PREVIOUS	現在表示されているソースの直前のブレークポイントにビューの位置を移動する。
HELP	使用できるソース・デバッガー・コマンドのオンライン・ヘルプ情報を表示する。
SET	デバッグ・オプション (FIND 要求の大/小文字の区別、デバッグ・モードで実動ファイルを更新できるか、オリジナルのプログラム・モデル (OPM) ソース・デバッグのサポートを使用可能または使用不可にする、など) を変更できる。
WATCH	現在活動状態のウォッチの状況のリストを表示する。

関連タスク:

422 ページの『デバッグ・セッションへのプログラム・オブジェクトの追加』
デバッグ・セッションを開始した後、セッションにプログラム・オブジェクトをさらに追加することができます。

437 ページの『変数の値の変更』
変数の値を変更するには、EVAL コマンドと代入演算子 (=) を使用します。

439 ページの『名前を変数、式、またはコマンドと等しくする』
名前を変数、式、またはデバッグ・コマンドと簡単に等しくするには、EQUATE デバッグ・コマンドを使用します。

関連資料:

430 ページの『BREAK および CLEAR デバッグ・コマンドを使用した、条件付きブレークポイントの設定および削除』
条件付きブレークポイントの設定と除去を行う別の方法として、BREAK および CLEAR デバッグ・コマンドを使用する方法があります。

429 ページの『条件付きブレークポイントの設定および除去』
「モジュール・ブレークポイントの処理」画面を使用して、ブレークポイントを設定または除去できます。

434 ページの『変数の表示』
変数の値を表示するには、「モジュール・ソースの表示」画面または EVAL デバッグ・コマンドを使用します。

439 ページの『変数の属性の表示』
変数の属性とは、変数のサイズ (バイト単位) とタイプです。この属性を表示するには、属性 (ATTR) デバッグ・コマンドを使用します。

デバッグ・セッションに対するプログラム・オブジェクトの準備

統合言語環境 (ILE) ソース・デバッガーを使用するには、まずプログラム・オブジェクトを準備する必要があります。

そのためには、**制御言語モジュール作成 (CRTCLMOD)** または**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドのいずれかを使用して、DBGVIEW オプションを指定する必要があります。

デバッグする ILE CL モジュール・オブジェクトごとに、以下の 3 つのビューのうちのいずれかを作成できます。

- ルート・ソース・ビュー
- リスト・ビュー
- ステートメント・ビュー

関連情報:

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

ルート・ソース・ビューを使用した **ILE** プログラムのデバッグ**:**

ルート・ソース・ビューには、ソース・メンバーのソース・ステートメントがあります。

統合言語環境 (ILE) ソース・デバッガー使用時にルート・ソース・ビューを使用すると、モジュール・オブジェクト (*MODULE) を作成する際に ILE CL コンパイラーによりルート・ソース・ビューが作成されます。

注**:** モジュール・オブジェクトを作成する場合、ビューにソース・ステートメントを複写する代わりにルート・ソース・メンバーのソース・ステートメントの位置を参照します。したがって、モジュールを作成してからルート・ソース・メンバーに基づいて作成されたモジュールをデバッグするまでの間に、そのメンバーの修正、名前変更、または移動を行わないでください。

ルート・ソース・ビューを使用して ILE CL モジュール・オブジェクトをデバッグする場合は、**CL モジュール作成 (CRTCLMOD)** コマンドか **バインド CL 作成 (CRTBNDCL)** コマンドの DBGVIEW パラメーターに *SOURCE オプションか *ALL オプションを使用してください。

ルート・ソース・ビューの作成方法の一例を次に示します。

```
CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*SOURCE)
```

DBGVIEW パラメーターに *SOURCE を指定して **CL モジュール作成 (CRTCLMOD)** コマンドを使用すると、モジュール・オブジェクト *MYPGM* のルート・ソース・ビューが作成されます。

関連情報:

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

リスト・ビューを使用しての **ILE** プログラムのデバッグ**:**

リスト・ビューは、統合言語環境 (ILE) CL コンパイラーで作成されるコンパイル・リストまたはスプール・ファイルのソース・コード部分と同様です。

リスト・ビューを使用して ILE CL モジュール・オブジェクトをデバッグする場合は、モジュールの作成時に **制御言語モジュール作成 (CRTCLMOD)** コマンドまたは**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドの DBGVIEW パラメーターに *LIST または *ALL オプションを使用します。

リスト・ビューの作成方法の一例を次に示します。

```
CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*LIST)
```

関連情報:

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

デバッグ・リスト・ビューの暗号化**:**

コンパイルした CL プロシージャーのデバッグ・リスト・ビューを生成モジュール (*MODULE) オブジェクトに格納する際に、暗号化したものを格納することができます。

制御言語モジュール作成 (CRTCLMOD) コマンドまたは**バインド制御言語プログラム作成 (CRTBNDCL)** コマンドでデバッグ・ビュー (DBGVIEW) パラメーターに *LIST を指定することによって、 ILE CL プロシージャーのコンパイラー・リスト・ビューを CL モジュールに作成することができます。リスト・ビューを使用すると、ILE CL プロシージャーのデバッグが容易になります。ただし、リスト・ビューは、その CL モジュールを含むプログラムまたはサービス・プログラムのオブジェクトに対して十分な権限を持っていれば、誰でも参照できてしまいます。

IBM i 7.1 では、 ILE 制御言語モジュール作成時にリスト・ビューのデバッグ・データを暗号化できるようにするサポートが追加されました。暗号鍵の値は、**CRTCLMOD** コマンドおよび **CRTBNDCL** コマンドのデバッグ暗号鍵 (DBGENCKEY) パラメーターに指定できます。デバッグ・セッションを開始すると、暗号鍵の値を求められます。 CL モジュール作成時に指定された値と同じ値がデバッグ・セッションに指定されないと、リスト・ビューは表示されません。

このデバッグ・リスト・ビュー暗号化のサポートにより、別のシステムのユーザーがデバッグ・リスト・ビューを使用して CL ソース・コードを参照できてしまうことがないようにして、プログラムまたはサービス・プログラムをそのシステムに送ることができるようにします。 CL プロシージャーで問題が発生した場合、その別のシステムにリモートでサインオンします。別のシステムにサインオンすると、デバッグ・セッションを開始して、暗号鍵を提供できるようになります。暗号鍵が入力されると、リスト・ビューのデータがデバッグ・セッションで入手可能になります。

関連情報:

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

ステートメント・ビューを使用した **ILE** プログラムのデバッグ**:**

ステートメント・ビューには CL ソース・データがまったく含まれていません。ただし、コンパイラー・リスト中にあるプロシージャー名とステートメント番号を使用してブレークポイントを追加することができます。

ステートメント・ビューを使用して統合言語環境 (ILE) CL モジュール・オブジェクトをデバッグする場合には、コンパイラー・リストの複写が必要です。

注**:** ステートメント・ビューを使用して ILE CL モジュール・オブジェクトをデバッグする場合には、「モジュール・ソースの表示」画面にデータが表示されません。

ステートメント・ビューを使用して ILE CL モジュール・オブジェクトをデバッグする場合には、モジュールを作成する際に **CL モジュール作成 (CRTCLMOD)** コマンドか**バインド CL 作成 (CRTBNDCL)** コマンドの DBGVIEW パラメーターに *STMT、*SOURCE、*LIST、または *ALL オプションを使用してください。

ステートメント・ビューの作成方法の一例を次に示します。

```
CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QLSRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*STMT)
```

関連情報:

バインド制御言語プログラム作成 (CRTBNDCL) コマンド

制御言語モジュール作成 (CRTCLMOD) コマンド

ILE ソース・デバッガーの開始

デバッグ・ビューの作成後に、統合言語環境 (ILE) ソース・デバッガーを使用してアプリケーションのデバッグを開始できます。

ILE ソース・デバッガーを開始する場合には、 **デバッグ開始 (STRDBG)** コマンドを使用してください。デバッガーを開始すると、**デバッグ・モード終了 (ENDDBG)** コマンドを入力するまで活動状態を保ちます。

最初に、デバッグ・セッションに 20 個までのプログラム・オブジェクトと、20 個までのサービス・プログラムを追加できます。 これは、STRDBG コマンドのプログラム (PGM) パラメーターおよびサービス・プログラム (SRVPGM) パラメーターを使用して行います。 プログラム・オブジェクトの、ILE プログラムまたはオリジナル・プログラム・モデル (OPM) プログラムの組み合わせは任意です。 3 つのプログラム・オブジェクトを使用してデバッグ・セッションを開始する場合には、次のように入力します。

```
STRDBG PGM(*LIBL/MYPGM1 *LIBL/MYPGM2 *LIBL/MYPGM3) SRVPGM(*LIBL/SRVPGM1 *LIBL/SRVPGM2)
DBGMODSRC(*YES)
```

注**:** プログラム・オブジェクトをデバッグ・セッションに追加する際には、そのオブジェクトに対する *CHANGE 権限が必要です。

STRDBG コマンドを入力すると、ILE プログラム・オブジェクトに関する「モジュール・ソースの表示」画面が表示されます。そのプログラム・オブジェクトにバインドされている最初のモジュール・オブジェクトとデバッグ・データが表示されます。

ILE ソース・デバッガーを使用して OPM プログラムのデバッグを行うためのオプションが、ユーザーに用意されています。 OPM プログラムの作成時に、ソース・デバッグ・データが組み込まれます。これ

は、**CL プログラム作成 (CRTCLPGM)** コマンドに、OPTION(*SRCDBG) または OPTION(*LSTDBG) パラメーターだけを指定して行います。ソース・デバッグ・データは実際にはプログラム・オブジェクトの一部です。

ソース・デバッグ・データを含んで、作成された OPM プログラムを ILE ソース・デバッガーに追加するには、**デバッグ開始 (STRDBG)** コマンドでプログラム (PGM) および OPM ソース・レベル・デバッグ (OPMSRC) パラメーターを使用します。ソース・デバッグ・データを含んで作成された OPM プログラムを使用してデバッグ・セッションを開始するには、次のように入力します。

```
STRDBG PGM(*LIBL/MYOPMPGM) OPMSRC(*YES) DSPMODSRC(*YES)
```

関連情報:

デバッグ開始 (STRDBG) コマンド

デバッグ・セッションへのプログラム・オブジェクトの追加

デバッグ・セッションを開始した後、セッションにプログラム・オブジェクトをさらに追加することができます。

統合言語環境 (ILE) プログラム・オブジェクトおよびサービス・プログラムをデバッグ・セッションに追加する場合は、オプション 1 (プログラムの追加) を使用し、「モジュール・リストの処理」画面の最初の行にプログラム・オブジェクトの名前を入力してください。「モジュール・ソースの表示」画面で F14 (モジュール・リストの処理) を押すと、「モジュール・リストの処理」画面にアクセスできます。サービス・プログラムを追加する場合は、デフォルトのプログラム・タイプを *PGM から *SRVPGM に変更してください。デバッグ・セッションに入れられる ILE プログラム・オブジェクトとサービス・プログラムの数に制限はなく、任意の時点で行えます。

```
                          モジュール・リストの処理
                                                        システム:   SYSTEM01
オプションを入力して，実行キーを押してください。
  1= プログラムの追加    4= プログラムの除去    5= モジュール・ソースの表示
  8= モジュール停止点の処理

OPT     プログラム/モジュール    ライブラリー        タイプ
 1       weekday2               *LIBL              *PGM
         DSPWKDAY               MYLIB              *PGM
         DSPWKDAY                                  *MODULE      選択済み
         AABP1                                     *MODULE




                                                                  終わり
コマンド
===>
F3=終了    F4=プロンプト    F5=最新表示    F9=コマンドの複写    F12=取り消し
```

図 *8.* デバッグ・セッションへの *ILE* プログラム・オブジェクトの追加： 実行キーを押すと、プログラム WEEKDAY2 がデバッグ・セッションに追加される。

```
                          モジュール・リストの処理
                                                          システム:   SYSTEM01
オプションを入力して、実行キーを押してください。
  1= プログラムの追加    4= プログラムの除去    5= モジュール・ソースの表示
  8= モジュール停止点の処理

Opt     Program/module     Library        Type
         WEEKDAY2          *LIBL          *PGM
         WEEKDAY2          MYLIB          *PGM
         WEEKDAY2                         *MODULE
         DSPWKDAY          MYLIB          *PGM
         DSPWKDAY                         *MODULE     選択済み
         AABP1                            *MODULE




                                                                  終わり
コマンド
===>
F3=終了    F4=プロンプト    F5=最新表示    F9=コマンドの複写    F12=取り消し
ソース・デバッガーにプログラム WEEKDAY2 が追加された。
```

図 *9.* デバッグ・セッションへの *ILE* プログラム・オブジェクトの追加： 画面の最下部の情報メッセージは、プログラム WEEKDAY2 がデバッグ・セッションに追加されたことを示している。

プログラム・オブジェクトのデバッグ・セッションへの追加が終了したら、「モジュール・リストの処理」画面で F3 (終了) を押して「モジュール・ソースの表示」画面に戻ってください。オプション 5 (モジュール・ソースの表示) を使用して、モジュールの選択や表示を行うこともできます。

オリジナル・プログラム・モデル (OPM) プログラムをデバッグ・セッションに追加するには、**プログラムの追加 (ADDPGM)** コマンドを使用してください。デバッグ・セッションには、任意の時点で 20 個までの OPM プログラムを入れることができます。ソース・デバッグ・データを含む OPM プログラムをデバッグ・セッションに追加するには、「モジュール・リストの処理」画面のオプション 1 (プログラムの追加) を使用します。 (これは、デバッグ・セッションが OPM ソース・レベル・デバッグを許可する場合に可能です。)OPM ソース・レベル・デバッグを許可するには、デバッグ・セッションを開始する際に、**デバッグ開始 (STRDBG)** コマンドで OPMSRC パラメーターを使用します。OPMSRC パラメーターが**デバッグ開始 (STRDBG)** コマンドで指定されていない場合は、OPM ソース・レベル・デバッグを活動化してください。 これを行うには、**デバッグ変更 (CHGDBG)** コマンドで OPM ソース・レベル・デバッグ (OPMSRC) パラメーターを使用します。代わりに SET デバッグ・コマンドを使用して、OPM ソース・デバッグ・サポート・オプションを変更することもできます。

関連概念:

417 ページの『デバッグ・コマンド』
統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

関連情報:

デバッグ変更 (CHGDBG) コマンド

プログラム追加 (ADDPGM) コマンド

デバッグ開始 (STRDBG) コマンド

デバッグ・セッションからのプログラム・オブジェクトの除去

デバッグ・セッションを開始した後で、プログラム・オブジェクトをそのセッションから除去できます。

統合言語環境 (ILE) プログラム・オブジェクトおよびサービス・プログラムをデバッグ・セッションから除去する場合は、「モジュール・リストの処理」画面で、除去したいプログラム・オブジェクトの横にオプ

ション 4 (プログラムの除去) を入力してください。「モジュール・ソースの表示」画面で F14 (モジュール・リストの処理) を押すと、「モジュール・リストの処理」画面にアクセスできます。サービス・プログラムを除去する場合は、デフォルトのプログラム・タイプを *PGM から *SRVPGM に変更してください。

```
                          モジュール・リストの処理
                                                      システム:   SYSTEM01
オプションを入力して，実行キーを押してください。
  1= プログラム追加    4= プログラム除去    5= モジュール・ソースの表示
  8= モジュール停止点の処理

OPT     プログラム/モジュール     ライブラリー        タイプ
                                  *LIBL             *PGM
 4        WEEKDAY2                MYLIB             *PGM
          WEEKDAY2                                  *MODULE
          DSPWKDAY                MYLIB             *PGM
          DSPWKDAY                                  *MODULE     選択済み
          AABP1                                     *MODULE









                                                                終わり
コマンド
===>
F3=終了    F4=プロンプト    F5=最新表示    F9=コマンドの複写    F12=取り消し
```

図 *10.* デバッグ・セッションからの *ILE* プログラム・オブジェクトの除去： 実行キーを押すと、プログラム WEEKDAY2 がデバッグ・セッションから除去される。

```
                          モジュール・リストの処理
                                                      システム:   SYSTEM01
オプションを入力して，実行キーを押してください。
  1= プログラム追加    4= プログラム除去    5= モジュール・ソースの表示
  8= モジュール停止点の処理

OPT     プログラム/モジュール     ライブラリー        タイプ
                                  *LIBL             *PGM
          DSPWKDAY                MYLIB             *PGM
          DSPWKDAY                                  *MODULE     選択済み
          AABP1                                     *MODULE









                                                                終わり
コマンド
===>
F3=終了    F4=プロンプト    F5=最新表示    F9=コマンドの複写    F12=取り消し
ソース・デバッガーからプログラム WEEKDAY2 が除去された。
```

図 *11.* デバッグ・セッションからの *ILE* プログラム・オブジェクトの除去

プログラム・オブジェクトのデバッグ・セッションからの除去が終了したら、「モジュール・リストの処理」画面で F3 (終了) を押して「モジュール・ソースの表示」画面に戻ってください。

注**:** プログラムをデバッグ・セッションから除去する際には、そのプログラムに対する *CHANGE 権限が必要です。

オリジナル・プログラム・モデル (OPM) プログラムをデバッグ・セッションから除去する場合は、 **プログラム除去 (RMVPGM)** コマンドを使用してください。 OPM ソース・レベルのデバッグが活動状態にある場合、ソース・デバッグ・データを含んで作成した OPM プログラムは「モジュール・リストの処理」画面にリストされます。これらのプログラムをデバッグ・セッションから除去するには、「モジュール・リストの処理」画面のオプション 4 (プログラムの除去) を使用します。

プログラム・ソースの表示

プログラム・オブジェクトのソースを表示するには、「モジュール・ソースの表示」画面を使用します。

「モジュール・ソースの表示」画面には、プログラム・オブジェクトのソースが表示されます。一度に 1 つのモジュール・オブジェクトが表示されます。以下のいずれかのデバッグ・ビュー・オプションを使用してモジュール・オブジェクトをコンパイルした場合は、そのモジュール・オブジェクトのソースを表示できます。

- DBGVIEW(*ALL)
- DBGVIEW(*SOURCE)
- DBGVIEW(*LIST)

「モジュール・ソースの表示」画面の表示内容を変更するには、以下の 2 つの方法があります。

- ビューを変更する。
- モジュールを変更する。

ビューを変更すると、統合言語環境 (ILE) ソース・デバッガーは変更対象のビューでの位置と同じ位置にマップされます。モジュールを変更すると、表示されているビューの実行可能ステートメントがメモリー内に保管され、そのモジュールが再表示される時点で表示されます。ブレークポイントのある行の番号は強調表示されます。ブレークポイント、ステップ、またはメッセージにより、プログラムが停止して画面が表示されると、そのイベントが起こったソース行が強調表示されます。

モジュール・オブジェクトの変更:

「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトを変更するには、「モジュール・リストの表示」画面にある 5 (モジュール・ソースの表示) を使用します。「モジュール・ソースの表示」画面で F14 (モジュール・リストの処理) を押すと、「モジュール・リストの処理」画面にアクセスできます。

次に「モジュール・ソースの表示」画面が記載されています。

モジュール・オブジェクトを選択する場合は、表示したいモジュール・オブジェクトの横に 5 (モジュール・ソースの表示) を入力してください。

```
                          モジュール・ソースの表示

プログラム:   DSPWKDAY   ライブラリー:   MYLIB          モジュール:   DSPWKDAY
    24         500-           CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
    25         600-           IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
    26         700                         VAR(&WEEKDAY) VALUE('日曜日'))
    27         800-           ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGV
    28         900                         VAR(&WEEKDAY) VALUE('月曜日')))
    29        1000-           ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGV
    30        1100                        VAR(&WEEKDAY) VALUE('火曜日')))
    31        1200-           ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGV
    32        1300                        VAR(&WEEKDAY) VALUE('水曜日')))
    33        1400-           ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGV
    34        1500                        VAR(&WEEKDAY) VALUE('木曜日')))
    35        1600-           ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGV
    36        1700                        VAR(&WEEKDAY) VALUE('金曜日')))
    37        1800-           ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGV
    38        1900                        VAR(&WEEKDAY) VALUE('土曜日')))
                                                                    続く...
デバッグ  . .

F3= プログラム終了     F6= 停止点の追加/消去    F10= ステップ    F11= 変数の表示
F12=再開    F17=変数監視    F18=監視の処理       F24=キーの続き
```

図 *12.* モジュール・ビューの表示

表示したいモジュール・オブジェクトを選択してから、実行キーを押してください。選択したモジュール・オブジェクトが「モジュール・ソースの表示」画面に表示されます。

モジュール・オブジェクトを変更する別の方法として、DISPLAY デバッグ・コマンドを使用する方法があります。デバッグ・コマンド行を次のように入力してください。

```
DISPLAY MODULE モジュール名
```

モジュール名のモジュール・オブジェクトが表示されます。 モジュール・オブジェクトは、デバッグ・セッションに追加されているプログラム中か、サービス・プログラムのオブジェクト中に存在していなければなりません。

モジュール・オブジェクト・ビューの変更**:**

「モジュール・ソースの表示」画面のモジュール・オブジェクトのビューを、「ビューの選択」画面を用いて変更します。「モジュール・ソースの表示」画面で F15 (ビューの選択) を押すと、「ビューの選択」画面にアクセスできます。

ILE CL モジュール・オブジェクトを作成する際に指定した値に応じて、ILE CL モジュール・オブジェクトの次のビューを使用できます。

- ルート・ソース・ビュー
- リスト・ビュー
- ステートメント・ビュー

現行のビューはウィンドウの最上部にリストされ、使用できる他のビューはその下に表示されます。作成に使用するデバッグ・オプションに応じて、プログラム・オブジェクト内の各モジュール・オブジェクトは、さまざまなビューのセットを使用できます。

ビューを選択する場合は、表示したいビューの横に 1 (選択) を入力してください。

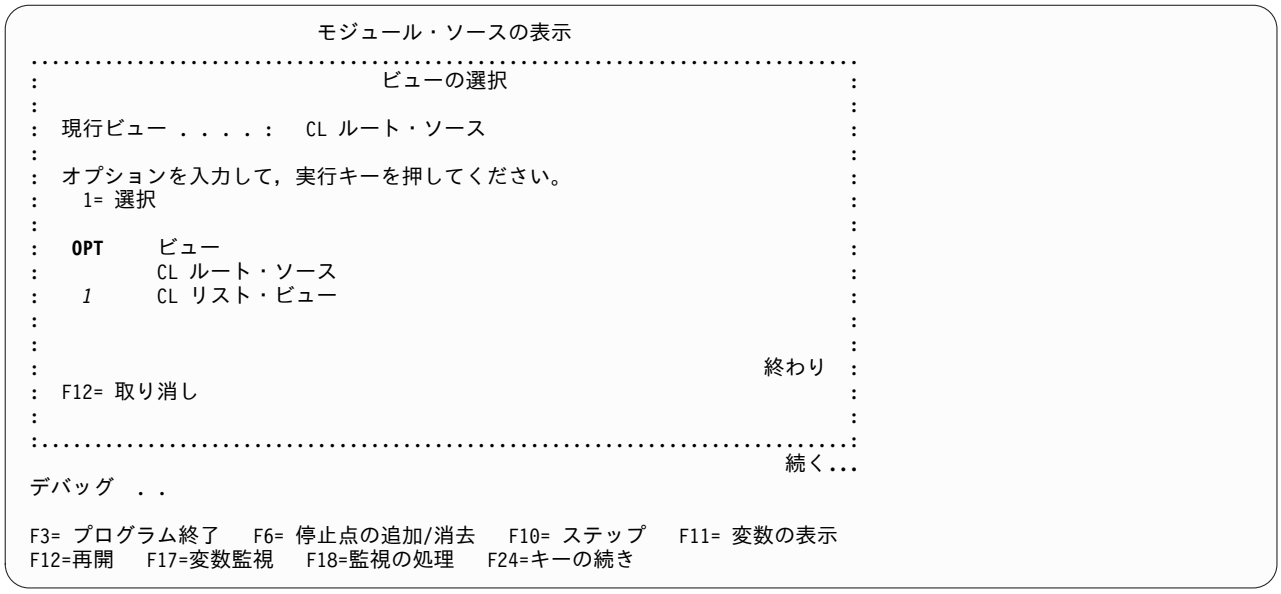


図 *13.* モジュール・オブジェクトのビューの変更

表示したいモジュール・オブジェクトのビューを選択してから、実行キーを押してください。選択したモジュール・オブジェクトのビューが「モジュール・ソースの表示」画面に表示されます。

ブレークポイントの設定および除去

ブレークポイントは、プログラムの実行中に特定の個所でプログラム・オブジェクトを停止します。

無条件ブレークポイントは、プログラム・オブジェクトを特定のステートメントで停止します。条件付きブレークポイントは、特定の条件を満たす特定のステートメントで、プログラム・オブジェクトを停止します。

プログラム・オブジェクトが停止すると、「モジュール・ソースの表示」画面が表示されます。該当するモジュール・オブジェクトが、ブレークポイントがある行に位置するソースとともに表示されます。この行は強調表示されます。この時点で変数の評価、ブレークポイントの追加設定、およびデバッグ・コマンドの実行を行うことができます。

ブレークポイントを使用する際には、ブレークポイントに関する以下の特性を知っている必要があります。

- ブレークポイントが迂回されると (例えば **Goto (GOTO)** ステートメントを使用して)、そのブレークポイントは処理されない。
- ブレークポイントがステートメント上に設定されている場合は、そのステートメントが処理される前にブレークポイントが処理される。
- 条件付きブレークポイントのあるステートメントに達すると、そのステートメントが処理される前に、そのブレークポイントに関連する条件式が評価される。
- ブレークポイント機能は、デバッグ・コマンドを用いて指定する。

  ブレークポイント機能には以下のものがあります。
  - プログラム・オブジェクトにブレークポイントを追加する。
  - プログラム・オブジェクトからブレークポイントを除去する。
  - ブレークポイント情報を表示する。
  - ブレークポイントに達した後で、プログラム・オブジェクトの実行を再開する。

無条件ブレークポイントの設定および除去:

システムでは、複数の方法で無条件ブレークポイントを設定および除去できます。

無条件ブレークポイントの設定と除去を行うには、以下のものを使用します。
- 「モジュール・ソースの表示」画面の F6 (ブレークポイントの追加/消去)
- 「モジュール・ソースの表示」画面の F13 (モジュール・ブレークポイントの処理)
- BREAK デバッグ・コマンド (ブレークポイントを設定する場合)
- CLEAR デバッグ・コマンド (ブレークポイントを除去する場合)

無条件ブレークポイントの設定と除去を行う最も簡単な方法は、「モジュール・ソースの表示」画面で F6 (ブレークポイントの追加/消去) を使用することです。 F6 を使用して無条件ブレークポイントを設定する場合は、ブレークポイントを追加したい行にカーソルを置き、F6 を押してください。無条件ブレークポイントがその行に設定されます。無条件ブレークポイントを除去する場合は、ブレークポイントを除去したい行にカーソルを置き、F6 を押してください。無条件ブレークポイントがその行から除去されます。

設定したい無条件ブレークポイントごとに前記のステップを繰り返してください。

注: ブレークポイントを設定したい行が実行可能ステートメントでない場合、ブレークポイントは次の実行可能ステートメントに設定されます。

ブレークポイントを設定した後で、F3 (終了) を押して「モジュール・ソースの表示」画面を終了してください。「モジュール・ソースの表示」画面で F21 (コマンド行) を使用して、コマンド行からプログラムを呼び出すこともできます。

プログラム・オブジェクトを呼び出してください。ブレークポイントに達すると、プログラムが停止して「モジュール・ソースの表示」画面が再表示されます。この時点で変数の評価、ブレークポイントの追加設定、およびデバッグ・コマンドの実行を行うことができます。

無条件ブレークポイントの設定と除去を行う別の方法として、BREAK および CLEAR デバッグ・コマンドを使用する方法があります。

BREAK デバッグ・コマンドを使用して無条件ブレークポイントを設定する場合は、デバッグ・コマンド行に次のように入力してください。

```
BREAK 行番号
```

行番号は、ブレークポイントを設定するモジュール・オブジェクトの現在表示されているビューの行番号です。

CLEAR デバッグ・コマンドを使用して無条件ブレークポイントを除去する場合は、デバッグ・コマンド行に次のように入力してください。

```
CLEAR 行番号
```

行番号 は、ブレークポイントを除去するモジュール・オブジェクトの現在表示されているビューの行番号です。

ステートメント・ビューを使用している場合、行番号は表示されません。ステートメント・ビューに無条件ブレークポイントを設定する場合は、デバッグ・コマンド行に次のように入力してください。

```
BREAK プロシージャー名/ステートメント番号
```

プロシージャー名は、ユーザーの CL モジュールの名前です。ステートメント番号 (コンパイラー・リストに基づく) は、停止するステートメントの番号です。

条件付きブレークポイントの設定および除去:

「モジュール・ブレークポイントの処理」画面を使用して、ブレークポイントを設定または除去できます。

関連概念:

417 ページの『デバッグ・コマンド』
統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

「モジュール・ブレークポイントの処理」画面の使用:

ブレークポイントを設定または除去するには、「モジュール・ブレークポイントの処理」画面を使用します。

注: 条件付きブレークポイント用にサポートされている関係演算子は、<、>、=、<=、>=、および <> (等しくない) です。

「モジュール・ソースの表示」画面で F13 (モジュール・ブレークポイントの処理) を押すと、「モジュール・ブレークポイントの処理」画面にアクセスできます。以下の図に「モジュール・ブレークポイントの処理」画面が記載されています。条件付きブレークポイントを設定する場合は、以下のように入力して Enter キーを押してください。

- *Opt* フィールドに 1 (追加)
- 行 フィールドに、ブレークポイントを設定したいデバッガー行の番号
- 条件 フィールドに条件式

例えば、デバッガー行 35 に条件付きブレークポイントを設定するには、以下の図に記載されているとおり、以下のように入力して Enter キーを押してください。

- *Opt* フィールドに 1 (追加)
- 行 フィールドに 35
- 条件 フィールドに &I=21

条件付きブレークポイントを除去するには、除去するブレークポイントの横の *Opt* フィールドに 4 (消去) と入力し、Enter キーを押してください。この方法で無条件ブレークポイントも除去できます。

```
                    モジュール停止点の処理
                                                  システム:   SYSTEM01
プログラム . . . :   MYPGM              ライブラリー  . . . :   MYLIB
  モジュール . . :   MYMOD              タイプ  . . . . . :   *PGM

オプションを入力して，実行キーを押してください。
  1= 追加   4= 消去

OPT     行          条件
 1      35____      &I=21____________________
 _      _____       ________________________________
```

図 *14.* 条件付きブレークポイントの設定

設定または除去したい条件付きブレークポイントごとに前記のステップを繰り返してください。

注: ブレークポイントを設定したい行が実行可能ステートメントでない場合、ブレークポイントは次の実行可能ステートメントに設定されます。

設定または除去したいブレークポイントをすべて指定した後で、F3 (終了) を押して「モジュール・ソースの表示」画面に戻ってください。

その後で F3 (終了) を押して「モジュール・ソースの表示」画面を終了してください。「モジュール・ソースの表示」画面で F21 (コマンド入力) を使用して、コマンド行からプログラム・オブジェクトを呼び出すこともできます。

プログラム・オブジェクトを呼び出してください。条件付きブレークポイントのあるステートメントに達すると、そのブレークポイントに関連する条件式が評価されてからそのステートメントが実行されます。結果が偽の場合、プログラム・オブジェクトは実行を継続します。結果が真ならばプログラム・オブジェクトは停止し、「モジュール・ソースの表示」画面が表示されます。この時点で変数の評価、ブレークポイントの追加設定、およびデバッグ・コマンドの実行を行うことができます。

BREAK および **CLEAR** デバッグ・コマンドを使用した、条件付きブレークポイントの設定および削除**:**

条件付きブレークポイントの設定と除去を行う別の方法として、BREAK および CLEAR デバッグ・コマンドを使用する方法があります。

BREAK デバッグ・コマンドを使用して条件付きブレークポイントを設定する場合は、デバッグ・コマンド行に次のように入力してください。

```
BREAK 行番号 WHEN 式
```

行番号は、ブレークポイントを設定するモジュール・オブジェクトの現在表示されているビューの行番号です。式は、ブレークポイントが検出された時点で評価される条件式です。 条件付きブレークポイント用にサポートされている関係演算子は、<、>、=、<=、>=、および <> (等しくない) です。

数値以外の条件付きブレークポイントの式の場合、短い式は暗黙にブランクが埋め込まれてから比較されます。この暗黙埋め込みが行われてから、各国語分類順序 (NLSS) 変換が行われます。

CLEAR デバッグ・コマンドを使用して条件付きブレークポイントを除去する場合は、デバッグ・コマンド行に次のように入力してください。

```
CLEAR 行番号
```

行番号は、ブレークポイントを除去するモジュール・オブジェクトのビュー (現在表示されているもの) の番号です。

ステートメント・ビューの場合、行番号は表示されません。ステートメント・ビューに条件付きブレークポイントを設定する場合は、デバッグ・コマンド行に次のように入力してください。

```
BREAK プロシージャー名/ステートメント名 WHEN 式
```

プロシージャー名は、ユーザーの CL モジュールの名前です。ステートメント番号 (コンパイラー・リストに基づく) は、停止させるステートメントの番号です。

関連概念:

417 ページの『デバッグ・コマンド』
統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

431 ページの『各国語分類順序』
各国語分類順序 (NLSS) は、Char-8 タイプの数値以外の条件付きブレークポイント式だけに適用されます。

各国語分類順序:

各国語分類順序 (NLSS) は、Char-8 タイプの数値以外の条件付きブレークポイント式だけに適用されます。

数値以外の条件付きブレークポイントの式は、以下の 2 つのタイプに分かれます。

- Char- 8: 各文字は 8 ビット。
- Char-16: 各文字は 16 ビット (DBCS)。

表 26に、数値以外の条件付きブレークポイント式の有効な組み合わせを示します。

Char-8 タイプの式のソース・デバッガーで使用される分類順序テーブルは、**制御言語モジュール作成 (CRTCLMOD)** または **バインド制御言語プログラム作成 (CRTBNDCL)** コマンドの SRTSEQ パラメーターに指定されている分類順序テーブルです。

解析される分類順序テーブルが *HEX の場合、分類順序テーブルは使用されません。したがって、ソース・デバッガーは、文字の 16 進数値を使用して分類順序を決定します。それ以外の場合は、指定された分類順序テーブルを使用して各バイトに重みを割り当ててから、比較を行います。シフトアウト/シフトイン文字を含むバイトとその間のバイトには、重みは割り当てられません。

注: 分類順序テーブルの名前は、コンパイル時に保管されます。デバッグ時に、ソース・デバッガーはコンパイルの際に保管された名前を使用して、分類順序テーブルにアクセスします。コンパイル時に指定された分類順序テーブルが *HEX または *JOBRUN 以外のものである場合は、デバッグが開始される前にその分類順序テーブルを変更しないことが重要です。テーブルが損傷しているか削除されているためにアクセスできない場合、ソース・デバッガーは *HEX 分類順序テーブルを使用します。

表 *26.* 数値以外の条件付きブレークポイント式

タイプ	可能性
Char-8	• 文字変数と比較される文字変数 • 文字リテラル[1] と比較される文字変数 • 16 進数リテラル[2] と比較される文字変数 • 文字変数と比較される文字リテラル[1] • 文字リテラル[1] と比較される文字リテラル[1] • 16 進数リテラル[2] と比較される文字リテラル[1] • 文字変数[1] と比較される 16 進数リテラル[2] • 文字リテラル[1] と比較される 16 進数リテラル[2] • 16 進数リテラル[2] と比較される 16 進数リテラル[2]
Char 16	• DBCS 文字変数と比較される DBCS 文字変数 • 図形リテラル[3] と比較される DBCS 文字変数 • 16 進数リテラル[2] と比較される DBCS 文字変数 • DBCS 文字変数と比較される図形リテラル[3] • 図形リテラル[3] と比較される図形リテラル[3] • 16 進数リテラル[2] と比較される図形リテラル[3] • DBCS 文字変数と比較される 16 進数リテラル[2] • 図形リテラル[3] と比較される 16 進数リテラル[2]

表 26. 数値以外の条件付きブレークポイント式 (続き)

タイプ	可能性

注:

1 文字リテラルの形式は 'abc' です。

2 16 進数リテラルは、X'16 進数字' という形式をしています。

3 図形リテラルの形式は、G'<so>DBCS データ<si>' です。シフトアウトは <so> と表され、シフトインは <si> と表されます。

関連資料:

430 ページの『BREAK および CLEAR デバッグ・コマンドを使用した、条件付きブレークポイントの設定および削除』
条件付きブレークポイントの設定と除去を行う別の方法として、BREAK および CLEAR デバッグ・コマンドを使用する方法があります。

例: 条件付きブレークポイント:

以下の例では、条件付きブレークポイントの設定方法を示します。

```
CL 宣言:              DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(1)
                  DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(2)
                  DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1)
                  DCL    VAR(&DEC2) TYPE(*DEC) LEN(4 1)


デバッグ・コマンド:     BREAK 31 WHEN &DEC1 = 48.1

デバッグ・コマンド:     BREAK 31 WHEN &DEC2 > &DEC1

デバッグ・コマンド:     BREAK 31 WHEN &CHAR2 <> 'A'

注:                   比較が行われる前に、1 つのブランクが 'A' の右に
                      暗黙に埋め込まれます。


デバッグ・コマンド:     BREAK 31 WHEN %SUBSTR(&CHAR2 2 1)
<= X'F1'

デバッグ・コマンド:     BREAK 31 WHEN %SUBSTR(&CHAR2 1 1)
>= &CHAR1

デバッグ・コマンド:     BREAK 31 WHEN %SUBSTR(&CHAR2 1 1)
< %SUBSTR(&CHAR2 2 1)
```

%SUBSTR 組み込み関数を使用すると、文字ストリング変数をサブストリングにすることができます。最初の引数はストリングの識別コード、2 番目の引数は開始位置、および 3 番目の引数は 1 バイト文字か 2 バイト文字の数字でなければなりません。 1 つまたは複数のスペースで引数を区切ります。

すべてのブレークポイントを除去する:

「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトを含むプログラム・オブジェクトから、すべてのブレークポイント (条件付きおよび無条件) を除去するには、CLEAR PGM デバッグ・コマンドを使用します。

このデバッグ・コマンドを使用する場合には、デバッグ・コマンド行に次のように入力してください。

```
CLEAR PGM
```

そのプログラムまたはサービス・プログラムにバインドされているすべてのモジュールから、ブレークポイントが除去されます。

命令のステップ処理の使用

ブレークポイントを検出した後で、プログラム・オブジェクトのステートメントを指定数だけ実行してから、そのプログラムを再停止します。

プログラムを停止後、「モジュール・ソースの表示」画面に戻ります。プログラム・オブジェクトは、プログラムが停止したモジュール・オブジェクトで、その次のステートメントから実行し始めます。通常はブレークポイントを使用してプログラム・オブジェクトを停止します。

プログラム・オブジェクトを一度に 1 つのステートメントだけステップスルーする最も簡単な方法は、「モジュール・ソースの表示」画面の F10 (ステップ) または F22 (ステップ・イン) を使用することです。ステップオーバーは、F10 (ステップ) のデフォルト・モードです。F10 (ステップ) または F22 (ステップ・イン) を押すと、「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトの次のステートメントが実行され、プログラム・オブジェクトが再停止します。

注**:** F10 (ステップ) または F22 (ステップ・イン) を使用する際に、ステップスルーするステートメント数を指定することはできません。 F10 (ステップ) または F22 (ステップ・イン) を押すと、ステップは 1 つだけ実行されます。

プログラム・オブジェクトをステップスルーする別の方法として、STEP デバッグ・コマンドを使用する方法があります。 STEP デバッグ・コマンドを使用すると、1 つのステップで複数のステートメントを実行できます。

F10 (ステップ**)** によるプログラム・オブジェクトのステップオーバー、または **F22 (**ステップ・イン**)** によるプログラム・オブジェクトへのステップ・イン**:**

呼び出されるプログラム・オブジェクトをステップオーバーすると、CALL ステートメントと、呼び出されるプログラム・オブジェクトとは 1 つのステップとして実行されます。呼び出されるプログラム・オブジェクトをステップイントゥすると、呼び出されたプログラム・オブジェクト中の各ステートメントが 1 つのステップとして実行されます。

呼び出されたプログラム・オブジェクトの実行を完了した後に、呼び出し側プログラム・オブジェクトは次のステップで停止します。呼び出されたプログラムがデバッグ・データを使用してコンパイルされている場合、呼び出されたプログラム・オブジェクトが「モジュール・ソースの表示」画面に表示され、ユーザーはそのプログラムをデバッグする正当な権限を持ちます。

ステップオーバーはデフォルトのステップ・モードです。次のものを使用すると、プログラム・オブジェクトをステップオーバーできます。

- 「モジュール・ソースの表示」画面の F10 (ステップ)
- STEP OVER デバッグ・コマンド

次のものを使用すると、プログラム・オブジェクトにステップイントゥできます。

- 「モジュール・ソースの表示」画面の F22 (ステップ・イン)
- STEP INTO デバッグ・コマンド

STEP デバッグ・コマンドを使用したプログラム・オブジェクトのステップスルー**:**

STEP デバッグ・コマンドを使用して実行するステートメント数のデフォルト値は 1 です。ただし、ステップ番号を変更できます。

STEP デバッグ・コマンドを使用してプログラム・オブジェクトをステップスルーする場合は、デバッグ・コマンド行に次のように入力してください。

```
STEP ステートメント数
```

ステートメント数は、次のステップで実行したいプログラム・オブジェクトのステートメントの数です。この実行後にそのプログラム・オブジェクトは再停止します。例えば、デバッグ・コマンド行に次のように入力してください。

```
STEP 5
```

ユーザー・プログラム・オブジェクトの次の 5 つのステートメントが実行されてから、そのプログラム・オブジェクトが停止して「モジュール・ソースの表示」画面が表示されます。

別の方法として、STEP OVER デバッグ・コマンドを使用して、デバッグ・セッション中に呼び出されたプログラム・オブジェクトをステップオーバーできます。 STEP OVER デバッグ・コマンドを使用する場合は、デバッグ・コマンド行に次のように入力してください。

```
STEP ステートメント数 OVER
```

実行されるステートメントの中に他のプログラム・オブジェクトに対する CALL ステートメントが含まれている場合は、統合処理言語環境 (ILE) ソース・デバッガーは呼び出されるプログラム・オブジェクトをステップオーバーします。

また、STEP INTO デバッグ・コマンドを使用して、デバッグ・セッション中に呼び出されたプログラム・オブジェクトをステップイントゥできます。 STEP INTO デバッグ・コマンドを使用する場合は、デバッグ・コマンド行に次のように入力してください。

```
STEP ステートメント数 INTO
```

実行されるステートメントの中に他のプログラム・オブジェクトに対する CALL ステートメントが含まれている場合は、デバッガーは呼び出されるプログラム・オブジェクトをステップイントゥします。呼び出されるプログラム・オブジェクト中の各ステートメントがステップとして数えられます。呼び出されたプログラム・オブジェクト中でステップが終了すると、その呼び出されたプログラム・オブジェクトは「モジュール・ソースの表示」画面に表示されます。例えば、デバッグ・コマンド行に次のように入力してください。

```
STEP 5 INTO
```

プログラム・オブジェクトの次の 5 つのステートメントが実行されます。3 番目のステートメントが他のプログラム・オブジェクトに対する CALL ステートメントである場合には、呼び出す側のプログラム・オブジェクトの 2 つのステートメントと、呼び出されるプログラム・オブジェクトの最初から 3 つ目までのステートメントが実行されます。

変数の表示

変数の値を表示するには、「モジュール・ソースの表示」画面または EVAL デバッグ・コマンドを使用します。

「モジュール・ソースの表示」画面を使用して変数を表示するには、表示する変数の上にカーソルを置き、F11 (変数の表示) を押します。「モジュール・ソースの表示」画面の最下部にメッセージ行が表示され、

その行に現行の変数の値が表示されます。

```
                          モジュール・ソースの表示

プログラム:   DSPWKDAY       ライブラリー:   MYLIB          モジュール:   DSPWKDAY
     4               DCL         VAR(&MSGTEXT) TYPE(*CHAR) LEN(20)
     5               CALL        PGM(WEEKDAY2) PARM(&DAYOFWK)
     6               IF          COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
     7                             VAR(&WEEKDAY) VALUE('日曜日'))
     8               ELSE        CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGVAR +
     9                             VAR(&WEEKDAY) VALUE('月曜日')))
    10               ELSE        CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGVAR +
    11                             VAR(&WEEKDAY) VALUE('火曜日')))
    12               ELSE        CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGVAR +
    13                             VAR(&WEEKDAY) VALUE('水曜日')))
    14               ELSE        CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGVAR +
    15                             VAR(&WEEKDAY) VALUE('木曜日')))
    16               ELSE        CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGVAR +
    17                             VAR(&WEEKDAY) VALUE('金曜日')))
    18               ELSE        CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGVAR +
                                                                  続く...
デバッグ  . .

F3= プログラム終了    F6= 停止点の追加/消去       F10= ステップ    F11= 変数の表示
F12= 再開     F17= 変数監視    F18= 監視の処理     F24= キーの続き
&DAYOFWK = 3.
```

図 *15. F11 (*変数の表示*)* の使用による変数の表示

EVAL デバッグ・コマンドを使用して変数の値を判別することもできます。 EVAL デバッグ・コマンドを使用して変数の値を表示する場合は、デバッグ・コマンド行で以下を入力します。

```
EVAL 変数名
```

変数名 は、表示する変数の名前です。EVAL デバッグ・コマンドを「モジュール・ソースの表示」画面で入力し、その値が 1 行で表示できる場合には、変数の値がメッセージ行に表示されます。値を 1 行で表示できない場合は、「評価式」画面に表示されます。

例えば、前の例に示されているモジュール・オブジェクトの行 7 の変数 *&DAYOFWK;* の値を表示するには、以下を入力します。

```
EVAL &DAYOFWK
```

「モジュール・ソースの表示」画面のメッセージ行には、前の例のように `&DAYOFWK = 3.` と表示されます。

EVAL コマンドで使用される変数の有効範囲は、QUAL コマンドを使用して定義します。ただし、CL モジュールに含まれている変数の有効範囲はグローバルなので、それらの変数の有効範囲を特別に定義する必要はありません。

関連概念:

417 ページの『デバッグ・コマンド』
統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

例**:** ロジック変数の表示**:**

この例では、EVAL デバッグ・コマンドを使用してロジック変数の値を表します。

```
CL 宣言:                DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')
```

```
デバッグ・コマンド:     EVAL &LGL1

結果:                   &LGL1 = '1'
```

例: 文字変数の表示:

この例では、EVAL デバッグ・コマンドを使用して文字変数の値を表示しています。

```
CL 宣言:
   DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('EXAMPLE')

デバッグ・コマンド:     EVAL &CHAR1

結果:                   &CHAR1 = 'EXAMPLE   '

デバッグ・コマンド:     EVAL %SUBSTR(&CHAR1 5 3)

結果:                   %SUBSTR(&CHAR1 5 3) = 'PLE'

デバッグ・コマンド:     EVAL %SUBSTR(&CHAR1 7 4)

結果:                   %SUBSTR(&CHAR1 7 4) = 'E   '
```

%SUBSTR 組み込み関数を使用すると、文字ストリング変数をサブストリングにすることができます。最初の引数はストリングの識別コード、2 番目の引数は開始位置、および 3 番目の引数は 1 バイト文字か 2 バイト文字の数字でなければなりません。 1 つまたは複数のスペースで引数を区切ります。

例: **10** 進変数の表示:

この例は、EVAL デバッグ・コマンドを使用して 10 進変数の値を表示します。

```
CL 宣言:
   DCL    VAR(&DEC1) TYPE(*DEC) LEN(4 1) VALUE(73.1)

CL 宣言:
   DCL    VAR(&DEC2) TYPE(*DEC) LEN(3 1) VALUE(12.5)

デバッグ・コマンド:     EVAL &DEC1

結果:                   &DEC1 = 073.1

デバッグ・コマンド:     EVAL &DEC2

結果:                   &DEC2 = 12.5
```

例: 変数を **16** 進値として表示する:

16 進形式で変数値を表示するには、EVAL デバッグ・コマンドを使用します。

変数を 16 進形式で表示するには、デバッグ・コマンド行に次のように入力します

```
EVAL 変数名: x  バイト数
```

変数名は、16 進形式で表示する変数の名前です。'x' は、変数を 16 進形式で表示することを指定しており、バイト数は、表示されるバイト数を示しています。'x' の後に長さが指定されていない場合は、変数のサイズが長さとして使用されます。最小値の 16 バイトが常に表示されます。変数の長さが 16 バイト未満の場合、残りのスペースは 16 バイトの境界に達するまでゼロで埋め込まれます。

```
CL 宣言:              DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('ABC')
                DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10) VALUE('DEF')
```

```
デバッグ・コマンド:    EVAL &CHAR1:X 32

結果:
00000     C1C2C340 40404040 4040C4C5 C6404040 ABC       DEF
00010     40404040 00000000 00000000 00000000     ............
```

変数の値の変更

変数の値を変更するには、EVAL コマンドと代入演算子 (=) を使用します。

EVAL コマンドで使用される変数の有効範囲は、QUAL コマンドを使用して定義します。ただし、CL モジュールに含まれている変数の有効範囲はグローバルなので、それらの変数の有効範囲を特別に定義する必要はありません。

EVAL デバッグ・コマンドを使用して、変数の定義に合うように、指定された変数に数字、文字、および 16 進データを割り当てることができます。

変数の値を変更する場合は、デバッグ・コマンド行に次のように入力してください。

```
EVAL 変数名 = 値
```

変数名 は変更したい変数の名前で、値は変数名に割り当てたい識別コードかリテラル値です。

次の例を参照してください。

```
EVAL &COUNTER = 3.0
```

上記の例は *&COUNTER;* の値を 3.0 に変更し、「モジュール・ソースの表示」画面のメッセージ行に以下を表示します。

```
&COUNTER = 3.0 = 3.0
```

変数名と変更対象の値の後に結果が表示されます。

値を文字変数に割り当てる場合には、以下の規則が適用されます。

- ソース式の長さがターゲット式の長さより短い場合は、ターゲット式ではデータは左そろえされ、残りの桁はブランクで埋め込まれる。
- ソース式の長さがターゲット式の長さより長い場合は、ターゲット式ではデータは左そろえされ、ターゲット式の長さに切り捨てられる。

注: DBCS 変数には次のいずれかを割り当てることができます。

- 別の DBCS 変数
- G'<so>DBCS data<si>' 形式の図形リテラル
- X'16 進数字'形式の 16 進リテラル

関連概念:

417 ページの『デバッグ・コマンド』

統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

例: ロジック変数の変更:

ロジック変数の値を変更するには、EVAL デバッグ・コマンドを使用します。

```
CL 宣言:                DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')
                  DCL    VAR(&LGL2) TYPE(*LGL)


デバッグ・コマンド:     EVAL &LGL1
```

```
結果:                   &LGL1 = '1'

デバッグ・コマンド:     EVAL &LGL1 = X'F0'
結果:                   &LGL1 = X'F0' = '0'

デバッグ・コマンド:     EVAL &LGL2 = &LGL1
結果:                   &LGL2 = &LGL1 = '0'
```

例: 文字変数の変更:

この例では、EVAL デバッグ・コマンドを使用して文字変数の値を変更します。

```
CL 宣言:            DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(1) VALUE
('A')
                    DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10)

デバッグ・コマンド:     EVAL &CHAR1 = 'B'
結果:                   &CHAR1 = 'B' = 'B'

デバッグ・コマンド:     EVAL &CHAR1 = X'F0F1F2F3'
結果:                   &CHAR1 = 'F0F1F2F3' = '0'

デバッグ・コマンド:     EVAL &CHAR2 = 'ABC'
結果:                   &CHAR2 = 'ABC' = 'ABC       '

デバッグ・コマンド:     EVAL %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1)
結果:                   %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1) = 'C '
注:                     変数 &CHAR には 'C C       ' が入っています。
```

%SUBSTR 組み込み関数を使用すると、文字ストリング変数をサブストリングにすることができます。最初の引数はストリングの識別コード、2 番目の引数は開始位置、および 3 番目の引数は 1 バイト文字か 2 バイト文字の数字でなければなりません。 1 つまたは複数のスペースで引数を区切ります。

例: **10** 進変数の変更:

10 進変数の値を変更するには、EVAL デバッグ・コマンドを使用します。

```
CL 宣言:                DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1) VALUE(73.1)
                    DCL    VAR(&DEC2) TYPE(*DEC) LEN(2 1) VALUE(3.1)

デバッグ・コマンド:     EVAL &DEC1 = 12.3
結果:                   &DEC1 = 12.3 = 12.3

デバッグ・コマンド:     EVAL &DEC1 = &DEC2
結果:                   &DEC1 = &DEC2 = 03.1
```

変数の属性の表示

変数の属性とは、変数のサイズ (バイト単位) とタイプです。この属性を表示するには、属性 (ATTR) デバッグ・コマンドを使用します。

ATTR デバッグ・コマンドの使用例を次に示します。

```
CL 宣言:                DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10)

デバッグ・コマンド:     ATTR &CHAR2

結果:                   TYPE = FIXED LENGTH STRING, LENGTH = 10 BYTES


CL 宣言:                DCL    VAR(&DEC) TYPE(*DEC) LEN(3 1)

デバッグ・コマンド:     ATTR &DEC

結果:                   TYPE = PACKED(3,1), LENGTH = 2 BYTES
```

関連概念:

417 ページの『デバッグ・コマンド』

統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

名前を変数、式、またはコマンドと等しくする

名前を変数、式、またはデバッグ・コマンドと簡単に等しくするには、EQUATE デバッグ・コマンドを使用します。

等しくした後でその名前を単独で、または他の式とともに使用できます。この名前を他の式とともに使用すると、名前の値が決定されてから式が評価されます。この名前は、デバッグ・セッションが終了するか名前が除去されるまでアクティブのままです。

名前を変数、式、またはデバッグ・コマンドと等しくするには、デバッグ・コマンド行で以下を入力します。

```
EQUATE 省略名 定義
```

省略名 は変数、式、またはデバッグ・コマンドと等しくしたい名前で、定義 は名前と等しくする対象の変数、式、またはデバッグ・コマンドです。

例えば、*&COUNTER* という名前の変数の内容を表示する *DC* という名前の省略名を定義するには、デバッグ・コマンド行で以下を入力します。

```
EQUATE DC EVAL &COUNTER
```

この操作により、*DC* がデバッグ・コマンド行に入力されるたびに EVAL *&COUNTER* コマンドが実行されます。

EQUATE コマンドに入力できる文字の最大数は 144 です。定義が指定されず、かつ前回の EQUATE コマンドで名前が定義されている場合は、前の定義は除去されます。名前が以前に定義されていない場合は、エラー・メッセージが表示されます。

EQUATE デバッグ・コマンドを使用してデバッグ・セッションに定義されている名前を表示する場合は、デバッグ・コマンド行で以下のように入力します。

```
DISPLAY EQUATE
```

「式の評価」画面にアクティブな名前のリストが表示されます。

関連概念:

417 ページの『デバッグ・コマンド』

統合言語環境 (ILE) ソース・デバッガーとともに、多数のデバッグ・コマンドを使用できます。

ソース・デバッグおよび IBM i グローバリゼーション

ソース・デバッグを処理しているときは、IBM i グローバリゼーションに関する特定の条件に注意してください。

ILE CL には、以下の条件があります。

- 「モジュール・ソースの表示」画面にビューが表示されると、ソース・デバッガーはすべてのデータをデバッグ・ジョブのコード化文字セット識別コード (CCSID) に変換する。
- リテラルを変数に割り当てている場合、ソース・デバッガーは引用符付きリテラル (例えば 'abc') に対して CCSID 変換を実行しない。引用符付きリテラルでは大文字と小文字を区別します。

***SOURCE ビューの処理:**

CL ルート・ソース・ビューの処理を行う場合に限り、この条件が適用されます。ソース・ファイルの CCSID がモジュールの CCSID と異なっている場合は、ソース・デバッガーは可変文字 (#、@、$) を含む CL 識別コードを認識できない。

モジュールの表示 (DSPMOD) CL コマンドを使用して、モジュールの CCSID を調べることができます。CL ルート・ソース・ビューの処理を行う必要があり、かつソース・ファイル CCSID とモジュール CCSID が異なっている場合には、次のいずれかの処置をとることができます。

- CL ソースの CCSID が、コンパイル時ジョブの CCSID と必ず同じになるようにする。
- コンパイル時ジョブの CCSID を 65 535 に変更してコンパイルする。
- 前記の 2 つのオプションが行えない場合は、CL リスト・ビューを使用する。

関連情報:

モジュール表示 (DSPMOD) コマンド

 ILE の概念

一時的にステップを除去する操作

特定の制御言語 (CL) コマンドを使用してライブラリーまたはプログラムを指定する場合、ブレークポイントまたはステップをデバッグ中にプログラムから一時的に除去することができます。

ブレークポイントとステップは、CL コマンドの実行が完了すると復元されます。ブレークポイントやステップが除去された場合には、CPD190A メッセージがジョブ・ログ内に入ります。それらの復元時には、さらに別の CPD190A メッセージがジョブ・ログ内に入ります。

ブレークポイントやステップを一時的に除去できる CL コマンドを以下に示します。

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

注: これらの CL コマンドをプログラムで処理している時に BREAK または STEP コマンドを発行すると、エラー・メッセージ CPF7102 が表示されます。

オリジナル・プログラム・モデル・プログラムのデバッグ

オリジナル・プログラム・モデル (OPM) プログラムをデバッグするには、テスト機能を使用します。これらの機能は、対話方式でもバッチ・ジョブの中でも使用できる 1 組のコマンドを介して実行することができます。

テスト機能とは、ユーザーがアプリケーション・プログラムの作成や保守を行う場合の補助を目的とした機能のことです。このテスト機能を使用して、プログラムを特殊なテスト環境で実行させ、そのテスト環境下でプログラムの処理過程を綿密に観察し、制御できます。ユーザーはテスト機能を使用し、ユーザー・プログラムと対話を行うことができます。これらの機能を使用して、次のタスクを実行できます。

- プログラムの処理順序をトレース (追跡) し、処理されたステートメント、および処理過程の各ポイントにおけるプログラム変数の値を表示する。
- プログラム内の任意のステートメント (ブレークポイント (停止点) と呼ばれる) で処理を停止し、制御権を受け取って、変数の値の表示や変更、あるいは他のユーザー定義プログラムの呼び出しを行う。

テストされるプログラムには、テスト用の特別なコマンドを含める必要はありません。したがって、テストされるプログラムは、何の変更も加えずにそのまま通常の処理に使用することができます。テスト用のコマンドはすべて、テストされるプログラムの永続的な部分ではなくそのプログラムを使用するジョブの中で指定されます。テスト・コマンドを使用すると、高水準言語 (HLL) プログラムを作成する際に用いた用語と同じ用語でプログラムとの記号による対話ができます。変数を名前で参照し、ステートメントを番号で参照します。 (この番号は、プログラムのソース・リストで使用されている番号です。) また、テスト機能はセットアップしているジョブだけに適用できます。同じプログラムを他のジョブで同時に使用しても、テスト機能のセットアップに影響を及ぼすことはありません。

注記**:**

1. 使用頻度の高い並行プログラムをシステムで直接デバッグすると、使用頻度の高い当該プログラムをこのシステムで実行しているジョブのパフォーマンスに影響します。プログラムのコピーを作成して、コピー版をデバッグすることで、パフォーマンスの問題は解決します。

関連タスク:

476 ページの『ライブラリーの使用』
ライブラリーは、関連するオブジェクトをグループ化し、名前によってオブジェクトを探索するために使用するオブジェクトです。したがって、ライブラリーはオブジェクトのグループの登録簿のようなものです。

関連資料:

101 ページの『テストおよびデバッグに使用するパラメーター値』
IBM i オペレーティング・システムには、プログラムの実行時に行われる処理をプログラマーが監視できる機能が組み込まれています。

デバッグ・モードの開始

デバッグ・モードは特殊な環境で、通常のシステム機能に加えてテスト機能を使用することができます。

テストを始めるには、プログラムをデバッグ・モードにしなければなりません。テスト機能は、デバッグ・モードの外部では使用できません。デバッグ・モードを開始するには、**デバッグ開始 (STRDBG)** コマンドを使用しなければなりません。このコマンドは、ユーザーのプログラムをデバッグ・モードに設定するだけでなく、デバッグしようとするプログラムなどのテスト情報を指定するためにも使用します。ユーザーのプログラムは**デバッグ・モード終了 (ENDDBG)** コマンドまたは**プログラム除去 (RMVPGM)** コマンドが出されるか、あるいは現行の経路指定ステップが終了するまでデバッグ・モードの状態を持続します。

注: System i ナビゲーター のシステム・デバッグ・マネージャー機能を使用して、システムを検査するためにデバッグを選択した場合には、 **デバッグ開始 (STRDBG)** コマンドを発行することによってグラフィカル・インターフェースが表示されます。この場合、ユーザー・リストで指定されたユーザーの一人が **デバッグ開始 (STRDBG)** コマンドを発行する際には、コマンド入力画面ではなく System i ナビゲーター のシステム・デバッグ・マネージャーが表示されます。 **デバッグ・モード終了 (ENDDBG)** コマンドが入力されたときに、システム・デバッグ・マネージャーのデバッグが選択されていない場合には、**デバッグ開始 (STRDBG)** コマンドにより、コマンド入力画面を使用してシステム・デバッガーが再び始動されます。

次の **デバッグ開始 (STRDBG)** コマンドはジョブをデバッグ・モードにして、プログラム CUS310 をデバッグの対象として追加します。

```
STRDBG  PGM(CUS310)
```

統合言語環境 (ILE) ソース・デバッガーを使用すると、オリジナル・プログラム・モデル (OPM) プログラムをデバッグすることができます。ソース・デバッグ・データを含む OPM プログラムを作成するには、 **CL プログラム作成 (CRTCLPGM)** コマンドに OPTION(*SRCDBG) または OPTION(*LSTDBG) パラメーターを指定します。 ソース・デバッグ・データは実際にはプログラム・オブジェクトの一部です。

ソース・デバッグ・データを含んで作成された OPM プログラムを ILE ソース・デバッガーに追加するには、STRDBG コマンドでプログラム (PGM) および OPM ソース・レベル・デバッグ (OPMSRC) パラメーターを使用します。 ソース・デバッグ・データを含んで作成された OPM プログラムを使用してデバッグ・セッションを開始するには、次のように入力します。

```
STRDBG PGM(*LIBL/MYOPMPGM) OPMSRC(*YES) DSPMODSRC(*YES)
```

関連タスク:

416 ページの『ILE プログラムのデバッグ』

統合言語環境 (ILE) プログラムをデバッグするには、ILE ソース・デバッガーを使用します。

関連情報:

デバッグ変更 (CHGDBG) コマンド

デバッグ開始 (STRDBG) コマンド

デバッグ終了 (ENDDBG) コマンド

プログラム除去 (RMVPGM) コマンド

CL コマンド検索プログラム

デバッグ・モードへのプログラムの追加:

プログラムをデバッグする前に、そのプログラムをデバッグ・モードにする必要があります。

どんなプログラムもデバッグ・モードで実行することができます。プログラムをデバッグ・モードにするには、**デバッグ開始 (STRDBG)** コマンドの PGM パラメーターにそのプログラムを指定するか、**プログラム追加 (ADDPGM)** コマンドによりそのプログラムをデバッグ・セッションに追加します。 1 つのジョブで、同時にデバッグの対象にするプログラムを最高 20 個まで指定することができます。なお、プログラムをデバッグ・モードに追加するには、変更 (*CHANGE) 権限を持っていなければなりません。

デバッグ・モードに、**デバッグ開始 (STRDBG)** または**プログラム追加 (ADDPGM)** コマンドのどちらか、あるいはその両方を使用して 20 個のプログラムを指定した場合、そのデバッグ・ジョブにさらにプログラムを追加するには、以前に指定したプログラムをいくつか除去しなければなりません。 このような場合には、**プログラム除去 (RMVPGM)** コマンドを使用してください。デバッグ・モードが終了すると、すべてのプログラムが自動的にデバッグ・モードから除去されます。

デバッグ・モードを開始する時点で、ユーザーはどのプログラムをデフォルトのプログラムにするかを指定することができます。デフォルトのプログラム名を指定しておけば、PGM パラメーターを持つデバッグ・コマンドを使用する際にプログラムの名前をそのつど指定する手間が省けます。これは、1 つのプログラムだけをデバッグする場合に役立ちます。例えば、**ブレークポイント追加 (ADDBKP)** コマンドを使用する場合などに、PGM パラメーターにプログラムの名前を指定しなくても、デフォルトのプログラムがブレークポイントを追加するプログラムであると見なされます。デフォルトのプログラムの名前は、デバッグの対象にするプログラムのリスト (PGM パラメーター) で指定しなければなりません。リストに複数のプログラムを指定する場合には、DFTPGM パラメーターにデフォルトのプログラムの名前を指定することができます。 DFTPGM パラメーターの指定がない場合は、STRDBG コマンドの PGM パラメーターのリストで最初に指定したプログラムがデフォルトのプログラムと見なされます。

デフォルトのプログラムはテストの過程で、**デバッグ変更 (CHGDBG)** または**プログラム追加 (ADDPGM)** コマンドを使用して、随時変更することができます。

注**:** デバッグ・モードにあるプログラムが削除、再作成、あるいは保管されて記憶域が解放された場合、そのプログラムを参照しようとすると (**プログラム除去 (RMVPGM)** コマンドの場合を除き) 機能チェックが生じます。この場合は、RMVPGM コマンドを使用してそのプログラムを除去するか、**デバッグ終了 (ENDDBG)** コマンドでデバッグ・モードを終了しなければなりません。そのプログラムに変更を加えて再びデバッグしたい場合には、まずそのプログラムをデバッグ・モードから除去して再作成した後、それをデバッグ・モードに追加する必要があります (**プログラム追加 (ADDPGM)** コマンド)。

関連情報:

デバッグ変更 (CHGDBG) コマンド

プログラム追加 (ADDPGM) コマンド

デバッグ開始 (STRDBG) コマンド

プログラム除去 (RMVPGM) コマンド

CL コマンド検索プログラム

実動ライブラリー内のデータベース・ファイルの更新の防止**:**

デバッグ・モードであっても実動ライブラリーのファイルを使用することができるので、誤ってそれらのファイルを更新してしまわないようにする必要があります。

その際、実動ライブラリーのデータベース・ファイルが誤って変更されてしまうのを防ぐために、**デバッグ開始 (STRDBG)** コマンドで UPDPROD(*NO) を指定するか、またはこのパラメーターのデフォルト値の *NO をそのまま使用します。これによって、更新や新しいレコードの追加の対象としてオープンできるのはテスト・ライブラリーのファイルだけになります。実動ライブラリーのデータベース・ファイルを更新や新しいレコードの追加の対象としてオープンしたい場合、あるいは実動用の物理ファイルからメンバーを削除したい場合には、UPDPROD(*YES) を指定できます。

この機能はライブラリー・リストによっても使用できます。デバッグ・ジョブのライブラリー・リストで、テスト・ライブラリーが実動ライブラリーの前になるようにします。デバッグされるプログラムによって更新したい実動ファイルは、テスト・ライブラリーに複写しておいてください。そうすれば、プログラムの実行時にテスト・ライブラリーのファイルが使用されるため、実動ファイルを意図せずに更新することはなくなります。

呼び出しスタックの表示

呼び出しスタックを表示するには、**デバッグ表示 (DSPDBG)** コマンドを使用します。

呼び出しスタックは、以下のことを示します。

- 現在デバッグされているプログラム
- 呼び出し命令番号、あるいはプログラムの処理が停止する各ブレークポイントの命令番号
- プログラムの反復レベル
- デバッグ・モードにはあるが、まだ呼び出されていないプログラムの名前

プログラムの呼び出しとは、そのプログラムに自動 記憶域を割り振り、マシンの処理をそのプログラムに移すことです。呼び出しスタックには一連の呼び出しがあります。プログラムが処理を終了するか制御権を移動すると、そのプログラムは呼び出しスタックから除去されます。

あるプログラムは、最初の呼び出しがまだ呼び出しスタックにある間に何回も呼び出されることがあります。 1 つのプログラムの各呼び出しは、そのプログラムのそれぞれ 1 つの反復レベルに相当します。

呼び出しが終了 (プログラムが制御権を戻すか、制御権を他へ移した時点) すると、自動記憶域はシステムに返されます。

注:

1. CL プログラムは、再帰指定が可能です。つまり CL プログラムは、直接または呼び出したプログラムを介して間接に CL プログラム自身を呼び出すことができます。
2. 高水準言語によっては再帰的なプログラム呼び出しができないものがあります。一方、再帰的なプログラム呼び出しが可能であるだけでなく、プログラム内部のプロシージャーの再帰指定も行える言語もあります。 (本書では、反復レベル という語はプログラムが呼び出しスタックに呼び出される回数を指しています。プロシージャーの反復レベルについては「プロシージャー反復レベル」と明示します。)
3. CL コマンドのすべてのコマンドおよび画面では、プログラム修飾名反復レベルだけを使用しています。

関連タスク:

277 ページの『プログラムおよびプロシージャー相互間の制御の受け渡しと通信』

プログラム呼び出し (CALL)、**結合プロシージャー呼び出し (CALLPRC)**、および **戻り (RETURN)** コマンドを使用して、プログラムとプロシージャー相互間での制御の受け渡しを行うことができます。

関連情報:

デバッグ表示 (DSPDBG) コマンド

プログラムの活動化:

プログラムの活動化とは、そのプログラムに静的ストレージを割り振ることです。

次のいずれかが生じた時点で活動状態は終了します。

- 現在の経路指定ステップが終了した時点。
- プログラムを活動化した要求が取り消された場合。
- プログラムの最後の (あるいは唯一の) 呼び出しを中止するような**資源再利用 (RCLRSC)** コマンドが実行された場合。

さらに、プログラム呼び出しの時点で行われる処置によって活動状態が無効になることがあります。どのような処理がこれに該当するかは、プログラムが書かれた言語 (HLL または CL) によって異なります。

プログラムが非活動化されると、その静的ストレージはシステムに返されます。プログラムの正常な非活動化の時点は、そのプログラムが書かれた言語 (HLL または CL) によって異なります。 CL プログラムは、そのプログラムが終了した時点で常に非活動化されます。

DFTACTGRP(*YES) でコンパイルされる RPG/400 プログラムまたは ILE RPG プログラムは、プログラムの終了前であっても、最終レコード標識 (LR) がオンに設定された時点で非活動化されます。 戻り操作の指定があり LR がオフの場合には、プログラムは非活動化されません。

関連情報:

資源再利用 (RCLRSC) コマンド

監視されていないメッセージの処理

対話式デバッグ・ジョブでは機能チェックが発生した場合、システムはデフォルトの処理を行い、プログラムを停止せずにユーザーに制御権を渡します。

通常、あるプログラムが監視の対象となっていないエスケープ・メッセージを受け取ると、システムは機能チェック・メッセージ (CPF9999) をそのプログラムのプログラム・メッセージ待ち行列へ送り、プログラムは処理を停止します。ただし、高水準言語 (HLL) プログラム・コンパイラーによっては、機能チェック・メッセージまたはプログラム内で発生する可能性のあるメッセージを監視する機能を挿入することがあります。 (照会メッセージがプログラム・メッセージ表示画面に送られます。) これによって、ユーザーはプログラムを自由に終了させることができます。対話式デバッグ・ジョブでは機能チェックが発生した場合、システムは、監視されていないメッセージのディスプレイに以下を表示します。

- そのメッセージ
- メッセージを出した MI 命令番号および HLL ステートメント ID (可能な場合)
- メッセージが送られたプログラムの名前およびその反復レベル

非監視メッセージ・ブレークポイント表示画面の例を次に示します。

```
                非監視メッセージ停止点の表示

ステートメント/命令  . . . . . :   440 /0077
プログラム . . . . . . . . . . . :   TETEST
反復レベル . . . . . . . . . . . :   1

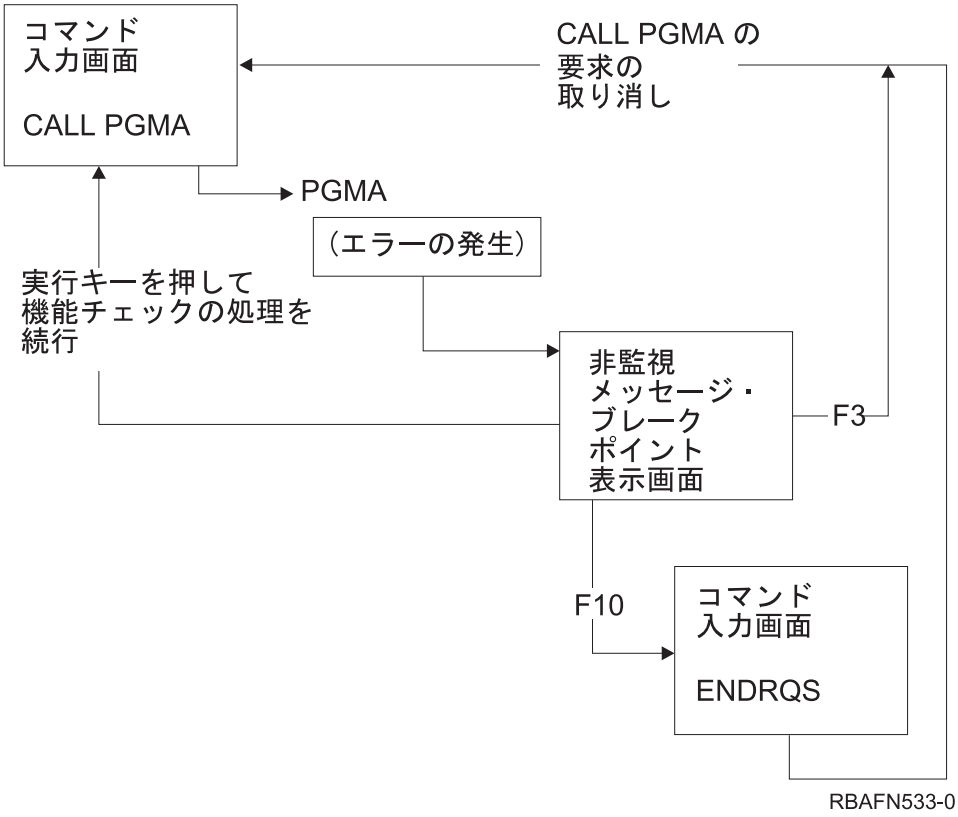
コマンド・エラーが発生した。

続行するためには，実行キーを押してください。

F3= プログラム終了    F10= コマンド入力
```

テスト機能を使用することによって、ユーザーはエラーの原因を究明することができます。ただし、エラーの生じた当初の要求は、エラーの発生点で停止したままになっています。エラーの生じた要求を呼び出しスタックから除去するには、非監視メッセージ・ブレークポイント表示画面が表示されている時点で**要求終了 (ENDRQS)** コマンドを使用するか、F3 キーを押します。非監視メッセージ・ブレークポイント表示画面が表示されている時点で実行キーを押すと、通常の機能チェック処理を続行させることができます。 F10 キーを押してコマンド入力画面を呼び出した場合、非監視メッセージ・ブレークポイント表示画面に戻るには F3 キーを押さなければなりません。

以下の図は、**ENDRQS** コマンドの処理方法を示しています。



プログラム呼び出しは、**ENDRQS** コマンドが入力されると打ち切られます。 (この図では、PGMA のプログラム呼び出しが打ち切られます。)

関連情報:

要求終了 (ENDRQS) コマンド

ブレークポイント

ブレークポイントとはプログラム内の特定の位置のことで、システムはその位置でプログラムの実行を停止し、ディスプレイ装置のユーザー (対話モードの場合) または**ブレークポイント追加 (ADDBKP)** コマンドの BKPPGM パラメーターで指定されたプログラムに制御権を渡します (バッチ・モードの場合)。

関連情報:

ブレークポイント追加 (ADDBKP) コマンド

プログラムへのブレークポイントの追加:

デバッグするプログラムにブレークポイントを追加するには、**ブレークポイント追加 (ADDBKP)** コマンドを使用します。

ユーザーは、1 つの **ADDBKP** コマンドで最高 10 個までのステートメント ID を指定することができます。ある **ADDBKP** コマンドで指定したプログラム変数は、そのコマンドで指定したブレークポイントにのみ適用されます。 変数は、1 つの **ADDBKP** コマンドで最高 10 個まで指定できます。

また、ブレークポイントを追加したいプログラムの名前を指定することもできます。 ブレークポイントを追加したいプログラムの名前を指定しなかった場合には、 STRDBG、CHGDBG、または ADDPGM コマンドで指定されたデフォルトのプログラムにブレークポイントが追加されます。

プログラムにブレークポイントを追加するには、ステートメント ID を指定します。ステートメント ID として使用できるのは以下のとおりです。

- ステートメント・ラベル
- ステートメント番号
- マシン・インターフェース (MI) 命令番号

プログラムにブレークポイントを追加する場合にプログラム変数も同時に指定しておけば、ブレークポイントに到達した時点で指定した変数の値または値の一部を表示させることができます。これらの変数は文字形式、あるいは 16 進数形式で表示することができます。

プログラムは、ブレークポイントに対応する命令が処理される前に実行を停止します。対話式ジョブの場合、システムはプログラムがどのブレークポイントで停止したか、および指定に従ってその停止時点でのプログラム変数の値を表示します。

高水準言語プログラムでは、同一の内部命令に、異なるステートメントやラベルがマッピングされる (対応づけられる) ことがあります。 これは、プログラムにいくつかの非動作ステートメント (DO と ENDDO など) が連続している場合に起こります。どのステートメント、あるいはラベルが同一の命令にマッピングされているかを判別するには、IRP リストを使用します。

同一の命令にいくつかの異なるステートメントがマッピングされた結果、新しいブレークポイントを追加することによって、以前に別のステートメントで追加されたブレークポイントが定義し直される場合があります。そのような場合、以前のブレークポイントは新しいブレークポイントで置き換えられます。つまり以前のブレークポイントは除去され、新しいブレークポイントが追加されます。ユーザーはこのような情報が表示された場合、次のいずれかを行うことができます。

- F3 キーを押すことによって、最新の要求を終了させる。
- 実行キーを押すことによって、プログラムを続行する。
- F10 キーを押すことによって、次の要求レベルのコマンド入力画面に進む。この画面から次のことを行うことができます。
  - 対話式デバッグ環境で使用可能な CL コマンドを入力する。ユーザー・プログラムの変数の値の表示または変更、デバッグ・モードへのプログラムの追加またはデバッグ・モードからの除去、あるいは他のデバッグ・コマンドの実行を行うことができます。
  - **ブレークポイント再開始 (RSMBKP)** コマンドを入力することによって、プログラムの実行を再開する。
  - F3 キーを押して、ブレークポイント表示画面へ戻る。
  - **要求終了 (ENDRQS)** コマンドを入力することによって、前の要求レベルのコマンド入力画面へ戻る。

バッチ・ジョブの場合には、ブレークポイントに到達した時点でブレークポイント・プログラムを呼び出すことができます。ブレークポイント情報を処理するには、このようなブレークポイント・プログラムを作成しておかなければなりません。ブレークポイント情報はブレークポイント・プログラムに渡されます。ブレークポイント・プログラムは CL プログラムなど別のプログラムで、対話式ジョブの場合に対話方式で入力するのと同じコマンド (機能の要求) を使用することができます。例えば、このプログラムは変数の表示や変更、あるいはブレークポイントの追加や除去を行うことができます。また、バッチ・ジョブで有効なものであれば、どのような機能でも要求することができます。ブレークポイント・プログラムが処理を完了すると、デバッグされているプログラムの実行が開始されます。

デバッグ・ジョブの各ブレークポイントごとに、メッセージがジョブ・ログに記録されます。

次の **ADDBKP** コマンドによって、プログラム CUS310 にブレークポイントが追加されます。プログラム CUS310 はデフォルトのプログラムなので、プログラム名を指定する必要はありません。 2 番目のブレークポイントに到達すると、変数 &ARBAL の値が表示されます。

```
ADDBKP  STMT(900)
ADDBKP  STMT(2200)  PGMVAR('&ARBAL')
```

注: CL 変数は、単一引用符で囲む必要があります。

CUS310 のソースは以下のとおりです。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
5761WDS V7R1M0  100416                SEU SOURCE LISTING

ソース・ファイル. . . . . .  QGPL/QCLSRC
メンバー  . . . . . .  CUS310

SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+...
6 ...+... 7 ...+... 8 ...
  100 PGM   PARM(&NBRITEMS &ITEMPRC &PARBAL &PTOTBAL)
  200   DCL VAR(&PARBAL)   TYPE(*DEC) LEN(15 5) /*
INPUT AREA INV BALANCE */
  300   DCL VAR(&PTOTBAL)  TYPE(*DEC) LEN(15 5) /*
INPUT TOTAL INV BALANCE*/
  400   DCL VAR(&NBRITEMS) TYPE(*DEC) LEN(15 5) /*
NUMBER OF ITEMS        */
  500   DCL VAR(&ITEMPRC)  TYPE(*DEC) LEN(15 5) /*
PRICE OF THE ITEM      */
  600   DCL VAR(&ARBAL)    TYPE(*DEC) LEN(5 2)  /*
AREA INVENTORY BALANCE */
  700   DCL VAR(&TOTBAL)   TYPE(*DEC) LEN(5 2)  /*
TOTAL INVENTORY BALANCE*/
  800   DCL VAR(&TOTITEM)  TYPE(*DEC) LEN(5 2)  /*
TOTAL PRICE OF ITEMS   */
  900   CHGVAR     VAR(&ARBAL) VALUE(&PARBAL)
 1000   CHGVAR     VAR(&TOTBAL) VALUE(&PTOTBAL)
 1100   IF COND(&NBRITEMS *EQ 0) THEN(DO)
 1200      SNDPGMMSG  MSG('The number of items is zero.  This item +
 1300                   should be ordered.') TOMSGQ(INVLIB/INVQUEUE)
 1400      GOTO       CMDLBL(EXIT)
 1500   ENDDO
 1600   CHGVAR     VAR(&TOTITEM) VALUE(&NBRITEMS * &ITEMPRC)
 1700   IF COND(&NBRITEMS *GT 50) THEN(DO)
 1800      SNDPGMMSG  MSG('Too much inventory for this item.') +
 1900                 TOMSGQ(INVLIB/INVQUEUE)
 2000   ENDDO
 2100   CHGVAR     VAR(&ARBAL)   VALUE(&ARBAL + &TOTITEM)
 2200   IF COND(&ARBAL *GT 1000) THEN(DO)
 2300      SNDPGMMSG  MSG('The area has too much money in +
 2400                   inventory.') TOMSGQ(INVLIB/INVQUEUE)
 2500   ENDDO
 2600   CHGVAR     VAR(&TOTBAL)  VALUE(&TOTBAL + &TOTITEM)
 2700   EXIT:      ENDPGM
```

以下の図は、最初のブレークポイントに到達した結果として表示される画面です。

```
                          停止点の表示

ステートメント/命令  . . . . . :   900 /0009
プログラム . . . . . . . . . . . :   CUS310
反復レベル . . . . . . . . . . . . . :   1

続行するためには，実行キーを押してください。

F3= プログラム終了    F10= コマンド入力
```

次の図は、2 番目のブレークポイントに到達した結果として表示される画面です。

```
                          停止点の表示

ステートメント/命令. . . . . :   2200 /0022
プログラム . . . . . . . . . . . :   CUS310
反復レベル . . . . . . . . . . . . . :   1
開始桁 . . . . . . . . . . . . . . . :   1
形式 . . . . . . . . . . . . . . . . :   *CHAR
長さ   . . . . . . . . . . . . . . . :   *DCL

変数 . . . . . . . . . . . . . . . . :   &ARBAL
  タイプ . . . . . . . . . . . . . . :     PACKED
  長さ   . . . . . . . . . . . . . . :     5 2
 '610.00'

続行するためには，実行キーを押してください。

F3= プログラム終了    F10= コマンド入力
```

変数 &ARBAL が表示されています。 (&ARBAL の値は、プログラムに渡されたパラメーター値に応じて異なる点に注意してください。) ここで F10 キーを押してコマンド入力画面を表示して変数 &ARBAL の値を変更することにより、その後のプログラムの処理を変更することができます。変数の値を変更するには、**プログラム変数変更 (CHGPGMVAR)** コマンドを使用します。

関連情報:

ブレークポイント再開 (RSMBKP) コマンド

ブレークポイント追加 (ADDBKP) コマンド

CL コマンド検索プログラム

条件付きブレークポイントの追加:

デバッグ中のプログラムに条件付ブレークポイントを追加するには、**ブレークポイントの追加 (ADDBKP)** コマンドを使用してステートメントおよび条件を指定します。

システムは指定の条件が満たされると、指定されたステートメントの位置でプログラムの処理を停止します。

ADDBKP コマンドにスキップ値を指定することができます。スキップ値とは、システムがプログラムを停止させるまでに指定のステートメントを何回処理しなければならないかを示す値です。例えば、ステートメント 1200 を 100 回処理した時点でそのステートメントの位置でプログラムを停止させたい場合、次のコマンドを入力します。

```
ADDBKP STMT(1200) SKIP(100)
```

複数のステートメントの指定がある場合に SKIP パラメーターを指定すると、各ステートメントの処理回数が個々にカウントされます。次に示すコマンドを実行すると、ユーザー・プログラムはステートメント 150 またはステートメント 200 で停止しますが、停止するのはそのステートメントが 400 回処理された場合だけです。

```
ADDBKP STMT(150 200) SKIP(400)
```

ステートメント 150 が 400 回処理され、ステートメント 200 が 300 回だけ処理されている場合、プログラムはステートメント 200 の位置では停止しません。

SKIP パラメーターで指定された回数だけステートメントが処理されていない場合、**ブレークポイント表示 (DSPBKP)** コマンドを使用してそのステートメントの処理回数を表示させることができます。あるステートメントに関する処理回数をゼロにリセットしたい場合には、そのステートメントに対してブレークポイントを入力し直してください。

ADDBKP コマンドに、より一般的なブレークポイント条件を指定することができます。この条件式には、オペランドとしてプログラム変数、演算子、およびもう 1 つの他の変数または定数を指定します。例えば、変数 &X が 1000 を超えた場合にステートメント 1500 の位置でプログラムを停止させたい場合、次のコマンドを入力してください。

```
ADDBKP STMT(1500) PGMVAR('&X') BKPCOND(*PGMVAR1 *GT 1000)
```

BKPCOND パラメーターには 3 つの値が必要です。

- この例において、最初の値には PGMVAR パラメーターで指定した最初の変数を指定しています。 (3 番目の変数を指定する場合は、*PGMVAR3 を指定します。)
- 2 番目の値は、演算子でなければなりません。
- 3 番目の値には、定数または他の変数を指定します。定数としては、数値、文字ストリング、またはビット・ストリングを指定できますが、どの場合にも最初の値として指定したプログラム変数と同じタイプでなければなりません。

ブレークポイントの複合条件を指定するために、SKIP パラメーターと BKPCOND パラメーターを同時に使用することができます。例えば、ステートメント 1000 が 50 回処理され、さらに文字ストリング &STR の値が TRUE の場合にだけステートメント 1000 でプログラムを停止させたい場合、次のコマンドを入力してください。

```
ADDBKP STMT(1000) PGMVAR('&STR') SKIP(50)
       BKPCOND(*PGMVAR1 *EQ 'TRUE ')
```

関連資料:

113 ページの『式の中の演算子』
式の中で使用する演算子は、その式の中のオペランドに対して行う演算、または、オペランドとオペランドとの関係を指定します。

関連情報:

ブレークポイント追加 (ADDBKP) コマンド

ブレークポイント表示 (DSPBKP) コマンド

プログラムからブレークポイントを除去する:

プログラムからブレークポイントを除去するには、**ブレークポイント除去 (RMVBKP)** コマンドを使用します。

ブレークポイントを除去するには、ブレークポイントとして定義されているステートメントのステートメント番号を指定しなければなりません。

関連情報:

ブレークポイント除去 (RMVBKP) コマンド

トレース

トレース (追跡) とは、プログラム中のステートメントが処理された順序を記録する処理のことです。

トレースはブレークポイントと異なり、トレースの過程でユーザーに制御権を与えることはありません。システムは、処理されたトレースの対象となっているステートメントを記録します。しかし、プログラムの処理が完了しても、トレース情報は自動的には表示されません。 **トレース・データ表示 (DSPTRCDTA)** コマンドを使用して、トレース情報の表示を要求する必要があります。この要求によって、対象のステートメントが処理された順序、および該当する場合は、**トレース追加 (ADDTRC)** コマンドで指定された変数の値が表示されます。

関連情報:

トレース追加 (ADDTRC) コマンド

トレース・データ表示 (DSPTRCDTA) コマンド

トレースのプログラムへの追加:

トレースを追加するには、トレースの対象となるステートメントを指定し、また同時にプログラム変数の名前も必要に応じて指定します。

指定した変数の値は、トレースされるステートメントの実行前に記録されます。また、トレースの対象となっているステートメントが最後に実行された時点以降、変数の値が変化した場合にだけその値を記録するように指定することもできます。これらの変数は文字形式、あるいは 16 進数形式で表示することができます。

トレースの対象とするステートメントを指定するには、次のような方法があります。

- トレースを開始するステートメント ID コードと終了するステートメント ID を指定する。
- プログラム中のすべてのステートメントのトレースを指定する。
- トレースの対象とする個々のステートメントのステートメント ID を指定する。

デバッグ開始 (STRDBG) または**デバッグ変更 (CHGDBG)** コマンドでは、そのジョブでいくつのステートメント・トレース情報を記録できるか、およびその最大数に到達した場合にシステムにどのような処置をとらせるかを指定することができます。 最大数に達した場合、システムは (ユーザーの指定に応じて) 次のいずれかの処置をとります。

- 対話式ジョブの場合には、以下のどちらかを行うことができます。
  - トレースを停止する (*STOPTRC)。制御権はユーザーに渡されます (ブレークポイントが発生します)。トレース定義をいくつか除去するか (**トレース除去 (RMVTRC)** コマンド)、トレース・データを消去するか (**トレース・データの消去 (CLRTRCDTA)** コマンド)、あるいは最大数を変更することができます (**デバッグ変更 (CHGDBG)** コマンドの MAXTRC パラメーター)。
  - トレースを続行する (*WRAP)。以前に記録されたトレース・データは、この時点以後に記録されるトレース・データによって書き換えられていきます。
- バッチ・ジョブの場合には、次のどちらかを行うことができます。
  - トレースを停止する (*STOPTRC)。トレースの定義は除去され、プログラムは処理を続けます。
  - トレースを続行する (*WRAP)。以前に記録されたトレース・データは、この時点以後に記録されるトレース・データによって書き換えられていきます。

デバッグ・ジョブの実行中に、**デバッグ変更 (CHGDBG)** コマンドを使用することにより、任意の時点で最大数とデフォルトの処置を変更することができます。ただし、既に記録されたトレースには何の影響も与えません。

どのような時点でも 1 つのプログラムに指定できるステートメントの範囲は、合計 5 つまでです。これは、そのプログラムに対するすべての**トレース追加 (ADDTRC)** コマンドを通じての合計です。また指定できる変数の数は、1 ステートメント範囲につき 10 個までです。

高水準言語プログラムでは、同一の内部命令に、異なるステートメントやラベルがマッピングされる (対応づけられる) ことがあります。 これは、プログラム中にいくつかの非動作ステートメント (DO と END DO など) が連続している場合に起こります。どのステートメント、あるいはラベルが同一の命令にマッピングされているかを判別するには、IRP リストを使用します。

CL 変数を指定する場合は、& と変数名を単一引用符で囲む必要があります。以下にその例を示します。

```
ADDTRC PGMVAR('&IN01')
```

ステートメントの範囲を指定する場合、通常はトレースを終了するステートメントのソース・ステートメント番号の方が、トレースを開始するステートメントの番号より大きくなります。しかし、トレースはマシン・インターフェース (MI) 命令をもとに実行され、一方、コンパイラー (特に RPG/400) によっては MI 命令の順序がソース・ステートメントの順序と異なるプログラムが生成される場合があります。このため場合によっては、終了のステートメントの MI 番号が開始のステートメントの MI 番号より小さくなり、ユーザーにメッセージ CPF1982 が出されることがあります。

このメッセージを受け取った場合には、次のいずれかを行ってください。

- プログラムのすべてのステートメントをトレースする。
- ステートメントの範囲を 1 ステートメントに制限する。
- そのプログラムのプログラムの中間表現 (IRP) リストから得られる MI 命令番号を使用する。

次の**トレース追加 (ADDTRC)** コマンドは、プログラム CUS310 にトレースを追加しています。プログラム CUS310 はデフォルトのプログラムなので、プログラム名を指定する必要はありません。 変数 &TOTBAL の値が記録されるのは、トレースの対象となるステートメントが処理されてから次に処理されるまでの間に、その値が変化した場合だけです。

```
ADDTRC STMT((900 2700))  PGMVAR('&TOTBAL')  OUTVAR(*CHG)
```

次の画面はこのトレースの結果を示しています。画面の表示は**トレース・データ表示 (DSPTRCDTA)** コマンドを使用して要求します。欄見出しが必ずしもすべての画面に表示されていないことに注意してください。

```
                          トレース・データの表示

                 ステートメント
プログラム       命令                  反復レベル
順序番号
CUS310           900                       1
          1



 開始位置 . . .. . . . . . . . . . . . :   1
 長さ  . . . . . . . . . . . . . . . . :   *DCL
 形式  . . . . . . . . . . . . . . . . :   *CHAR

 変数  . . . . . . . . . . . . . . . . :   &TOTBAL
   タイプ. . . . . . . . . . . . . . . :     PACKED
   長さ  . . . . . . . . . . . . . . . :     5 2
  '    .00'

                 ステートメント/
プログラム       命令                  反復レベル
順序番号
CUS310           1000                      1
          2
CUS310           1100                      1
          3  +



続行するためには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

```
                          トレース・データの表示


 開始位置  . . . . . . . . . . . . . . :   1
 長さ  . . . . . . . . . . . . . . . . :   *DCL
 形式  . . . . . . . . . . . . . . . . :   *CHAR

*変数  . . . . . . . . . . . . . . . . :   &TOTBAL
   タイプ. . . . . . . . . . . . . . . :     PACKED
   長さ  . . . . . . . . . . . . . . . :     5 2
  '   1.00'

                 ステートメント/
プログラム       命令                  反復レベル
順序番号
CUS310           1600                      1
           4
CUS310           1700                      1
           5

CUS310           2200                      1
           7









続行するためには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

```
                          トレース・データの表示

CUS310          2700                          1
           9


 開始位置  . . . . . . . . . . . . . . . :   1
 長さ  . . . . . . . . . . . . . . . . . :   *DCL
 形式  . . . . . . . . . . . . . . . . . :   *CHAR

*変数  . . . . . . . . . . . . . . . . . :   &TOTBAL
   タイプ. . . . . . . . . . . . . . . . :     PACKED
   長さ  . . . . . . . . . . . . . . . . :     5 2
  '   2.00'









続行するためには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

関連タスク:

461 ページの『マシン・インターフェース・レベルでのデバッグ』
ユーザー・プログラムをマシン・インターフェース (MI) レベルでデバッグするには、コマンドの PGMVAR パラメーターに MI オブジェクト定義ベクトル (ODV) 番号を指定し、STMT パラメーターに MI 命令番号を指定します。

関連情報:

デバッグ変更 (CHGDBG) コマンド

デバッグ開始 (STRDBG) コマンド

CL コマンド検索プログラム

命令のステップ処理の使用:

プログラムを 1 命令ごとに実行させる (ステップスルー) には、**デバッグ開始 (STRDBG)** コマンドまたは **デバッグ変更 (CHGDBG)** コマンドを使用します。

これらのコマンドを使用する場合は、MAXTRC パラメーターを 1 に設定し、TRCFULL パラメーターを *STOPTRC に設定します。トレースの範囲を**トレース追加 (ADDTRC)** コマンドで指定し、プログラムがその範囲内で 1 つの命令を実行すると、ブレークポイント表示画面とエラー・メッセージが表示されます。ここで実行キーを押すと、トレース範囲の次の命令が処理され、別のブレークポイント表示画面と、前と同じエラー・メッセージが表示されます。トレースが完了した時点で、トレース・データにはトレースされた命令のリストが入っています。このデータは、**トレース・データ表示 (DSPTRCDTA)** コマンドを入力することにより表示できます。

トレース内でのブレークポイントの使用:

トレース範囲の内部でブレークポイントを使用することができます。

トレース範囲内のブレークポイントでは、そのトレース・データを表示して (**トレース・データ表示 (DSPTRCDTA)** コマンド)、何らかの処置をとる必要があるかどうかを判断することができます。トレース・データは、ブレークポイントでプログラムが停止する前に記録されます。トレース情報には、そのステートメントが実行される前のすべての変数の値が含まれています。

システムからのトレース情報の除去:

トレース情報の表示後にその情報をシステムから除去するか、それともシステムに残しておくかを指定するには、**トレース・データ表示 (DSPTRCDTA)** コマンドを使用します。

トレース情報をシステムに残す場合は、他のトレース情報がそれに追加されていきます。この情報は (除去されない限り) デバッグ・ジョブが終了するか、または ENDDBG コマンドが実行要求されるまでシステム内に保持されます。ユーザーはまた、**トレース・データの消去 (CLRTRCDTA)** コマンドを使用して、システムからトレース情報を除去することもできます。

関連情報:

トレース・データ表示 (DSPTRCDTA) コマンド

トレース・データ消去 (CLRTRCDTA) コマンド

プログラムからトレースを除去する:

トレース除去 (RMVTRC) コマンドは、1 つまたは複数の**トレース追加 (ADDTRC)** コマンドで指定されたすべてまたは一部のトレース範囲を除去します。

トレース範囲を除去するには、**トレース除去 (RMVTRC)** コマンドでステートメント識別コードの範囲を指定するか、またはトレース範囲すべての一括除去を指定します。

トレース除去 (RMVTRC) コマンドで STMT パラメーターを使用すれば、以下のものを指定することができます。

- **トレース追加 (ADDTRC)** コマンドでどのようなトレースの定義が行われていたかに関係なく、指定のプログラムのすべての高水準言語 (HLL) ステートメントまたは機械語命令 (あるいはその両方) をトレースの対象にしない。
- 除去するトレース範囲の開始位置と終了位置を示す HLL ステートメントまたは機械語命令 (あるいはその両方) の ID。

プログラム除去 (RMVPGM) コマンドと **デバッグ終了 (ENDDBG)** コマンドも、トレースを除去するために使用することができますが、この 2 つのコマンドはプログラムをデバッグ・モードからも除去します。

関連情報:

トレース追加 (ADDTRC) コマンド

トレース除去 (RMVTRC) コマンド

デバッグ・モード終了 (ENDDBG) コマンド

プログラム除去 (RMVPGM) コマンド

テスト情報の表示

テスト情報を表示すると、デバッグ・モードでジョブを見直す際に役立ちます。

どのようなプログラムがデバッグ・モードにあるか、およびそれらのプログラムにどのようなブレークポイントとトレースが定義されているかを表示することができます。さらに、デバッグ・モードのプログラムの状況を表示することもできます。

テスト情報の表示には、次のコマンドを使用します。

- **デバッグ表示 (DSPDBG)**。現在の呼び出しスタック、およびデバッグ・モードにあるプログラムの名前を表示し、さらに次のような情報も表示します。
  - どのプログラムがブレークポイントで停止しているか

  - どのプログラムが現在呼び出されているか
  - 呼び出されているプログラムの要求レベル
  - そのデバッグ・ジョブで選択されているデバッグ・オプション
- **ブレークポイント表示 (DSPBKP)** コマンド。このコマンドは、プログラムに現在定義されているブレークポイントの位置を表示します。
- **トレース表示 (DSPTRC)** コマンド。このコマンドは、プログラムに現在定義されているステートメントまたはステートメント範囲を表示します。

関連情報:

デバッグ表示 (DSPDBG) コマンド

トレース表示 (DSPTRC) コマンド

ブレークポイント表示 (DSPBKP) コマンド

変数の値の表示

変数名を **ブレークポイント追加 (ADDBKP)** コマンドに指定すると、プログラム変数の値が自動的にブレークポイント表示画面に表示されます。また、F10 を押して、コマンド入力画面を表示し、ブレークポイントで **プログラム変数表示 (DSPPGMVAR)** コマンドを入力することもできます。

1 つの **プログラム変数表示 (DSPPGMVAR)** コマンドに指定できる変数の数は 10 個だけです。文字変数およびビット変数に関しては、ユーザーはその変数の値の表示を開始する位置と表示する長さとを、システムに指定することができます。変数は、文字形式または 16 進数形式で表示することができます。

注:

1. 配列変数を指定した場合には、次のいずれかを行うことができます。
   a. 表示したい配列要素の添え字の値を指定する。この添え字の値は、整数の値またはプログラム内の数値変数の名前のどちらでも構いません。
   b. 添え字を入力せず、配列全体を表示する。
   c. 1 つの次元を除くすべての次元の添え字を指定し、その除外した次元の添え字として 1 つのアスタリスクを指定する。これによって、配列の 1 つの次元のクロスセクションを表示することができます。
2. 変数の名前は単純名か修飾名のどちらで指定しても構いませんが、必ず単一引用符で前後を囲みます。修飾名は、次の 2 つのどちらかの方法で指定することができます。
   a. OF または IN という特殊区切り語と交互に現れる変数名を、修飾レベルの低いものから高いものへという順序で並べる。この場合、変数の名前と特殊区切り語との間にブランクを 1 つ入れて区切らなければなりません。
   b. 変数名相互間をピリオドで区切り、修飾レベルの低いものから高いものへという順序で並べる。

次の **プログラム変数表示 (DSPPGMVAR)** コマンドは、プログラム CUS310 で使用されている変数 ARBAL を表示します。プログラム CUS310 はデフォルトのプログラムなので、プログラム名を指定する必要はありません。 値の全体が文字形式で表示されます。

```
DSPPGMVAR  PGMVAR('&ARBAL')
```

この結果、次のような画面が表示されます。

```
                    プログラム変数の表示

プログラム . . . . . :   CUS310
反復レベル . . . . . :   1
開始桁 . . . . . :   1
形式 . . . . . :   *CHAR
桁数 . . . . . :   *DCL

変数 . . . . . :   &ARBAL
  タイプ . . . . . :   PACKED
  長さ . . . . . :   5 2
 '610.00'









続行するためには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

高水準言語 (HLL) によっては、ユーザーが指定したポインター変数 (HLL ポインター) を基底とする変数が使用できます。基底付き変数のポインターを明示的に指定しない場合は、HLL の宣言 (ある場合) によって指定されたポインターが使用されます。 HLL の宣言で基底付き変数の基底ポインターが明示的に指定されていない場合には、基底ポインターを明示的に指定しなければなりません。 PGMVAR パラメーターには、基底付き変数を参照するための基底ポインターを最高 5 個まで明示的に指定することができます。複数の基底ポインターを指定した場合は、最初の基底ポインターが 2 番目の基底ポインターを見つけるために使用され、2 番目の基底ポインターが 3 番目を見つけるのに使用される、というようになります。そして、基底ポインターのリストの最後にあるポインターが、該当の変数を見つけるのに使用されます。

関連情報:

ブレークポイント追加 (ADDBKP) コマンド

プログラム変数表示 (DSPPGMVAR) コマンド

変数の値の変更

デバッグ・モードでプログラム変数の値を変更するには、**プログラム変数変更 (CHGPGMVAR)** コマンド、 **高水準言語ポインター変更 (CHGHLLPTR)** コマンド、または**ポインター変更 (CHGPTR)** コマンドを使用します。

プログラム変数の値の変更は、変数の名前を指定し、同時にその変数のデータ・タイプに適合する値を指定して行います。例えば、その変数が文字タイプの変数であれば、文字タイプの値を指定しなければなりません。

変数の値を変更する際は、その変数が自動変数であるか静的変数であるかに留意してください。それは、この 2 つの変数では記憶域の割り振りが異なるためです。自動変数の記憶域は、プログラムの呼び出しに対応しています。プログラムが呼び出されるたびに、その変数の新しいコピーが自動記憶域に置かれます。自動変数に対する変更の効力は、その変更が行われたプログラム呼び出しの中だけに限定されます。

注: 言語によっては、呼び出しの定義がプログラム・レベルでなくプロシージャー・レベルで行われている場合があります。そのような言語では、自動変数の記憶域はプロシージャー呼び出しに対応しています。したがって、プロシージャーが呼び出されるたびに、その変数の新たなコピーが作られます。そして、自動変数に対する変更は、そのプロシージャーが呼び出されている間だけしか効力を持ちません。また、変更でき

るのは最新のプロシージャー呼び出しの自動変数だけです。なお、各コマンドの RCRLVL (反復レベル) パラメーターはプログラム単位で適用され、プロシージャー単位では適用されません。

静的変数の場合、記憶域は活動化に対応しています。 プログラムが何度呼び出されても、静的変数のコピーは記憶域内にただ 1 つだけしか存在しません。 静的変数に対する変更は、その活動状態が継続している限り効力を持ちます。

プログラム変数が静的変数か自動変数かを判別するには、その変数を使用するプログラムの作成時に、プログラムの中間表現 (IRP) リストを要求します (GENOPT パラメーターの *LIST および *XREF の指定)。

配列変数の値を変更する場合は、配列の要素を必ず指定しなければなりません。したがって、変更したい配列要素の添え字の値を指定する必要があります。

関連情報:

プログラム変数変更 (CHGPGMVAR) コマンド

ポインター変更 (CHGPTR) コマンド

HLL ポインター変更 (CHGHLLPTR) コマンド

ジョブを使用して他のジョブをデバッグする理由

以下のような状況では、あるジョブで実行されているプログラムを別のジョブを用いてデバッグすることができます。

- バッチ・ジョブを対話式ジョブによってデバッグできる。
- 対話式ジョブを他の対話式ジョブからデバッグできる。この場合、デバッグ情報の表示によりアプリケーション・プログラムの表示が中断されることはありません。
- ループしている対話式ジョブまたはバッチ・ジョブを中断させて、デバッグ・モードにすることができる。

ジョブ待ち行列に投入されたバッチ・ジョブのデバッグ:

ジョブ待ち行列に投入されたバッチ・ジョブのデバッグに他のジョブを用いることによって、バッチ・ジョブを処理の開始前にデバッグ・モードにして、ブレークポイントおよびトレースを設定することができます。

ジョブ待ち行列に投入されるバッチ・ジョブをデバッグする手順は以下のとおりです。

1. **ジョブ投入 (SBMJOB)** コマンドまたはジョブを自動的に投入するプログラムを使用して、HOLD(*YES) でバッチ・ジョブを投入する。

   ```
   SBMJOB HOLD(*YES)
   ```

2. **投入ジョブ処理 (WRKSBMJOB)** コマンドまたは**ジョブ待ち行列処理 (WRKJOBQ)** コマンドを用いて、そのジョブの修飾ジョブ名 (番号 / ユーザー / 名前) を判別する。 **ジョブ投入 (SBMJOB)** コマンドを実行した場合には、コマンドの処理が完了した時点で完了メッセージが表示され、そこに名前が示されます。

   WRKJOBQ (ジョブ待ち行列の処理) コマンドは、特定のジョブ待ち行列で開始を待っているすべてのジョブを表示します。該当するジョブに対してこの画面からオプション 5 を選択すると、そのジョブ名が得られます。

3. このバッチ・ジョブのデバッグに使用したい画面から、 **サービス・ジョブ開始 (STRSRVJOB)** コマンドを次のように入力する。

   ```
   STRSRVJOB JOB(qualified-job-name)
   ```

4. STRDBG コマンドを入力して、デバッグしたいすべてのプログラムの名前を指定する。対象のジョブがジョブ待ち行列上で待機中の間は、これ以外のデバッグ・コマンドは入力できません。
5. **ジョブ待ち行列解放 (RLSJOBQ)** コマンドを使用して、そのジョブ待ち行列を解放する。該当のジョブが開始可能になると、そのジョブのデバッグを開始できることを示す画面が表示されます。 F10 キーを押してコマンド入力画面を表示してください。
6. コマンド入力画面から、**ブレークポイント追加 (ADDBKP)** コマンドや**トレース追加 (ADDTRC)** コマンドなどのデバッグ・コマンドを入力する。
7. F3 キーを押してコマンド入力画面を終了させ、実行キーを押してそのバッチ・ジョブを開始させる。
8. そのジョブがブレークポイントで停止すると、通常のブレークポイント表示画面が表示されます。ジョブが終了した時点では、ブレークポイントおよびトレースを追加することはできません。また、変数を表示したり変更することもできません。ただしトレース・データは、**トレース・データ表示 (DSPTRCDTA)** コマンドを使用して表示することができます。
9. 他のバッチ・ジョブをデバッグする必要がある場合、まず**デバッグ・モード終了 (ENDDBG)** コマンドを使用してデバッグを終了させ、次に**サービス・ジョブ終了 (ENDSRVJOB)** コマンドを使用してジョブのサービスを終了させてください。

関連情報:

ブレークポイント追加 (ADDBKP) コマンド

トレース追加 (ADDTRC) コマンド

CL コマンド検索プログラム

ジョブ待ち行列から開始されないバッチ・ジョブのデバッグ:

システムで開始されるが、ジョブ待ち行列には投入されないバッチ・ジョブをデバッグできる場合があります。そのようなジョブは、実行の開始前に停止させることはできませんが、通常はデバッグを行うことができます。

ジョブ待ち行列から開始されないジョブをデバッグするには、次のことを行ってください。

1. ジョブの開始時点で呼び出されるプログラムの名前を変更する。例えば、ジョブがプログラム CUST310 を実行する場合、そのプログラムの名前を CUST310DBG などに変更してください。
2. プログラムの本来の名前 (変更前の名前) と同じ名前の小さな CL プログラムを作成する。小さな CL プログラムでは、**ジョブ延期 (DLYJOB)** コマンドを使用して 1 分延期させ、次に CALL コマンドを使用して、名前が変更されたプログラムを呼び出します。
3. CL プログラムを強制的に 1 分間遅らせるバッチ・ジョブの開始を認める。
4. **活動ジョブ処理 (WRKACTJOB)** コマンドを使用して、実行中のバッチ・ジョブを見つける。画面上で、そのジョブの前にオプション 5 を入力して修飾ジョブ名を表示させてください。
5. **サービス・ジョブ開始 (STRSRVJOB)** コマンドを次のように入力する。

   ```
   STRSRVJOB JOB(qualified-job-name)
   ```

6. STRDBG および他のデバッグ・コマンド (**ブレークポイント追加 (ADDBKP)** コマンドや**トレース追加 (ADDTRC)** コマンドなど) を入力する。デバッグの進め方は通常と同じです。

関連情報:

ブレークポイント追加 (ADDBKP) コマンド

トレース追加 (ADDTRC) コマンド

CL コマンド検索プログラム

実行中のジョブのデバッグ:

既に実行されているジョブのデバッグは、そのジョブがこれから実行するステートメントをユーザーが知っている場合に可能です。

例えば、ジョブがループしている場合、またはジョブがデバッグしたいプログラムの実行をまだ完了していない場合には、実行中のジョブのデバッグを行うことができます。実行中のジョブのデバッグを行う手順は以下のとおりです。

1. **活動ジョブ処理 (WRKACTJOB)** コマンドを使用して、実行中のジョブを見つける。画面上で、そのジョブの前にオプション 5 を入力して修飾ジョブ名を表示させてください。
2. **サービス・ジョブ開始 (STRSRVJOB)** コマンドを次のように入力する。

   ```
   STRSRVJOB JOB(qualified-job-name)
   ```

3. **デバッグ開始 (STRDBG)** コマンドを入力する。 (このコマンドを入力しても、そのジョブの実行は停止しない)。

   注: **デバッグ表示 (DSPDBG)** を使用して呼び出しスタックを表示することができます。ただし、何らかの理由でプログラムが停止した場合を除き、呼び出しスタックは瞬間的に正しく表示されるだけで、プログラムの実行が続行されます。
4. 実行されるステートメントが分かっていれば、 **ブレークポイント追加 (ADDBKP)** コマンドにより、ジョブを特定のステートメントの位置で停止させることができる。

   実行されるステートメントが分かっていない場合には、次のようにしてください。
   a. **トレース追加 (ADDTRC)** コマンドを入力する。
   b. しばらくの後、**トレース除去 (RMVTRC)** コマンドを入力してプログラムのトレースを停止させる。
   c. **トレース・データ表示 (DSPTRCDTA)** コマンドを入力して、処理が終了したステートメントを表示させる。このトレース・データから、次に処理すべきステートメント (例えば、プログラム・ループ内のステートメントなど) を判別してください。
   d. **ブレークポイント追加 (ADDBKP)** コマンドにより、ジョブを特定のステートメントの位置で停止させる。
5. プログラムがブレークポイントで停止した時点で必要なデバッグ・コマンドを入力する。

関連情報:

デバッグ表示 (DSPDBG) コマンド

トレース・データ表示 (DSPTRCDTA) コマンド

ブレークポイント追加 (ADDBKP) コマンド

トレース追加 (ADDTRC) コマンド

CL コマンド検索プログラム

別の対話式ジョブのデバッグ:

ジョブが実行中であるか、あるいはメニュー画面やコマンド入力画面で待機中であるかにかかわらず、別のディスプレイ装置からジョブをデバッグすることができます。

他の対話式ジョブのデバッグを行うには、次のステップに従ってください。

1. デバッグしたいジョブの修飾ジョブ名を入手する。この名前を決定するには、デバッグされるジョブの表示画面から**ジョブ表示 (DSPJOB)** コマンドを入力するか、**活動ジョブ処理 (WRKACTJOB)** コマンドを実行します。

2. 修飾ジョブ名を使用して**サービス・ジョブ開始 (STRSRVJOB)** コマンドを入力する。
3. **デバッグ開始 (STRDBG)** コマンドおよび必要な他のすべてのデバッグ・コマンドを入力する。ジョブが既に実行中の場合には、**デバッグ表示 (DSPDBG)** コマンドを入力して、プログラム内のどのステートメントが処理中であるかを判別する必要があります。

デバッグされているジョブがブレークポイントで停止している時、そのディスプレイ装置はロックされています。

関連情報:

デバッグ開始 (STRDBG) コマンド

デバッグ表示 (DSPDBG) コマンド

CL コマンド検索プログラム

ジョブを別のジョブからデバッグする際の考慮事項**:**

ほとんどのジョブは、別のジョブからデバッグできますが、以下の点を考慮する必要があります。

- 保留または中断しているジョブをデバッグすることはできない (例えば、他のグループ・ジョブや 2 次ジョブを実行している場合など)。
- **サービス・ジョブ開始 (STRSRVJOB)** コマンドによって他のジョブにサービス処理を行っている場合には、サービス処理を行っているジョブもデバッグすることはできない。デバッグ・コマンドは、サービスを受けるジョブに対してだけ適用されます。サービスを行っているジョブをデバッグする場合には、他のジョブに対するサービスを終了させるか、別のジョブによってデバッグする必要があります。
- デバッグ・コマンドは他のジョブで作動しますが、それはそのジョブがブレークポイントで停止していない場合でも可能です。例えば、実行中のジョブをデバッグしているときに**プログラム変数表示 (DSPPGMVAR)** コマンドを入力すると、指定した変数が表示されます。そのジョブは実行を続行するので、その変数の値はコマンドの入力後直ちに変わることがあります。
- デバッグされているジョブは、デバッグ・コマンドに対応できるだけの優先順位が必要となる。優先順位が低いバッチ・ジョブをデバッグすると、そのジョブが処理時間をまったく獲得できない場合、デバッグ・コマンドを出してもジョブからの応答を待つだけになってしまいます。ジョブが応答しない場合はコマンドが終了し、エラー・メッセージが表示されます。
- 自分自身をデバッグしているジョブに対して、サービス処理やデバッグを行うことはできない。ただし、他のジョブのサービス処理やデバッグを行っているジョブに対して、サービス処理やデバッグを行うことはできます。

マシン・インターフェース・レベルでのデバッグ

ユーザー・プログラムをマシン・インターフェース (MI) レベルでデバッグするには、コマンドの PGMVAR パラメーターに MI オブジェクト定義ベクトル (ODV) 番号を指定し、STMT パラメーターに MI 命令番号を指定します。

ブレークポイントの場合、システムは、高水準言語 (HLL) ステートメント番号で停止するのと同様に、該当する MI 命令番号で停止します。システムに MI レベルでデバッグすることを伝えるために、ODV または MI 命令番号の前に必ずスラッシュ (/) を付け、それを単一引用符で囲む必要があります (`'/1A'` など)。

ODV および MI 命令番号は、ほとんどの高水準言語コンパイラーが生成する IRP リストから入手できます。プログラム作成時に IRP リストを生成したい場合、GENOPT パラメーターに *LIST 値を指定してください。

注**:** マシン・インターフェース・レベルでデバッグを行う場合は、マシン・インターフェース・レベルで定義された特性だけを使用することができます。通常、テスト環境へ渡される HLL 特性は、マシン・インターフェース・レベルのデバッグでは使用できません。このような HLL 特性には、変数タイプ、長さ、小数部分の長さ、および配列情報などがあります。例えば、HLL プログラムの数値変数は、小数点により位置合わせが正しく行われないまま表示されたり、文字ストリングとして表示される場合があります。

関連タスク:

451 ページの『トレースのプログラムへの追加』
トレースを追加するには、トレースの対象となるステートメントを指定し、また同時にプログラム変数の名前も必要に応じて指定します。

セキュリティーの考慮事項

プログラムをデバッグするには、ユーザーはそのプログラムに対する変更 (*CHANGE) 権限を持っていなければなりません。

他のユーザーのユーザー・プロファイルを借用して変更 (*CHANGE) 権限を得ても、プログラムをデバッグする権限を有するものとは見なされません。これは本来権限のないユーザーが、他のユーザーのユーザー・プロファイルを借用してデバッグ・モードでプログラム・データにアクセスするのを防止するためです。

また、他のユーザーの権限を借用してプログラムをデバッグしている場合、ユーザー定義のブレークポイントでは、ユーザーは自分自身のユーザー・プロファイルで指定されている権限だけを持ち、借用したプロファイルが持つ権限を使用することはできません。 **ブレークポイント追加 (ADDBKP)** コマンドで追加されたブレークポイントであろうと、また監視の対象になっていないエスケープ・メッセージが原因で生じたブレークポイントであろうと、すべてのブレークポイントに関して、それ以前のプログラム呼び出しによって借用した権限は、そのユーザーのものとして使用することはできません。

一時的にブレークポイントを除去する操作

特定の CL コマンドを使用してライブラリーまたはプログラムを指定する場合、デバッグ機能の実行中に、ブレークポイントまたはステートメント・トレースをプログラムから一時的に除去することができます。

ブレークポイントとステートメントのトレースは、CL コマンドの実行が完了すると、復元されます。ブレークポイントまたはトレースが除去された場合に CPD190A メッセージがジョブ・ログ内に入ります。それらの復元時には、さらに別の CPD190A メッセージがジョブ・ログ内に入ります。

以下の CL コマンドを使用した場合に、ブレークポイントまたはステートメント・トレースがプログラムから一時的に削除されます。

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

注**:** プログラム中でこれらの CL コマンドを実行している時は、プログラムにブレークポイントまたはトレースを追加できない場合があります。プログラム中でこれらのコマンドのいずれかを実行している時に**ブレークポイント追加 (ADDBKP)** コマンドまたは**トレース追加 (ADDTRC)** コマンドを入力すると、エラー・メッセージ CPF7102 が発行されます。

関連情報:

オブジェクトおよびライブラリー

オブジェクトおよびライブラリーに固有のタスクおよび概念には、オブジェクトに対して実行される機能、ライブラリーの作成、およびオブジェクト権限の指定が含まれます。

オブジェクトとは、名前が付けられた記憶域空間のことで、それ自身を記述する 1 組の特性で構成されており、データを含む場合もあります。記憶域に存在しており、記憶域のスペースを占有し、操作することのできるものはオブジェクトです。オブジェクトの属性には、オブジェクトの名前、タイプ、サイズ、作成日、およびオブジェクトを作成したユーザーによる記述などがあります。また、オブジェクトの値とは、そのオブジェクトに保管されている情報の集まりのことです。例えば、プログラムの値とはプログラムを構成しているコードのことであり、ファイルの値とはファイルを構成しているレコードの集まりのことです。オブジェクトとは概念を表す用語で、項目が何であるかに関係なく、システムに保管できるあらゆる項目に言及する際に使用されます。

注: オブジェクトは、ライブラリーとディレクトリーの両方に常駐させることができます。 (以前は、オブジェクトはライブラリーにしか常駐できませんでした。) このトピックに含まれる情報は、ライブラリーに常駐するオブジェクトに関するものだけです。

関連概念:

120 ページの『IBM i オブジェクト』
オブジェクト は、記憶域に存在する (スペースを占める) 名前付きの単位であり、オペレーティング・システムはこの単位に対して操作を実行します。IBM i オペレーティング・システムがすべてのデータ処理情報を保管して処理するための手段は、IBM i のオブジェクトによって提供されます。

関連タスク:

309 ページの『データ域のロックと割り振り』
データ域のロックと割り振りは、データ域が複数のジョブから一度にアクセスされないことを保証するのに役立ちます。

関連情報:

統合ファイルシステム

オブジェクト

オブジェクトとは、名前が付けられた記憶域空間のことで、それ自身を記述する 1 組の特性で構成されており、データを含む場合もあります。記憶域に存在しており、記憶域のスペースを占有し、操作することのできるものはオブジェクトです。

オブジェクト・タイプおよび共通属性

システム上の各オブジェクト・タイプは、システム内に固有の目的を持ち、そのオブジェクトを記述する共通の属性セットを持っています。

各オブジェクト・タイプには、そのオブジェクト・タイプを処理する一連のコマンドが用意されています。

関連概念:

124 ページの『外部オブジェクト・タイプ』
外部オブジェクトの多くのタイプは、ライブラリーに保管されます。一部のタイプの外部オブジェクトは、ディレクトリー内の統合ファイル・システムのみに保管できます。

関連資料:

488 ページの『オブジェクト記述の表示』
オブジェクト記述表示 (DSPOBJD) コマンドまたは**オブジェクト処理 (WRKOBJ)** コマンドを使用して、オブジェクトの記述を表示することができます。

関連情報:

オブジェクト記述表示 (DSPOBJD) コマンド

オブジェクトに対して実行される機能

オブジェクトに対して多くの機能を実行することができます。システムが自動的に実行する機能と、ユーザーがコマンドから要求する機能があります。

システムが自動的に実行する機能:

自動的に実行される機能によるオブジェクトの処理は、一貫性、安全性、および正確性が保証されています。

自動的に実行される機能には次のものがあります。

- オブジェクト・タイプの検査。オブジェクト・タイプに対して実行可能な機能であることを確認するために、オブジェクトのタイプおよびオブジェクトに対して実行する機能のタイプがシステムにより検査されます。例えば、CALL コマンドに指定されたオブジェクトがプログラムでなかった場合に、呼び出し機能は実行できません。
- オブジェクト権限の検査。ユーザーがオブジェクト・タイプに対して機能を実行する権限があることを確認するために、オブジェクト、機能、およびユーザーがシステムにより検査されます。例えば、USERA に OBJB を使用する権限がまったくない場合には、ユーザーはそのオブジェクトに対してどのような機能の実行も要求することはできません。
- オブジェクト・ロックの強制。複数のユーザーが同じオブジェクトを同時に使用しようとした場合には、オブジェクトの保全性がシステムにより保たれます。同じオブジェクトに対して同時に変更を行おうとするとロックアウトが生じます。また、オブジェクトの変更中は、他のユーザーがこのオブジェクトを使用することはできません。
- オブジェクトの損傷の検出と通知。オブジェクトの処理過程のエラーはシステムにより監視され、認識不能なオブジェクト内容が原因で起こった不測の障害は通知されます。このような障害の通知には、オブジェクトの損傷を示す標準的なメッセージが使用されます。システムは、このような障害がめったに起こらないように設計されており、万一障害が起こった場合でも、障害を監視し通知することにより、システムの保全性が保たれるようになっています。

コマンドを使用して実行できる機能:

コマンドを使用して要求できる機能には、2 つのタイプがあります。オブジェクト・タイプごとに異なる特定の機能と、共通機能です。

- 各オブジェクト・タイプごとに異なる特定の機能。例えば、作成、変更、および表示などの機能です。特定の機能については、本書のオブジェクト・タイプについての項で説明します。
- 共通機能。オブジェクトに対して一般的に適用される共通機能。本書では以下の機能について説明しています。

表 27. オブジェクトに関する共通機能

476 ページの『複数のオブジェクトまたは 1 つのオブジェクトの探索』
478 ページの『ライブラリー仕様の権限』

表 27. オブジェクトに関する共通機能 (続き)

482 ページの『ライブラリーへのオブジェクトの収容』
488 ページの『オブジェクトの記述』
488 ページの『オブジェクト記述の表示』
492 ページの『オブジェクト記述の検索』
495 ページの『システム内の不要オブジェクトの検出』
501 ページの『ライブラリー間のオブジェクト移動』
504 ページの『複製オブジェクトの作成』
506 ページの『オブジェクト名の変更』
510 ページの『オブジェクトの削除』
511 ページの『リソースの割り振り』
514 ページの『オブジェクトのロック状態の表示』
518 ページの『オブジェクトの存在の検査』

ライブラリー

IBM i オペレーティング・システムでは、オブジェクトはグループ化されて、ライブラリーと呼ばれる特殊なオブジェクトに入れられます。

オブジェクトはライブラリーを使用して探索されます。ライブラリー内のオブジェクトにアクセスするためには、そのライブラリーおよびオブジェクトに対する権限がなければなりません。

オブジェクト名を指定したパラメーターと同じパラメーターにライブラリー名も指定した場合には、オブジェクト名は修飾名 と呼ばれます。

ライブラリーを作成する時点で、そのライブラリーをどのユーザー補助記憶域プール (ASP) に作成するかを指定することができます。ライブラリーは、基本ユーザー ASP または独立 ASP に作成することができます。そのライブラリーに作成されるすべてのオブジェクトは、そのライブラリーと同じ ASP に作成されます。

修飾名の指定が必要なコマンドを入力する場合には、オブジェクト名は次のようになります。

```
DISTLIB/ORD040C
```

この例では、受注プログラム ORD040C はライブラリー DISTLIB に入っていることを示しています。

プロンプト機能を使用してコマンドを入力する場合には、オブジェクト名およびライブラリー名の両方のプロンプトが表示され、修飾名の入力が求められます。多くのコマンドで特定のライブラリー名を指定することも、 *CURLIB (ジョブの現行ライブラリー) を指定することも、あるいはライブラリー・リストを使用することもできます。

関連概念:

480 ページの『オブジェクトのセキュリティーに関する考慮事項』
システムは、ユーザーが参照したオブジェクトにアクセスする場合に、ユーザーが要求した方法でそのオブジェクトを使用するための権限を持っているかどうかを判断するための検査を行います。

関連タスク:

478 ページの『ライブラリー仕様の権限』
特定の権限をライブラリーのユーザーに付与できます。

関連情報:

独立ディスク・プールの例

ライブラリー・リスト

システムは、ライブラリー・リストを使用してオブジェクトを見つけます。

修飾名を指定することができるコマンドでは、ライブラリー名の指定を省略することができます。この指定を省略すると、次のいずれかの処理が行われます。

- 作成コマンドの場合には、オブジェクトはそのタイプに応じて、ユーザーの現行ライブラリー (*CURLIB) またはシステム・ライブラリーのいずれかに作成されます。例えば、プログラムは *CURLIB に作成され、そこに入れられます。権限リストはシステム・ライブラリー QSYS に作成され保管されます。
- 作成コマンド以外のコマンドの場合には、システムは通常、ライブラリー・リストを使用してオブジェクトを見つけます。

IBM i オペレーティング・システムによって使用されるライブラリー・リストは、次の 4 つの部分から成り立っています。

システム部分
: ライブラリー・リストのシステム部分には、システムに必要なオブジェクトが入っています。

プロダクト (実行) ライブラリー
: ライブラリー・リストには 2 つのプロダクト・ライブラリーを含めることができます。システムはプロダクト・ライブラリーを使用して、コマンドの処理を QSYS 以外のライブラリーに依存する言語およびユーティリティーをサポートします。

  ユーザー・コマンドおよびユーザー・メニューの場合も、関連するオブジェクトが確実に見つかるように、 **コマンド作成 (CRTCMD)** コマンドおよび**メニュー作成 (CRTMNU)** コマンドの PRDLIB パラメーターでプロダクト・ライブラリーを指定することができます。

  プロダクト・ライブラリー (例えば、QRPG など) はシステムにより管理され、必要に応じてライブラリー・リスト内のプロダクト・ライブラリーとして確保されている位置に入れられます。現行ライブラリー、またはライブラリー・リストのユーザー部分にあるライブラリーと重複していても構いません。

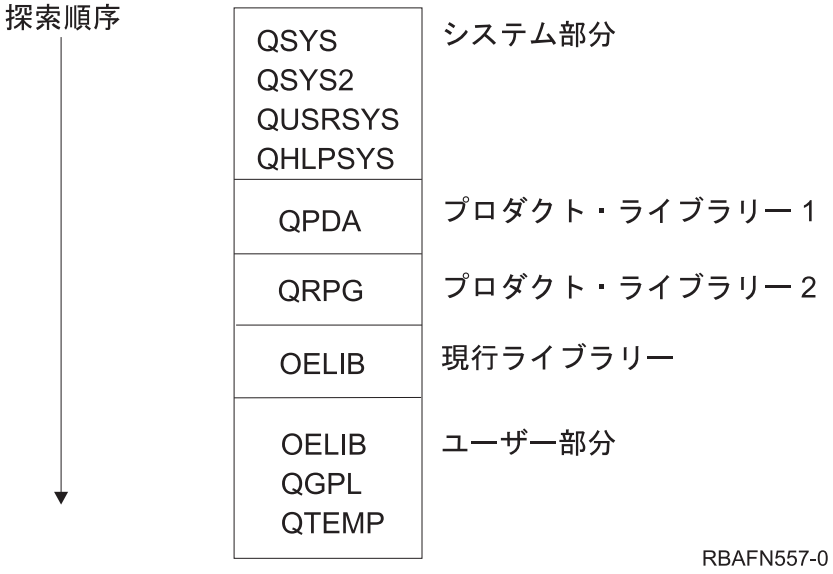
  例えば、プロダクト・ライブラリーを使用するコマンドまたはメニューを開始したときに、プロダクト・ライブラリーがライブラリー・リストにあるとします。新しいコマンドが終了するか、ユーザーが新しいメニューの使用を終えるまで、システムはライブラリー・リスト内のプロダクト・ライブラリーを、新しいプロダクト・ライブラリーに置き換えようとします。

現行ライブラリー
: 現行ライブラリーは、ライブラリー・リスト内の任意のライブラリーと重複していてもよく、そうでなくても差し支えありません。多くのコマンドで、現行ライブラリーを示す *CURLIB をライブラリーとして使用して、ジョブの現行ライブラリーとして指定されているライブラリーを指定することができます。ライブラリー・リストに現行ライブラリーが存在せず、しかも *CURLIB がライブラリーとして指定されている場合には、QGPL が使用されます。現行ライブラリー変更 (CHGCURLIB) コマンドまたはライブラリー・リスト変更 (CHGLIBL) コマンドを使用して、ジョブの現行ライブラリーを変更することができます。

ユーザー部分
: ライブラリー・リストのユーザー部分には、システムのユーザーおよびアプリケーションにより参

照されるライブラリーが入っています。ユーザー部分、プロダクト・ライブラリー、および現行ライブラリーは、システムの各ジョブごとに異なっていても差し支えありません。ライブラリーの数は、250 個までに制限されています。

次の図は、ライブラリー・リストの構成の一例です。



注**:** 原始ステートメント入力ユーティリティー (SEU) を使用する場合には、システムはライブラリー QPDA をプロダクト・ライブラリー 1 に入れます。 SEU を使用してソースの構文検査を行う場合には、2 番目のプロダクト・ライブラリーがプロダクト・ライブラリー 2 に追加されます。例えば、RPG ソースの構文を検査する場合には、QPDA がプロダクト・ライブラリー 1、QRPG がプロダクト・ライブラリー 2 になります。これ以外のシステムでは多くの場合、プロダクト・ライブラリー 2 は使用されません。

関連情報:

メニュー作成 (CRTMNU) コマンド

ライブラリー・リスト変更 (CHGLIBL) コマンド

コマンド作成 (CRTCMD) コマンド

現行ライブラリー変更 (CHGCURLIB) コマンド

ライブラリー・リスト項目追加 (ADDLIBLE) コマンド

ライブラリー・リスト編集 (EDTLIBL) コマンド

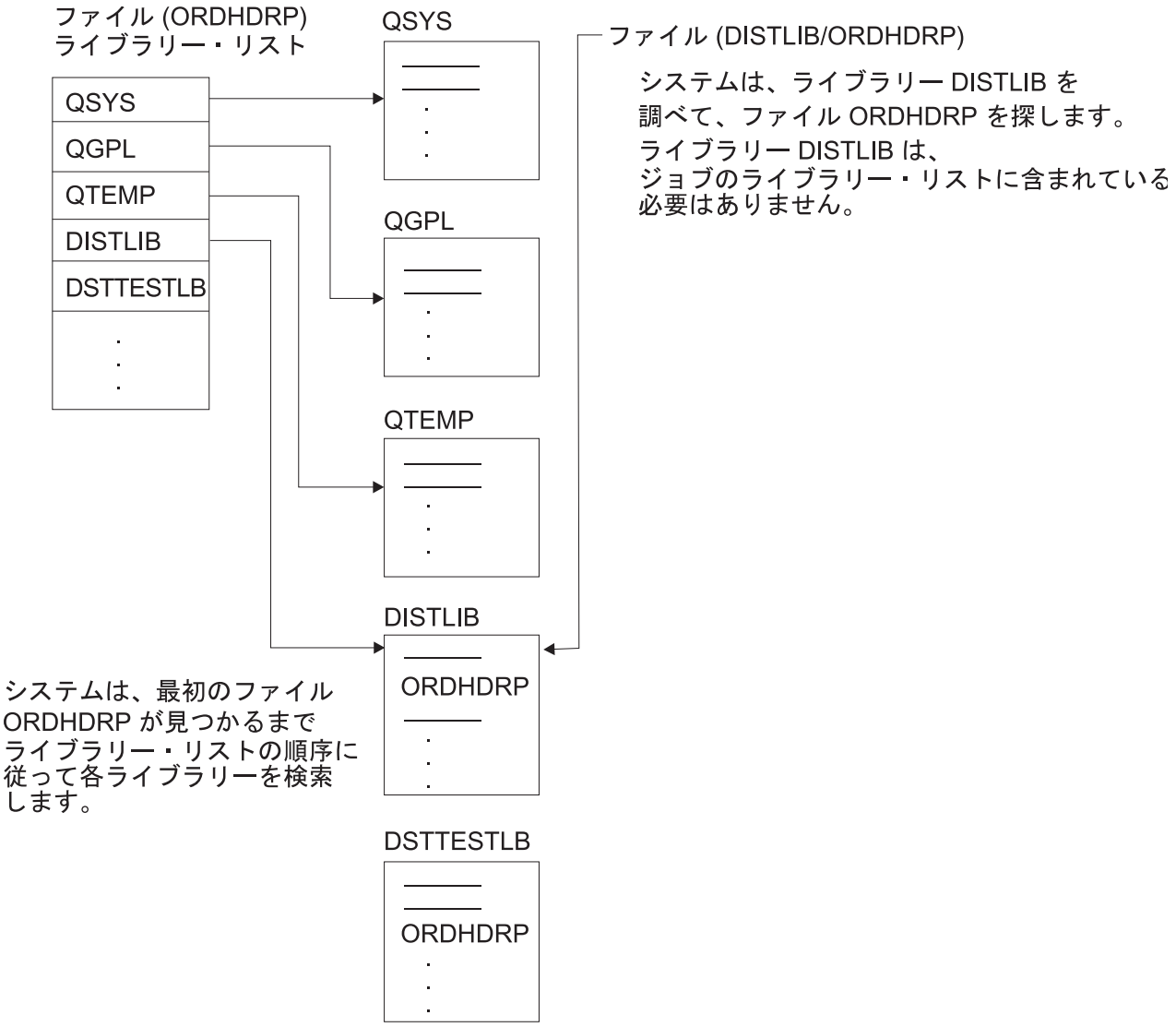
ライブラリー・リスト項目除去 (RMVLIBLE) コマンド

ライブラリー・リストの使用に関する機能**:**

ライブラリー・リストの使用によって、システム上のオブジェクトの探索が簡略化されます。

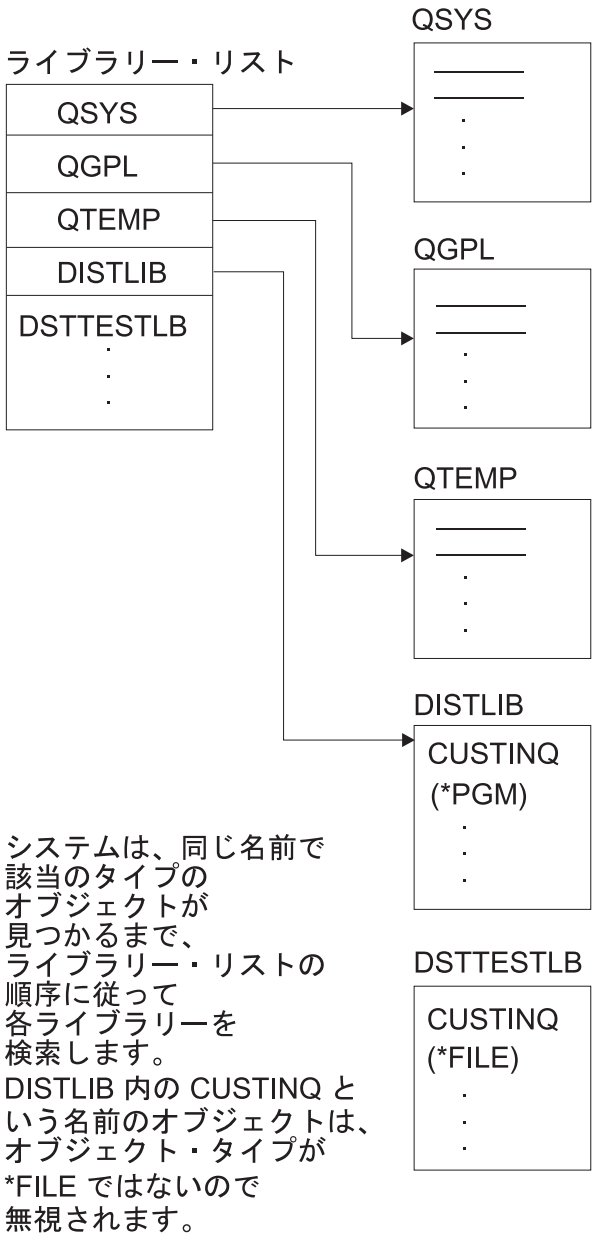
各ジョブには、関連するライブラリー・リストが存在します。ライブラリー・リストを使用してオブジェクトを見つける場合には、指定の名前とタイプのオブジェクトが見つかるまで、リスト内の各ライブラリーがリスト内での順序に従って探索されます。同じタイプと名前を持つオブジェクトがリスト内に 2 つ以上ある場合には、ライブラリー・リスト内で最初に見つかったライブラリーのオブジェクトを取り出します。次の図は、ライブラリー・リスト (*LIBL) を使用する場合とライブラリー名を指定した場合のオブジェクトの探索方法を示しています。

注**:** 別の方法として、 *LIBL ではなく *NLVLIBL を使用してコマンドを修飾することもできます。 CL プログラム、コマンド行、または他の普段コマンドを入力しているところでコマンドを入力します。システムは *NLVLIBL を使用することによって、 *CMD オブジェクトを探索するのに使用するライブラリーを決定します。 *NLVLIBL を指定すると、ライブラリー・リスト内の各国語サポート・ライブラリーだけが探索されます。

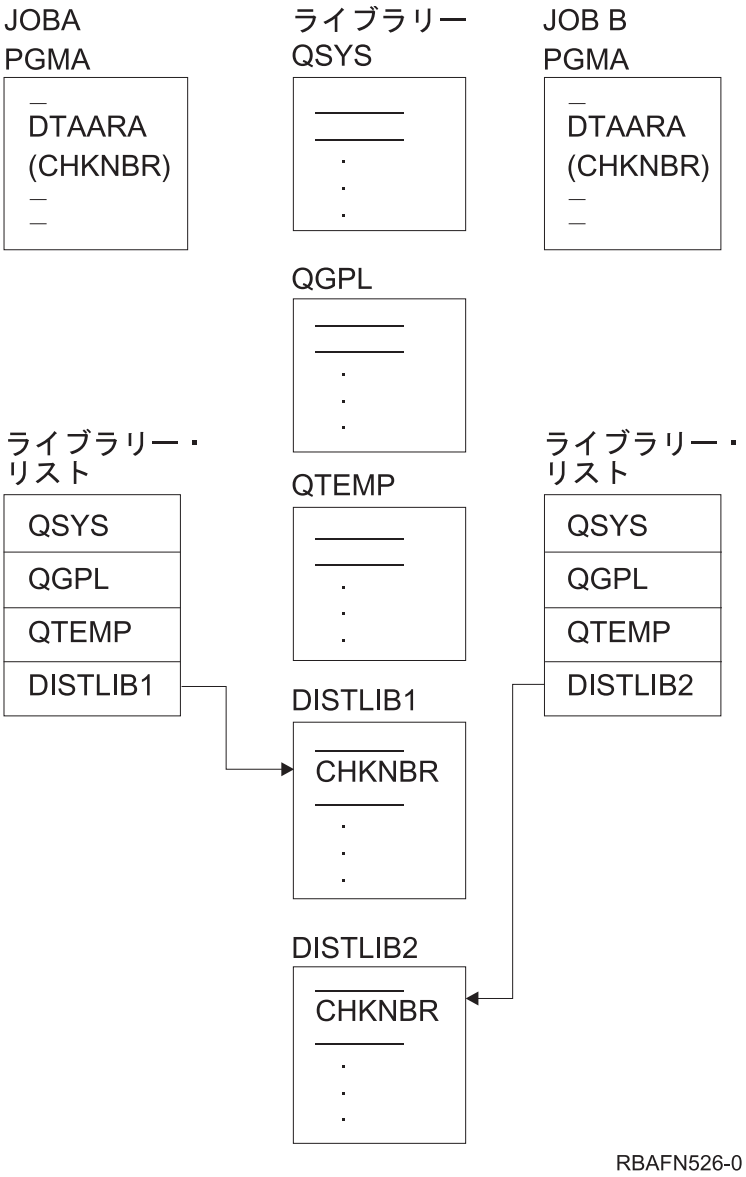


次の図は、名前が同じでタイプの異なる 2 つのオブジェクトがライブラリー・リストに入っている場合にどうなるかを示しています。次のように指定すると、システムはライブラリー・リストからタイプが *FILE の CUSTINQ を探索します。

```
DSPOBJD OBJ(*LIBL/CUSTINQ) OBJTYPE(*FILE)
```



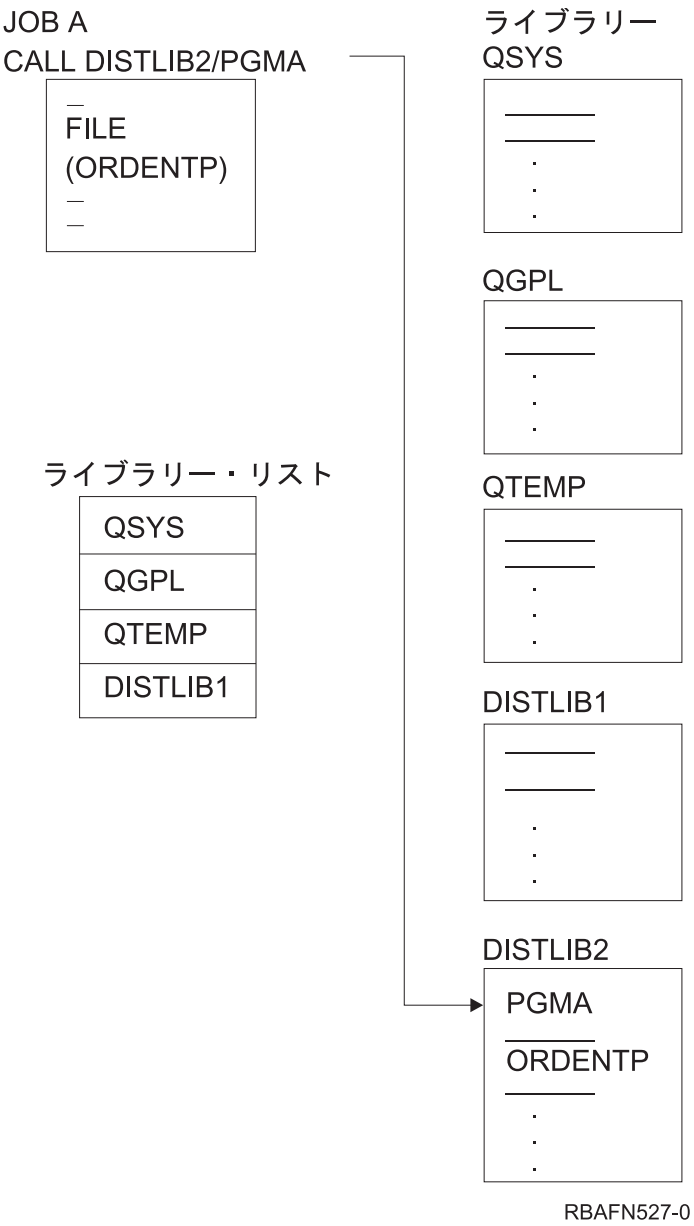
一般に、ライブラリー・リストの方が修飾名より柔軟性が高く、使い方も簡単です。ライブラリー名を入力しないで済むという利点に加えて、さらに重要な利点は、アプリケーションで異なるデータを用いてその機能を実行したい場合に、アプリケーションを変更する必要がなく、別々のライブラリー・リストを使用して行うということです。例えば、PGMA という CL プログラムでデータ域 CHKNBR を更新する場合、ライブラリー名が指定されていなければ、ライブラリー・リストの使い方次第で、異なるライブラリーの CHKNBR という名前のデータ域を同じプログラムで更新することができます。次の図では、JOBA と JOBB がどちらも PGMA を呼び出しています。



ただし、次のような場合には、修飾名を使用するほうが便利です。

- 使用するオブジェクトがジョブのライブラリー・リストに入っていない場合。
- ライブラリー・リスト内に同名のオブジェクトが複数存在し、特定のライブラリーのオブジェクトを使用したい場合。
- セキュリティーの理由から、特定のライブラリーを使用したい場合。

ただし、修飾名を使用してプログラムを呼び出し、そのプログラムで修飾されていない名前のファイルをオープンしようとした場合には、それらのファイルがライブラリー・リストに入っていなければ、オープンすることはできません。次の例を参照してください。



この例では、PGMA に対する呼び出しは、CALL コマンドでそのプログラム名が修飾されているので正常に実行されます。しかし、そのプログラムでファイル ORDENTP をオープンしようとすると、そのファイルがライブラリー・リスト内のどのライブラリーにも含まれておらず、しかもファイル名が修飾されていないので、オープン操作は失敗します。ライブラリー DISTLIB2 をライブラリー・リストに追加するか、または修飾ファイル名を指定して、このプログラムでそのファイルをオープンすることができます。高水準言語の中には、修飾ファイル名を指定できないものもあります。その場合は、一時変更 (OVRxxx) コマンドを使用して修飾名を指定することができます。

関連情報:

オブジェクト署名および署名検査

ジョブのライブラリー・リスト:

各ジョブのライブラリー・リストは、システム部分、ユーザー部分、現行ライブラリー、およびプロダクト・ライブラリーという 4 つの部分で構成されています。

システム部分だけは、必ずライブラリー・リストに組み込まれていなければなりません。

システムの出荷時には、システム値 QSYSLIBL にライブラリー・リストのシステム部分のライブラリーの名前として、 QSYS、QSYS2、QHLPSYS、および QUSRSYS の各ライブラリー名が含まれています。システム値 QUSRLIBL には、ライブラリー・リストのユーザー部分を示すライブラリーの名前が含まれています。

QSYSLIBL には最高 15 個のライブラリー名を入れることができ、QUSRLIBL には最高 25 個のライブラリー名を入れることができます。ジョブのライブラリー・リストのシステム部分を変更したい場合には、**システム・ライブラリー・リスト変更 (CHGSYSLIBL)** コマンドを使用してください。 QSYSLIBL または QUSRLIBL の値を変更する場合には、 **システム値変更 (CHGSYSVAL)** コマンドを使用します。これらのシステム値に対する変更は、システム値の変更後に新たに開始されるジョブから有効になります。

関連情報:

システム値変更 (CHGSYSVAL) コマンド

システム・ライブラリー・リスト変更 (CHGSYSLIB) コマンド

ライブラリー・リストの変更:

ライブラリー・リストをさまざまな方法で変更することができます。

ジョブを実行する場合に、 **ライブラリー・リスト項目追加 (ADDLIBLE)** コマンドによるライブラリー・リストへの項目の追加、 **ライブラリー・リスト項目除去 (RMVLIBLE)** コマンドによるライブラリー・リストからの項目の除去、あるいは **ライブラリー・リストの変更 (CHGLIBL)** コマンドや **ライブラリー・リストの編集 (EDTLIBL)** コマンドによるライブラリー・リスト内のライブラリーの変更などを行うことができます。これらのコマンドはライブラリー・リストのユーザー部分を変更するためのもので、システム部分を変更することはできません。

現行ライブラリーは、 **現行ライブラリー変更 (CHGCURLIB)** コマンドまたは CHGLIBL コマンドを使用して変更または追加することができます。現行ライブラリーは、サインオン時にユーザーのユーザー・プロファイルにより、または**ジョブ投入 (SBMJOB)** コマンドを使用して変更することができます。プロダクト・ライブラリーは、CL コマンドを用いて追加することはできません。これらのライブラリーは、それを使用するコマンドまたはメニューが実行される時点でシステムにより追加されます。プロダクト・ライブラリーは CL コマンドによって変更することはできませんが、ライブラリー・リスト変更 (QLICHGLL) API によって変更することができます。

上記の各コマンドを使用した場合に、ライブラリー・リストに対する変更が行われるのは、そのコマンドを実行するジョブに対してだけであり、その変更が有効なのはそのジョブが実行されている間、またはジョブのライブラリー・リストを再度変更するまでの間だけです。これらのコマンドを使用してライブラリー・リストを変更する場合には、コマンドの実行時に対象のライブラリーが存在していなければなりません。ご使用のジョブのライブラリー・リストにあるライブラリーは、削除できません。別のジョブのライブラリー・リストにある場合でも、ライブラリー検索リストでライブラリーをロックするようにシステム値 QLIBLCKLVL が設定されていると、削除できません。

ジョブが開始されると、ジョブ記述に指定されている値または **ジョブ投入 (SBMJOB)** コマンドに指定されている値に基づいて、ライブラリー・リストのユーザー部分が決定されます。 *SYSVAL という値を指定することもできます。この値を指定すると、システム値 QUSRLIBL に指定されているライブラリーがライブラリー・リストのユーザー部分になります。 **バッチ・ジョブ (BCHJOB)** コマンドまたは **ジョブ投入 (SBMJOB)** コマンドの両方にライブラリー名を指定した場合には、ジョブ記述およびシステム値 QUSRLIBL に指定されているライブラリーはどちらも、BCHJOB コマンドまたは SBMJOB コマンドに指定したライブラリー名により一時変更されます。

QUSRLIBL に指定されているライブラリー・リストのユーザー部分が、個々のジョブにおける種々のコマンドにより一時変更される場合の順序は次のとおりです。

- ジョブ記述にライブラリー・リストを指定することにより、ジョブの実行時点で QUSRLIBL に指定されているライブラリー・リストを、このジョブ記述のライブラリー・リストで一時変更することができます。
- **バッチ・ジョブ (BCHJOB)** コマンドまたは **ジョブ投入 (SBMJOB)** コマンドを使用してジョブを投入する場合には、そのコマンドにライブラリー・リストを指定することができます。このリストは、ジョブ記述またはシステム値 QUSRLIBL に指定されているライブラリー・リストを一時変更します。
- **ジョブ投入 (SBMJOB)** コマンドを使用してジョブを投入する場合には、ライブラリー・リストとして *CURRENT (デフォルト値) を指定することができます。 値 *CURRENT は、**ジョブ投入 (SBMJOB)** コマンドを発行したジョブのライブラリー・リストを使用することを示します。
- ジョブ内では、ADDLIBLE コマンド、RMVLIBLE コマンド、または CHGLIBL コマンドを使用することができます。これらのコマンドを使用すると、それまでのライブラリー・リストが一時変更されます。
- ジョブの現行ライブラリーは、CHGCURLIB コマンドまたは CHGLIBL コマンドを使用して変更することができます。

ライブラリー・リストの変更のたびに CHGLIBL コマンドを入力する代わりに、このコマンドを次のように CL プログラムに入れておくこともできます。

```
PGM  /* SETLIBL - Set library list */
CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
ENDPGM
```

通常、このライブラリー・リストを使用して処理を実行する場合には、プログラムを毎回呼び出す代わりに、初期プログラムを作成してライブラリー・リストを設定することができます。

```
PGM  /* Initial program for QPGMR */
CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
TFRCTL PGM(QPGMMENU)
ENDPGM
```

このプログラムを作成し、それを適用するユーザー・プロファイルを変更して、新しい初期プログラムを指定しなければなりません。このプログラムから QPGMMENU プログラムに制御が移り、プログラマー・メニューが表示されます。

場合によっては、初期プログラムに指定したライブラリー・リストにライブラリーを追加する必要が生じることもありますが、その場合は ADDLIBLE コマンドを使用して、必要なライブラリーをライブラリー・リストに追加することができます。例えば、次のコマンドを使用すると、ライブラリー JONES がライブラリー・リストの末尾に追加されます。

```
ADDLIBLE LIB(JONES) POSITION(*LAST)
```

ジョブの一部で別のライブラリー・リストが必要になる場合には、現行のライブラリー・リストを保管し、後でそれを復元するための次のような CL プログラムを作成することができます。

```
       PGM
       DCL &LIBL *CHAR 2750
       DCL &CMD *CHAR 2760
(1)    RTVJOBA USRLIBL(&LIBL)
(2)    CHGLIBL (QGPL QTEMP)
       .
       .
       .
(3)    CHGVAR &CMD ('CHGLIBL (' *CAT &LIBL *TCAT ')')
(4)    CALL QCMDEXC (&CMD 2760)
```

```
.
.
.
ENDPGM
```

(1) ライブラリー・リストを保管するためのコマンド。ライブラリー・リストは変数 &LIBL に保管されます。各ライブラリー名は 10 バイトを占め (必要に応じて右側にブランクが埋め込まれる)、ライブラリー名の間にはブランクが挿入されます。

(2) このコマンドにより、後続の機能での必要性に合わせてライブラリー・リストが変更されます。

(3) **変数変更 (CHGVAR)** コマンドは、CHGLIBL コマンドを変数 &CMD の値として組み立てます。

(4) QCMDEXC が呼び出されて変数 &CMD のコマンド・ストリングを処理します。 CALL コマンドでは結合を行うことができないので、QCMDEXC を呼び出す前に、CHGVAR コマンドが必要になります。

関連情報:

実行管理機能におけるジョブ記述

CL コマンド検索プログラム

現行ライブラリー変更 (CHGCURLIB) コマンド

ジョブ投入 (SBMJOB) コマンド

ライブラリー・リスト項目追加 (ADDLIBLE) コマンド

ライブラリー・リスト項目除去 (RMVLIBLE) コマンド

ライブラリー・リストの使用に関する考慮事項:

ライブラリー・リストをセットアップして使用する場合は、以下の情報を考慮してください。

- ライブラリー・リストに入れるライブラリーは、システムに存在しているものでなければなりません。IBM i オペレーティング・システムを開始すると、システム値 QSYSLIBL および QUSRLIBL へのアクセスが行われます。どちらかの値に入っているライブラリーが、システムに実在していない場合は、システム・オペレーター・メッセージ待ち行列 (QSYSOPR) にメッセージが送られて、そのライブラリーは無視され、IBM i オペレーティング・システムはそのライブラリーを除外して開始されます。IBM i オペレーティング・システムが開始された後では、活動ジョブのライブラリー・リストにあるライブラリーを削除することはできません。ご使用のジョブのライブラリー・リストにあるライブラリーは削除できません。別のジョブの場合でも、ライブラリー検索リストでライブラリーをロックするようにシステム値 QLIBLCKLVL が設定されていると、削除できません。ジョブ記述、**バッチ・ジョブ (BCHJOB)** コマンド、または**ジョブ投入 (SBMJOB)** コマンドに指定されたライブラリー・リストに入っているライブラリーの中に、実在しないか使用不能なライブラリーが含まれている場合は、そのジョブを開始することはできません。
- ライブラリー・リスト内のライブラリーは、それを使用する必要のあるすべてのユーザーに対して認可されていなければなりません。ライブラリー・リストを、例えば **SBMJOB** コマンド、**ジョブ (JOB)** コマンドまたは**ジョブ記述作成 (CRTJOBD)** コマンドで初期設定する場合には、ユーザーは、該当ライブラリーに対するオブジェクト操作権限を持っている必要があります。その権限がなければ、そのジョブを開始することはできません。また、**ライブラリー・リスト項目追加 (ADDLIBLE)** コマンドまたは**ライブラリー・リスト変更 (CHGLIBL)** コマンドを使用してライブラリー・リストに追加されるライブラリーに対して、 *USE 権限を持っていなければなりません。
- 借用したユーザー・プロファイルのもとで実行しているプログラムで、現行のユーザーに権限のないライブラリーをライブラリー・リストに追加し、そのライブラリーをプログラムの終了前にライブラリ

ー・リストから除去しなかった場合には、ユーザーはプログラムの終了後も、そのライブラリーに対してアクセス (*USE 権限) できることになります。ただし、この状態が起こるのは、オブジェクトへのアクセスに *LIBL が指定されている場合だけです。

- ライブラリー・リストのライブラリーの数をできるだけ少なくする方が、システムのパフォーマンスが向上します。

関連情報:

CL コマンド検索プログラム

ジョブ投入 (SBMJOB) コマンド

バッチ・ジョブ (BCHJOB) コマンド

ライブラリー・リスト項目追加 (ADDLIBLE) コマンド

ライブラリー・リスト変更 (CHGLIBL) コマンド

ライブラリー・リストの表示:

ライブラリー・リスト表示 (DSPLIBL) コマンドを使用して、現在実行中のジョブのライブラリー・リストを表示することができます。

画面には、ライブラリー・リストのすべてのライブラリーが、ライブラリー・リストにおける順序に従って表示されます。

活動状態のジョブのライブラリー・リストは、**ジョブ表示 (DSPJOB)** コマンドを使用し、「ジョブの表示」メニューでオプション 13 を選択することによっても表示することができます。

関連情報:

ジョブ表示 (DSPJOB) コマンド

ライブラリー・リスト表示 (DSPLIBL) コマンド

ライブラリー・リストを使用する、オブジェクトの検索:

ライブラリー・リストを使用して、複数のオブジェクトまたは 1 つのオブジェクトを検索できます。総称検索は複数のオブジェクトの検索に使用できます。

総称オブジェクト名検索:

総称検索を使用すると、複数のオブジェクトを検索することができます。

同じ文字列で始まるオブジェクト名を持つ複数のオブジェクトを見つけたい場合があります (実際に見つかるのは 1 つだけの場合もあります)。このような探索を総称検索 と呼び、多くのコマンドで使用することができます。

総称検索を使用する場合には、オブジェクト名の代わりに総称名をコマンドに指定します。総称名は、該当のすべてのオブジェクト名に共通する一連の文字 (オブジェクトのグループを識別するための文字群) と、その末尾にある 1 個のアスタリスクで構成されます。指定の文字群で始まる名前を持つオブジェクトで、ユーザーに権限のあるすべてのオブジェクトに対して、要求した機能が実行されます。例えば、総称名 ORD* を指定して**オブジェクト記述表示 (DSPOBJD)** コマンドを入力した場合には、その名前が ORD で始まるオブジェクト (複数の場合もある) のオブジェクト記述が表示されます。

次のように総称名にライブラリー修飾子を使用することにより、総称検索の範囲を限定することができます (指定可能なライブラリー名のパラメーター値は括弧に入れて示しています)。

- 指定のライブラリー。要求した操作は、指定したライブラリーに存在し、しかも総称名に該当するオブジェクトに対してだけ実行されます。
- ジョブのライブラリー・リスト (*LIBL)。ライブラリー・リストにおける順序に従って、各ライブラリーが探索されます。要求した操作は、ジョブのライブラリー・リストに指定されている各ライブラリー内の、総称名に該当する名前を持つオブジェクトに対して実行されます。
- ジョブの現行ライブラリー (*CURLIB)。ジョブの現行ライブラリーが探索されます。現行ライブラリーが存在していない場合には、QGPL が使用されます。
- ジョブのライブラリー・リストのユーザー部分にあるすべてのライブラリー (*USRLIBL)。現行ライブラリー (*CURLIB) も含め、該当のライブラリーが、ライブラリー・リストにおける順序に従って探索されます。要求した操作は、ジョブのライブラリー・リストのユーザー部分に指定されている各ライブラリー内の、総称名に該当する名前を持つオブジェクトに対して実行されます。
- ユーザーに権限のあるすべてのユーザー・ライブラリー (*ALLUSR) と、 Q の文字で始まるライブラリー。
- 該当の各ライブラリーが英数字順に探索されます。 *ALLUSR を指定した場合は、# で始まる S/36 環境のライブラリー、つまり #CGULIB、#COBLIB、 #DFULIB、#DSULIB、#RPGLIB、#SDALIB、および #SEULIB の各ライブラリーは探索されません。要求した操作は、ユーザーに権限のあるすべてのユーザー・ライブラリー内の、総称名に該当する名前をもつオブジェクトに対して実行されます。
- システム上の、ユーザーに権限のあるすべてのライブラリー (*ALL)。該当の各ライブラリーが英数字順に探索されます。要求した操作は、ユーザーに権限のあるシステムのすべてのライブラリー中の、総称名に該当する名前をもつオブジェクトに対して実行されます。

関連タスク:

478 ページの『ライブラリーの作成』
ライブラリーを作成する場合には、**ライブラリー作成 (CRTLIB)** コマンドを使用します。

510 ページの『オブジェクトの削除』
オブジェクトを削除するには、**オブジェクト削除 (DLTOBJ)** コマンド、そのオブジェクトのタイプに対応する削除 (DLTxxx) コマンド、あるいは「オブジェクトの処理」画面 (**「ライブラリーの処理 (WRKLIB)」**画面から表示される画面) の削除オプションを使用することができます。

関連情報:

総称ライブラリー名

複数のオブジェクトまたは *1* つのオブジェクトの探索*:*

総称名の指定が可能なすべてのコマンドで、1 つのオブジェクト名を (アスタリスクを付けずに) 指定して、複数のオブジェクトを探索することができます。

オブジェクト名を指定し、ライブラリー名として *ALL または *ALLUSR を指定すれば、システムでは複数のオブジェクトを見つけるための探索が行われ、指定の名前およびタイプを持ち、しかもユーザーに権限のある複数のオブジェクトが探索の結果として戻されます。総称名を指定するか、オブジェクト名を付けて *ALL、 *ALLUSR、またはライブラリーを指定する場合には、サポートされるすべてのオブジェクト・タイプ (すなわちオブジェクト・タイプ *ALL) を指定することができます。

ライブラリーの使用

ライブラリーは、関連するオブジェクトをグループ化し、名前によってオブジェクトを探索するために使用するオブジェクトです。したがって、ライブラリーはオブジェクトのグループの登録簿のようなものです。

ライブラリーを使用すると、以下の作業を実行することができます。

- ユーザー別にオブジェクトをグループ化する。これにより、システム上のオブジェクトの管理が容易になります。例えば、ユーザー JOE が使用できるファイルはすべてライブラリー JOELIB に入れることができます。
- アプリケーション別に、そのアプリケーションで使用するすべてのオブジェクトを 1 つのグループにする。例えば、受注関係のファイルおよびプログラムをすべて受注ライブラリー DISTLIB に入れることができます。ライブラリー・リストにライブラリーを 1 つ追加するだけで、受注関係のファイルおよびプログラムはすべてライブラリー・リストによって参照することができます。これは、受注ファイルまたは受注プログラムを使用する際に、ライブラリー名の指定を省きたい場合に大変便利です。
- セキュリティーの強化。例えば、どのユーザーにライブラリーの使用権限があり、そのユーザーはライブラリーにどのような操作を行うことが許されるかを指定することができます。
- 新しく作成したオブジェクトに対して、ライブラリーの CRTAUT パラメーターの値をもとに自動的に権限リストと共通認可を割り当てることにより、セキュリティーを簡略化することができます。新しく作成したオブジェクトの監査属性は、オブジェクト監査の作成 (CRTOBJAUD) パラメーター値に基づいて設定することができます。
- 同時に保管復元するオブジェクトを同じライブラリーにグループ化することにより、保管/復元操作を簡略化することができます。つまり、**オブジェクト保管 (SAVOBJ)** コマンドを使用してオブジェクトを個別に保管する代わりに、 **ライブラリー保管 (SAVLIB)** コマンドを使用することができます。
- 複数のライブラリーを使用してテストを行うことができます。
- 複数の実動ライブラリーの使用。例えば、ある実動ライブラリーはソース・ファイル用およびオブジェクトの作成用として使用し、別のライブラリーはアプリケーション・プログラムおよびファイル用として使用し、またもう 1 つはめったに保管しないオブジェクト用として使用し、さらにもう 1 つは頻繁に保管するオブジェクト用として使用するなどの使い方が可能です。

複数のライブラリーを使用すると、オブジェクトがさらに使用しやすくなります。例えば、同名の 2 つのファイルを別のライブラリーに入れておくことによって、片方をテストに使用し、もう一方を通常の処理に使用することができます。プログラムにライブラリー名を指定しない限り、テスト用か通常処理用かでプログラム内のファイル名を変更する必要はありません。どのライブラリーを使用するかは、ライブラリー・リストの使用により制御することができます。 (同じタイプのオブジェクトに同じ名前を付けることができるのは、それらのオブジェクトが異なるライブラリーに入っている場合だけです。)

ライブラリーには、実動ライブラリーとテスト・ライブラリーの 2 つのタイプがあります。実動ライブラリーは通常の処理に使用します。デバッグ・モードでは、実動ライブラリーのデータベース・ファイルが更新されないように保護することができます。デバッグ・モード間には、テスト・ライブラリーのファイルはどれも固有な指定をせずに更新することができます。

関連概念:

122 ページの『ライブラリー・オブジェクト』
ライブラリーとは、関連するオブジェクトをグループ化し、それらのオブジェクトの使用時に名前で見つけられるようにするために使用されるオブジェクトです。したがって、ライブラリーはオブジェクトのグループの登録簿のようなものです。

関連タスク:

441 ページの『オリジナル・プログラム・モデル・プログラムのデバッグ』
オリジナル・プログラム・モデル (OPM) プログラムをデバッグするには、テスト機能を使用します。これらの機能は、対話方式でもバッチ・ジョブの中でも使用できる 1 組のコマンドを介して実行することができます。

ライブラリーの作成

ライブラリーを作成する場合には、**ライブラリー作成 (CRTLIB)** コマンドを使用します。

例えば、次の例の **ライブラリー作成 (CRTLIB)** コマンドによって、受注業務のファイルやプログラムを入れるのに使用するライブラリーが作成されます。このライブラリーの名前は DISTLIB で、実動ライブラリーです。すべてのユーザーに対して与えられるデフォルトの権限では、ユーザーはこのライブラリーにアクセスすることができません。このライブラリーに作成されるオブジェクトは、CRTAUT パラメーターの値に基づいて *CHANGE の権限がデフォルトの共通認可として与えられます。

```
CRTLIB   LIB(DISTLIB) TYPE(*PROD) CRTAUT(*CHANGE) CRTOBJAUD(*USRPRF) +
         ASP(1) ASPDEV(*ASP) AUT(*EXCLUDE) TEXT('Distribution library')
```

文字 Q で始まる名前のライブラリーを作成してはなりません。総称検索の場合に、文字 Q で始まる名前のライブラリー (QRPG、QPDA など) の大半は、システム・ライブラリーと見なされます。

関連概念:

475 ページの『総称オブジェクト名検索』
総称検索を使用すると、複数のオブジェクトを検索することができます。

関連情報:

ライブラリー作成 (CRTLIB) コマンド

ライブラリー仕様の権限

特定の権限をライブラリーのユーザーに付与できます。

関連概念:

465 ページの『ライブラリー』
IBM i オペレーティング・システムでは、オブジェクトはグループ化されて、ライブラリーと呼ばれる特殊なオブジェクトに入れられます。

480 ページの『オブジェクトのセキュリティーに関する考慮事項』
システムは、ユーザーが参照したオブジェクトにアクセスする場合に、ユーザーが要求した方法でそのオブジェクトを使用するための権限を持っているかどうかを判断するための検査を行います。

関連情報:

セキュリティー参照

オブジェクト権限**:**

オブジェクト権限のタイプには、オブジェクト操作権限、オブジェクト管理権限、およびオブジェクト存在権限があります。

ライブラリーに対するオブジェクト操作権限を与えられたユーザーは、そのライブラリーの記述を表示することができます。

ライブラリーに対するオブジェクト管理権限を持つユーザーは次のことを行うことができます。

- 権限の認可および取り消し。権限を与えたり取り消したりすることができるのは、自分がもっている権限に限られます。 *ALLOBJ を持つオブジェクト所有者またはユーザーだけが、ライブラリーでのオブジェクト管理権限を与えることができます。
- ライブラリー名の変更。

オブジェクト存在権限および使用権限を持つユーザーは、ライブラリーの削除を行うことができます。

オブジェクト存在権限およびオブジェクト操作権限を持つユーザーは、ライブラリーの所有権の移転を行うことができます。

データ権限:

データ権限のタイプには、追加権限、読み取り権限、更新権限、実行権限、および削除権限があります。

ライブラリーに対する追加権限および読み取り権限により、ライブラリーに新しいオブジェクトを作成すること、およびオブジェクトをライブラリーに移動することが可能になります。

ライブラリーでの更新権限および実行権限があれば、ユーザーはライブラリーにあるオブジェクトの名前を変更することができます (ユーザーがそのオブジェクトへの権限も持っている場合)。

削除権限により、ユーザーはオブジェクトから項目を削除することができます。ライブラリーの削除権限を持っていても、ライブラリー内のオブジェクトを削除することはできません。オブジェクトを削除できるかどうかは、ライブラリー中のオブジェクトに対する権限によって決まります。

実行権限により、ユーザーはライブラリー中でオブジェクトを探索することができます。

複合の権限:

複合の権限のタイプには、*USE 権限、*CHANGE 権限、*ALL 権限、および *EXCLUDE 権限があります。

ライブラリーに対する *USE 権限 (オブジェクト操作権限、読み取り権限、および実行権限から構成される権限) を持つユーザーは、以下を実行することができます。

- ライブラリーを使用してオブジェクトを探索する。
- ライブラリーの内容を表示する。
- ライブラリーをライブラリー・リストに入れる。
- ライブラリーを保管する (必要な権限をそのオブジェクトに対して持っている場合)。
- ライブラリーからオブジェクトを削除する (ユーザーがそのライブラリーの該当オブジェクトに対する権限を持っている場合)。

ライブラリーに対する *CHANGE 権限 (オブジェクト操作権限とライブラリーに対するすべてのデータ権限から構成される権限) を持つユーザーは、以下を実行することができます。

- ライブラリーを使用してオブジェクトを探索する。
- ライブラリーの内容を表示する。
- ライブラリーをライブラリー・リストに入れる。
- ライブラリーを保管する (必要な権限をそのオブジェクトに対して持っている場合)。
- ライブラリーからオブジェクトを削除する (ユーザーがそのライブラリーの該当オブジェクトに対する権限を持っている場合)。
- オブジェクトをライブラリーに追加する。

*ALL 権限は、すべてのオブジェクト権限とすべてのデータ権限を与えます。 *ALL 権限を持つユーザーは、ライブラリーの削除、ライブラリーのセキュリティーの指定、ライブラリーの変更、およびライブラリーの記述と内容の表示を行うことができます。

*EXCLUDE 権限は、ユーザーがオブジェクトにアクセスできないようにします。

ユーザーの使用しているライブラリーに関連する権限を表示するときは、 **オブジェクト権限表示 (DSPOBJAUT)** コマンドを使用することができます。

オブジェクトのセキュリティーに関する考慮事項

システムは、ユーザーが参照したオブジェクトにアクセスする場合に、ユーザーが要求した方法でそのオブジェクトを使用するための権限を持っているかどうかを判断するための検査を行います。

ユーザーは通常、2 つのレベルの権限を持っていなければなりません。

- 要求している機能の実行対象となるオブジェクトを使用するための権限を持っていなければなりません。
- オブジェクトが入っているライブラリーに対する権限を持っていなければなりません。ライブラリー・リストを使用する場合には、リスト内のライブラリーに対する権限を持っていなければなりません。

オブジェクト権限は次に示すように、システムのセキュリティー機能により制御されます。

- オブジェクトの所有者および *ALLOBJ 特殊権限を持つユーザーは、オブジェクトに対するすべての権限を持ち、オブジェクトについての権限の認可および取り消しを他のユーザーに対して行うことができます。
- ユーザーは、オブジェクトに対する私用認可を与えられない場合には、共通認可を持ちます。

セキュリティーの対象となるプログラム (機密保護担当者のユーザー・プロファイルを借用するプログラムなど) を作成する場合には、さらに特殊な考慮事項があります。

関連概念:

465 ページの『ライブラリー』
IBM i オペレーティング・システムでは、オブジェクトはグループ化されて、ライブラリーと呼ばれる特殊なオブジェクトに入れられます。

関連タスク:

478 ページの『ライブラリー仕様の権限』
特定の権限をライブラリーのユーザーに付与できます。

関連情報:

セキュリティー参照

監査ジャーナル項目の表示コマンドを使用することにより、セキュリティー・ジャーナル監査報告書を生成することができます。:

監査ジャーナル項目の表示 (DSPAUDJRNE) コマンドを使用することにより、セキュリティー・ジャーナル監査報告書を生成することができます。

監査ジャーナル項目の表示 (DSPAUDJRNE) コマンドにより生成された報告書は、コマンドで指定した監査項目タイプおよびユーザー・プロファイルに基づいています。報告書を特定の時間枠に制限したり、切り離されたジャーナル・レシーバーを検索することができます。これらの報告書は、活動状態の画面または出力待ち行列に送ることができます。

制約事項: このコマンドを使用するには、 *ALLOBJ および *AUDIT 特殊権限が必要です。

関連情報:

監査ジャーナル項目表示 (DSPAUDJRNE) コマンド

デフォルトの共通認可の設定

ライブラリーにオブジェクトを作成する場合、そのライブラリーの CRTAUT 値を使用して、そのオブジェクトに対するデフォルトの共通認可を設定することができます。

以下のコマンドを指定します。

```
CRTLIB LIB(TESTLIB) CRTAUT(*USE) AUT(*LIBCRTAUT)
```

このコマンドにより、ライブラリー TESTLIB が作成されます。ライブラリー TESTLIB に作成されるオブジェクトは、すべてデフォルト値により *USE 権限を共通認可として持ちます。ライブラリー TESTLIB に対する共通認可は、ライブラリー QSYS の CRTAUT の値によって決まります。

以下のコマンドを指定します。

```
CRTDTAARA DTAARA(TESTLIB/DTA1) TYPE(*CHAR) +
          AUT(*LIBCRTAUT)

CRTDTAARA DTAARA(TESTLIB/DTA2) TYPE(*CHAR) +
          AUT(*EXCLUDE)
```

これにより、ライブラリー TESTLIB にデータ域 DTA1 が作成されます。 DTA1 の共通認可は、ライブラリー TESTLIB の CRTAUT の値に基づき、*USE になります。

また、ライブラリー TESTLIB にデータ域 DTA2 が作成されます。 DTA2 の共通認可は *EXCLUDE です。 *EXCLUDE は**データ域作成 (CRTDTAARA)** コマンドの CRTDTAARA パラメーターに指定されています。

ライブラリーにオブジェクトを作成する場合、権限リストを使用してオブジェクトを保護することもできます。

以下のコマンドを指定します。

```
CRTAUTL AUTL(PAYROLL)
CRTLIB  LIB(PAYLIB) CRTAUT(PAYROLL) +
        AUT(*EXCLUDE)
```

PAYROLL という権限リストが作成されます。ライブラリー PAYLIB は *EXCLUDE を共通認可として作成されます。デフォルト値により、ライブラリー PAYLIB に作成されるオブジェクトは、権限リスト PAYROLL で保護されます。

以下のコマンドを指定します。

```
CRTPF FILE(PAYLIB/PAYFILE) +
      AUT(*LIBCRTAUT)

CRTPF FILE(PAYLIB/PAYACC) +
      AUT(*CHANGE)
```

ファイル PAYFILE がライブラリー PAYLIB に作成されます。 PAYFILE は権限リスト PAYROLL によって保護されます。 PAYFILE の共通認可は、**物理ファイル作成 (CRTPF)** コマンドによって *AUTL に設定されます。 *AUTL は、PAYFILE に対する共通認可が、PAYFILE を保護する権限リスト、すなわち権限リスト PAYROLL からとられることを意味します。

また、ライブラリー PAYLIB にファイル PAYACC が作成されます。ファイル PAYACC に対する共通認可は、**物理ファイル作成 (CRTPF)** コマンドの AUT パラメーターで指定されているので、この共通認可は *CHANGE になります。

注**:** 多くの CRT (作成) コマンドに存在する AUT パラメーターの *LIBCRTAUT の値は、オブジェクトの共通認可をそのオブジェクトが作成されるライブラリーの CRTAUT の値に設定することを意味します。

ライブラリーの CRTAUT の値は、そのライブラリーに作成されるオブジェクトの共通使用に対するデフォルトの権限を指定します。指定できる値は次のとおりです。

***ALL** すべての共通認可

***CHANGE**
変更権限

***EXCLUDE**
排他権限

***SYSVAL**
作成されるオブジェクトの共通認可は、システム値 QCRTAUT に指定された値になります。

***USE** 使用権限

権限リスト名
指定の権限リストによってオブジェクトが保護されます。

デフォルトの監査属性の設定

ライブラリーにオブジェクトを作成する場合、そのオブジェクトの CRTOBJAUD 値を使用して、そのオブジェクトに対するデフォルトの監査属性を設定することができます。

以下のコマンドを指定して、PAYROLL ライブラリーに作成されたオブジェクトがすべて、読み取りと変更の両方のアクセスに関して監査を受けます。

```
CRTLIB LIB(PAYROLL) AUT(*EXCLUDE) CRTAUT(*EXCLUDE) CRTOBJAUD(*ALL)
```

関連情報:

セキュリティー参照

ライブラリーへのオブジェクトの収容

オブジェクトを作成する場合、そのオブジェクトはライブラリーに入れられます。

ライブラリーを指定しなかった場合には、オブジェクトはジョブの現行ライブラリー (*CURLIB)、またはジョブの現行ライブラリーの指定がない場合には QGPL に入れられます。ライブラリーを作成する場合、**ライブラリー作成 (CRTLIB)** コマンドの CRTAUT パラメーターを使用して共通認可を指定することができます。ライブラリーに作成されるオブジェクトは、すべてそのライブラリーの CRTAUT 値に指定された共通認可が想定されます。ライブラリーを指定するには、修飾名、すなわちライブラリー名とオブジェクト名を指定してください。例えば、次の**物理ファイル作成 (CRTPF)** コマンドによって、受注物理ファイル ORDHDRP が作成され、DISTLIB に入れられます。

```
CRTPF FILE(DISTLIB/ORDHDRP)
```

オブジェクトをライブラリーに入れるためには、そのライブラリーに対する読み取り権限および追加権限を持っていなければなりません。

同じタイプの複数のオブジェクトを同じ名前で、同じライブラリーに入れることはできません。例えば、ORDHDRP という名前の 2 つのファイルを両方ともライブラリー DISTLIB に入れることはできません。既にライブラリーに入っているオブジェクトと同じ名前で同じタイプのオブジェクトをライブラリーに入れようとすると、その要求はシステムにより拒否され、理由を示すメッセージが表示されます。

注**:** QSYS ライブラリーは、システム・オブジェクトに対してだけ使用してください。 IBM i オペレーティング・システムの新しいリリースの導入時には変更は失われるので、他のライセンス・プログラムを QSYS ライブラリーに復元してはなりません。

関連情報:

ライブラリー作成 (CRTLIB) コマンド

物理ファイル作成 (CRTPF) コマンド

ライブラリーの削除およびクリア

ライブラリー削除 (DLTLIB) コマンドを使用してライブラリーを削除した場合には、ライブラリー自体とともにライブラリー内のオブジェクトも削除されます。

ライブラリー消去 (CLRLIB) コマンドを使用してライブラリーを消去した場合には、ライブラリー内のオブジェクトは削除されますが、ライブラリーは削除されません。ライブラリーを削除または消去する場合に必要なことは、ライブラリー名を指定することだけです。以下にその例を示します。

```
DLTLIB LIB(DISTLIB)
CLRLIB LIB(DISTLIB)
```

ライブラリーを削除するには、ライブラリーとそのライブラリー内のオブジェクトの両方に対するオブジェクト存在権限、およびライブラリーに対する使用権限を持っていなければなりません。ライブラリーを削除しようとして、そのライブラリー内の一部のオブジェクトに対するオブジェクト存在権限を持っていなかった場合には、そのライブラリー自体、およびユーザーに権限のないオブジェクトは削除されません。ユーザーが権限を持つオブジェクトはすべて削除されます。ライブラリーを削除しようとして、そのライブラリーに対するオブジェクト存在権限を持っていなかった場合には、そのライブラリーは削除されないばかりでなく、ライブラリー内のオブジェクトもまったく削除されません。特定のオブジェクト (それに対するオブジェクト存在権限を持っているもの) を削除したい場合には、 **プログラム削除 (DLTPGM)** コマンドなど、該当のオブジェクト・タイプに対する削除コマンドを使用することができます。

活動状態のジョブのライブラリー・リストに含まれているライブラリーを削除することはできません。そのようなライブラリーを削除したい場合には、ジョブ・ステップが終了するまで待たなければなりません。したがって、次の経路指定ステップが開始される前に、ライブラリーを削除しなければなりません。ライブラリーを削除する場合には、そのライブラリーまたはライブラリー内のオブジェクトを必要としているユーザーが存在しないことを確認しなければなりません。

システム値 QSYSLIBL および QUSRLIBL により定義された初期ライブラリー・リストに含まれているライブラリーの場合には、そのライブラリーを削除するためには、次のステップに従う必要があります。

1. **システム値変更 (CHGSYSVAL)** コマンドを使用して、該当するシステム値を変更してライブラリーを除去してください。 (システム値変更コマンドは、実行中のジョブのライブラリー・リストには影響を与えません。)
2. **ライブラリー・リスト変更 (CHGLIBL)** コマンドを使用して、ジョブのライブラリー・リストを変更してください。

   システム・ライブラリー・リスト変更 (CHGSYSLIBL) コマンド、 **ライブラリー・リスト項目追加 (ADDLIBLE)** コマンド、 **ライブラリー・リスト編集 (EDTLIBL)** コマンド、および**ライブラリー・リスト項目除去 (RMVLIBLE)** コマンドも、ライブラリー・リストを変更するために使用することができます。
3. **ライブラリー削除 (DLTLIB)** コマンドを使用して、ライブラリーとそのライブラリー内のオブジェクトを削除してください。

注**:** ライブラリー QSYS を削除することはできません。また、その中のオブジェクトを削除してはなりません。システムが正常に働くためには QSYS に入っているオブジェクトが必要なので、QSYS 中のオブジェクトを削除すると、システムが正常に作動しなくなることがあります。また、ライブラリー QGPL にも、システムが処理を効率的に行うために必要なオブジェクトがいくつか含まれているので、QGPL も削除してはなりません。ライブラリー QRECOVERY は、システムによる内部使用のためのものなので、ユーザーはこのライブラリーを使用してはなりません。ライブラリー QRECOVERY には、システムの正常な操作に必要なオブジェクトが入っています。

ライブラリーを消去するには、ライブラリー内のオブジェクトに対するオブジェクト存在権限、およびライブラリーに対する使用権限を持っていなければなりません。ライブラリーを消去しようとして、そのライブラリー内の一部のオブジェクトに対するオブジェクト存在権限を持っていなかった場合には、権限のないオブジェクトはライブラリーから削除されません。また、他のユーザーに割り振られているオブジェクトがある場合には、そのオブジェクトも削除されません。

関連タスク:

510 ページの『オブジェクトの削除』

オブジェクトを削除するには、**オブジェクト削除 (DLTOBJ)** コマンド、そのオブジェクトのタイプに対応する削除 (DLTxxx) コマンド、あるいは「オブジェクトの処理」画面 (**「ライブラリーの処理 (WRKLIB)」**画面から表示される画面) の削除オプションを使用することができます。

関連情報:

CL コマンド検索プログラム

ライブラリー消去 (CLRLIB) コマンド

ライブラリー削除 (DLTLIB) コマンド

ライブラリー・リスト変更 (CHGLIBL) コマンド

システム値変更 (CHGSYSVAL) コマンド

ライブラリー・リスト項目除去 (RMVLIBLE) コマンド

ライブラリー名およびその内容の表示

ライブラリー表示 (DSPLIB) コマンドまたは**ライブラリー処理 (WRKLIB)** コマンドを使用して、ユーザーが権限を持っているすべてのライブラリーを表示または印刷し、該当するライブラリー内の各オブジェクトについての基本情報を表示することができます。

これにより得られるオブジェクトの基本情報は次のとおりです。

- オブジェクトの名前およびタイプ
- オブジェクトの属性
- オブジェクトのサイズ
- オブジェクトの作成時に指定されたテキスト

ライブラリー表示 (DSPLIB) コマンドでは、1 つまたは複数の特定のライブラリー名を指定することができるので、その場合にはライブラリー選択画面を無視します。このリストでは、オブジェクトはライブラリー別に、各ライブラリー内ではオブジェクト・タイプ別に分類され、各オブジェクト・タイプ内では英数字順に示されます。ライブラリーの順序は、次のいずれかになります。

- **ライブラリー表示 (DSPLIB)** コマンドに複数のライブラリー名を指定した場合、それらのライブラリーは、表示コマンドで指定されている順序に従って表示されます。
- **ライブラリー表示 (DSPLIB)** コマンドに *LIBL または *USRLIBL を指定した場合、表示されるライブラリーの順序は、ジョブのライブラリー・リストにおけるライブラリーの順序と同じになります。

- **ライブラリー表示 (DSPLIB)** コマンドに *ALL または *ALLUSR を指定した場合、ライブラリーは英数字順に表示されます。ユーザーは、表示するライブラリーに対する読み取り権限を持っていなくてはなりません。

例えば、次の**ライブラリー表示 (DSPLIB)** コマンドは、DISTLIB に入っているオブジェクトのリストを表示します。

```
DSPLIB  LIB(DISTLIB)  OUTPUT(*)
```

OUTPUT パラメーターのアスタリスク (*) は、ライブラリーを、対話式処理の場合にはディスプレイ装置に表示し、バッチ処理の場合には印刷することを示します。対話式処理でリストを印刷したい場合には、デフォルトの * の代わりに *PRINT を指定してください。

関連情報:

ライブラリー表示 (DSPLIB) コマンド

ライブラリー処理 (WRKLIB) コマンド

ライブラリー記述の表示および検索

ライブラリー記述表示 (DSPLIBD) コマンドおよび**ライブラリー記述の検索 (RTVLIBD)** コマンドを使用して、ライブラリー記述の表示や検索を行うことができます。

ライブラリー記述情報には以下が含まれます。

- ライブラリーのタイプ (PROD または TEST のいずれか)
- ライブラリーの補助記憶域プール番号
- ライブラリーの補助記憶域プール装置名
- ライブラリーの補助記憶域プール・グループ装置名
- ライブラリーの作成権限
- ライブラリーのオブジェクト監査の作成
- ライブラリーのテキスト記述
- ライブラリーの現在のジャーナル状況
- ライブラリーがジャーナル処理された現在または最後のジャーナルの名前
- ジャーナルを含むライブラリーの名前
- ジャーナル・レシーバーに書き込まれたイメージ
- 除外されたジャーナル項目
- 新規オブジェクトがジャーナルを継承するかどうか
- ジャーナル継承規則が一時変更されるかどうか
- ジャーナルの最終開始日時
- 最も古いジャーナル・レシーバーの名前
- 開始ジャーナル・レシーバーを含むライブラリーの名前
- 開始ジャーナル・レシーバーの補助記憶域プール (ASP) 装置
- 開始ジャーナル・レシーバーの ASP グループ

関連情報:

ライブラリー記述表示 (DSPLIBD) コマンド

ライブラリー記述検索 (RTVLIBD) コマンド

各国語バージョンの変更

IBM i ライセンス・プログラムでは、同じシステムで異なる言語を使用することができます。このサポートにより、あるユーザーにはある言語で情報を提供し、別のユーザーには別の言語で情報を提供することができます。

各国語バージョンを変更するには、静的プロンプト・メッセージまたは動的プロンプト・メッセージのいずれかを使用できます。

制御言語の静的なプロンプト・メッセージ

静的なプロンプト・メッセージは、コマンド定義オブジェクトに保管されます。ライブラリー・リストに各国語のライブラリーを追加すると、異なる各国語バージョンの情報が表示されます。

静的なプロンプト・メッセージを使用すると、ユーザー可読情報 (表示画面、メッセージ、印刷出力、およびオンライン・ヘルプ情報) に使用する言語は、ジョブのライブラリー・リストにより制御されます。ライブラリー・リストのシステム部分に言語ライブラリーを追加することによって、情報を別の言語バージョンで表示または印刷することができます。 1 次言語では、各ライセンス・プログラムの実行コードとテキスト・データが該当の各国語バージョンになります。 2 次言語では、すべてのライセンス・プログラムのテキスト・データが該当の各国語バージョンになります。

システムの 1 次言語についての言語情報は、 IBM ライセンス・プログラム用のライブラリーと同じライブラリーに保管されます。例えば、システムの 1 次言語が英語の場合には、QSYS、QHLPSYS、および QSSP などのライブラリーには、英語による情報が保管されます。ライブラリー QSYS および QHLPSYS は、ライブラリー・リストのシステム部分に入っています。他のライセンス・プログラム用のライブラリー (ILE RPG for IBM i の QRPGLE など) は、必要になった時点でシステムによりライブラリー・リストに追加されます。

システムの 1 次言語以外の各国語バージョンは、2 次言語ライブラリーに導入されます。各 2 次言語ライブラリーには、すべての IBM ライセンス・プログラムの表示画面、メッセージ、コマンド・プロンプト、およびヘルプについての 1 つの各国語バージョンが入っています。 2 次言語ライブラリーの名前の形式は、QSYSnnnn (nnnn は言語機能コード) です。例えば、フランス語の機能コードは 2928 なので、フランス語の場合の 2 次言語ライブラリー名は QSYS2928 となります。

情報をシステムの 1 次言語で表示したい場合には、特別の操作は必要ではありません。システムの 1 次言語とは別の言語で情報が提供されるようにしたい場合には、ライブラリー・リストを変更して、必要な言語の各国語ライブラリーが、ライブラリー・リスト上で各国語情報を含む他のすべてのライブラリーよりも前に置かれるようにしなければなりません。次のオプションのいずれかを使用して、必要な各国語ライブラリーが先頭に置かれるようにすることができます。

- CRTSBSD または CHGSBSD の SYSLIBLE パラメーターを指定して、画面、メッセージなどが特定の言語で表示されるようにすることができます。以下にその例を示します。

  ```
  CRTSBSD SBSD(QSBSD 2928) POOLS((1 *NOTSG)) SYSLIBLE(QSYS2928)
  ```

- CHGSYSLIBL コマンドで LIB パラメーターを使用して、必要な各国語ライブラリーが、ライブラリー・リストの先頭にくるよう指定することができます。以下にその例を示します。

  ```
  CHGSYSLIBL  LIB(QSYS2928)
  ```

- 対話式ジョブのライブラリー・リストの先頭に、必要な各国語ライブラリーを指定するように、ユーザー・プロファイルの初期プログラムをセットアップすることができます。これは、ユーザーがサインオンのたびに CHGSYSLIBL コマンドを実行したくない場合に便利なオプションです。このためには、初期プログラムにシステム・ライブラリー・リスト変更 (CHGSYSLIBL) コマンドを指定して、必要な各国語ライブラリーをライブラリー・リストの先頭に追加します。

注: システムの出荷時に CHGSYSLIBL コマンドに対して設定されている権限では、このコマンドを実行する権限がすべてのユーザーに与えられているわけではありません。

ユーザーに CHGSYSLIBL コマンドに対する権限を認可せずに、そのユーザーがこのコマンドを実行できるようにするためには、CHGSYSLIBL コマンドを組み込んだ CL プログラムを作成することができます。このプログラムの所有者は機密保護担当者であり、作成された後は、プログラムの使用者が機密保護担当者の権限を借用することになります。このプログラムを実行する権限を持つユーザーは、このプログラムを使用してユーザーのジョブのライブラリー・リストのシステム部分を変更することができます。ライブラリー・リストをフランス語ユーザー用に設定するためのプログラムの例を次に示します。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
  CHGSYSLIBL  LIB(QSYS2928)  /* Use French information */
ENDPGM
```

制御言語の動的プロンプト・メッセージ

コマンド (*CMD) が作成されたときに、そのコマンドのオブジェクトに保管されるメッセージ識別コードを使用して、メッセージ・ファイルからプロンプト・メッセージを動的に検索することができます。この機能により、単一のコマンドに複数の各国語で表記されたプロンプト・メッセージを含めることができます。

動的プロンプト・メッセージを使用するには、**コマンド作成 (CRTCMD)** コマンドを使用してコマンド定義オブジェクトを作成し、*CMD オブジェクトで PROMPT パラメーターおよび CHOICE パラメーターのメッセージ識別コードを指定して、プロンプト・メッセージ・ファイル (PMTFILE) パラメーターの最初の項目にメッセージ・ファイルを指定して、2 番目の項目に *DYNAMIC を指定する必要があります。プロンプト時にライブラリー・リストで必要な各国語でプロンプト・メッセージ・ファイルのコピーを作成すると、同じコマンドを任意の各国語でプロンプト表示させることができます。

CMD、PARM、QUAL、または ELEM コマンド定義ステートメントの PROMPT パラメーターまたは CHOICE パラメーターに対して指定されたメッセージ識別コードは、コマンドが作成されるときと、コマンドがプロンプト表示されるときの両方において、プロンプト・メッセージ・ファイルに存在する必要があります。

コマンドがプロンプト表示されたときに、メッセージ・ファイル検索中のエラーが生じた場合、すべてのプロンプト・テキストは *CMD オブジェクトに保管されたプロンプト・メッセージの静的コピーから取得されます。メッセージ・ファイルはあるが、メッセージ・ファイルに個々のプロンプト・テキストがない場合、*CMD オブジェクトに保管されているプロンプト・テキストの静的コピーがメッセージに使用されます。

関連タスク:

537 ページの『メッセージ記述の定義』
事前定義メッセージは、メッセージ・ファイルに保管されます。

320 ページの『CL コマンドの定義』
コマンドを作成するには、まずコマンド定義ステートメントによりそのコマンドを定義しなければなりません。

関連情報:

コマンド作成 (CRTCMD) コマンド

オブジェクトの記述

オブジェクトを作成する場合は常に、作成コマンドの TEXT パラメーターの 50 文字のフィールドで、そのオブジェクトを記述できます。

一部のコマンドではデフォルト値 *SRCMBRTXT を使用することができます。これは、作成するオブジェクトのテキストを、そのオブジェクトの作成に使用するソース・メンバーのテキストからとることを示す値です。ただし、このデフォルト値が使用できるのは、データベース・ソース・ファイルに入っているソースから作成されるオブジェクトの場合だけです。

作成コマンドのソース入力が装置ファイルまたはインライン・ファイルの場合、またはソースを使用しない場合には、テキストのデフォルト値はブランクになります。指定したテキストはオブジェクト記述の一部として、オブジェクト記述表示 (DSPOBJD) コマンドまたはライブラリー表示 (DSPLIB) コマンドを使用して表示することができます。このテキストは、オブジェクト記述変更 (CHGOBJD) コマンド、または各オブジェクト・タイプに対応する多くの変更 (CHGxxx) コマンドを使用して変更することができます。

オブジェクト記述の表示

オブジェクト記述表示 (DSPOBJD) コマンドまたは**オブジェクト処理 (WRKOBJ)** コマンドを使用して、オブジェクトの記述を表示することができます。

オブジェクト記述は、システムに存在しているが不要なオブジェクトを判別するのに役立ちます。バッチ処理で実行する場合には、オブジェクト記述を印刷したり、データベース・ファイルへ書き込んだりすることができます。対話式処理で実行する場合には、オブジェクト記述の表示、印刷、およびデータベース・ファイルへの書き出しが可能です。

オブジェクト記述として、基本属性、明細属性、および保守属性を表示することができます。表示されるオブジェクト記述は次のとおりです。

表 *28.* オブジェクト記述として表示される属性

基本属性	明細属性	保守属性 (注を参照)
• オブジェクト名 • ライブラリー名 • ライブラリー ASP 装置 • ライブラリー ASP グループ装置 • オブジェクト・タイプ • 拡張属性 • オブジェクト・サイズ • テキスト記述 (一部)	• オブジェクト名 • ライブラリー名 • ライブラリー ASP 装置 • オブジェクト・タイプ • 所有者 • ライブラリー ASP グループ装置 • 1 次グループ • 拡張属性 • ユーザー定義属性 • テキスト記述 • 作成日と作成時刻 • オブジェクト作成者 • オブジェクトを作成したシステム • オブジェクト定義域 • 変更日時 • 使用状況データ収集の有無 • 最終使用日付 • 使用日数カウント • 使用カウントのリセット日付 • プログラムによる変更の可否 • オブジェクト監査値 • デジタル署名 • デジタル署名システム・トラステッド・ソース • デジタル署名複数署名 • 権限収集値 • オブジェクト・サイズ • オフライン・サイズ • 関連したスペースのサイズ • 最適なスペース位置合わせ • 解放状況 • 圧縮状況 • オブジェクト ASP 番号 • オブジェクト・オーバーフロー • オブジェクト ASP 装置 • オブジェクト ASP グループ装置 • ジャーナル状況 • 現行または最終ジャーナル • ジャーナル・イメージ • 除外するジャーナル項目 • リモート・ジャーナル・フィルター • ジャーナル開始日時 • 適用するジャーナル・レシーバー開始 • ジャーナルを開始するためのライブラリー名 • ジャーナルを開始するためのライブラリー ASP 装置 • ジャーナルを開始するためのライブラリー ASP グループ装置 • 使用日時の保管 • 使用日時の復元 • 保管コマンド • 装置タイプ • 順序番号 • ファイル・ラベル ID • 保管形式	• オブジェクト名 • ライブラリー名 • ライブラリー ASP 装置 • ライブラリー ASP グループ装置 • オブジェクト・タイプ • ソース・ファイルおよびソース・ライブラリー • メンバー名 • 拡張属性 • ユーザー定義属性 • 解放状況 • オブジェクト・サイズ • 作成日と作成時刻 • ソース・ファイルのメンバーの最終更新の日付と時刻 • システム・レベル • コンパイラー • オブジェクト制御レベル • プログラムによる変更 • ユーザーによる変更の有無 • ライセンス・プログラム • PTF 番号 • APAR ID • オブジェクトのテキスト記述、またはオブジェクト状況条件

注:

1. 保守情報は、オブジェクトを作成したシステムのレベル、およびオブジェクトが出荷後に変更されたかを判別するために、プログラミング・サポート担当者が使用するものです。この情報の一部には、オブジェクトの作成に使用されたソース・メンバーや、オブジェクトの作成のもととなったソースが最後に変更された日付が示されているので、ユーザーにとっても役に立つことがあります。
2. ライブラリー・オブジェクトには、ライブラリーに含まれているオブジェクトの名前 だけが入っています。オブジェクト・タイプ *LIB に対して DSPOBJD を使用した場合には、オブジェクト・サイズ情報に示されるのはライブラリー・オブジェクトのサイズだけで、ライブラリーに含まれているオブジェクトの合計サイズではありません。

   ライブラリー記述の検索 (QLIRLIBD) API または DSPLIB OUTPUT(*PRINT) コマンドを使用することによって、ライブラリーの合計サイズを出すことができます。

オブジェクト記述表示 (DSPOBJD) コマンドまたは **オブジェクト処理 (WRKOBJ)** コマンドを使用することにより、ユーザーが権限を持つライブラリー内のオブジェクトを次の基準でリストすることができます。

- 名前
- 総称名
- タイプ
- オブジェクト・タイプ内の名前または総称名

オブジェクトは、ライブラリー別にリストされ、1 つのライブラリーの中ではタイプ別にリストされます。1 つのオブジェクト・タイプの中では、オブジェクトは英数字順にリストされます。

*FULL オプションまたは *SERVICE オプションを指定して多くのオブジェクトを表示したい場合には、バッチ・ジョブで **オブジェクト記述表示 (DSPOBJD)** コマンドを使用することもできます。出力は、ディスプレイ装置で表示する代わりに、スプール印刷ファイルに出力して印刷するか、データベース・ファイルに出力することができます。出力をデータベース・ファイルに書き込む場合、オブジェクトのすべての属性がファイルに書き込まれます。このファイルのレコード様式を表示したい場合には、 **ファイル・フィールド記述表示 (DSPFFD)** コマンドに、ライブラリー QSYS のファイル QADSPOBJ を指定して入力してください。

次のコマンドによって、ORD で始まる名前を持つ受注ファイル (つまり、DISTLIB 中のファイル) の記述が表示されます。 ORD* は総称名です。

```
DSPOBJD   OBJ(DISTLIB/ORD*)  OBJTYPE(*FILE) +
          DETAIL(*BASIC)  OUTPUT(*)
```

この結果表示される基本情報画面は次のとおりです。

```
                    オブジェクト記述の表示 - 基本リスト
                                                        ライブラリー 1 の 1
ライブラリー . . . . :    DISTLIB     ライブラリー ASP 装置. . . . :    *SYSBAS
                                      ライブラリー ASP グループ  . :    *SYSBAS

オプションを入力して，実行キーを押してください。
  5= 全属性の表示   8= 保守属性の表示

OPT  オブジェクト    タイプ     属性        サイズ    テキスト
 _    ORDDTLP        *FILE      PF          8192     発注明細
 _    ORDHDRP        *FILE      PF          8192     発注見出し

                                                                  終わり
F3= 終了   F12= 取り消し   F17= 先頭   F18= 最後
```

*BASIC の代わりに *FULL を指定するか、または基本情報画面で ORDDTLP の前の OPT 欄に 5 を入力すると、次の明細情報画面が表示されます。

```
                    オブジェクト記述の表示 - 明細
                                                      ライブラリー 1 の 1
オブジェクト . . . . . :   ORDDTLP        属性  . . . . . :   PF
  ライブラリー . . . . . :     DISTLIB      所有者  . . . . . :   QSECOFR
ライブラリー ASP 装置  . . :   *SYSBAS       ライブラリー ASP グループ . :   *SYSBAS
タイプ . . . . . :   *FILE          1 次グループ  . . . . . :   *NONE

ユーザー定義の情報 :
  属性  . . . . . :
  テキスト  . . . . . :

作成情報 :
  作成日/時刻 . . . . . :   06/08/05  10:17:03
  作成ユーザー  . . . . . :   QSECOFR
  作成されたシステム. . . . . :   SYSTEM01
  オブジェクト定義域  . . . . . :   *SYSTEM

                                                                  続く...
続行するには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

```
                    オブジェクト記述の表示 - 明細
                                                      ライブラリー 1 の 1
オブジェクト . . . . . :   ORDDTLP        属性  . . . . . :   PF
  ライブラリー . . . . . :     DISTLIB      所有者  . . . . . :   QSECOFR
ライブラリー ASP 装置  . . :   *SYSBAS       ライブラリー ASP グループ . :   *SYSBAS
タイプ . . . . . :   *FILE          1 次グループ  . . . . . :   *NONE

変更／使用状況情報 :
  変更日付/時刻  . . . . . :   05/11/90  10:03:02
  使用状況データの収集 . . . . . :   YES
  最終使用日 . . . . . :   05/11/90
  使用日数 . . . . . :   20
    リセット日 . . . . . :   03/10/90
  プログラムによる変更可能 . . . . . :   YES
監査／保全性情報 :
  オブジェクト監査値 . . . . . :   *NONE
  デジタル署名 . . . . . :   NO

                                                                  続く...
続行するには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

```
                    オブジェクト記述の表示 - 明細
                                                        ライブラリー 1 の 1
オブジェクト . . . . . :   ORDDTLP         属性 . . . . . :   PF
  ライブラリー . . . . . :     DISTLIB      所有者 . . . . . :   QSECOFR
ライブラリー ASP 装置 . . :   *SYSBAS       ライブラリー ASP グループ . :   *SYSBAS
タイプ . . . . . :   *FILE          1 次グループ . . . . . :   *NONE

記憶域情報 :
  サイズ . . . . . :   32768
  オフラインのサイズ . . . . . :   0
  関連したスペース・サイズ . . . . . :   3840
  最適スペース位置合わせ . . . . . :   NO
  解放 . . . . . :   NO
  圧縮 . . . . . :   INELIGIBLE
  オブジェクト ASP 番号 . . . . . :   1
    オブジェクトのオーバーフロー . . . . . :   NO
  オブジェクト ASP 装置 . . . . . :   *SYSBAS
ジャーナリング情報 :
  現在ジャーナル処理中 . . . . . :   NO

                                                              続く...
続行するには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

```
                    オブジェクト記述の表示 - 明細
                                                        ライブラリー 1 の 1
オブジェクト . . . . . :   ORDDTLP         属性 . . . . . :   PF
  ライブラリー . . . . . :     DISTLIB      所有者 . . . . . :   QSECOFR
ライブラリー ASP 装置 . . :   *SYSBAS       ライブラリー ASP グループ . :   *SYSBAS
タイプ . . . . . :   *FILE          1 次グループ . . . . . :   *NONE

保管/復元情報 :
  保管日/時刻 . . . . . :
  復元日/時刻 . . . . . :
  保管コマンド . . . . . :
  装置タイプ. . . . . :

                                                              終わり
続行するには，実行キーを押してください。

F3= 終了    F12= 取り消し
```

関連概念:

463 ページの『オブジェクト・タイプおよび共通属性』
システム上の各オブジェクト・タイプは、システム内に固有の目的を持ち、そのオブジェクトを記述する共通の属性セットを持っています。

オブジェクト記述の検索

オブジェクト記述の検索 (RTVOBJD) コマンドは、特定のオブジェクトの記述を CL プログラムまたはプロシージャーに戻します。

オブジェクト記述を検索するには変数を使用します。オブジェクト記述から、システムで不要なオブジェクトを判別することができます。

また、オブジェクト記述の取得 (QUSROBJD) API を使用して、特定のオブジェクトの説明をプログラムまたはプロシージャーに戻すこともできます。システムは変数を使用して記述を戻します。

オブジェクト記述の検索 (RTVOBJD) コマンドは、オブジェクトの変数として次の記述を戻すことができます。

- オブジェクトが入っているライブラリーの名前。
- オブジェクトの拡張属性 (プログラム・タイプやファイル・タイプなど)

- ユーザー定義属性
- オブジェクトのテキスト記述
- オブジェクト所有者のユーザー・プロファイルの名前
- オブジェクトの 1 次グループの名前
- オブジェクト ASP 番号
- ライブラリー ASP 番号
- オブジェクト ASP 装置
- オブジェクト ASP グループ装置
- ライブラリー ASP 装置
- ライブラリー ASP グループ装置
- オブジェクトが、常駐している ASP をオーバーフローしたかどうかの表示
- オブジェクトが作成された日付と時刻
- オブジェクトが最後に変更された日付と時刻
- オブジェクトが最後に保管された日付と時刻
- オブジェクトが SAVACT (*LIB、*SYSDFN、または *YES) 保管操作の間に最後に保管された日時
- オブジェクトが最後に復元された日付と時刻
- オブジェクト作成者のユーザー・プロファイルの名前
- オブジェクトを作成したシステム
- オブジェクト定義域
- 使用状況データ収集の有無
- オブジェクトが最後に使用された日付と時刻
- オブジェクトが使用された日数
- 使用日数が最後にリセットされた日付
- オブジェクト・データの記憶域状況
- オブジェクトの圧縮状況
- オブジェクトのサイズ (バイト数)
- オブジェクトの主に関連するスペースのサイズ (バイト数)
- オブジェクトに関連付けられたスペースが最適に調整されているかどうかの表示
- 最後の保管時の記憶域でのオブジェクトのサイズ (バイト数)
- オブジェクトの保管に使用したコマンド
- オブジェクトをテープに保管した場合に生成されたテープ順序番号
- オブジェクトを保管するために使用したテープ・ボリュームまたはディスケット・ボリューム
- オブジェクトが最後に保管された装置のタイプ
- オブジェクトが保管ファイルに保管された場合、その保管ファイルの名前
- オブジェクトが保管ファイルに保管された場合、その保管ファイルが入っているライブラリーの名前
- オブジェクトの保管時に使用されたファイル・ラベル
- オブジェクトの作成に使用されたソース・ファイルの名前
- オブジェクトの作成に使用されたソース・ファイルが入っているライブラリーの名前
- ソース・ファイルの該当メンバーの名前
- ソース・ファイルの該当メンバーが最後に更新された日付と時刻

- オブジェクト作成時のオペレーティング・システムのレベル
- コンパイラーのライセンス・プログラム識別コード、リリース・レベル、およびモディフィケーション・レベル
- 作成されたオブジェクトに関するオブジェクト制御レベル
- オブジェクト記述変更 (QLICOBJD) API によるオブジェクト変更の可否に関する情報
- オブジェクト記述変更 (QLICOBJD) API によるオブジェクト修正の有無の指示
- ユーザーによるプログラム変更の有無に関する情報
- 検索対象のオブジェクトがライセンス・プログラムの一部である場合、そのライセンス・プログラムの名前、リリース・レベル、およびモディフィケーション・レベル
- 検索対象のオブジェクトを作成する原因になったプログラム一時修正 (PTF) の番号
- プログラム診断依頼書 (APAR) の識別
- オブジェクトへの監査のタイプ
- オブジェクトがデジタル署名であるかどうか
- デジタル署名システム・トラステッド・ソース
- デジタル署名複数署名
- 権限収集値
- オブジェクトの現行ジャーナル状況
- 現行または最終ジャーナル
- ジャーナル・イメージ情報
- 除外するジャーナル項目情報
- リモート・ジャーナル・フィルター
- ジャーナルの最終開始日時
- **ジャーナル処理済み変更適用 (APYJRNCHG)** または **ジャーナル処理済み変更除去 (RMVJRNCHG)** コマンドを適切に使用する必要のある最も古いジャーナル・レシーバーの名前
- 開始ジャーナル・レシーバーを含むライブラリーの名前
- 開始ジャーナル・レシーバーを含むライブラリーに記憶域が割り振られている補助記憶域プール (ASP) 装置の名前
- 開始ジャーナル・レシーバーを含むライブラリーに記憶域が割り振られている補助記憶域プール (ASP) グループの名前

関連情報:

オブジェクト記述の検索 (QUSROBJD) API

オブジェクト記述検索 (RTVOBJD) コマンド

例: オブジェクト記述検索コマンドの使用

この CL ソースの例では、**オブジェクト記述検索 (RTVOBJD)** コマンドは、特定のオブジェクトの記述を検索します。

MOBJ というオブジェクトは現行ライブラリー (MYLIB) にあると想定しています。

```
DCL  &LIB      TYPE(*CHAR) LEN(10)
DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
DCL  &USEDATE  TYPE(*CHAR) LEN(7)
DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
DCL  &RESET    TYPE(*CHAR) LEN(7)
.
```

```
.
.
RTVOBJD    OBJ(MYLIB/MOBJ) OBJTYPE(*FILE) RTNLIB(&LIB)
           CRTDATE(&CRTDATE) USEDATE(&USEDATE)
           USECOUNT(&USECNT) RESETDATE(&RESET)
```

このプログラムでは次の情報が検索されます。

- 現行ライブラリーの名前 (MYLIB) が &LIB という名前の CL 変数に入れられます。
- MOBJ の作成日が &CRTDATE という名前の CL 変数に入れられます。
- MOBJ を最後に使用した日付が &USEDATE という名前の CL 変数に入れられます。
- MOBJ を使用した日数が &USECNT という名前の CL 変数に入れられます。使用日数のカウントの開始日は、&RESET という名前の CL 変数に入れられる値です。

オブジェクトに関する作成情報

オブジェクトが作成される際に、オブジェクトの作成者およびオブジェクトが作成されたシステムに関する情報がオブジェクト記述に保管されます。

この情報は、オブジェクト管理および保守に役立ちます。

- オブジェクトの作成者
  - オブジェクトの作成者は、作成操作を実行するユーザー・プロファイルです。これは、ユーザー・プロファイルがグループ・プロファイルを持ち、グループ・プロファイルがそのオブジェクトを所有している場合であっても当てはまります。
  - オブジェクトの作成者は、所有者の変更時にも変わりません。
  - オブジェクトが復元されるときの作成者は、媒体上のオブジェクトの作成者です。
  - オブジェクトを複製するときのオブジェクトの作成者は、**複製オブジェクト作成 (CRTDUPOBJ)** コマンドを実行するユーザーです。
  - IBM 提供オブジェクトの作成者は、*IBM です。
  - バージョン 1 リリース 3.0 の導入前に既にシステムに存在していたユーザー・オブジェクトの場合、作成者はブランクです。
- オブジェクトが作成されたシステム
  - オブジェクトを復元する場合、媒体上のオブジェクトが以前に作成されたシステムに、そのオブジェクトが作成されます。
  - IBM 提供のオブジェクトの場合、フィールドに作成されるシステムは 00000000 です。
  - バージョン 1 リリース 3.0 の導入前に既にシステムに存在していたオブジェクトの場合、フィールドに作成されるシステムはブランクです。

システム内の不要オブジェクトの検出

システム上の不要になったオブジェクトは、オブジェクト記述として示される以下の情報から判別することができます。

未使用のオブジェクトを検出するために、最後に使用された日付および最後に変更された日付の両方を調べます。変更コマンドは、それらのコマンドがオブジェクトを削除して再度作成するか、変更操作によりオブジェクトが変更の一部として読み込まれない限り、最後に使用された日付を更新しません。

- 最後に変更された日付および時刻
  - オブジェクトの作成または変更時にはシステム時刻がオブジェクトに打刻され、変更が生じた日時を示します。

- 最終使用日
  - 最終使用日は 1 日に 1 回だけ更新されます (使用当日で、そのオブジェクトが初めて使用される時点で更新されるだけです)。システムの日付が使用されます。
  - オブジェクト使用の試みが失敗すると、最終使用日は更新されません。ユーザーが権限のないオブジェクトを変更しようとしても、最終使用日は更新されません。
  - 新しいオブジェクトの場合、最終使用日はブランクです。
  - システムに既存のオブジェクトを復元した場合、最終使用日はシステム上のオブジェクトの日付が使用されます。復元時にシステムに存在していない場合、最終使用日はブランクです。
  - 削除した後で、復元操作によって作り直したオブジェクトの場合、最終使用日は失われます。
  - データベース・ファイルの最終使用日は、ファイル内のメンバー数がゼロの場合は、更新されません。例えば、**複製オブジェクト作成 (CRTDUPOBJ)** を使用してコピーを作成したが、データベース・ファイル内にまったくメンバーがないという場合、最終使用日は更新されません。
  - データベース・ファイルの最終使用日は、最終使用日が最新であるファイル・メンバーの最終使用日です。
  - 論理ファイルの場合、最終使用日は、論理メンバー (またはカーソル) を最後に使用した日付です。
  - 物理ファイルの場合、最終使用日は、データ・スペース内のデータを物理アクセスまたは論理アクセスによって最後に使用した日付です。
  - 削除した後で、名前変更操作によって作り直したオブジェクトの場合、最終使用日は失われます。
- 最終アクティビティー日
  - 最終アクティビティー日とは、データ転送、セッションや会話の確立、または装置記述に関するハードウェアの使用が行われた最終日です。
  - 最終使用日 は、装置またはコントローラー状況、あるいはその両方がオンに変更の処理を完了したときに更新されるため、最終使用日は構成記述の使用を正確には反映していません。例えば、仮想装置またはコントローラー、あるいはその両方で ONLINE(*YES) に設定すると、それらの装置でデータ転送または通信が確立されていなくても、最終使用日は初期プログラム・ロード (IPL) が処理され、装置がオンに変更されたときに更新されます。ただし、最終アクティビティー日 は更新されませんので、構成記述の使用を正確に反映することができます。

表 *29.* 使用状況情報の更新

オブジェクトのタイプ	コマンドと操作
すべてのオブジェクト・タイプ	**複製オブジェクト作成 (CRTDUPOBJ)** コマンド、およびオブジェクトのコピーに CRTDUPOBJ を使用する他のコマンド (**ライブラリー・コピー (CPYLIB)** コマンドなど) **オブジェクト権限認可 (GRTOBJAUT)** コマンド (参照されるオブジェクトの場合)
バインド・ディレクトリー	他のモジュールまたはバインド・ディレクトリーにバインドして、バインド・プログラム (CRTPGM コマンド) またはバインド・サービス・プログラム (CRTSRVPGM コマンド) を作成した場合。プログラムの更新 (UPDPGM) コマンドまたはサービス・プログラムの更新 (UPDSRVPGM) コマンドで更新した場合。
変更要求記述	**コマンド CRQ 活動の変更 (CHGCMDCRQA)**
図表形式	**図表表示 (DSPCHT)** コマンド
C ロケール記述	**C ロケール記述ソースの検索 (RTVCLDSRC)** コマンド、または C プログラムで参照された場合
クラス	ジョブ開始に使用された場合

表 29. 使用状況情報の更新 (続き)

オブジェクトのタイプ	コマンドと操作
コマンド	実行された場合 CL プログラムでコンパイルされた場合 原始ステートメント入力ユーティリティー (SEU) ソースの入力時に指示された場合 検査モードでコマンド処理プログラムを呼び出した場合 **注:** コマンド行からプロンプトを要求し、その後で F3 を押しても、コマンドの使用とは見なされません。
通信サイド情報 (CSI)	CPI 通信会話初期設定 (CMINIT) 呼び出しを使用して、サイド情報オブジェクトの種々の会話特性を初期設定する場合
接続リスト	接続リストが構成変更オン保留中状況を超えた場合
システム共通プロダクト・マップ	CSP アプリケーションで参照された場合
システム共通プロダクト・テーブル	CSP アプリケーションで参照された場合
制御装置記述	制御装置が構成変更オン保留中状況を超えた場合
装置記述	入出力装置が構成変更オン保留中状況を超えた場合
データ域	**データ域検索 (RTVDTAARA)** コマンド **データ域表示 (DSPDTAARA)** コマンド
データ待ち行列	以下の API の使用状況情報は、ジョブごとに 1 回だけ更新されます (API の 1 つを最初に開始したとき)。 データ待ち行列送信 (QSNDDTAQ) API データ待ち行列受信 (QRCVDTAQ) API データ待ち行列検索 (QMHQRDQD) API データ待ち行列読み取り (QMHRDQM) API
ファイル (他の指定がない限り、データベース・ファイルのみ)	クローズ時 (クローズの時点で、装置ファイルや保管ファイルなどの他のファイルも更新されます) 消去時 初期設定時 再編成時 コマンド: • **ジャーナル処理済み変更適用 (APYJRNCHG)** コマンド • **ジャーナル処理済み変更除去 (RMVJRNCHG)** コマンド
フォント資源	印刷操作で参照された場合
書式定義	印刷操作で参照された場合
グラフィック記号セット	GDDM* または PGR 図形アプリケーション・プログラムによって参照された場合 内部で、または GSLSS を使ってロードされた場合
ジョブ記述	ジョブ設定に使用された場合

表 29. 使用状況情報の更新 (続き)

オブジェクトのタイプ	コマンドと操作
ジョブ・スケジュール	システムが、ジョブ・スケジュール項目に対してジョブを投入した場合 ユーザーが WRKJOBSCDE パネルから「オプション 10 = 即時投入」を選択した場合
ジョブ待ち行列	待ち行列に項目を入れた場合や、待ち行列から項目を除去した場合
回線記述	回線が構成変更オン保留中状況を超えた場合
ロケール	ロケール検索 API QLGRTVLC 有効な *LOCALE オブジェクトに対するパス名がユーザー・プロファイル *LOCALE 値に含まれている状態でジョブを開始する場合
管理収集	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。
メディア定義	SAVLIB、SAVOBJ、RSTLIB、RSTOBJ、SAVCHGOBJ コマンド。BRMS API、QSRSAVO API。
メニュー	GO コマンドを使ってメニューが表示された場合
メッセージ・ファイル	メッセージを QCPFMSG、##MSG1、##MSG2、および QSSPMSG 以外のメッセージ・ファイルから検索した場合 (ジョブ・ログを作成した場合、メッセージ待ち行列を表示した場合、QHST ログ内のメッセージに関してヘルプを要求した場合、プログラムがマーク・メッセージ以外のメッセージを受け取った場合など) メッセージ・ファイル組み合わせ (MRGMSGF) コマンド (メッセージ・ファイルが QCPFMSG、##MSG1、 ##MSG2、または QSSPMSG の場合を除く)
メッセージ待ち行列	メッセージが QSYSOPR と QHST 以外のメッセージ待ち行列に対して、送信、受信、または表示される場合
モジュール	他のモジュールまたはバインド・ディレクトリーにバインドして、バインド・プログラム (CRTPGM コマンド) またはバインド・サービス・プログラム (CRTSRVPGM コマンド) を作成した場合。 **プログラムの更新 (UPDPGM)** コマンドまたは**サービス・プログラムの更新 (UPDSRVPGM)** コマンドで更新した場合。
ネットワーク・インターフェース記述	ネットワーク・インターフェース記述が構成変更オン保留中状況を超えた場合
ノード・リスト	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。
出力待ち行列	待ち行列に項目を入れた場合や、待ち行列から項目を除去した場合
オーバーレイ	印刷操作で参照された場合
ページ定義	印刷操作で参照された場合
ページ・セグメント	印刷操作で参照された場合
パネル・グループ	特定のプロンプトまたはパネルに関するヘルプ情報を要求するためにヘルプ・キーを押した場合、使用日付が更新されます。 パネル・グループからパネルが表示されるか、印刷された場合
PDF マップ	PDF マップ項目の追加 (QPQAPME) API **PDF マップ項目の処理 (WRKPDFMAPE)** コマンド
印刷記述子グループ	印刷操作で参照された場合
プロダクト可用性	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。

表 29. 使用状況情報の更新 (続き)

オブジェクトのタイプ	コマンドと操作
プロダクト・ロード	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。
プログラム	**CL ソース検索 (RTVCLSRC)** コマンド 実行された場合、しかもシステム・プログラムでない場合
PSF 構成	印刷操作で参照された場合
QUERY 定義	報告書の生成に使用された場合 抽出またはエクスポートされた場合
Query 管理機能書式	報告書の生成に使用された場合 抽出またはエクスポートされた場合
Query 管理機能プログラム	報告書の生成に使用された場合 抽出またはエクスポートされた場合
検索索引	オンライン・ヘルプ情報で F11 キーを押した場合 **検索見出しの開始 (STRSCHIDX)** コマンドを使用した場合
サーバー記憶域	**構成変更 (VRYCFG)** がネットワーク・サーバー記述オブジェクトに対して実行されます。
サービス・プログラム	バインド・サービス・プログラムが活動化された場合
SQL パッケージ	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。
SQL XML スキーマ・リポジトリー	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。
サブシステム記述	サブシステムが開始された場合
スペル援助ディクショナリー	他の辞書の作成に使用された場合 検索した場合 スペル検査時に辞書の中に単語があったが、その辞書が IBM 提供のスペル援助辞書でない場合
表	変換用にプログラムで使用された場合
タイム・ゾーン記述	• タイム・ゾーン記述を持つジョブを開始する。 • DST (夏時間) 境界を横切ったときに新しい現行オフセットを計算するため、タイム・ゾーン記述を参照する。
ユーザー・プロファイル	そのプロファイルでジョブが開始された場合 そのプロファイルがグループ・プロファイルで、そのグループのメンバーを使ってジョブが開始された場合 **ユーザー権限認可 (GRTUSRAUT)** コマンド (参照されたプロファイルに関して)
ワークステーション・ユーザーのカスタマイズ部分	すべてのオブジェクト・タイプに影響を与えるコマンドと操作によってのみ更新されます。

以下は、オブジェクト記述に提供されるオブジェクト使用状況に関する追加情報です。

- 使用日数

- 使用日数は、最新の使用日が更新されると増えます。
- システムに既存のオブジェクトを復元した場合、使用日数はシステム上のオブジェクトの値が使用されます。復元時にシステムに存在していない場合、使用日数はゼロです。
- 削除した後で、復元操作で作り直したオブジェクトは、使用日数が失われます。
- 新しいオブジェクトの場合、使用日数はゼロです。

  注: IBM i オペレーティング・システムは、元の装置ファイルと新しい装置ファイルの違いが判別できません。装置ファイルをシステムに復元する時に、同じ名前の装置ファイルが既に存在する場合、カウントに使用される日付をゼロにリセットしたいなら、既存のファイルを削除しなければなりません。ファイルが削除されないと、システムはこれを元のオブジェクトの復元操作と解釈して、使用される日付のカウントを保持します。
- データベース・ファイルの使用日数は、すべてのファイル・メンバーについての使用日数の合計です。合計がオーバーフローした場合、最大値 (使用日数フィールドの) が示されます。

- 使用日数がリセットされた日付
  - 使用日数を**オブジェクト記述変更 (CHGOBJD)** コマンドまたはオブジェクト記述変更 (QLICOBJD) API を使用してリセットした場合、その日付は記録されます。したがってユーザーは、使用日数の有効期間を知ることができます。
  - ファイルについて使用日数をリセットすると、すべてのメンバーの使用日数もリセットされます。

使用日数および最終使用日を削除できるのは、次のような場合です。

- システム上にある損傷を受けたオブジェクトを復元する場合。
- システムが制限状態ではないときに、プログラムを復元する場合。

オブジェクト記述表示 (DSPOBJD) コマンドを使用して、オブジェクトの詳細な記述を表示することができます。このコマンドは、オブジェクト記述を出力ファイルに書き出す場合にも使用できます。オブジェクト記述の検索には、**オブジェクト記述の検索 (RTVOBJD)** コマンドを使用します。

注: Retrieve Object Description (QUSROBJD) API では、オブジェクト記述検索コマンドと同じ情報を提供しています。

メンバー記述の検索 (RTVMBRD) コマンドおよび**ファイル記述表示 (DSPFD)** コマンドを使用すると、ファイル内のメンバーについて同様な情報が得られます。

次のオブジェクト・タイプについては、オブジェクト使用状況情報は更新されません。

- 警報テーブル (*ALRTBL)
- 権限リスト (*AUTL)
- 構成リスト (*CFGL)
- サービス・クラス記述 (*COSD)
- データ・ディクショナリー (*DTADCT)
- 漢字 (2 バイト文字) 文字セット辞書 (*IGCDCT)
- 漢字 (2 バイト文字) 文字セット分類 (*IGCSRT)
- 漢字 (2 バイト文字) 文字セット・テーブル (*IGCTBL)
- 編集記述 (*EDTD)
- 出口登録 (*EXITRG)
- フィルター (*FTR)

- 用紙制御テーブル (*FCT)
- フォルダー (*FLR)
- インターネット・パケット交換記述 (*IPXD)
- ジャーナル (*JRN)
- ジャーナル・レシーバー (*JRNRCV)
- ライブラリー (*LIB)
- モード記述 (*MODD)
- ネットワーク・サーバー構成 (*NWSCFG)
- ネットワーク・サーバー記述 (*NWSD)
- NetBIOS 記述 (*NTBD)
- プロダクト定義 (*PRDDFN)
- 参照コード変換テーブル (*RCT)
- セッション記述 (*SSND)
- S/36 マシン記述 (*S36)
- ユーザー定義 SQL タイプ (*SQLUDT)
- ユーザー待ち行列 (*USRQ)

関連情報:

CL コマンド検索プログラム

ライブラリー間のオブジェクト移動

オブジェクト移動 (MOVOBJ) コマンドは、ライブラリー間でオブジェクトを移動させます。

あるライブラリーから別のライブラリーへオブジェクトを移すことによって、オブジェクトを一時的に使用不能にすることができ、オブジェクトの古いバージョンを新しいバージョンで置き換えることができます。例えば、新しいマスター・ファイルを作成する場合に、それが古いマスター・ファイルとは別のライブラリーに一時的に入るようにすることができます。通常、古いマスター・ファイルのデータを新しいマスター・ファイルにコピーする必要があるので、新しいマスター・ファイルの作成が完了するまで、古いマスター・ファイルは削除することができません。必要な処理が完了したら、古いマスター・ファイルを削除し、それが入っていたライブラリーに新しいマスター・ファイルを移すことができます。

オブジェクトの移動ができるのは、オブジェクトに対するオブジェクト管理権限、移したいオブジェクトが入っているライブラリーに対する削除権限と実行権限、およびオブジェクトの移動先のライブラリーに対する追加権限と読み取り権限を持っているユーザーだけです。

一時ライブラリー QTEMP のオブジェクトを他のライブラリーに移すことはできますが、QTEMP へのオブジェクトの移動はできません。また、出力待ち行列は、それが空でない限り移動させることはできません。

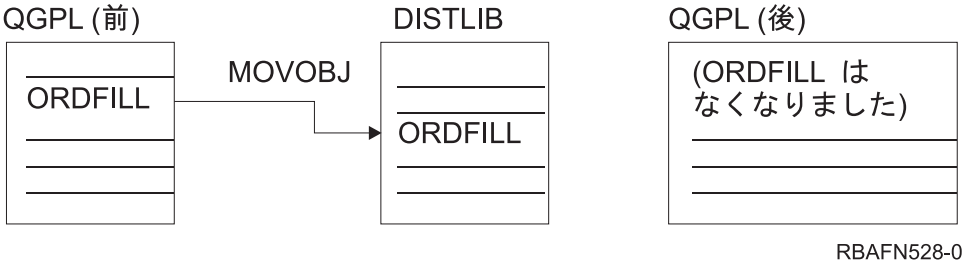
ジャーナルおよびジャーナル・レシーバーは、それが本来作成されたライブラリーに戻す場合に限り移動が可能です。 **ジャーナル・オブジェクトが記憶域再利用 (RCLSTG)** コマンドにより QRCL に入れられた場合には、元のライブラリーに戻してからでなければ、そのジャーナル・オブジェクトは操作可能となりません。

次のオブジェクトは、移動させることができません。

- 権限リスト (*AUTL)

- サービス・クラス記述 (*COSD)
- クラスター資源グループ (*CRG)
- 構成リスト (*CFGL)
- 接続リスト (*CNNL)
- 制御装置記述 (*CTLD)
- データ・ディクショナリー (*DTADCT)
- 装置記述 (*DEVD)
- ディスプレイ装置メッセージ待ち行列 (*MSGQ)
- 文書 (*DOC)
- 編集記述 (*EDTD)
- 出口登録 (*EXITRG)
- フォルダー (*FLR)
- 漢字 (2 バイト文字) 文字セット (DBCS) フォント・テーブル (*IGCTBL)
- イメージ・カタログ (*IMGCLG)
- インターネット・パケット交換記述 (*IPXD)
- ジョブ・スケジュール (*JOBSCD)
- ライブラリー (*LIB)
- 回線記述 (*LIND)
- モード記述 (*MODD)
- NetBIOS 記述 (*NTBD)
- ネットワーク・インターフェース記述 (*NWID)
- ネットワーク・サーバー構成 (*NWSCFG)
- 構造化照会言語 (SQL) パッケージ (*SQLPKG)
- 構造化照会言語 (SQL) XML スキーマ・リポジトリー (*SQLXSR)
- システム/36 マシン記述 (*S36)
- システム活動記録ログ (QHST)
- システム・オペレーター・メッセージ待ち行列 (QSYSOPR)
- 時間帯の記述 (*TIMZON)
- ユーザー定義 SQL タイプ (*SQLUDT)
- ユーザー・プロファイル (*USRPRF)

次の例では、あるファイルを (作成時にそのファイルが入れられた) QGPL から受注ライブラリー DISTLIB に移動させ、他の受注ファイルと同じグループに入るようにしています。



オブジェクトを移動させるためには、移動先のライブラリー (TOLIB) およびオブジェクト・タイプ (OBJTYPE) を指定しなければなりません。

```
MOVOBJ  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)  TOLIB(DISTLIB)
```

オブジェクトを移動する場合には、他のオブジェクトが依存しているオブジェクトを移動しないように注意しなければなりません。例えば、CL プロシージャーでは、そのプロシージャーで使用されるコマンドのコマンド定義が、実行時にも、モジュールの作成時と同じライブラリーに入っていることが不可欠の条件になる場合があります。プロシージャーのコンパイル時および実行時には、コマンド定義は指定のライブラリーに入っているか、あるいは *LIBL が指定されている場合はライブラリー・リスト内のライブラリーに入っています。ライブラリー名が指定されている場合には、コマンド定義は実行時にも、コンパイル時と同じライブラリーに入っていなければなりません。 *LIBL が指定されている場合には、コマンド定義はライブラリー・リスト内のライブラリーに移動する限り、プログラムの実行時とコンパイル時でライブラリーが異なっていても差し支えありません。同様に、ユーザーが作成するアプリケーション・プログラムの場合にも、特定のオブジェクトが特定のライブラリーに入っていなければ、正常に実行できない場合もあります。

別のオブジェクトを参照するオブジェクトは、 (オブジェクトのライブラリーとして *LIBL が指定できる場合でも) 参照するオブジェクトのライブラリーに依存していることがあります。したがって、オブジェクトを移動する場合には、他のオブジェクトにおけるそのオブジェクトへの参照を変更しなければなりません。別のオブジェクトを参照するオブジェクトの例を次に示します。

- サブシステム記述は、ジョブ待ち行列、クラス、メッセージ待ち行列、およびプログラムを参照します。
- コマンド定義は、プログラム、メッセージ・ファイル、ヘルプ・パネル・グループ、および REXX プロシージャーを含むプロシージャー・ファイルを参照します。
- 装置ファイルは、出力待ち行列を参照します。
- 装置記述は、変換テーブルを参照します。
- ジョブ記述は、ジョブ待ち行列および出力待ち行列を参照します。
- データベース・ファイルは、他のデータベース・ファイルを参照します。
- 論理ファイルは、物理ファイルまたは様式選択項目を参照します。
- ユーザー・プロファイルは、プログラム、メニュー、ジョブ記述、メッセージ待ち行列、および出力待ち行列を参照します。
- CL プログラムは、表示装置ファイル、データ域、および他のプログラムを参照します。
- 表示装置ファイルは、データベース・ファイルを参照します。
- 印刷装置ファイルは、出力待ち行列を参照します。

注**:** システム・ライブラリー QSYS に入っているオブジェクトを移動する場合は注意しなければなりません。 QSYS 内のオブジェクトは、システムの有効な作動に必要なもので、システムが必ず見つけられるようになっていなければなりません。汎用ライブラリー QGPL 内のいくつかのオブジェクト、特にジョブ待ち行列および出力待ち行列についても、同じことが言えます。

オブジェクト移動 (MOVOBJ) コマンドはオブジェクトを一度に 1 つだけ移動します。

関連タスク:

506 ページの『オブジェクト名の変更』

オブジェクト名変更 (RNMOBJ) コマンドは、オブジェクト名を変更します。オブジェクト名を変更することができるのは、そのオブジェクトに対するオブジェクト管理権限、およびそのオブジェクトが入っているライブラリーに対する更新権限と実行権限を持っている場合だけです。

関連情報:

複製オブジェクトの作成

複製オブジェクト作成 (CRTDUPOBJ) コマンドを使用して、既存のオブジェクトのコピーを作成することができます。

複製されたオブジェクトは、もとのオブジェクトと同じオブジェクト・タイプおよび同じ権限属性を備えており、もとのオブジェクトと同じ補助記憶域プール (ASP) に作成されます。複製オブジェクト作成コマンドを発行したユーザーが、複製オブジェクトの所有者になります。

注**:** ジャーナル処理対象のファイルの複製オブジェクトを作成した場合には、その複製オブジェクト (ファイル) ではジャーナル処理は活動化されません。しかし、後でこのオブジェクトを選択し、ジャーナル処理をすることができます。複製オブジェクトを作成したが、そのオブジェクト (ファイル) にまったくメンバーがない場合は、最後の使用日フィールドはブランクになり、使用日数のカウントはゼロになります。

オブジェクトを複製するためには、もとのオブジェクトに対する管理権限と使用権限、複製されたオブジェクトを入れるライブラリー (受け入れライブラリー) に対する使用権限と追加権限、もとのオブジェクトが入っているライブラリー (取り出しライブラリー) に対する使用権限、および処理するユーザー・プロファイルに対する追加権限を持っていることが必要です。

権限リストを複製するためには、もとのオブジェクトに対する権限リスト管理権限、およびライブラリー QSYS に対する追加権限とオブジェクト操作権限を持っていることが必要です。

ジョブ待ち行列、メッセージ待ち行列、出力待ち行列、およびデータ待ち行列の場合には、その定義だけが複製されます。ジョブ待ち行列および出力待ち行列は、一時ライブラリー (QTEMP) に複製することはできません。物理ファイルまたは保管ファイルの場合には、ファイル内のデータも同時に複製するかどうかを指定することができます。

下記のオブジェクトは、複製することができません。

- サービス・クラス記述 (*COSD)
- クラスター資源グループ (*CRG)
- 構成リスト (*CFGL)
- 接続リスト (*CNNL)
- 制御装置記述 (*CTLD)
- データ・ディクショナリー (*DTADCT)
- 装置記述 (*DEVD)
- データ待ち行列 (*DTAQ)
- 文書 (*DOC)
- 編集記述 (*EDTD)
- 出口登録 (*EXITRG)
- フォルダー (*FLR)
- 漢字フォント・テーブル (*IGCTBL)
- イメージ・カタログ (*IMGCLG)
- インターネット・パケット交換記述 (*IPXD)
- ジョブ・スケジュール (*JOBSCD)
- ジャーナル (*JRN)

- ジャーナル・レシーバー (*JRNRCV)
- ライブラリー (*LIB)
- 回線記述 (*LIND)
- モード記述 (*MODD)
- ネットワーク・インターフェース記述 (*NWID)
- ネットワーク・サーバー構成 (*NWSCFG)
- ネットワーク・サーバー記述 (*NWSD)
- 参照コード変換テーブル (*RCT)
- サーバー記憶域 (*SVRSTG)
- スペル援助辞書 (*SPADCT)
- SQL パッケージ (*SQLPKG)
- SQL XML スキーマ・リポジトリー (*SQLXSR)
- システム/36 マシン記述 (*S36)
- システム・オペレーター・メッセージ待ち行列 (QSYSOPR)
- システム活動記録ログ (QHST)
- 時間帯の記述 (*TIMZON)
- ユーザー定義 SQL タイプ (*SQLUDT)
- ユーザー・プロファイル (*USRPRF)
- ユーザー待ち行列 (*USRQ)

ファイル内のデータの一部のみを複製する場合には、**複製オブジェクト作成 (CRTDUPOBJ)** コマンドに続けて、選択値を指定した**ファイル・コピー (CPYF)** コマンドを使用します。

次のコマンドによって、受注見出し物理ファイルの複製が作成され、ファイルのデータもコピーされます。

```
CRTDUPOBJ OBJ(ORDHDRP) FROMLIB(DSTPRODLIB) OBJTYPE(*FILE) +
          TOLIB(DISTLIB2) NEWOBJ(*SAME) DATA(*YES)
```

複製オブジェクトを作成する場合には、別のオブジェクトを参照しているオブジェクトの複製の作成による結果を考慮する必要があります。多くのオブジェクトは、名前により他のオブジェクトを参照しており、そのような参照の多くは特定のライブラリー名により修飾されています。したがって、複製オブジェクトが入れられたライブラリーとは別のライブラリーに入っているオブジェクトに対する参照が、複製オブジェクトに含まれることも起こり得ます。ファイル以外のすべてのオブジェクト・タイプでは、他のオブジェクトへの参照は複製オブジェクトにもコピーされます。ファイルの場合には、複製オブジェクトはもとのファイルの様式を共用することになります。

複製元のライブラリーに入っていて、論理ファイルの基礎となっている物理ファイルは、すべて複製先のライブラリーにも入っていなければなりません。複製元のライブラリーと複製先のライブラリーの物理ファイルのレコード様式名およびレコード・レベル識別コードが比較され、両ファイル内の物理ファイルが一致しなければ、論理ファイルの複製は行われません。

論理ファイルで、複製元のライブラリーに入っている様式選択項目が使用されている場合には、その様式選択項目が複製先のライブラリーの中にもあるものと見なされます。

関連情報:

複製オブジェクト作成 (CRTDUPOBJ) コマンド

オブジェクト名の変更

オブジェクト名変更 (RNMOBJ) コマンドは、オブジェクト名を変更します。オブジェクト名を変更することができるのは、そのオブジェクトに対するオブジェクト管理権限、およびそのオブジェクトが入っているライブラリーに対する更新権限と実行権限を持っている場合だけです。

権限リストの名前を更新するためには、権限リスト管理権限、およびライブラリー QSYS に対する更新権限と読み取り権限が必要です。

次のオブジェクトの名前は、変更することができません。

- サービス・クラス記述 (*COSD)
- クラスター資源グループ (*CRG)
- データ・ディクショナリー (*DTADCT)
- 漢字フォント・テーブル (*IGCTBL)
- ディスプレイ装置メッセージ待ち行列 (*MSGQ)
- 文書 (*DOC)
- 出口登録 (*EXITRG)
- フォルダー (*FLR)
- ジョブ・スケジュール (*JOBSCD)
- ジャーナル (*JRN)
- ジャーナル・レシーバー (*JRNRCV)
- モード記述 (*MODD)
- ネットワーク・サーバー構成 (*NWSCFG)
- ネットワーク・サーバー記述 (*NWSD)
- SQL パッケージ (*SQLPKG)
- SQL XML スキーマ・リポジトリー (*SQLXSR)
- システム/36 マシン記述 (*S36)
- システム活動記録ログ (QHST)
- システム・ライブラリー QSYS、および一時ライブラリー QTEMP
- システム・オペレーター・メッセージ待ち行列 (QSYSOPR)
- 時間帯の記述 (*TIMZON)
- ユーザー定義 SQL タイプ (*SQLUDT)
- ユーザー・プロファイル (*USRPRF)

出力待ち行列は、それが空でない限り、名前を変更することはできません。 IBM 提供のコマンドは、ライセンス・プログラムでも使用されているので、コマンド名を変更してはなりません。

オブジェクトの名前を変更するためには、オブジェクトの現在の名前、変更後の名前、およびオブジェクト・タイプを指定します。

次の RNMOBJ コマンドによって、オブジェクト名 ORDERL が ORDFILL に変更されます。

```
RNMOBJ   OBJ(QGPL/ORDERL)  OBJTYPE(*FILE)   NEWOBJ(ORDFILL)
```

オブジェクト自体は同じライブラリーに入ったままなので、新しいオブジェクト名として修飾名を指定することはできません。 **オブジェクトの名前変更 (RNMOBJ)** コマンドを発行したときに、名前の変更を指定し

たオブジェクトが使用中であった場合には、コマンドは実行されますが、オブジェクトの名前は変更されません。その結果として、システムからのメッセージが表示されます。

オブジェクト名を変更する場合には、他のオブジェクトが依存しているオブジェクトの名前を変更しないように注意しなければなりません。例えば、CL プログラムでは、プログラムで使用されるコマンドのコマンド定義の名前が、プログラムの実行時にもプログラムのコンパイル時と同じでなければなりません。したがって、プログラムの作成後、プログラムの実行時までにコマンド定義の名前が変更されていると、コマンドが見つからないためにプログラムは実行できなくなります。同様に、ユーザーが作成するアプリケーション・プログラムでも、プログラムの作成時と実行時で名前が同じでなければならないオブジェクトがいくつかあります。

ジャーナル、ジャーナル・レシーバー、データ・ディクショナリー、クラスター資源グループ、または SQL パッケージが入っているライブラリーの名前を変更することはできません。

別のオブジェクトを参照するオブジェクトは、(ライブラリーとして *LIBL が指定できる場合でも) 参照されるオブジェクトおよびライブラリーの名前に依存していることがあります。したがって、オブジェクト名を変更する場合には、そのオブジェクトが他のオブジェクトで参照されている場合は、それらの参照名もすべて変更しなければなりません。

物理ファイルまたは論理ファイルの名前を変更しても、ファイル内のメンバーの名前は変更されません。物理ファイルまたは論理ファイルのメンバー名は、メンバー名変更 (RNMM) コマンドを使用して変更することができます。

注: システム・ライブラリー QSYS に入っているオブジェクトの名前を変更する場合は注意しなければなりません。 QSYS 内のオブジェクトは、システムの有効な作動に必要なもので、システムが必ず見つけられるようになっていなければなりません。汎用ライブラリー QGPL 内のいくつかのオブジェクトについても同じことが言えます。

関連資料:

501 ページの『ライブラリー間のオブジェクト移動』

オブジェクト移動 (MOVOBJ) コマンドは、ライブラリー間でオブジェクトを移動させます。

関連情報:

オブジェクト名変更 (RNMOBJ) コマンド

オブジェクトの圧縮または圧縮解除

オブジェクトの圧縮 (CPROBJ) コマンドを使用して、システムのディスク・スペースを節約するために、選択したオブジェクトを圧縮することができます。 **オブジェクトの圧縮解除 (DCPOBJ)** コマンドを使用して、圧縮されたオブジェクトを圧縮解除することができます。

圧縮および圧縮解除をサポートしているオブジェクト・タイプは、
*PGM、*SRVPGM、*MODULE、*PNLGRP、*MENU (UIM メニューのみ)、および *FILE (表示装置ファイルまたは印刷装置ファイルのみ) です。データベース・ファイルは、圧縮することができません。 IBM i が提供するオブジェクトと同様に、ユーザー作成のオブジェクトも、圧縮または圧縮解除を行うことができます。オブジェクトの圧縮状況の判別や検索には、**オブジェクト記述表示 (DSPOBJD)** コマンド (*FULL 表示)、または**オブジェクト記述の検索 (RTVOBJD)** コマンドを使用します。

関連情報:

オブジェクト圧縮 (CPROBJ) コマンド

オブジェクト圧縮解除 (DCPOBJ) コマンド

オブジェクト記述検索 (RTVOBJD) コマンド

オブジェクトの圧縮の制約

オブジェクト・タイプ *PGM、*SRVPGM、*MODULE、*PNLGRP、*MENU、および *FILE (表示装置ファイルおよび印刷装置ファイルのみ) は、CPROBJ または DCPOBJ コマンドを使用して圧縮または圧縮解除を行うことができます。

オブジェクトは、次の両方の項目が当てはまる場合にだけ圧縮されます。

- システムがそのオブジェクトに対して排他ロックを入手できる場合。
- 圧縮されたサイズによりディスク・スペースが節約される場合。

オブジェクトの圧縮には、次の制約があります。

- オペレーティング・システムのバージョン 1、リリース 3 より前に作成されたプログラムは圧縮できません。
- オペレーティング・システムのバージョン 3、リリース 6 より前に作成された、再変換されていないプログラム、サービス・プログラム、またはモジュールは、圧縮できません。
- IBM 提供のライブラリー QSYS および QSSP に存在するプログラムは、そのプログラムのページング・プールの値が *BASE である場合を除き、圧縮できません。プログラムのページング・プールの値を調べるには、 **プログラム表示 (DSPPGM)** コマンドを使用します。 QSYS と QSSP 以外のライブラリーのプログラムは、ページング・プールの値に関係なく、圧縮することができます。
- 属性が UIM であるメニューだけが圧縮可能です。
- ファイルは、属性が DSPF および PRTF のファイルのみを圧縮できます。
- システム・ライブラリーのプログラム・オブジェクトを圧縮するには、システムは制限状態 (すべてのサブシステムが終了している) でなければなりません。
- プログラムは、システムで実行している間に圧縮してはなりません。システムで実行中に圧縮すると、プログラムは異常終了します。

制限状態でない場合、次の表に示すように複数のジョブを使用して、圧縮をより速く実行することができます。

表 *30.* 複数のジョブを使用したオブジェクトの圧縮

オブジェクト・タイプ	**IBM** 提供	ユーザー提供
*FILE	ジョブ 3: QSYS	ジョブ 7: USRLIB1
*MENU	ジョブ 2: QSYS	ジョブ 8: USRLIB1
*MODULE	適用外	ジョブ 10: USRLIB1
*PGM	制限状態のみ	ジョブ 5: USRLIB1
*PNLGRP	ジョブ 1: QSYS ジョブ 4: QHLPSYS	ジョブ 6: USRLIB1
*SRVPGM	ジョブ 11: QSYS	ジョブ 9: USRLIB1

オブジェクトの一時的な圧縮解除

圧縮されたオブジェクトが使用される時点で、システムは自動的に圧縮を一時解除します。

一時的に圧縮解除されたオブジェクトは、次の時点までその状態を持続します。

- システムの IPL。これは、一時的に圧縮解除されたオブジェクトを削除します (圧縮されたオブジェクトは存続します)。

- 一時的に圧縮解除されたオブジェクトの記憶域を再利用する目的で、 **一時記憶域の再利用 (RCLTMPSTG)** コマンドを使用した場合。この結果、一時的に圧縮解除されたオブジェクトは、それらが指定の日数の間使用されなかった場合には削除されます (圧縮されたオブジェクトは存続します)。
- 一時的に圧縮解除されたオブジェクトが 2 日間以上使用されるか、同じ IPL で 5 回以上使用された場合。この場合、そのオブジェクトは永続的に圧縮解除されます。
- DCPOBJ コマンドを使用してオブジェクトを圧縮解除した場合。この場合、そのオブジェクトは永続的に圧縮解除されます。
- システムがそのオブジェクトに対する排他ロックを入手している場合。

注:

1. オブジェクトのタイプが *PGM、*SRVPGM、または *MODULE の場合には、一時的な圧縮の解除はできません。圧縮されたプログラムの呼び出し、またはデバッグを行った場合、そのプログラムには永続的な圧縮解除が自動的に行われます。
2. 圧縮されたファイル・オブジェクトは、オープンの時点で自動的に圧縮解除されます。
3. 圧縮されたファイルの記述を検索する場合、そのファイルは一時的に圧縮解除されます。以下に、このようなファイルの検索の例が 2 つあります。
   - **ファイル・フィールド記述表示 (DSPFFD)** コマンドを使用して、ファイルのフィールド・レベルの情報を表示する。
   - **ファイル宣言 (DCLF)** コマンドを使用して、ファイルを宣言する。

オブジェクトの自動的な圧縮解除

IBM i または他の IBM ライセンス・プログラムに付属している圧縮されたオブジェクトは、ライセンス・プログラムの導入後に システムが圧縮を解除します。

圧縮解除が行われるのは、システムに使用可能な記憶域が十分にある場合だけです。

システムは QDCPOBJx と呼ばれるシステム・ジョブを自動的に開始して、オブジェクトの圧縮を解除します。

QDCPOBJ ジョブの数は、プロセッサー数 + 1 になります。このジョブは、ユーザーが変更、終了、および停止することのできない、優先順位 60 で実行するシステム・ジョブです。 QDCPOBJx ジョブは、 **活動ジョブ処理 (WRKACTJOB)** コマンドで表示すると、以下に示すいずれかの状態になる場合があります。

- RUN (実行中): ジョブは、オブジェクトの圧縮解除中です。
- EVTW (イベント待ち): ジョブは、圧縮解除中ではありません。他のオブジェクトを圧縮解除する必要がある場合に、ジョブがアクティブになります。 (つまり、ライセンス・プログラムを追加して導入する場合)
- DLYW (遅延待ち): ジョブが一時的に停止されています。以下の状態が発生すると、QDCPOBJx ジョブが停止する原因となります。
  - システムが、制限状態で実行している。
  - ライセンス・プログラムが、「ライセンス・プログラムの処理」画面で導入された。ジョブは、最長 15 分まで遅延待ち状態にしておくことができます。その後、圧縮解除ジョブが開始されます。
- LCKW (ロック待ち): ジョブは、内部ロック待ちです。この状態は、QDCPOBJ ジョブが DLYW 状態にある場合によく発生します。

既存のオペレーティング・システムに追加してオペレーティング・システムを導入する場合は、次の記憶域要件が適用されます。

- QDCPOBJx ジョブを開始するために、未使用の記憶域が 250MB 以上必要です。
- システムに使用可能な記憶域が 750MB 以上あれば、導入したばかりのシステム・オブジェクトすべての圧縮を解除するようジョブが投入されます。
- システムの使用可能な記憶域が 250MB より小さい場合、ジョブは投入されず、オブジェクトは使用時に圧縮解除されることになります。
- システムの使用可能な記憶域が 250MB と 750MB の間である場合は、使用される回数の多いオブジェクトだけが自動的に圧縮解除されます。

使用回数の多いオブジェクト とは、5 回以上使用されており、最後に使用されてから 14 日間を超えていないオブジェクトのことです。使用回数の少ない他のオブジェクトは圧縮したままです。

ライセンス内部コード (LIC) 画面で導入オプション 2、つまりライセンス内部コードの導入とシステムの初期設定を使用して始動したシステムにオペレーティング・システムを導入する場合は、未使用の記憶域が最低 1000MB 以上必要です。

最後のシステムの終了時に QDCPOBJx ジョブがアクティブの場合、QDCPOBJx ジョブは次回の IPL 時に再始動します。

オブジェクトの削除

オブジェクトを削除するには、**オブジェクト削除 (DLTOBJ)** コマンド、そのオブジェクトのタイプに対応する削除 (DLTxxx) コマンド、あるいは「オブジェクトの処理」画面 (**「ライブラリーの処理 (WRKLIB)」**画面から表示される画面) の削除オプションを使用することができます。

オブジェクトを削除するには、オブジェクトに対するオブジェクト存在権限およびライブラリーに対する実行権限を持っていなければなりません。権限リストを削除することができるのは、権限リストの所有者または *ALLOBJ 特殊権限を持っているユーザーだけです。

オブジェクトを削除する場合には、オブジェクトを必要としている他のユーザーや、そのオブジェクトを使用中のユーザーが存在しないことを確認しなければなりません。通常、他のユーザーが使用しているオブジェクトは、削除することはできません。ただし、プログラムは、呼び出し前に**オブジェクト割り振り (ALCOBJ)** コマンドを使用して割り振っている場合を除いて、削除することができます。

一部の作成コマンド (プログラム、コマンド、および装置ファイルを作成するコマンドなど) には REPLACE オプションがあります。このオプションを使用すれば、ユーザーは既に置き換えが済んでいるオブジェクトの旧バージョンも使用し続けることができます。システムは、これらのオブジェクトの旧バージョンをライブラリー QRPLOBJ に保管します。

システム・ライブラリーに入っているオブジェクトの削除には十分な注意が必要です。このライブラリー中のオブジェクトは、システムの正常な動作に欠かせないものです。

ほとんどの削除コマンドでは、オブジェクト名の代わりに総称名を指定することができます。総称名による削除を行う前に、**オブジェクト記述表示 (DSPOBJD)** コマンドで総称名を指定して、総称名により削除されるのが削除したいオブジェクトのみであることを確認してください。

関連概念:

475 ページの『総称オブジェクト名検索』
総称検索を使用すると、複数のオブジェクトを検索することができます。

関連資料:

483 ページの『ライブラリーの削除およびクリア』
ライブラリー削除 (DLTLIB) コマンドを使用してライブラリーを削除した場合には、ライブラリー自体とともにライブラリー内のオブジェクトも削除されます。

関連情報:

オブジェクト割り振り (ALCOBJ) コマンド

オブジェクト記述表示 (DSPOBJD) コマンド

リソースの割り振り

システム上でのオブジェクトの割り振りは、保全性を確保し、可能な限り高度な並行処理ができるような形で行われます。

オブジェクトは、それに対して複数の操作を同時に行うことができるように保護されています。例えば、オブジェクトは、2 人のユーザーが同時にそのオブジェクトを読み取ることができるように、あるいは 1 人のユーザーがオブジェクトの読み取りと更新を行い、別のユーザーはそのオブジェクトの読み取りだけが可能であるように割り振られます。

IBM i オペレーティング・システムでは、オブジェクトの割り振りは、そのオブジェクトに対して実行される機能に応じて行われます。以下にその例を示します。

- あるユーザーがオブジェクトを表示またはダンプしている間は、別のユーザーはそのオブジェクトを読み取ることができます。
- あるユーザーがオブジェクトの変更、削除、名前変更、または移動を行っている間は、他のユーザーはそのオブジェクトを使用することはできません。
- あるユーザーがオブジェクトの保管を行っている間は、他のユーザーはそのオブジェクトを読み取ることはできますが、更新または削除することはできません。あるユーザーがオブジェクトを復元している間は、他のユーザーはそのオブジェクトの読み取りまたは更新を行うことはできません。
- あるユーザーが入力をするためにデータベース・ファイルをオープンしている間は、別のユーザーはそのファイルを読み取ることができます。あるユーザーが出力をするためにデータベース・ファイルをオープンしている間は、別のユーザーはそのファイルを更新することができます。
- あるユーザーが装置ファイルをオープンしている間は、別のユーザーにはそのファイルの読み取りだけが許されます。

一般に、オブジェクトは必要になった時点で割り振られます。つまり、ジョブ・ステップでオブジェクトが必要になった時点でオブジェクトがジョブ・ステップに割り振られ、使用され、そして別のジョブで使用できるように割り振りが解除されます。オブジェクトは、そのオブジェクトを要求した最初のジョブに割り振られます。プログラムで、オブジェクトを要求どおりに割り振ることができなかった場合の例外状態の処置を指定することができます。

ジョブでオブジェクトが必要になる前にオブジェクトをジョブに割り振ってオブジェクトの可用性を確保しておき、一部しか完了していない機能がオブジェクトの割り振りを待たなくてもよいようにしたい場合があります。これを、オブジェクトの事前割り振りと呼びます。 **オブジェクト割り振り (ALCOBJ)** コマンドを使用して、オブジェクトを事前割り振りすることができます。オブジェクトを割り振るためには、そのオブジェクトに対するオブジェクト存在権限、オブジェクト管理権限、またはオブジェクト操作権限を持っていなければなりません。

割り振られたオブジェクトは、経路指定ステップの終了時に自動的に割り振り解除されます。それ以外の時点でオブジェクトの割り振りを解除したい場合には、 **オブジェクト割り振り解除 (DLCOBJ)** コマンドを使用してください。

プログラムを削除から保護するために、プログラム呼び出しの前に割り振ることができます。プログラムが複数のジョブで同時に実行されることを防止したい場合には、他のジョブでプログラムを呼び出す前に、各ジョブでそのプログラムを占有ロック状態にしなければなりません。

ALCOBJ コマンドまたは DLCOBJ コマンドを使用して APPC 装置記述を割り振ることはできません。

次に示すのは、更新の目的で 2 つのファイル・メンバーを必要とするバッチ・ジョブの例です。どちらのファイルのメンバーも、更新中に他のプログラムが読み取ることができますが、このバッチ・ジョブの実行中は他のプログラムからこれらのファイルのメンバーを更新することはできません。各ファイルの最初のメンバーは、読み取り可能占有ロック状態で事前割り振りされます。

```
//JOB  JOBD(ORDER)
  ALCOBJ  OBJ((FILEA *FILE *EXCLRD) (FILEB *FILE *EXCLRD))
  CALL  PROGX
//ENDJOB
```

割り振られたオブジェクトは、他のユーザーが必要としている場合もあるので、使用を終了した時点で直ちに割り振りを解除しなければなりません。ただし、経路指定ステップの終了時には、割り振られたオブジェクトは自動的に割り振り解除されます。

FILEA および FILEB の最初のメンバーが事前に割り振られている場合には、読み取り可能占有の制約は効力がなくなります。ファイルを使用する場合には、ファイルの使用中に変更されないようにするために、ファイルを事前に割り振っておくことができます。

注**:** 1 つのオブジェクトを (複数の割り振りコマンドを使用して) 2 回以上割り振った場合には、1 つの DLCOBJ コマンドではそのオブジェクトの割り振りを完全に解除することはできません。各割り振りコマンドに対応する数の割り振り解除コマンドが必要です。

ファイル作成コマンドの WAITRCD パラメーターには、レコード・ロックについての待機時間を指定します。 **クラス作成 (CRTCLS)** コマンドの DFTWAIT パラメーターは、他のオブジェクトに対する待機時間を指定するためのものです。

関連タスク:

537 ページの『メッセージ記述の定義』
事前定義メッセージは、メッセージ・ファイルに保管されます。

536 ページの『メッセージ』
メッセージは、ユーザーとプログラム間の通信に使用されます。

関連情報:

オブジェクト割り振り (ALCOBJ) コマンド

オブジェクト割り振り解除 (DLCOBJ) コマンド

データベース・プログラミング: レコードのロック

オブジェクトのロック状態

ロック状態 とは、オブジェクトの用途、およびそのオブジェクトが共用されるかどうかを識別するものです。

オブジェクトは、その使用目的 (読み取りまたは更新)、および共用 (複数のジョブによる同時使用) が可能かどうかに基づいて割り振られます。ファイルおよびメンバーは必ず *SHRRD で割り振られ、ファイル・データはロック状態で指定されたロック・レベルで割り振られます。ロック状態には、次の 5 種類があります (パラメーター値を括弧に入れて示してあります)。

- 占有 (*EXCL)。オブジェクトは要求元ジョブの占有使用として確保され、他のジョブはこのオブジェクトを使用することができません。ただし、オブジェクトが既に別のジョブに割り振られている場合には、他のジョブはそのオブジェクトを占有使用することはできません。実行している機能が完了するまで他のユーザーがオブジェクトにアクセスできないようにしたい場合には、このロック状態が必要です。
- 読み取り可能占有 (*EXCLRD)。オブジェクトはそれを要求したジョブに割り振られますが、他のジョブからもそのオブジェクトを読み取ることができます。他のユーザーがオブジェクトに対して読み取り以外の操作を行うことができないようにしたい場合には、このロック状態が必要です。
- 更新共用 (*SHRUPD)。オブジェクトは、更新についても読み取りについても、別のジョブとの共用が可能です。すなわち、別のユーザーが、同じオブジェクトに対して読み取り共用ロック状態または更新共用ロック状態のいずれかを要求することができます。オブジェクトを変更する処理で、他のユーザーによる同じオブジェクトの読み取りまたは変更を可能にしたい場合には、このロック状態が適切です。
- 更新不可共用 (*SHRNUP)。ジョブが、更新不可共用ロック状態または読み取り共用ロック状態を要求した場合には、そのオブジェクトを別のジョブと共用することができます。オブジェクトを変更する処理で、他のユーザーによるそのオブジェクトの変更をできないようにしたい場合には、このロック状態が必要です。
- 読み取り共用 (*SHRRD)。ユーザーがオブジェクトの占有使用を要求していなければ、オブジェクトは別のジョブと共用することができます。すなわち、別のユーザーは、そのオブジェクトに対して読み取り可能占有、更新共用、読み取り共用、または更新不可共用の各ロック状態を要求することができます。

注**:** ライブラリーの割り振りは、そのライブラリー内のオブジェクトに対して実行できる操作を制限するものではありません。すなわち、あるジョブによりライブラリーが読み取り可能占有ロック状態または更新共用ロック状態になった場合には、他のジョブはそのライブラリーにオブジェクトを入れたり、そのライブラリーからオブジェクトを除去したりすることはできません。ただし、この場合でも他のジョブはライブラリー内のオブジェクトを更新することができます。

次の表は、オブジェクトに対する有効なロック状態の組み合わせを示しています。

表 *31.* 有効なロック状態の組み合わせ

あるジョブのロック状態**:**	別のジョブのロック状態**:**
*EXCL	なし
*EXCLRD	*SHRRD
*SHRUPD	*SHRUPD または *SHRRD
*SHRNUP	*SHRNUP または *SHRRD
*SHRRD	*EXCLRD、*SHRUPD、*SHRNUP、または *SHRRD

ほとんどのオブジェクト・タイプに対して、5 つすべてのロック状態
(*EXCL、*EXCLRD、*SHRUPD、*SHRNUP、および *SHRRD) を指定することができます。ただし、このことはすべてのオブジェクト・タイプに当てはまるわけではありません。次の表には、5 つすべてのロック状態を指定することのできないオブジェクト・タイプが、そのオブジェクト・タイプに有効なロック状態とともにリストされています。

表 *32.* 特定のオブジェクト・タイプについて有効なロック状態

オブジェクト・タイプ	***EXCL**	***EXCLRD**	***SHRUPD**	***SHRNUP**	***SHRRD**
装置記述		x			
ライブラリー		x	x	x	x

表 32. 特定のオブジェクト・タイプについて有効なロック状態 (続き)

オブジェクト・タイプ	*EXCL	*EXCLRD	*SHRUPD	*SHRNUP	*SHRRD
メッセージ待ち行列	x				x
パネル・グループ	x	x			
プログラム	x	x			x
サブシステム記述	x				

ロック状態にないオブジェクト、または特定のロック状態を要求していないオブジェクトに対して DLCOBJ コマンドを発行した場合でも、エラーとはなりません。

オブジェクトのロック状態は、次の例に示すように、変更することができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
ALCOBJ  OBJ((FILEX *FILE *EXCL)) WAIT(0)
CALL PGMA
ALCOBJ  OBJ((FILEX *FILE *EXCLRD))
DLCOBJ  OBJ((FILEX *FILE *EXCL))
CALL PGMB
DLCOBJ  OBJ((FILEX *FILE *EXCLRD))
ENDPGM
```

ファイル FILEX は、PGMA に占有として割り振られますが、PGMB に対しては読み取り可能占有として割り振られます。

ファイル内のデータ・レコードを割り振るためには、レコード・ロックを使用することができます。 **ファイル作成 (CRTF)** コマンドの WAITFILE パラメーターを使用して、ユーザーのプログラムがファイルの割り振りを待つ時間 (タイムアウトが起こるまでの待機時間) を指定することもできます。

オブジェクトのロック状態の表示

オブジェクト・ロック処理 (WRKOBJLCK) コマンドまたは**ジョブ処理 (WRKJOB)** コマンドを使用して、オブジェクトのロック状態を表示することができます。

オブジェクト・ロック処理 (WRKOBJLCK) コマンドは、指定のオブジェクトに対してシステム内で要求されているすべてのロック状態を表示します。このコマンドでは、獲得されているロック状態と待機中のロック状態の両方が表示されます。データベース・ファイルの場合、**オブジェクト・ロック処理 (WRKOBJLCK)** コマンドは、ファイル・レベル (オブジェクト・レベル) のロックを表示しますが、レコード・レベルのロックは表示しません。例えば、データベース・ファイルが更新の目的でオープンされている場合には、そのファイル自体のロックは表示されますが、ファイル内のレコードについてのロック状態は表示されません。データベース・ファイルのメンバーに対するロックもまた、**オブジェクト・ロック処理 (WRKOBJLCK)** コマンドを使用して表示することができます。

オブジェクト・ロック処理 (WRKJOB) コマンドを使用した場合には、ジョブ表示メニューでロック・オプションを選択することができます。このオプションを選択すると、指定の活動ジョブの未処理のロック要求、ジョブで獲得されているロック、およびジョブで待機中のロックがすべて表示されます。ただし、ジョブがデータベース・レコードのロックを待機している場合であっても、その状況はオブジェクト・ロック表示画面には表示されません。

次のコマンドによって、論理ファイル ORDFILL に対するシステムでのロック要求がすべて表示されます。

```
WRKOBJLCK  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)
```

このコマンドによって表示される画面は次のとおりです。

```
                          オブジェクト・ロックの処理

システム :   SYSTEM01
 オブジェクト:   ORDFILL        ライブラリー:   QGPL           タイプ:   *FILE-LGL

 オプションを入力して，実行キーを押してください。
   4= ジョブの終了   5= ジョブの処理   8= ジョブ・ロックの処理

 有効
 OPT     ジョブ         ユーザー       ロック       状況      範囲      スレッド
  _      WORKST04       QSECOFR       *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB
                                      *SHRRD       HELD      *JOB

          続く...
 F3= 終了   F5= 最新表示   F6= メンバー・ロックの処理   F12= 取り消し
```

関連情報:

オブジェクト・ロック処理 (WRKOBJLCK) コマンド

ジョブ処理 (WRKJOB) コマンド

CL プログラム内のオブジェクトへのアクセス

CL プログラムからオブジェクトにアクセスするには、そのオブジェクトを参照するコマンドを実行しているときに、オブジェクトは指定したライブラリーに存在する必要があります。

CL プログラム・コマンドおよびプロシージャーでオブジェクトを参照する場合の規則は、プログラム中ではなく単独で実行されるコマンドでオブジェクトを参照する場合と同じです。オブジェクト名は、修飾して指定することも、修飾しないで指定することもできます。修飾されていないオブジェクト名は、ライブラリー・リストを走査することによって探索されます。

CL プロシージャーおよびプログラムで参照されるオブジェクトの大部分は、それを参照するコマンドが実行される時点で初めてアクセスされます。オブジェクトの名前を修飾する (ライブラリー / オブジェクト名) 場合、そのオブジェクトは、それを参照するコマンドが実行される時点では指定のライブラリーに存在していなければなりません。しかし、プログラムを作成する時点ではそのライブラリーに存在している必要はありません。したがって、ほとんどのオブジェクトの場合、実行時のライブラリーに基づいて CL ソース・ステートメントに修飾名を指定することができます。

CL ソース・ステートメントでオブジェクト名を修飾せずに、ライブラリー・リストを参照する (*LIBL/オブジェクト名) ようにすれば、すべてのオブジェクトについて、このような実行時に関する考慮事項に注意を払う必要はなくなります。コンパイル時にライブラリー・リストを参照する場合には、コマンドの実行時にライブラリー・リストに入っているライブラリーであれば、オブジェクトはどのライブラリーに入っていても構いません。ただし、同じ名前を持つオブジェクトが異なるライブラリーに入っていないことが前提条件となります。ライブラリー・リストを使用する場合には、プロシージャー作成からコマンド実行までの間に、オブジェクトを別のライブラリーに移すこともできます。

オブジェクトはそのオブジェクトを参照するコマンドが実行されるまで存在する必要はないので、この CL プログラムは、プログラム PAYROLL がコンパイル時に存在しないとしても、正常にコンパイルされます。

```
PGM /*TEST*/
DCL...
MONMSG...
.
.
.
CALL PGM(QGPL/PAYROLL)
.
.
.
ENDPGM
```

つまり PAYROLL は、プログラム TEST が活動化される時点では存在している必要はなく、**プログラム呼び出し (CALL)** コマンドが実行される時点で存在していればよいわけです。次の場合、呼び出されるプログラムは、**プログラム呼び出し (CALL)** コマンドの直前に呼び出し側プログラム内で作成されます。

```
PGM /*TEST*/
DCL...
.
.
.
MONMSG
.
.
.
CRTCLPGM PGM(QGPL/PAYROLL)
CALL PGM(QGPL/PAYROLL)
.
.
.
ENDPGM
```

制御言語プログラム作成 (CRTCLPGM) や**データ域作成 (CRTDTAARA)** などの作成コマンドの場合、コンパイル時または実行時にアクセスされるオブジェクトは作成コマンドの定義であって、作成されるオブジェクトではないことに注意してください。 作成コマンドを使用する場合には、その作成コマンドの定義はコンパイル時にそのコマンドの修飾に使用するライブラリーに存在していなければなりません。 (*LIBL を使用した場合は、そのファイルはライブラリー・リスト内のいずれかのライブラリーに入っていなければなりません。)

関連概念:

181 ページの『CL ソース・プログラムの構成要素』
CL ソース・プログラムの構成要素として入力する各ソース・ステートメントが実際には CL コマンドであっても、このソースは、多数の典型的な CL ソース・プログラムで使用される基本的な構成要素に類別できます。

コマンド定義、ファイル、およびプロシージャーへのアクセス

CL プログラムからコマンド定義またはファイルにアクセスするには、その制御言語プログラム作成時、およびそのコマンド定義またはファイルを参照しているコマンドの実行時に、そのコマンド定義またはファイルが存在している必要があります。

コマンド定義またはファイルを参照するソース・ステートメントから CL プログラムを作成する場合、次の 2 つの要件を満たす必要があります。

- オブジェクトはプログラムの作成時に存在していなければならない。

- オブジェクトは、そのオブジェクトを参照するコマンドが実行されるときに存在していなければならない。

したがって、ファイル宣言 (DCLF) コマンドを使用する場合には、ファイルを参照するプログラムを作成する前に、そのファイルを作成しておく必要があります。

コマンド定義へのアクセス:

コマンド定義は、作成時およびコマンドの実行時にアクセスされます。

コマンドは、構文検査を行うために、そのコマンドを使用するプログラムが作成される時点で存在していなければなりません。作成時にコマンド定義を修飾した場合には、そのコマンドは作成中に参照されたライブラリーに存在していなければならず、処理時にも同じライブラリーに存在していなければなりません。ライブラリーの修飾名を使用しなかった場合には、作成時にも実行時にも、ライブラリー・リスト内のいずれか 1 つのライブラリーに存在していれば、どのライブラリーに存在していても構いません。

次の場合に、コマンド名は修飾名でなければなりません。

- プログラムの実行中に、コマンドの定義にライブラリー・リストを通じてアクセスできない場合。
- 実行時にコマンドの特定のインスタンスを必要としている状況で、同じ名前を持つ複数のコマンド定義が存在している場合。

コマンドの名前は、プログラムの作成時と実行時で同じでなければなりません。コマンドを参照するプログラムを作成した後でそのコマンドの名前を変更すると、エラーが起こります。これは、プログラムの実行時にそのコマンドを見つけることができなくなるためです。ただし、コマンドのパラメーターのデフォルト値を変更すると、コマンドの実行時には新しいデフォルト値が使用されます。

関連情報:

コマンド変更 (CHGCMD) コマンド

ファイルへのアクセス:

CL モジュールまたはオリジナル・プログラム・モデル (OPM) プログラムをコンパイルするときには、それらのモジュールまたはプログラムが使用するファイルが存在していなければなりません。

ファイル宣言 (DCLF) コマンドが含まれているプログラム・モジュールをコンパイルすると、コンパイラーはファイルにアクセスします。ファイルは、モジュールを使用するプログラムまたはサービス・プログラムを作成するときには、存在しなくても構いません。

ファイルを作成するには、まずデータ記述仕様 (DDS) をソース・ファイルに入力します。 DDS は、レコード様式およびレコード中のフィールドを記述するためのもので、 **表示装置ファイル作成 (CRTDSPF)** コマンドを使用することによりその情報がコンパイルされ、ファイル・オブジェクトが作成されます。

注: DDS を使用して他のタイプのファイルを作成することもできます。どのタイプにもそれぞれ専用のコマンドがあります。 **物理ファイル作成 (CRTPF)** コマンドと**論理ファイル作成 (CRTLF)** コマンドは、CL プログラムおよびプロシージャーで使用できるファイルを作成するためのコマンドです。

DDS に記述するフィールドは、入力フィールドと出力フィールドのどちらか (または両方) であり、プログラムまたはモジュールのコンパイル時に、システムは CL プログラムまたはプロシージャー内のフィールドを変数として宣言します。これらの変数を通じて、画面からのデータがプログラムにより処理されます。

物理ファイルの作成に DDS を使用しなかった場合、システムは CL 変数を宣言してレコード全体を収容します。この変数の名前はファイルの名前と同じであり、またその長さはファイルのレコード長と同じです。

CL プログラムおよびプロシージャーは、特殊な CL コマンドを使用する場合を除き、表示装置ファイルおよびデータベース・ファイル以外のどのようなファイル・タイプのデータも処理できません。

ファイルの作成後は、DDS を削除することもできますが、これはあまりお勧めできません。ファイルを参照する CL プログラムまたはモジュールをシステムがコンパイルした後は、そのファイルを削除することもできます。そのファイルを参照するコマンド (**ファイル受信 (RCVF)** など) がプログラム内で実行されるときに、そのファイルが存在する場合に、このことが当てはまります。

コマンド定義について説明した修飾名に関する規則は、ファイルにも当てはまります。

関連タスク:

519 ページの『CL プログラムまたはプロシージャーでのファイルの処理』
CL プロシージャーおよびプログラムでは、表示装置ファイルとデータベース・ファイルの 2 つのタイプのファイルがサポートされています。

プロシージャーへのアクセス:

バインド・プロシージャーの呼び出し (CALLPRC) によって指定されたプロシージャーは、それを参照するモジュールの作成時に存在していなくても構いません。

そのプロシージャーを使用するプログラムやサービス・プログラムを作成するために、そのプロシージャーが存在する必要はありません。呼び出されるプロシージャーは、次のいずれかの位置に存在することができます。

- **プログラム作成 (CRTPGM)** または**サービス・プログラム作成 (CRTSRVPGM)** コマンドの MODULE パラメーターで指定されているモジュール内。
- BNDSRVPGM パラメーターで指定されているサービス・プログラム内。サービス・プログラムは、実行時に使用可能でなければなりません。
- **CRTPGM** コマンドまたは **CRTSRVPGM** コマンドの BNDDIR パラメーターで指定されたバインド・ディレクトリー内にリストされている、サービス・プログラムまたはモジュール内。バインド・ディレクトリーとモジュールは、実行時に使用可能でなくても構いません。

オブジェクトの存在の検査

プログラムでオブジェクトを使用する前に、そのオブジェクトが存在しているかどうか、およびそれを使用するために必要な権限があるかどうかを判別するための検査を行ってください。

この検査は、1 つの機能が同時に複数のオブジェクトを使用する場合に役立ちます。

オブジェクトの存在を検査するためには、**オブジェクト検査 (CHKOBJ)** コマンドを使用します。このコマンドは、プロシージャーまたはプログラムのどこで使用しても構いません。 **オブジェクト検査 (CHKOBJ)** コマンドの形式は次のとおりです。

```
CHKOBJ  OBJ(ライブラリー名 / オブジェクト名)  OBJTYPE(オブジェクト・タイプ)
```

オプションの他のパラメーターを使用すれば、オブジェクト権限の検査を行うことができます。権限の有無を検査してファイルをオープンしたい場合には、操作権限とデータ権限の両方を検査しなければなりません。

このコマンドを実行すると、オブジェクトの検査の結果を報告するメッセージがプログラムまたはプロシージャーに送信されます。そのメッセージを監視し、必要な処理を行うことができます。

次の例では、**メッセージ・モニター (MONMSG)** コマンドを使用して、「オブジェクトが見つからない」ことを示すエスケープ・メッセージだけを監視しています。 **オブジェクト・チェック (CHKOBJ)** コマンドの実行結果として出されるすべてのメッセージのリストについては、**オブジェクト検査 (CHKOBJ)** コマンドのオンライン・ヘルプ情報を参照してください。

オブジェクト検査 (CHKOBJ) コマンドでは、オブジェクトの割り振りは行われません。アプリケーションでは多くの場合、オブジェクトの存在の検査だけでは十分とは言えず、アプリケーション側でオブジェクトの割り振りを行うことが必要です。 **オブジェクト割り振り (ALCOBJ)** コマンドには、オブジェクトの存在の検査とオブジェクトの割り振りの両方の機能があります。

テープ検査 (CHKTAP) コマンドか**ディスケット検査 (CHKDKT)** コマンドを使用することにより、特定のテープやディスケットが駆動機構に設定されて使用可能になっているかどうかを確認することができます。これらのコマンドによって、CL プログラムで監視できるエスケープ・メッセージも発行されます。

```
          CHKOBJ  OBJ(OELIB/PGMA)  OBJTYPE(*PGM)
          MONMSG  MSGID(CPF9801)  EXEC(GOTO NOTFOUND)
          CALL OELIB/PGMA
          .
          .
          .
NOTFOUND: CALL FIX001 /*PGMA Not Found Routine*/
          ENDPGM
```

関連タスク:

537 ページの『メッセージ記述の定義』
事前定義メッセージは、メッセージ・ファイルに保管されます。

536 ページの『メッセージ』
メッセージは、ユーザーとプログラム間の通信に使用されます。

関連資料:

254 ページの『**メッセージ・モニター** コマンド』
メッセージ・モニター (MONMSG) コマンドは、**MONMSG** コマンドが使用されている CL プログラムまたはプロシージャーの呼び出しスタックに送られてくる、エスケープ・メッセージ、通知メッセージ、または状況メッセージの監視に使用されます。

関連情報:

オブジェクト検査 (CHKOBJ) コマンド

CL プログラムまたはプロシージャーでのファイルの処理

CL プロシージャーおよびプログラムでは、表示装置ファイルとデータベース・ファイルの 2 つのタイプのファイルがサポートされています。

ユーザーは、表示内容をワークステーションに送り、ワークステーションからプロシージャーまたはプログラムで使用する入力を受け取ることができます。また、プロシージャーまたはプログラムで使用するデータをデータベース・ファイルから読み取ることができます。

注**:** データベース・ファイルを CL プロシージャーやプログラムで使用できるようにするためには、DCLF コマンドおよび**ファイル受信 (RCVF)** コマンドを使用します。

CL プロシージャーまたはプログラムでファイルを使用する場合には、次の操作を行わなければなりません。

- DDS ソースを使用してフィールドと条件を記述し、表示レコードまたはデータベース・レコードの様式を定義する。データベース・ファイルの場合には、DDS の使用は必須ではありません。
- **表示装置ファイル作成 (CRTDSPF)**、**物理ファイル作成 (CRTPF)**、または**論理ファイル作成 (CRTLF)** コマンドを使用してそのファイルを作成する。サブファイル (メッセージ・サブファイルを除く) は、CL プロシージャーおよびプログラムでは使用できません。
- データベース・ファイルの場合、**物理ファイル・メンバー追加 (ADDPFM)** コマンドか
  論理ファイル・メンバー追加 (ADDLFM)

  コマンドを使用してファイルにメンバーを追加する。メンバーを CRTPF コマンドか CRTLF コマンドで追加した場合には、この操作は不要です。プロシージャーやプログラムの実行時にはファイル内にメンバーが存在していなければなりませんが、その作成時にはメンバーが存在している必要はありません。
- CL プロシージャーでは DCLF コマンドを使用して該当のファイルを参照し、CL ソースでは適切なデータ操作用の CL コマンドでそのレコード様式を参照する。
- CL モジュールを作成する。
- プログラムまたはサービス・プログラムを作成する。

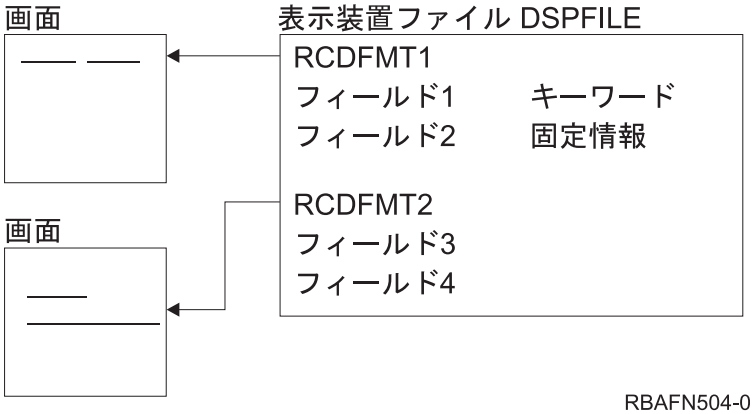
1 つの CL プロシージャーでは、最大で 5 つの表示装置ファイルまたはデータベース・ファイルを参照できます。データベース・ファイルと表示装置ファイルには、同じコマンドが使用されるので、この 2 つのファイルに対するサポートは類似しています。ただし、次に示すような多少の相違があります。

- 次の事項は、CL プロシージャーおよびプログラムで使用されるデータベース・ファイルにだけ当てはまります。
  - 1 つのレコード様式を持ったデータベース・ファイルだけが CL プロシージャーまたはプログラムで使用できる。
  - ファイルは、物理ファイルまたは論理ファイルのいずれでもよく、また論理ファイルは、複数の物理ファイル・メンバーを基礎として定義されていてもよい。
  - RCVF コマンドを使用した入力操作だけが実行できる。データベース・ファイルの場合には、**ファイル受信 (RCVF)** コマンドの WAIT および DEV の各パラメーターは使用することはできません。さらに、SNDF、SNDRCVF、および ENDRCV コマンドはデータベース・ファイルの場合に使用することはできません。
  - DDS は、CL プロシージャーまたはプログラムで参照される物理ファイルの作成に必須ではない。物理ファイルの作成に DDS を使用しなかった場合には、ファイルのレコード様式はファイルと同じ名前になり、そのレコード様式にファイルと同じ名前でファイルのレコード長 (CRTPF コマンドの RCDLEN パラメーター) と同じ長さのフィールドが 1 つだけ入ります。
  - ファイルには、モジュールまたはプログラム用に作成される時点ではメンバーが存在していなくても構わない。ただし、そのファイルをプログラムが処理するときには、メンバーが存在していなければなりません。
  - 最初の**ファイル受信 (RCVF)** コマンドが処理された時点で、ファイルは入力専用としてオープンされる。この時点では、ファイルおよびそのメンバーが存在していなければなりません。
  - ファイルは、プロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムが制御を戻すかファイルの終わりに達するまで、オープンされたままになる。ファイルの終わりに達すると、CL プロシージャーまたはプログラムにメッセージ CPF0864 が送られ、そのファイルに対してそれ以上の操作はできなくなります。したがって、プロシージャーまたはプログラムでこのメッセージを監視し、ファイルの終わりに達した時点で適切な処置を取るようにコーディングしなければなりません。

- 次の事項は、CL プロシージャーおよびプログラムで使用される表示装置ファイルにだけ当てはまります。
  - 表示装置ファイルには、最高 99 個のレコード様式を指定することができる。
  - 表示装置ファイルに対しては、すべてのデータ操作コマンド (SNDF、SNDRCVF、RCVF、ENDRCV、および WAIT) を使用することができる。
  - 表示装置ファイルは DDS を使用して定義しなければならない。
  - 表示装置ファイルは、最初の SNDF コマンド、SNDRCVF コマンド、または RCVF コマンドが実行される時点で、入出力共用としてオープンされる。ファイルは、プロシージャーまたは OPM プログラムが制御を戻すまで、オープンされた状態になります。

注: 最初の送信コマンドまたは受信コマンドが実行されるまでは、どちらのタイプのファイルもオープンされません。したがって、最初の送信または受信の前であれば、使用するファイルをプロシージャーやプログラムの実行過程で作成することも、一時変更することもできます。ただし、ファイルは、モジュールまたはプログラムのコンパイル前に存在していなければなりません。

表示画面のレコード様式は、DDS でレコード様式として指定します。各レコード様式には、フィールド (入力、出力、および入出力共用)、条件か標識、および固定情報を入れることができ、 1 つの表示装置ファイルには複数のレコード様式を入れることができます。表示装置ファイル名、レコード様式名、およびフィールド名は固有のものでなければなりません。これは CL プロシージャーおよびプログラムでは必須条件ではありませんが、他の高水準言語 (HLL) で固有性が要求されることがあるためです。



RBAFN504-0

関連概念:

181 ページの『CL ソース・プログラムの構成要素』
CL ソース・プログラムの構成要素として入力する各ソース・ステートメントが実際には CL コマンドであっても、このソースは、多数の典型的な CL ソース・プログラムで使用される基本的な構成要素に類別できます。

関連タスク:

517 ページの『ファイルへのアクセス』
CL モジュールまたはオリジナル・プログラム・モデル (OPM) プログラムをコンパイルするときには、それらのモジュールまたはプログラムが使用するファイルが存在していなければなりません。

関連情報:

CL コマンド検索プログラム

Application Display Programming

データ操作コマンド

CL プロシージャーまたはプログラムでは、データ操作コマンドと呼ばれるいくつかのコマンドを使用することができます。

これらのコマンドを使用すると、表示装置ファイルを参照してディスプレイ装置との間でデータのやりとりを行うことができます。また、これらのコマンドを使用してデータベース・ファイルを参照し、データベース・ファイルからレコードを読み取ることもできます。データ操作コマンドには次のものがあります。

ファイル宣言 (DCLF)
: プロシージャーやプログラムで使用する表示装置ファイルまたはデータベース・ファイルを定義するためのコマンドです。ファイル内のフィールドは、プロシージャーまたはプログラムで使用する変数として自動的に宣言されます。

ファイル送信 (SNDF)
: データをディスプレイ装置に出力するためのコマンドです。

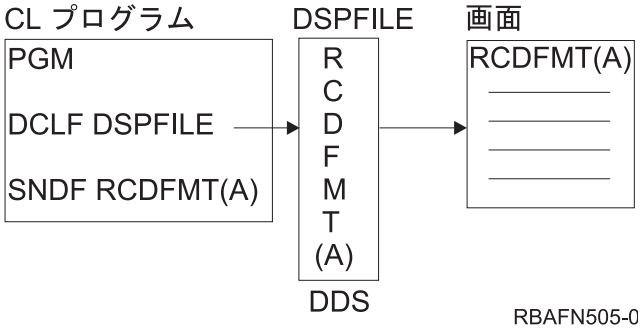
ファイル受信 (RCVF)
: ディスプレイ装置またはデータベースからデータを読み取るためのコマンドです。

ファイル送信/受信 (SNDRCVF)
: ディスプレイ装置にデータを送って入力を要求し、必要に応じてディスプレイ装置からのデータを読み取るためのコマンドです。

表示装置ファイル一時変更 (OVRDSPF)
: プロシージャーまたはプログラムで使用する表示装置ファイルを実行時に一時変更するためのコマンドです。

データベース・ファイル一時変更 (OVRDBF)
: プロシージャーまたはプログラムで使用するデータベース・ファイルを実行時に一時変更するためのコマンドです。

これらのコマンドを使用することにより、実行中の CL プログラムは DDS で定義されている表示機能を使用してディスプレイ装置との間でデータのやりとりを行い、またデータベース・ファイルからレコードを読み取ることができます。 DDS には、メニューの作成と、多くの CL アプリケーションの特徴である基本的なアプリケーション・データの要求を行うための機能が備わっています。



表示画面またはレコードのフィールドは、ファイルに対する DDS で指定します。これらのフィールドを CL プロシージャーまたはプログラムで使用するためには、CL プロシージャーまたはプログラム中の **ファイル宣言 (DCLF)** コマンドで該当のファイルを参照しなければなりません。この参照によって、ファイル内のフィールドおよび標識は、変数としてプロシージャーまたはプログラム内で自動的に宣言されます。こ

れらの変数は CL コマンドで自由に使用することができますが、その主な目的は、ディスプレイ装置との間で情報のやりとりをすることにあります。 **ファイル宣言 (DCLF)** コマンドは実行時には使用されません。

表示画面の様式および各フィールドに対するオプションは、装置ファイルに指定されて標識を使って制御されます。 DDS および CL サポートでは、最高 99 個まで標識の値を使用することができます。標識変数は、**ファイル宣言 (DCLF)** コマンドで参照される装置ファイルのレコード様式に定義された各標識に対応して、 &IN01 から &IN99 までの名前を持つ論理変数として CL プロシージャーまたはプログラムの中で宣言されます。標識を使用することによって、フィールドの表示やデータ管理表示機能の制御を行うことができ、表示装置ファイルからプロシージャーまたはプログラムに応答情報を与えることもできます。標識は、データベース・ファイルには使用できません。

関連情報:

CL コマンド検索プログラム

CL プログラムまたはプロシージャー内のファイル

ファイルの各フィールドに対応する変数を宣言できるように、CL モジュールおよびプログラムの作成過程で **ファイル宣言 (DCLF)** コマンドをコンパイルする際にファイルがアクセスされます。

コンパイル時にファイルの名前を修飾した場合、実行時にも同じライブラリーに入っていなければなりません。また、コンパイル時にライブラリー・リストを使用した場合は、そのファイルは実行時にライブラリー・リスト内のいずれかのライブラリーに入っていなければなりません。

関連情報:

ファイル宣言 (DCLF) コマンド

CL プログラムまたはプロシージャー内のファイルのオープンおよびクローズ

ファイルがオープンおよびクローズされる時間と方法がわかれば、実行中のプログラムまたはプロシージャー間でオープン・データ・パスを共用できます。

CL サポートを使用する場合、参照されたファイルは、最初の送信操作、受信操作、送信 / 受信操作を実行した時点で暗黙的にオープンされます。オープンされた表示装置ファイルは、それをオープンしたプロシージャーやオリジナル・プログラム・モデル (OPM) プログラムが制御を戻すか、あるいは他のプロシージャーやプログラムに制御を移すまで、オープンされた状態になっています。オープンされたデータベース・ファイルは、ファイルの終わりに達するか、そのファイルをオープンしたプロシージャーや OPM プログラムが制御を戻すか、または他のプロシージャーやプログラムに制御を移した時点でクローズされます。データベース・ファイルはいったんクローズされると、そのプロシージャーまたは OPM プログラムと同じ呼び出し中に、再度オープンすることはできません。

データベース・ファイルがオープンされると、事前に **データベース一時変更 (OVRDBF)** コマンドにより別のメンバー (MBR パラメーター) が指定されていない限り、ファイルの最初のメンバーがオープンされます。エラーによりプロシージャーまたは OPM プログラムが終了した場合には、ファイルもクローズされます。ファイルは、それがオープンされたプロシージャーまたは OPM プログラムが終了するまで、オープンされた状態になっているので、実行中のプロシージャーおよびプログラム相互間で、容易にそのオープン・データ・パスを共用することができます。次のどちらかの条件を満たしている場合には、1 つのプロシージャーまたはプログラムでファイルをオープンし、そのオープン・データ・パスを別のプロシージャーまたはプログラムと共用することができます。

- ファイルが SHARE(*YES) 属性を指定して作成されているか、またはその属性に変更されている。
- SHARE(*YES) を指定した一時変更が、そのファイルに対して有効である。

このようにして、任意の 2 つのプロシージャーまたはプログラム相互間でファイルを共用することができます。システムでオープン・データ・パスを共用する場合の、使用可能な機能に関する詳細については、オンライン・ヘルプを使用してください。さらに IBM は、**表示装置ファイル作成 (CRTDSPF)**、**物理ファイル作成 (CRTPF)**、および**論理ファイル作成 (CRTLF)** の各コマンドでの SHARE パラメーターについての説明をオンラインで提供しています。 CL プロシージャーまたは OPM プログラムでオープンされる場合、表示装置ファイルは常に入出力共用としてオープンされ、データベース・ファイルは入力専用でオープンされます。

ファイルを使用する CL プロシージャーおよびプログラムでは、 **資源再利用 (RCLRSC)** コマンドに LVL(*CALLER) を指定してはなりません。 LVL(*CALLER) を指定すると、プロシージャーまたは OPM プログラムによりオープンされたすべてのファイルが直ちにクローズされ、ファイルへのアクセスを行うと異常終了することになります。

ファイル宣言

ファイル宣言 (DCLF) コマンドは、表示装置ファイルまたはデータベース・ファイルを CL プロシージャーまたはプログラムに対して宣言するために使用します。 **ファイル宣言 (DCLF)** コマンドを使用して、テープ装置ファイル、印刷装置ファイル、および混合ファイルなどのファイルを宣言することはできません。

1 つの CL プロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムでは、最大で 5 つのファイルを宣言できます。 **DCLF** コマンドには次のパラメーターがあります。

```
DCLF   FILE(ライブラリー名 / ファイル名)
       RCDFMT(レコード様式名)
       OPNID(open_id_name)
```

ファイルは、モジュールまたはプログラムのコンパイル前に存在していなければならないことに注意してください。

プロシージャーまたはプログラムで表示装置ファイルを使用する場合に、DDS で入出力共用フィールドを指定しなければならないことがあります。これらのフィールドは、プロシージャーまたはプログラムでは変数として処理されます。 **ファイル宣言 (DCLF)** コマンドを処理すると、CL コンパイラーは、ファイルの各レコード様式に定義されている個々のフィールドおよびオプション標識に対応する CL 変数を宣言します。フィールドの場合には、フィールド名の前にアンパーサンド (&) を付けたものが CL 変数名になります。また、オプション標識の場合には、標識の前に &IN を付けたものが CL 変数名になります。

オープン・ファイル ID (OPNID) パラメーターを使用すれば、宣言済みファイルのインスタンスを一意的に識別でき、複数のファイルを宣言できます。 OPNID パラメーターを使用する場合、フィールドでは、フィールド名の前にアンパーサンド (&) と OPNID 値と下線 (_) を付けたものが CL 変数名になります。オプション標識では、標識の前にアンパーサンド (&)、OPNID 値、下線、および "IN" を付けたものが CL 変数名になります。

例えば、INPUT という名前のフィールドおよび標識 10 が **ファイル宣言 (DCLF)** で定義されている場合には、DCLF コマンドにより、フィールドは &INPUT として、標識は &IN10 として自動的に宣言されます。この宣言は、CL モジュールまたはプログラムのコンパイル時に行われます。 1 つのコマンドには最高 50 個のレコード様式名を指定することができますが、変数を指定することはできません。データベース・ファイルの場合には、指定できるレコード様式は 1 つだけです。

ライブラリー MCGANN に表示装置ファイル CNTRLDSP を作成するために、次の DDS を入力したと想定します。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MASTER
     A                                      CA01(01 'F1 RESPONSE')
     A            TEXT         300      2  4
     A            RESPONSE      15   1  8  4 BLINK
     A
     A
```

この表示装置ファイルから 3 つの変数 (&IN01、&TEXT、&RESPONSE) を使用できるようになります。このファイルを参照する CL プロシージャーでは、次の DCLF ソース・ステートメントを入力するだけです。

```
   DCLF MCGANN/CNTRLDSP
```

コンパイラーはこのステートメントを展開して、表示装置ファイルに対応するすべての変数を個別に宣言します。コンパイラー・リストに示される展開された宣言は次のようになります。

```
  •
  •
  •
   00500- DCLF(MCGANN/CNTRLDSP)
 04/02/03
          QUALIFIED FILE NAME - MCGANN/CNTRLDSP
             RECORD FORMAT NAME - MASTER
               CL VARIABLE               TYPE     LENGTH     PRECISION (IF *DEC)
                &IN01                    *LGL        1
                &TEXT                    *CHAR     300
                &RESPONSE                *CHAR      15
  •
  •
  •
```

以下のように DCLF ソース・ステートメントに OPNID パラメーターが含まれる場合、

```
DCLF MCGANN/CNTRLDSP OPNID(OPENID1)
```

コンパイラー・リストに示される展開された宣言は次のようになります。

```
  •
  •
  •
   00500- DCLF FILE(MCGANN/CNTRLDSP) OPNID(OPENID1)                  04/02/03
          QUALIFIED FILE NAME - MCGANN/CNTRLDSP
             RECORD FORMAT NAME - MASTER
               CL VARIABLE               TYPE     LENGTH     PRECISION     TEXT
                &OPENID1_IN01            *LGL        1
                &OPENID1_TEXT            *CHAR     300
                &OPENID1_RESPONSE        *CHAR      15
  •
  •
  •
```

関連情報:

ファイル宣言 (DCLF) コマンド

表示装置ファイルによるデータの送信および受信

表示装置ファイルに対するデータの送信 (書き込み) や受信 (読み取り) を行うために CL プロシージャーおよびプログラムで使用できるコマンドは、**ファイル送信 (SNDF)**、**ファイル受信 (RCVF)**、および **ファイル送信/受信 (SNDRCVF)** の 3 つのコマンドだけです。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

ファイル送信 (SNDF) コマンドを実行すると、レコード様式の出力フィールドまたは入出力共用フィールドに対応する変数の内容をシステムは形式設定し、それをディスプレイ装置に送信して表示します。これは **ファイル受信 (RCVF)** コマンドを実行した場合も同様です。表示画面上のレコード様式の入力フィールドまたは入出力共用フィールドの値が読み取られ、対応する CL 変数に入れられます。

ファイル送信/受信 (SNDRCVF) コマンドは、CL 変数の内容をディスプレイ装置に送り、次に RCVF コマンドと同じ機能を果たすことによって、更新されたフィールドを表示画面から読み取ります。 CL ではゾーン 10 進数はサポートされないことに注意してください。したがって、表示装置ファイル内でゾーン 10 進数として定義されているフィールドは、CL プロシージャーまたはプログラムでは *DEC フィールドとして定義されます。 *DEC フィールドはパック 10 進数として内部的にサポートされ、パック 10 進データとゾーン 10 進データの間のタイプの変換は CL コマンドにより行われます。同一の表示位置を別個に指定することによって表示装置ファイル内のフィールドがオーバーラップした場合でも、オーバーラップしない別個の CL 変数が定義されます。浮動小数点データを含むレコード様式は、CL プロシージャーまたはプログラムでは使用できません。

注: ワークステーションに対する **ファイル送信/受信 (SNDRCVF)** コマンドまたは **ファイル受信 (RCVF)** コマンドに WAIT(*NO) を指定した場合、システムは WAIT コマンドを使用してデータの受信を行います。 INVITE DDS キーワードを含むレコード様式を用いる **ファイル送信 (SNDF)** コマンドが発行された場合も同様です。

メッセージ・サブファイルは別として、サブファイル・レコードの送信や受信を行おうとすると、実行時エラーが起こります。 DDS で表示装置ファイルに指定するその他の機能はほとんど使用できますが、使用できないものもいくつか (可変開始行番号を使用する機能など) あります。

次の例は、典型的なオペレーター・メニューを作成し、**ファイル送信/受信 (SNDRCVF)** コマンドを使用してデータの送信と受信を行うために必要なステップを示しています。メニューは次のとおりです。

```
オペレーター・メニュー

  1. 買掛管理
  2. 売掛管理
  90. サインオフ



オプション:
```

まず、次の DDS ソースを入力します。レコード様式は MENU で、OPTION は入力可能フィールドです。 OPTION フィールドには DSPATR(MDT) が指定されています。これにより、オペレーターが何も入力しない場合でも、このフィールドの値が有効かどうかの検査がシステムにより行われます。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MENU
     A                                  1  2'オペレーター・メニュー'
     A                                  3  4'1. 買掛管理'
     A                                  5  4'2. 売掛管理'
     A                                  5  4'90. サインオフ'

     A                                  7  2'オプション'
     A            OPTION         2Y 01   + 2VALUES(1 2 90) DSPATR(MDT)
     A
     A
```

CRTDSPF コマンドを入力して、この表示装置ファイルを作成します。 CL プログラミングでは、RPG for IBM i などの他の言語の場合と異なり、表示装置ファイル名 (この例では INTMENU) とレコード様式 (この例では MENU) を同じにすることもできます。

表示装置ファイルは、画面設計機能 (SDA) を使用して作成することもできます。

次に、メニューを実行するための CL ソースを入力します。

このメニューの CL ソースは次のとおりです。

```
         PGM /* OPERATOR MENU */
         DCLF INTMENU
BEGIN:   SNDRCVF RCDFMT(MENU)
         IF COND(&OPTION *EQ 1) THEN(CALL ACTSPAYMNU)
         IF COND(&OPTION *EQ 2) THEN(CALL ACTSRCVMNU)
         IF COND(&OPTION *EQ 90) THEN(SIGNOFF)
         GOTO BEGIN
         ENDPGM
```

このソースをコンパイルすると、DCLF コマンドにより入力フィールド OPTION がプロシージャーに CL 変数として自動的に宣言されます。

ファイル送信/受信 (SNDRCVF) コマンドでは、デフォルト値の WAIT(*YES) が使用されます。したがって、このプログラムは入力がプログラムにより受け取られるまで待機します。

関連タスク:

536 ページの『メッセージ』
メッセージは、ユーザーとプログラム間の通信に使用されます。

関連情報:

ファイル送信 (SNDF) コマンド

ファイル受信 (RCVF) コマンド

ファイル送信/受信 (SNDRCVF) コマンド

例: メニューを制御する CL プログラムまたはプロシージャーの作成

この例では、メニューを表示して制御するための CL プログラムまたはプロシージャーの作成方法を示しています。

この例では、CL プロシージャー ORD040C を作成します。このプロシージャーは、受注管理部門の汎用メニューの表示を制御し、メニューで選択されたオプションに基づいて、どの高水準言語 (HLL) プロシージャーを呼び出せばよいのかを判断するためのものです。このプロシージャーにより、ディスプレイ装置にメニューが表示されます。

この受注管理部門の汎用メニューは次のとおりです。

```
受注管理部門の汎用メニュー

  1 得意先ファイル照会
  2 品目ファイル照会
  3 得意先名検索
  4 得意先オーダー照会
  5 既存のオーダー照会
  6 オーダーの入力
 98 メニュー終了

オプション:
```

表示装置ファイル ORD040CD の DDS は次のとおりです。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* MENU ORDO4OCD ORDER DEPT GENERAL MENU
     A
     A          R MENU                       TEXT('General Menu')
     A                                    1  2'受注管理部門の汎用メニュー'
     A                                    3  3'1 得意先ファイル照会'
     A                                    4  3'2 品目ファイル照会'
     A                                    5  3'3 得意先名検索'
     A                                    6  3'4 得意先オーダー照会'
     A                                    7  3'5 既存のオーダー照会'
     A                                    8  3'6 オーダーの入力'
     A                                    9  2'98 メニュー終了'
     A                                   11  2'オプション'
     A            RESP           2Y001 11 10VALUES(1 2 3 4 5 6 98)
     A                                       DSPATR(MDT)
     A
     A
```

ORD040C のソース・プロシージャーは次のとおりです。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
       PGM /* ORD040C Order Dept General Menu */
       DCLF FILE(ORD040CD)
START: SNDRCVF RCDFMT(MENU)
       SELECT
         WHEN (&RESP=1) THEN(CALLPRC CUS210)  /* Customer inquiry */
         WHEN (&RESP=2) THEN(CALLPRC ITM210)  /* Item inquiry     */
         WHEN (&RESP=3) THEN(CALLPRC CUS220)  /* Cust name search */
         WHEN (&RESP=4) THEN(CALLPRC ORD215)  /* Orders by cust   */
         WHEN (&RESP=5) THEN(CALLPRC ORD220)  /* Existing order   */
         WHEN (&RESP=6) THEN(CALLPRC ORD410C) /* Order entry      */
         WHEN (&RESP=98) THEN(RETURN)         /* End of Menu      */
       ENDSELECT
       GOTO START
       ENDPGM
```

DCLF コマンドは、**ファイル送信/受信 (SNDRCVF)** コマンドの実行時に、受注管理部門汎用メニューを様式化するためにシステムで必要となるフィールド属性がどのファイルに入っているかを示しています。**ファイル送信 (SNDF)**、**ファイル受信 (RCVF)**、または**ファイルの送信/受信 (SNDRCVF)** のいずれかのコマンドでレコード様式が使用されている場合には、指定のファイルのレコード様式内の各フィールドに対応する変数がシステムにより自動的に宣言されます。自動的に宣言される各フィールドの変数名は、フィールド名の前にアンパーサンド (&) を付けたものです。例えば、ORD040C の応答フィールド RESP の変数名は &RESP となります。

このメニューの操作についてのその他の注意事項は次のとおりです。

- メニューをディスプレイ装置に送り、選択されたオプションをディスプレイ装置から受信するためには、**ファイル送信/受信 (SNDRCVF)** コマンドを使用します。
- メニューから選択したオプションが 98 である場合には、ORD040C からそれを呼び出したプロシージャーに制御が戻ります。
- 択一的な値として応答を処理するためには、ELSE ステートメントが必要です。

注**:** このメニューは**プログラム呼び出し (CALL)** コマンドを使用して実行されます。 **メニュー表示 (GO)** コマンドを使用して実行されるこのメニューの詳細は、「アプリケーション表示プログラミング」ブックを参照してください。

関連情報:

ファイル送信 (SNDF) コマンド

ファイル受信 (RCVF) コマンド

ファイル送信/受信 (SNDRCVF) コマンド

Application Display Programming

CL プロシージャーまたはプログラムでの表示装置ファイルの一時変更 (OVRDSPF コマンド)

表示装置ファイル一時変更 (OVRDSPF) コマンドを使用して、CL プロシージャーまたはプログラムで指定した表示装置ファイルの置き換えや、既存の表示装置ファイルの特定のパラメーターの変更を行うことができます。これは、モジュールまたはプログラムのコンパイル後に名前の変更や移動を行ったファイルの場合に特に役立ちます。

OVRDSPF コマンドの初期のパラメーターは次のとおりです。

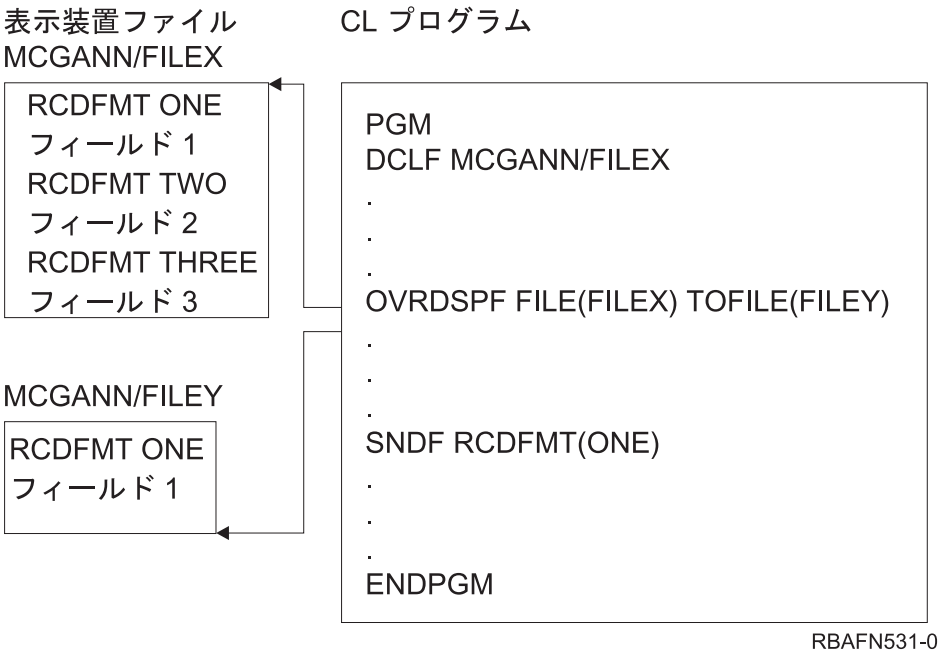
```
OVRDSPF   FILE(一時変更ファイル名) TOFILE(新規ファイル名)
          DEV(device-name)
```

OVRDSPF コマンドを CL プロシージャーまたはプログラムにより参照されたファイルに対して使用できるのは、モジュールまたはプログラムの作成時にそのファイルが DCLF コマンドに指定されており、しかもそれが表示装置ファイルである場合だけです。プログラムの実行時に使用されるファイルは、モジュールまたはプログラムの作成時に参照されたファイルと同じタイプでなければなりません。

OVRDSPF コマンドは、一時変更したいファイルをオープンする前に実行する必要があります。オープンは、最初に送信コマンドまたは受信コマンドを使用した時点で行われます。システムがファイルを一時変更するのは、以下の条件のいずれかが発生した場合です。

- **OVRDSPF** コマンドを含むプロシージャーやプログラムでファイルがオープンされた場合。
- CALLPRC コマンドにより制御を移された他のプロシージャーでファイルがオープンされた場合。
- CALL コマンドにより制御を移された他のプログラムでファイルがオープンされた場合。

別のファイルに一時変更する場合には、**ファイル送信 (SNDF)** コマンド、**ファイル受信 (RCVF)** コマンド、または**ファイル送信/受信 (SNDRCVF)** コマンドで参照されているレコード様式名が一時変更後の新しいファイルにも入っていることだけが条件になります。次の例では、表示装置ファイル FILEY にはレコード様式 TWO または THREE は必要ありません。



元のファイル名と一時変更先のファイル名が参照するレコード様式名に、同じフィールド定義と標識名が同じ順序で入っていることを確認しなければなりません。 LVLCHK(*NO) を指定すると、予期しない結果が生じることがあります。

OVRDSPF コマンドが提供された場合は、**SNDF**、**RCVF**、および **SNDRCVF** コマンドの DEV パラメーターには、別の考慮事項も関係しています。 **RCVF**、**SNDF**、または **SNDRCVF** コマンドの DEV パラメーターに *FILE を指定した場合、システムは自動的に、一時変更されるファイルに対応する正しい装置を操作の対象にします。 RCVF、SNDF、または SNDRCVF コマンドの DEV キーワードに特定の装置が指定されている場合には、次のどちらかが起こる場合があります。

- 単一装置表示装置ファイルを使用していて、その表示装置ファイルを **RCVF**、 **SNDF**、または **SNDRCVF** の各コマンドに指定された装置以外の装置に一時変更した場合には、エラーが発生します。
- 複数装置表示装置ファイルを使用していて、**RCVF**、**SNDF**、または **SNDRCVF** の各コマンドに指定された装置が、**OVRDSPF** コマンドに指定された装置に含まれていなかった場合には、エラーが発生します。

関連情報:

表示装置ファイル指定変更 (OVRDSPF) コマンド

ファイル送信 (SNDF) コマンド

ファイル受信 (RCVF) コマンド

ファイル送信/受信 (SNDRCVF) コマンド

複数装置表示装置ファイルの処理

複数ディスプレイ装置構成が使用されるのは、1 人の要求元により呼び出された 1 つのジョブが、1 つの表示装置ファイルを介して複数のディスプレイ装置と交信を行う場合です。 1 つの CL プログラムまたは

プロシージャーで処理できる表示装置ファイルは 1 つだけですが、その表示装置ファイル、またはその表示装置ファイルの異なるレコード様式を、複数のディスプレイ装置に送ることができます。

システムの通常の操作モードでは、ワークステーション・ユーザーがサインオンするとその対話式ジョブの要求元になります。プログラムまたは各ユーザーは、プログラムまたはプロシージャー内の表示装置ファイルも含めて、プログラムまたはプロシージャーの論理コピーを使用するので、多数のユーザーがこの操作を同時に行うことができます。このような使用方法では、個々の要求元が別個のジョブを呼び出します。これは、複数のディスプレイ装置を使用しているとは見なされません。

複数装置表示装置ファイルで主として使用するコマンドは、次のとおりです。

受信終了 (ENDRCV)
これは、満たされなかった入力要求を取り消すためのコマンドです。

待機 (WAIT)
これは、WAIT(*NO) が指定されている 1 つまたは複数の**ファイル受信 (RCVF)** コマンドまたは SNDRCVF コマンドを事前に発行するか、あるいは INVITE DDS キーワードを含むレコード様式に対して 1 つまたは複数の**ファイル送信 (SNDF)** コマンドを事前に発行することによって、ユーザー・データを要求した任意のディスプレイ装置から入力を受け入れるためのコマンドです。

複数装置表示装置ファイルを使用する場合には、その表示装置ファイルの作成時点の CRTDSPF コマンドの DEV パラメーターかその変更時点の CHGDSPF コマンドの DEV パラメーターに、または一時変更コマンドの DEV パラメーターにそれぞれ装置名を指定しなければならず、また装置の数は、CRTDSPF コマンドの MAXDEV パラメーターに指定した数以下でなければなりません。

複数ディスプレイ装置構成は、**ファイル送信/受信 (SNDRCVF)** および RCVF コマンドに影響を与え、**待機 (WAIT)** または**受信終了 (ENDRCV)** コマンドを使用しなければならないこともあります。複数装置表示装置で**ファイル受信 (RCVF)** または**ファイル送信 / 受信 (SNDRCVF)** コマンドを使用する場合、デフォルト値 WAIT(*YES) は DEV パラメーターに指定されている装置からプログラムに入力可能フィールドが戻されるまで、それ以降の処理が行われないようにします。しかし、応答は遅れることもあるので、WAIT(*NO) を指定して受信操作が満たされる前にプロシージャーまたはプログラムが他のコマンドの実行を続けることができるようにする場合もあります。

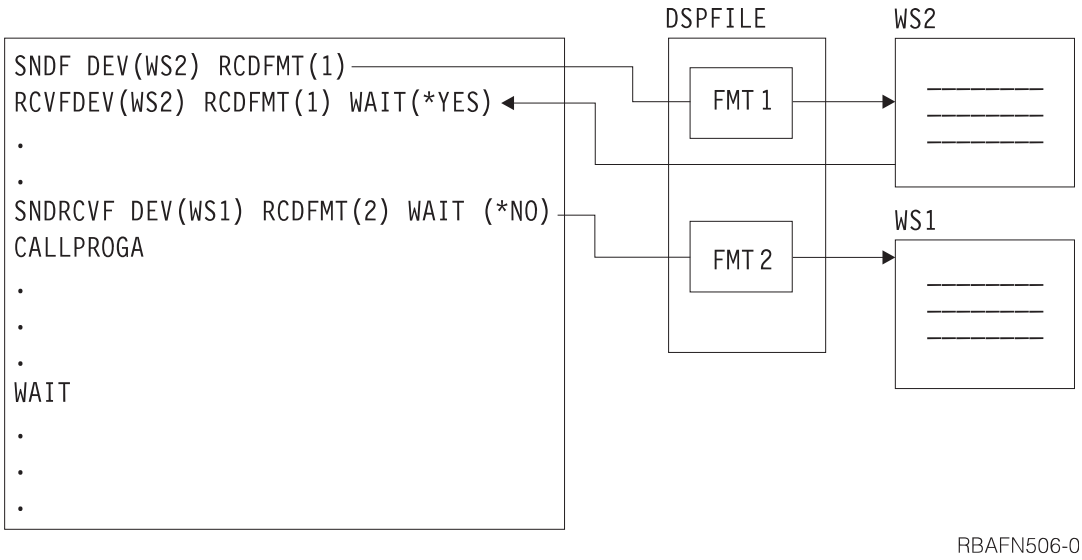
RCVF または**ファイル送信/受信 (SNDRCVF)** コマンドを使用し、WAIT(*NO) を指定した場合には、**待機 (WAIT)** コマンドを実行するまで CL プロシージャーまたはプログラムの処理が続行されます。

DDS の INVITE キーワードを含むレコード様式に対して**ファイル送信 (SNDF)** コマンドを使用するのは、WAIT(*NO) を指定して**ファイル送信/受信 (SNDRCVF)** コマンドを使用するのと同様です。DDS INVITE キーワードは、**ファイル送信/受信 (SNDRCVF)** および**ファイル受信 (RCVF)** コマンドでは無視されます。

データ・レコードにアクセスするためには、**待機 (WAIT)** コマンドを発行しなければなりません。使用可能なデータがない場合には、ディスプレイ装置からデータを受信するまで、あるいは CRTDSPF、CHGDSPF、または OVRDSPF のいずれかのコマンドの WAITRCD パラメーターで表示装置ファイルに対して指定されている制限時間が経過するまで、処理は抑止されます。制限時間が経過すると、メッセージ CPF0889 が出されます。

待機 (WAIT) の条件は、ENDJOB、ENDSYS、PWRDWNSYS、および ENDSBS の各コマンドの制御オプションによるジョブの取り消しが実行された場合にも満たされます。この場合には、メッセージ CPF0888 が発行されるだけでデータは戻されません。事前に受信要求がないのに WAIT コマンドを出した場合 (RCVF . . . WAIT(*NO) など) には、処理エラーが起こります。

典型的な複数ディスプレイ装置構成 (およびそのコーディング) の例を、次に示します。



前の例では、最初の 2 つのコマンドには、デフォルト値を用いた典型的な手順が示されています。処理は、WS2 からの受信操作が完了するまで待機します。 WS2 は DEV パラメーターに指定されているので、RCVF コマンドはこれ以前に他のワークステーションに対する未処理の要求 (ここでは示されていません) が満たされた場合でも、WS2 が応答するまでは実行されません。

しかし、SNDRCVF コマンドには WAIT(*NO) が指定されているので、WS1 からの応答があるまで待機することはなく、処理は続行され、PROGA が呼び出されます。この後、WAIT コマンドによって未処理の要求がワークステーションによって満たされるか、またはその機能が時間切れになるまで処理は抑止されます。

WAIT コマンドの形式は次のとおりです。

```
WAIT DEV(CL 変数名)
```

DEV パラメーターを指定した場合には、CL 変数には応答した装置の名前が入ります。 (デフォルト値は *NONE です。) 複数の受信要求 (RCVF. . . WAIT(*NO) など) がある場合には、この変数には、 **待機 (WAIT)** コマンドの実行後最初に応答した装置の名前が入り、処理が続けられます。受信したデータは、表示装置ファイルの該当フィールドに対応する変数に入ります。

WAIT(*YES) を指定した RCVF コマンドは、特定の装置からのデータの受け入れを待機するために使用することができます。この場合、受信要求を開始した操作と RCVF コマンドの両方に、同じレコード様式名を指定しなければなりません。

いくつかの受信要求が未処理で、しかも特定のディスプレイ装置からの応答がなければそれ以上処理を進めることができない場合があります。次の例では、3 つのコマンドに WAIT(*NO) が指定されていますが、WS3 が応答するまでは、ラベル LOOP の後の処理に進むことはできません。

```
        PGM
        .
        .
        .
        SNDF DEV(WS1) RCDFMT(ONE)
        SNDF DEV(WS2) RCDFMT(TWO)
        SNDRCVF DEV(WS3) RCDFMT(THREE) WAIT(*NO)
        RCVF DEV(WS2) RCDFMT(TWO) WAIT(*NO)
```

```
         RCVF DEV(WS1) RCDFMT(ONE) WAIT(*NO)
         CALL...
         CALL...
         .
         .
         RCVF DEV(WS3) RCDFMT(THREE) WAIT(*YES)
LOOP:    WAIT DEV(&WSNAME)
         MONMSG CPF0882 EXEC(GOTO REPLY)
         .
         .
         .
         GOTO LOOP
REPLY:   CALL...
         .
         .
         .
         ENDPGM
```

CL プロシージャーおよびプログラムでは、**受信終了 (ENDRCV)** コマンドもサポートされます。このコマンドを使用して、未完了の入力要求を取り消すことができます。 **ファイル送信 (SNDF)** または**ファイル送信/受信 (SNDRCVF)** コマンドもまた、未完了の入力要求を取り消すことになります。ただし、**ファイル送信 (SNDF)** または**ファイル送信/受信 (SNDRCVF)** コマンドが実行された時点で使用可能なデータがあった場合には、メッセージ CPF0887 が送られます。この場合、WAIT コマンドまたは**ファイル受信 (RCVF)** コマンドを使用してデータを受け取るか、あるいは**受信終了 (ENDRCV)** コマンドを用いてその要求を明示的に取り消してからでなければ、**ファイル送信 (SNDF)** または**ファイル送信/受信 (SNDRCVF)** コマンドを実行することはできません。

関連概念:

191 ページの『CL コマンドで使用する変数』
変数とは名前を持つ可変の値のことであり、その名前を参照することによってアクセスまたは変更することができます。

関連情報:

待機 (WAIT) コマンド

受信終了 (ENDRCV) コマンド

ファイル送信 (SNDF) コマンド

ファイル受信 (RCVF) コマンド

ファイル送信/受信 (SNDRCVF) コマンド

データベース・ファイルからのデータの受信 (RCVF コマンド)

データベース・ファイルからデータを受信するには、**ファイル受信 (RCVF)** コマンドを使用します。

RCVF コマンドを実行すると、ファイルのアクセス・パス上で次の順番のレコードが読み取られ、データベース・レコード様式に定義されたフィールドの値が、そのフィールドに対応する CL 変数に入れられます。 CL ではゾーン 10 進数または 2 進数はサポートされないことに注意してください。したがって、データベース・ファイルにゾーン 10 進数または 2 進数として定義されたフィールドがある場合には、CL プロシージャーまたはプログラムでは *DEC フィールドとして定義されます。 *DEC フィールドはパック 10 進数として内部的にサポートされ、**RCVF** コマンドにより、必要に応じてゾーン 10 進数および 2 進数からパック 10 進数への変換が行われます。浮動小数点データを含むデータベース・ファイルは、CL プロシージャーまたはプログラムでは使用できません。

ファイルの終わりに達すると、メッセージ CPF0864 がプロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムに送られます。このメッセージが発行された場合、**RCVF** コマンドが処理されても、レコード様式に対応して宣言された CL 変数は変更されません。ユーザーはこのメッセージを監視し

て、ファイル終了時の適切な処置を取らなければなりません。ファイルの終わりに達した後でさらに RCVF コマンドを実行しようとすると、メッセージ CPF0864 が再び送信されます。

RCVF コマンドがファイルの終わりに達した場合は、**データベース・ファイル・クローズ (CLOSE)** コマンドを使用してファイルをクローズすることができます。ファイルがクローズされると、次の **RCVF** コマンドがファイルを暗黙的に再オープンし、レコードを読み取ります。

関連情報:

ファイル受信 (RCVF) コマンド

データベース・ファイル・クローズ (CLOSE) コマンド

ファイルを複数回読み取る (CLOSE コマンド)

データベース・ファイル・クローズ (CLOSE) コマンドを使用して、 **ファイル受信 (RCVF)** コマンドで暗黙的にオープンされるデータベース・ファイルをクローズできます。

CLOSE コマンドを使用して、データベース・ファイル内のレコードを複数回処理できます。ファイルがクローズされると、次の **RCVF** コマンドがファイルを暗黙的に再オープンし、レコードを読み取ります。既にクローズされているデータベース・ファイル、または一度もオープンされていないデータベース・ファイルについても、**CLOSE** コマンドを使用できます。エラー・メッセージは送信されません。CL プログラムまたはプロシージャーが終了すると、その CL プログラムまたはプロシージャー内で暗黙的にオープンされたすべてのデータベースと表示装置ファイルが、暗黙的にクローズされます。

関連情報:

ファイル受信 (RCVF) コマンド

データベース・ファイル・クローズ (CLOSE) コマンド

CL プロシージャーまたはプログラムでのデータベース・ファイルの一時変更 (OVRDBF コマンド)

データベース・ファイル一時変更 (OVRDBF) コマンドは、CL プロシージャーまたはプログラムに指定されているデータベース・ファイルの一時変更や、既存のデータベース・ファイルの特定のパラメーターの変更を行います。

これは、プロシージャーまたはプログラムの作成後に名前の変更や移動を行ったファイルの場合に特に役立ちます。また、最初のメンバー以外のファイル・メンバーにアクセスしたい場合にも、このコマンドを使用することができます。

OVRDBF コマンドの初期のパラメーターは次のとおりです。

```
OVRDBF FILE(一時変更ファイル名) TOFILE(新規ファイル名)
       MBR(メンバー名)
```

データベース・ファイル一時変更 (OVRDBF) コマンドを CL プロシージャーまたはプログラムにより参照されたファイルに対して使用できるのは、モジュールまたはプログラムの作成時に、そのファイルが**ファイル宣言 (DCLF)** コマンドに指定されており、しかもそれがデータベース・ファイルである場合だけです。プログラムの実行時に使用されるファイルは、モジュールまたはプログラムの作成時に参照されたファイルと同じタイプのものでなければなりません。

データベース・ファイル一時変更 (OVRDBF) コマンドは、一時変更したいファイルをオープンする前に実行する必要があります (オープンは、最初に**ファイル受信 (RCVF)** コマンドを使用した時点で行われます)。ファイルが一時変更されるのは、それが**データベース・ファイル一時変更 (OVRDBF)** コマンドを含むプロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムでオープンされた場合、CALL コマ

ンドにより制御を移された他のプログラムでオープンされた場合、または CALLPRC コマンドにより制御を移された他のプロシージャーでオープンされた場合です。

異なるファイルへの一時変更を行う場合には、一時変更ファイルはレコード様式を 1 つだけ備えたファイルでなければなりません。 DDS で複数のレコード様式が定義されている論理ファイルは、それが 1 つの物理ファイル・メンバーだけを基礎として定義されている場合に限り使用することができます。 DDS で定義されているレコード様式が 1 つだけの論理ファイルの場合には、複数の物理ファイル・メンバーを基礎として定義されていても構いません。様式の名前は、プログラムの作成時に参照された様式名と同じである必要はありません。ただし、一時変更先のファイルのデータ形式が元のファイルのデータ形式と同じであることを確認しなければなりません。 LVLCHK(*NO) を指定すると、予期しない結果が生じることがあります。

関連情報:

データベース・ファイル指定変更 (OVRDBF) コマンド

表示コマンドからの出力ファイル

IBM 提供の多くの表示コマンドは、そのコマンドからの出力をデータベース・ファイルに入れます (OUTFILE パラメーター)。これらのファイルのデータは、CL プロシージャーまたはプログラムで読み取って処理することができます。

次に示す CL プロシージャーは、ユーザー名およびライブラリー名の 2 つのパラメーターを受け取ります。プロシージャーはライブラリー内のすべてのプログラム、ファイル、およびデータ域の名前を判別し、指定のユーザーに通常の権限を認可します。

注**:** コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
     PGM PARM(&USER &LIB)
     DCL &USER *CHAR 10
     DCL &LIB *CHAR 10
(1)  DCLF QSYS/QADSPOBJ
(2)  DSPOBJD OBJ(&LIB/*ALL) OBJTYPE(*FILE *PGM *DTAARA) +
     OUTPUT(*OUTFILE) OUTFILE(QTEMP/DSPOBJD)
(3)  OVRDBF QADSPOBJ TOFILE(QTEMP/DSPOBJD)
(4)  READ:  RCVF
(5)  MONMSG CPF0864 EXEC(RETURN)  /* EXIT WHEN END OF FILE REACHED  */
(6)  GRTOBJAUT OBJ(&ODLBNM/&ODOBNM) OBJTYPE(&ODOBTP) +
     USER(&USER) AUT(*CHANGE)
     GOTO READ                    /*GO BACK FOR NEXT RECORD*/
     ENDPGM
```

(1) 宣言されているファイル (ライブラリー QSYS の QADSPOBJ) は、IBM 提供のファイルで、**オブジェクト記述の表示 (DSPOBJD)** コマンドにより使用されるファイルです。このファイルは、出力ファイルの作成時にこのコマンドにより参照される 1 次ファイルです。また、CL コンパイラーはレコードの様式を判別し、レコード様式中のフィールドに対応する変数を宣言するためにこのファイルを参照します。

(2) **DSPOBJD** コマンドは、DSPOBJD という名前のファイルをライブラリー QTEMP に作成します。このファイルは、ファイル QADSPOBJ と同じ様式を持ちます。

(3) **データベース・ファイル一時変更 (OVRDBF)** コマンドは、宣言されているファイル (QADSPOBJ) を **DSPOBJD** コマンドにより作成されたファイルに一時変更します。

(4) **ファイル受信 (RCVF)** コマンドは、この DSPOBJD によるファイルからレコードを読み取ります。レコード中の各フィールドの値が、**ファイル宣言 (DCLF)** コマンドにより暗黙的に宣言された

対応する CL 変数にコピーされます。**OVRDBF** コマンドが使用されているので、ファイル QSYS/QADSPOBJ の代わりにファイル QTEMP/DSPOBJD が読み取られます (ファイル QSYS/QADSPOBJ は読み取られません)。

(5) メッセージ CPF0864 を監視します。このメッセージはファイルの終わりに達したことを示します。したがって、このメッセージが出ると、プロシージャーは制御を呼び出し元のプロシージャーに戻します。

(6) **オブジェクト権限認可 (GRTOBJAUT)** コマンドは、**RCVF** コマンドにより読み取ったオブジェクト名、ライブラリー名、およびオブジェクト・タイプに対応する変数を使用して実行されます。

メッセージ

メッセージは、ユーザーとプログラム間の通信に使用されます。

IBM i オペレーティング・システムでは、プロシージャーまたはプログラム間、ジョブ間、ユーザー間、およびユーザーとプロシージャーまたはプログラム間の通信が、メッセージによって行われます。メッセージの送受信の形態には以下のようなものがあります。

- あるシステム・ユーザーから別のシステム・ユーザーへ。これは、メッセージの受信者が現在システムを使用していない場合でも行うことできます。
- あるオリジナル・プログラム・モデル (OPM) プログラムまたは統合言語環境 (ILE) プロシージャーから別の OPM プログラムまたは ILE プロシージャーへ。
- あるプログラムまたはプロシージャーからシステム・ユーザーへ。これは、メッセージの受信者が現在システムを使用していない場合でも行うことができます。

メッセージは事前定義メッセージ、または即時メッセージにすることができます。

- 事前定義メッセージは、それを使用するプログラムの外部で作成され、存在します。事前定義メッセージは、メッセージ・ファイルに保管され、メッセージ番号を持ちます。システム事前定義メッセージの例は次のとおりです。

  ```
  CPF0006  コマンドでエラーが起こった。
  ```

- 即時メッセージは、送信元によって、その送信時に作成されます。即時メッセージは、メッセージ・ファイルに保管されません。ディスプレイ装置で受信される即時メッセージの例は、次のとおりです。

  ```
  送信元  . . . :   QSYSOPR      06/12/05   10:50:54
  System going down at 11:00; please sign off
  ```

システム・ユーザーが対話式で送ることができるのは、即時メッセージおよび応答だけです。

OPM プログラムまたは ILE プロシージャーでは、ユーザー定義のデータを含む即時メッセージまたは事前定義メッセージを送信することができます。さらに、プログラムまたはプロシージャーは次のことを行うことができます。

- メッセージを受け取る
- メッセージ・ファイルからメッセージを検索して、それをプログラム変数に入れる
- メッセージ待ち行列からメッセージを除去する
- メッセージを監視する

システムには、システム内でのプログラム間通信、およびシステムとそのユーザー間の通信を行えるようにする、大量の事前定義メッセージが提供されます。発注する各ライセンス・プログラムには、そのメッセー

ジが適用されるライセンス・プログラムと同じライブラリーに保管されているメッセージ・ファイルがあります。例えば、システム・メッセージは、ライブラリー QSYS 内のファイル QCPFMSG に保管されています。

メッセージ・ファイル内の各事前定義メッセージは、7 文字のコードで固有に識別され、メッセージ記述によって定義されます。 メッセージ記述には、メッセージ・テキストおよびメッセージ・ヘルプ・テキスト、重大度レベル、有効な応答値、デフォルト応答値、および各種の他の属性が入っています。

システム内で送受信されるすべてのメッセージは、メッセージ待ち行列を介して伝送されます。コマンドなどの直接要求への応答として発行されるメッセージは、要求がなされた画面に自動的に表示されます。他のすべてのメッセージについては、ユーザー、プログラムまたはプロシージャーが、待ち行列からメッセージを受信するかまたはそれを表示しなければなりません。システムには、複数の IBM 提供メッセージ待ち行列があります。

さらにシステムは、ログに発行されるメッセージのいくつかを作成します。ジョブ・ログには、ジョブに入力された要求に関連する情報が入り、活動記録ログにはジョブ、サブシステム、および装置状況の情報が入ります。

関連概念:

558 ページの『メッセージ待ち行列のタイプ』
システムには、異なるタイプのメッセージ待ち行列 (ワークステーション・メッセージ待ち行列、ユーザー・プロファイル・メッセージ待ち行列、ジョブ・メッセージ待ち行列、システム・オペレーター・メッセージ待ち行列、および活動記録メッセージ待ち行列) があります。

関連タスク:

511 ページの『リソースの割り振り』
システム上でのオブジェクトの割り振りは、保全性を確保し、可能な限り高度な並行処理ができるような形で行われます。

518 ページの『オブジェクトの存在の検査』
プログラムでオブジェクトを使用する前に、そのオブジェクトが存在しているかどうか、およびそれを使用するために必要な権限があるかどうかを判別するための検査を行ってください。

525 ページの『表示装置ファイルによるデータの送信および受信』
表示装置ファイルに対するデータの送信 (書き込み) や受信 (読み取り) を行うために CL プロシージャーおよびプログラムで使用できるコマンドは、**ファイル送信 (SNDF)**、**ファイル受信 (RCVF)**、および **ファイル送信/受信 (SNDRCVF)** の 3 つのコマンドだけです。

関連資料:

254 ページの『**メッセージ・モニター** コマンド』
メッセージ・モニター (MONMSG) コマンドは、**MONMSG** コマンドが使用されている CL プログラムまたはプロシージャーの呼び出しスタックに送られてくる、エスケープ・メッセージ、通知メッセージ、または状況メッセージの監視に使用されます。

関連情報:

メッセージ記述追加 (ADDMSGD) コマンド

メッセージ記述の定義

事前定義メッセージは、メッセージ・ファイルに保管されます。

固有のメッセージ・ファイルおよびメッセージ記述を作成することができます。事前定義メッセージの作成により、同じメッセージを複数のプロシージャーまたはプログラム内で使用できますが、定義は 1 回だけです。さらに、事前定義メッセージを、それを使用するプロシージャーおよびプログラムに影響を与えず

に、英語以外の言語に変更、変換する (ユーザーが表示するメッセージに基づいて) こともできます。メッセージがプロシージャーまたはプログラム内に入っている場合は、そのメッセージを変更する際に、モジュールまたはプログラムを再コンパイルしなければなりません。

独自のメッセージおよびメッセージ・ファイルを作成する以外にも、システム・メッセージ処理機能により、次のことを行えます。

- メッセージ待ち行列の作成および変更 (**メッセージ待ち行列作成 (CRTMSGQ)** コマンド、**メッセージ待ち行列変更 (CHGMSGQ)** コマンド、および **メッセージ待ち行列処理 (WRKMSGQ)** コマンド)
- メッセージ・ファイルの作成および変更 (**メッセージ・ファイル作成 (CRTMSGF)** コマンド、**メッセージ・ファイルの変更 (CHGMSGF)** コマンド)
- メッセージ記述の追加 (**メッセージ記述追加 (ADDMSGD)** コマンド)
- メッセージ記述の変更 (**メッセージ記述変更 (CHGMSGD)** コマンド)
- メッセージ記述の除去 (**メッセージ記述除去 (RMVMSGD)** コマンド)
- 即時メッセージの送信 (**メッセージ送信 (SNDMSG)** コマンド、**中断メッセージ送信 (SNDBRKMSG)** コマンド、**プログラム・メッセージ送信 (SNDPGMMSG)** コマンド、および**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンド)
- メッセージ待ち行列内のメッセージの表示 (**メッセージの表示 (DSPMSG)** および**メッセージ処理 (WRKMSG)**)
- メッセージ・ファイル内のメッセージ記述の表示 (**メッセージ記述表示 (DSPMSGD)** および**メッセージ記述処理 (WRKMSGD)** コマンド)
- CL プロシージャーまたはプログラムを使用して、次のことを行います。
  - メッセージをワークステーション・ユーザーまたはシステム・オペレーターに送信する (**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンド)
  - メッセージをメッセージ待ち行列に送信する (**プログラム・メッセージ送信 (SNDPGMMSG)** コマンド)
  - メッセージ待ち行列からメッセージを受信する (**メッセージ受信 (RCVMSG)** コマンド)
  - メッセージ待ち行列に応答を送信する (**応答送信 (SNDRPY)** コマンド)
  - メッセージ・ファイルからメッセージを検索する (**メッセージ検索 (RTVMSG)** コマンド)
  - メッセージ待ち行列からメッセージを除去する (**メッセージ除去 (RMVMSG)** コマンド)
  - 呼び出しメッセージ待ち行列に送信される、エスケープ、通知、および状況のメッセージを監視する (**メッセージ・モニター (MONMSG)** コマンド)
- システム応答リストを使用して、ジョブによって送信される事前定義された照会メッセージの応答を指定する (**システム応答リスト項目追加 (ADDRPYLE)** コマンド、**システム応答リスト項目変更 (CHGRPYLE)** コマンド、**システム応答リスト項目除去 (RMVRPYLE)** コマンド、および**システム応答リスト項目処理 (WRKRPYLE)** コマンド)

メッセージは、送信時に、次のいずれかのタイプとして定義されます。

- 情報 (*INFO)。機能の条件に関する情報を送るメッセージ。
- 照会 (*INQ)。情報を送るが、応答も要求するメッセージ。
- 通知 (*NOTIFY)。プロシージャーまたはプログラムが、訂正処置またはその呼び出しプロシージャーあるいはプログラムからの応答を要求する状態を記述するメッセージ。プロシージャーまたはプログラムは、それが呼び出すプログラムまたはプロシージャーからの通知メッセージの着信を監視することができます。
- 応答 (*RPY)。受信した照会または通知メッセージに対する応答のメッセージ。
- 送信元のコピー (*COPY)。送信元によって保管される照会または通知メッセージのコピー。

- 要求 (*RQS)。受信プロシージャーまたはプログラムの機能を要求するメッセージ。 (例えば、CL コマンドが要求メッセージです。)
- 完了 (*COMP)。作業の完了状況を送信するメッセージ。
- 診断 (*DIAG)。システム機能の処理、アプリケーション・プログラム、または入力データにおけるエラーに関するメッセージ。
- 状況 (*STATUS)。プロシージャーまたはプログラムによって行われる作業の状況を記述メッセージ。プロシージャーまたはプログラムは、それが呼び出すプログラムまたはプロシージャーからの状況メッセージの着信を監視することができますが、状況メッセージはジョブ・ログには表示されません。外部メッセージ待ち行列 (*EXT) に送信される状況メッセージは、ディスプレイ装置に表示され、ディスプレイ装置ユーザーに進行中の操作を通知するのに使用されます。
- エスケープ (*ESCAPE)。プロシージャーまたはプログラムが異常終了する条件を記述するメッセージ。プロシージャーまたはプログラムでは、それが呼び出すプログラムまたはプロシージャーからの、あるいはマシンからのエスケープ・メッセージの着信を監視することができます。エスケープ・メッセージの送信後、送信元のプログラムに制御は戻されません。

関連概念:

487 ページの『制御言語の動的プロンプト・メッセージ』
コマンド (*CMD) が作成されたときに、そのコマンドのオブジェクトに保管されるメッセージ識別コードを使用して、メッセージ・ファイルからプロンプト・メッセージを動的に検索することができます。この機能により、単一のコマンドに複数の各国語で表記されたプロンプト・メッセージを含めることができます。

関連タスク:

511 ページの『リソースの割り振り』
システム上でのオブジェクトの割り振りは、保全性を確保し、可能な限り高度な並行処理ができるような形で行われます。

518 ページの『オブジェクトの存在の検査』
プログラムでオブジェクトを使用する前に、そのオブジェクトが存在しているかどうか、およびそれを使用するために必要な権限があるかどうかを判別するための検査を行ってください。

関連資料:

254 ページの『**メッセージ・モニター** コマンド』
メッセージ・モニター (MONMSG) コマンドは、**MONMSG** コマンドが使用されている CL プログラムまたはプロシージャーの呼び出しスタックに送られてくる、エスケープ・メッセージ、通知メッセージ、または状況メッセージの監視に使用されます。

関連情報:

CL コマンド検索プログラム

メッセージ・ファイルの作成

独自の事前定義メッセージを作成するには、まずメッセージを入れるメッセージ・ファイルを作成しなければなりません。

メッセージ・ファイル作成 (CRTMSGF) コマンドを使用して、メッセージ・ファイルを作成します。次に**メッセージ記述追加 (ADDMSGD)** コマンドを使用して、独自のメッセージを記述し、それをメッセージ・ファイルに入れます。

メッセージ・ファイル作成 (CRTMSGF) コマンドでは、SIZE パラメーター上にキロバイト単位で最大サイズを指定できます。次の式を使用すると、最大数を判別できます。

```
S + (I x N)
```

ここでは、以下の変数が真です。

S 初期記憶域の量

I 毎回追加される記憶域の量

N 記憶域を追加する回数

デフォルトでは、S は 10、I は 2、N は *NOMAX です。

例えば、S は 5、I は 1、N は 2 として指定します。ファイルが 5KB の初期記憶量に達すると、システムは自動的に新たに 1KB を初期記憶域に追加します。追加される量 (1KB) は、記憶域に 2 回追加して、合計 7KB にできます。 *NOMAX を N として指定すると、メッセージ・ファイルの最大サイズは 16MB になります。

メッセージ・ファイルの最大サイズを指定すると、メッセージ・ファイルがいっぱいになっても、そのメッセージ・ファイルのサイズは変更できません。その場合は、新たにメッセージ・ファイルを作成して、その新規ファイル内にメッセージを作成し直す必要があります。 **メッセージ・ファイル組み合わせ (MRGMSGF)** コマンドを使用すると、あるメッセージ・ファイルから別のメッセージ・ファイルにメッセージ記述を複写することができます。このステップを回避したい場合は、メッセージ作成時のメッセージ・ファイルに必要なサイズを計算するか、または *NOMAX を指定することが重要です。

関連情報:

CL コマンド検索プログラム

メッセージ記述追加 (ADDMSGD) コマンド

メッセージ・ファイル作成 (CRTMSGF) コマンド

独立 **ASP** 内のメッセージ・ファイル**:**

メッセージ・ファイルは独立補助記憶域プール (ASP) 内に作成できますが、独立 ASP はオフラインで用いられることがあるため推奨されていません。独立 ASP がオフラインの場合には、ジョブ・ログ内のメッセージおよびメッセージ待ち行列が正しく表示されません。

メッセージ・ファイルのサイズの決定**:**

メッセージ・ファイルのサイズを決定するには、この公式を使用する必要があります。 **メッセージ記述追加 (ADDMSGD)** コマンドのパラメーターは、括弧内に指定します。

- メッセージ索引は、42 バイトを基底に、そのメッセージの長さを加えたものと等価です。
- メッセージ・テキスト (MSG) は、16 バイトを基底にそのメッセージの長さを加えたものと等価です。
- メッセージ・ヘルプ情報 (SECLVL) (ある場合) は、16 バイトを基底に、そのメッセージ・ヘルプの長さを加えたものと等価です。
- 形式 (FMT) (ある場合) は、14 バイトに (3 x FMTS 数) を加えたものと等価です。
- タイプおよび長さ (TYPE および LEN) は、48 バイトと等価です。
- 特殊な値 (SPCVAL) は、2 に (64 x SPCVAL 数) を加えたものと等価です。
- 値 (VALUES) は、32 x (VALUES 数) と等価です。
- 範囲 (RANGE) は、64 バイトと等価です。
- 関係 (REL) は、その関係の長さと等価です。
- デフォルト値 (DFT) は、そのデフォルト応答の長さと等価です。
- デフォルト・プログラム、問題ログ、およびダンプ・リスト (DFTPGM、LOGPRB、 DMPLST) は、35 に (DMPLST 内の数× 2) を加えたものと等価です。

- ALROPT は 12 バイトと等価です。

メッセージ・ファイル内で可能な最小の項目は 59 バイトであり、可能な最大の項目は 5764 バイトです。次の表では、可能な最大の項目を説明しています。

表 *33.* 可能な最大の項目

属性	可能な最大の項目
メッセージ索引	42 バイト
メッセージ・テキスト	148 バイト
メッセージ・ヘルプ・テキスト	3016 バイト
99 の形式	311 バイト
タイプおよび長さ	48 バイト
20 の特殊値	1282 バイト
20 の値	640 バイト
デフォルトの応答値	32 バイト
デフォルトのプログラムおよびダンプ・リスト	233 バイト
警報オプション	12 バイト

次の例では、**メッセージ・ファイル作成 (CRTMSGF)** コマンドでメッセージ・ファイル USRMSG を作成します。

```
CRTMSGF   MSGF(QGPL/USRMSG) +
          TEXT('Message file for user-created messages')
```

RPG for IBM i 内で DSPLY 操作コードと一緒に使用するメッセージ・ファイルを作成する場合、そのメッセージ・ファイルは QUSERMSG と命名しなければなりません。

関連情報:

メッセージ記述追加 (ADDMSGD) コマンド

メッセージ・ファイル作成 (CRTMSGF) コマンド

メッセージのファイルへの追加

メッセージ記述追加 (ADDMSGD) コマンドは、事前定義メッセージを記述し、作成したメッセージ・ファイルにそれらを追加します。

ADDMSGD コマンドで、メッセージ識別コード、メッセージを入れるメッセージ・ファイルの名前、およびメッセージ記述を指定します。メッセージ記述には、次のものを指定できます。

- オプションの置換変数のあるメッセージ・テキスト (必須)
- オプションの置換変数のあるメッセージ・ヘルプ・テキスト
- 重大度コード
- 置換変数に使用するメッセージ・データの形式の記述
- 応答の妥当性検査チェック値
- 応答のデフォルト値
- エスケープ・メッセージのデフォルト・メッセージ処理処置
- 作成レベル
- 警報オプション
- エラー・ログ内の項目

- コード化文字セット ID (CCSID)

次のコマンドも、メッセージ記述に使用できます。

メッセージ記述変更 (CHGMSGD)
メッセージ記述を変更します。

メッセージ記述表示 (DSPMSGD)
メッセージ記述を表示します。 (メッセージ識別コードの範囲は、このコマンド内に指定できます。)

メッセージ記述除去 (RMVMSGD)
メッセージ・ファイルからメッセージ記述を除去します。

メッセージ検索 (RTVMSG)
メッセージ・ファイルからメッセージを検索します。

メッセージ・ファイル組み合わせ (MRGMSGF)
メッセージ・ファイルのメッセージを別のメッセージ・ファイルに組み合わせます。

メッセージ記述処理 (WRKMSGD)
ファイル内のメッセージのリストを表示し、メッセージ記述を追加、変更、または削除できるようにします。

メッセージ識別コードの割り当て:

メッセージ記述追加 (ADDMSGD) コマンドに指定するメッセージ識別コードは、メッセージの参照に使用されます。メッセージ識別コードは、メッセージ記述の名前です。

メッセージ識別コードは、7 文字でなければなりません。

```
pppmmnn
```

この場合、ppp はプロダクトまたはアプリケーション・コードであり、mm は数字グループ・コードであり、nn は数字サブタイプ・コードです。 mmnn として指定される数値は、プロダクト・メッセージまたはアプリケーション・メッセージのセットをさらに分割するときに使用できます。数字グループ・コードおよびサブタイプ・コードは、0 から 9 の 10 進数、および A から F の文字で構成されます。

例えば、次のメッセージは CPF の 1234 です。

```
CPF1234
```

独自のメッセージを作成するときは、文字 U をプロダクト・コード内の最初の文字として使用すると、自分のメッセージをシステム・メッセージと区別しやすくなります。以下にその例を示します。

```
USR3567
```

このコードの最初の文字は英字でなければなりませんが、2 番目と 3 番目の文字は英数字にできます。グループ・コードおよびサブタイプ・コードは、0 から 9 の 10 進数、および A から F の文字で構成しなければなりません。この範囲は、16 進数のセットと呼ばれますが、メッセージ識別コードのソートでは、A から F は文字として処理されることに注意してください。

例えば、メッセージ記述の範囲を表示する場合、CPFA000 は CPF1000 に先行します。

メッセージ識別コード内で 00 の数字サブタイプを使用する場合は、注意が必要です。エスケープ、通知、または状況メッセージとして送信できる (したがって、監視できる) メッセージについて 00 の数字サブタ

イプ・コードを使用する場合は、 **メッセージ・モニター (MONMSG)** コマンド内の 00 のサブタイプ・コードによって、その数字グループ内のすべてのメッセージが監視されます。

関連タスク:

599 ページの『CL プログラムまたはプロシージャー内のメッセージの監視』

*ESCAPE メッセージ、*STATUS メッセージ、および *NOTIFY メッセージは、モニター可能なメッセージであり、プログラムまたはプロシージャーで使用される各 CL コマンドによって発行されます。

関連情報:

メッセージ記述追加 (ADDMSGD) コマンド

メッセージおよびメッセージ・ヘルプの定義:

メッセージ記述追加 (ADDMSGD) コマンドには、2 つのレベルのメッセージを定義できます。メッセージのテキストは必須であり、これによりメッセージ発行の原因となった条件が識別されます。メッセージ・ヘルプはオプションです。これは、条件を詳しく説明したり、取るべき訂正処置を説明するものです。

メッセージ・ヘルプを獲得するには、メッセージが表示されるときにディスプレイ装置ユーザーがカーソルをメッセージ行に移動して、ヘルプ・キーを押します。 3 つの様式制御文字を使用すると、ディスプレイ装置でメッセージ・ヘルプをフォーマットすることができます。これらの文字を使用すると、メッセージ・ヘルプ (通常はオンライン・ヘルプ情報) を、ユーザーにとってより読みやすいものにすることができます。

3 つの様式制御文字にはそれぞれブランクを続け、メッセージ・テキストから区別させなければなりません。

&Nb (ここで、**b** はブランク**)**
: テキストを新規行 (桁 2) に改行します。テキストが 1 行より長い場合、テキストの終わりまで、または別の様式制御文字が検出されるまで、次の行は桁 4 に字下げします。

&Pb (ここで、**b** はブランク**)**
: テキストを新規行に改行し、桁 6 に字下げします。テキストが 1 行より長い場合、テキストの終わりまで、または別の様式制御文字が検出されるまで、次の行は桁 4 から始まります。

&Bb (ここで、**b** はブランク**)**
: テキストを新規行に改行し、桁 4 から開始します。テキストが 1 行より長い場合、テキストの終わりまで、または別の様式制御文字が検出されるまで、次の行は桁 6 に字下げします。

置換変数の定義:

メッセージ記述追加 (ADDMSGD) コマンドの FMT パラメーターで、第 1 レベルまたは第 2 レベルのメッセージに置換変数を指定することができます。

例を以下に示します。

```
File &1 not found
```

これには、置換変数 &1 が入っています。メッセージが表示または検索されると、変数 &1 は検出されなかったファイルの名前に置換されます。この名前は、メッセージの送信元が提供します。例を以下に示します。

```
File ORDHDRP not found
```

これを次のものと比較します。

```
File not found
```

置換変数は、メッセージをより具体的に、より意味のあるものにします。

置換変数は、& (アンパーサンド) で開始し、n を続けます。この場合 n は、1 から 99 の任意の数字です。例えば、次のメッセージの場合、

```
File &1 not found
```

置換変数は次のように定義されます。

```
FMT((*CHAR 10))
```

置換変数に番号を付ける際には、1 から始めて連続した番号を付けなければなりません。例えば、&1、&2、&3 のようにします。ただし、メッセージ記述で定義されているすべての置換変数をメッセージ内で使用する必要はありません。

例えば、次のようなメッセージがあるとします。

```
File &3 not available
```

このメッセージでは &1 および &2 が使用されていませんが、これも有効な形式です。ただし、このようなメッセージを作成する場合にも、**メッセージ記述追加 (ADDMSGD)** コマンドの FMT パラメーターで &1、&2、&3 を定義する必要があります。上記のメッセージの場合、FMT パラメーターは、次のようになります。

```
FMT((*CHAR 10) (*CHAR 2) (*CHAR 10))
```

この例では、最初の値が &1 を、2 番目の値が &2 を、そして 3 番目の値が &3 を記述しています。&3 を使用するためには、&1 および &2 の値が存在していなければなりません。さらに、このメッセージを送信する場合は、プログラム・メッセージ送信 (SNDPGMMSG) コマンドの MSGDTA パラメーターにも、FMT パラメーターで記述したすべてのデータが含まれていなければなりません。上記のメッセージを送るためには、MSGDTA パラメーターに少なくとも 22 文字の長さが必要です。

上記のメッセージについては、FMT パラメーターを次のように指定することもできます。

```
FMT((*CHAR 0) (*CHAR 0) (*CHAR 10))
```

&1 および &2 はメッセージで使用されないため、長さ 0 として記述できます。そうすれば、メッセージ・データが送信される必要がなくなります。 (この場合、SNDPGMMSG コマンドの MSGDTA パラメーターに必要な長さは 10 文字です。)

メッセージ内で &3 を使用し、FMT パラメーターに &1 および &2 を含めるという例は、DMPLST パラメーターに &1 および &2 が指定されている場合です。 (DMPLST パラメーターは、エスケープ・メッセージを監視していないプログラムに対して、このメッセージがエスケープ・メッセージとして送られた場合に、データをダンプすることを指定するためのものです。)

置換変数は、FMT パラメーターで指定されている順序と同じ順序でメッセージに指定する必要はありません。例えば、FMT パラメーターで次のように 3 つの値を指定したとします。

```
FMT((*CHAR 10) (*CHAR 10) (*CHAR 7))
```

これらの置換変数は、メッセージ内で、例えば次のように使用できます。

```
Object &1 of type &3 in library &2 is not available
```

CL プロシージャーまたはプログラムからこのメッセージを送る場合は、メッセージ・データとして送る値を次のように連結することができます。

```
SNDPGMMSG .....MSGDTA(&OBJ *CAT &LIB *CAT &OBJTYPE)
```

メッセージ記述追加 **(ADDMSGD)** コマンドには、データ・タイプおよび長さ (オプション) を指定して、メッセージ・データ・フィールドの形式を置換変数に指定する必要があります。メッセージ・データ・フィールドに有効な値は次のとおりです。

- 2 進数 (*BIN)。符号付き 10 進整数として形式化された 2 進整数 (長さは 2、4、または 8 バイト)。長さが指定されないと、システムは 2 進整数が 2 であると想定します。
- 文字ストリング (*CHAR)。これは、単一引用符で囲んでいない文字データのストリングです。 末尾ブランクは削除されます。メッセージ記述で長さが指定されていない場合には、送信元によりフィールドの長さが決定されます。
- 変換可能文字ストリング (*CCHAR)。これは、単一引用符で囲んでいない文字データのストリングです。 末尾ブランクは削除されます。長さは必ず送信元によって決定されます。この種のデータが、65535 または 65534 以外の CCSID タグを持つメッセージ待ち行列に送信される場合、データはメッセージ・データの CCSID から、メッセージ待ち行列の CCSID に変換されます。この種のデータを、受信機能または表示機能を用いてメッセージ待ち行列から獲得するときにも、変換が行われます。 CCSID を用いたメッセージ・ハンドラーの使用方法について詳しくは、メッセージの CCSID サポートを参照してください。
- 協定世界時日付スタンプ (*UTCD)。協定世界時 (UTC) でのシステムの日付タイム・スタンプを含む、8 バイトのフィールド。出力メッセージの日付は、ジョブに指定されたタイム・ゾーンにより UTC で調整されます。日付は、日付形式 QDATFMT および日付区切り文字 QDATSEP のジョブ定義属性を使用して形式化されます。この 8 バイトのフィールドが 16 進数のゼロ (X'0000000000000000') として渡されると、その値は *N として形式化されます。
- 協定世界時の日付およびタイム・スタンプ (*UTC)。 UTC でのシステムの日付タイム・スタンプを含む、8 バイトのフィールド。出力用に形式化された日付タイム・スタンプでは、日付の後にブランクの区切り文字と時間が表示されます。出力メッセージの日時は、ジョブに指定されたタイム・ゾーンにより UTC で調整されます。日付は、日付形式 (QDATFMT)、および日付区切り文字 (QDATSEP) のジョブ定義属性を使用して形式化されます。時間は、時間区切り文字 QTIMSEP のジョブ定義属性を使用して形式化されます。この 8 バイトのフィールドが 16 進数のゼロ (X'0000000000000000') として渡されると、その値は *N として形式化されます。
- 協定世界時タイム・スタンプ (*UTCT)。UTC でのシステムの日付タイム・スタンプを含む、8 バイトのフィールド。出力メッセージの時間は、ジョブに指定されたタイム・ゾーンにより UTC で調整され、時間区切り文字 QTIMSEP のジョブ定義属性を使用して形式化されます。この 8 バイトのフィールドが 16 進数のゼロ (X'0000000000000000') として渡されると、その値は *N として形式化されます。
- 日付およびタイム・スタンプ (*DTS)。 8 バイトのシステム日付およびタイム・スタンプ。日付はシステム値 QDATFMT および QDATSEP の指定に従って形式化され、時刻は hh:mm:ss の形式になります。
- 10 進数 (*DEC)。小数点を含む符号付き 10 進数として形式化されたパック 10 進数。 長さを指定しなければなりません。小数点以下の桁数のデフォルト値は 0 です。
- 16 進数 (*HEX)。単一引用符で囲み、その前に X を付けたストリング。ストリングの各バイトは、それぞれ 2 つの 16 進文字 (0 から 9 および A から F) に変換されます。メッセージ記述で長さが指定されていない場合には、送信元によりフィールドの長さが決定されます。
- 引用符付き文字ストリング (*QTDCHAR)。これは文字データのストリングを単一引用符で囲んだものです。この場合、先行ブランクまたは末尾ブランクは削除されません。メッセージ記述で長さが指定されていない場合には、送信元によりフィールドの長さが決定されます。
- 時間間隔 (*ITV)。各種のタイムアウト条件として使用する時間を示す 8 バイトの時間間隔 (秒未満の値は秒単位に繰り上げられます)。

- 符号なし 2 進数 (*UBIN)。符号なし 10 進整数として形式化された符号なし 2 進整数 (長さは 2、4、または 8 バイト)。長さが指定されないと、システムは 2 進整数が 2 であると想定します。

以下に示すデータ・タイプは、IBM 提供のメッセージ記述内でのみ有効であり、その他のメッセージについて使用してはなりません。

- スペース・ポインター (*SPP)。 16 バイトの、プログラム・オブジェクトを指すポインター。ダンプでは、オブジェクト内のデータは、 *HEX タイプ・データと同じように形式化されます。 *SPP は、メッセージ内の置換テキストとしては使用できません。これを使用できるのは、**メッセージ記述追加 (ADDMSGD)** コマンドの DMPLST パラメーターの一部としてだけです。
- システム・ポインター (*SYP)。 16 バイトの、システム・オブジェクトへのポインター。メッセージまたはメッセージ・ヘルプでは、オブジェクトの 10 文字の名前が、 *CHAR タイプ・データと同じように形式化されます。

重大度コードの割り当て:

メッセージ記述追加 (ADDMSGD) コマンドでメッセージに割り当てる重大度コードは、そのメッセージの重大度を示します。

重大度コードが高くなるほど、深刻な状況を示します。以下に、使用可能な重大度コードとその意味をリストします。 (これらの重大度コードとその意味は、IBM 事前定義メッセージに割り当てられた重大度コードと一致しています。)

00: 情報。 情報目的専用。エラーは検出されず、応答も不要です。このメッセージは、機能が処理中であるか、または機能が正しく完了したことを示します。

10: 警告。 潜在的なエラー条件が存在しています。プロシージャーまたはプログラムは、欠落した入力の提供など、デフォルト値が取られる場合があります。操作の結果は、成功と仮定されます。

20: エラー。 エラーが検出されますが、それはおそらく自動リカバリー・プロシージャーが適用されるものです。処理は続行されます。デフォルト値が取られて、エラーのある入力が置換される場合があります。操作の結果は、必ずしも正しいとは限りません。機能は部分的にしか完了されていないことがあります。例えば、リスト内のいくつかの項目が正しく処理されても、他の項目が失敗しています。

30: 重大エラー。 検出されたエラーが重大すぎて自動リカバリーが行えず、可能なデフォルト値がありません。このエラーがソース・データの中にある場合には、その入力レコード全体がスキップされます。プロシージャー処理またはプログラム処理の際にエラーが発生した場合、それによってプロシージャーまたはプログラムの異常終了 (重大度 40) が引き起こされます。操作結果は有効ではありません。

40: プロシージャーまたは機能の異常終了。 おそらくプロシージャーまたはプログラムが無効なデータを処理できなかったため、あるいはユーザーがそれを取り消したために、操作が終了します。

50: ジョブの異常終了。 ジョブが終了したか、開始されていませんでした。経路指定ステップが異常終了したかまたは開始に失敗したか、ジョブ・レベル機能要求通りに実行されなかったか、あるいは、ジョブが取り消された可能性があります。

60: システム状況。 システム・オペレーターによってのみ発行されます。装置、サブシステム、またはシステムの状況、あるいは警告のいずれかを示します。

70: 装置保全性。 システム・オペレーターによってのみ発行されます。このメッセージは、ある装置が誤動作状態にあるか、または何らかの動作不能状態になっていることを示します。ユーザーが障害から回復できる場合もあれば、サービス担当者の援助が必要な場合もあります。

80: システム警報。 重大度コード 80 のメッセージが、即時メッセージとして発行されます。これは、直ちにシステムを停止するほど重大でなくても、予防手段を取らないとより重大になる可能性のある条件の警告も行います。

90: システム保全性。 システム・オペレーターによってのみ発行されます。これは、サブシステムまたはシステムの操作不能にしている条件を記述するものです。

99: 処置。 応答の入力、印刷装置用紙の変更、またはディスケットの交換など、手動の処置が必要です。

関連資料:

95 ページの『SEV パラメーター』

重大度 (SEV) パラメーターは、重大度コードを指定します。

関連情報:

メッセージ記述追加 (ADDMSGD) コマンド

応答に対する妥当性検査の指定:

メッセージ記述追加 (ADDMSGD) コマンドには、照会メッセージまたは通知メッセージに対する有効な応答のタイプを指定することができます。

指定できる事項は次のとおりです (括弧内はパラメーターを示しています)。

- 応答のタイプ (TYPE)
  - 10 進数 (*DEC)
  - 文字 (*CHAR)
  - 英字 (*ALPHA)
  - 名前 (*NAME)
- 応答の最大長 (LEN)
  - 10 進数の場合は 15 桁 (小数点以下の桁数 9)
  - 文字および英字の場合は 32 文字
  - 名前の場合は 10 文字

  注: 妥当性検査 (VALUES、RANGE、REL、SPCVAL、DFT) の指定がない場合は、タイプ *CHAR および *ALPHA の場合の応答の最大長は 132 文字です。
- 応答として使用できる値
  - 値のリスト (VALUES)
  - 特殊値のリスト (SPCVAL)
  - 値の範囲 (RANGE)
  - 応答値が一致しなければならない関係 (REL)

  注: 特殊値とは、受諾されるが、どの妥当性検査値にも該当しない値のことです。

ディスプレイ装置ユーザーがメッセージへの応答を入力するとき、キーボードは下段シフトにあり、したがって小文字が入力されます。プログラムが大文字の応答を必要としている場合は、次のいずれかの処置を行うことができます。

- 小文字から大文字への変換をデフォルト値とする変換テーブル・オプションをサポートする、SNDUSRMSG を使用する。

- VALUES パラメーターに大文字だけを指定することによって、ディスプレイ装置ユーザーに大文字による入力を求める。
- VALUES 値を大文字で指定するとともに、SPCVAL パラメーターを用いて対応する小文字を大文字に変換する。
- 入力される文字がすべて英字 (A から Z) である場合は、TYPE(*NAME) を使用する。この場合、文字は大文字に変換されてから検査されます。

即時メッセージの送信および応答の処理:

この例は、プログラムまたはプロシージャーがどのようにして照会メッセージを送信し、その応答を処理するのかを示しています。

この例では、プロシージャーが次のタスクを完了します。

- 即時照会メッセージを QSYSOPR に送る。
- イエスまたはノー (Y または N) の応答を求める。
- 有効な応答が入力されたかどうか確認する。
- オペレーターが 120 秒以内に応答しなかった場合は、タイムアウトを生じさせる。

```
           PGM
           DCL        &MSGKEY *CHAR LEN(4)
           DCL        &MSGRPY *CHAR LEN(1)
SNDMSG:    SNDPGMMSG  MSG('.... Reply Y or N') TOMSGQ(QSYSOPR) +
                        MSGTYPE(*INQ) KEYVAR(&MSGKEY)
           RCVMSG     MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120) +
                        MSG(&MSGRPY)
           IF         ((&MSGRPY *EQ 'Y') *OR (&MSGRPY *EQ 'y')) DO
           .
           .
           GOTO       END
           ENDDO      /* Reply of Y */
           IF         ((&MSGRPY *EQ 'N') *OR (&MSGRPY *EQ 'n')) DO
           .
           .
           GOTO       END
           ENDDO      /* Reply of N */
           IF         (&MSGRPY *NE ' ') DO
           SNDPGMMSG  MSG('Reply was not Y or N, try again') +
                        TOMSGQ(QSYSOPR)
           GOTO       SNDMSG
           ENDDO      /* Reply not valid */
    /* Timeout occurred */
           SNDPGMMSG  MSG('No reply from the previous message +
                        was received in 120 seconds and a 'Y'' +
                        value was assumed') TOMSGQ(QSYSOPR)
           .
           .
END:       ENDPGM
```

SNDUSRMSG コマンドはタイムアウト・オプションをサポートしない (応答を受け取るかまたはジョブが取り消されるまで待機する) ので、このプロシージャーで代わりに SNDUSRMSG コマンドを使用することはできません。

SNDPGMMSG コマンドはメッセージを送信します。コマンドには KEYVAR パラメーターが指定されます。これにより、このメッセージを固有のものとし、応答が RCVMSG コマンドと正しく対応付けられるようにする、メッセージ参照キーを返します。 KEYVAR の値は、長さ 4 の文字フィールドとして定義しなければなりません。

RCVMSG コマンドでは、特定のメッセージ (応答) を受け取るために、MSGKEY パラメーターに、SNDPGMMSG コマンドからのメッセージ参照キーの値を指定します。応答は MSG パラメーターに返されます。 WAIT パラメーターには、タイムアウトが生じるまでに応答を待つ時間が指定されています。

応答を受け取ると、Y または N の値が大文字であるか小文字であるかを、プログラムまたはプロシージャー・ロジックが検査します。通常、オペレーターはその文字を小文字の値として入力します。オペレーターが Y または N 以外の非ブランク文字を入力した場合には、プログラムまたはプロシージャーにより別のメッセージが送られ、照会メッセージが繰り返されます。

オペレーターがブランクを入力すると、プログラムまたはプロシージャーに応答は送られません。プログラムまたはプロシージャーにブランクが返される場合は、タイムアウトが生じます (オペレーターが応答を返しません)。プログラムまたはプロシージャーは、応答が受信されず、デフォルト値が取られたことを示すメッセージをシステム・オペレーターに送ります (Y 値は、メッセージ待ち行列で Y と示されます)。デフォルト値による Y の値は応答として表示されないため、メッセージ待ち行列を調べるだけでは、そのメッセージに対して応答する必要があるのか、または既にタイムアウトが生じているのかを判別できません。また、いったん送られたメッセージを、プログラムまたはプロシージャーがメッセージ待ち行列から除去する方法もありません。 2 番目のメッセージは、このような事態が生じるのを最小限に抑え、何が起きたかについての監査証跡を提供するためのものです。

タイムアウトが生じてからオペレーターがメッセージに応答しても、その応答は無視されます。応答が無視されたことはオペレーターには通知されません。

関連タスク:

614 ページの『送信側コピー・メッセージを使用して応答を取得する』

照会メッセージが送信されると、応答が要求されます。照会メッセージの送信者が応答を取得できるようにするため、送信側コピー・メッセージが発行され、内部的に照会メッセージと関連付けられます。

2 バイト文字を含む即時メッセージの送信:

2 バイト文字テキストを含む即時メッセージを送信する場合は、テキストの長さを 2 バイト文字 37 個にシフト制御文字を加えたものに制限してください。メッセージのサイズを制限することにより、メッセージを正しく表示することができます。

応答のデフォルト値の定義:

メッセージ記述追加 (ADDMSGD) コマンドを使用して、ユーザー定義のメッセージに対する応答のデフォルト値を指定することができます。

デフォルトの応答は、そのメッセージに対する他の応答と同じ妥当性検査の値を満たしているか、またはメッセージ記述に特殊値として指定されていなければなりません。デフォルト値が使用されるのは、ユーザーのメッセージ待ち行列に送信されたすべての照会メッセージに対してデフォルトの応答を返すことを、ユーザーが (CHGMSGQ コマンドで) 指定した場合です。また、未応答の照会メッセージが削除された場合もデフォルト応答が送信されます。例えば、ワークステーション・ユーザーは、DSPMSG コマンドを用いてメッセージを表示し、F13 (すべての除去) または F11 (メッセージの除去) を押して未応答の照会メッセージを削除します。

このほかにも、デフォルト応答は、ジョブ属性 INQMSGRPY が *DFT に設定されている場合にも使用され、またその値が *SYSRPYL に設定されている場合にも使用されます。デフォルト応答は、システム応答リストを用いて変更することができます。

デフォルト応答は「プログラム・メッセージの表示」画面 (*EXT) に送られたメッセージを表示する画面) でも使用されます。デフォルト応答の送信が行われるのは、次の 2 つの条件のどちらかが発生したときです。

- 「プログラム・メッセージの表示」画面に未応答の照会メッセージが表示され、ユーザーがキー入力による応答を何もせずに Enter (続行) を押した場合。
- ユーザーが F3 キーを押して、「プログラム・メッセージの表示」画面を終了させた場合。

エスケープ・メッセージに対するデフォルトのメッセージ処理の指定:

エスケープ・メッセージとして送信できるメッセージを作成する場合は、そのメッセージが送信されたときに別の方法で処理されない場合に使用される、デフォルトのメッセージ処理処置をセットアップすることができます。

デフォルト・メッセージ処理処置は、次の事項から構成されます。

- デフォルト・プログラム名。メッセージ処理のためのデフォルト処置を行うために呼び出されるプログラム。デフォルト・プログラムには、次のパラメーターが渡されます。
  - 呼び出しメッセージ待ち行列名。このパラメーターは、メッセージの送信元を示す、多くのフィールドから構成されます。
  - メッセージ参照キー (4 文字)。呼び出しメッセージ待ち行列上のエスケープ・メッセージのメッセージ参照キー。
- ダンプ・リスト。 ダンプを取るオブジェクトを示すメッセージ・データ・フィールド番号 (置換変数の番号と同じ) のリスト。

  さらに、次の項目もダンプできます。
  - ジョブのデータ域
  - ジョブの内部マシン・データ構造
  - ジョブ

    ジョブのダンプ・リストを指定することは、**ジョブ表示 (DSPJOB)** コマンドにパラメーター JOB(*) OUTPUT(*PRINT) を指定することと同じです。

メッセージ記述にデフォルト処置を指定しないと、ジョブのダンプ・リストが取られます (DSPJOB JOB(*) OUTPUT(*PRINT) と同じ)。

メッセージに指定するデフォルト処置が取られるのは、エスケープ・メッセージが処理されることなくメッセージ・パーコレーション処置が完了した後だけです。

関連タスク:

603 ページの『非監視メッセージの CL 処理』
ユーザーが監視を指定しなかったメッセージについては、システムがデフォルト値による監視と処理を行います。

関連情報:

メッセージの処理 API

出口プログラム

例: 最後の診断メッセージをエスケープ・メッセージとして送信する:

この例は、最後の診断メッセージをエスケープ・メッセージとして送信するプログラムです。

次のプログラムは、エスケープ・メッセージが後に続く診断メッセージが送信された場合に使用できるデフォルト・プログラムの例です。このプログラムは、この単一 CL プロシージャーを持つ、オリジナル・プログラム・モデル (OPM) CL プログラムまたは統合言語環境 (ILE) プログラムにすることができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
            PGM        PARM(&MSGQ &MRK)
            DCL        VAR(&MRK) TYPE(*CHAR) LEN(4)
            DCL        VAR(&MSGQ) TYPE(*CHAR) LEN(6381)
            DCL        VAR(&QNAME) TYPE(*CHAR) LEN(4096)
            DCL        VAR(&MODNAME) TYPE(*CHAR) LEN(10)
            DCL        VAR(&BPGMNAME) TYPE(*CHAR) LEN(10)
            DCL        VAR(&BLANKMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
            DCL        VAR(&DIAGMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
            DCL        VAR(&SAVEMRK) TYPE(*CHAR) LEN(4)
            DCL        VAR(&MSGID) TYPE(*CHAR) LEN(7)
            DCL        VAR(&MSGDTA) TYPE(*CHAR) LEN(100)
            DCL        VAR(&MSGF) TYPE(*CHAR) LEN(10)
            DCL        VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
            DCL        VAR(&OFFSET) TYPE(*DEC)
            DCL        VAR(&LENGTH) TYPE(*DEC)

            /* Check for OPM program type                          */

            IF         (%SST(&MSGQ 277 1) *EQ '0') THEN(DO)
              CHGVAR     VAR(&QNAME) VALUE(%SST(&MSGQ 1 10))
              CHGVAR     VAR(&MODNAME) VALUE('*NONE')
              CHGVAR     VAR(&BPGMNAME) VALUE('*NONE')
             ENDDO
            ELSE DO
            /* Not an OPM program; always use the long procedure name */
              CHGVAR     VAR(&OFFSET) VALUE(%BIN(&MSGQ 281 4))
              CHGVAR     VAR(&LENGTH) VALUE(%BIN(&MSGQ 285 4))
              CHGVAR     VAR(&QNAME) VALUE(%SST(&MSGQ &OFFSET &LENGTH))
              CHGVAR     VAR(&MODNAME) VALUE(%SST(&MSGQ 11 10))
              CHGVAR     VAR(&BPGMNAME) VALUE(%SST(&MSGQ 1 10))
             ENDDO
GETNEXTMSG: CHGVAR     VAR(&SAVEMRK) VALUE(&DIAGMRK)
            RCVMSG     PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
                         MSGTYPE(*DIAG) RMV(*NO) KEYVAR(&DIAGMRK)
            IF         (&DIAGMRK *NE &BLANKMRK) THEN(GOTO GETNEXTMSG)
            ELSE IF (&SAVEMRK *NE ' ') THEN(DO)
            /* If no diag message is sent, no message is sent to the previous program */
                RCVMSG     PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
                           MSGKEY(&SAVEMRK) RMV(*NO) MSGDTA(&MSGDTA) +
                           MSGID(&MSGID) MSGF(&MSGF) MSGFLIB(&MSGLIB)
                SNDPGMMSG  MSGID(&MSGID) MSGF(&MSGLIB/&MSGF) +
                           MSGDTA(&MSGDTA) TOPGMQ(*PRV (&QNAME +
                           &MODNAME &BPGMNAME))
                           MSGTYPE(*ESCAPE)
              ENDDO
            ENDPGM
```

このプログラムは、すべての診断メッセージを FIFO 順で受け取ります。さらに、最後の診断メッセージをエスケープ・メッセージとして送信し、前のプログラムがそのメッセージを監視できるようにします。

警報オプションの指定:

メッセージ記述追加 (ADDMSGD) コマンドでは、警報オプションを指定して、あるメッセージが出されたときに警報が作成されるようにできます。

警報オプションが指定されたメッセージが出されると、SNA 警報が作成され、問題管理機能フォーカル・ポイントに送られます。 メッセージに対して作成する警報は、**警報記述の追加 (ADDALRD)** コマンドを使用して定義できます。

関連情報:

 DSNX Support (PDF)

例: メッセージの記述:

この例は、アプリケーションで使用されるメッセージの作成方法を示しています。

次の例では、受注業務などのアプリケーションで使用されるメッセージを、**メッセージ記述追加 (ADDMSGD)** コマンドで作成します。 このメッセージの例は、画面に入力された得意先番号が見つからない場合に出されるもので、メッセージ・テキストは次のとおりです。

```
Customer number &1 not found
```

このメッセージの **メッセージ記述追加 (ADDMSGD)** コマンドは、次のとおりです。

```
ADDMSGD   MSGID(USR4310) +
          MSGF(QGPL/USRMSG)  +
          MSG('Customer number &1 not found') +
          SECLVL('Change customer number') +
          SEV(40) +
          FMT((*CHAR 8))
```

メッセージは、QGPL ライブラリー内の USRMSG ファイルに追加されます。

DSPMSGD または WRKMSGD コマンドを使用すると、メッセージ記述を印刷または表示することができます。

SECLVL パラメーターにより、非常に簡単なテキストが提供されます。このテキストを詳細メッセージ情報の画面に表示するには、SECLVL('メッセージ・テキスト') を指定します。このパラメーターに指定したテキストは、そのメッセージにカーソルを置きヘルプ・キーを押すと、詳細メッセージ情報の画面に表示されます。

2 バイト・メッセージの定義:

2 バイト・テキストを含むメッセージを定義するには、**メッセージ記述追加 (ADDMSGD)** コマンドを使用して CL プロシージャーまたはプログラムを作成します。定義したメッセージはメッセージ・ファイルに入れられ、通常と同じように送信されます。

プログラムを作成するときは、次のステップに従ってください。

1. プログラムの入ったソース・ファイルが 2 バイト・ファイルであるかどうか確認してください。 **ソース物理ファイル作成 (CRTSRCPF)** コマンドで IGCDTA(*YES) を指定します。
2. 原始ステートメント入力ユーティリティー (SEU) を使用してプログラムを入力します。 2 バイト文字を用いる CL コマンドは、SEU の使用以外に入力方法はありません。このため、2 バイト・メッセージは CL プログラム内で作成しなければなりません。
3. メッセージ全体が正しく表示または印刷されるようにするためには、メッセージの長さを 37 個の 2 バイト文字に制限してください。

   メッセージ・モニター (MONMSG) コマンドを使用する場合には、比較データ (CMPDATA) パラメーターも 6 文字の 2 バイト文字に制限してください。

4. 英数字メッセージ・ファイルが 2 バイト・メッセージ・ファイルで置き換えられる場合 (2 バイト・ディスプレイ装置にのみ送られる変換済みメッセージのファイルなど) は、英数字メッセージ・ファイルを一時変更するために、次のようなコマンドを入力してください。

   ```
   OVRMSGF MSGF(QCPFMSG) TOMSGF(DBCSLIB/QCPFMSG)
   ```

   2 バイト・メッセージは、2 バイト・ディスプレイ装置でしか表示できません。

メッセージの表示

メッセージ記述処理 (WRKMSGD) または**メッセージ記述表示 (DSPMSGD)** コマンドを使用して、メッセージ・ファイル内の 1 つのメッセージまたは一連のメッセージ記述を表示または印刷します。

メッセージ記述表示 (DSPMSGD) コマンドの使用例を次に示します。

```
DSPMSGD   RANGE(*FIRST  IDU0571)  MSGF(QIDU/QIDUMSG)
          FMTTXT(*NO)  OUTPUT(*PRINT)
```

コマンドを使用して生成可能な例外メッセージを参照するには、コマンドの記述を参照してください。

メッセージ・ファイル検索

システムは、メッセージ記述の検索元のメッセージ・ファイルを検索します。

メッセージ・ファイルからメッセージを取り出すために検索を実行するとき、システムは次の 2 つのステップを使用します。

1. システムは、メッセージ・ファイル名に適用されるすべての一時変更を処理します。
2. メッセージ・ファイル名が一時変更されていない場合、システムはメッセージ・ファイル名に基づいてそのメッセージ・ファイルを検索し、さらにメッセージ使用時に指定されたライブラリーを検索します。

システム・メッセージ・ファイル検索:

メッセージ・ファイルが送信された際に指定されたメッセージ・ファイル名およびライブラリーは、メッセージ記述の検索元のメッセージ・ファイルの検索に使用されます。

メッセージ・ファイルが一時変更されていない場合、およびメッセージ・ファイル名が一時変更されているが、一時変更されたファイルにメッセージ ID が入っていない場合も同様です。

システム検索は、メッセージ・ファイル・ライブラリーを *CURLIB と *LIBL のどちらとして指定するかによって異なります。以下は、*CURLIB および *LIBL についての検索パスの説明です。

- メッセージ・ファイル・ライブラリーを *CURLIB として指定するか、または明示的に指定する場合

  システムは、指定されたライブラリーまたはジョブの現行ライブラリー (*CURLIB) 内で、指定されたメッセージ・ファイルを検索します。
- メッセージ・ファイル・ライブラリーを *LIBL として指定する場合

  システムは、ジョブのライブラリー・リスト (*LIBL) 内で、指定されたメッセージ・ファイルを検索します。指定された名前を持つ最初のメッセージ・ファイルが検出されると、検索は停止されます。

メッセージ・ファイルが検出されたが、それにメッセージ識別コードの記述が入っていない場合は、欠落しているメッセージ記述の代わりに、QCPFMSG 内のメッセージ CPF2457 のメッセージ属性とテキストが使用されます。

メッセージ・ファイルが検出されない場合、システムは、メッセージの送信時に使用されたメッセージ・ファイルからのメッセージの取り出しを試みます。

注: メッセージ・ファイルが検出されても、損傷あるいは権限問題のためにアクセスできないことがあります。

メッセージ・ファイルの一時変更:

メッセージ・ファイルの一時変更の作成 (メッセージ・ファイル一時変更コマンド)、削除 (一時変更削除コマンド)、および表示 (一時変更表示コマンド) 操作は、他のタイプの一時変更操作と類似しています。 ただし、これらのコマンドでは、メッセージ・ファイルの属性ではなく、名前だけが一時変更され、一時変更操作適用の規則は若干異なります。

メッセージ・ファイルを一時変更するには、 **メッセージ・ファイル一時変更 (OVRMSGF)** コマンドを使用します。一時変更されるファイルは、MSGF パラメーターに指定し、それに置き換わるファイルは、TOMSGF パラメーターに指定します。

例えば、QCPFMSG を USRMSGF という名前のユーザー・メッセージ・ファイルに一時変更する場合は、次のコマンドを使用します。

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSGF)
```

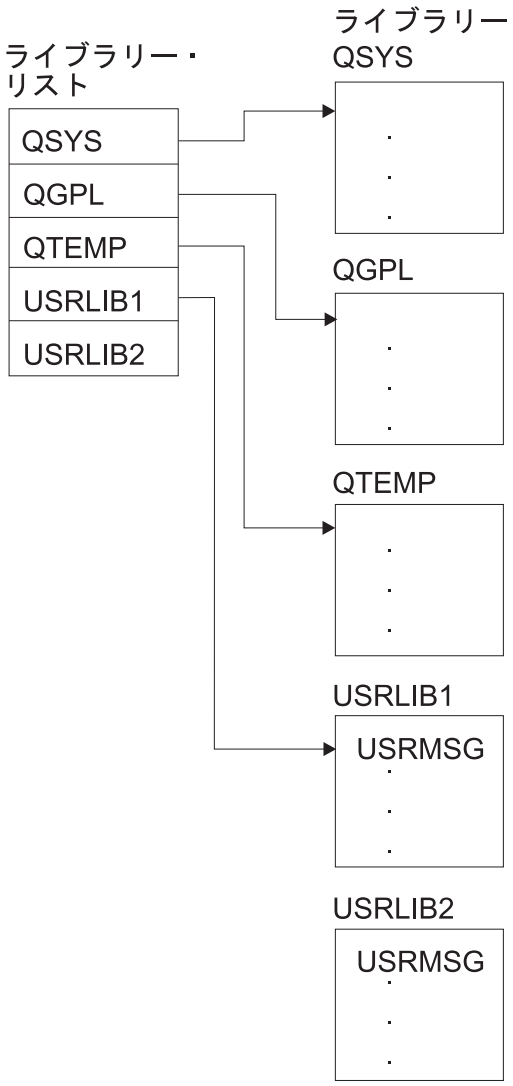
事前定義されたメッセージが検索または表示されるときは、メッセージ記述を見つけるために一時変更後のファイルが検索されます。そのファイルでメッセージ記述が検出されない場合は、一時変更前のファイルが検索されます。

メッセージ・ファイルを一時変更するには、次のような理由がいくつかあります。

- デフォルトの応答またはダンプ・リストを変更するため。元のメッセージ記述に指定されているデフォルト応答またはダンプ・リストが必要な条件を満たしていない場合は、そのメッセージに対して異なる応答またはダンプ・リストを指定したメッセージ記述に基づいて、別のメッセージ・ファイルを作成することができます。このようにしていくつかの異なる操作環境を設定し、それぞれに異なるデフォルト応答を適用することができます。
- メッセージの重大度レベルを変更するため。
- デフォルト・プログラムを用意するため。
- メッセージのテキストを変更するため。テキストがブランクの場合は、外見上ユーザーには何もメッセージが送られなかったのと同じです。例えば、 **ファイル・コピー (CPYF)** コマンドが発行する状況メッセージを表示しないようにすることができます。
- メッセージを自国語に変換するため。英語のメッセージ・ファイルを他の言語のメッセージ・ファイルに一時変更することができます。 (すべてのメッセージを変更する場合は、メッセージ・ファイルの一時変更を行うのではなく、ジョブのライブラリー・リストを使用してメッセージ・ファイルの順序を変更してください。)

このほかにも、メッセージ検索を行うメッセージ・ファイルを選択する方法として、ジョブのライブラリー・リスト上でファイルの順序を変更する方法があります。ただし、この方法を用いる場合、メッセージの検索は、指定された名前を持つ最初のメッセージ・ファイルが検出された時点で終了します。そのファイルに該当のメッセージがなくても、検索は停止されます。

例えば、ライブラリー USRLIB1 に USRMSG という名前のメッセージ・ファイルがあり、ライブラリー USRLIB2 にも USRMSG という名前の別のメッセージ・ファイルがあるとします。この場合、USRLIB1 のメッセージ・ファイルを使用したければ、ライブラリー・リスト上で USRLIB1 が USRLIB2 より前になるようにします。



RBAFN507-0

システムは、正しい名前を持つ最初のメッセージ・ファイルを検索します。そのファイルにメッセージが入っていない場合、検索は停止されます。しかし、OVRMSGF コマンドを使用すると、システムは一時変更済みのファイルを検索し、さらにメッセージがそこにない場合は、一時変更前のファイルを検索します。

例: メッセージ・ファイルの一時変更:

この例は、ジョブで使用されるメッセージの変更方法を示しています。

あるジョブで使用する IBM 提供のメッセージを変更したいとします。例えば、メッセージ CPC2191 のテキストは次のとおりです。

```
YYY のタイプ *ZZZ のオブジェクト XXX が削除された。
```

これを次のように変更したいとします。

```
YYY のオブジェクト XXX が削除された。
```

FMT の記述方法の詳細は、CPC2191 の詳細記述を表示すればわかります。

まず、次のようにメッセージ・ファイルを作成します。

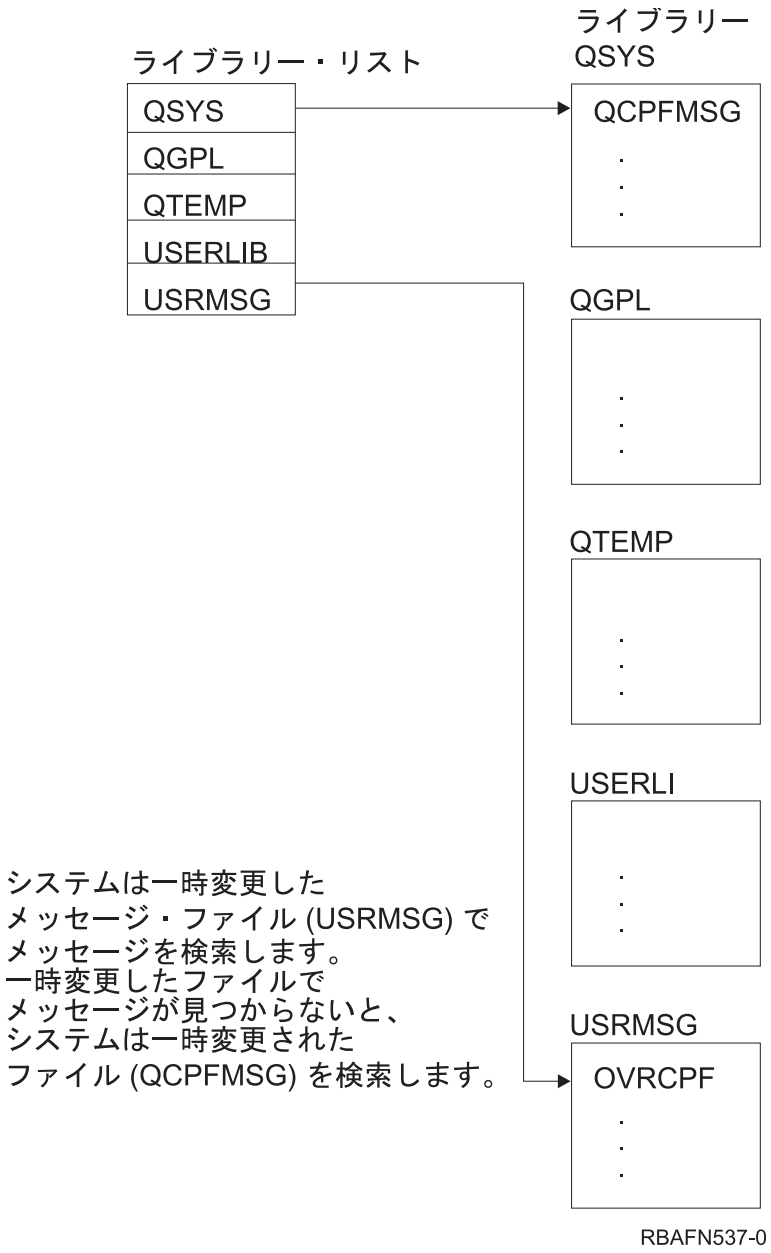
```
CRTMSGF MSGF(USRMSG/OVRCPF)
```

次に、メッセージ CPC2191 を基底として新たにメッセージを作成し、それをメッセージ・ファイルに追加します。

```
ADDMSGD  MSGID(CPC2191) MSGF(USRMSG/OVRCPF) +
         MSG('Object &1 in &2 deleted') +
         SEV(00) FMT((*CHAR 10) (*CHAR 10))
```

次に、そのジョブを実行する時点で、**メッセージ・ファイル一時変更 (OVRMSGF)** コマンドを使用して、メッセージ・ファイルを一時変更します。

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSG/OVRCPF)
```



このメッセージをすべてのジョブで使用できるように変更したい場合は、 **メッセージ記述変更 (CHGMSGD)** コマンドを使用して、メッセージを変更できます。この場合、システムのメッセージ・ファイルを一時変更する必要はありません。

CHGMSGD コマンドを使用して IBM 提供のメッセージを変更する場合、システムの新規リリースを導入するたびにそのメッセージを変更する必要があります。メッセージを再変更するときは、行いたい変更をすべて 1 つの入力ストリームまたはプログラムに入れて、随時実行可能な状態にできます。

一時変更後のファイルをさらに一時変更することもできます。例えば、1 つのジョブ内で次のような **メッセージ・ファイル一時変更 (OVRMSGF)** コマンドを指定することができます。

```
OVRMSGF  MSGF(MSGFILE1)  TOMSGF(MSGFILE2)
OVRMSGF  MSGF(MSGFILE2)  TOMSGF(MSGFILE3)
```

まず、MSGFILE1 が MSGFILE2 に一時変更されます。次に、MSGFILE2 が MSGFILE3 に一時変更されます。メッセージの送信時、ファイルはこの順序で検索されます。

1. MSGFILE3
2. MSGFILE2
3. MSGFILE1

メッセージ・ファイルの一時変更を防止することもできます。その場合は、**OVRMSGF** コマンドに SECURE パラメーターを指定しなければなりません。

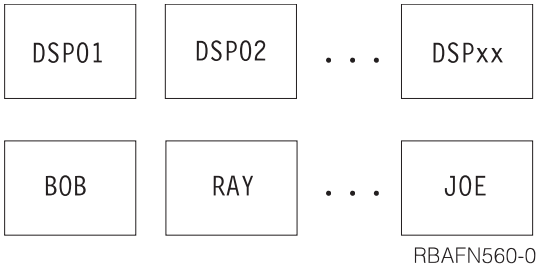
メッセージ待ち行列

システム上の全メッセージは、メッセージ待ち行列に送信されます。そのメッセージ待ち行列に関連するシステム・ユーザーまたはプログラムは、その待ち行列からメッセージを受信します。同様に、メッセージに対する応答も、その応答を要求しているユーザーまたはプログラムのメッセージ待ち行列に送り返されます。

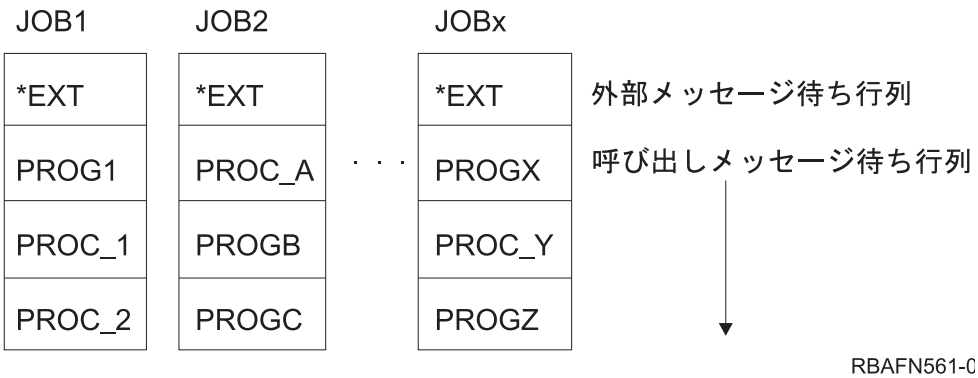
メッセージ待ち行列のタイプ

システムには、異なるタイプのメッセージ待ち行列 (ワークステーション・メッセージ待ち行列、ユーザー・プロファイル・メッセージ待ち行列、ジョブ・メッセージ待ち行列、システム・オペレーター・メッセージ待ち行列、および活動記録メッセージ待ち行列) があります。

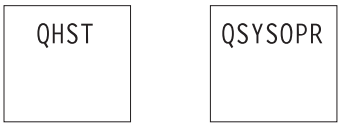
次の図は、IBM 提供のメッセージ待ち行列を示しています。メッセージ待ち行列はディスプレイ装置ごと、およびユーザー・プロファイルごとにそれぞれ 1 つずつ用意されています (この場合、DSP01 および DSP02 はディスプレイ装置名であり、BOB および RAY はユーザー・プロファイル名です)。



ジョブ・メッセージ待ち行列は、システム上で実行される各ジョブごとに提供されます。ジョブごとに外部メッセージ待ち行列 (*EXT) が設定され、同時にジョブ内でのオリジナル・プログラム・モデル (OPM) プログラムまたは統合言語環境 (ILE) プロシージャーの呼び出しごとに、そのプログラムまたはプロシージャーの呼び出しメッセージ待ち行列が用意されます。



さらに、システム活動記録ログ用のメッセージ待ち行列 (QHST)、およびシステム・オペレーター用のメッセージ待ち行列 (QSYSOPR) も用意されています。



RBAFN562-0

これらのメッセージ待ち行列は次のように使用します。

- ワークステーション・メッセージ待ち行列は、ワークステーション・ユーザー相互間、およびワークステーション・ユーザーとシステム・オペレーター間のメッセージ送受信に使用します。この待ち行列の名前は、ワークステーションの名前と同じです。この待ち行列は、システムに対してワークステーションが定義された時点で、システムにより作成されます。
- ユーザー・プロファイル・メッセージ待ち行列は、ユーザー相互間の交信に使用されます。この待ち行列は、ユーザー・プロファイルを作成した時点で、自動的にライブラリー QUSRSYS 内に作成されます。
- ジョブ・メッセージ待ち行列は、処理されるべき要求 (コマンドなど) を受けるため、およびその処理結果として生じるメッセージを送るために使用されます。メッセージはジョブの要求元に送られます。ジョブ・メッセージ待ち行列は各ジョブごとに用意され、そのジョブが存在している間だけ存在します。ジョブ・メッセージ待ち行列は、外部メッセージ待ち行列 (*EXT) および呼び出しスタック項目のメッセージ待ち行列の集合から構成されます。
- システム・オペレーター待ち行列 (QSYSOPR) は、システム、ディスプレイ装置ユーザー、およびアプリケーション・プログラムからのメッセージの受け取りと、それに対する応答のために使用されます。
- 活動記録ログ・メッセージ待ち行列は、システム内のジョブに対して使用され、高レベルのシステム活動の記録を持ちます。

これらのメッセージ待ち行列以外にも、システム・ユーザーにメッセージを送るため、およびアプリケーション・プログラム間でメッセージを受け渡しするための独自のメッセージ待ち行列を作成することができます。

関連概念:

563 ページの『ジョブ・メッセージ待ち行列』
ジョブ・メッセージ待ち行列は、システム上のジョブごとに作成され、そのジョブのすべてのメッセージ要件を処理します。単一ジョブのジョブ・メッセージ待ち行列は、外部メッセージ待ち行列 (*EXT) および一連の呼び出しメッセージ待ち行列から構成されます。

関連タスク:

536 ページの『メッセージ』
メッセージは、ユーザーとプログラム間の通信に使用されます。

メッセージ待ち行列の作成または変更

独自のユーザー待ち行列を作成するには、 **メッセージ待ち行列作成 (CRTMSGQ)** コマンドを使用します。また、**メッセージ待ち行列変更 (CHGMSGQ)** コマンドを使用して、既存のユーザー・メッセージ待ち行列の属性の一部を変更することもできます。メッセージ待ち行列の内容を表示するには、**メッセージの表示 (DSPMSG)** または **メッセージ処理 (WRKMSG)** コマンドを使用します。

メッセージ待ち行列の属性には次のものがあります。

- メッセージ待ち行列への変更を直ちにディスクへ書き込まなければならないかどうか。変更が直ちにディスク装置に書き込まれると、システム障害などが生じてもメッセージは消失されません。ただし、これを行うとシステム・パフォーマンスが低下することがあります。
- メッセージ待ち行列に着信したメッセージの転送方式。メッセージ待ち行列の作成時点では、転送の方式は保留転送として定義されます。ディスプレイ装置がサインオンされるときには、ユーザーのメッセージ待ち行列は、そのユーザーのユーザー・プロファイルで指定されているモードに設定されます。**CHGMSGQ** コマンドでは、次のいずれかの転送タイプを指定することができます。
  - 中断転送。ジョブが中断されて、メッセージを転送するためのプログラムが呼び出されます。中断転送を要求する **CHGMSGQ** コマンドにユーザー・プログラムが指定されていない場合、または *SAME が指定されている場合は、**DSPMSG** コマンドによって自動的にメッセージが表示されます。ジョブに対する中断メッセージは、**ジョブ変更 (CHGJOB)** コマンドの BRKMSG パラメーターによって制御できます。
  - 通知転送。メッセージが待ち行列に着信していることが、メッセージ標識または音響警報 (あるいはその両方) によってディスプレイ装置ユーザーに通知されます。ディスプレイ装置ユーザーは、**DSPMSG** または **WRKMSG** コマンドを使用することによってメッセージを表示できます。
  - 保留転送。ディスプレイ装置ユーザーが **DSPMSG** または **WRKMSG** コマンドを使用することによってメッセージを要求するまで、メッセージ待ち行列はメッセージを保持します。
  - デフォルト転送。メッセージはすべて無視され、応答を必要とするメッセージに対してはデフォルト応答が送られます。
- 中断転送の場合のメッセージの処理方法
  - **DSPMSG** コマンドを自動的に実行する。対話式ジョブの場合は、重大度コードが高ければ、メッセージがディスプレイ装置に表示されます。バッチ・ジョブの場合は、重大度コードが高ければ、メッセージはスプール印刷装置ファイルにリストされます。
  - メッセージを処理する中断処理プログラムを呼び出す。この場合は、**CHGMSGQ** コマンドを使用して、呼び出されるプログラムを指定し、転送方式を中断転送モードに設定しなければなりません。中断処理プログラムによって中断モードにある間に、他のジョブが待ち行列上にある照会メッセージに応答できるようにするかどうかを指定することもできます。
- 中断転送または通知転送の場合に、メッセージを選別するための重大度コード。指定した最低の重大度コードと同じか、またはそれより大きい重大度コードを持つメッセージだけが表示されます。待ち行列作成の時点では、最低の重大度コードは 00 に設定されます。最低の重大度コードを変更するときは、**CHGMSGQ** コマンドを使用しなければなりません。

  DSPMSG コマンドを使用してメッセージ待ち行列のメッセージを表示するときは、重大度コード・フィルター (SEV) パラメーターを用いて、表示するメッセージをフィルターできます。作成時にメッセージ待ち行列に指定される重大度フィルターよりも、このフィルターがよく使用されます。このフィルターを使用するには、DSPMSG SEV(*MSGQ) を指定してください。 **DSPMSG** コマンドを使用すると、中断メッセージおよび通知メッセージのフィルターに使用される現行重大度コードが判別されます。このコードは、メッセージ画面の見出し行に表示されます。
- メッセージ待ち行列に関連するコード化文字セット ID (CCSID)。この待ち行列に送信されるメッセージは、この CCSID に変換されます。メッセージ待ち行列 CCSID が 65534 または 65535 の場合、変換は行われません。メッセージ待ち行列 CCSID が 65534 の場合、各メッセージには固有の CCSID が入ります。これは送信元によって確立されるものです。
- 標準メッセージ待ち行列の警報許可。警報許可を指定するのは、作成する待ち行列によって、そこに送信される警報メッセージから警報を作成できる場合です。
- メッセージ待ち行列がいっぱいになったときの処置。

– いっぱいになった待ち行列にメッセージを送信したプログラムまたはユーザーに、CPF2460 (メッセージ待ち行列を拡張できない) を送信します。
– 待ち行列を折り返します。折り返しが行われると待ち行列上にある古いメッセージが除去され、待ち行列に送られてきた新しいメッセージのスペースが用意されます。

メッセージ待ち行列 QHST については、この属性を変更することはできません。QHST がいっぱいになると無条件で CPF2460 が送られてきます。 IBM は、この属性をラップに設定して QSYSOPR を出荷します。

注**:** ワークステーションの装置記述が作成される時点で、システムはその装置に対する処置を要求するすべてのメッセージを受け取るために、その装置関連のメッセージ待ち行列を設定します。ワークステーション印刷装置、テープ駆動装置、および APPC 装置の場合は、装置記述の作成時点で、MSGQ パラメーターを使用してメッセージ待ち行列を指定することができます。これらの装置に対してメッセージ待ち行列を指定しなかった場合には、デフォルト値により、QSYSOPR がそのメッセージ待ち行列として使用されます。その他の装置はすべて、作成時点で QSYSOPR メッセージ待ち行列に割り当てられます。

ユーザー・プロファイルで定義されているメッセージ待ち行列は、ユーザー・メッセージ待ち行列と呼びます。ユーザーがそのユーザー・プロファイルを使用してシステムにサインオンすると、ユーザー・メッセージ待ち行列は、ユーザー・プロファイルで指定されている転送モードになります。

ユーザーがあるディスプレイ装置にサインオンしたときに、ユーザー・メッセージ待ち行列が中断転送モードまたは通知転送モードになっている場合は、次に別のディスプレイ装置にサインオンしても、ユーザー・メッセージ待ち行列の転送モードは変更されません。またあるジョブでユーザー・メッセージ待ち行列が中断転送モードまたは通知転送モードになっている場合に、別のジョブでそのメッセージ待ち行列の転送モードを変更することはできません (これはワークステーション・メッセージ待ち行列および QSYSOPR メッセージ待ち行列の場合も同じです)。

ユーザーがディスプレイ装置からサインオフするか、またはジョブが予測外に終了すると、ユーザー・メッセージ待ち行列の転送モードが保留モードに変更されます (このジョブについてのユーザー・メッセージ待ち行列の転送モードが中断モードまたは通知モードになっている場合)。また、2 次ジョブに移った場合にも、ユーザー・メッセージ待ち行列の転送モードは、中断モードまたは通知モードから保留モードに変更されます。これを行うには、**2 次ジョブへの移行 (TFRSECJOB)** コマンドを使用するか、あるいはシステム要求キーを押し、システム要求メニューのオプション 1 を指定します。

2 次ジョブに移行した後は、ユーザーは自分のユーザー・プロファイルを使用してサインオンします。ユーザー・メッセージ待ち行列は、そのユーザー・プロファイル内で指定されている転送モードになります。したがって、ユーザー・メッセージ待ち行列も 2 次ジョブに移行することができます。ユーザーはこのようにして、ユーザー・メッセージ待ち行列を用いて 2 つのジョブ間を切り換えることができます。

しかし、代替ジョブに移った後に、自分のユーザー・プロファイル以外のユーザー・プロファイルを使用してサインオンした場合は、移行元のジョブのユーザー・メッセージ待ち行列は、保留転送モードのままになります。サインオンに使用したユーザー・プロファイルのユーザー・メッセージ待ち行列は、そのユーザー・プロファイル内に指定されている転送モードになります。このように、ユーザー・メッセージ待ち行列が、他のユーザーによって中断モードまたは通知転送モードにされることがあります。本来のユーザーが最初のジョブに戻ったときに、まだそのユーザーのユーザー・メッセージ待ち行列が他のユーザーによってその転送モードにされたままになっている場合は、そのユーザー・メッセージ待ち行列の転送モードをもとの転送モードに戻すことはできません。

QSYSOPR メッセージ待ち行列は、特に変更しない限りシステム・オペレーター用のメッセージ待ち行列です。システム・オペレーターの場合にも、上記のような状況が生じることがあります。

関連情報:

メッセージ待ち行列作成 (CRTMSGQ) コマンド

メッセージ待ち行列変更 (CHGMSGQ) コマンド

メッセージ表示 (DSPMSG) コマンド

メッセージ処理 (WRKMSG) コマンド

独立 **ASP** 内のメッセージ待ち行列**:**

IBM では、独立補助記憶域プール (ASP) 内のメッセージ待ち行列は中断モードにしないように推奨しています。

メッセージ待ち行列が中断モードの場合、メッセージがメッセージ待ち行列に送信される際にスレッドのライブラリー名スペースにメッセージ待ち行列がないなら、中断プログラムは呼び出されません。スレッドの ASP グループの独立 ASP 内のライブラリーと、システム ASP (ASP 番号 1) 内のライブラリー、および基本ユーザー ASP (ASP 番号 2-32) は、スレッド用のライブラリー名スペースを構成します。

照会メッセージをメッセージ待ち行列に送信する際に、宛先メッセージ待ち行列と応答メッセージ待ち行列の両方がシステム ASP または同じ独立 ASP 内のどちらかにある必要があります。そうしないと、どちらかのメッセージ待ち行列がオフラインになっていた場合、応答が応答メッセージ待ち行列に送信されないことがあります。

以下の状態ではメッセージを受信することができません。

- オフに構成変更されている独立 ASP 内の待ち時間のあるメッセージ待ち行列から
- オフに構成変更されている独立 ASP 内のメッセージ待ち行列に送信された照会メッセージに対する応答として

中断処理プログラムは、スレッド用のライブラリー名スペースを変更できません。

関連情報:

独立ディスク・プールの例

中断モードにあるメッセージ待ち行列**:**

中断処理プログラムは、中断転送モードにあるメッセージ待ち行列に着信したメッセージの重大度コードが、重大度コード・フィルターと同じかそれよりも高い場合は、いつでも呼び出すことができます。

中断処理プログラムを要求するには、そのプログラムの名前と中断転送を、同じ**メッセージ待ち行列変更 (CHGMSGQ)** コマンドで指定する必要があります。中断処理プログラムは、**メッセージ受信 (RCVMSG)** コマンドによってメッセージを受信する必要があります。これによってメッセージは処理済みのマークが付けられ、プログラムは再度呼び出されなくなります。 IBM 提供の中断処理プログラムは、デフォルトで **CHGMSGQ** コマンドで使用することができます。例を以下に示します。

```
CHGMSGQ MSGQ(name) DLVRY(*break)
```

注**:** 中断されたプログラムがディスプレイ装置からの入力データを待っている場合、このプログラムは、表示装置ファイルをオープンできません。

関連タスク:

611 ページの『中断処理プログラム』

中断処理プログラムは、*BREAK モードになっているメッセージ待ち行列にメッセージが到着した時に自動的に呼び出されるプログラムです。

メッセージ待ち行列を自動的に中断モードにする:

メッセージ待ち行列を自動的に中断モードにすることで、QSYSOPR メッセージ待ち行列を監視できます。

システムは、始動時に、制御サブシステムの開始の時点で、QSYSOPR メッセージ待ち行列を中断転送モードにします。しかし、システム・オペレーターがサインオフすると、このメッセージ待ち行列は保留転送モードになります。システム・オペレーターが再びサインオンすると、待ち行列 QSYSOPR は QSYSOPR ユーザー・プロファイルに指定されているモードになります。

CL 初期プログラム内の次のプロシージャーを使用すると、QSYSOPR メッセージ待ち行列を中断モードにすることができます。初期プログラムはこれと類似したプロシージャーを使用して、ユーザーのユーザー・プロファイルで指定されているもの以外のメッセージ待ち行列を監視することができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM /* Procedure to place a msg queue in break mode */
CHGMSGQ  QSYSOPR DLVRY(*BREAK) SEV(50)
MONMSG   MSGID(CPF0000) EXEC(SNDPGMMSG MSG('Unable to put QSYSOPR +
         message queue in *BREAK mode') TOPGMQ(*EXT))
ENDPGM
```

このプロシージャーは、QSYSOPR メッセージ待ち行列を重大度レベル 50 の中断転送モードに設定しようとします。これが失敗すると、外部ジョブ・メッセージ待ち行列 (*EXT) にメッセージが送られます。そして、このプロシージャーが入ったプログラムが終了すると、初期メニューが表示されます。重大度レベル 50 を使用すると、ワークステーション・ユーザーの作業を中断する中断メッセージの数を減らせます。障害が生じるのは、他のユーザーが既に QSYSOPR を中断モードにしている場合です。

システム応答リストを使用して、システムが指定された事前定義の照会メッセージに応答するようにすると、ディスプレイ装置ユーザーが応答する必要がなくなります。

関連タスク:

615 ページの『システム応答リストの使用』
システム応答リストを使用することによって、特定の事前定義照会メッセージに対してシステムが自動的に応答するように指定することができます。

ジョブ・メッセージ待ち行列

ジョブ・メッセージ待ち行列は、システム上のジョブごとに作成され、そのジョブのすべてのメッセージ要件を処理します。単一ジョブのジョブ・メッセージ待ち行列は、外部メッセージ待ち行列 (*EXT) および一連の呼び出しメッセージ待ち行列から構成されます。

呼び出しメッセージ待ち行列は、ジョブ内で呼び出される各統合言語環境 (ILE) プロシージャーおよびオリジナル・プログラム・モデル (OPM) プログラムに割り当てられます。さらに、各ジョブごとにジョブ・ログが作成されます。ジョブ・ログとは、ジョブ内に送信されるすべてのメッセージをその発生順に保守する論理待ち行列です。メッセージは *EXT 待ち行列または呼び出しメッセージ待ち行列に送信することができます。ジョブ・ログにはメッセージを送信しないでください。 *EXT または呼び出しメッセージ待ち行列に送信されたメッセージは、システムによってジョブ・ログに論理的に追加されます。

関連概念:

558 ページの『メッセージ待ち行列のタイプ』
システムには、異なるタイプのメッセージ待ち行列 (ワークステーション・メッセージ待ち行列、ユーザー・プロファイル・メッセージ待ち行列、ジョブ・メッセージ待ち行列、システム・オペレーター・メッセ

ージ待ち行列、および活動記録メッセージ待ち行列) があります。

外部メッセージ待ち行列:

外部メッセージ待ち行列 (*EXT) を使用すると、ジョブの外部要求側 (ディスプレイ装置ユーザーなど) と通信できます。

*EXT に送信されたメッセージは、以下の方法で表示されます。

- プログラム・メッセージ表示画面

  対話式ジョブの情報、照会、または通知メッセージがその外部メッセージ待ち行列に送信される場合、そのメッセージはプログラム・メッセージ表示画面に表示され、 プログラムまたはプロシージャーは、ディスプレイ装置ユーザーからの照会または通知メッセージへの応答を待ちます。 ユーザーが応答を入力せずに実行キーまたは F3 (終了) を押すと、デフォルト・メッセージ応答がメッセージの送信側に返されます。デフォルトのメッセージ応答がない場合、*N が送信されます。 照会または通知メッセージがバッチ・ジョブの外部メッセージ待ち行列に送信されると、システムはデフォルト応答をメッセージの送信元に戻します。 デフォルト・メッセージ応答がない場合は、*N が応答になります。 システム応答リストは、照会の表示または *EXT への照会に対するデフォルト応答の送信を一時変更することがあります。

- ディスプレイ装置のメッセージ行

  対話式ジョブの外部メッセージ待ち行列に状況メッセージが送信された場合、そのメッセージはディスプレイ装置のメッセージ行に表示されます。このような状況メッセージは、長時間にわたる操作の進行状況をディスプレイ装置ユーザーに知らせるために使用できます。例えば、複数のメンバーを持つファイルを複写する CPYF コマンドを実行するときなど、システムは状況メッセージを送信します。

  注: アプリケーションが長時間の操作を完了した時点で、ユーザーは別のメッセージを出して、画面上のメッセージ行を消去しなければなりません。この目的で、メッセージ CPI9801 (ブランク・メッセージ) を使用できます。例を以下に示します。

```
PGM
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Status 1')  +
           TOPGMQ(*EXT)  MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Status 2')  +
           TOPGMQ(*EXT)  MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPI9801)  MSGF(QCPFMSG)  TOPGMQ(*EXT)  +
           MSGTYPE(*STATUS)
.
.
.
ENDPGM
```

ジョブの外部メッセージ待ち行列に送信されたメッセージ (状況メッセージを除く) もジョブ・ログに入れられます。

関連概念:

620 ページの『ジョブ・ログ』
各ジョブには、ジョブ・ログが関連付けられています。

571 ページの『状況メッセージ』
CL プロシージャーまたは CL プログラムから、ジョブの外部メッセージ待ち行列 (*EXT) または呼び出しメッセージ待ち行列に状況メッセージを送信するには、**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドを使用します。

呼び出しメッセージ待ち行列:

呼び出しメッセージ待ち行列を使用すると、あるプログラムまたはプロシージャーと別のプログラムまたはプロシージャーの間で、メッセージを送信することができます。

プログラムまたはプロシージャーが呼び出しスタック上にある (まだ戻されていない) 限り、その呼び出しメッセージ待ち行列は活動状態であり、そのプログラムまたはプロシージャーにメッセージを送信できます。プログラムまたはプロシージャーが戻されると、メッセージを送信できなくなります。呼び出しメッセージ待ち行列に送信できるメッセージ・タイプには、情報、要求、完了、診断、状況、エスケープ、および通知があります。

OPM プログラムまたは ILE プロシージャーの呼び出しメッセージ待ち行列が作成されるのは、そのプログラムまたはプロシージャーが呼び出されるときです。呼び出しメッセージ待ち行列は、そのプログラムまたはプロシージャーが実行される呼び出しスタック項目のみに排他的に関連付けられます。呼び出しメッセージ待ち行列は、呼び出しスタック項目を識別することによって間接的に識別されます。呼び出しスタック項目は、その呼び出しスタック項目内で実行されるプログラムまたはプロシージャーの名前によって識別されます。

OPM プログラムの場合、関連する呼び出しスタック項目は、 (最高) 10 文字のプログラム名で識別されます。 ILE プロシージャーの場合、関連する呼び出しスタック項目は、 (最高) 4096 文字のプロシージャー名、(最高) 10 文字のモジュール名、および (最高) 10 文字のプログラム名から構成される、3 部分の名前によって識別されます。モジュール名は、プロシージャーがコンパイルされたモジュールの名前です。ILE プログラム名は、モジュールがバインドされた ILE プログラムの名前です。

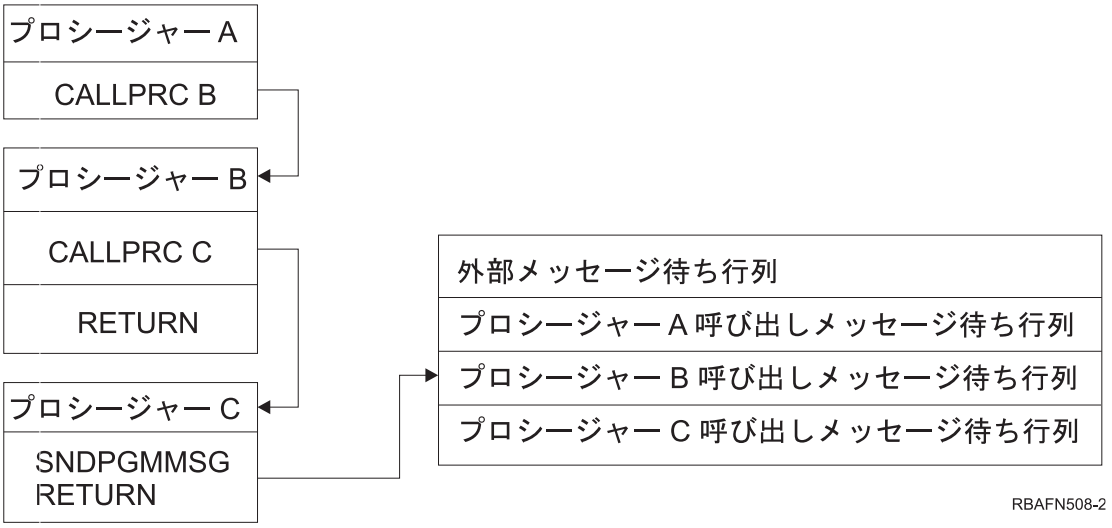
ILE プロシージャーの呼び出しスタック項目を識別するときには、プロシージャー名を指定するだけで十分です。プロシージャー名だけで呼び出しスタック項目が固有に識別されない場合は、モジュール名または ILE プログラム名も指定できます。メッセージの送信時に、プログラムまたはプロシージャーが複数の呼び出しスタック上にある場合、指定された名前は、そのプログラムまたはプロシージャーの最後に呼び出されたオカレンスを識別します。

OPM または ILE プログラムが呼び出しスタック上にあるときにコンパイルされ、置換された場合は、プログラム名を使用して呼び出しスタック項目を参照するときに注意が必要です。置換操作が実行された時点より前からスタック上にある呼び出しスタック項目については、名前参照は置換されたオブジェクト (現在 QRPLOBJ 内に存在している) に解決されます。これらの名前参照は、置換されたオブジェクトが引き続き QRPLOBJ ライブラリー内に存在している限りは有効です。置換操作が実行された時点よりも新しいスタック項目については、名前参照はプログラムの新しいバージョンに関するものになります。使用するバージョンが判別される方法のため、プログラムをライブラリー QRPLOBJ に直接入れてはなりません。このライブラリーは、プログラムの置換されたバージョンのために排他的に使用しなければなりません。直接 QRPLOBJ に入れたプログラムへの名前参照は、失敗します。

プログラム・オブジェクトが除去または名前変更されたが、そのオカレンスが呼び出しスタック上にある場合は、除去されたプログラムへの名前参照、または古い名前を用いた名前参照は失敗します。 ILE プロシージャーの場合、参照にプロシージャーおよびモジュール名のみを使用していれば、プログラムの名前変更は名前参照に影響しません。 ILE プログラム名も使用している場合は、名前参照が失敗します。

プログラムまたはプロシージャーが終了すると、プログラムまたはプロシージャーの呼び出しスタック項目用のメッセージ待ち行列は、使用できなくなります。関連する呼び出しメッセージ待ち行列上にあったメッセージは、その時点で、そのメッセージのメッセージ参照キーを用いて参照することしかできなくなります。

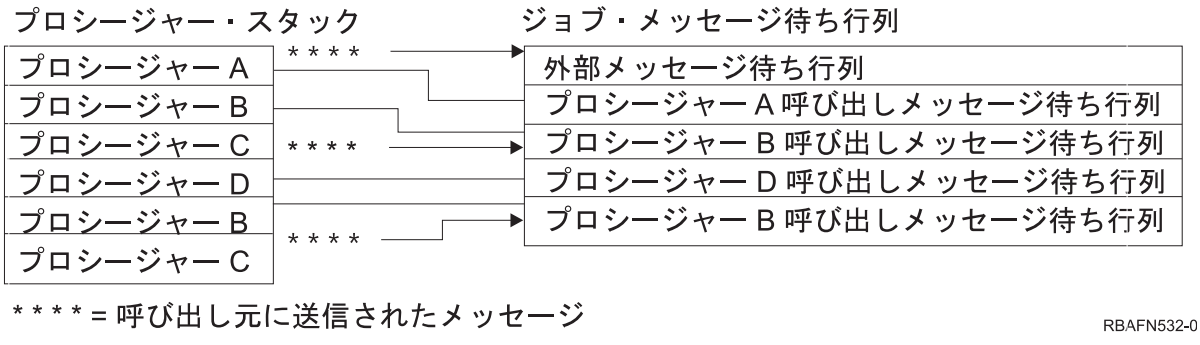
例えば、プロシージャー A がプロシージャー B を呼び出し、プロシージャー B がプロシージャー C を呼び出し、プロシージャー C はプロシージャー B にメッセージを送信して終了するとします。 メッセージはプロシージャー B で使用可能です。しかし、プロシージャー B が終了すると、その呼び出しメッセージ待ち行列は使用できなくなるため、メッセージがジョブ・ログ内に示されている場合でも、プロシージャー A からプロシージャー B へはアクセスできません。プロシージャー A は、プロシージャー B に送信されるメッセージに対してメッセージ参照キーを持っていなければ、そのメッセージにアクセスできません。



プロシージャー A が特定のメッセージを削除する必要がある場合は、次のことを行えます。

- プロシージャー C に、特定のメッセージをプロシージャー A へ送信させる。
- プロシージャー B にメッセージをプロシージャー A へ移動させる、または再送信させる。

以下の図は、プロシージャー呼び出し、ジョブ・メッセージ待ち行列、および呼び出しスタック項目待ち行列の関係を示すものです。プロシージャー A はメッセージをそれ自体および *EXT に送信してからプロシージャー B を呼び出します。プロシージャー B はプロシージャー C を呼び出します。プロシージャー C は、メッセージをその呼び出し元 (プロシージャー B) に送信してからプロシージャー D を呼び出します。プロシージャー D は、メッセージをそれ自体に送信してから、プロシージャー B を呼び出します。結合線 (-----) は、どのプロシージャー呼び出しがどのメッセージ待ち行列と関連するかを示します。



前の図では、プロシージャー B には 2 つの呼び出しスタック項目待ち行列があります (プロシージャーの呼び出しごとに 1 つずつ)。 プロシージャー C にはメッセージが送られていないため、プロシージャー C のメッセージ待ち行列はありません。プロシージャー C がプロシージャー B にメッセージを送信すると、メッセージは、プロシージャー B の最後の呼び出しの呼び出しスタック項目待ち行列に入れられます。

注: コマンド入力画面を使用している場合は、F10 (詳細なメッセージの組み込み) を押すことによって、ジョブ・メッセージ待ち行列に送信されたすべてのメッセージを表示できます。 メッセージが表示された後、いずれかのロール・キーを用いて、画面表示を上下に移動することができます。

さらに、**ジョブ・ログ表示 (DSPJOBLOG)** コマンドを使用すると、ジョブのメッセージを表示することができます。

関連タスク:

608 ページの『終了したプログラムまたはプロシージャーからメッセージを受信する』
アクティブでない呼び出しメッセージ待ち行列に送信されたジョブ・ログからメッセージを受信する必要がある場合があります。

575 ページの『呼び出しスタック項目の識別』
CL プロシージャーが、オリジナル・プログラム・モデル (OPM) プログラムまたは他の統合言語環境 (ILE) プロシージャーにメッセージを送る予定がある場合は、そのメッセージの送信先となる呼び出しスタック項目を識別する必要があります。

システム・ユーザーへのメッセージの送信に使用されるコマンド

システム・ユーザーにメッセージを送信する際に使用できるコマンドには、次のようなものがあります。

- **メッセージ送信 (SNDMSG)**
- **中断メッセージ送信 (SNDBRKMSG)**
- **プログラム・メッセージ送信 (SNDPGMMSG)**
- **ユーザー・メッセージ送信 (SNDUSRMSG)**

プログラム・メッセージ送信 (SNDPGMMSG) および **ユーザー・メッセージ送信 (SNDUSRMSG)** は、バッチまたは対話式のオリジナル・プログラム・モデル (OPM) プログラム、または統合言語環境 ILE プロシージャーの中でだけ使用できます。これらのコマンドはコマンド入力行に入力できません。 **メッセージ送信 (SNDMSG)** コマンドはシステム・オペレーター・メッセージ待ち行列 (QSYSOPR)、ワークステーション・メッセージ待ち行列、またはユーザー・メッセージ待ち行列に、情報メッセージまたは照会メッセージを送るためのものです。情報メッセージは、同時に複数のメッセージ待ち行列に送ることもできます。しかし、照会メッセージは一度に 1 つのメッセージ待ち行列にしか送れません。メッセージは、そのメッセージ待ち行列で指定されている転送タイプに応じて転送されます。メッセージ待ち行列が中断モードでない限り、メッセージによってユーザーの作業が中断されることはありません。

以下の **メッセージ送信 (SNDMSG)** コマンドは、ディスプレイ装置ユーザーによりシステム・オペレーターに送られるものです。

```
SNDMSG   MSG('Mount tape on device TAP1') TOUSR(*SYSOPR)
```

中断メッセージ送信 (SNDBRKMSG) コマンドは、ワークステーション、プログラム、またはジョブから 1 つ以上のディスプレイ装置に即時メッセージを送信します。このメッセージは、受け取り側のメッセージ待ち行列がどのような転送モードに設定されていても、中断モードで転送されます。このコマンドは、ワークステーションのメッセージ待ち行列にメッセージを送る場合に限り使用することができます。ディスプレイ装置ユーザーの注意を直ちに喚起する必要があるようなメッセージを送る場合には、**中断メッセージ送信**

(SNDBRKMSG) コマンドを使用する必要があります。 **ジョブ変更 (CHGJOB)** コマンドの BRKMSG パラメーターの指定によっては、各ジョブに制御権があるため、必ずしもメッセージが中断を起こすかどうかはわかりません。

照会メッセージを送る場合には、ユーザーが使用しているディスプレイ装置の待ち行列とは異なるメッセージ待ち行列に応答を返すように指定することもできます。

以下の**中断メッセージ送信 (SNDBRKMSG)** コマンドはシステム・オペレーターにより、ディスプレイ装置のすべてのメッセージ待ち行列に送られます。

```
SNDBRKMSG MSG('System going down in 15 minutes')
          TOMSGQ(*ALLWS)
```

このメッセージ送信の不便な点は、メッセージが送られる時点で活動状態にあるユーザーだけでなく、すべてのユーザーにメッセージが送られる点です。

関連資料:

『CL プログラムからのメッセージの送信に使用されるコマンド』

CL プロシージャーまたはプログラムからのメッセージの送信には、 **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドまたは**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用します。

CL プログラムからのメッセージの送信に使用されるコマンド

CL プロシージャーまたはプログラムからのメッセージの送信には、 **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドまたは**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用します。

SNDPGMMSG コマンドを使用すれば、以下のタイプのメッセージを送ることができます。

- 情報
- 照会
- 完了
- 診断
- 要求
- エスケープ
- 状況
- 通知

CL プロシージャーまたは CL プログラムからのメッセージは、以下のタイプの待ち行列に送ることができます。

- ジョブの要求元の外部メッセージ待ち行列
- ジョブにより呼び出されるプログラムまたはプロシージャーの、呼び出しメッセージ待ち行列
- システム・オペレーター・メッセージ待ち行列
- ワークステーション・メッセージ待ち行列
- ユーザー・メッセージ待ち行列

プロシージャーまたはプログラムからメッセージを送る場合、SNDPGMMSG コマンドに以下の事項を指定できます。

- メッセージ識別コードまたは即時メッセージ。 メッセージ識別コードは、事前定義メッセージのメッセージ記述の名前です。

- メッセージ・ファイル。これは、事前定義メッセージが送られる時点でそのメッセージ記述が入っているメッセージ・ファイルの名前です。
- メッセージ・データ・フィールド。事前定義メッセージを送る場合には、メッセージの置換変数の値をこれらのフィールドに入れます。各フィールドの形式はメッセージ記述で記述されていなければなりません。即時メッセージを送る場合には、メッセージ・データ・フィールドはありません。
- メッセージを受け取るメッセージ待ち行列またはユーザー。
- メッセージ・タイプ。 以下の表は、どのタイプのメッセージをどのタイプのメッセージ待ち行列に送ることができるかを示しています (V = 有効)。

表 34. メッセージ待ち行列タイプについて有効なメッセージ・タイプ

メッセージ・タイプ	メッセージ待ち行列タイプ				
	外部	呼び出し	**QSYSOPR**	ワークステーション	ユーザー
情報	V	V	V	V	V
照会	V		V	V	V
完了	V	V	V	V	V
診断	V	V	V	V	V
要求	V	V			
エスケープ		V			
状況	V	V			
通知	V	V			

- コード化文字セット ID (CCSID)。コード化文字セット ID (CCSID) を指定します。メッセージやメッセージ・データはこの形式で送信されます。
- 応答メッセージ待ち行列。照会メッセージに対する応答を受け取るメッセージ待ち行列の名前。デフォルトでは、照会メッセージを送信したプロシージャーまたはプログラムの呼び出しメッセージ待ち行列へ応答が送られます。
- キー変数名。メッセージのメッセージ参照キーを受け入れる CL 変数の名前。

552 ページの『例: メッセージの記述』で作成したメッセージを送るには以下のコマンドを使用します。

```
SNDPGMMSG   MSGID(USR4310) MSGF(QGPL/USRMSG) +
            MSGDTA(&CUSNO) TOPGMQ(*EXT) +
            MSGTYPE(*INFO)
```

このメッセージの置換変数は顧客番号です。顧客番号は変化するため、メッセージに固定した顧客番号を指定することはできません。その代わりに、CL プロシージャーまたはプログラム内で、顧客番号 (&CUSNO) を表す CL 変数を宣言します。さらに、この変数をメッセージ・データ・フィールドとして指定します。このメッセージが送られるさいには、以下のように変数の現在の値がメッセージに入れられます。

```
Customer number 35500 not found
```

さらに、どのディスプレイ装置でプロシージャーやプログラムが使用されているかが必ずしもわかるわけではなく、したがってメッセージをどのディスプレイ装置メッセージ待ち行列に送るか (TOMSGQ パラメーター) を正しく指定できません。このような場合には、TOPGMQ パラメーターに外部メッセージ待ち行列 *EXT を指定します。

関連概念:

579 ページの『名前による基本項目の識別』
基本呼び出しスタック項目を識別するには、その項目で実行するオリジナル・プログラム・モデル (OPM)

プログラムまたは統合言語環境 (ILE) プロシージャーの名前を指定します。

関連タスク:

592 ページの『要求メッセージの受信』
要求メッセージの受信は、CL プロシージャーまたは CL プログラムが CL コマンドを処理するための 1 つの方法です。

関連資料:

567 ページの『システム・ユーザーへのメッセージの送信に使用されるコマンド』
システム・ユーザーにメッセージを送信する際に使用できるコマンドには、次のようなものがあります。

関連情報:

メッセージ待ち行列のタイプ

照会メッセージおよび通知メッセージ

ディスプレイ装置ユーザー、システム・オペレーター、またはユーザー定義のメッセージ待ち行列に、照会メッセージまたは通知メッセージを送信するには、**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用します。

ユーザー・メッセージ送信 (SNDUSRMSG) コマンドを使用してユーザーに照会メッセージを送った場合には、プロシージャーまたはプログラムはそのユーザーからの応答を待ちます。このメッセージは即時メッセージまたは事前定義メッセージのどちらでも構いません。対話式ジョブの場合には、デフォルト値によってディスプレイ装置のオペレーターにメッセージが送られます。バッチ・ジョブの場合には、デフォルト値によってシステム・オペレーターにメッセージが送られます。 **ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用してプロシージャーまたはプログラムからメッセージを送る場合には、**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドで以下を指定することができます。

- メッセージ識別コードまたは即時メッセージ。 メッセージ識別コードは、事前定義メッセージのメッセージ記述の名前です。
- メッセージ・ファイル。これは、事前定義メッセージが送られる時点でそのメッセージ記述が入っているメッセージ・ファイルの名前です。
- メッセージ・データ・フィールド。事前定義メッセージを送る場合には、メッセージの置換変数の値をこれらのフィールドに入れます。各フィールドの形式はメッセージ記述で記述されていなければなりません。即時メッセージを送る場合には、メッセージ・データ・フィールドはありません。
- 照会メッセージに対する有効な応答。
- 照会メッセージに対するデフォルトの応答の値。
- メッセージ・タイプ。
- メッセージの送り先のメッセージ待ち行列。
- メッセージ応答。照会メッセージに対する応答として受け取った値が入る CL 変数 (必要な場合)。
- 変換テーブル。応答値を変換するために使用する変換テーブル (必要な場合)。このテーブルは通常、小文字から大文字への変換のために使用されます。
- コード化文字セット ID (CCSID)。コード化文字セット ID (CCSID) を指定します。メッセージやメッセージ・データはこの形式で送信されます。

完了および診断メッセージ

診断および完了メッセージを送るには、**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用します。

これらのタイプのメッセージは、CL プロシージャーまたは CL プログラムからどのようなメッセージ待ち行列にも送ることができます。診断メッセージは、CL プロシージャーまたは CL プログラムが検出し

たエラーを、呼び出しプロシージャーまたはプログラムに知らせます。完了メッセージは、CL プロシージャーまたは CL プログラムによって行われた処理の結果を知らせます。

通常、エスケープ・メッセージが呼び出し側プログラムまたはプロシージャーのメッセージ待ち行列に送られ、どのような問題が生じたかや、診断メッセージが同時に送られたことを知らせます。ただし、完了メッセージの場合には通常、要求された機能が既に実行されているため、エスケープ・メッセージは送られません。

完了メッセージの送信の一例として、システム・オペレーターが特定のオブジェクトを保管するために、コマンド入力画面を使用して CL プログラム SAVPAY を呼び出したと仮定します。この CL プログラムには以下のプロシージャーのみが含まれています。このプロシージャーはオブジェクトを保管してから完了メッセージを出します。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
PGM
SAVOBJ OBJ(PAY1 PAY2) LIB(PAYROLL) DEV(TAP01)
SNDPGMMSG MSG('Payroll objects have been saved') MSGTYPE(*COMP)
ENDPGM
```

オブジェクト保管 (SAVOBJ) コマンドが正常に実行されなかった場合には、CL プロシージャーに機能チェックが生じるので、システム・オペレーターはその障害の原因を示すエスケープ・メッセージを見つけるために詳細なメッセージを表示しなければなりません。 SAVOBJ コマンドが正常に完了した場合には、コマンド入力画面を表示したプログラムの呼び出しメッセージ待ち行列に完了メッセージが送られます。

完了メッセージの利点の 1 つは、IBM 提供のコマンドとの一貫性です。 IBM コマンドの多くは処理が正しく完了したことを示す完了メッセージを出します。 ジョブ・ログに送られたメッセージのタイプを調べることは、問題判別に役立ちます。

状況メッセージ

CL プロシージャーまたは CL プログラムから、ジョブの外部メッセージ待ち行列 (*EXT) または呼び出しメッセージ待ち行列に状況メッセージを送信するには、**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドを使用します。

状況メッセージが呼び出しメッセージ待ち行列に送られた場合、受信側プログラムまたはプロシージャーは状況メッセージの到着を監視し、そのメッセージに示されている状況に対処することができます。受信側のプログラムまたはプロシージャーがメッセージを監視しないときは、制御権は送信側に戻され、処理が再開されます。

状況メッセージは、ジョブのジョブ・ログには含まれません。

関連概念:

564 ページの『外部メッセージ待ち行列』
外部メッセージ待ち行列 (*EXT) を使用すると、ジョブの外部要求側 (ディスプレイ装置ユーザーなど) と通信できます。

エスケープ・メッセージおよび通知メッセージ

エスケープ・メッセージおよび通知メッセージはどちらも、CL プログラムまたはプロシージャーからメッセージ待ち行列に送信することができます。

エスケープ・メッセージは、**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドを使用して、CL プログラムまたはプロシージャーから呼び出し側プログラムまたはプロシージャーの呼び出しメッセージ待ち行列

に送ることができます。エスケープ・メッセージは呼び出し側に、プロシージャーまたはプログラムが異常終了したこととその理由を知らせます。呼び出し側はエスケープ・メッセージの到着を監視して、そのメッセージに示されている状況に対処することができます。呼び出し側がその状況に対処する際には、エスケープ・メッセージを送ったプログラムに制御権が返されません。

呼び出し側が同一のプログラム中にある別のプロシージャーである場合は、そのプログラム自体は終了しません。エスケープ・メッセージが送信されたプロシージャーの実行を継続することができます。エスケープ・メッセージがプログラムの呼び出し側自体に送られた場合は、そのプログラムの活動状態のプロシージャーはすべて即時に終了します。結果として、そのプログラムの実行を継続することはできません。呼び出し側がエスケープ・メッセージを監視していない場合は、デフォルトのシステム処置がとられます。

通知メッセージは CL プログラムまたはプロシージャーから、呼び出し側プログラムまたはプロシージャーのメッセージ待ち行列、あるいは外部メッセージ待ち行列に送ることができます。通知メッセージは、処理を続行することができる状況を呼び出し側に知らせるためのものです。 呼び出し元のプログラムまたはプロシージャーは、通知メッセージの到着を監視し、そのメッセージに示されている状況に対処することができます。 呼び出し元が統合言語環境のプロシージャーである場合、そのプロシージャーは以下の機能を実行できます。

- 条件を扱うことができる。
- 呼び出し側に応答を返すことができる。
- 処理を続行するための送信プロシージャーを許可することができる。

呼び出し側がオリジナル・プログラム・モデル (OPM) プログラムでありメッセージ・モニターを行っていない場合は、送信側はデフォルト応答を受け取ります。呼び出し側が ILE のプロシージャーである場合は、メッセージは制御境界にパーコレートを行います。監視が検出されない場合、システムは送信側にデフォルト応答を返します。このプログラムの処理はその後で再開されます。

即時メッセージは、エスケープ・メッセージまたは通知メッセージとして送ることはできません。システムには、アプリケーションの即時エスケープ・メッセージおよび即時通知メッセージとして使用できるメッセージ CPF9898 が定義されています。以下にその例を示します。

```
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGDTA('Error condition') +
          MSGTYPE(*ESCAPE)
```

関連タスク:

599 ページの『CL プログラムまたはプロシージャー内のメッセージの監視』
*ESCAPE メッセージ、*STATUS メッセージ、および *NOTIFY メッセージは、モニター可能なメッセージであり、プログラムまたはプロシージャーで使用される各 CL コマンドによって発行されます。

例: メッセージ送信

以下の例は、さまざまなメッセージの送信方法を示しています。

例 1: 完了メッセージの送信

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

次の CL プロシージャーを使用すると、ディスプレイ装置ユーザーは、**ジョブ投入 (SBMJOB)** コマンドを入力する代わりに、このプロシージャーが入っている CL プログラムを呼び出すことによってジョブを投入することができます。ジョブの投入が終わると、このプロシージャーは完了メッセージを送ります。

```
PGM
SBMJOB JOB(WKLYPAY) JOBD(USERA) RQSDTA('CALL WKLY PARM(PAY1)')
SNDPGMMSG MSG('WKLYPAY job submitted') MSGTYPE(*COMP)
ENDPGM
```

例 2: 変数テキスト付き完了メッセージの送信

次の CL プロシージャーは、このプロシージャー内から呼び出されるプログラムから受け取ったパラメーターに基づいてメッセージを送信します。メッセージは CL プロシージャーによって完了メッセージとして送られます。 (RCDCNT フィールドは、PGMA として定義されています。)

```
PGM
DCL &RCDCNT TYPE(*CHAR) LEN(3)
CALL PGMA PARM(&RCDCNT)
SNDPGMMSG MSG('PGMA completed' *BCAT &RCDCNT *BCAT +
          'records processed') MSGTYPE(*COMP)
ENDPGM
```

例 3: 照会メッセージの送信および応答の受信

次のプロシージャーは、システム・オペレーターに特殊な用紙の装てんを要求するメッセージを送信します。 **メッセージ受信 (RCVMSG)** コマンドは、そのメッセージに対する応答が返されるまで待機します。システム・オペレーターは、照会メッセージに対する応答として少なくとも 1 文字を入力する必要がありますが、プロシージャーはその応答値を使用する残りのコードを含んでいません。

```
PGM
DCL SNDRCOPY TYPE(*CHAR) LEN(4)
SNDPGMMSG   MSG('Load special form') TOUSR(*SYSOPR) +
            KEYVAR(SNDRCOPY) MSGTYPE(*INQ)
RCVMSG MSGTYPE(*RPY) MSGKEY(SNDRCOPY) WAIT(120)
.
.
.
ENDPGM
```

注:

1. 応答が返されるまでプロシージャーが待機するようにするためには、**メッセージ受信 (RCVMSG)** コマンドに WAIT パラメーターを指定しなければなりません。 WAIT パラメーターの指定がない場合、プロシージャーは応答を受け取らずに **メッセージ受信 (RCVMSG)** コマンドの次のコマンドに移って処理を続けます。
2. **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドの変数 SNDRCOPY は、照会メッセージに関連付けられた送信側コピー・メッセージのメッセージ・キーをプロシージャーに戻します。これは、応答を受信するために必要です。このキーがメッセージ・タイプ *RPY で **メッセージ受信 (RCVMSG)** コマンドで指定されている場合、関連する照会メッセージに入力された応答が戻されます。SNDRCOPY メッセージ・キーは、**メッセージ受信 (RCVMSG)** コマンドの MSGKEY パラメーターで指定されます。

例 4: SNDUSRMSG を使用した照会メッセージの送信および応答の受信

次のプロシージャーは、システム・オペレーター (バッチ・モードで実行された場合) またはディスプレイ装置オペレーター (ディスプレイ装置から実行された場合) にメッセージを送信します。このプロシージャーは、Y または N を大文字でも小文字でも受け入れます。 (プログラムのロジックを簡単にするために、小文字の値は変換テーブル (デフォルトでは **ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドの TRNTBL パラメーター) によって大文字に変換されます。)入力された値がこの 4 つの値 (Y、N、y、n) のどれでもなかった場合には、応答が無効であることを示す **ユーザー・メッセージ送信 (SNDUSRMSG)** からのメッセージがオペレーターに対して出されます。

```
PGM
DCL &REPLY *CHAR LEN(1)
.
.
SNDUSRMSG MSG('Update YTD Information Y or N') VALUES(Y N) +
     MSGRPY(&REPLY)
IF (&REPLY *EQ Y)
 DO
 .
 .
 .
 ENDDO
ELSE
 DO
 .
 .
 ENDDO
 .
 .
 .
ENDPGM
```

例 5: エスケープ・メッセージの送信

次のプロシージャーは、メッセージ CPF9898 を使用してエスケープ・メッセージを送信します。メッセージのテキストは、'Procedure detected failure' です。即時メッセージはエスケープ・メッセージとして使用できないため、メッセージ CPF9898 を、メッセージ・データとしてのメッセージと一緒に使用します。

```
   PGM
   .
   .
   .
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
     MSGDTA('Procedure detected failure')
   .
   .
   ENDPGM
```

例 6: 複数ユーザーに対する情報メッセージの送信

次のプロシージャーを使用すると、システム・オペレーターはいくつかのディスプレイ装置にメッセージを送信することができます。システム・オペレーターがこのプロシージャーを含むプログラムを呼び出すと、このプロシージャーによってプロンプトが表示されます。システム・オペレーターは、このプロンプトを使用して、送信されるメッセージのタイプとメッセージのテキストを入力することができます。このプロシージャーは、メッセージの日付、時刻、およびテキストを連結します。

```
PGM
DCLF WSMSGD
DCL &MSG TYPE(*CHAR) LEN(150)
DCL &HOUR TYPE(*CHAR) LEN(2)
DCL &MINUTE TYPE(*CHAR) LEN(2)
DCL &MONTH TYPE(*CHAR) LEN(2)
DCL &DAY TYPE(*CHAR) LEN(2)
DCL &WORKHR TYPE(*DEC) LEN(2 0)
SNDRCVF RCDFMT(PROMPT)
IF &IN91 RETURN /* Request was ended  */
RTVSYSVAL QMONTH RTNVAR(&MONTH)
RTVSYSVAL QDAY RTNVAR(&DAY)
RTVSYSVAL QHOUR RTNVAR(&HOUR)
IF (&HOUR *GT '12') DO /* Change from military time */
CHGVAR &WORKHR &HOUR
CHGVAR &WORKHR (&WORKHR - 12)
CHGVAR &HOUR &WORKHR
```

```
            ENDDO
            RTVSYSVAL QMINUTE RTNVAR(&MINUTE)
            CHGVAR    &MSG ('From Sys Opr ' *CAT &MONTH *CAT '/' +
                      *CAT &DAY +
                      *BCAT &HOUR *CAT ':' *CAT &MINUTE +
                      *BCAT &TEXT)
            IF (&TYPE *EQ 'B') GOTO BREAK
NORMAL:     SNDPGMMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
            GOTO ENDMSG
BREAK:      SNDBRKMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
ENDMSG:     SNDPGMMSG MSG('Message sent to display stations') +
                      MSGTYPE(*COMP)
            ENDPGM
```

このプログラムで使用する表示装置ファイル (WSMSGD) の DDS は以下のとおりです。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                      DSPSIZ(24 80)
     A          R PROMPT                    TEXT('Prompt')
     A                                      BLINK
     A                                      CA03(91 'Return')
     A                                   1  2'Send Messages To Workstations'
     A                                       DSPATR(HI)
     A                                   3  2'TYPE'
     A            TYPE           1   1    +2VALUES('N' 'B')
     A                                     CHECK(ME)
     A                                     DSPATR(MDT)
     A                                    +3'(N = No breaks  B = Break)'
     A                                   5  2'Text'
     A            TEXT         100   1    +2LOWER
     A
     A
```

システム・オペレーターがプロンプトに対して以下のように入力したとします。

```
B

Please sign off by 3:30 today
```

すると、次のような中断メッセージが送られます。

```
From Sys Opr 10/30 02:00  Please sign off by 3:30 today
```

関連タスク:

614 ページの『送信側コピー・メッセージを使用して応答を取得する』
照会メッセージが送信されると、応答が要求されます。照会メッセージの送信者が応答を取得できるようにするため、送信側コピー・メッセージが発行され、内部的に照会メッセージと関連付けられます。

呼び出しスタック項目の識別

CL プロシージャーが、オリジナル・プログラム・モデル (OPM) プログラムまたは他の統合言語環境 (ILE) プロシージャーにメッセージを送る予定がある場合は、そのメッセージの送信先となる呼び出しスタック項目を識別する必要があります。

メッセージは、識別された呼び出しスタック項目の呼び出しメッセージ待ち行列に送られます。

プログラム・メッセージ送信 (SNDPGMMSG) コマンドの TOPGMQ パラメーターを使用して、メッセージの送信先となる呼び出しスタック項目を識別します。呼び出しスタック項目の識別コードは以下の 2 つの部分で構成されています。

- 基本項目の仕様

この仕様は、呼び出しスタック内のプログラムまたはプロシージャーを識別します (これは TOPGMQ のエレメント 2 です)。

- 基本項目のオフセット仕様

  オフセット仕様 (TOPGMQ の要素 1) は、基本項目 (*SAME) と基本項目の呼び出し側 (*PRV) のどちらかにメッセージを送ったかどうか識別します。

TOPGMQ(*PRV *) 仕様は、基本項目が**プログラム・メッセージ送信 (SNDPGMMSG)** コマンドを使用するプロシージャーが実行される項目であると識別します。オフセットは、この基本項目の直前の項目として指定されます。この仕様は、前述のコマンドを使用するプロシージャーの呼び出し側を識別します。

基本項目 TOPGMQ の要素 2 を識別する方法を理解するためには、ILE プログラム実行時の呼び出しスタックについても理解する必要があります。このことについては 2 つのプログラムを使って説明します。プログラム CLPGM1 は OPM CL プログラムで、プログラム CLPGM2 は ILE プログラムです。プログラム CLPGM2 は ILE なので、複数のプロシージャーで構成することができます。CLPROC1、CLPROC2、CLPROC3、および CLPROC4 などがこれに該当します。実行時には以下の呼び出しが行われます。

- 最初に CLPGM1 が呼び出される。
- CLPGM1 が CLPGM2 を呼び出す。
- CLPGM2 が CLPROC1 を呼び出す。
- CLPROC1 が CLPROC2 を呼び出す。
- CLPROC2 が CLPROC3 または CLPROC4 を呼び出す。

次の図は、CLPROC2 が CLPROC4 を呼び出すときの呼び出しスタックの構造に関する考慮事項を示しています。

- 呼び出しスタック項目と OPM プログラムは 1 対 1 で対応しています。OPM プログラムの呼び出しごとに 1 つの新規項目が呼び出しスタックに追加されます。
- ILE プログラムは 1 つの単位としてはスタックに表示されません。その代わり、ILE プログラムが呼び出されると、そのプログラム中にプロシージャーが呼び出されるたびに 1 つの項目がスタックに追加されます。したがって、ILE プログラムではなく ILE プロシージャーにメッセージを送ることになります。

注: ILE プログラムが呼び出された時点で最初に実行するプロシージャーは、プログラムのプログラム入り口プロシージャー (PEP) です。 CL において、このプロシージャー (_CL_PEP) はシステムにより生成され、ユーザーが指定した最初のプロシージャーを呼び出します。この例では、PEP の項目は OPM プログラム CLPGM1 の項目とプロシージャー CLPROC1 の項目の間にあります。

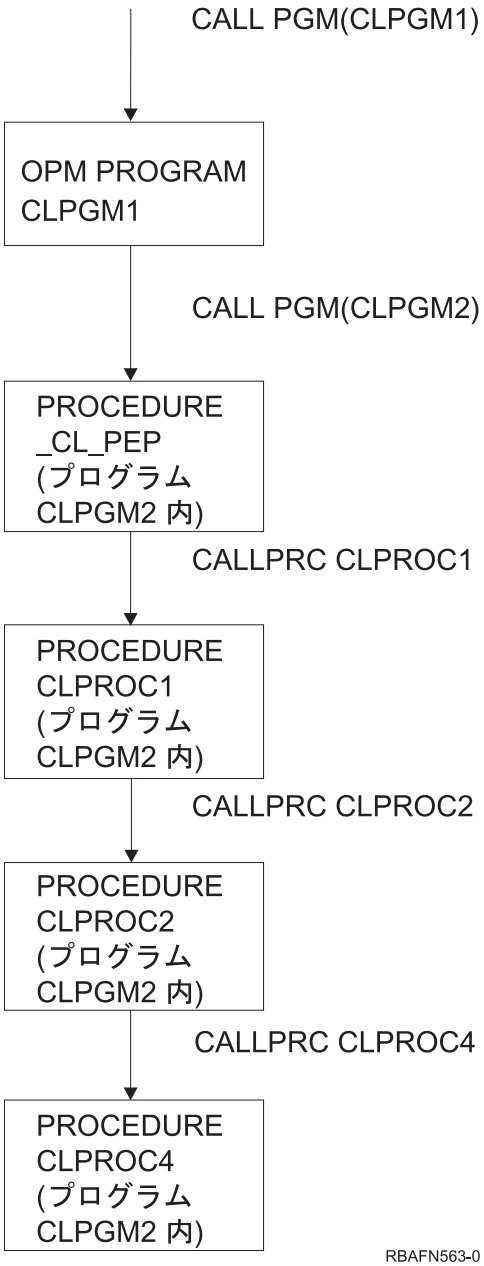


図 *16. OPM* プログラムを持つランタイム呼び出しスタックおよび複数プロシージャーを持つ *ILE* プログラムの例

関連概念:

565 ページの『呼び出しメッセージ待ち行列』
呼び出しメッセージ待ち行列を使用すると、あるプログラムまたはプロシージャーと別のプログラムまたはプロシージャーの間で、メッセージを送信することができます。

プログラム・メッセージ送信コマンドの基本項目としての使用**:**

プログラム・メッセージ送信 (SNDPGMMSG) コマンドを基本呼び出しスタック入力として実行するプログラムまたはプロシージャーを使用するには、TOPGMQ パラメーターを使用することができます。

TOPGMQ パラメーターで TOPGMQ(*SAME *) または TOPGMQ(*PRV *) を指定すると、 **SNDPGMMSG** コマンドを使用するプロシージャーの項目は基本項目として使用されます。TOPGMQ(*SAME *) を指定すると、プロシージャーはメッセージをそのプロシージャー自身に送ります。 TOPGMQ(*PRV *) を指定すると、プログラムまたはプロシージャーはメッセージを呼び出し側に送ります。

注**:** TOPGMQ(*PRV *) を指定してプロシージャーがメッセージを呼び出し側に送るようにする場合には、以下の情報を知っていなければなりません。

- CLPROC4 と CLPROC2 がメッセージを呼び出し側に戻した場合には、それらのプロシージャーを含むプログラムからそのメッセージがなくなることはありません。このメッセージは、同一のプログラム中のプロシージャーの間に送られます。 CLPGM2 の呼び出し側である CLPGM1 にメッセージを送ることが目的である場合には、TOPGMQ(*PRV *) を指定するのは適切ではありません。
- CLPROC1 がメッセージを呼び出し側に戻した場合は、プログラム入り口プロシージャーはスキップされます。呼び出し側が PEP である場合でもメッセージは CLPGM1 に送られます。 TOPGMQ(*PRV *) を指定すると、PEP 項目は可視状態ではなく、送信操作に組み込まれません。他の何らかの方法で TOPGMQ を指定すると、PEP は、送信側に可視 になります。

以下の図には、CLPROC1、CLPROC2、および CLPROC4 のそれぞれが各プロシージャーの呼び出し側にメッセージを送信した場合の結果が図示されています。

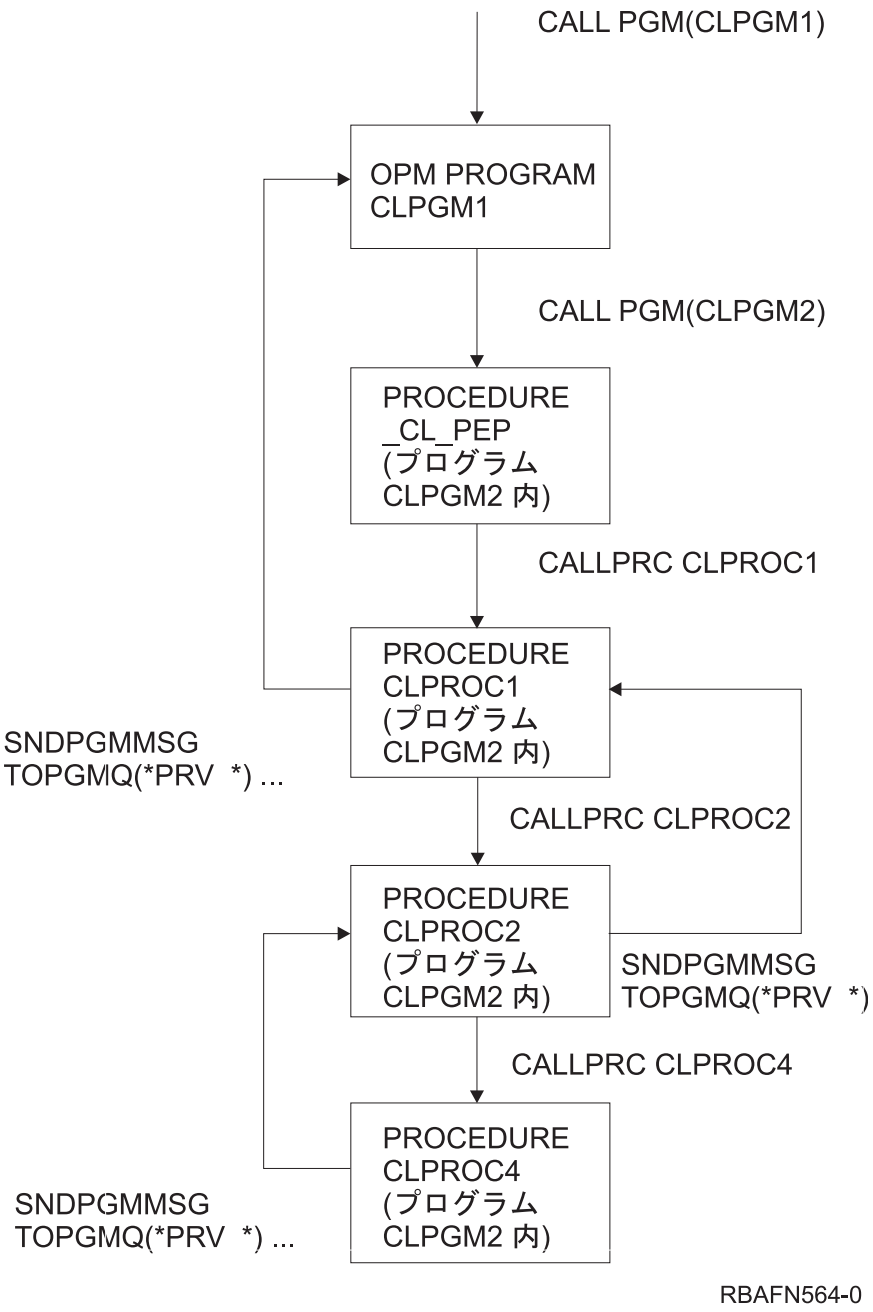


図 *17. TOPGMQ(*PRV *) の例*

注**:** PEP は、(*PRV *) には可視ではないので、 CLPROC1 からのメッセージは CLPGM1 に送信されます。

名前による基本項目の識別**:**

基本呼び出しスタック項目を識別するには、その項目で実行するオリジナル・プログラム・モデル (OPM) プログラムまたは統合言語環境 (ILE) プロシージャーの名前を指定します。

単純名 (1 つの部分) または、複合名 (2 または 3 つの部分) のどちらかを付けることができます。次に単純名および複合名の説明を示します。

- 単純名

1 つの OPM プログラムか ILE プロシージャーを識別する場合は、単純名を使用します。 10 文字以下の長さの単純名を付けた場合は、システムはその名前が OPM プログラムか ILE プロシージャーのどちらかの名前であると判断します。基本項目は、その名前で呼び出された最新の OPM プログラムまたは ILE プロシージャーであると識別されます。

名前の長さが 10 文字を超えている場合は、システムはその名前が ILE プロシージャーの名前であると判断します (OPM プログラム名は 10 文字を超えることはできません)。基本項目は、その名前で呼び出された最新のプロシージャーであると識別されます。 OPM プログラムを実行している項目は考慮されません。

単純名を使用してメッセージを送る例については、 581 ページの図 18 を参照してください。この例では CLPROC4 はメッセージを CLPROC2 に送り、CLPROC2 はメッセージを CLPGM1 に送っています。

- 複合名

  複合名は 2 つまたは 3 つの部分で構成されており、それらの部分は以下のとおりです。

  – モジュール名

    モジュール名は、プロシージャーがコンパイルされたモジュールの名前です。

  – プログラム名

    プログラム名とは、プロシージャーがバインドされたプログラム名の名前です。

  – プロシージャー名

  メッセージの送信先のプロシージャーを個別に識別したい場合は、複合名を以下のいずれかの組み合わせで使用することができます。

  – プロシージャー名、モジュール名、プログラム名
  – プロシージャー名とモジュール名
  – プロシージャー名とプログラム名

    モジュール名を *NONE として指定する必要があります。

  複合名を使用する場合、識別される基本項目で OPM プログラムを実行することはできません。

複合名を使用してメッセージを送る例については、 582 ページの図 19 を参照してください。この例では、2 つの部分 (プロシージャー名、プログラム名) で構成されている名前を使用して、CLPROC4 が CLPROC1 にメッセージを送っています。

OPM プログラムまたは ILE プロシージャーの完全な名前を使用するのではなく、名前の一部を使用することができます。オンライン・コマンド・ヘルプ・テキストには、呼び出しスタック名の一部を指定する方法が記載されています。例えば、プログラム・メッセージ送信 (SNDPGMMSG) コマンドを参照してください。

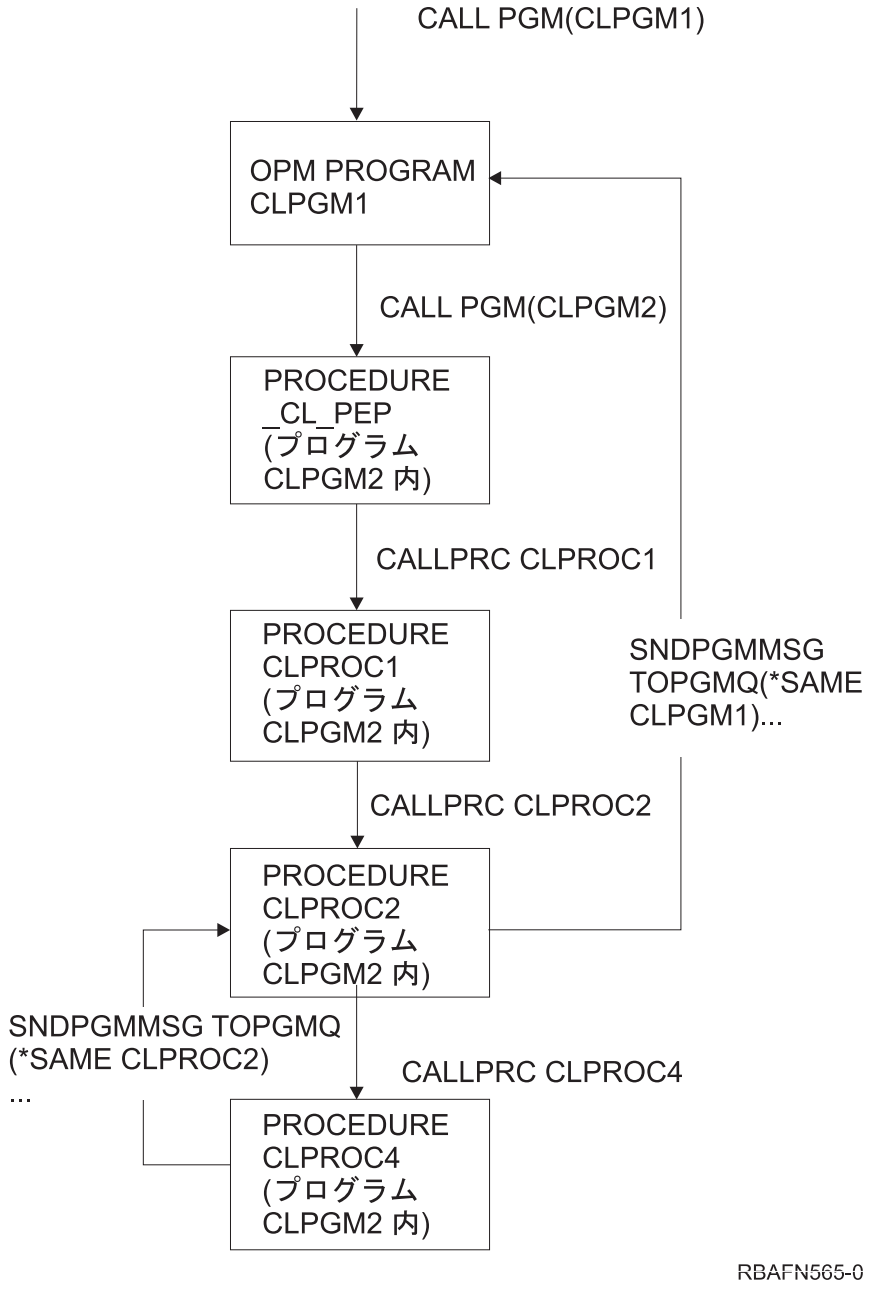


図 *18.* 基本項目としての単純名の使用例

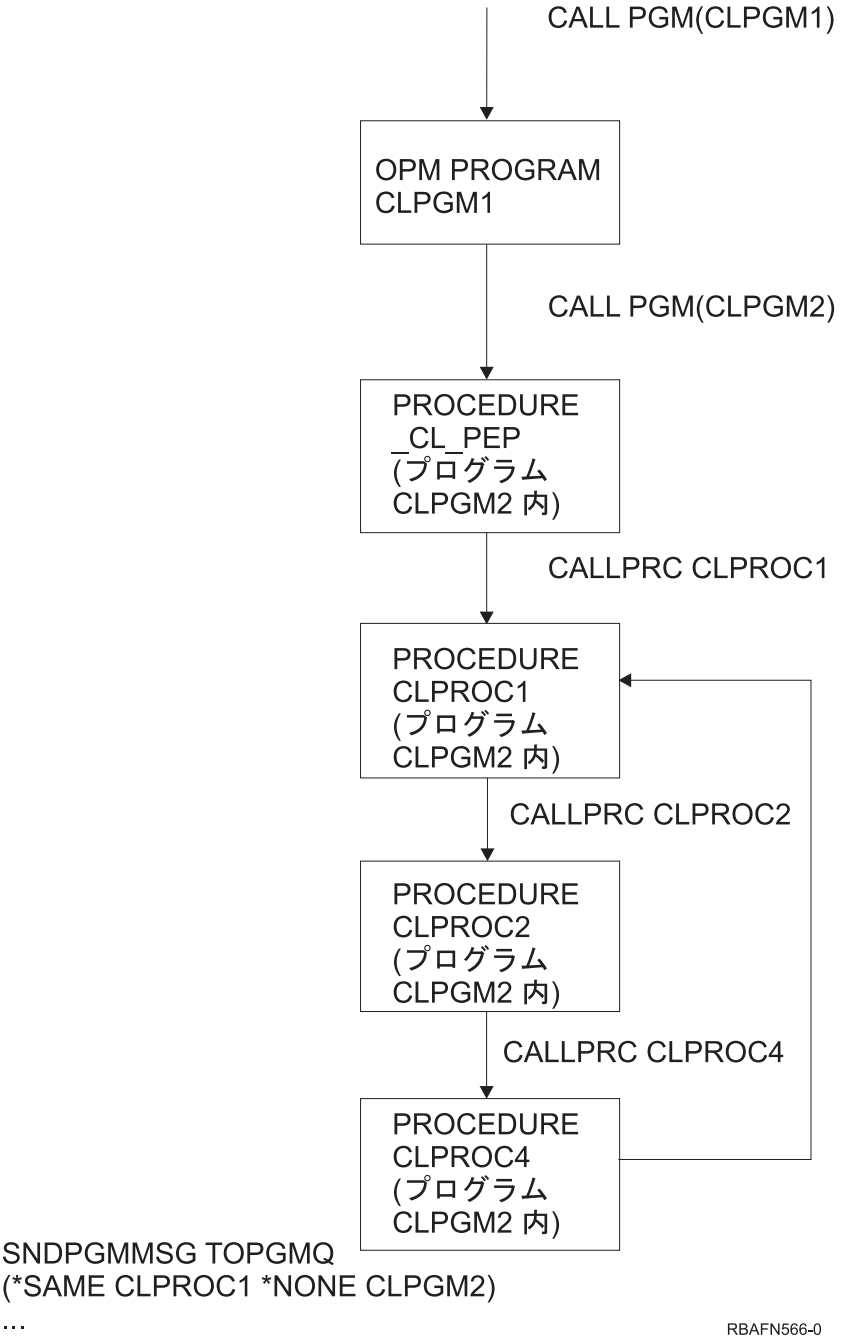


図 *19.* 基本項目としての複合名の使用例

関連資料:

568 ページの『CL プログラムからのメッセージの送信に使用されるコマンド』
CL プロシージャーまたはプログラムからのメッセージの送信には、 **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドまたは**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用します。

関連情報:

プログラム・メッセージ送信 (SNDPGMMSG) コマンド

基本項目としてのプログラム境界の使用 **(*PGMBDY):**

基本呼び出しスタック項目としてプログラム境界を指定するには、*PGMBDY 特殊値を使用します。

特殊値 *PGMBDY を単独でまたはプログラム名とともに使用して、CL プログラムの PEP を識別します。識別された CL プログラムの PEP の項目は基本項目になります。このオプションは、CL プロシージャー内からそのプロシージャーを含むプログラムの境界の外部へメッセージを送りたい場合に役立ちます。

特殊値 *PGMBDY を使用してメッセージを送る例については、最初の図を参照してください。この例では、CLPROC4 は CLPGM1 に直接メッセージを送っています。CLPGM1 は CLPROC4 に含まれているプログラム CLPGM2 の呼び出し側です。 CLPGM2 を呼び出したプログラムがわからない場合や、メッセージを送るプロシージャーと比較される PEP の位置がわからない場合でも、CLPROC4 はこの操作を行えます。この例ではプログラム名を指定せず、*PGMBDY だけを使用します。したがって境界が識別されるプログラムは、メッセージを送るプロシージャーを含むプログラムです。

特殊値 *PGMBDY とプログラム名を使用してメッセージを送る例については、2 番目の図を参照してください。次のプログラムとプロシージャーが 2 番目の図で使用されます。

- CLPGM1 と CLPGM2。これらは前の例で定義されています。
- CLPGM3。これは、別の統合言語環境 (ILE) プログラムです。
- CLPGM3 中の CLPROCA。メッセージは CLPROCA から CLPGM2 の呼び出し側に送られます。

特殊値 *PGMBDY とプログラム名 CLPGM2 を使用すると、メッセージは CLPROCA から CLPGM2 の呼び出し側に送られます。

この例で、TOPGMQ パラメーターが TOPGMQ(*PRV _CL_PEP) と指定されると、メッセージは CLPGM2 の呼び出し側ではなく CLPGM3 の呼び出し側に送られます。名前で呼び出された最新のプロシージャーが CLPGM3 の PEP なので、このようになります。

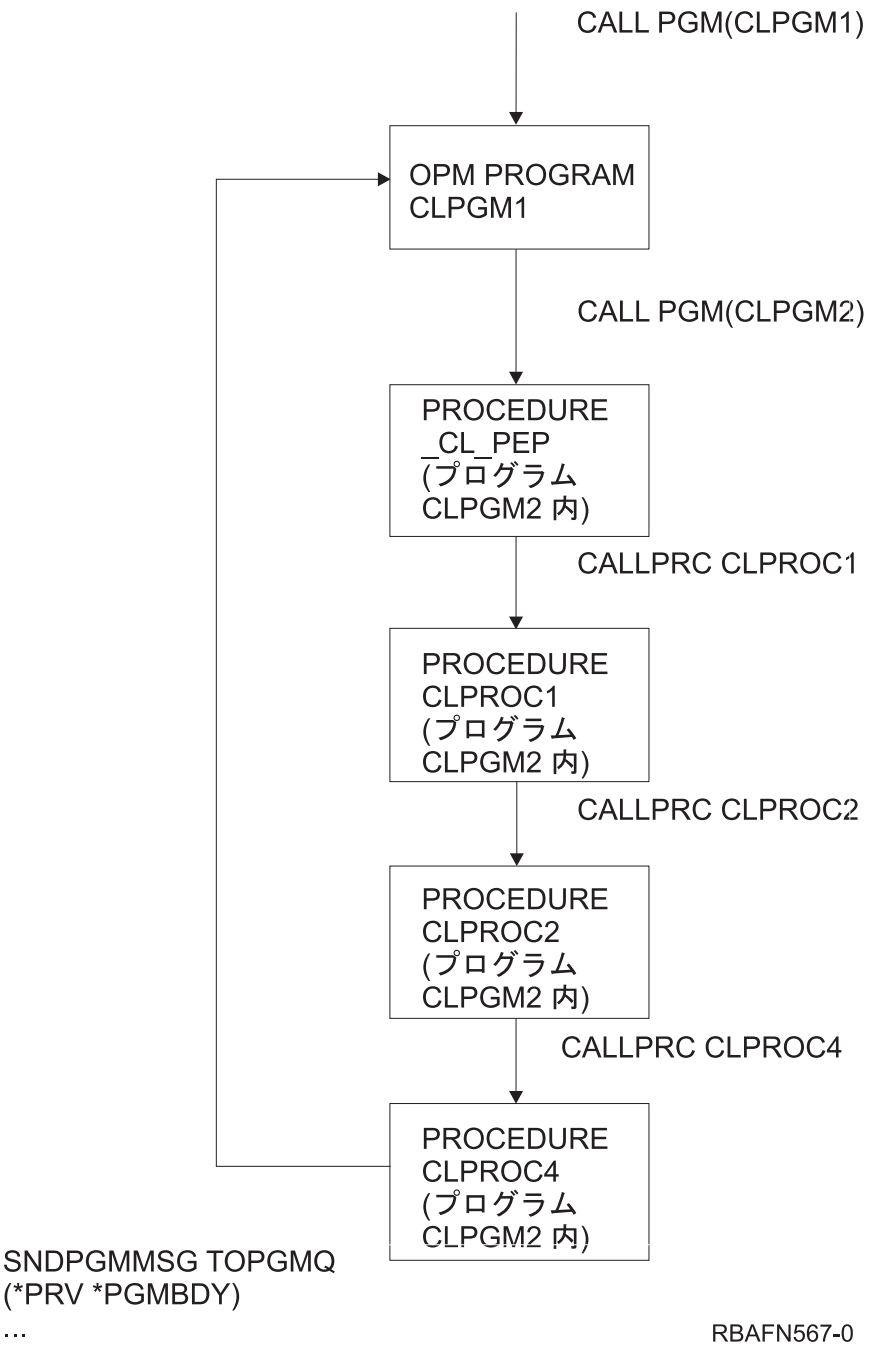


図 *20.* 単純名としての **PGMBDY* の使用例

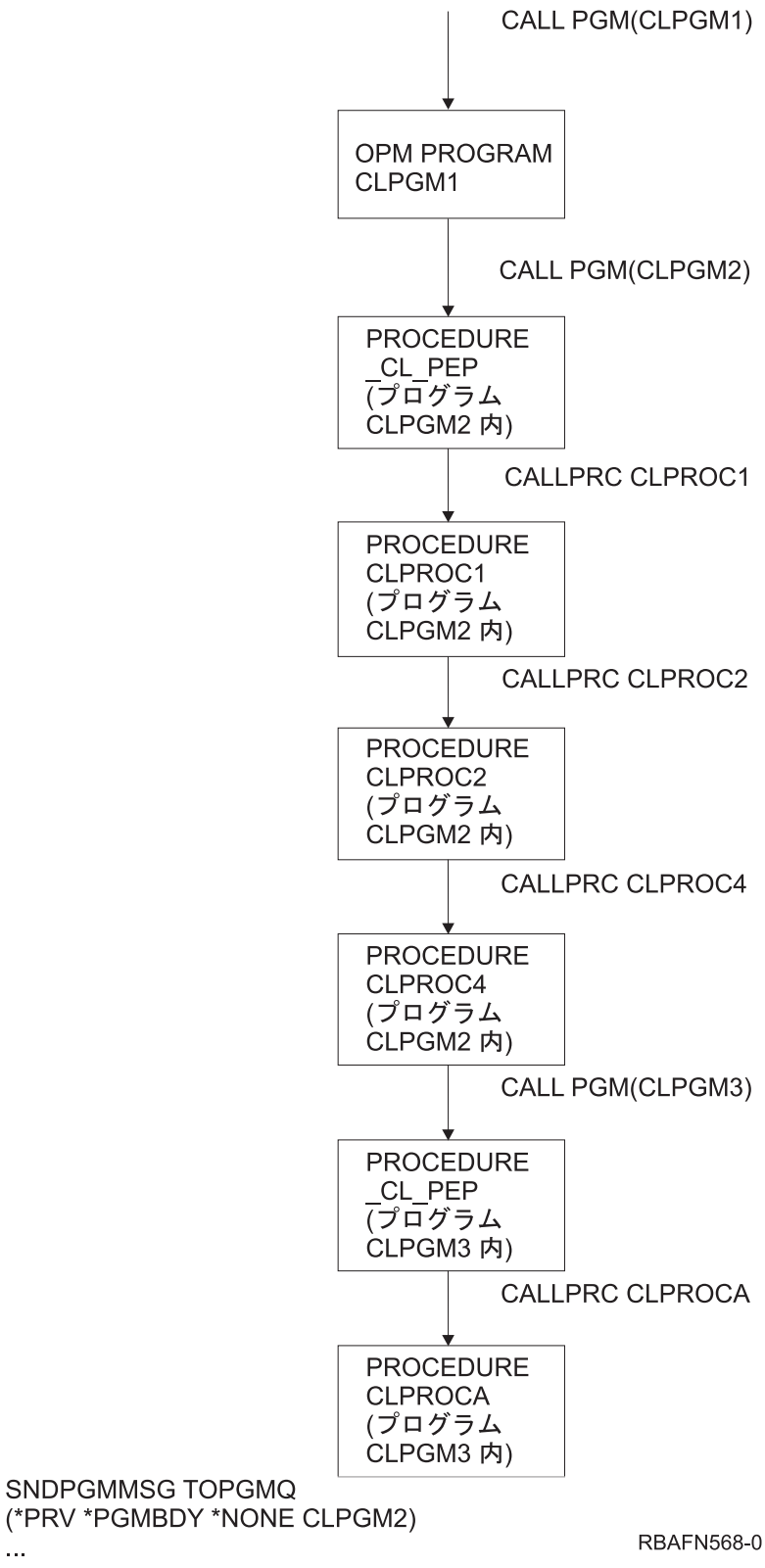


図 *21.* 複合名としての **PGMBDY* の使用例

オリジナル・プログラム・モデル (OPM) プログラムを使用する場合も特殊値 *PGMBDY を使用できます。 OPM プログラム名と *PGMBDY を指定すると、OPM プログラム名だけを使用した場合と同じ結果

になります。例えば TOPGMQ (*SAME *PGMBDY *NONE OPM プログラム名) は、TOPGMQ (*SAME OPM プログラム名) と同じ場所にメッセージを送ります。自分自身を再帰的に呼び出す OPM プログラムにメッセージを送る場合は例外です。TOPGMQ (*SAME プログラム名) と指定すると、最後の再帰レベルにメッセージを送ります。しかし、TOPGMQ (*SAME *PGMBDY *NONE プログラム名) と指定すると最初の再帰レベルにメッセージを送ります。下の図には、PGM1 が呼び出されてから PGM1 自身をさらに 2 回再帰的に呼び出す方法が説明されています。 3 回目の再帰レベルで PGM1 は PGM2 を呼び出します。 PGM2 はその後メッセージを PGM1 に戻します。 PGM1 名だけを使用してプログラム・メッセージを送ると、メッセージは PGM1 の 3 回目の再帰レベルに送られます。 PGM1 名と特殊値 *PGMBDY を使用してプログラム・メッセージを送ると、メッセージは PGM1 の 1 回目の再帰レベルに送られます。

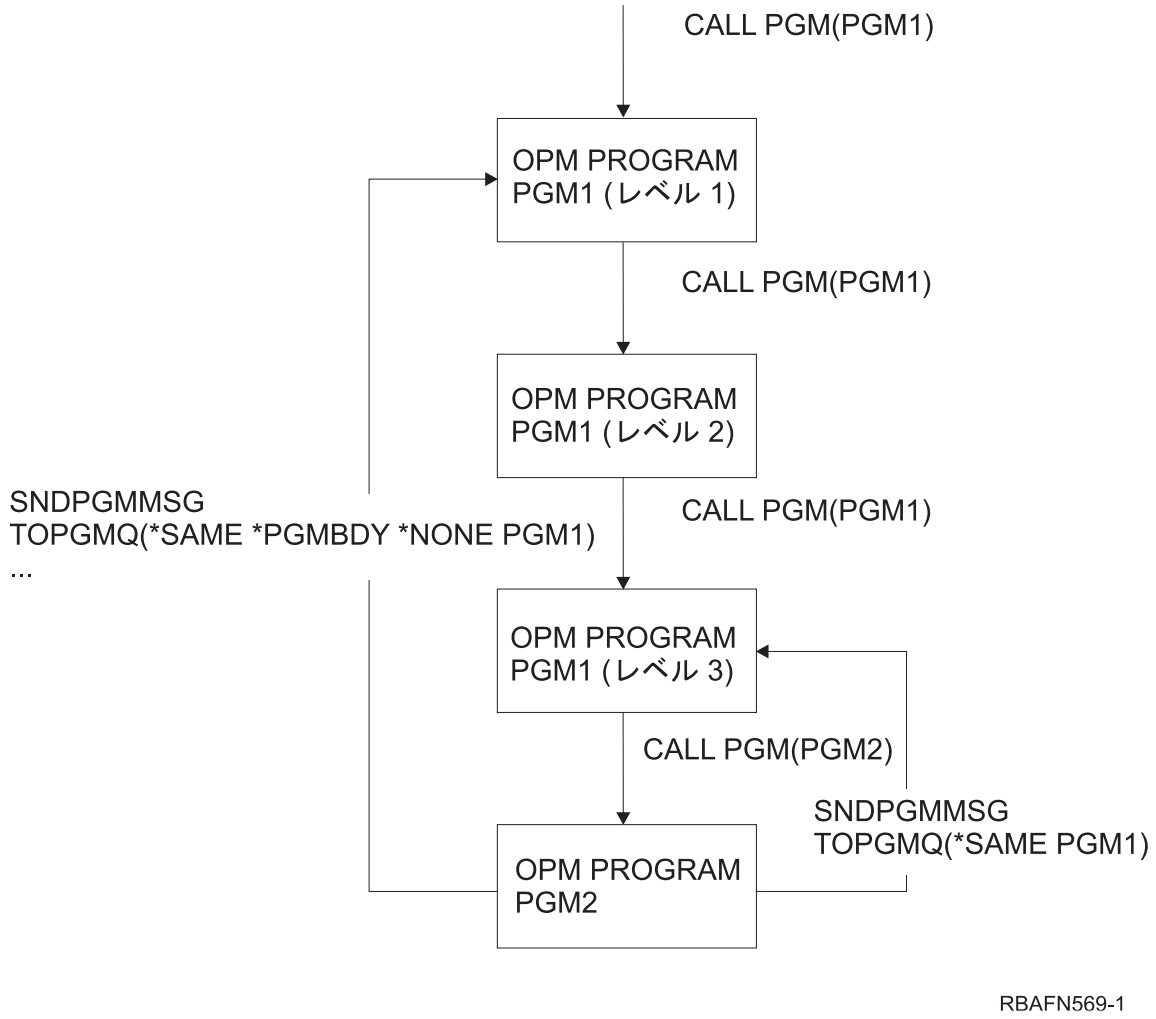


図 22. 再帰を伴った *OPM* 名を持つ **PGMBDY* の使用例

基本項目として最後に呼び出されたプロシージャーの使用 **(*PGMNAME):**

基本呼び出しスタック項目として最後に呼び出されたプロシージャーを指定するには、*PGMNAME 特殊値を使用します。

プロシージャーの名前が分からない場合に、最後に呼び出された統合言語環境 (ILE) プログラムのプロシージャーにメッセージを戻すことができます。特殊値 *PGMNAME は基本項目名を、識別されたプログラ

ムの最後に呼び出されたプロシージャーの名前として使用するため ILE プログラム名と共に使用されます。この例では以下の 3 つのプログラムが使用されています。

- CLPGM51 は、プロシージャー PROCA および PROCB を持つ ILE プログラムです。
- CLPGM52 および CLPGM53 は、両方ともオリジナル・プログラム・モデル (OPM) プログラムです。
- CLPGM53。これはメッセージを CLPGM51 に送りますが、最後に呼び出されたプロシージャーを認識しません。

特殊値 *PGMNAME とプログラム名 CLPGM51 を使用して送信を行います。

全部ではなく一部の CL プログラムだけを ILE プログラムに変換する場合には、特殊値 *PGMNAME が役立ちます。例えば、CLPGM71 は OPM CL プログラムです。CLPGM73 はメッセージを CLPGM71 に送り、TOPGMQ(*SAME CLPGM71) を指定します。 CLPGM71 が ILE に変換されると、CLPGM73 (OPM) 内の *PGM 名を持つ **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドのみが実行可能です。単純名として CLPGM71 を指定すると、CLPGM71 の呼び出しスタックに項目が何もないため機能しません。このコマンドを TOPGMQ(*SAME *PGMNAME *NONE CLPGM71) に変更すると、CLPGM73 は、プロシージャー名に使用した名前にかかわらず、CLPGM71 に正常にメッセージを送ることができます。

特殊値 *PGMNAME を OPM プログラム名とともに使用することもできます。この場合の結果はプログラム名を使用した場合と同じです。例えば、TOPGMQ (*SAME *PGMNAME *NONE OPM プログラム名) は TOPGMQ (*SAME OPM プログラム名) と同じ場所にメッセージを送ります。メッセージを OPM プログラム名と ILE プログラム名のどちらに送るかユーザーが判別できない場合は、*PGMNAME を使用することを考慮する必要があります。

関連概念:

『基本項目としての制御境界の使用 (*CTLBDY)』

基本呼び出しスタック項目を最も近い制御境界の項目として識別するには、特殊値 *CTLBDY を使用します。

基本項目としての制御境界の使用 **(*CTLBDY):**

基本呼び出しスタック項目を最も近い制御境界の項目として識別するには、特殊値 *CTLBDY を使用します。

2 つの呼び出しスタック項目が 2 つの別個の活動化グループ内で実行している場合、制御境界はこの 2 つの項目の間にあります。この特殊値を使用して識別される項目は、メッセージの送信先の項目として同一の活動化グループ内で実行しています。

特殊値 *CTLBDY を使用してメッセージを送信する例については、2 番目の図を参照してください。この例の 3 つのプログラム (CLPGM91、CLPGM92、および CLPGM93) はすべて統合言語環境 (ILE) プログラムです。CLPGM91 は活動化グループ AG91 で実行し、CLPGM92 と CLPGM93 の両方は活動化グループ AG92 で実行します。この例では、PROC93A は AG92 の境界の直前にある項目にメッセージを送ります。

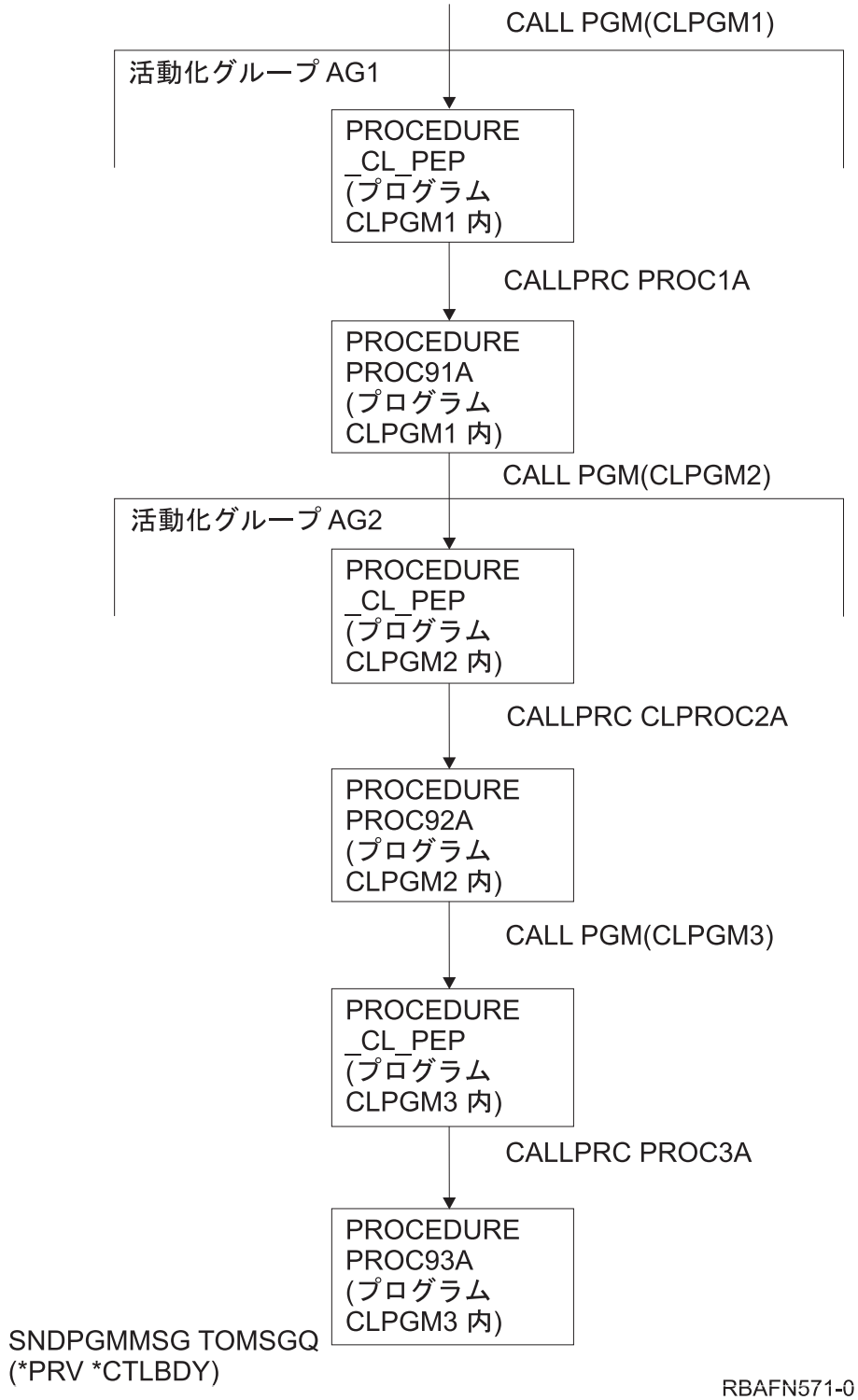


図 *23. *CTLBDY の使用例*

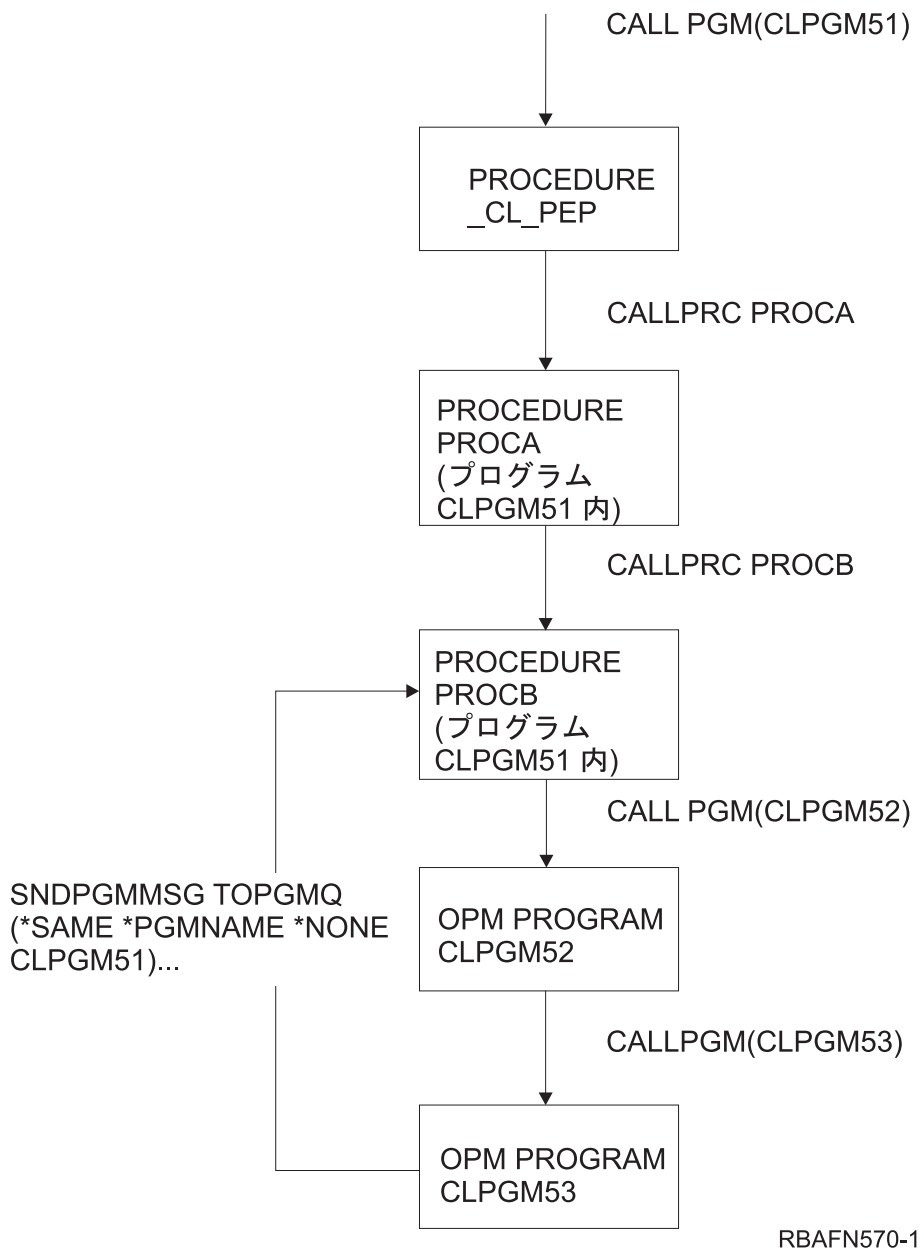


図 *24. *PGMNAME の例*

関連概念:

586 ページの『基本項目として最後に呼び出されたプロシージャーの使用 (*PGMNAME)』
基本呼び出しスタック項目として最後に呼び出されたプロシージャーを指定するには、*PGMNAME 特殊値を使用します。

サービス・プログラムの考慮事項:

統合言語環境 (ILE) プログラムは、多くの点で ILE サービス・プログラムと異なります。

前述の説明は ILE プログラムと ILE サービス・プログラムの両方に適用されます。 ILE プログラムと ILE サービス・プログラムの最も重要な相違点は、メッセージの処理に関するものです。サービス・プログラムには PEP がありません。

基本項目の識別に使用するすべてのオプションに PEP が必要なわけではなく、例外的に必要とされるのは、名前 _CL_PEP を明示的に使用する場合だけです。例えば、TOPGMQ(*PRV *PGMBDY) と指定すると、常に ILE プログラムかサービス・プログラムの呼び出し側にメッセージを送ります。 ILE プログラムの場合は、PEP は *PGMBDY 値により基本項目として識別されます。 ILE サービス・プログラムの場合は、そのサービス・プログラム内で最初に呼び出されたプロシージャーの項目が *PGMBDY 値により識別されます。

CL プロシージャーまたはプログラムによるメッセージの受信

プロシージャーまたはプログラムのメッセージ待ち行列からメッセージを取得するには、**メッセージ受信 (RCVMSG)** コマンドを使用します。

メッセージは以下の方法で受け取ることができます。

- メッセージ・タイプ別の受信。すべてのタイプまたは特定のタイプのメッセージの受信を指定することができます (MSGTYPE パラメーター)。新しいメッセージ (プロシージャーまたはプログラムでまだ受け取っていないメッセージ) の場合には、メッセージの受信は先入れ先出し (FIFO) の順序で行われます。ただし、ESCAPE タイプのメッセージは後入れ先出し (LIFO) の順序で受け取られます。
- メッセージ参照キーによる受信。以下のいずれかの操作が可能です。
  - メッセージ参照キーを用いたメッセージの受信。システムは、メッセージ待ち行列上の各メッセージにメッセージ参照キーを割り当て、キーが印字不可能なために、変数データとして引き渡します。この変数は CL プロシージャーまたは CL プログラムで宣言する必要があります (DCL コマンド)。また、**メッセージ受信 (RCVMSG)** コマンドに MSGKEY パラメーターを指定して、キーを渡す CL 変数を識別する必要があります。これは、特定のメッセージが受信されることを示します。
  - 指定のメッセージ参照キーを持つメッセージの次の、メッセージ待ち行列上のメッセージの受信。MSGKEY パラメーターの指定に加えて、MSGTYPE(*NEXT) も指定しなければなりません。
  - 指定のメッセージ参照キーを持つメッセージの前の、メッセージ待ち行列上のメッセージの受信。MSGKEY パラメーターの指定に加えて、MSGTYPE(*PRV) も指定しなければなりません。
- メッセージ待ち行列上の位置によるメッセージの受信。メッセージ待ち行列上の最初のメッセージに対しては MSGTYPE(*FIRST) を指定しなければなりません。最後のメッセージに対しては、MSGTYPE(*LAST) を指定します。
- メッセージ・タイプとメッセージ参照キーの両方による受信 (MSGTYPE パラメーターおよび MSGKEY パラメーター)。

メッセージを受け取る場合に、以下の事項を指定できます。

- メッセージ待ち行列。どのメッセージ待ち行列からメッセージを受け取るかを指定します。
- メッセージ・タイプ。 特定のメッセージ・タイプまたは *ANY の値を指定して、任意のタイプのメッセージを表します。
- メッセージの着信を待機するかどうか。待ち時間が経過してもメッセージを受け取ることができなかった場合には、値を戻すように指定した各 CL 変数はブランク (数字の場合はゼロ) で埋められ、**メッセージ受信 (RCVMSG)** コマンドを実行しているプロシージャーまたはプログラムに制御権が返されます。
- メッセージをその受信後にメッセージ待ち行列から除去するかどうか。除去しない場合には、そのメッセージはメッセージ待ち行列上で古いメッセージとなり、メッセージ参照キーを使用する場合に限り再び (プロシージャーによって) 受け取ることができます。しかし、**メッセージ待ち行列変更 (CHGMSGQ)**

コマンドによってメッセージ待ち行列上のメッセージが新しいメッセージにリセットされる場合、メッセージ参照キーを使用しなくてもそのメッセージを受け取ることができます。ただし、既に応答が返されている照会メッセージは、新規メッセージの状態にリセットすることはできません。

- 変換する CCSID 形式。メッセージ・テキストを戻すさいの CCSID 形式を指定します。
- 以下の情報が入る CL 変数のグループ (各事項が**メッセージ受信 (RCVMSG)** コマンドのそれぞれ 1 つの変数に対応します)。1 つ以上の以下の戻り変数を指定できます。
  - メッセージ待ち行列上のメッセージの参照キー (4 文字の文字変数)
  - メッセージ (可変長の文字変数)
  - 置換変数データの長さを含む、メッセージの長さ (5 桁の 10 進変数)
  - メッセージのオンライン・ヘルプ情報 (可変長の文字変数)
  - 置換変数データの長さを含む、メッセージ・ヘルプの長さ (5 桁の 10 進変数)
  - メッセージの送信側によって提供された置換変数に対応するメッセージ・データ (可変長の文字変数)
  - メッセージ・データの長さ (5 桁の 10 進変数)
  - メッセージ識別コード (7 文字の文字変数)
  - 重大度コード (2 桁の 10 進変数)
  - メッセージの送信元 (最低 80 文字の文字変数、現行のユーザー情報を取得するには最低 87 文字が必要)
  - 受け取ったメッセージのタイプ (2 文字の文字変数)
  - 受け取ったメッセージの警報オプション (9 文字の文字変数)
  - 事前定義メッセージが入っているメッセージ・ファイル (10 文字の文字変数)
  - メッセージを受け取るために使用されるメッセージ・ファイルが入っているメッセージ・ファイル・ライブラリーの名前 (10 文字の文字変数)
  - メッセージを送るために使用されたメッセージ・ファイルが入っているメッセージ・ファイル・ライブラリーの名前 (10 文字の文字変数)
  - メッセージ・データ CCSID (5 桁の 10 進変数)。これは、戻される置換データに関連するコード化文字セット ID です。
  - テキスト・データ CCSID (5 桁の 10 進変数)。これは、メッセージおよびメッセージ・ヘルプ・パラメーターによって戻されるテキストに関連するコード化文字セット ID です。

以下のサンプルの**メッセージ受信 (RCVMSG)** コマンドでは、メッセージは、QGPL ライブラリーのメッセージ待ち行列 INVN から取得されます。受信されたメッセージ・テキストは、変数 &MSG に配置されます。*ANY は、MSGTYPE パラメーターのデフォルト値です。

```
RCVMSG MSGQ(QGPL/INVN) MSGTYPE(*ANY) MSG(&MSG)
```

CL 以外の言語で作成された統合言語環境 (ILE) プロシージャーの呼び出しスタック項目メッセージ待ち行列で処理を行う場合には、例外が処理される前であれば例外メッセージ (エスケープまたは通知) を受け取ることができます。 **メッセージ受信 (RCVMSG)** コマンドを使用して、メッセージの受け取りと、例外が処理されたことをシステムへ通知することの両方を行うことができます。

上記の処置は、RMV キーワードを使用して制御することができます。このキーワードに *NO を指定すると、例外が処理され、そのメッセージは古いメッセージとしてメッセージ待ち行列に残ります。また *KEEPEXCP を指定すると例外は処理されず、そのメッセージは新しいメッセージとしてメッセージ待ち行列に残ります。 *YES を指定した場合には、例外メッセージは処理され、そのメッセージはメッセージ待ち行列から除去されます。

RTNTYPE キーワードを使用して、受け取ったメッセージが例外メッセージであるかどうかを確認することができます。また例外メッセージであれば、例外が処理されているかどうかを確認することができます。

関連タスク:

598 ページの『メッセージ待ち行列からメッセージを除去する』
メッセージは、除去されるまでメッセージ待ち行列に保持されます。

要求メッセージの受信

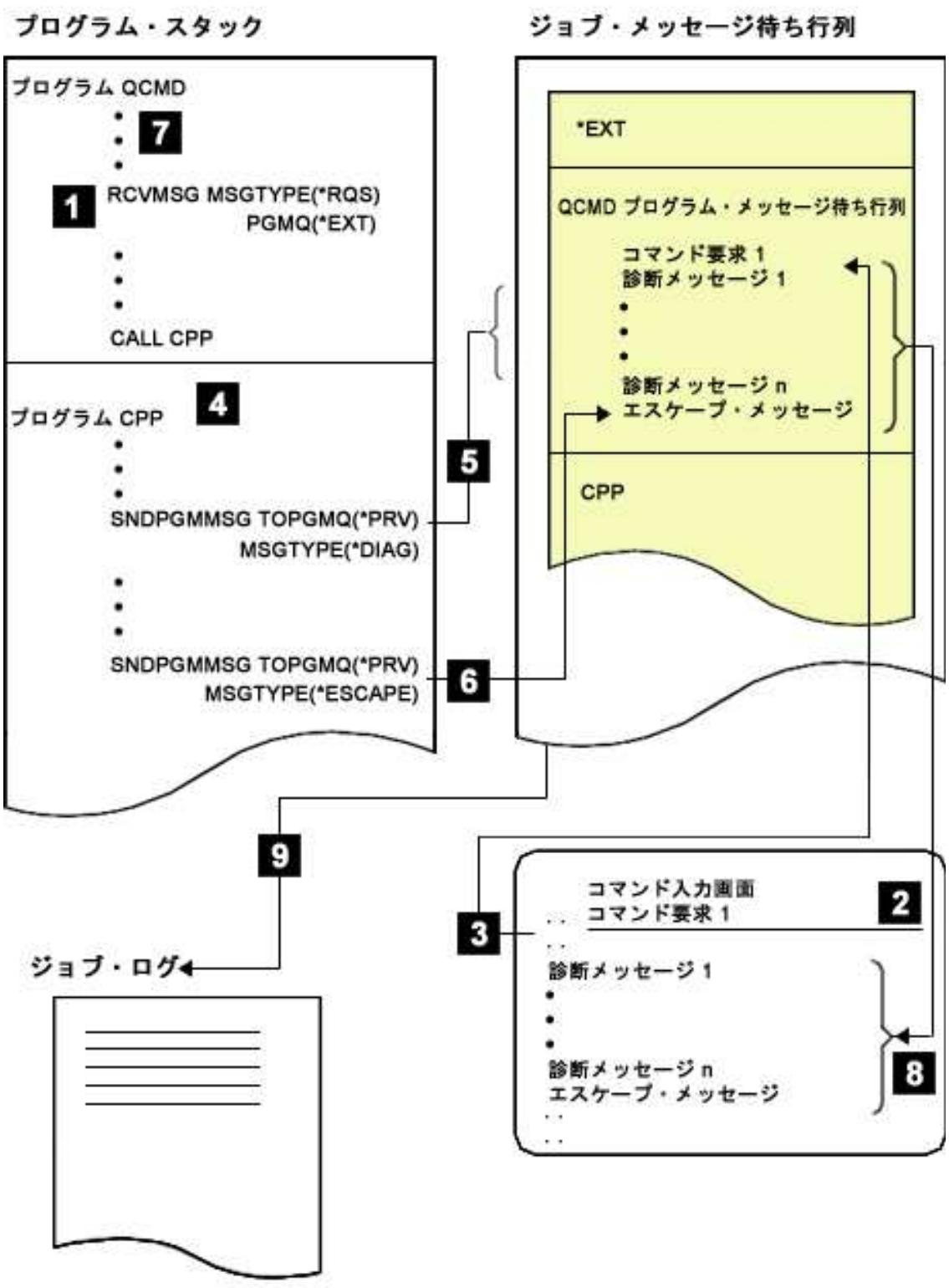
要求メッセージの受信は、CL プロシージャーまたは CL プログラムが CL コマンドを処理するための 1 つの方法です。

例えば、ユーザーのプロシージャーまたはプログラムは、ディスプレイ装置からの入力 (コマンド) を受け取り、入力の分析と処理の結果として生じるメッセージを処理することができます。コマンドは要求メッセージとして送信されるため、処理可能です。通常、要求メッセージはジョブの外部メッセージ待ち行列 (*EXT) から受け取ります。バッチ・ジョブの場合には、受け取る要求は入力ストリームから読み込まれたものです。対話式ジョブの場合には、要求は、ディスプレイ装置ユーザーがコマンド入力画面で一度に 1 つずつ入力します。例えば、CL コマンドは IBM 提供の CL 処理プログラム QCMD によって受け取られる要求です。

ユーザーのプロシージャーまたはプログラムは要求メッセージのデータの構文を定義し、その要求を解釈し、エラーがあればその診断を行わなければなりません。要求の分析や要求機能の実行過程で、多くのエラーが検出される場合があります。エラーがある場合には、そのプロシージャーまたはプログラムの呼び出しメッセージ待ち行列にメッセージが送られます。これらのメッセージはプロシージャーまたはプログラムによって処理され、そして次の要求メッセージが受け取られます。このように、要求処理のサイクルは、要求メッセージの受信、プロシージャーまたはプログラムによる要求の分析および実行とその結果の表示、そして次の受信というサイクルで続けられます。バッチ・ジョブでは、受け取るべき要求メッセージがなくなった時点で、その旨を知らせるエスケープ・メッセージがユーザーのプロシージャーまたはプログラムに送られます。

1 つのジョブの複数のオリジナル・プログラム・モデル (OPM) プログラムまたは統合言語環境 (ILE) プロシージャーによって、要求メッセージを受け取り、処理することが可能です。後のプログラム呼び出しにより受け取られた要求は、それより前の (すなわち上位の) プログラム呼び出しにより受け取られた要求の内部にネストされるものと見なされます。各ネスト・レベルにおける要求処理のサイクルは、それぞれ他のレベルから独立しています。 ILE プログラム内では、そのプログラム内の 1 つまたは複数のプロシージャーに要求メッセージを送ることができます。複数のプロシージャーが要求を処理している場合は、同一の ILE プログラムではネストが起こらず、ネスト・レベルは独立したままです。

以下の図は、QCMD により要求メッセージがどのように処理されるかを示しています。



1　　CL 処理プログラム QCMD が、*EXT から要求メッセージを受け取ります。

2 *EXT 上に要求メッセージがない場合には、コマンド入力画面が表示されます。ディスプレイ装置ユーザーが、この画面にコマンドを入力します。入力したコマンドは、要求メッセージとして *EXT に入れられます。

3 ステップ 2 により、*EXT に要求が配置されたため、ステップ 1 の RCVMSG は継続可能です。次に、そのコマンドは、QCMD の呼び出しメッセージ待ち行列に移され、そこから QCMD に渡され、ステップ 1 を完了します。

4 コマンドが分析され、そのコマンド処理プログラム (CPP) が呼び出されます。

5 コマンド処理プログラムは、QCMD の呼び出しメッセージ待ち行列に診断メッセージを送ります。

6 次に、コマンド処理プログラムは、QCMD の呼び出しメッセージ待ち行列にエスケープ・メッセージを送ります。このエスケープ・メッセージは、待ち行列上に診断メッセージがあること、および QCMD が CPP の処理を打ち切らなければならないことを QCMD に伝えます。

7 QCMD は、要求チェックのエスケープ・メッセージ (CPF9901) または機能チェックのエスケープ・メッセージ (CPF9999) の着信を監視します。次に、QCMD は次の要求メッセージの受信を試みます。

注**:** 要求処理プログラムがメッセージ CPF9901 または CPF9999 を受け取った場合には、そのプログラムは資源再利用 (RCLRSC) コマンドを実行しなければなりません。また、要求処理プログラムは、メッセージ CPF1907 (要求終了) および CPF2415 (コマンド入力画面でユーザーが F3 または F12 キーを押したことを示すメッセージ) の監視も行う必要があります。

8 要求メッセージの処理が行われたので、QCMD の呼び出しメッセージ待ち行列上のすべてのメッセージがコマンド入力画面に表示され、ディスプレイ装置ユーザーに対してさらに別のコマンドの入力を求めるプロンプトがその画面に表示されます。

9 前の要求メッセージ (コマンド) とそれに関連したメッセージが、このジョブについて指定されているメッセージ・ロギング・レベルに従ってジョブ・ログに入れられます。

関連概念:

620 ページの『メッセージのログ』
メッセージのログには、ジョブ・ログと活動記録ログという 2 つのタイプがあります。

関連資料:

568 ページの『CL プログラムからのメッセージの送信に使用されるコマンド』
CL プロシージャーまたはプログラムからのメッセージの送信には、 **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドまたは**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用します。

要求処理プロシージャーおよびプログラムの書き込み

プロシージャーまたはプログラムを要求処理プログラム・プロシージャーまたはプログラムとして作成するには、要求メッセージを送信および受信し、要求メッセージを除去しない必要があります。 **プログラム・メッセージ送信 (SNDPGMMSG)** および**メッセージ受信 (RCVMSG)** コマンドによって、プロシージャーまたはプログラムを要求処理プログラムにすることができます。

CL プロシージャーをプログラム内の要求処理プログラムとして指定すると、いくつかの利点があります。3 つの利点を次に挙げます。

- 592 ページの『要求メッセージの受信』に述べたように、要求メッセージが処理される。
- システム要求メニューから、またはジョブの切断機能の一部として使用できる、**要求終了 (ENDRQS)** コマンドの使用が可能になる。
- メッセージをフィルターにかけることが可能になる。

例えば、以下のようなコマンドを使用すると、プロシージャーまたはプログラムを要求処理プログラムにすることができます。

```
SNDPGMMSG  MSG('Request Message') TOPGMQ(*EXT) MSGTYPE(*RQS)
RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
```

PGMQ *EXT から要求メッセージを受信します。要求メッセージを受信するとそのメッセージは、**RCVMSG** コマンドを指定したプロシージャーまたはプログラムの呼び出しメッセージ待ち行列に移動します (実際には、除去され再送される)。したがって、メッセージを除去する場合は、後で正しい呼び出しメッセージ待ち行列を使用することが必要です。

メッセージ参照キー (MRK) を使用して要求メッセージを除去する場合は、**SNDPGMMSG** コマンドではなく **RCVMSG** コマンドの KEYVAR キーワードから MRK を取得してください。(要求は他の呼び出しメッセージ待ち行列に移動しているため、MRK は要求メッセージを受信するときに変更します。) 要求メッセージが呼び出しメッセージ待ち行列から除去される場合、プロシージャーまたはプログラムは要求処理プログラムではないので、RMV(*NO) を **RCVMSG** コマンドに指定する必要があります。

プロシージャーまたはプログラムは、要求メッセージを受け取った時点で要求処理プログラムになります。プロシージャーまたはプログラムが要求処理プログラムである場合は、呼び出された他のプロシージャーまたはプログラムを、システム要求メニューのオプション 2 (異常終了) を用いて終了させることができます。要求処理プログラム・プロシージャーまたはプログラムには、メッセージ CPF1907 に対する監視機能を含めておかなければなりません (**メッセージ・モニター (MONMSG)** コマンド)。このことが必要な理由は、要求終了機能が (システム要求メニューのオプション 2、または要求終了コマンドによって) 実行されると、このメッセージが要求処理プログラムに送られるためです。

プロシージャーまたはプログラムは、プロシージャーが終了 (正常終了または異常終了) するまで、または **RCVMSG** コマンドによって要求処理プログラムの呼び出しメッセージ待ち行列からすべての要求メッセージが除去されるまで、要求処理プログラムとして機能します。例えば、以下のコマンドを出すと、すべての要求メッセージがメッセージ待ち行列から除去され、したがって要求処理も終了します。

```
RMVMSG  CLEAR(*ALL)
```

IBM i コマンド分析プログラムに IBM i コマンドの要求メッセージを処理させるには、QCAPCMD API を呼び出し、メッセージ参照キーを指定してください。メッセージ参照キーは、要求メッセージの受信時に入手することができます。コマンド処理 (QCAPCMD) は、ジョブ・ログ内の要求メッセージを更新し、提供された新しい値を追加します。また、QCAPCMD は、ジョブ・ログ内で隠されるすべてのパラメーター値 (パスワードなど) を隠します。システムは、次の 2 つの条件のどちらかが存在するときには、ジョブ・ログ内の要求メッセージを更新しません。

- コマンド実行 (QCMDEXC または QCAEXEC) API が使用されている。
- QCAPCMD に対するメッセージ参照キーの提供に失敗した。

要求処理プログラムの有無の判別

ジョブに要求処理プログラムがあるかどうかを判別するには、そのジョブの呼び出しスタックを表示する必要があります。

そのためには、**ジョブ表示 (DSPJOB)** コマンドまたは**ジョブ処理 (WRKJOB)** コマンドでオプション 11 を使用するか、または WRKACTJOB の画面にリストされた該当のジョブに対してオプション 10 を選択してください。ジョブの呼び出しスタックの画面の要求レベル列に番号が表示される場合、番号のあるプログラムまたは統合言語環境 (ILE) プロシージャーは要求処理プログラムです。以下の例では、QCMD と QTEVIREF の 2 つが要求処理プログラムです。

```
                          呼び出しスタックの表示

システム:   S0000000
ジョブ:   WS31        ユーザー:   QSECOFR    番号:   000173

オプションを入力して，実行キーを押してください。
  5= 詳細表示

     要求      プログラムまたは
Opt  レベル    プロシージャー      ライブラリー        ステートメント      命令
               QCMD                QSYS                                    /01DC
       1       QCMD                QSYS                                    /016B
               QTECADTR            QSYS                                    /0001
       2       QTEVIREF            QSYS                                    /02BA









            終わり

F3= 終了     F5= 最新表示    F11= 活動化グループの表示    F12= 取り消し
F17= 先頭    F18= 最後       F21= 組み込み    F22= フィールド全体の表示
```

次に示すのは要求処理プロシージャーの一例です。

```
PGM
    SNDPGMMSG  MSG('Request Message') TOPGMQ(*EXT) MSGTYPE(*RQS)
    RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
      .
      .
      .
    CALL   PGM(PGMONE)
    MONMSG MSGID(CPF1907)
      .
      .
      .
    RMVMSG CLEAR(*ALL)
    CALL   PGM(PGMTWO)
      .
      .
      .
ENDPGM
```

最初の 2 つのコマンドにより、このプロシージャーは要求処理プログラムになります。このプロシージャーは、**メッセージ除去 (RMVMSG)** コマンドが実行されるまで要求処理プログラムとして機能します。プログラム PGMONE の呼び出し後にメッセージ・モニター・コマンドが続いていますが、これは、PGMONE から要求処理プログラムに対して終了要求が送られる可能性があるためです。この監視機能が使用されていないと、終了要求が出された場合に機能チェックが発生します。要求処理は**メッセージ除去 (RMVMSG)** コマンドにより終了するので、PGMTWO の呼び出しの後にはメッセージ・モニターは指定されていません。

要求プロシージャーまたはプログラムが呼び出されていないときに終了要求が出された場合にはエラー・メッセージが出されて、終了操作は行われないことに注意してください。

注**:** サンプル・プログラムでは、**メッセージ受信 (RCVMSG)** コマンドは、要求プロセッサーになるために必要な最小限の数のパラメーターを使用しています。要求メッセージを受け取りたいが、除去はしたくないということを伝えなければなりません。また、メッセージ要求の発信元の特定の呼び出し待ち行列を識別する必要もあります。他のパラメーターは、必要に応じて追加されます。

メッセージ・ファイルからのメッセージ記述の検索

メッセージ・ファイルからメッセージのテキストを検索して、CL プログラムまたはプロシージャーの変数に入れるには、**メッセージ検索 (RTVMSG)** コマンドを使用します。**RTVMSG** コマンドは、事前定義メッセージ記述を処理します。

以下の項目に加えて、メッセージ識別コードとメッセージ・ファイル名を指定することができます。

- 変換する CCSID 形式。メッセージ・テキストとデータを戻す際のコード化文字セット ID を指定します。
- メッセージ・データ・フィールド。置換変数に挿入されるメッセージ・データ
- メッセージ・データ CCSID。メッセージ・データを送信する際に考慮されるコード化文字セット ID を指定します。
- 以下の情報が入る CL 変数のグループ (各事項がそれぞれ 1 つの変数に対応します)。
  - メッセージ (可変長の文字変数)
  - 置換変数データの長さを含む、メッセージの長さ (5 桁の 10 進変数)
  - メッセージのヘルプ情報 (可変長の文字変数)
  - 置換変数データの長さを含む、メッセージ・ヘルプの長さ (5 桁の 10 進変数)
  - 重大度コード (2 桁の 10 進変数)
  - 警報オプション (9 文字の文字変数)
  - サービス活動ログのログ問題 (1 文字の文字変数)
  - メッセージ・データ CCSID (5 桁の 10 進変数)。これは、戻される置換データに関連するコード化文字セット ID です。
  - テキスト・データ CCSID (5 桁の 10 進変数)。これは、メッセージおよびメッセージ・ヘルプ・パラメーターによって戻されるテキストに関連するコード化文字セット ID です。

例えば、以下のコマンドを出すと、メッセージ USR1001 に関するメッセージ記述がメッセージ・ファイル USRMSG に追加されます。

```
ADDMSGD   MSGID(USR1001) MSGF(QGPL/USRMSG) +
          MSG('File &1 not found in library &2') +
          SECLVL('Change file name or library name') +
          SEV(40) FMT((*CHAR 10) (*CHAR 10))
```

次のコマンドでは、検索されたメッセージ USR1001 の中で、10 文字の変数 &FILE 内のファイル名 INVENT と、10 文字の変数 &LIB 内のライブラリー名 QGPL の置換が起こります。

```
DCL &FILE TYPE(*CHAR) LEN(10) VALUE(INVENT)
DCL &LIB TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL &A TYPE(*CHAR) LEN(20)
DCL &MSG TYPE(*CHAR) LEN(50)
CHGVAR VAR(&A) VALUE(&FILE||&LIB)
RTVMSG   MSGID(USR1001) MSGF(QGPL/USRMSG) +
         MSGDTA(&A) MSG(&MSG)
```

置換変数 &1 と &2 のデータは、プロシージャー変数 &A に入っています。この変数の中では、プロシージャー変数 &FILE と &LIB の値が連結されています。検索コマンドが実行されたら、CL 変数 &MSG には、次のメッセージが入ります。

```
File INVENT not found in library QGPL
```

メッセージ検索 (RTVMSG) コマンドで MSGDTA パラメーターが使用されない場合は、CL 変数 &MSG に次のメッセージが入ります。

```
File not found in library
```

変数 &MSG にメッセージが入れられた後、次の作業を行うことができます。

- **プログラム・メッセージ送信 (SNDPGMMSG)** コマンドを使用してそのメッセージを送る。
- DDS のメッセージ行 (38 文字目が M の行) のテキストとしてその変数を使用する。
- メッセージ・サブファイルを使用する。
- メッセージを印刷または表示する。

メッセージ待ち行列からメッセージを除去する

メッセージは、除去されるまでメッセージ待ち行列に保持されます。

これらのメッセージは、**メッセージ除去 (RMVMSG)** コマンド、**メッセージ待ち行列消去 (CLRMSGQ)** コマンド、**メッセージ検索 (RTVMSG)** コマンドおよび**応答送信 (SNDRPY)** コマンドの RMV パラメーター、「メッセージ表示」画面の除去機能キー、または「メッセージ待ち行列の処理」画面のメッセージ待ち行列消去オプションを使用して除去することができます。除去の対象として以下のいずれかを選択することができます。

- 1 つのメッセージ
- すべてのメッセージ
- 未応答のメッセージを除くすべてのメッセージ
- すべての古いメッセージ
- すべての新しいメッセージ
- すべての非活動プログラムからのすべてのメッセージ

メッセージ除去 (RMVMSG) コマンドを使用して 1 つのメッセージを除去する場合、または**メッセージ検索 (RTVMSG)** コマンドを使用して 1 つの古いメッセージを除去する場合には、除去するメッセージのメッセージ参照キーを指定します。

注**:** メッセージ参照キーは、メッセージの受信およびメッセージに対する応答にも使用できます。

まだ応答していない照会メッセージを除去する場合は、そのメッセージの送信側にデフォルト応答が送られ、その照会メッセージと応答が除去されます。既に応答済みの照会メッセージを除去する場合には、そのメッセージとそれに対する応答の両方が除去されます。

すべての非活動プログラムまたはプロシージャー宛のすべてのメッセージをユーザーのジョブ・メッセージ待ち行列から除去する場合には、**メッセージ除去 (RMVMSG)** コマンドの PGMQ パラメーターに *ALLINACT を指定し、CLEAR パラメーターに *ALL を指定します。すべての非活動メッセージを除去する前にジョブ・ログを印刷したい場合には、OUTPUT パラメーターに *PRINT を指定した**ジョブ・ログ表示 (DSPJOBLOG)** コマンドを使用します。

統合言語環境 (ILE) プロシージャーの呼び出しメッセージ待ち行列を処理する場合、**メッセージ除去 (RMVMSG)** コマンドの実行時に、処理されていない例外に対する例外メッセージが待ち行列にある可能性があります。このコマンドの RMVEXCP キーワードを使用すれば、このタイプのメッセージに対する処置を制御することができます。このキーワードに *YES が指定されていると、RMVMSG コマンドは例外を処理し、メッセージを除去します。 *NO が指定されていると、メッセージは除去されません。その結果として、例外は処理されません。

以下の**メッセージ除去 (RMVMSG)** コマンドは、ユーザー・メッセージ待ち行列 JONES からメッセージを 1 つ除去します。メッセージ参照キーは、CL 変数 &MRKEY に入っています。

```
DCL &MRKEY TYPE(*CHAR) LEN(4)
RCVMSG MSGQ(JONES) RMV(*NO) KEYVAR(&MRKEY)
RMVMSG MSGQ(JONES) MSGKEY(&MRKEY)
```

以下の**メッセージ除去 (RMVMSG)** コマンドは、そのコマンドを含むプロシージャーの呼び出しメッセージ待ち行列からすべてのメッセージを除去します。

```
RMVMSG CLEAR(*ALL)
```

関連タスク:

590 ページの『CL プロシージャーまたはプログラムによるメッセージの受信』
プロシージャーまたはプログラムのメッセージ待ち行列からメッセージを取得するには、**メッセージ受信 (RCVMSG)** コマンドを使用します。

CL プログラムまたはプロシージャー内のメッセージの監視

*ESCAPE メッセージ、*STATUS メッセージ、および *NOTIFY メッセージは、モニター可能なメッセージであり、プログラムまたはプロシージャーで使用される各 CL コマンドによって発行されます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

IBM 提供の各コマンドは、そのヘルプ文書内では、それが生成する例外メッセージを識別しています。この情報を使用して、プログラムまたはプロシージャーにおいてモニターの対象とするメッセージを判別できます。

例外メッセージには、エスケープ、通知、および状況メッセージがあります。これらのメッセージは、ユーザーのプロシージャーまたはプログラム、あるいは別のプロシージャーまたはプログラム内のコマンドから、ユーザーの CL プロシージャーまたはプログラムの呼び出しメッセージ待ち行列に送られます。診断メッセージは監視することができません。

メッセージ・モニター (MONMSG) コマンドを使用すると、そのコマンドで指定されている条件に応じて、呼び出しメッセージ待ち行列に送信されてくる 1 つ以上のメッセージを監視することができます。したがって、**MONMSG** コマンドを使用して、該当する条件が存在している場合は、MONMSG コマンドで指定されている CL コマンドが実行されるよう指定することができます。**MONMSG** コマンドによるロジックは以下のとおりです。

エスケープ・メッセージ: エスケープ・メッセージは、送信側を強制的に終了させたエラー条件をプロシージャーまたはプログラムに伝えるために送信されます。エスケープ・メッセージを監視することにより、訂正処置または終結処置を行った上でプロシージャーまたはプログラムを終了させることができます。

状況または通知メッセージ: 状況および通知メッセージは、送信側を終了させるほど重大ではない異常条件をプロシージャーまたはプログラムに伝えるために送信されます。状況または通知メッセージを監視することにより、プロシージャーまたはプログラムでこのような条件を検出し、機能が続行されないようにすることができます。

メッセージを監視させるためには、以下の 2 つのレベルの **MONMSG** コマンドを使用することができます。

- プロシージャー・レベル。ユーザーの CL プロシージャーまたは CL プログラム中の最後の宣言コマンドの直後に **MONMSG** コマンドを指定すると、プログラムまたはプロシージャー中のすべてのコマンドから送り出される、エスケープ・メッセージ、通知メッセージ、または状況メッセージを監視できます。これはプロシージャー・レベルの **MONMSG** コマンドと呼ばれます。1 つのプロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムでプロシージャー・レベルの MONMSG コマンドを最高 100 個まで使用することができます。 (1 つの CL プロシージャーまたは OPM プログラムで、合

計 1000 個の MONMSG コマンドを含めることができます。) この方式を用いることによって、すべてのコマンドについて同じエスケープ・メッセージを常に同じ方法で処理することができます。 EXEC パラメーターはオプションであり、このパラメーターに指定できるのは GOTO コマンドだけです。

- 特定コマンド・レベル。プロシージャーまたはプログラム内の特定のコマンドの直後に MONMSG コマンドを指定すると、そのコマンドから出されたエスケープ・メッセージ、通知メッセージ、または状況メッセージだけを監視することができます。これはコマンド・レベルの MONMSG コマンドと呼ばれます。 1 つのコマンドに対してコマンド・レベルの MONMSG コマンドを最高 100 個まで使用することができます。この方式を使用することにより、異なるエスケープ・メッセージに対してそれぞれ異なる処置をとることができます。

エスケープ・メッセージ、状況メッセージ、または通知メッセージを監視するためには、MONMSG コマンドで、監視したい総称メッセージ識別コードを以下のいずれかの形式で指定しなければなりません。

- `pppmmnn`

  特定のメッセージを監視します。例えば、MCH1211 は 0 除算のエスケープ・メッセージのメッセージ識別コードです。

- `pppmm00`

  特定のライセンス・プログラム (ppp) および mm で指定される数字で始まる総称メッセージ識別コードを持つメッセージを監視します。例えば、CPF5100 は、CPF51 で始まるすべての通知メッセージ、状況メッセージ、およびエスケープ・メッセージを監視することを示します。

- `ppp0000`

  特定のライセンス・プログラム・コード (ppp) で始まる総称識別コードを持つすべてのメッセージを監視します。例えば、`CPF0000` は `CPF` で始まるすべての通知メッセージ、状況メッセージ、およびエスケープ・メッセージが監視することを示します。

  注**:** システム全体の導入、保管、または復元などのシステム機能の実行時には、重要な情報が失われる恐れがあるので、`MONMSG CPF0000` を使用しないでください。

- CPF9999

  機能チェック・メッセージを監視します。エラー・メッセージが監視されていない場合には、それは CPF9999 (機能チェック) となります。

注**:** 監視が行われているときには、一般に、通知メッセージまたは状況メッセージが送られた時点で、ユーザーの監視機能に制御権が渡されます。

メッセージ識別コードによるエスケープ・メッセージ・モニターに加えて、**MONMSG** コマンドを指定した文字ストリングと、そのメッセージのメッセージ・データと比較することもできます。例えば次のようなコマンドで指定すると、ファイル MYFILE に関するエスケープ・メッセージ (CPF5101) が監視されます。このファイルの名前は、プログラムが CPF5101 メッセージを送信した場合のメッセージ・データとして指定されています。

```
MONMSG MSGID(CPF5101) CMPDTA(MYFILE) EXEC(GOTO EOJ)
```

比較に使用できる文字数は 28 までで、比較はメッセージ・データの最初のフィールドの最初の文字から始められます。比較データとメッセージ・データとが一致すると、EXEC パラメーターに指定されている処置が行われます。

MONMSG コマンドの EXEC パラメーターには、そのエスケープ・メッセージに対する処置を指定します。EXEC パラメーターには、PGM、ENDPGM、IF、ELSE、 DCL、DCLF、ENDDO、および MONMSG

を除く任意のコマンドを指定することができます。 EXEC パラメーターには DO コマンドを指定することができ、その DO グループ内のコマンドが実行されます。コマンドまたは DO グループ (EXEC パラメーターに指定した) が実行されると、ユーザーのプロシージャーまたはプログラム内の、そのエスケープ・メッセージを送ったコマンドの次のコマンドに制御権が戻されます。ただし、GOTO コマンドまたは RETURN コマンドを指定した場合には、制御権は返されません。 EXEC パラメーターを指定しない場合には、エスケープ・メッセージは無視され、プログラムまたはプロシージャーが続行されます。

以下の例では、**変数変更 (CHGVAR)** コマンドについて、除算のエスケープ・メッセージ (メッセージ ID MCH1211) が監視されます。

```
CHGVAR VAR(&A) VALUE(&A / &B)
MONMSG MSGID(MCH1211) EXEC(CHGVAR VAR(&A) VALUE(1))
```

変数 &A の値は、&A を &B で割った値に変更されます。 &B が 0 の場合は、除算が実行できず、 0 除算のエスケープ・メッセージがプロシージャーに送られます。 この場合、&A の値は 1 に変更されます (EXEC パラメーターで指定されているとおり)。 &B がゼロであるかどうかをテストし、ゼロでない場合にのみ除算を実行することもできます。 これは、エスケープ・メッセージの操作と監視を試みるよりも効果的です。

以下の例では、プロシージャーは、**オブジェクト検査 (CHKOBJ)** コマンドによるエスケープ・メッセージ CPF9801 (オブジェクトが見つからない) を監視しています。

```
          PGM
          CHKOBJ LIB1/PGMA *PGM
          MONMSG MSGID(CPF9801) EXEC(GOTO NOTFOUND)
          CALL LIB1/PGMA
          RETURN
NOTFOUND: CALL FIX001 /* PGMA Not Found Routine */
          ENDPGM
```

以下の CL プロシージャーには、2 つの**プログラム呼び出し (CALL)** コマンドと、CPF0001 を監視するプロシージャー・レベルの MONMSG コマンドが含まれています。このエスケープ・メッセージは、**プログラム呼び出し (CALL)** コマンドが正常に完了しなかった場合に出されます。どちらかの CALL コマンドが正しく実行されなかった場合には、プロシージャーは完了メッセージを送って終了します。

```
        PGM
        MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
        CALL PROGA
        CALL PROGB
        RETURN
ERROR:  SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*COMP)
        ENDPGM
```

プロシージャー・レベルの MONMSG コマンドに EXEC パラメーターの指定がない場合には、その MONMSG コマンドの対象となるエスケープ・メッセージはすべて無視されます。つまり、IF コマンド条件以外のコマンドで該当のエスケープ・メッセージが生じた場合には、そのエスケープ・メッセージが出されなかったとすれば次に実行するはずのコマンドに移り、プロシージャーの処理を続行します。 IF コマンド条件でエスケープ・メッセージが生じた場合には、プロシージャーはその IF コマンドの条件が偽であるものとして、処理を続行します。以下の例は、プロシージャー内のいくつかの異なる個所でエスケープ・メッセージが生じた場合に、それぞれどのような処置がとられるかを示しています。

```
PGM
DCL &A TYPE(*DEC) LEN(5 0)
DCL &B TYPE(*DEC) LEN(5 0)
MONMSG MSGID(CPF0001 MCH1211)
CALL PGMA PARM(&A &B)
```

```
IF (&A/&B *EQ 5) THEN(CALL PGMB)
ELSE CALL PGMC
CALL PGMD
ENDPGM
```

エスケープ・メッセージがどこで生じるかに応じて、それぞれ以下のような処置がとられます。

- PGMA の呼び出しの時点で CPF0001 が出された場合には、プロシージャーは IF コマンドに移って処理を再開します。
- IF コマンドで MCH1211 (0 除算) が出された場合には、その IF 条件は偽であると見なされ、プロシージャーは PGMC の呼び出しに移って処理を再開します。
- PGMB または PGMC の呼び出しの時点で CPF0001 が出された場合には、プロシージャーは PGMD の呼び出しに移って処理を再開します。
- PGMD の呼び出しの時点で CPF0001 が出された場合には、プロシージャーは ENDPGM コマンドに移って処理を再開し、その結果呼び出しプロシージャーへ制御権が戻されます。

プロシージャーまたはプログラム中の特定のコマンド、およびその他の任意のコマンドから出される同じエスケープ・メッセージを同時に監視することもできます。これには 2 つの **MONMSG** コマンドが必要です。1 つの **MONMSG** コマンドは、そのエスケープ・メッセージに対して特別の処置を必要とするコマンドの次に指定します。そのコマンドでエスケープ・メッセージが出された場合には、この **MONMSG** コマンドが使用されます。もう 1 つの **MONMSG** コマンドは最後の宣言コマンドの次に指定します。これにより、特定のコマンド以外のすべてのコマンドについては、この **MONMSG** コマンドが使用されます。

MONMSG コマンドは、コード化されている CL プロシージャーまたは OPM プログラムだけに適用されます。あるプロシージャーからの MONMSG コマンドは別のプロシージャーには適用されません。このことは両方とも同じプログラムの一部である場合でも当てはまります。各コマンドのオンライン・ヘルプおよび文書には、コマンドに対して発行されるエスケープ、通知、および状況メッセージのリストが記載されています。ユーザーが定義したすべてのメッセージのリストも保管しておいてください。

注**:** 前の段落は、ILE のプロシージャーには当てはまりません。これは、メッセージがパーコレートを行う方法が異なるからです。 システムは、プロシージャーに送信されたすべてのエスケープ・メッセージを MONMSG が処理することを必要とします。このことが行われないとメッセージは、それを処理する **MONMSG** があるプロシージャーまたは制御境界となるプロシージャーを見つけるまで、呼び出しスタックをパーコレートします。

関連概念:

571 ページの『エスケープ・メッセージおよび通知メッセージ』
エスケープ・メッセージおよび通知メッセージはどちらも、CL プログラムまたはプロシージャーからメッセージ待ち行列に送信することができます。

関連タスク:

542 ページの『メッセージ識別コードの割り当て』
メッセージ記述追加 (ADDMSGD) コマンドに指定するメッセージ識別コードは、メッセージの参照に使用されます。メッセージ識別コードは、メッセージ記述の名前です。

メッセージの監視

システムには、メッセージの監視を可能にするイベントの監視機能が用意されています。

イベントの監視機能を使用すると、システムが監視するメッセージを指定できます。これらのメッセージを指定すると、ユーザー出口プログラムが呼び出され、必要なアクションを実行します。メッセージを送信しようとするメッセージ待ち行列またはジョブ・ログを指定する必要があります。メッセージ・データ、送信

元プログラム、または送信先プログラムと比較するテキスト・ストリングを指定できます。メッセージ・タイプまたは重大度で監視を選択することも可能です。この監視選択機能を使用すると、特定のメッセージのみを通知できます。

注: メッセージ監視を開始するときには、注意が必要です。監視通知の非終止ループを作成しないでください。例えば、ジョブ開始メッセージ CPF1124 について *ALL ジョブでメッセージ監視を開始しないでください。

監視開始 (STRWCH) コマンドを使用して、監視セッションを開始し、指定したメッセージが発生した場合に通知させるようにします。指定したメッセージ待ち行列またはログにメッセージの監視済みが追加されると、監視出口プログラムが呼び出されます。監視出口プログラムは、QUSRWRK サブシステムのジョブで実行されます。

監視処理 (WRKWCH) コマンドを使用して、システム上のアクティブな監視のリストが示されたパネルを表示できます。**監視終了 (ENDWCH)** コマンドを使用して、監視セッションを終了することができます。

監視開始 (QSCSWCH) API および 監視終了 (QSCEWCH) API の用法は、STRWCH コマンドおよび ENDWCH コマンドの用法とよく似ています。

監視情報検索 (QSCRWCHI) および監視リスト検索 (QSCRWCHL) の API を使用して、監視に関する情報を入手できます。

また、イベントの監視機能も、次のいくつかのトレース・コマンドに組み込まれます。

- **トレース開始 (STRTRC)**
- **通信トレース開始 (STRCMNTRC)**
- **内部事象トレース (TRCINT)**
- **接続トレース (TRCCNN)**
- **TCP アプリケーション・トレース (TRCTCPAPP)**

監視サポートは、特定のあらかじめ定められた基準に一致する場合、自動的にトレースを監視および終了することにより、トレース機能を強化します。監視サポートを使用すると、有用なトレース・データを失わずに、トレース監視に要する時間を削減することができます。

関連情報:

イベントの監視機能

監視 API

監視開始 (STRWCH) コマンド

監視終了 (ENDWCH) コマンド

監視処理 (WRKWCH) コマンド

監視開始 (QSCSWCH) API

監視終了 (QSCEWCH) API

監視情報検索 (QSCRWCHI) API

監視リスト検索 (QSCRWCHL) API

非監視メッセージの CL 処理

ユーザーが監視を指定しなかったメッセージについては、システムがデフォルト値による監視と処理を行います。

コマンド、プログラム、およびプロシージャーを呼び出すプログラムまたはプロシージャーに、さまざまなエスケープ・メッセージが送られる可能性があります。ユーザーのプログラムまたはプロシージャーの機能に関係するエスケープ・メッセージだけを監視して処理したいという場合もあります。しかしその場合でも、メッセージをすべて監視し、処理するわけではないでしょう。

デフォルトの処置では、プログラムまたはプロシージャーでエラーが起こったものと見なされます。プログラムまたはプロシージャーのデバッグを行っている場合には、そのメッセージはユーザーのディスプレイ装置に送られます。ここで、ユーザーはコマンドを入力して、エラーを分析し、訂正することができます。プログラムまたはプロシージャーのデバッグを行っていない場合は、システムはメッセージ・パーコレーション機能を実行します。

メッセージ・パーコレーションは、2 つのステップからなる機能で、以下のタスクを行います。

- 呼び出しスタック内でエスケープ・メッセージをステップ 1 つ分だけ前に移動する。
- プログラムまたはプロシージャーに、エスケープ・メッセージ用の**メッセージ・モニター (MONMSG)** コマンドがあるかどうかを判別する。

プログラムまたはプロシージャーに、エスケープ・メッセージ用の**メッセージ・モニター (MONMSG)** コマンドがある場合、メッセージ・パーコレーション処理は停止し、システムは **メッセージ・モニター (MONMSG)** コマンドで指定した処置を行います。メッセージ・パーコレーションは **メッセージ・モニター (MONMSG)** コマンドを検出するか、または最も近くの制御境界を検出するまで継続します。つまり、エスケープ・メッセージは制御境界を超えてパーコレートを行うことはありません。

メッセージに適する**メッセージ・モニター (MONMSG)** コマンドを含むプログラムまたはプロシージャーを検出する前に制御境界を検出することによって、機能チェック処理が開始されます。 システムはオリジナルのエスケープ・メッセージに関する処置は完了したと見なします。 その後、システムは機能チェック・メッセージ (CPF9999) を、オリジナルのエスケープの対象であるプログラムまたはプロシージャーに送ります。 機能チェック・メッセージ用の**メッセージ・モニター (MONMSG)** コマンドがこのプログラムまたはプロシージャーにある場合は、このコマンドで指定されている処置を行います。 このプロシージャーに MONMSG が指定されていないと、ジョブが対話式ジョブである場合にシステムはワークステーション・オペレーターに照会メッセージを送ります。 ワークステーション・オペレーターでは、以下のいずれかの応答を返すことができます。

R プログラムまたはプロシージャー中の失敗したコマンドを再試行します。

I メッセージを無視します。プログラムまたはプロシージャー中の次のコマンドで処理を継続します。

C プログラムまたはプロシージャーを取り消して、呼び出しスタック上の直前のプログラムまたはプロシージャーに、機能チェックをパーコレートします。

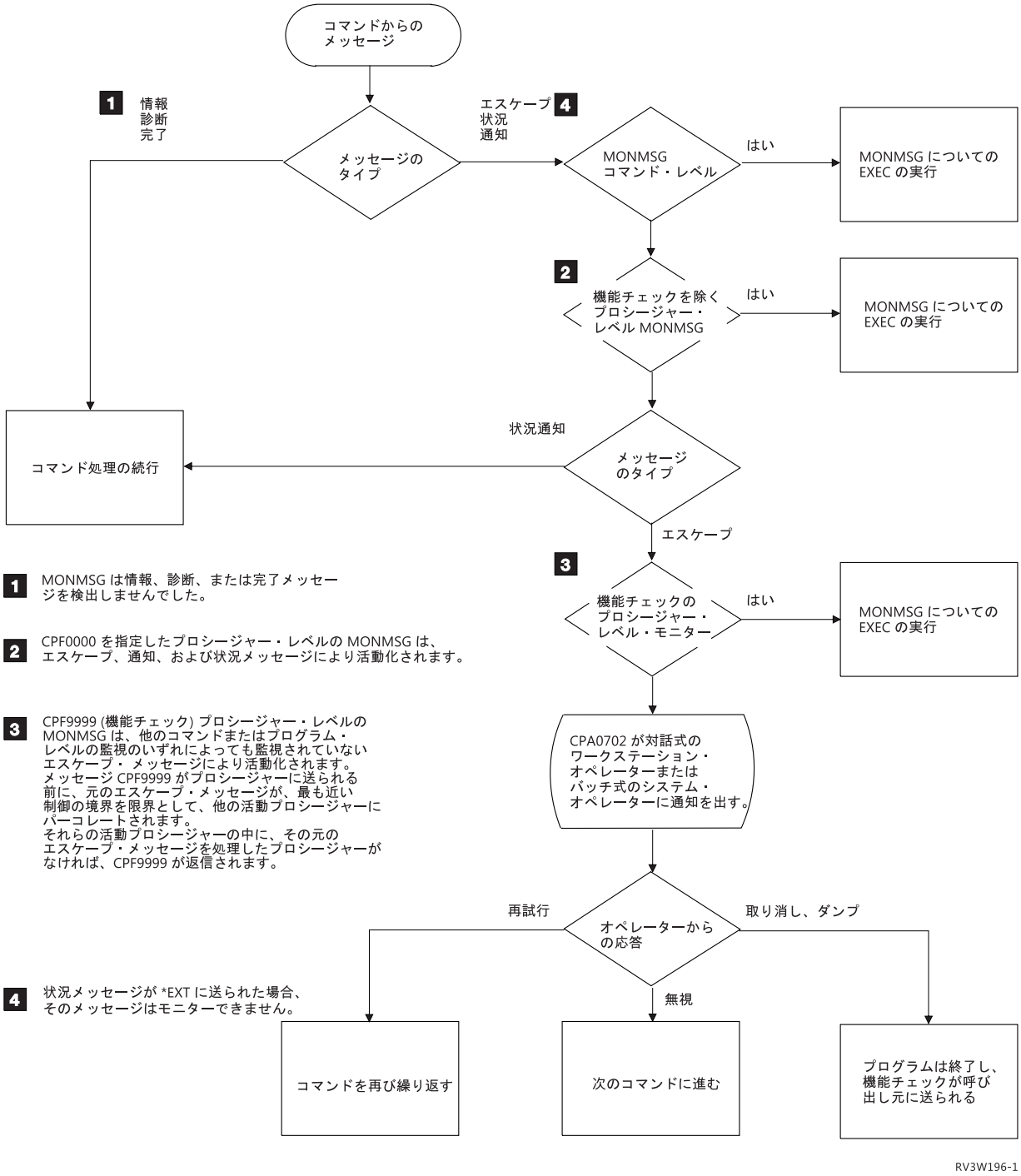
D 失敗したプログラムまたはプロシージャーの呼び出しスタック項目をダンプし、プログラムまたはプロシージャーを取り消し、呼び出しスタック上の直前のプログラムまたはプロシージャーに機能チェックをパーコレートします。 ワークステーション・オペレーターが応答を入力しない場合やジョブがバッチ・ジョブである場合は、これがデフォルトの処置です。

システムは制御境界を超えて機能チェックをパーコレートすることはありません。応答によって、機能チェックが活動化グループ境界を超えて移動するようになる場合、その機能チェックに関する処置はそれ以降停止します。システムは活動化グループ境界に達したプログラムまたはプロシージャーをすべて取り消し、エスケープ・メッセージ CEE9901 を直前の呼び出しスタック項目に送ります。

この機能チェックのエスケープ・メッセージを監視することによって、プログラムまたはプロシージャーをクリーンアップしたり、終了したりすることができます。また、プログラムまたはプロシージャーの処理を続行することもできます。

注: 監視対象となっていないエスケープに関するメッセージ記述でデフォルトの処置が指定されている場合は、デフォルトの処理を行うプログラムが呼び出されてから機能チェック・メッセージが送られます。デフォルトの処理を行うプログラムが戻ると、機能チェック処理が開始されます。

次の図は、CL がメッセージを監視し、デフォルトの処理に対する機能チェックをワークステーション・ユーザーに通知する方法を示しています。



関連タスク:

550 ページの『エスケープ・メッセージに対するデフォルトのメッセージ処理の指定』

エスケープ・メッセージとして送信できるメッセージを作成する場合は、そのメッセージが送信されたときに別の方法で処理されない場合に使用される、デフォルトのメッセージ処理処置をセットアップすることができます。

通知メッセージの監視

ユーザーのプロシージャーまたはプログラム内のコマンドによって、またはそれらが呼び出すプロシージャーまたはプログラムによって、そのユーザーの CL プロシージャーまたはプログラムの呼び出しメッセージ待ち行列に送られる通知メッセージも監視できます。

通知メッセージは、必ずしもエラーとして扱うことができないような状態を、ユーザーのプロシージャーまたはプログラムに伝えるために送られます。通知メッセージを監視することによって、そのような状態が生じなかった場合とは異なる処置を指定することができます。通知メッセージを出す IBM 提供のコマンドはごく少数です。

通知メッセージの監視と処理は、エスケープ・メッセージの監視と処理に類似しています。異なる点は、メッセージの監視と処理を行わなかった場合にどのような処置がとられるかということだけです。通知メッセージは、活動化グループの境界内のプロシージャー間でパーコレートすることもできます。活動化グループの境界に達したけれどもその境界に対する**メッセージ・モニター (MONMSG)** コマンドが検出されない場合には、通知メッセージの送信側にデフォルト応答が自動的に戻され、送信側は処理を続行することができます。エスケープ・メッセージの場合と異なり、監視対象となっていない通知メッセージは、ユーザーのプロシージャーまたはプログラムのエラーを示すものとは見なされません。

状況メッセージの監視

ユーザーの CL プロシージャー中のコマンド、あるいはそのプロシージャーが呼び出したプロシージャーおよびプログラムから送られた状況メッセージは、監視することができます。

状況メッセージは、送信側で行われている処理の状況を、ユーザーのプロシージャーに知らせるためのものです。状況メッセージを監視することによって、送信側のプログラムまたはプロシージャーがそれ以上処理を進めないようにすることができます。

状況メッセージの場合、メッセージ情報はメッセージ待ち行列に保管されません。したがって、状況メッセージを受け取ることはできません。

状況メッセージが監視されていないと、エスケープ・メッセージや通知メッセージの場合と同様にパーコレートされます。活動化グループ境界に達したが**メッセージ監視 (MONMSG)** コマンドが検出されない場合には、そのメッセージに関する処置は完了したと見なされ、メッセージの送信側に制御権が戻されて処理が継続します。状況メッセージは、処理の続行が可能な状態において検出された正常な状況を伝えるために送られることもよくあります。

状況メッセージは外部メッセージ待ち行列に送られ、対話式に画面に表示され、ある機能による処理が進行中であることをユーザーに知らせます。例えば、**ファイル・コピー (CPYF)** コマンドの実行中には、複写操作が進行中であることを知らせるメッセージが出されます。

状況メッセージとして送ることができるのはメッセージ・ファイルに含まれているメッセージだけであり、即時メッセージを状況メッセージとして送ることはできません。ユーザーは、システム提供のメッセージ ID である CPF9898 を使用することができます。また既存のメッセージ記述がない場合には、メッセージ・データを提供して状況メッセージを送ることができます。

機能が完了すると、プロシージャーまたはプログラムは、状況メッセージを対話式画面から除去します。コマンドを使用してメッセージを除去することはできませんが、ブランクのメッセージを別の状況メッセージとして *EXT に送ると、そのメッセージは除去されたようになります。この目的に、システム提供のメッセージ ID CPI9801 を使用することができます。 IBM i プログラムに制御権が戻された時点で、CPI9801 メッセージを送ることなく *STATUS メッセージが 24 行目から消去されます。以下の例は、メッセージ ID CPF9898 および CPI9801 の一般的な用法を示しています。

```
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) +
          MSGDTA('Function xxx being performed') +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)
    •
    •   /*  Your processing function  */
    •
SNDPGMMSG MSGID(CPI9801) MSGF(QCPFMSG) +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)
```

状況メッセージの表示の抑制

コマンドから状況メッセージが送られないようにすることはできませんが、状況メッセージが画面の最下行に表示されないようにすることはできます。

状況メッセージが表示されないようにするには、以下の 2 つが望ましい方法です。

- **ユーザー・プロファイル変更 (CHGUSRPRF)** コマンド

  ユーザー・プロファイルを変更することによって、そのプロファイルを使用してサインオンした場合に状況メッセージが表示されないようにすることができます。これを行うには、CHGUSRPRF コマンドを使用し、ユーザー・オプション (USROPT) パラメーターに *NOSTSMSG を指定します。

- **ジョブ変更 (CHGJOB)** コマンド

  現在実行中のジョブを変更して、状況メッセージが表示されないようにすることができます。これを行うには、CHGJOB コマンドを使用し、状況メッセージ (STSMSG) パラメーターに *NONE を指定します。 CHGJOB コマンドの STSMSG パラメーターに *NORMAL を指定すれば、状況メッセージを表示することができます。

第 3 の代替方法として、**メッセージ・ファイル一時変更 (OVRMSGF)** コマンドを使用して、状況メッセージ ID をブランクのメッセージに変更することもできますが、これはあまり良い方法ではありません。

終了したプログラムまたはプロシージャーからメッセージを受信する

アクティブでない呼び出しメッセージ待ち行列に送信されたジョブ・ログからメッセージを受信する必要がある場合があります。

例えば、PGMA は PGMB を呼び出し、PGMB は、診断メッセージを自らに送信し、エスケープ・メッセージを PGMA に送信します。PGMA は、PGMA に送信されたエスケープ・メッセージを容易に受信できますが、場合によっては、診断メッセージも受信する必要があります。診断は、アクティブでない呼び出しメッセージ待ち行列にあるため、メッセージは、メッセージ参照キーを使用して受信する必要があります。このことは、実行可能ですが、1 つの単純なメッセージ受信コマンドを使用する場合より多くの作業を必要とします。

この実行方法を理解する際に、以下の考慮事項は重要です。

- **メッセージ受信 (RCVMSG)** コマンド (または QMHRCVPM API) がコード化され、自らの呼び出しメッセージ待ち行列からキーによりメッセージを受信する場合、メッセージの送信先の呼び出しメッセージ待ち行列が何であるか、またはプロシージャーがアクティブであるかに関係なく、そのキー (存在する場合) を使用してジョブ・ログからメッセージを受信します。
- ジョブ・ログのメッセージのメッセージ参照キーは、(外部的に 4 文字の値として文書化されている場合であっても) 4 バイト符号なし整数 (送信メッセージごとに増分される) として扱うことができます。

終了したプログラムまたはプロシージャーからメッセージを取得するには、以下のステップを実行してください。

1. メッセージを送信 (オプションでメッセージを削除) して、開始メッセージ参照キーを取得する。

2. 対象のメッセージを送信する機能を実行する。
3. 別のメッセージを送信して、終了メッセージ参照キーを取得する。
4. ループでは、ステップ 1 および 3 で識別された開始および終了メッセージ・キー間のメッセージ参照キーを増分し、各キーのメッセージを受信します。メッセージ参照キーには、ギャップが発生することがあるため、メッセージ受信機能の CPF2410 例外を処理する必要があります。ギャップは、削除されたメッセージ、または内部例外処理から発生することがあります。

注: このトピックでは、終了したプログラムまたはプロシージャーからメッセージを受信する 1 つの方法の手順と例を説明します。ジョブ・ログの一部またはすべてのメッセージへのアクセスに使用できる手法は他にもあります。これらは、以下のリストで要約されていますが、詳細については説明されておらず、サンプル・プログラムも提供されていません。

- ジョブ・ログを出力ファイルに書き込む。**ジョブ・ログ表示 (DSPJOBLOG)** コマンドで OUTPUT(*OUTFILE) を指定したり、QMHCTLJL API を使用してジョブ・ログの出力ファイル指定を定義してジョブを終了したり、または**ジョブ・ログ表示 (DSPJOBLOG)** コマンドで OUTPUT(*APIDFN) を指定したりすることによって、これを実行することができます。これを実行した場合、ジョブ・ログのすべてのメッセージは、出力ファイルに書き込まれます。このため、プログラムまたはプロシージャーの終了時に呼び出しメッセージ待ち行列に残されたメッセージが含まれます。
- ジョブ・ログ・メッセージをリストする (QMHLJOBL) API を使用して、ジョブ・ログからユーザー・スペースにメッセージをコピーします。ジョブ・ログまたはサブセットのすべてのメッセージを選択し、各メッセージに対してどのフィールドが戻されるかを選択することができます。ユーザー・スペースの内容へのアクセスに使用できる API があります。QUSPTRUS API は、ユーザー・スペースのデータへのポインターの取得に使用できます。その後、ポインターをサポートする言語でポインターを使用できます。API QUSRTVUS を使用して、ユーザー・スペースからデータを取得し、ポインターをサポートしない言語を使用することができます。

前に説明したプロシージャーのサンプル・プログラムを以下に示します。このプログラムは、RCVMSG コマンドおよび QMHRCVPM API を使用しています。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
*****************************************************************************
 *
 * RCVMSG コマンドを使用した CL プログラム例
 *
 *
 *****************************************************************************

 このプログラム例の利点は、簡単にコード化できるということで、理解も容易です。
 一方、このプログラムの欠点は、プログラムが CPF2410 例外のモニターおよび処理を行わなくてはならないという
 ことであり、これはパフォーマンスに重点を置く場合には好ましくありません。


PGM
DCL &LOWKEY *CHAR 4
DCL &HIKEY *CHAR 4
DCL &MSGKEY *CHAR 4
DCL &MSG *CHAR 256
/*-------------------------------------------------------------*/
/* OBTAIN STARTING MESSAGE REFERENCE KEY                       */
/*-------------------------------------------------------------*/
SNDPGMMSG  MSG(TEST) TOPGMQ(*SAME) KEYVAR(&LOWKEY)
RMVMSG     MSGKEY(&LOWKEY)
/*-------------------------------------------------------------*/
/* EXECUTE FUNCTION                                            */
/*-------------------------------------------------------------*/
```

```
  ---- Insert whatever command(s) or program call(s) you want ----
  ---- to handle messages for here                              ----

/*-------------------------------------------------------------------*/
/* OBTAIN ENDING MESSAGE REFERENCE KEY                               */
/*-------------------------------------------------------------------*/
SNDPGMMSG  MSG(TEST) TOPGMQ(*SAME) KEYVAR(&HIKEY)
RMVMSG     MSGKEY(&HIKEY)
/*-------------------------------------------------------------------*/
/* LOOP TO RECEIVE MESSAGES ON INACTIVE INVOCATION                   */
/*-------------------------------------------------------------------*/
CHGVAR %BIN(&MSGKEY 1 4)  (%BIN(&LOWKEY 1 4) + 1)
LOOP:
RCVMSG     PGMQ(*SAME (*)) MSGKEY(&MSGKEY) RMV(*NO) MSG(&MSG)
MONMSG CPF2410 EXEC(DO) /* HANDLE MSGKEY NOT FOUND                  */
  RCVMSG     MSGTYPE(*EXCP) RMV(*YES) /* REMOVE UNWANTED EXCEPTION */
  GOTO       SKIP
ENDDO

    ---- Insert code here to do whatever processing is needed  ----
    ---- for the message.  You may need to add additional      ----
    ---- values, such as message ID, message type, etc., to    ----
    ---- the RCVMSG command.                                   ----

SKIP:
CHGVAR %BIN(&MSGKEY 1 4)  (%BIN(&MSGKEY 1 4) + 1)
IF (&MSGKEY *LT &HIKEY) GOTO LOOP
ENDPGM
```

```
 ******************************************************************************
 *
 * QMHRCVPM API を使用した CL プログラム
 *
 ******************************************************************************
```

このサンプルは、最初のサンプルと類似しています。 RCVMSG CL コマンドではなく、QMHRCVPM API を使用するという点のみ異なります。 エラー・コード構成を使用することにより、CPF2410 例外を処理する必要がなく、またメッセージ・キーが検出されない場合に例外を送信するために必要なオーバーヘッドが除去されます。 この例では、API によって戻される構成からメッセージ・テキストを抽出する方法を示します。メッセージが即時メッセージ (つまり、メッセージ ID ではない) 場合、メッセージは「置換データまたは即時メッセージ・テキスト (Replacement data or impromptu message text)」領域にあります。ここにない場合は、「置換データ (Replacement data)」フィールドに続く「メッセージ (Message)」領域にあります。このプログラム例はメッセージ長を検査して、これらのフィールドからメッセージに使用するフィールドを決定します。

```
PGM
DCL &LOWKEY *CHAR 4
DCL &HIKEY *CHAR 4
DCL &MSGKEY *CHAR 4
DCL &MSG *CHAR 256
/*-------------------------------------------------------------------*/
/* Some messages have a large amount of message data, in which case */
/* the size of the &MSGINFO variable will not be adequate.  If you  */
/* expect to receive messages with a large amount of message data,  */
/* you will need to increase the size of this variable accordingly. */
/* Be sure to also change the value that is put into &MSGINFOL to   */
/* reflect the size of the variable.                                 */
/*-------------------------------------------------------------------*/
DCL &MSGINFO *CHAR 1000
DCL &MSGINFOL *CHAR 4
DCL &ERRCODE *CHAR 16
DCL &MSGOFFS *DEC (4 0)
DCL &MSGLEN *DEC (4 0)
/*-------------------------------------------------------------------*/
/* OBTAIN STARTING MESSAGE REFERENCE KEY                             */
```

```
/*-------------------------------------------------------------------*/
SNDPGMMSG  MSG(TEST) TOPGMQ(*SAME) KEYVAR(&LOWKEY)
RMVMSG     MSGKEY(&LOWKEY)
/*-------------------------------------------------------------------*/
/* EXECUTE FUNCTION                                                  */
/*-------------------------------------------------------------------*/

  ---- Insert whatever command(s) or program call(s) you want ----
  ---- to handle messages for here                            ----

/*-------------------------------------------------------------------*/
/* OBTAIN ENDING MESSAGE REFERENCE KEY                               */
/*-------------------------------------------------------------------*/
SNDPGMMSG  MSG(TEST) TOPGMQ(*SAME) KEYVAR(&HIKEY)
RMVMSG     MSGKEY(&HIKEY)
/*-------------------------------------------------------------------*/
/* LOOP TO RECEIVE MESSAGES WITH QMHRCVPM API                        */
/*-------------------------------------------------------------------*/
CHGVAR %BIN(&MSGKEY 1 4)  (%BIN(&LOWKEY 1 4) + 1)
CHGVAR %BIN(&MSGINFOL 1 4) 1000
CHGVAR %BIN(&ERRCODE 1 4) 16
LOOP2:
CALL QMHRCVPM (&MSGINFO &MSGINFOL RCVM0200 '*          ' +
               X'00000000' '*ANY      ' &MSGKEY X'00000000' +
               '*SAME     ' &ERRCODE)
IF ((%BIN(&MSGINFO 5 4) *GT 0) *AND (%BIN(&ERRCODE 5 4) *EQ 0)) +
  DO /* IF A MESSAGE WAS RECEIVED */
    IF (%BIN(&MSGINFO 161 4) *EQ 0) +
      DO /* IMPROMPTU MESSAGE */
        CHGVAR &MSGLEN %BIN(&MSGINFO 153 4)
        CHGVAR &MSGOFFS 177
      ENDDO
    ELSE +
      DO /* STORED MESSAGE */
        CHGVAR &MSGLEN %BIN(&MSGINFO 161 4)
        CHGVAR &MSGOFFS (177 + %BIN(&MSGINFO 153 4))
      ENDDO
    CHGVAR &MSG %SST(&MSGINFO &MSGOFFS &MSGLEN)

    ---- Insert code here to do whatever processing is needed  ----
    ---- for the message.  You can extract additional          ----
    ---- values, such as message ID, message type, etc.,       ----
    ---- from the message information structure if necessary.  ----

  ENDDO
CHGVAR %BIN(&MSGKEY 1 4)  (%BIN(&MSGKEY 1 4) + 1)
IF (&MSGKEY *LT &HIKEY) GOTO LOOP2
ENDPGM
```

関連概念:

565 ページの『呼び出しメッセージ待ち行列』
呼び出しメッセージ待ち行列を使用すると、あるプログラムまたはプロシージャーと別のプログラムまたはプロシージャーの間で、メッセージを送信することができます。

中断処理プログラム

中断処理プログラムは、*BREAK モードになっているメッセージ待ち行列にメッセージが到着した時に自動的に呼び出されるプログラムです。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

同じ**メッセージ待ち行列変更 (CHGMSGQ)** コマンドで、このプログラムの名前と中断転送名の両方が指定されていなければなりません。 **CHGMSGQ** コマンドではプログラムを指定しますが、そのプログラム中の 1 つまたは複数のプロシージャーがメッセージを処理します。このプログラム中のプロシージャーは、 **メッセージ受信 (RCVMSG)** コマンドを使用してメッセージを受け取らなければなりません。 中断転送に対するメッセージを処理するために呼び出されたユーザー定義のプログラムにパラメーターが渡されて、メッセージの受け取りと処理が行われます (厳密には、プログラム中で最初に実行するプロシージャーにこのパラメーターが渡されます)。パラメーターは、中断の原因となるメッセージのメッセージ待ち行列およびメッセージ参照キー (MRK) を識別します。中断処理プログラムが呼び出されると、中断モードのメッセージ待ち行列があるジョブに割り込み、実行されます。中断処理プログラムに渡されるパラメーターに関する詳細は、Break Handling Exit Program API に関する情報を参照してください。中断処理プログラムが終了すると、中断されたプログラムが処理を再開します。

以下のプログラム (PGMA) は中断処理の例で、1 つのプロシージャーだけで構成されています。

```
PGM PARM(&MSGQ &MSGLIB &MRK)
DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&MRK) TYPE(*CHAR) LEN(4)
DCL VAR(&MSG) TYPE(*CHAR) LEN(75)
RCVMSG  MSGQ(&MSGLIB/&MSGQ) MSGKEY(&MRK) +
        MSG(&MSG)
.
.
.
ENDPGM
```

中断処理プログラムを作成した後に、以下のようなコマンドを実行すればそのプログラムが QSYSMSG メッセージ待ち行列に結び付けられます。

```
CHGMSGQ   MSGQ(QSYS/QSYSMSG) DLVRY(*BREAK) PGM(PGMA)
```

注**:**

1. メッセージは、処理の際にメッセージ待ち行列から除去してください。メッセージ待ち行列が中断モードに置かれると、待ち行列上の任意のメッセージは、中断処理プログラムを呼び出すようになります。CHGMSGQ 時、中断プログラムは、メッセージ重大度基準を満たすメッセージがいくつ見つかるかに関わらず、待ち行列のメッセージ重大度の基準を満たすメッセージが見つかった場合一度のみ呼び出されるので、中断処理プログラムは待ち行列のメッセージをすべて処理する必要があります。CHGMSGQ 後、中断プログラムはメッセージ重大度基準を満たす各メッセージに対して呼び出されます。
2. メッセージを受け取るプロシージャーまたはプログラムをゼロ以外の待機時間でコード化して、メッセージを受け取るようにすることはできません。 **RCVMSG** コマンドには、ゼロ以外の値の待機パラメーターを指定してください。ジョブが中断処理イベントを実行している間は、システムがメッセージ到着イベントを処理することはできません。
3. メッセージ待ち行列の重大度は、送信されるメッセージの重大度がメッセージ待ち行列の重大度以上の場合のみ中断処理プログラムを呼び出す必要があることを示すように設定することができます。

中断処理プログラムの例は、プログラムがメッセージを送るようにするためのものです。このメッセージは、通常は QSYSOPR 待ち行列、QSYSOPR 以外の待ち行列、または QSYSOPR と、その他の待ち行列の両方に送られます。

中断メッセージを処理するユーザー定義のプログラムの例を次に示します (この例も 1 つのプロシージャーだけで構成されています)。このプログラムが使用されると、ディスプレイ装置ユーザーは、メッセージ

CPA5243 (印刷装置 &1 で READY、START、または START/STOP を押してください) および CPA5316 (印刷装置 &3 の位置合わせを確認してください) に応答する必要がありません。

```
BRKPGM:     PGM (&MSGQ &MSGQLIB &MSGMRK)
            DCL &MSGQ TYPE(*CHAR) LEN(10)
            DCL &MSGQLIB TYPE(*CHAR) LEN(10)
            DCL &MSGMRK TYPE(*CHAR) LEN(4)
            DCL &MSGID TYPE(*CHAR) LEN(7)
            RCVMSG MSGQ(&MSGQLIB/&MSGQ) MSGKEY(&MSGMRK) +
                   MSGID(&MSGID) RMV(*NO)
            /* Ignore message CPA5243 */
            IF (&MSGID *EQ 'CPA5243') GOTO ENDBRKPGM
            /* Reply to forms alignment message */
            IF (&MSGID *EQ 'CPA5316') +
                   DO
                   SNDRPY MSGKEY(&MSGMRK) MSGQ(&MSGQLIB/&MSGQ) RPY(I)
                   ENDDO
            /* Other messages require user intervention */
            ELSE CMD(DSPMSG MSGQ(&MSGQLIB/&MSGQ))
ENDBRKPGM:  ENDPGM
```

重要**:**

中断処理プログラムの以前の例で、CPA5316 メッセージが **メッセージの表示 (DSPMSG)** コマンドの実行中に待ち行列に到着する必要がある場合、DSPMSG 画面は中断メッセージおよび CPA5316 メッセージの原因となったもとのメッセージを示します。 DSPMSG 画面は、処理を進める前に、オペレーターが CPA5316 のメッセージに応答するのを待ちます。

注**:** このプログラムは、割り込まれたプログラムが画面からの入力データを待っている場合、表示装置ファイルをオープンすることはできません。

システム応答リストを使用して、システムが事前定義照会メッセージに対する応答を出すように指定することができます。その場合、ディスプレイ装置ユーザーは応答する必要はありません。

メッセージ処理機能の実行中に表示の抑止と復元が確実に行われるようにするには、ユーザー中断処理プログラム中のプロシージャーに抑止および復元プロシージャーが必要です。中断および復元プロシージャーは、以下の条件が存在する場合に限り必要です。

- 中断プログラム中のプロシージャーが他のメニューや画面を表示する場合。
- 中断プログラムが、他のメニューや画面を表示する可能性のある他のプログラムを呼び出す場合。

以下の例には、表示の抑止と復元に必要なユーザーのプロシージャーおよび表示装置ファイルを示してあります。

注**:** 表示装置ファイルを作成するには RSTDSP(*YES) を指定しなければなりません。

```
      A          R SAVFMT                    OVERLAY  KEEP
      A*
      A          R DUMMY                     OVERLAY
      A                                      KEEP
      A                                      ASSUME
      A            DUMMYR         1A     1  2DSPATR(ND)


   PGM PARM(&MSGQ &MSGLIB &MRK)
       DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
       DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
       DCL VAR(&MRK) TYPE(*DEC) LEN(4)
       DCLF FILE(UDDS/BRKPGMFM)
```

```
      SNDF RCDFMT(SAVFMT)
      CALL PGM(User's Break Program)
      SNDF RCDFMT(SAVFMT)
   ENDPGM
```

ユーザー指定の中断処理プログラムが対話式ジョブに割り込まないようにするために、そのプログラムを投入してバッチで実行することもできます。メッセージを受け取ってから**ジョブ投入 (SBMJOB)** を実行する中断処理プログラムを指定すると、この操作を行えます。 SBMJOB は、使用したい任意のパラメーターを含む現行の中断処理プログラムに対する呼び出しを実行します。 (例として、受信メッセージの情報があります。) それから制御権が対話式ジョブに戻り、操作が通常どおり継続します。

関連概念:

562 ページの『中断モードにあるメッセージ待ち行列』
中断処理プログラムは、中断転送モードにあるメッセージ待ち行列に着信したメッセージの重大度コードが、重大度コード・フィルターと同じかそれよりも高い場合は、いつでも呼び出すことができます。

関連タスク:

615 ページの『システム応答リストの使用』
システム応答リストを使用することによって、特定の事前定義照会メッセージに対してシステムが自動的に応答するように指定することができます。

関連情報:

Break Handling Exit Program API

照会メッセージに対する応答の処理方法

応答の処理には、応答を取得するための送信側コピー・メッセージの使用、応答を送信するジョブの検索、システム応答リストの使用、および応答処理出口プログラムの使用が含まれます。

送信側コピー・メッセージを使用して応答を取得する

照会メッセージが送信されると、応答が要求されます。照会メッセージの送信者が応答を取得できるようにするため、送信側コピー・メッセージが発行され、内部的に照会メッセージと関連付けられます。

送信側コピー・メッセージは、照会メッセージのコピーであり、送信者はこれを使用して、関連付けられた照会メッセージに送信された応答を取得します。照会の送信および応答の取得は、ソース・プログラム内で**ユーザー・メッセージ送信 (SNDUSRMSG)** コマンドを使用すると、かなり容易に実行できます。 **SNDUSRMSG** コマンドが使用されていない場合、照会を送信し応答を取得する機能を、CL ソース・プログラム内の**プログラム・メッセージ送信 (SNDPGMMSG)** および**メッセージ受信 (RCVMSG)** コマンドにより実行できます。照会は、**SNDPGMMSG** コマンドを使用して送信でき、応答は、**RCVMSG** コマンドを使用して取得できます。例えば、照会メッセージが **SNDPGMMSG** コマンドを使用して送信された場合、送信側コピーのメッセージ参照キー (MRK) を **SNDPGMMSG** コマンドの KEYVAR パラメーターに戻すことができます。送信側コピー・メッセージは、**SNDPGMMSG** コマンドで指定された応答メッセージ待ち行列に配置されます。応答が照会メッセージに送信される場合、内部メッセージ処理により、同じ応答が送信側コピー・メッセージにも送信されます。次に、プログラムは以下の作業を実行することで、**RCVMSG** コマンドにより応答を取得できます。

- **SNDPGMMSG** の KEYVAR パラメーターからの送信側コピーの MRK を使用して、**RCVMSG** コマンドの MSGKEY パラメーターで指定する。
- 応答のメッセージ・タイプを指定する。
- 照会メッセージに送信する応答に許可する待機時間を指定する。

関連タスク:

548 ページの『即時メッセージの送信および応答の処理』
この例は、プログラムまたはプロシージャーがどのようにして照会メッセージを送信し、その応答を処理す

るのかを示しています。

関連資料:

572 ページの『例: メッセージ送信』
以下の例は、さまざまなメッセージの送信方法を示しています。

応答を送信したジョブの検出

各種標準メッセージ待ち行列内のある照会メッセージに応答したジョブを知る必要がある場合があります。

この情報は、**メッセージの表示 (DSPMSG)** では取得できません。その理由は、応答は照会メッセージ (または送信者コピー) に関連付けられているので、応答の詳細を表示したときに表示される情報は、応答ではなく、照会メッセージに関連するものだからです。応答の送信側は、DSPMSG msgqname OUTPUT(*PRINT) を実行すると表示することができます。生成されたスプール・ファイルには、応答メッセージの送信ジョブが含まれています。

当該のメッセージ待ち行列が QSYSOPR の場合は、別の方法でこの情報を取得することができます。DSPLOG が照会メッセージの時刻近くの期間で指定されている場合は、応答を見つけ、その応答メッセージの上にカーソルを置き、ヘルプを押します。それから F9 を押して詳細を表示し、応答を送信したジョブを確認します。

システム応答リストの使用

システム応答リストを使用することによって、特定の事前定義照会メッセージに対してシステムが自動的に応答するように指定することができます。

照会メッセージのみが自動的にシステム応答リストに応答できます。

システム応答リストには、メッセージ ID、照会メッセージのメッセージ・データに一致する必要のあるオプションの比較データ、各メッセージに対する応答値、および DSPJOB OUTPUT(*PRINT) を効果的に実行するダンプ属性が含まれています。システム応答リストは、システム応答リストを使用しているジョブから送られた事前定義照会メッセージに対してだけ適用されます。ジョブで照会メッセージに対してシステム応答リストを使用するように指定するためには、以下のコマンドで INQMSGRPY(*SYSRPYL) パラメーターを指定します。

- **バッチ・ジョブ (BCHJOB)**
- **ジョブ投入 (SBMJOB)**
- **ジョブ変更 (CHGJOB)**
- **ジョブ記述作成 (CRTJOBD)**
- **ジョブ記述変更 (CHGJOBD)**

システム応答リストを使用しているジョブから事前定義照会メッセージが送られると、システムは順序番号の昇順に従って応答リストを走査し、そのメッセージのメッセージ ID、および必要があれば照会メッセージの比較データと比較して、一致する項目を検索します。該当する項目が見つかった場合には、そこに指定されている応答が返されるので、ユーザーは応答を入力する必要はありません。該当する項目が見つからなかった場合、そのメッセージは、ディスプレイ装置ユーザー (対話式ジョブの場合) またはシステム・オペレーター (バッチ・ジョブの場合) に送られます。

システムの出荷時点では、システム応答リストには以下のような項目が含まれています。

順序番号	メッセージ **ID**	比較値	応答	**Dump**
10	CPA0700	*NONE	D	*YES

順序番号	メッセージ **ID**	比較値	応答	**Dump**
20	RPG0000	*NONE	D	*YES
30	CBE0000	*NONE	D	*YES
40	PLI0000	*NONE	D	*YES

上記の項目は、応答リストを使用しているジョブからメッセージ CPA0700 から CPA0799、RPG0000 から RPG9999、CBE0000 から CBE9999、または PLI0000 から PLI9999 (これらはプログラムの失敗を示す) が送られた場合に応答として D が送り返され、ジョブ・ダンプがとられることを示しています。システムにこれらの項目を使用させるためには、照会メッセージ応答ジョブ属性を *SYSRPYL に設定して、ジョブでシステム応答リストが使用されるように指定しておく必要があります。

上記以外の照会メッセージをシステム応答リストに追加したい場合には、 **システム応答リスト項目追加 (ADDRPYLE)** コマンドを使用してください。このコマンドでは、順序番号、メッセージ識別コード、オプションの比較データ、比較データ CCSID、応答処置、およびダンプ属性を指定することができます。ADDRPYLE コマンドの機能は、 **システム応答リスト項目処理 (WRKRPYLE)** コマンドを使用して容易にアクセスすることができます。

システム応答リストに指定する照会メッセージに対して、次のような応答処置を指定することができます (括弧内はパラメーター値です)。

- 照会メッセージに対してデフォルトの応答を送る (*DFT)。この場合には、メッセージに対するデフォルトの応答が送られます。メッセージ ID がデフォルトの応答値を識別しない場合、*N が応答として送信されます。そのメッセージは表示されず、デフォルトの処理プログラムも呼び出されません。
- ワークステーション・ユーザーまたはシステム・オペレーターによるメッセージへの応答が必要 (*RQD)。メッセージが送られるメッセージ待ち行列 (対話式ジョブの場合はワークステーション・メッセージ待ち行列、バッチ・ジョブの場合は QSYSOPR) が中断モードになっている場合はそのメッセージが表示され、ワークステーション・ユーザーはそのメッセージに応答しなければなりません。このオプションを選択した場合には、システム応答リストを使用していない場合と同じ結果になります。
- システム応答リスト項目の中で指定されている応答を送る (最高 32 文字のメッセージ応答)。この場合には、指定されている応答がメッセージへの応答として送られます。そのメッセージは表示されず、デフォルトの処理プログラムも呼び出されません。

以下のコマンドは、メッセージ RPG1241、RPG1200、CPA4002、CPA5316、およびその他の照会メッセージに関する項目を、システム応答リストに追加するためのものです。

- `ADDRPYLE SEQNBR(15) MSGID(RPG1241) RPY(C)`
- `ADDRPYLE SEQNBR(18) MSGID(RPG1200) RPY(*DFT) DUMP(*YES)`
- `ADDRPYLE SEQNBR(22) MSGID(CPA4002) RPY(*RQD) +  CMPDTA('QSYSPRT')`
- `ADDRPYLE SEQNBR(25) MSGID(CPA4002) RPY(G)`
- `ADDRPYLE SEQNBR(27) MSGID(CPA5316) RPY(I) DUMP(*NO) +  CMPDTA('QSYSPRT' 21)`
- `ADDRPYLE SEQNBR(9999) MSGID(*ANY) RPY(*DFT)`

上記の結果、システム応答リストは次のようになります。

順序番号	メッセージ **ID**	比較値 **(b** はブランク**)**	比較開始位置	応答	**Dump**
10	CPA0700		1	D	*YES
15	RPG1241		1	C	*NO

順序番号	メッセージ **ID**	比較値 **(b** はブランク**)**	比較開始位置	応答	**Dump**
18	RPG1200		1	*DFT	*YES
20	RPG0000		1	D	*YES
22	CPA4002	'QSYSPRT'	1	*RQD	*NO
25	CPA4002		1	G	*NO
27	CPA5316	'QSYSPRT'	21	I	*NO
30	CBE0000		1	D	*YES
40	PLI0000		1	D	*YES
9999	*ANY		1	*DFT	*NO

このシステム応答リストを使用するジョブの場合、システム応答リストに追加されたメッセージがジョブによって送られるときに、次のことが生じます。

- 順序番号 15 の場合、システム応答リストを使用するジョブから RPG1241 が送られると、C の応答が返され、ジョブはダンプされません。
- 順序番号 18 の場合、ジョブから RPG1200 照会メッセージが送られたときにデフォルトの応答が返されるように、総称識別コードが使用されています。デフォルトの応答は、メッセージ記述で指定されているデフォルトの応答またはシステムのデフォルトの応答です。デフォルトの応答が返される前に、ジョブのダンプがとられます。追加された直前の項目 (順序番号 15) は、メッセージ RPG1241 のこの項目をオーバーライドするため、RPG1241 は順序番号 15 (18 ではない) に対してアクションを実行します。
- 順序番号 22 の場合、照会メッセージ CPA4002 が QSYSPRT の比較データと一緒に送られると、そのメッセージがディスプレイ装置ユーザーに送られ、ユーザーはそれに応答しなければなりません。

  開始位置を指定せずに比較値を指定した場合、その比較値はメッセージ中の置換データの 1 桁目から始まるメッセージ・データと比較されます。

  順序番号 22 では、印刷装置名が QSYSPRT であるかどうかをテストします。開始位置の異なる置換値の場合のテストの例については、順序番号 27 を参照してください。
- 順序番号 25 の場合、照会メッセージ CPA4002 (印刷装置 &1 の位置合わせを確認してください) が QSYSPRT 以外の比較データと一緒に送られると、G の応答が送られます。ジョブのダンプはとられません。順序番号 22 では、印刷装置が QSYSPRT の場合、用紙の位置合わせメッセージに対するオペレーターの応答が必要とされます。順序番号 25 では、その他の装置に対して用紙の位置合わせ照会メッセージが出された場合に、デフォルトの応答を G= 実行 (Go) と見なすことが定義されています。
- 順序番号 27 の場合、照会メッセージ CPA5316 が TESTEDFILETSTLIBRARYQSYSPRT の比較データと一緒に送られると、I の応答が送られます。

  比較値および開始位置を指定した場合には、比較値は開始位置以降の照会メッセージのメッセージ・データと比較されます。この例では、21 文字目は 3 番目の置換変数の開始位置です。メッセージ CPA5316 の場合、最初の 4 つの置換変数は以下のとおりです。

置換変数の順序	置換変数	タイプ	サイズ
&1	ODP ファイル名	*CHAR	10
&2	ODP ライブラリー名	*CHAR	10
&3	ODP 装置名	*CHAR	10
&4	最初の行の行番号	*BIN	2

したがって、順序番号 27 では、応答が返される前に ODP 装置名が QSYSPRT であるかどうかがテストされます。

- 順序番号 9999 の場合には、順序番号がそれより小さいどの項目にも該当しないすべての事前定義メッセージに対応するものとしてメッセージ識別コード *ANY が指定されており、それらの照会メッセージに対してはデフォルトの応答が送られます。システム応答リストにこの項目が含まれていなかった場合には、システム応答リストに含まれていないすべての定義済み照会メッセージに対して、ディスプレイ装置のオペレーターが応答しなければならなくなります。

比較値に *CCHAR がある場合には、送信機能からのメッセージ・データがシステム応答リスト中に保管されているメッセージ・データの CCSID に変換されてから、比較が行われます。変換されるのは *CCHAR タイプのデータだけです。

注意:
***CCHAR** データを比較データとして使用する場合に、以下の制約事項が適用されます。

- このタイプの応答リスト項目を追加する場合に、***CCHAR** データを他のデータと混用することはできません。
- 比較データには ***CCHAR** データの長さを含めることはできません。

***CCHAR** データを混用したり ***CCHAR** データの長さを含めると、予測不能な結果が生じることがあります。

項目は、**システム応答リスト項目除去 (RMVRPYLE)** コマンドによって除去されるまで、応答リストに存続します。 **システム応答リスト項目変更 (CHGRPYLE)** コマンドを使用して応答リスト項目の属性を変更することができ、 **システム応答リスト項目処理 (WRKRPYLE)** コマンドを使用して、現在の応答リストの応答項目を表示することができます。

ジョブ・ログは、ADDRPYLE、CHGRPYLE、または RMVRPYLE を用いてシステム応答リストが更新されるときに、変更の成功を示す完了メッセージを受け取ります。活動記録ログ QHST も、変更を記録するための完了メッセージを受け取ります。

メッセージがシステム応答リストに追加されると、CL プログラムを作成して項目を追加するのに役立ちます。例えば、応答リストが損傷し、IPL の後にクリアされる場合、ユーザーが項目を手動で再追加するのではなく、CL プログラムを呼び出して項目を再追加することができます。

システム応答リスト項目を更新するようにプログラムが作成されていない場合、システム応答リスト項目が失われた後に当該リストを再作成する次のような代替策があります。

- システム応答リストに変更が行われた後で、**システム情報検索 (RTVSYSINF)** コマンドを使用して、ライブラリーにデータを保存する。 これによって、システム応答リストに加えて他のシステム情報も保存されることに注意してください。その後、システム応答リストを更新する必要があるときに、**システム情報更新 (UPDSYSINF)** コマンドを使用できます。UPDSYSINF コマンドは、RTVSYSINF コマンドを使用して保存されたデータを使用します。
- 別の代替案は、バックアップからシステム応答リストを復元することである。このことは、システムの保存操作が完了していること、および保存操作前にシステム応答リストにすべての必要な項目が追加されていることを前提にしています。以下のステップに従って、基本オペレーティング・システムのスリップ・インストールを実行する必要があります。
  1. 手動モードのキーを使用して、IPL を実行する。
  2. 「DST IPL/install (DST IPL/インストール)」画面で、オプション 2 を選択してインストールする。

3. 「Install Operating System (オペレーティング・システムのインストール)」画面 (「time/date (時刻/日付)」画面など) で、オプション 2 を選択して、インストール・オプションを変更する。
4. 「Specify Install Options (インストール・オプションの指定)」画面で、オプション 1 を選択し、メディアからプログラムおよび言語オプションを復元する。
5. 「Specify Restore Options (リストア・オプションの指定)」画面で、オプション 1 を選択して、MESSAGE REPLY LIST を復元する。これは、画面上の 3 番目の項目です。

   注: この画面のデフォルト値は通常 2 または 3 で、項目を復元しないことを示します。
6. インストールを続行する。コマンド入力画面が表示されます。
7. 手動モードからキーを取得する。

関連タスク:

563 ページの『メッセージ待ち行列を自動的に中断モードにする』
メッセージ待ち行列を自動的に中断モードにすることで、QSYSOPR メッセージ待ち行列を監視できます。

611 ページの『中断処理プログラム』
中断処理プログラムは、*BREAK モードになっているメッセージ待ち行列にメッセージが到着した時に自動的に呼び出されるプログラムです。

関連情報:

メッセージ記述追加 (ADDMSG) コマンド

応答処理出口プログラムの使用

応答処理出口プログラムを使用すると、照会メッセージに応答が送られるときにユーザー提供の出口プログラムを呼び出せます。

この出口プログラムは、応答値の受け入れ、拒否、または置換を行うことができます。

関連情報:

Reply Handling Exit Program

CL プログラムまたはプロシージャー内のメッセージ・サブファイル

CL プログラムおよび CL プロシージャーおよびプログラム内では、メッセージ・サブファイルが、サポートされる唯一のサブファイルのタイプです。メッセージ・サブファイルによって、制御プログラムまたはプロシージャーは、1 つ以上のエラー・メッセージを表示することができます。

サブファイル・メッセージ・サポートを使用するには、サブファイル・メッセージ制御レコードを用いて**ファイル送信 (SNDF)** コマンドまたは**ファイル送信/受信 (SNDRCVF)** コマンドを実行します。DDS には SFLPGMQ データを提供し、常時 SFLINZ を活動状態にしておいてください。

CL プロシージャーおよびプログラム内でメッセージ・サブファイルを使用するときは、プロシージャーまたはプログラムの名前を指定しなければなりません。 DDS の SFLPGMQ キーワードに * を指定することはできません。プロシージャーまたはオリジナル・プログラム・モデル (OPM) プログラムの名前を指定すると、そのプロシージャーまたはプログラムのメッセージ待ち行列に送られるすべてのメッセージが、呼び出しメッセージ待ち行列から取り出されて、メッセージ・サブファイルに入れられます。現行要求に関連するすべてのメッセージは、CALL メッセージ待ち行列から取り出されて、メッセージ・サブファイルに入れられます。

関連概念:

DDS

メッセージのログ

メッセージのログには、ジョブ・ログと活動記録ログという 2 つのタイプがあります。

ジョブ・ログには、ジョブで入力された要求に関連した情報が保管されます。システム活動記録ログ (QHST) には、システムにおいてジョブの開始および終了の活動の記録などのシステム・データが入ります。

関連タスク:

592 ページの『要求メッセージの受信』
要求メッセージの受信は、CL プロシージャーまたは CL プログラムが CL コマンドを処理するための 1 つの方法です。

ジョブ・ログ

各ジョブには、ジョブ・ログが関連付けられています。

ジョブ・ログには、そのジョブに関する以下の項目を含めることができます。

- ジョブ内のコマンド。
- CL プログラム中の各コマンド。ただしそのプログラムが LOG(*YES) または LOG(*JOB) を指定して作成されているか、LOGCLPGM(*YES) を指定した**ジョブ変更 (CHGJOB)** コマンドが実行されている場合。
- 要求元に送られ、その呼び出しメッセージ待ち行列から除去されていないすべてのメッセージおよびメッセージ・ヘルプ。

関連概念:

564 ページの『外部メッセージ待ち行列』
外部メッセージ待ち行列 (*EXT) を使用すると、ジョブの外部要求側 (ディスプレイ装置ユーザーなど) と通信できます。

関連タスク:

266 ページの『CL プログラムまたはプロシージャーのコマンドのロギング』
CL プログラムまたはプロシージャー内で実行されているほとんどの CL コマンドをジョブ・ログに書き込む (記録する) よう指定できます。

ファイルへのジョブ・ログの書き込み:

ジョブの終了時に、ジョブ・ログを出力ファイル QPJOBLOG かデータベース・ファイルに書き込むことができます。

出力ファイル QPJOBLOG に書き込むと、ジョブ・ログを印刷できます。データベース・ファイルに書き込むと、データベース機能を使用してジョブ・ログ情報を照会できます。正常に実行されたジョブについては、ジョブ・ログを書き込まないように指定することもできます。ジョブのジョブ・ログ出力ジョブ属性が *PND の場合は、ジョブ・ログの終了時にジョブ・ログが作成されませんが、保留ジョブ・ログはまだ**ジョブ・ログ処理 (WRKJOBLOG)** コマンドからアクセスできます。

QMHCTLJL API を使用すると、データベース・ファイルにジョブ・ログを書き込むことができます。ジョブ・ログをデータベースに書き込むと、1 つまたは 2 つのファイルが生成されます。 1 次ファイルには、メッセージ ID、メッセージ重大度、メッセージ・タイプ、およびメッセージ・データなどのメッセージの必須情報が入っています。 2 次ファイルには、メッセージ・テキストの印刷メッセージが入っています。 2 次ファイルの作成はオプションで、QMHCTLJL API のパラメーターで制御されます。両方のファイルとも外部記述され、システムのデータベースおよび照会機能を使用して処理することができます。

関連資料:

633 ページの『ジョブ・ログ出力ファイル』
ジョブ・ログ出力ファイルを使用する際は、出力ファイルのレコード形式を記述するファイルおよびモデル・ファイルに必要なジョブ・ログ情報を検討してください。

関連情報:

ジョブ・ログ出力制御 (QMHCTLJL) API

ジョブ・ログの保留 (Job log pending)

ジョブ・ログに書き込まれた制御情報:

システムがジョブ・ログに記録する情報を制御するには、**ジョブ記述作成 (CRTJOBD)** コマンドで LOG パラメーターを指定します。 レベルは、**ジョブ変更 (CHGJOB)** コマンドまたは**ジョブ記述変更 (CHGJOBD)** コマンドを使用して変更することができます。

LOG パラメーターは、メッセージ・レベル、メッセージ重大度、およびメッセージ・テキスト・レベルの 3 つの値によって構成されます。

最初の値であるメッセージ・レベルには、以下のレベルがあります。

レベル 説明

0 データは記録されません。

1 記録される情報は、ジョブの外部メッセージ待ち行列に送られたメッセージで、指定したメッセージ重大度と同じか、またはそれより大きい重大度を持つすべてのメッセージだけです。このタイプのメッセージは、ジョブの開始や終了の日時、および完了時のジョブの状況を示すものです。

2 以下の情報が記録されます。

- レベル 1 のロギング情報。
- 指定した重大度コードと同じかまたはそれより高い重大度を持つ高レベル・メッセージが出された要求。要求がログに記録される場合、それに関するメッセージもすべてログに記録されます。

3 以下の情報が記録されます。

- ロギング・レベル 1 および 2 の情報。
- すべての要求。
- CL **プログラムのコマンドのログ**・ジョブ属性、および CL プログラムの**ログ**属性によって許可されている場合に、CL プログラムによって実行されるコマンド。

4 以下の情報が記録されます。

- トレース・メッセージを含む、指定した重大度コードと同じかまたはそれより高い重大度を持つすべての要求およびすべてのメッセージ。
- CL **プログラムのコマンドのログ**・ジョブ属性、および CL プログラムの**ログ**属性によって許可されている場合に、CL プログラムによって実行されるコマンド。

  注: 高レベル・メッセージとは、要求を受け取るプログラムのプログラム・メッセージ待ち行列に送られるメッセージです。 例えば、QCMD は、要求を受け取る IBM 提供の要求処理プログラムです。

2 番目の値、すなわちメッセージ重大度には、エラー・メッセージをジョブ・ログに記録する基準となる重大度レベルをログ・レベルとともに指定します。指定できる値は 0 から 99 までです。

LOG パラメーターの 3 番目の値、すなわちメッセージ・テキスト・レベルには、ジョブ・ログに記録するメッセージ・テキストのレベルを指定します。指定できる値は次のいずれかです。

***SAME**
メッセージ・テキスト・レベルの現在の値は変更されません。

***MSG**
メッセージ・テキストだけがジョブ・ログに記録されます (メッセージ・ヘルプは記録されません)。

***SECLVL**
メッセージおよびメッセージ・ヘルプ (原因およびリカバリー) がジョブ・ログに書き込まれます。

ジョブ・ログ・メッセージのフィルター**:**

メッセージにフィルターをかけるのは、ジョブに設定されているメッセージ・ロギング・レベルに基づいて、ジョブ・ログからメッセージを除去するプロセスです。

それぞれの新しい要求が要求処理プログラムにより受け取られる前に、メッセージがフィルターにかけられます。

すべての CL コマンドがプログラム内で呼び出されるたびにフィルターがかけられるわけではありません。したがって、CL プログラムが対話式に実行されるか、またはバッチに投入される場合、プログラムは要求プロセッサーではないのでプログラムが終了した後にフィルターが実行されます。

注**:** *NOLIST は正常に終了するジョブに対してスプールされたジョブ・ログを作成しないことを指定するため、ログ・レベル 0 を指定してこのログからメッセージを除去することはバッチ・ジョブ内のシステム資源の浪費になります。

例**:** ジョブ・ログに書き込まれた制御情報**:**

この例は、ジョブ・メッセージ待ち行列に記録される情報およびジョブ・ログに記録される情報に対してロギング・レベルが与える影響を示しています。

この例は、コマンドが対話式に実行されている場合、各コマンドが実行され、次の要求が受信されたときに、フィルター操作が行われることも示しています。

注**:** 例には、高レベル・メッセージと詳細なメッセージの両方のログ・レベルが組み込まれています。高レベル・メッセージは、メッセージ というテキストで始まります。詳細メッセージは 詳細メッセージ というテキストで始まります。

1. 次の CHGJOB コマンドは、ロギング・レベル 2、メッセージ重大度が 50、そして 1 次レベルのメッセージだけをジョブ・ログに書き込むこと (*MSG) を指定しています。

```
                          コマンド入力                    SYSTEM1
                                                 要求レベル:   1
前のコマンドおよびメッセージ :
    > CHGJOB LOG(2 50 *MSG)
```

2. PGMA は、コマンド入力画面を使用して、重大度コードが 20、50、および 60 の 3 つの通知メッセージを、それ自体の呼び出しメッセージ待ち行列、および呼び出し側または直前の呼び出しメッセージ待ち行列 (例えば *PRV で、この例では QCMD) に送信します。PGMA がそれ自身の呼び出しメッセージ待ち行列に送るメッセージは、詳細なメッセージと呼ばれます。詳細なメッセージとは、下位レベルのプログラム呼び出しの呼び出しメッセージ待ち行列に送られるメッセージのことです。

PGMB は、それ自身の呼び出しメッセージ待ち行列に重大度コードが 40 および 50 の 2 つの情報メッセージを送ります。これらは詳細なメッセージです。 PGMB はまた、重大度コードが 10 の 1 つの通知メッセージ (高レベル・メッセージ) を *PRV に送信します。

次の画面では、PGMA および PGMB が呼び出された後には、CHGJOB コマンドが画面に表示されなくなっていることに注意してください。ロギング・レベル 2 に従って、指定された重大度以上の重大度を持つ高レベル・メッセージが出された要求のみがジョブ・ログに保管されますが、この要求についてはメッセージは出されていません。 次の要求 CALL PGMA が受信されたときに、CHGJOB が除去されたか、ジョブ・ログからフィルター操作されました。新規要求が受信されると、ログ・レベルに従って直前の要求がフィルター操作されます。このような高レベル・メッセージが出されていた場合、発行済みの詳細メッセージはジョブ・ログに保管され、F10 を押すことにより表示できます。

```
                          コマンド入力                    SYSTEM1
                                                  要求レベル:  1
前のコマンドおよびメッセージ :
      > CALL PGMA
        メッセージ重大度 20 - PGMA
        メッセージ重大度 50 - PGMA
        メッセージ重大度 60 - PGMA
      > CALL PGMB
        メッセージ重大度 10 - PGMB









                                                            終わり
コマンドを入力して，実行キーを押してください。
===> ____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=終了          F4=プロンプト   F9=コマンドの複写  F10=詳細なメッセージの組み込み
 F11=全画面表示   F12=取り消し    F13=情報援助      F24=キーの続き
```

要求 CALL PGMA の結果、現在のログ重大度以上の高レベル・メッセージが出されるので、要求 CALL PGMB が入力されると、フィルター操作される PGMA からのメッセージはありません。

3. コマンド入力画面で F10 (詳細なメッセージの組み込み) を押すと、入力された要求 CALL PGMA に関連するすべてのメッセージが表示されます。要求 CALL PGMB に対するフィルター操作はまだ行われていないので、その要求に対するメッセージもすべて表示されます。

```
                                コマンド入力                    SYSTEM1
                                                        要求レベル:   1
すべての前のコマンドおよびメッセージ :
   > CALL PGMA
     詳細なメッセージ重大度 20 - PGMA
     詳細なメッセージ重大度 50 - PGMA
     詳細なメッセージ重大度 60 - PGMA
     メッセージ重大度 20 - PGMA
     メッセージ重大度 50 - PGMA
     メッセージ重大度 60 - PGMA
   > CALL PGMB
     詳細なメッセージ重大度 40 - PGMB
     詳細なメッセージ重大度 50 - PGMB
     メッセージ重大度 10 - PGMB

                                                              終わり
コマンドを入力して，実行キーを押してください。
===> ____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
 F3=終了          F4=プロンプト   F9=コマンドの複写  F10=詳細なメッセージの除外
 F11=全画面表示   F12=取り消し    F13=情報援助       F24=キーの続き
```

4. 別のコマンド (この例ではもう 1 つの CHGJOB) が入力されると、CALL PGMB コマンドおよびすべてのメッセージ (詳細なメッセージを含む) が除去されます。これは、この要求に関連している高レベル・メッセージの重大度コードが、CHGJOB コマンドで指定されている重大度コードと同じでないかそれよりも高いためです。 CALL PGMA コマンドおよびそれに関するメッセージはそのまま残されます。これは、その要求に対して出された高レベル・メッセージのうちの少なくとも 1 つが、指定された重大度コードと同じかまたはそれより高い重大度コードを持っているためです。詳細なメッセージの組み込みをやめるには F10 をもう一度押します。

   以下の画面では、CHGJOB コマンドでロギング・レベルが 3 であること、メッセージ重大度が 40 であること、およびメッセージの 1 次レベルと 2 次レベルの両方のテキストをジョブ・ログに書き込むことが指定されています。ロギング・レベル 3 の場合はすべての要求が保管されるので、別のコマンドが入力されても、CHGJOB コマンドは画面上に残ります。

   PGMC は、重大度コードが 30 と 40 の 2 つのメッセージを、その呼び出し側の呼び出しメッセージ待ち行列 (*PRV) に送信します。

   PGMD は、*PRV に重大度レベルが 10 の 1 つのメッセージを送ります。

```
                                コマンド入力                    SYSTEM1
                                                        要求レベル:  1
前のコマンドおよびメッセージ :
     > CALL PGMA
       メッセージ重大度 20 - PGMA
       メッセージ重大度 50 - PGMA
       メッセージ重大度 60 - PGMA
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       メッセージ重大度 30 - PGMC
       メッセージ重大度 40 - PGMC
     > CALL PGMD
       メッセージ重大度 10 - PGMD


                                                                終わり
コマンドを入力して，実行キーを押してください。
===> ____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=終了          F4=プロンプト  F9=コマンドの複写  F10=詳細なメッセージの組み込み
 F11=全画面表示   F12=取り消し   F13=情報援助      F24=キーの続き
```

5. CALL PGMD コマンドの入力の後で別のコマンド (CALL PGME) が入力された場合、CALL PGMD コマンドは画面上に残されますが、それについてのメッセージは削除されます。メッセージが削除される (ジョブ・ログからフィルター操作される) のは、その重大度コードが CHGJOB コマンドの LOG パラメーターに指定されている重大度コードより小さいためです。

   ジョブを終了して、ジョブ・ログを印刷するために、コマンド SIGNOFF *LIST が入力されます。

```
                                コマンド入力                    SYSTEM1
                                                        要求レベル:  1
前のコマンドおよびメッセージ :
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       メッセージ重大度 30 - PGMC
       メッセージ重大度 40 - PGMC
     > CALL PGMD
     > CALL PGME




                                                                終わり
コマンドを入力して，実行キーを押してください。
===> SIGNOFF *LIST___________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=終了          F4=プロンプト  F9=コマンドの複写  F10=詳細なメッセージの組み込み
 F11=全画面表示   F12=取り消し   F13=情報援助      F24=キーの続き
```

ジョブ・ログ (次のページに示されています) には、コマンド入力画面に残されているすべての要求およびすべてのメッセージが含まれています。さらに、最後の CHGJOB コマンドでの指定に応じて、各メッセージのメッセージ・ヘルプもジョブ・ログに記録されています。 2 番目の CHGJOB コマンドが入力された後で出されたメッセージだけでなく、ジョブの過程で出されたすべてのメッセージについてメッセージ・ヘルプがジョブ・ログに記録されている点に注意してください。

```
 5770SS1 V7R1M0 100416                        ジョブ・ログ                      SYSAS727 01/16/11 07:13:35         ページ  1
  ジョブ名 . . . . . . . . . . :   QPADEV000C      ユーザー . . . . . . . :  JOHNDOE      番号 . . . . . . . . . . . :   038518
  ジョブ記述 . . . . . . . . . :   QDFTJOBD        ライブラリー . . . . . :   QGPL
MSGID      タイプ                 SEV  日付      時刻           FROM PGM    ライブラリー  INST   TO PGM  ライブラリー  INST
CPF1124    通知                   00   01/16/11  07:13:19.570504  QWTPIIPP     QSYS        0613     *EXT                    *N
                                    メッセージ . . . . :   QSYS のサブシステム QINTER のジョブ 038518/JOHNDOE/QPADEV000C が
                                      01/16/11 07:13:19 に開始された。ジョブは、01/16/11 07:13:19 にシステムに入れられました。
*NONE      要求                     01/16/11  07:13:24.318144  QMHGSD       QSYS        0010     QCMD        QSYS        0178
                                    メッセージ . . . . :  -CALL PGMA
MSG1001    通知                   20   01/16/11  07:13:24.361064  PGMA         JOHNDOE     0029     PGMA        JOHNDOE     0029
                                    送信元ユーザー . . . . . . . . . :   MARYJANE

                                    メッセージ . . . . :   詳細なメッセージ重大度 20 - PGMA
                                    MSG1001 2 次レベル・テキスト - PGMA
MSG1002    通知                   50   01/16/11  07:13:24.361416  PGMA         JOHNDOE     0032     PGMA        JOHNDOE     0032
                                    メッセージ . . . . :   詳細なメッセージ重大度 50 - PGMA
                                    MSG1002 2 次レベル・テキスト - PGMA
MSG1003    通知                   60   01/16/11  07:13:24.361592  PGMA         JOHNDOE     0036     PGMA        JOHNDOE     0036
                                    メッセージ . . . . :   詳細なメッセージ重大度 60 - PGMA
                                    MSG1003 2 次レベル・テキスト - PGMA
MSG1004    通知                   20   01/16/11  07:13:24.361776  PGMA         JOHNDOE     003A     QCMD        QSYS        01A6
                                    メッセージ . . . . :   メッセージ重大度 20 - PGMA
                                    MSG1004 2 次レベル・テキスト - PGMA
MSG1005    通知                   50   01/16/11  07:13:24.362192  PGMA         JOHNDOE     0043     QCMD        QSYS        01A6
                                    送信元ユーザー . . . . . . . . . :   MARYJANE

                                    メッセージ . . . . :   メッセージ重大度 50 - PGMA
                                    MSG1005 2 次レベル・テキスト - PGMA
MSG1006    通知                   60   01/16/11  07:13:24.362552  PGMA         JOHNDOE     004C     QCMD        QSYS        01A6
                                    メッセージ . . . . :   メッセージ重大度 60 - PGMA
                                    MSG1006 2 次レベル・テキスト - PGMA
*NONE      要求                     01/16/11  07:13:24.370240  QMHGSD       QSYS        0018     QCMD        QSYS        0178
                                    メッセージ . . . . :  -CHGJOB LOG(3 40 *SECLVL)
*NONE      要求                     01/16/11  07:13:24.370672  QMHGSD       QSYS        001C     QCMD        QSYS        0178
                                    メッセージ . . . . :  -CALL PGMC
MSG100F    通知                   30   01/16/11  07:13:24.379256  PGMC         JOHNDOE     *STMT    QCMD        QSYS        01A6
                                    送信元ユーザー . . . . . . . . . :   MARYJANE

                                    送信元モジュール . . . . . . . . :   PGMC

                                    送信元プロシージャー  . . . . . .:   PGMC

                                    ステートメント . . . . . . . . . :   8000

                                    メッセージ . . . . :   メッセージ重大度 30 - PGMC
                                    MSG100F 2 次レベル・テキスト - PGMC
MSG1010    通知                   40   01/16/11  07:13:24.379608  PGMC         JOHNDOE     *STMT    QCMD        QSYS        01A6
                                    送信元モジュール . . . . . . . . :   PGMC

                                    送信元プロシージャー  . . . . . .:   PGMC

                                    ステートメント . . . . . . . . . :   8200

 5770SS1 V7R1M0 100416                        ジョブ・ログ                      LPAR3TLM 01/16/11 07:13:35         ページ  2
  ジョブ名 . . . . . . . . . . :   QPADEV000C      ユーザー . . . . . . . :  JOHNDOE      番号 . . . . . . . . . . . :   038518
  ジョブ記述 . . . . . . . . . :   QDFTJOBD        ライブラリー . . . . . :   QGPL
MSGID      タイプ                 SEV  日付      時刻           FROM PGM    ライブラリー  INST   TO PGM  ライブラリー  INST
                                    メッセージ . . . . :   メッセージ重大度 40 - PGMD
                                    MSG1010 2 次レベル・テキスト - PGMC
*NONE      要求                     01/16/11  07:13:24.379928  QMHGSD       QSYS        0020     QCMD        QSYS        0178
                                    メッセージ . . . . :  -CALL PGMD
*NONE      要求                     01/16/11  07:13:24.383568  QMHGSD       QSYS        0024     QCMD        QSYS        0178
                                    メッセージ . . . . :  -CALL PGME
*NONE      要求                     01/16/11  07:13:35.166408  QMHGSD       QSYS        073E     QCMD        QSYS        0178
                                    メッセージ . . . . :  -SIGNOFF *LIST
CPF1164    完了                   00   01/16/11  07:13:35.173496  QWTMCEOJ     QSYS        00BD     *EXT                    *N
                                    メッセージ . . . . :   07:13:35 に 01/16/11 でジョブ
                                      038518/JOHNDOE/QPADEV000C が終了した。1 秒が使用され、終了コードは 0 です。
                                    原因 . . . . . :   1 秒の処理装置時間を使用した後で 07:13:35 に 01/16/11 でジョブ
                                      038518/JOHNDOE/QPADEV000C が完了しました。ジョブの終了コードは 0 です。
                                      1 経路指定ステップ後に、2 次終了コード 0 でジョブが終了しました。
                                      ジョブの終了コードおよびその意味は次の通りです。0 - ジョブは正常に完了した。
                                      10 - 制御された終了または制御されたサブシステム終了時に、ジョブが正常に完了した。
                                      20 - ジョブが終了重大度 (ENDSEV ジョブ属性) を超えた。
                                      30 - ジョブが異常終了した。
                                      40 - 活動状態になる前にジョブが終了した。
                                      50 - ジョブが活動状態の時にジョブが終了
                                      した。
                                      60 - ジョブが活動状態の時に、サブシステムが異常終了した。70 - ジョブが活動
                                      状態の時にシステムが異常終了した。
                                      80 - ジョブが終了した (ENDJOBABN コマンド)。
                                      90 - 時間制限が終了した後でジョブが強制終了した (ENDJOBABN コマンド)。
                                      回復手順  . . . :   詳細については、INFORMATION CENTER (http://www.iseries.
                                      ibm.com/infocenter) の『実行管理機能』トピックを参照してください。
```

印刷されたジョブ・ログの各ページの始めにある見出しは、このジョブ・ログの対象となっているジョブおよび各項目の特性を示しています。

- オペレーティング・システムの製品 ID、バージョン、および日付
- システム名
- ジョブ・ログが印刷された日時

- ジョブの完全修飾名 (ジョブ名、ユーザー名、およびジョブ番号)
- ジョブを開始するために使用されたジョブ記述の名前。
- セクション番号。これは、ジョブ・ログが折り返され、ジョブ・メッセージ待ち行列の完全処理用に *PRTWRAP が指定されているために、ジョブ・ログが複数のセクションに分かれて印刷される場合に印刷されます。
- 各メッセージ項目の最初の行にある情報の見出し。

ジョブ・ログ内の各メッセージ項目について、以下の情報が印刷されます。

- 各メッセージの最初の行には以下の情報が含まれます。
  - メッセージ ID または *NONE。
  - メッセージ・タイプ。
  - メッセージ重大度。要求メッセージの場合、これはブランクです。
  - 各メッセージが送られた日時。
  - メッセージを送信したプログラムのプログラム名、ライブラリー名、および命令番号。
  - メッセージが送信されたプログラムのプログラム名、ライブラリー名、および命令番号。 *EXT は、メッセージがジョブの外部メッセージ待ち行列に送信されたことを示します。
- 修飾ジョブ名のユーザーで識別されているユーザーとは異なるユーザーによってメッセージが送られた場合は、メッセージを送信したユーザーの名前が別個の行に印刷されます。これは以下のいずれかの状態を示している可能性があります。
  - メッセージは、ジョブが異なるユーザー・プロファイルの下で実行されている間に送信された。前のサンプル・ジョブ・ログでは、メッセージ MSG1001、MSG1005、および MSG100F は、ジョブが異なるユーザー・プロファイルの下で実行されている間に送信されました。
  - 照会メッセージが別のユーザーによって応答された。
  - バッチ・ジョブが、そのバッチ・ジョブを実行しているユーザー・プロファイルとは異なるユーザーによって投入された。この場合、各要求メッセージで投入ユーザーの名前が含められます。
  - 別のユーザーがジョブ属性を変更し、監査が活動していなかったため、別のユーザーにより変更が行われたことをジョブに通知するために、メッセージがそのジョブに送信された。
- 送信側が統合言語環境 (ILE) プロシージャーである場合は、モジュール、プロシージャー、およびステートメント番号を識別する追加行が印刷されます。前のサンプル・ジョブ・ログでは、メッセージ MSG100F において、 PGMC が ILE プログラムです。
- ジョブがマルチスレッド・ジョブであり、メッセージが複数のスレッドから送信された場合は、各メッセージでスレッド ID が印刷されます。
- メッセージは 1 つ以上の行に印刷されます。
- ロギング・レベルにより 2 次レベルのテキストも含めることが示されている場合には、1 次レベルのメッセージの次の行以降に 2 次レベルのテキストが示されます。

関連概念:

『ジョブ・ログ送信者または受信者情報』
ジョブ・ログには、プログラムやプロシージャーの送信や受信についての情報が含まれています。

ジョブ・ログ送信者または受信者情報:

ジョブ・ログには、プログラムやプロシージャーの送信や受信についての情報が含まれています。

送信側または受信側が統合言語環境 (ILE) プロシージャーである場合、メッセージ項目にはプロシージャーの全名 (プロシージャー名、モジュール名、および ILE プログラム名) が含まれます。送信側または受信側がオリジナル・プログラム・モデル (OPM) である場合、OPM プログラム名だけが表示されます。

送信側または受信側が OPM プログラムである場合、命令番号は対応する命令番号を表しています。そのような番号は 1 つしかありません。送信側または受信側が ILE プロシージャーである場合、命令番号は MI 命令番号ではなく、高水準言語のステートメント番号を表しています。 ILE プロシージャーが最適化されていると (最大効率)、最高で 3 つまでの番号があります。最適化されたプロシージャーの単一ステートメント番号を判別するのは、必ずしも可能ではありません。与えられた番号が 2 つ以上ある場合には、それぞれの番号はメッセージが送信されたときにプロシージャーがあった可能性のある点を表しています。判別できる番号が 1 つもないこともあります。このような場合、メッセージは番号ではなく *N が表示されます。

ロギング・レベルは、バッチ・ジョブのログにも上記の例に示されているのと同じ影響を与えます。
APPC を使用するジョブの場合には、APPC の作業単位識別コードを示す行が見出しに含まれます。

関連概念:

622 ページの『例: ジョブ・ログに書き込まれた制御情報』
この例は、ジョブ・メッセージ待ち行列に記録される情報およびジョブ・ログに記録される情報に対してロギング・レベルが与える影響を示しています。

ジョブ・ログの表示:

ジョブ・ログを表示する方法は、ジョブの状況によって異なります。

- **ジョブ・ログ処理 (WRKJOBLOG)** コマンドを使用すると、完了ジョブの保留ジョブ・ログ、ジョブ・ログのすべてのスプール・ファイル、またはその両方を表示することができます。例えば、終了しているすべてのジョブの処理中のジョブ・ログのリストを表示するには、以下を入力します。

  ```
  WRKJOBLOG JOBLOGSTT(*PENDING)
  ```

- ジョブが活動中であるかジョブ待ち行列にある場合、またはジョブ・ログが処理中の場合は、**ジョブ・ログ表示 (DSPJOBLOG)** コマンドを使用します。例えば、ディスプレイ装置 WS1 の JSMITH というユーザーの対話式ジョブのジョブ・ログを表示したい場合には、次のように入力します。

  ```
  DSPJOBLOG JOB(nnnnnn/JSMITH/WS1)
  ```

  ここで nnnnnn はジョブ番号です。

- ジョブが終了し、ジョブ・ログが出力ファイルに書き込まれてはいるがまだ印刷されていない場合には、次のように **スプール・ファイル表示 (DSPSPLF)** コマンドを使用します。

  ```
  DSPSPLF FILE(QPJOBLOG) JOB(001293/FRED/WS3)
  ```

  このコマンドは、ディスプレイ装置 WS3 の FRED というユーザーに関連したジョブ番号 001293 のジョブ・ログを表示します。

ユーザーが自身の対話式ジョブのジョブ・ログを表示するには、次のいずれかを行います。

- 次のコマンドを入力します。

  ```
  DSPJOBLOG
  ```

- WRKJOB コマンドを入力し、「ジョブの処理」画面でオプション 10 (ジョブ・ログの表示) を選択します。
- コマンド入力画面で F10 キー (詳細メッセージの組み込み) を押します (このキーを押すとジョブ・ログに記録されているメッセージが表示されます)。

- ディスプレイ装置の入力禁止表示標識がオンになり、そのまま消えない場合には、次を行ってください。
  1. システム要求キーを押し、次に実行キーを押します。
  2. システム要求メニューで、オプション 3 (現行ジョブの表示) を選択します。
  3. ジョブの表示メニューで、オプション 10 (活動状態、またはジョブ待ち行列にある場合のジョブ・ログの表示) を選択します。
  4. 「ジョブ・ログ表示」画面に、処理要求として DSPJOB が現れます。 F10 (詳細メッセージの表示) キーを押します。
  5. 「すべてのメッセージの表示」画面で、前ページ・キーを用いて、システム要求キーを押す前に受け取られたメッセージを表示します。
- SIGNOFF コマンドに LOG(*LIST) を指定して、ワークステーションをサインオフします。

DSPJOBLOG コマンドを使用すると、「ジョブ・ログ表示」画面が表示されます。この画面には、プログラム名とそれに付随して以下のような特殊記号が示されます。

>> 実行中のコマンドまたは次に実行されるコマンド。例えば、プログラムが呼び出された場合には、そのプログラムの呼び出しが示されます。

> そのコマンドの処理が完了したことを示します。

. . そのコマンドがまだ処理されていないことを示します。

? 応答メッセージ。この記号は、応答を必要としているメッセージおよび既に応答済みのメッセージの両方に付けられています。

「ジョブ・ログ表示」画面では、以下のいずれかを行うことができます。

- F10 を押して詳細なメッセージを表示する。この画面には、高水準言語 (HLL) プログラム、または LOG が活動状態にある CL プログラムかプロシージャーで実行されたコマンドまたは命令が表示されます。
- カーソル移動キーを使用して、ジョブ・ログの終わりに進む。ジョブ・ログの終わりに早く進みたい場合には、F18 (最下部) を押します。 F18 を押した後で、実行中のコマンドを見るためにロールダウンを行わなければならないこともあります。
- カーソル移動キーを押して、ジョブ・ログの始めに戻る。ジョブ・ログの始めに早く戻りたい場合には、F17 (最上部) を押します。

DSPJOBLOG コマンドを使用して、ジョブを印刷したり表示する代わりにデータベース・ファイルに書き込むことができます。 2 つのオプションが使用できます。 1 つ目のオプションを使用すると、ファイルとメンバー名をコマンドに指定できます。このオプションでは、1 次ジョブ・ログ情報はコマンドで指定されたデータベース・ファイルに書き込まれます。 2 つ目のオプションを使用すると、コマンドを使用する際に以前に QMHCTLJL API を実行した時の情報を使用できます。このオプションでは、ジョブ・ログは API 呼び出しで指定されたファイル (1 つまたは複数) に書き込まれます。このオプションを使用すると、メッセージがファイルに書き込まれる際に 1 次ファイルと 2 次ファイルの両方を作成でき、メッセージのフィルターを実行できます。これらのオプションを両方とも使用すると、**DSPJOBLOG** コマンドが完了した時点で出力は表示されず、印刷に利用できるスプール・ファイルは存在しません。

関連情報:

ジョブ・ログの保留 (Job log pending)

ジョブ・ログの生成の抑制:

バッチ・ジョブの完了時にジョブ・ログが生成されないようにするためには、 **バッチ・ジョブ (BCHJOB)**、**ジョブ投入 (SBMJOB)**、ジョブ変更 (CHGJOB)、 **ジョブ記述作成 (CRTJOBD)**、または**ジョブ記述変更 (CHGJOBD)** コマンドの LOG パラメーターにメッセージ・テキスト・レベルの値として *NOLIST を指定することができます。

LOG パラメーターにメッセージ・テキスト・レベルの値として *NOLIST を指定した場合には、ジョブ終了コードが 20 以上でなければ、ジョブの終了時にジョブ・ログは生成されません。ジョブ終了コードが 20 以上である場合には、ジョブ・ログが生成されます。

対話式ジョブの場合には、SIGNOFF コマンドの LOG パラメーター値はそのジョブに指定されている LOG パラメーターの値より優先します。

ジョブが完了し、まだシステムに保留状態で残っているときにジョブ・ログが生成されないようにするには、**ジョブ投入 (SBMJOB)**、**ジョブ変更 (CHGJOB)**、**ジョブ記述作成 (CRTJOBD)**、または**ジョブ記述変更 (CHGJOBD)** のいずれかのコマンドの LOGOUTPUT パラメーターに *PND を指定します。LOG パラメーターに *NOLIST を指定した場合は、ジョブ・ログは生成されず、保留ジョブ・ログも存在しなくなります。保留ジョブ・ログを使用できるのは、ジョブが終了し、そのジョブ・ログ出力ジョブ属性が *PND の場合に、ジョブ・ログが正常に出力ファイルまたはデータベース・ファイルに書き込まれる場合だけです。**ジョブ・ログ処理 (WRKJOBLOG)** コマンドを使用すると、保留中および書き込まれたジョブ・ログをどちらも見つけることができます。

関連情報:

ジョブ・ログの保留 (Job log pending)

ジョブ・ログの考慮事項:

ジョブ・ログを使用する際は、次の注意事項について考慮してください。

- システム内のすべてのジョブについて出力待ち行列を変更するためには、**印刷装置ファイル変更 (CHGPRTF)** コマンドの OUTQ パラメーターまたは DEV パラメーターを使用して、ファイル QSYS/QPJOBLOG を変更してください。次に示すのはこれらのパラメーターの使用例です。

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          DEV (USRPRT)
  ```

  または

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          OUTQ(USROUTQ)
  ```

- 出力待ち行列 QEZJOBLOG を使用する QPJOBLOG 印刷装置ファイルは、操作援助機能の終結処置機能で使用されます。ジョブ・ログの自動終結処置機能を使用したい場合、印刷装置ファイルをこの出力待ち行列に直接接続する必要があります。
- ジョブのジョブ・ログが書き込まれる出力待ち行列を指定するためには、必ずファイル QPJOBLOG に OUTQ(*JOB) の指定があることを確認してください。 OUTQ パラメーターは、BCHJOB、CRTJOBD、CHGJOBD、または CHGJOB の各コマンドで使用することができます。次にこのコマンドの例を示します。

  ```
  CHGJOB OUTQ(USROUTQ)
  ```

  ジョブの開始時にデフォルトの OUTQ を変更した場合、すべてのスプール・ファイルが影響を受けます。ジョブ終了直前に変更した場合は、ジョブ・ログだけが影響を受けます。 **印刷装置ファイル一時変更 (OVRPRTF)** コマンドを使用してジョブ・ログを操作することはできません。

- ジョブの出力待ち行列が見つからない場合は、ジョブ・ログは作成されません。
- すべてのジョブ・ログを保持するには、ファイル QSYS/QPJOBLOG に対する CHGPRTF コマンドに HOLD(*YES) を指定します。 **スプール・ファイル解放 (RLSSPLF)** コマンドが実行された時点で、ジョブ・ログは書き出しプログラムに解放されます。次にこのコマンドの例を示します。

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          HOLD(*YES)
  ```

- システムが異常終了した場合、システム・オペレーターは開始プロンプトを使用して、異常終了の時点で活動状態にあったジョブのジョブ・ログを印刷するかどうかを指定することができます。
- ジョブ・ログを削除するためには、 **スプール・ファイル削除 (DLTSPLF)** コマンドを使用するか、または出力待ち行列画面で削除オプションを使用してください。
- **印刷装置ファイル変更 (CHGPRTF)** コマンドの USRDTA パラメーターを使用して、QSYS/QPJOBLOG ファイルに関するユーザー・データの値を変更した場合には、指定した値は「出力待ち行列処理」画面または「すべてのスプール・ファイルの処理」画面に表示されません。ユーザー・データの欄に表示されるのは、ジョブ・ログが印刷されたジョブのジョブ名です。
- プログラミングの手法でジョブ・ログを分析する場合は、QMHCTLJL API を使用してジョブ・ログをデータベース・ファイル (1 つまたは複数) に書き込んでください。データベース・ファイル内のレコードの様式は保証されていますが、印刷様式は保証されていません。新規のフィールドをジョブ・ログのレコードに追加する必要がある場合には、そのフィールドはレコードの末尾に追加されるので、プログラムは作業を継続できます。また、ファイルは正規化されているので、システムに備えられている照会機能を直接使用できます。

関連情報:

Operational Assistant APIs

対話式ジョブ・ログの考慮事項:

対話式ジョブ・ログを使用するときは、これらの事項を考慮してください。

IBM 提供のジョブ記述 QCTL、QINTER、および QPGMR のログ・レベルはどれも LOG(4 0 *NOLIST) です。したがって、すべてのメッセージの 1 次レベルおよび 2 次レベルのテキストがジョブ・ログに書き込まれます。ただし、SIGNOFF コマンドで *LIST を指定しなければ、ジョブ・ログは印刷されません。対話式ジョブのログ・レベルを変更したい場合には、**ジョブ変更 (CHGJOB)** コマンドまたは **ジョブ記述変更 (CHGJOBD)** コマンドを使用できます。

ディスプレイ装置ユーザーが IBM 提供のメニューまたはコマンド入力画面を使用する場合には、すべてのエラー・メッセージが表示されます。ディスプレイ装置ユーザーがユーザー作成の初期プログラムを使用する場合には、監視の対象となっていないメッセージが出されると初期プログラムは終了し、ジョブ・ログが生成されます。ただし、初期プログラムがメッセージを監視している場合には、メッセージを受け取った時点でそのプログラムに制御権が渡されます。この場合には、ジョブ・ログが作成されるようにして、どのようなエラーが起こったかを判別できるようにしておくことが重要です。例えば、次の初期プログラムがサインオフ・オプションを含むメニューを表示し、そのオプションのデフォルト値が *NOLIST であるとします。この初期プログラムはすべての例外状態を監視し、またはこのプログラムには例外状態が生じた場合に、サインオフ・オプションを *LIST に変更するための**変数変更 (CHGVAR)** コマンドが含まれています。

```
        PGM
        DCLF MENU
        DCL &SIGNOFFOPT TYPE(*CHAR) LEN(7) VALUE(*NOLIST)
        .
        .
        .
        MONMSG MSG(CPF0000) EXEC(GOTO ERROR)
PROMPT: SNDRCVF RCDFMT(PROMPT)
```

```
         CHGVAR &IN41 '0'
         .
         .
         .
         IF (&OPTION *EQ '90') SIGNOFF LOG(&SIGNOFFOPT)
         .
         .
         .
         GOTO PROMPT
ERROR:   CHGVAR &SIGNOFFOPT '*LIST'
         CHGVAR &IN41 '1'
         GOTO PROMPT
         ENDPGM
```

上記の例では例外状態が生じた場合には、CHGVAR コマンドにより SIGNOFF コマンドのオプションが *LIST に変更され、標識がオンに設定されます。この標識は、予期しない状態が生じたこと、およびそれに対してどのような処置をとるべきかを、ディスプレイ装置ユーザーに知らせるメッセージとして表示する固定情報の条件付けに使用することができます。

対話式ジョブで CL プログラムまたは CL プロシージャーが実行されている場合には、その CL コマンドがログに記録されるのは、ログ・レベルが 3 または 4 で、かつ以下のどちらかの条件が満たされている場合だけです。

- **制御言語プログラム作成 (CRTCLPGM)** コマンド、**制御言語モジュール作成 (CRTCLMOD)** コマンド、または**バインド CL プログラム (CRTBNDCL)** コマンドで LOG(*YES) を指定した場合。
- **制御言語プログラム作成 (CRTCLPGM)** コマンド、**制御言語モジュール作成 (CRTCLMOD)** コマンド、または**バインド CL プログラム (CRTBNDCL)** コマンドで LOG(*JOB) を指定し、現行の LOGCLPGM ジョブ属性が (*YES) である場合。

  LOGCLPGM ジョブ属性は SBMJOB、CRTJOBD、CRTJOBD、および CHGJOBD の各コマンドの LOGCLPGM パラメーターを使用して、設定および変更することができます。

バッチ・ジョブ・ログの考慮事項:

バッチ・ジョブ・ログを使用するときは、これらの事柄を考慮してください。

バッチ・アプリケーションの場合に、ログに記録される情報の量を変更したい場合があります。 IBM 提供のサブシステム QBATCH に関するジョブ記述で指定されているログ・レベル (LOG(4 0 *NOLIST)) の場合には、ジョブが異常終了すると詳細なログが提供されます。ジョブが正常に完了した場合には、ジョブ・ログは作成されません。

どのような場合にもジョブ・ログが印刷されるようにする場合には、**ジョブ記述変更 (CHGJOBD)** コマンドを使用してジョブ記述を変更するか、あるいは**バッチ・ジョブ (BCHJOB)** または**ジョブ投入 (SBMJOB)** コマンドで異なる LOG の値を指定します。ロギング・レベルについては、 620 ページの『ジョブ・ログ』を参照してください。

バッチ・ジョブで CL プログラムまたは CL プロシージャーを実行する場合に、その CL コマンドが記録されるのは、以下のコマンドを使用してモジュールかプログラムを作成する際に LOG(*YES) を指定した場合です。

- **制御言語プログラム作成 (CRTCLPGM)**
- **制御言語モジュール作成 (CRTCLMOD)**
- **バインド制御言語プログラム作成 (CRTBNDCL)**

CL コマンドは、CHGJOB コマンドと SBMJOB コマンドを使用する際に LOGCLPGM(*YES) を指定した場合にも、ログに記録されます。

ジョブ・ログ出力制御 **API** によるメッセージのフィルター操作**:**

QMHCTLJL API を使用しジョブ・ログをデータベース・ファイルに書き込む場合、追加のメッセージ・フィルターを指定できます。

この API を使用したメッセージのフィルターの指定は、ジョブが終了してメッセージのレコードがファイルに書き込まれる時点で適用されます。この時点までに、フィルターしようとしているメッセージは表示されています。したがって、このメッセージをジョブの実行中に見ることができます。ジョブ・ログが書き込まれる時点では、フィルターされるメッセージには、そのメッセージ用のファイルに書き込むレコードは存在していません。したがって、メッセージがジョブの実行中に表示されていても、そのメッセージは最終的に作成されるファイルに入れられるわけではありません。

ジョブ・ログ出力ファイル**:**

ジョブ・ログ出力ファイルを使用する際は、出力ファイルのレコード形式を記述するファイルおよびモデル・ファイルに必要なジョブ・ログ情報を検討してください。

関連概念:

620 ページの『ファイルへのジョブ・ログの書き込み』
ジョブの終了時に、ジョブ・ログを出力ファイル QPJOBLOG かデータベース・ファイルに書き込むことができます。

ジョブ・ログの作成*:*

ジョブ・ログ制御 (QMHCTLJL) API または **ジョブ・ログ表示 (DSPJOBLOG)** コマンドを使用して、1 つまたは 2 つのデータベース・ファイルに、ジョブのジョブ・ログを作成できます。

最初のデータベース・ファイルが 1 次ジョブ・ログ・ファイルになります。このファイルにはメッセージに関する必須情報が含まれています。例えばメッセージ識別コード、メッセージ・タイプ、メッセージの重大度などが該当します。処理用に選択したメッセージごとに、1 つのレコードがジョブ・ログ・ファイル内に作成されます。 2 つ目のファイルは 2 次ジョブ・ログ・ファイルです。 QMHCTLJL API を使用する場合に限り、このファイルを作成できます。ただし、これはオプションです。

2 次ジョブ・ログ・ファイルにはメッセージの 1 次レベルと 2 次レベルのテキストが含まれています。このテキストは印刷形式です。メッセージはすべてメッセージ記述と組み合わされ、その結果は 1 つまたは複数の印刷行に様式化されます。処理用に選択したメッセージごとに、複数のレコードを 2 次ジョブ・ログ・ファイル内に作成できます。1 次レベルと 2 次レベルの印刷行ごとに 1 つのレコードを作成できます。

メッセージ参照キーを使用すると、1 次ファイル内のレコードと 2 次ファイル内のレコードを関連付けることができます。 1 次ファイル内の各レコードには、関連メッセージのメッセージ参照キー (MRK) のフィールドが含まれています。 同様に、各 2 次ファイル・レコードには、関連メッセージの MRK が含まれています。メッセージの MRK は、ジョブのコンテキスト内で固有です。 1 次ファイル・レコードの MRK が認識されると、関連する 2 次レコードを即時に識別できます。その理由は、その 2 次レコードにも同一の MRK 値が入れられるからです。

1 次ジョブ・ログ・モデル*:*

IBM 提供の 1 次ジョブ・ログ・ファイルのモデルは、ライブラリー QSYS 内の QAMHJLPR です。 1 次レコード様式は QMHPFT です。

この様式に関する詳細を次に記述します。

フィールドの順序	フィールド名	データ・タイプ	長さ (バイト単位)	フィールドの説明
1	QMHJDT	DATE	10	ジョブ・ログが作成された日付
2	QMHJTM	TIME	8	ジョブ・ログが作成された時刻
3	QMHMRK	CHAR	4	メッセージ参照キー
4	QMHTYP	CHAR	10	メッセージ・タイプ
5	QMHSEV	BIN	4	メッセージ重大度
6	QMHMID	CHAR	7	メッセージ識別コード
7	QMHDAT	DATE	10	メッセージ送信日付
8	QMHTIM	TIME	8	メッセージ送信時刻
9	QMHMF	CHAR	20	メッセージ・ファイル名
10	QMHRPY	CHAR	4	応答参照キー
11	QMHRQS	CHAR	1	要求メッセージ状況
12	QMHSTY	CHAR	1	送信プログラムのタイプ
13	QMHRTY	CHAR	1	受信プログラムのタイプ
14	QMHSSN	BIN	4	送信プログラムのステートメント数
15	QMHRSN	BIN	4	受信プログラムのステートメント数
16	QMHCID	BIN	4	メッセージ・データまたは即時メッセージの CCSID
17	QMHPRL	CHAR	1	メッセージ・パーコレート標識
18	QMHSPR	VAR CHAR	256 MAX	送信プロシージャーの名前
19	QMHSMD	CHAR	10	送信モジュールの名前
20	QMHSPG	CHAR	12	送信プログラムの名前
21	QMHSLB	CHAR	10	送信ライブラリーの名前
22	QMHSTM	CHAR	30	送信プログラムのステートメント番号 (1 つまたは複数)
23	QMHRPR	VAR CHAR	256 MAX	受信プロシージャーの名前
24	QMHRMD	CHAR	10	受信モジュールの名前
25	QMHRPG	CHAR	10	受信プログラムの名前
26	QMHRLB	CHAR	10	受信プログラム・ライブラリーの名前
27	QMHRTM	CHAR	30	受信プログラムのステートメント番号 (1 つまたは複数)
28	QMHSYS	CHAR	8	システム名
29	QMHJOB	CHAR	26	修飾ジョブ名
30	QMHMDT	VAR CHAR	3000 MAX	メッセージ・データまたは即時メッセージ
31	QMHCSP	VAR CHAR	4096 MAX	送信プロシージャーの完全名
32	QMHCRP	VAR CHAR	4096 MAX	受信プロシージャーの完全名
33	QMHLSP	VAR CHAR	6144 MAX	送信プログラムのロング名
34	QMHTID	CHAR	8	スレッド
35	QMHMSC	ZONED	6,0	マイクロ秒
36	QMHFUS	CHAR	10	発信元ユーザー

このレコード内のフィールドの定義は次のとおりです。

QMHJDT

ジョブ・ログが作成された日付; DATE(10)

ジョブ・ログの作成を始めた日付。このフィールドは、データベース・レコード内の日付フィールドです。日付の形式は *ISO です。この日付フィールド内の値の形式は yyyy-mm-dd です。同一のジョブ・ログ用に作成されたレコードについては、このフィールド内の値は同一です。

QMHJTM
ジョブ・ログが作成された時刻; TIME(8)

ジョブ・ログの作成を始めた時刻。このフィールドは、データベース・レコード内に時刻フィールドとして定義されています。時刻の形式は *ISO に定義されています。この時刻フィールド内の値の形式は hh.mm.ss です。同一のジョブ・ログ用に作成されたレコードについては、このフィールド内の値は同一です。

QMHMRK
メッセージ参照キー; CHAR(4)

関連メッセージがジョブ・メッセージ待ち行列内で持っていたメッセージ参照キー。レコードは、メッセージ参照キーに基づいて厳密に昇順に 1 次データベース・ファイルに入れられます。 1 つのジョブ・ログ用に作成されたレコードの集合ではこのフィールドは各レコードごとに固有なので、レコードの固有キーとして使用できます。複数のジョブ・ログ用のレコードが同一のメンバーに入れられると、キーは固有でなくなることがあります。

QMHTYP
メッセージ・タイプ; CHAR(10)

関連メッセージのメッセージ・タイプ。このフィールドには、次のいずれかの特殊値が入れられます。

***CMD**
CL プログラムの実行によってログに記録されたコマンド

***COMP**
完了メッセージ・タイプ

***COPY**
送信側のコピー・メッセージ・タイプ

***DIAG**
診断メッセージ・タイプ

***ESCAPE**
エスケープ・メッセージ・タイプ

***INFO**
情報メッセージ・タイプ

***INQ** 照会メッセージ・タイプ

***NOTIFY**
通知メッセージ・タイプ

***RQS** 要求メッセージ・タイプ

***RPY** 応答メッセージ・タイプ

QMHSEV
メッセージ重大度; BIN(4)

メッセージの重大度。この値の範囲は、0 から 99 です。

QMHMID
メッセージ識別コード; CHAR(7)

メッセージのメッセージ識別コード。メッセージが即時メッセージで識別コードがない場合は、このフィールドには特殊値 *IMMED が入れられます。

QMHDAT

メッセージ送信日付; DATE(10)

メッセージが送られた日付。このフィールドは、日付フィールドとしてデータベース・レコード内に定義されています。日付の形式は *ISO です。このフィールド内の値の形式は yyyy-mm-dd です。

QMHTIM

メッセージ送信時刻; TIME(8)

メッセージが送られた時刻。このフィールドは、時刻フィールドとしてデータベース・レコード内に定義されています。時刻の形式は *ISO に定義されています。このフィールド内の値の形式は hh.mm.ss です。

QMHMF

メッセージ・ファイル; CHAR(20)

メッセージ・ファイルの名前。このファイルを使用してメッセージのメッセージ記述を獲得します。このフィールドの最初の 10 文字はメッセージ・ファイルの名前です。次の 10 文字はライブラリー名です。 QMHMID フィールドに *IMMED (即時メッセージを示す) が入っている場合は、このフィールドはすべてブランクです。

QMHRPY

応答参照キー; CHAR(4)

- メッセージのメッセージ・タイプが照会、通知、または送信側のコピーである場合は、関連応答メッセージのメッセージ参照キー。
- 応答メッセージが使用できない場合は、このフィールドにはヌル値 ('00000000'X) が入れられる。
- メッセージのメッセージ・タイプが照会、通知、または送信側のコピーでない場合も、このフィールドにはヌル値が入れられる。

メッセージ参照キーに基づいて厳密に昇順に保守されるので、応答メッセージのレコードは照会、通知、または送信側のコピー・メッセージのレコードの直後に置かれないことがあります。

QMHRQS

要求メッセージ状況; CHAR(1)

- メッセージ・タイプが *RQS の場合は、要求メッセージが実行されたかされなかったかを示す標識。
- 標識がゼロ ('F0'X) に設定されている場合、要求は実行されなかった。
- 標識が 1 ('F1'X) に設定されている場合、要求は実行された。

メッセージ・タイプが *RQS でない場合、この標識は常にゼロになります。

QMHSTY

送信プログラムのタイプ; CHAR(1)

送信プログラムがオリジナル・プログラム・モデル (OPM) プログラムと統合言語環境 (ILE) プログラムのどちらだったかを示す以下の値のある標識。

- 標識がゼロ ('F0'X) に設定されている場合、送信プログラムは OPM または名前が 12 文字以内のシステム・ライセンス内部コード (SLIC) プログラム。プログラム名は、QMHSPG および QMHLSP フィールドにあります。
- 標識が 1 ('F1'X) に設定されている場合、送信プログラムはプロシージャー名が 256 文字以内の ILE プログラム。プロシージャー名は、QMHSPR および QMHCSP フィールドにあります。

- 標識が 2 ('F2'X) に設定されている場合、送信プログラムはプロシージャー名が 257 文字以上 4096 文字以下の ILE プログラム。送信プロシージャーの完全名は QMHCSP フィールドにあり、QMHSPR フィールドはブランクです。
- 標識が 3 ('F3'X) に設定されている場合、送信プログラムは名前が 13 文字以上 256 文字以下の SLIC プログラム。送信プログラムの完全名は QMHLSP フィールドにあり、QMHSPG フィールドはブランクとなります。

QMHRTY

受信プログラムのタイプ; CHAR(1)

受信プログラムのタイプを示す標識。

- 標識がゼロ ('F0'X) に設定されている場合、受信プログラムは OPM プログラム。プログラム名は、QMHRPG フィールドにあります。
- 標識が 1 ('F1'X) に設定されている場合、受信プログラムはプロシージャー名が 256 文字以下の ILE プログラム。プロシージャー名は、QMHRPR および QMHCRP フィールドにあります。
- 標識が 2 ('F2'X) に設定されている場合、受信プログラムはプロシージャー名が 257 文字以上 4096 文字以下の ILE プログラム。完全なプロシージャー名は、フィールド QMHCRP にあります。フィールド QMHRPR はブランクになります。

QMHSSN

送信プログラムのステートメント数; BIN(4)

送信プログラムのステートメント番号の数。

- 送信プログラム・タイプ・フィールド QMHSTY がゼロ ('F0'X) または 3 ('F3'X) の場合、このフィールドの値は 0 か 1。
- 送信プログラム・タイプ・フィールドが 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドの値は 0、1、2、または 3。

このフィールドの値は、QMHSTM フィールド内にあるステートメント番号の個数を定義します。

QMHRSN

受信プログラムのステートメント数; BIN(4)

受信プログラムのステートメント番号の数。

- 受信プログラム・タイプ・フィールド QMHRTY がゼロ ('F0'X) の場合は、このフィールドの値は 0 か 1。
- 受信プログラム・タイプ・フィールドが 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドの値は 0、1、2、または 3。このフィールドの値は、QMHRTM フィールド内にあるステートメント番号の個数を定義します。

QMHCID

CCSID; BIN(4)

QMHMDT フィールドに入っているメッセージ・データまたは即時メッセージの CCSID。

QMHPRL

メッセージ・パーコレート標識; CHAR(1)

メッセージが受信プログラムにパーコレートされたかどうかを示す標識。

- メッセージがパーコレートされなかった場合、この標識はゼロ ('F0'X)。
- メッセージがパーコレートされた場合、この標識は 1 ('F1'X)。

メッセージのパーコレーションは ILE プログラム内に限り起こります。 したがって、受信プログラム・タイプ・フィールド QMHRTY が 1 ('F1'X) または 2 ('F2'X) の場合に限り、このフィールドは 1 になります。

QMHSPR

送信プロシージャーの名前; VAR CHAR(*)

- 送信プログラム・タイプ・フィールド QMHSTY がゼロ ('F0'X) または 3 ('F3'X) の場合、このフィールドの値は *N。
- 送信プログラム・タイプ・フィールド QMHSTY が 1 ('F1'X) の場合、このフィールドに送信 ILE プロシージャー名が入る。この名前の長さの最大値は 256 文字です。
- 送信プログラム・タイプ・フィールド QMHSTY が 2 ('F2'X) の場合、このフィールドはブランクで、QMHCSP フィールドに完全な名前が入る。

このフィールドには、送信プログラム・タイプが 1 ('F1'X) または 2 ('F2'X) のネストしたプロシージャー名を入れることができます。各プロシージャーは、コロンで区切られます。最も外側のプロシージャー名が最初に識別され、そこに含まれるプロシージャーが後に続きます。最も内側のプロシージャーは、ストリングの最後で識別されます。

QMHSMD

送信モジュールの名前; CHAR(10)

- 送信プログラム・タイプ・フィールド QMHSTY がゼロ ('F0'X) または 3 ('F3'X) の場合、このフィールドの値は *N。
- 送信プログラム・タイプ・フィールド QMHSTY が 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドに送信 ILE モジュール名が入る。

QMHSPG

送信プログラムの名前; CHAR(12)

- 送信プログラム・タイプ・フィールド QMHSTY がゼロ ('F0'X)、1 ('F1'X)、または 2 ('F2'X) の場合、このフィールドにはメッセージ送信元のプログラム名が入る。
- 送信プログラム・タイプが 3 ('F3'X) の場合、このフィールドはブランクになり、QMHLSP フィールドに送信プログラム名が入る。

QMHSLB

送信ライブラリーの名前; CHAR(10)

送信プログラムがあったライブラリーの名前。

QMHSTM

送信プログラムのステートメント番号 (1 つまたは複数); CHAR(30)

送信プログラムがメッセージを送ったステートメントの番号 (1 つまたは複数)。各ステートメント番号の長さは 10 文字です。

- 送信プログラム・タイプ・フィールド QMHSTY がゼロ ('F0'X) または 3 ('F3'X) の場合、最初の 10 文字に最大 1 つのステートメント番号が入る。ステートメント番号は MI 命令の番号を表します。 この番号は 16 進数です。
- 送信プログラム・タイプ・フィールドが 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドには 0 から 3 個のステートメント番号が入る。フィールド QMHSSN はこの個数を指定します。この場合、ステートメント番号は MI 命令の番号ではなく高水準言語ステートメントの番号です。 各番号は 10 進数です。

QMHRPR

受信プロシージャーの名前; VAR CHAR(*)

- 受信プログラム・タイプ・フィールドがゼロ ('F0'X) の場合、このフィールドの値は *N。
- 受信プログラム・タイプ・フィールド QMHRTY が 1 ('F1'X) の場合、このフィールドに ILE プロシージャー名が入る。この名前の長さの最大値は 256 文字です。
- 受信プログラム・タイプ・フィールド QMHRTY が 2 ('F2'X) の場合、このフィールドはブランクで、QMHCRP フィールドに完全な名前が入る。

このフィールドには、受信プログラム・タイプが 1 ('F1'X) または 2 ('F2'X) のネストしたプロシージャー名を入れることができます。各プロシージャーは、コロンで区切られます。最も外側のプロシージャー名が最初に識別され、そこに含まれるプロシージャーが後に続きます。最も内側のプロシージャーは、ストリングの最後で識別されます。

QMHRMD

受信モジュールの名前; CHAR(10)

- 受信プログラム・タイプ・フィールドがゼロ ('F0'X) の場合、このフィールドの値は *N。
- 受信プログラム・タイプ・フィールド QMHRTY が 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドには受信 ILE モジュール名が入る。

QMHRPG

受信プログラムの名前; CHAR(10)

メッセージ送信先の OPM プログラムまたは ILE プログラムのプログラム名。

QMHRLB

受信ライブラリーの名前; CHAR(10)

受信プログラムがあったライブラリーの名前。

QMHRTM

受信プログラムのステートメント番号 (1 つまたは複数); CHAR(30)

メッセージが送られたさいに受信プログラムが停止したステートメントの番号 (1 つまたは複数)。各ステートメント番号の長さは 10 文字です。

- 受信プログラム・タイプ・フィールド QMHRTY がゼロ ('F0'X) の場合、最初の 10 文字に最大 1 つのステートメント番号が入る。ステートメント番号は MI 命令の番号を表します。 この番号は 16 進数です。
- これ以外の受信プログラム・タイプの場合、このフィールドには 0 から 3 個のステートメント番号が入ります。 QMHRSN フィールドはこの個数を指定します。この場合、ステートメント番号は MI 命令の番号ではなく高水準言語ステートメントの番号です。 各番号は 10 進数です。

QMHSYS

システム名; CHAR(8)

ジョブ・ログが作成されたシステムの名前。

QMHJOB

修飾ジョブ名; CHAR(26)

メッセージのログがとられるジョブの完全修飾名。最初の 10 文字にはジョブ名、次の 10 文字にはユーザー名、最後の 6 文字にはジョブ番号が入れられます。

QMHMDT

メッセージ・データまたは即時メッセージ; VAR CHAR(*)

QMHMID フィールドに特殊値 *IMMED がある場合、このフィールドには即時メッセージが入ります。それ以外の場合、このフィールドにはメッセージが送られた時点で使用されたメッセージ・データ

が入ります。このフィールドに入れられる文字の最大値は 3000 文字です。即時メッセージやメッセージ・データが最大値より長い場合は、3000 文字で切り捨てられます。

メッセージ・データにポインターがある場合は、そのメッセージ・データがデータベース・ファイルに書き込まれるまでそのポインターは無効にされます。

QMHCSP

送信プロシージャーの完全名; CHAR(VAR)

- 送信プログラム・タイプがゼロ ('F0'X) または 3 ('F3'X) の場合、このフィールドはブランクになる。
- 送信プログラム・タイプが 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドには ILE プロシージャーの完全名が入る。この名前の長さの最大値は 4096 文字です。

  このフィールドには、各プロシージャー名をコロンで区切って、ネストしたプロシージャー名を入れることができます。最も外側のプロシージャー名が最初に識別され、そこに含まれるプロシージャーが後に続きます。最も内側のプロシージャーは、ストリングの最後で識別されます。

QMHCRP

受信プロシージャーの完全名; CHAR(VAR)

- 受信プログラム・タイプがゼロ ('F0'X) の場合、このフィールドはブランクになる。
- 受信プログラム・タイプが 1 ('F1'X) または 2 ('F2'X) の場合、このフィールドには ILE プロシージャーの完全名が入る。この名前の長さの最大値は 4096 文字です。

  このフィールドには、各プロシージャー名をコロンで区切って、ネストしたプロシージャー名を入れることができます。最も外側のプロシージャー名が最初に識別され、そこに含まれるプロシージャーが後に続きます。最も内側のプロシージャーは、ストリングの最後で識別されます。

QMHLSP

送信プログラムのロング名; CHAR(VAR)

このフィールドには、すべてのプログラム・タイプについて、メッセージ送信元の送信プログラムの完全名が入ります。この名前の長さの最大値は 6144 文字です。

QMHTID

スレッド; CHAR(8)

このフィールドは、メッセージを送信したジョブにあるスレッドを識別します。

QMHMSC

マイクロ秒; ZONED(6,0)

メッセージが送られた時刻のマイクロ秒部分。メッセージが送られた時刻をより正確に判別することができます。

QMHFUS

発信元ユーザー; CHAR(10)

メッセージが送信されたときにスレッドが実行されていたユーザー・プロファイルの名前。

IBM 提供の 2 次ジョブ・ログ・ファイルのモデルは、ライブラリー QSYS 内の QAMHJLSC です。 2 次レコードの様式は QMHSFT です。 2 次レコード様式に関する詳細を次に記述します。

フィールドの順序	フィールド名	データ・タイプ	長さ (バイト単位)	フィールドの説明
1	QMHJDS	DATE	10	ジョブ・ログが作成された日付
2	QMHJTS	TIME	8	ジョブ・ログが作成された時刻
3	QMHMKS	CHAR	4	メッセージ参照キー
7	QMHSYN	CHAR	8	システム名
8	QMHJBN	CHAR	26	修飾ジョブ名
4	QMHLNN	BIN	4	メッセージ行番号
5	QMHSID	BIN	4	テキスト行の CCSID
6	QMHTTY	CHAR	1	メッセージ・テキスト標識
9	QMHLIN	CHAR	78	メッセージ・テキスト行

フィールドの長さは、そのフィールドの合計バイト数を示します。

このレコード内のフィールドの定義は次のとおりです。

QMHJDS

ジョブ・ログが作成された日付; DATE(8)

ジョブ・ログの作成を始めた日付。このフィールドは、データベース・レコード内の日付フィールドです。日付の形式は *ISO です。このフィールド内の値の形式は yyyy-mm-dd です。同一のジョブ・ログ用に作成されたレコードについては、このフィールド内の値は同一です。

QMHJTS

ジョブ・ログが作成された時刻; TIME(8);

ジョブ・ログの作成を始めた時刻。このフィールドは、データベース・レコード内に時刻フィールドとして定義されています。時刻の形式は *ISO に定義されています。このフィールド内の値の形式は hh.mm.ss です。同一のジョブ・ログ用に作成されたレコードについては、このフィールド内の値は同一です。

QMHMKS

メッセージ参照キー; CHAR(4)

関連メッセージがジョブ・メッセージ待ち行列内で持っていたメッセージ参照キー。レコードは、メッセージ参照キーに基づいて昇順に 2 次データベース・ファイルに入れられます。特定のメッセージ参照キーに関する 2 次レコードが複数存在することがあります。このフィールドは、関連する 1 次レコード内にも存在しています。したがって、1 次レコードから参照キーを獲得すると、そのキーを使用して 2 次ファイルから関連レコードを読み取ることができます。

QMHSYN

システム名; CHAR(8)

ジョブ・ログが作成されたシステムの名前。

QMHJBN

修飾ジョブ名; CHAR(26)

メッセージのログがとられるジョブの完全修飾名。最初の 10 文字にはジョブ名、次の 10 文字にはユーザー名、最後の 6 文字にはジョブ番号が入れられます。

QMHLNN

メッセージ行番号; BIN(4)

テキスト・タイプ内の行の行番号。 1 次と 2 次の両方のレベルのテキストに対して、行番号はテキストの先頭行から始まり (番号は 1)、そのレベルの追加行ごとに 1 ずつ増えます。

QMHSID
メッセージ・テキスト行の CCSID; BIN(4)

QMHLIN フィールドに入っているメッセージ・テキスト行の CCSID。

QMHTTY
メッセージ・テキストのタイプ; CHAR(1)

QMHLIN フィールドに 1 次レベルと 2 次レベルのどちらのテキストの行があるかを指定する標識。このフィールドには次のどちらかの値が入れられます。

1 QMHLIN フィールドには 1 次レベルのテキストがある。

2 QMHLIN フィールドには 2 次レベルのテキストがある。

QMHLIN
メッセージ・テキスト行; CHAR(78)

このフィールドには、1 次レベルか 2 次レベルのテキストの行が 1 行入っています。

QHST 活動記録ログ

活動記録ログ (QHST) は、1 つのメッセージ待ち行列と、ログ・バージョンと呼ばれる物理ファイルとによって構成されます。

メッセージは活動記録ログ・メッセージ待ち行列 (QHST) に送られ、システムによって現行 (最新) のログ・バージョンの物理ファイルに書き込まれます。活動記録ログには、システム、サブシステム、ならびにジョブに関する情報、装置の状況、およびシステム・オペレーター・メッセージなど、システムの活動に関するハイレベルのトレース情報が含まれます。

1 つのログ・バージョンがいっぱいになると、ログに新しいバージョンが自動的に作成されます。各バージョンはどれも物理ファイルで、以下の形式の名前が付けられます。

```
Qxxxyydddn
```

ここで、以下が true です。

xxx ログ・タイプを示す 3 文字の記述 (HST)

yyddd ログ・バージョンの最初のメッセージが作成された日付 (年間通算日形式)

n 同一の通算日内での順序番号 (A から Z または 0 から 9)

注**:** 活動記録ログ・バージョン内のレコード数は、システム値 QHSTLOGSIZ で指定されます。また、サイズのみに基づいて新規ファイルを作成するのではなく、日ごとに新規ログ・バージョンを作成できる値 *DAILY もあります。

ログ・バージョン・ファイルのテキストは、ログ・バージョンにおける最初と最後のメッセージの日付と時刻を含みます。最初のメッセージの日付と時刻の位置は、テキストの 1 から 13 桁目です。最後のメッセージの位置は、14 から 26 桁目です。日付と時刻は、cyymmddhhmmss の形式です。以下にそれぞれの内容を示します。

c 世紀保護数字

yymmdd
メッセージが送られた日付

hhmmss
メッセージが送られた時刻

同じ通算日に最大 36 個のログ・バージョンを作成することができます。同じ日に 36 個より多くログ・バージョンを作成すると、次に使用可能な通算日が直後のログ・バージョンに使用されます。そして古いログ・バージョンをいくつか削除すると、名前を再び使用することができます。ログ・バージョンを削除し名前を再び使用すると、名前により順序付けられているログ・バージョンは順序が乱れてしまいます。

活動記録ファイルを処理するプログラムを作成するか、IBM 提供の関数を使用してファイルを処理することができます。IBM は、**ログ表示 (DSPLOG) CL コマンド**および活動記録メッセージ・リストのオープン (QMHOLHST) API を提供します。各ログについてそれぞれいくつかのバージョンが使用可能なので、ユーザーは処理したいログ・バージョンを選択しなければなりません。どのようなログ・バージョンが使用可能であるかを知りたい場合には、 **オブジェクト記述表示 (DSPOBJD)** コマンドを使用してください。例えば、以下のような **DSPOBJD** コマンドを使用すれば、使用可能な活動記録ログのバージョンを表示することができます。

```
DSPOBJD   OBJ(QSYS/QHST*)   OBJTYPE(*FILE)
```

オブジェクト処理 (WRKOBJ) コマンドによって表示される画面の削除オプションを使用して、システムに存在しているログを削除することができます。

また、**ログ表示 (DSPLOG)** コマンドを使用すれば、ログ内の情報を表示または印刷することができます。表示または印刷したい情報を選択するには、以下の事項を組み合わせて指定してください。

- 期間 (時間)
- ログ項目を送ったジョブの名前
- 項目のメッセージ識別コード

以下の **ログ表示 (DSPLOG)** コマンドは、当日のジョブ OEDAILY に関する使用可能なすべての項目を表示します。

```
DSPLOG JOB(OEDAILY)
```

このコマンドによって表示される画面は次のとおりです。

```
                        活動記録ログの内容の表示

ジョブ OEDAILY が開始された。
ライブラリー OELIB のデータベース・ファイル OEMSTR が満了した。
ジョブ OEDAILY が異常終了した。
ジョブ OEDAILY が開始された。
ジョブ OEDAILY が終了した。









        終わり
続行するためには，実行キーを押してください。

F3=終了    F10=すべての表示    F12=取り消し
```

システム日付または時刻を現在値よりも前の値に設定した場合、またはシステム日付と時刻を 48 時間より進めた場合、新しいログ・バージョンが開始されます。これにより 1 つのログ・バージョンのすべてのメッセージが年代順になることが保証されます。

V3R6M0 より前のリリースで作成されたログ・バージョンは、システム日付と時刻が前の値に設定されていた場合に、年代順ではない項目を含みます。したがって、ログ・バージョンを表示しようとした時点で一部の項目の欠落が生じることがあります。例えば、ログ・バージョンの中で 1988 年の項目の後に 1987 年の項目が含まれているとすれば、1987 年の項目を表示しようとして **ログ表示 (DSPLOG)** コマンドの PERIOD パラメーターに 1987 年の日付を指定しても該当の項目は表示されません。日時には常にシステム日付 (QDATE) およびシステム時刻 (QTIME) を使用するか、または PERIOD パラメーターを以下のように指定してください。

```
PERIOD((開始時刻 開始日付) (*AVAIL *END))
```

システム・ログ・メッセージ待ち行列が一定のサイズに達するか、または**ログ表示 (DSPLOG)** コマンドが使用されると、システムはそのメッセージ待ち行列に送られたメッセージを現行ログ・バージョンの物理ファイルに書き込みます。現行ログ・バージョンを最新のものにする場合には、DSPLOG コマンドに架空のメッセージ ID (例えば ###0000 など) を指定してください。メッセージは表示されませんが、メッセージ待ち行列のメッセージは、ログ・バージョン物理ファイルにコピーされ、このファイルを最新にします。

注**:** メッセージが QHST に送信され、**ログ表示 (DSPLOG)** コマンドが即時に実行される場合、 DSPLOG コマンドがログ・バージョン物理ファイルのメッセージを表示します。ログ・バージョンにコピーされているメッセージの数によっては、後で 2 番目の **ログ表示 (DSPLOG)** コマンドが実行されるまでメッセージが表示されないことがあります。

ログ表示 (DSPLOG) コマンドと出力パラメーター *PRINT, (DSPLOG OUTPUT(*PRINT)) を使用してログ情報を印刷する場合、各メッセージの 1 行だけが印刷されます。そのさい、各メッセージの最初の 105 文字が使用されます。

ログ表示 (DSPLOG) コマンドと出力パラメーター *PRTWRAP, (DSPLOG OUTPUT(*PRTWRAP)) を使用してログ情報を印刷する場合、105 文字より長いメッセージは、2000 文字までの制限まで追加の行を組み込むために折り返されます。

ログ表示 (DSPLOG) コマンドと出力パラメーター *PRTSECLVL を使用してログ情報を印刷する場合、105 文字より長いメッセージは、2000 文字までの制限まで追加の行を組み込むために折り返されます。また 2 次レベルのメッセージ・テキストも、使用可能な場合には、最大 6000 文字まで印刷されます。

ログ表示 (DSPLOG) (DSPLOG) コマンドを使用してログ情報を表示する場合、メッセージ・テキストの 105 文字だけが表示されます。 105 文字目以降の文字は、すべて切り捨てられます。

活動記録ログの形式**:**

システムのログ・メッセージ待ち行列に送信されたメッセージを保管する場合は、データベース・ファイルが使用されます。物理ファイル内のレコードがすべて同じ長さであるのに対して、ログに送られてくるメッセージの長さはそれぞれ異なるので、1 つのメッセージが複数のレコードにまたがることもあります。

メッセージに対応する各レコードには、それぞれ 3 つのフィールドがあります。

- システム日付およびシステム時刻 (長さ 8 の文字フィールド)。これは内部フィールドです。この日付と時刻は、変換された上でメッセージの中にも示されます。

- レコード番号 (2 バイトのフィールド)。このフィールドには、例えば最初のレコードについては 16 進数の 0001 が入り、2 番目のレコードについては 16 進数の 0002 が入ります。3 番目以降についても同様です。
- データ (長さ 132 の文字フィールド)。

最初のレコードの 3 番目のフィールド (データ) の形式は以下のとおりです。

内容	タイプ	長さ	レコード内の桁
ジョブ名	CHAR	26	11 から 36
変換された日付と時刻[1]	CHAR	13	37 から 49
メッセージ識別コード	CHAR	7	50 から 56
メッセージ・ファイル名	CHAR	10	57 から 66
ライブラリー名	CHAR	10	67 から 76
メッセージ・タイプ[2]	CHAR	2	77 から 78
重大度コード	CHAR	2	79 から 80
送信側のプログラム名[3]	CHAR	12	81 から 92
送信側のプログラム命令番号[4]	CHAR	4	93 から 96
受信側のプログラム名[3]	CHAR	10	97 から 106
受信側のプログラム命令番号[4]	CHAR	4	107 から 110
メッセージ・テキストの長さ	2 進数	2	111 から 112
メッセージ・データの長さ	2 進数	2	113 から 114
テキストまたはデータのコード化文字セット ID (CCSID)[5]	2 進数	4	115 から 118
送信側のユーザー・プロファイル	CHAR	10	119 から 128
未使用	CHAR	14	129 から 142

注:

[1] 形式: cyymmddhhmmss

ここで、

c 世紀数字 (yy ≥ 40 の場合 c=0、yy < 40 の場合 c = 1)

yymmdd メッセージが送られた年月日

hhmmss メッセージが送られた時分秒

[2] これは**メッセージ受信 (RCVMSG)** コマンドの RTNTYPE パラメーターの値と同じです。

[3] 送信側または受信側が ILE プロシージャーである場合、活動記録ログの項目には ILE プログラム名だけ記録されます。モジュール名およびプロシージャー名は活動記録ログに記録されません。

[4] 送信側または受信側が ILE プロシージャーである場合、送信側または受信側の命令番号は 0 です。

[5] メッセージが保管されているものである場合は、この CCSID は *CCHAR データとして定義されているメッセージ・データだけに適用されます。他のメッセージ・データは 65 535 と見なされます。それ以外のメッセージの場合は、この CCSID は即時メッセージの CCSID です。

2 番目以降のレコードの 3 番目のフィールド (データ) の形式は以下のとおりです。

内容	タイプ	長さ
メッセージ (即時メッセージ・テキストを含む)	CHAR	可変 [1]
メッセージ・データ	CHAR	可変 [2]

注:

[1] この長さは最初のレコード (桁 111 から 112) で指定され、132 を超えることはできません。

[2] この長さは最初のレコード (桁 113 から 114) で指定されます。

ログの新しいバージョンが開始されても、1 つのメッセージが分割されることはありません。つまり、1 つのメッセージの最初と最後のレコードは常に同じ QHST ログ・バージョン・ファイルに入れられます。

QHST ファイル処理**:**

高水準言語 (HLL) プログラムを使用して QHST ファイルを処理する場合は、同じメッセージでも出されるたびに長さが変わる点に注意してください。

メッセージには置換変数が含まれているので、メッセージの実際の長さは可変です。したがって同じメッセージであっても、それを使用するたびにメッセージ・データの開始位置が異なります。

QHST ジョブ開始メッセージおよび完了メッセージ**:**

システムは、ジョブ開始メッセージおよび完了メッセージに対して特殊な形式設定を行います。

メッセージ CPF1124 (ジョブ開始) および CPF1164 (ジョブ完了) の場合、メッセージ・データは常に 3 番目のレコードの 11 文字目から始まります。

ジョブ・アカウンティングを使用すると、CPF1124 および CPF1164 よりも多くの情報が得られます。単純なジョブ・アカウンティング機能には CPF1164 メッセージを使用してください。

パフォーマンスに関する情報は、メッセージ CPF1164 のテキストとしては表示されません。このメッセージは QHST ログに記録されているので、ユーザーはこのデータを検索するためのアプリケーション・プログラムに作成することができます。このパフォーマンス情報の形式は以下のとおりです。

パフォーマンス情報は、可変長の置換テキスト値として渡されます。これは、データが、最初の項目がデータの長さである構造になっていることを意味します。長さフィールドのサイズは、データの長さには含まれません。構造内の最初のデータ・フィールドは、ジョブがシステムに入力され、そのジョブの最初の経路指定ステップが開始された時点の時刻と日付です。時刻は 'hh:mm:ss' の形式で示されます。区切り文字は常にコロンです。日付はシステム値 QDATFMT により定義されている形式で示され、区切り文字にはシステム値 QDATSEP に指定されている文字が使用されます。また、この構造では、まずジョブがシステムに入力された時刻と日付が示され、その後にジョブ開始の時刻と日付が続きます。

ジョブがシステムに入力された時刻と日付は、システムが開始すべきジョブを認識した時点 (すなわちそのジョブに対するジョブ構成を用意する時点) です。対話式ジョブの場合には、ジョブの入力時刻はシステムがパスワードを認識した時刻です。バッチ・ジョブの場合には、これは**バッチ・ジョブ (BCHJOB)** コマンドまたは**ジョブ投入 (SBMJOB)** コマンドが処理された時刻です。監視ジョブ、読み取りプログラム、または書き出しプログラムの場合には、これはそれぞれ該当する開始コマンドが処理された時刻であり、自動開始ジョブの場合にはサブシステムの開始過程のどこかの時点です。

時刻と日付の後には、合計応答時間とトランザクション数が続きます。合計応答時間は、ワークステーションで実行キーを押してから次の画面が表示されるまでに要したジョブの処理時間を累算した値 (秒数) です。この情報は、WRKACTJOB 画面に示される情報に類似しています。このフィールドが意味を持つのは対話式ジョブの場合だけです。

システム障害またはジョブ異常終了が生じた場合には、最後のトランザクションは合計に含まれないこともあります。この場合にはジョブ終了コードは 40 またはそれより大きくなります。トランザクション数は、ジョブの実行中にシステムが累算した応答時間間隔の回数であり、これもコンソール・ジョブ以外の対話式ジョブの場合にだけ意味を持ちます。

トランザクション数の後には、補助記憶域同期入出力操作の回数が示されます。これは、その値がそのジョブの合計である点を除けば、WRKACTJOB の画面に示される AUXIO 欄の値と同じです。ジョブ終了時のジョブ終了コードが 70 であった場合には、最後の経路指定ステップに関するカウントはこの値には含まれないことがあります。さらに、あるジョブの実行途中に IPL がはさまり (**バッチ・ジョブ転送 (TFRBCHJOB)** コマンドを使用した場合)、IPL の後で活動状態になる前にそのジョブが終了した場合には、その値は 0 です。

パフォーマンス関連の統計値の最後のフィールドはジョブ・タイプです。このフィールドの値は次のいずれかです。

A 自動開始ジョブ

B バッチ・ジョブ

I 対話式ジョブ

M サブシステム監視ジョブ

R スプール読み取りプログラム

S システム・ジョブ

W スプール書き出しプログラム

X 開始ジョブ

メッセージ・データの開始位置が可変であるメッセージの場合には、以下の事項を行うことによってそのメッセージ・データにアクセスすることができます。

- メッセージ中の変数の長さを判別します。例えば、あるメッセージで以下の 5 つの変数が使用されているとします。

  ```
  ジョブ名     *CHAR 10
  ユーザー名   *CHAR 10
  ジョブ番号   *CHAR 6
  時刻         *CHAR 8
  日付         *CHAR 8
  ```

  これらの変数はメッセージ・データの最初の 42 桁に固定されます。
- メッセージ・データの位置を知るためには、以下の点を考慮します。
  - メッセージは 2 番目のレコードの 11 文字目から始まります。メッセージ項目が QHST データベース・ファイルにコピーされた時点でメッセージ ID またはメッセージ・ファイルが見つからなかった場合には、ファイル内のレコードは 1 つだけになります。メッセージまたはメッセージ・データを含む 2 つ目のレコードはありません。
  - メッセージの長さは、最初のレコードの 111 文字目から始まる 2 桁のフィールドに示されています。この長さは 2 進数で示されており、例えばメッセージの長さが 60 であるとすれば、このフィールドの値は 16 進数 `003C` になります。

したがって、メッセージの長さとメッセージの開始位置を使用することによって、メッセージ・データの位置を判別することができます。

QHST ファイルの削除**:**

ログ・バージョン物理ファイルはシステムに蓄積されるので、不要になった古いログを定期的に削除するようにしてください。

ログ・バージョンは、機密保護担当者以外は削除できないように作成されています。

操作援助機能 (Operational Assistant) には、古い QHST ファイルの削除を含む終結処置機能があります。 代替手段として、次の方法があります。

機密保護担当者であれば、以下のように指定してください。

```
WRKOBJ  OBJ(QSYS/QHST*)  OBJTYPE(*FILE)
```

不要になった古いファイルを削除するには、オプション 4 を使用してください。

QSYSMSG メッセージ待ち行列

QSYSMSG は、待ち行列に送信される重要なメッセージのリストを処理する場合に作成できる待ち行列です。

QSYSMSG メッセージ待ち行列は、ユーザーが必要に応じて QSYS ライブラリーに作成することのできる待ち行列です。この待ち行列が存在し、しかもそれに損傷がなければ、ある特定のメッセージは QSYSOPR メッセージ待ち行列ではなくこの待ち行列に送られるか、または QSYSOPR メッセージ待ち行列の他にこの待ち行列に送られます。このようにすれば、ある特定のメッセージが送られた場合にユーザー作成のプログラムが制御権を受け取るようにすることができます。 QSYSMSG の待ち行列は、そこで受け取りたい特定のメッセージがない限り作成しないでください。

QSYSMSG 待ち行列を作成するためには、以下のコマンドを入力します。

```
CRTMSGQ  QSYS/QSYSMSG +
         TEXT('Optional MSGQ to receive specific system messages')
```

QSYSMSG メッセージ待ち行列を作成すると、特定のメッセージがすべてこの待ち行列に送られます。ユーザーが特殊な処置を行いたいメッセージだけを受け入れ、その他のメッセージは QSYSOPR メッセージ待ち行列または他のメッセージ待ち行列に送るようにプログラムを作成することができます。このプログラムは中断処理プログラムとして作成する必要があります。

QSYSMSG メッセージ待ち行列へ送られるメッセージ

QSYSMSG メッセージ待ち行列に送られるメッセージは、次のグループに分類できます。 QSYSOPR の代わりに QSYSMSG メッセージ待ち行列に送られるメッセージ、 QSYSMSG、QSYSOPR、またはこの両方に送られるメッセージ、常に QSYSMSG と QSYSOPR の両方に送られるメッセージです。

QSYSMSG メッセージ待ち行列が存在する場合、システムは QSYSOPR の代わりに QSYSMSG に次のメッセージを送信します。

- CPF1269
- CPF1393
- CPF1397
- CPI2209
- CPI9014

- CPI96C0 から CPI96C7

システムは、特定のメッセージを、メッセージと一緒に送られる SRC (システム参照コード) とその SRC が重大メッセージ処理でログに記録されるかどうかに応じて、QSYSMSG、QSYSOPR、または QSYSMSG と QSYSOPR の両方に送信します。これらのメッセージには以下のものがあります。

- CPP0DDD
- CPP1604
- CPPEA02
- CPPEA04
- CPPEA05
- CPPEA12
- CPPEA13
- CPPEA26
- CPPEA32
- CPPEA38
- CPPEA39

システムは、このトピックにあるその他すべてのメッセージを QSYSMSG と QSYSOPR の両方に送信します。

CPD4070

リモート・ロケーション &5 装置記述 &4 から否定応答を受け取った。

CPF0907

重大な記憶域条件が存在する。HELP を押してください。

このメッセージは、システム補助記憶域プールの使用可能な補助記憶域の容量がしきい値に達した場合に送られます。

システム・サービス・ツール機能によって限界値を表示し、それを変更することができます。

CPF097F

ロード・ソース・ディスク装置が有効なロケーションにない。

このメッセージは、ロード・ソース・ディスク装置が有効なロケーションになく、システムが初期プログラムをロードできないときに送られます。

CPF1269

通信装置 &1 で受け取ったプログラム開始要求が理由コード &6、&7 で拒否された。

このメッセージは開始要求が拒否された場合に送られるもので、その拒否の原因を示す理由コードが含まれています。

APPC の使用中に無効なパスワードや無認可の状態が生じた場合、それは通常のジョブのエラー、またはセキュリティーの違反が生じている場合が考えられます。次の処置をとることにより、どのような状況かが判明するまで APPC 装置記述をそれ以上使用できないようにすることも可能です。

- メッセージを QSYSOPR メッセージ待ち行列にも送る。
- 機密保護担当者が検討できるように、その試みを記録しておく。

- モード終了 **(ENDMOD)** コマンドを使用して許容ジョブ数をゼロに設定する。これを行えば、該当の対等装置を現在使用しているジョブはそのまま活動状態ですが、状況が判明するまではその他のジョブは開始できなくなります。
- 一定時間内の無効な試みの回数を数える。無効な試みが何回行われたら、それに対する具体的な処置 (例えば最大セッション数をゼロに変更するなど) をとるかを規定する限界値をプログラムで設定することもできます。この限界値は、作業単位識別コード (ブランクでもよい) ごと、または APPC 装置記述ごと、あるいは APPC 環境全体について設定することができます。

CPF1393

装置 &3 でサブシステム &1 がユーザー・プロファイル &2 を使用できなくした。

このメッセージは、ユーザーがサインオンを数回行ったため、ユーザー・プロファイルが使用不可になった場合に送られます。

CPF1397

サブシステム &1 がユーザー &8 のワークステーション &3 をオフに構成変更した。

このメッセージは、システム値 QMAXSIGN で割り当てられた限界値に達したために、装置がオフに構成変更された場合に送られます。このメッセージは、ユーザーが入力したパスワードが無効であることを示します。 CPF1397 のメッセージ・データには、このメッセージを送り出した装置の名前が示されます。ユーザーはこの情報に基づいて、適切な処置をとるプログラムを作成することができます。このメッセージに対する処置として次のような処置を 1 つ、またはそれらをいくつか組み合わせて実行することが可能です。

- 同じメッセージを QSYSOPR メッセージ待ち行列にも送る。
- 機密保護担当者が検討できるように、その試みを記録しておく。
- 一定の時間の経過後、装置を自動的にオンに構成変更する。

CPF510E

装置 &4 で読み取りまたは書き出しの実行中に、ネットワーク・インターフェース &9 に障害が起こった。

システムはネットワーク・インターフェースの障害を検出し、ネットワーク・インターフェースのエラー・リカバリーを行おうとしています。

要求を再試行してください。問題が再び生じる場合は、 問題分析 **(ANZPRB)** コマンドを入力して、問題分析を実行してください。

CPF5167

リモート・ロケーション &5、装置記述 &4 の SNA セッションが異常終了した。

リモート制御装置から受信した、要求遮断 (RSHUTD)、要求回復 (RQR)、結合解放 (UNBIND)、またはスイッチ・オフの通知 (NOTIFY) コマンドのために、システム・ネットワーク体系 (SNA) セッションが終了しました。

リモート装置オペレーターに連絡して、通信サポートがセッションを終了した理由を判別してください。エラーを訂正して、要求を再試行してください。

CPF5244

リモート・ロケーション &5 装置記述 &4 でシステムの内部的な障害が起こった。

装置をオフに構成変更してください。装置をオンに構成変更して、要求を再試行してください。問題が続く場合は、問題を報告してください (ANZPRB コマンド)。

CPF5248

リモート・ロケーション &5 装置記述 &4 用に受け取ったデータの SNA プロトコル違反。

システム・ネットワーク体系 (SNA) 要求が、リモート・ロケーション &5、装置記述 &4 違反 SNA プロトコルのために受信されました。システムは、制御装置に対するセンス・データ &7 を伴う否定応答を受信しました。

制御装置の問題を訂正して、要求を再試行してください。

CPF5250

リモート・ロケーション &5 にセンス・データ &7 の否定応答を受け取った。

システムは、リモート・ロケーション &5 装置記述 &4 のセンス・データ &7 を伴う否定応答を受信しました。データの最初の 4 文字が、10xx、 08xx、または 0000 で始まっていません。システム・ネットワーク体系 (SNA) セッションが存在する場合、それが終了しました。

エラーを訂正して、要求を再試行してください。センス・データと否定応答の原因の詳細については、「*Systems Network Architecture Formats*」(GA27-3136) を参照してください。

CPF5251

リモート・ロケーション &5 に対する要求のパスワードまたはユーザー ID が正しくない。

システム・ネットワーク体系 (SNA) INIT-SELF コマンドが、有効な権限データのない金融機関リモート・ロケーション &5、装置記述 &4 のために受信されました。次のいずれかが生じました。

- システムは、ユーザー ID またはパスワードを検出できませんでした。
- システムは、ユーザー ID を検出できませんでした。
- パスワードがこのユーザー ID には無効でした。
- 装置記述 &4 を使用するための、ユーザー ID の権限がありません。
- ユーザー・プロファイルにアクセス不可能でした。
- ユーザー ID に無効な文字が含まれていました。

有効なユーザー ID とパスワードを使用して要求を再試行してください。ユーザーが装置に対して権限を持っていない場合、**オブジェクト権限認可 (GRTOBJAUT)** コマンドを使用して、その装置に対する権限を与えてください。

CPF5257

ライブラリー &3 ファイル &2 の装置またはメンバー &4 に障害がある。

読み取りまたは書き込み操作中にエラーが生じました。これが表示装置ファイルである場合、ディスプレイ装置が使用できない場合があります。

上記にリストしたメッセージを参照してエラーを訂正し、要求を再試行してください。問題が続く場合は、問題を報告してください (ANZPRB コマンド)。

CPF5260

ライブラリー &3 のファイル &2 の装置 &4 の交換接続が正常に行われなかった。

ファイルをクローズしてから、要求を再試行してください。

CPF5274

リモート・ロケーション &5 の &3 のファイル &2 の装置でエラー。

プログラムが、前にエラーが生じたプログラム装置 &4、リモート・ロケーション &5 に入力操作または出力操作を行おうとしました。

リモート・ロケーション &5 と関連のある装置をオフに構成変更してから、再びオンに構成変更します (VRYCFG または WRKCFGSTS コマンド)。その後、要求を再試行してください。

CPF5341

リモート・ロケーション &5、装置記述 &4 の SNA セッションは確立されなかった。

システム・ネットワーク体系 (SNA) セッションが確立されませんでした。同期データ・リンク制御 (SDLC) フレーム・サイズには、要求応答単位 (RU) サイズとの互換性がありません。これは構成エラーであるか、または SDLC フレーム・サイズが IBM i オペレーティング・システムによって小さな値と折衝されたかのどちらかです。 このことは、リモート制御装置で交換 ID (XID) コマンドを使用しているときに生じました。

装置記述の MAXLENRU パラメーターには、小売業および金融機関用装置の RU サイズの指定が含まれています。

回線記述の MAXFRAME パラメーターには、SDLC フレーム・サイズ仕様が含まれています。また、制御装置記述の MAXFRAME パラメーターに、小売業および金融機関用装置の指定が含まれています。

次の 1 つまたはそれ以上を行ってから、要求を再試行してください。

- フレーム・サイズが RU サイズと互換性のあることを検査します。
- 必要な場合は SDLC フレーム・サイズを増やすか、または RU サイズを少なくします。
- この構成がリモート制御装置と互換性のあることを検査します。
- 構成変更を行っている場合、この変更を有効にするには、オフに構成変更してからオンに構成変更しなければなりません。

CPF5342

装置記述 &4、リモート・ロケーション &5 で回線 &9 に障害があった。

システムは入力または出力の処理中に回線障害を検出し、この回線のエラー・リカバリーを行おうとしています。

要求を再試行してください。問題が続く場合は、問題分析を開始してください (ANZPRB コマンド)。

CPF5344

制御装置 &9、装置記述 &4 でエラー。

システムは制御装置の障害を検出し、制御装置のエラー・リカバリーを行おうとしています。

要求を再試行してください。問題が続く場合は、問題分析を開始してください (ANZPRB コマンド)。

CPF5346

リモート・ロケーション &5、装置記述 &4 にエラー。

ファイルをクローズします。装置をオフに構成変更します (VRYCFG コマンド)。ジョブ・ログにあるシステム・オペレーターのメッセージを調べて、装置をオンに構成変更する前に行っておく必要のある処置があるかどうかを判別します。エラーを訂正して、装置をオンに構成変更します (VRYCFG コマンド)。その後、要求を再試行してください。 問題が続く場合は、問題分析を開始してください (ANZPRB コマンド)。

CPF5355

タイプ *&9 の &8 にオブジェクト &7 を割り振ることができない。

オブジェクト &7 タイプ *&9 が別の処理で使用されているか、またはオンに構成変更されていないか、または拡張プログラム間通信に利用可能なセッションがないかのいずれかです。

ファイル &2 をクローズします。オブジェクト &7 が利用可能であるかまたはオンに構成変更されているときに、要求を再試行してください。割り振られなかったオブジェクトが APPC 用のものである場合、使用できるセッションがなかったことになります。 WAITFILE パラメーターを変更して、セッションが利用可能になるまでシステムがさらに長く待つことを許可することができま

す。モードを変更して (MAXSSN パラメーター)、より多くのセッションを使用可能にすることもできます。リモート・システムがより多くのセッションを受け入れられるように、再構成する必要があるかもしれません。十分なセッションのための構成がある場合、CHGSSNMAX コマンドを使用して、現行セッション限度を増やすことを試みることもできます。

CPF8AC4

予約されたライブラリー名 &7 は使用中である。

ユーザーが作成したライブラリー名は QDLS ファイル・システム用に予約されています。例えば、ユーザー ASP 5 の場合は、ライブラリー名 QDOC0005 がシステム用に予約されています。システムは、ユーザーがユーザー ASP 上で最初の文書ライブラリー・オブジェクト (DLO) を作成しようとしているときに、このライブラリーの作成を試行します。ただし、ユーザーが予約名 QDOC0005 を使用して独自のライブラリーを作成済みである場合は、メッセージ CPF8AC4 が送信されます。ユーザー作成のライブラリーは、ユーザー ASP で DLO を作成する前に削除または名前変更する必要があります。

CPF9E7C

i5/OS の猶予期間が満了した。

IBM i のソフトウェア・ライセンス猶予期間が満了しました。 次の初期プログラム・ロード (IPL) を正常終了させるには、ソフトウェア・ライセンス・キーが必要です。

新しい i5/OS ソフトウェア・ライセンス・キーについては、IBM 営業担当員または IBM ビジネス・パートナーに連絡してください。 **ライセンス・キー情報の追加 (ADDLICKEY)** コマンドを使用して、ソフトウェア・ライセンス・キーを追加します。

CPI091F

PWRDWNSYS &1 コマンドが進行中である。

このメッセージは、1 次区画が異常終了するときに 2 次区画に送信されます。

CPI0948

ディスク装置 &1 でミラー保護が保留されている。

システムが記憶装置を見つけることができませんでした。データは失われていません。 次の情報はシステム構成から記憶装置が脱落する前に、システムが記憶装置を配置していた位置を示します。

- ディスク製造番号: &5
- ディスク・タイプ: &3
- ディスク型式: &4
- 装置資源名: &26

次のことを行ってください。

1. システム資源構成リスト画面を使用して、脱落しているとして識別されている記憶装置を調べてください。
2. 記憶装置に電源ケーブルが正しく接続されていることを確認してください。

CPI0949

ディスク装置 &1 でミラー保護が保留されている。

ディスクのミラー保護が保留されました。

CPI0950

記憶装置は現在使用可能になった。

システム構成から脱落していた記憶装置が使用可能になりました。データは失われていません。

CPI0953

ASP &5 記憶域が論理しきい値に達した。

このメッセージは、指定された補助記憶域プール (ASP) の使用可能な記憶容量がしきい値に達した場合に送られます。 CPI0953 のメッセージ・データには補助記憶容量、補助記憶使用量、しきい値の比率 (%)、および使用可能な補助記憶域の比率 (%) が示されます。この情報に基づいて適切な処置をとることができます。

CPI0954

ASP &1 記憶域が限界を超えた。

このメッセージは、指定された ASP の使用可能な記憶域がすべて使用された場合に送られます。

CPI0955

システム ASP の非保護記憶域が限界を超えた。

このメッセージは、システム ASP の使用可能な記憶域がすべて使用された場合に送られます。

CPI095A

IASP &1 ミラー・コピー用の記憶域がしきい値に達した。

このメッセージは、指定された独立補助記憶域プール (IASP) のミラー・コピーで使用可能な記憶容量がしきい値に達した場合に送られます。 CPI095A のメッセージ・データには独立補助記憶容量、およびしきい値の比率 (%) が示されます。この情報に基づいて適切な処置をとることができます。

CPI0964

バッテリー低下状態になっている。

このメッセージは、外部の無停電電源装置または内部のバッテリーが機能低下の状態を示している場合に送られます。

CPI0965

システム装置のバッテリー・バックアップ機構に故障がある。

このメッセージは、システム装置のバッテリー・バックアップ機構のバッテリーまたはバッテリー・チャージャーに障害がある場合に送られます。

CPI0966

拡張装置のバッテリー・バックアップ機構に故障がある。

このメッセージは、拡張装置のバッテリー・バックアップ機構のバッテリーまたはバッテリー・チャージャーに障害がある場合に送られます。

CPI096B

地理的ミラーリングが IASP &1 に対し中断された。

このメッセージは、指定された独立補助記憶域プール (IASP) の地理的ミラーリングが、ミラー・コピーを含むシステムとの通信が行えなかったために中断されたことを示すために送られます。指定されたシステムを検査して、これが予期された状況であるか通信問題を示しているかを判別してください。

CPI096C

地理的ミラーリングが IASP &1 に対し、依然として中断されたままである。

このメッセージは、指定された独立補助記憶域プール (IASP) の地理的ミラーリングが、依然として中断されていることを示すために送られます。このメッセージは、以前に中断された独立補助記憶域プールの地理的ミラーリングを再開するための処理が行われていないことを示します。

CPI096D

IASP のミラー・コピーがリジェクトされた。

このメッセージは、独立補助記憶域プール (IASP) のミラー・コピーを構成する処理が、その独立補助記憶域プール (IASP) の実動コピーを含むシステムで試行されたことを示すために送られます。ミラー・コピーは異なるシステム上になければなりません。

CPI096E

ディスク装置の接続が脱落している。

このメッセージは、エンタープライズ・ストレージ・サブシステム (ESS) への予期される接続の一部が報告されていないことを示すために送られます。次の情報によってディスク装置が識別されます。

- ディスク製造番号: &5
- ディスク・タイプ: &3
- ディスク型式: &4

このメッセージは、ケーブルが切断されたか、構成が変更されたか、問題が発生したことを示すことがあります。前のディスク装置情報を使用して、ハードウェア・サービスのマネージャー・サービス・ツールで装置を見つけ、構成の変更であるか問題であるかを判別してください。

CPI0970

ディスク装置 &1 が作動していない。

ディスク装置 &1 の作動が停止しました。 データは失われていません。 次の情報は、作動していないディスク装置を識別するためのものです。

- ディスク製造番号: &3
- ディスク・タイプ: &5
- ディスク型式: &6
- ディスク・アドレス: &4
- IOP 資源名: &26
- 装置制御機構資源名: &27
- 装置資源名: &28

F14 を押して、問題分析を実行してください。

CPI0988

ディスク装置 &1 でミラー保護が再開中である。

このメッセージは、ディスク装置のミラーリングの同期化が開始され、ディスクのミラー保護が再開処理中の場合に送られます。ディスクのミラー保護が再開される前にシステムが行う処置の 1 つに、2 つのディスク装置に同じデータを入れるために 1 つのディスク装置から他のディスク装置にデータを複写する作業があります。データの複写中にシステムのパフォーマンスの低下が認められる場合があります。ディスク・データの複写が完了すると、このメッセージ待ち行列に対してメッセージ CPI0989 が送られ、ディスクのミラー保護が再開されます。

CPI0989

ディスク装置 &1 でミラー保護が再開された。

このメッセージは、ディスク装置のミラーリングの同期化が正常に完了した場合に送られます。システムは、1 つのディスク装置から他のディスク装置へのデータの複写を完了しました。ディスクのミラー保護が再開されます。

CPI0998

ディスク装置 &1 でエラーが起こった。

このメッセージは、ディスク装置 &1 でエラーが検出された場合に送られます。このメッセージには、問題分析を行うための、障害に関する情報は組み込まれません。

CPI0999

記憶域ディレクトリーのしきい値に達した。

記憶域ディレクトリーの容量の限界に近づいてきました。このことは、システムが危険な状態にある可能性を示しています。システムは IPL を受信するまで、このメッセージを繰り返します。

システムで使用している記憶域の量を削減しなければなりません。使用している記憶域の量を削減するには、次のことを行ってください。

- 必要のないオブジェクトをシステムから削除します。
- オンラインである必要のないオブジェクトを保管するために、オブジェクト保管 (SAVOBJ) コマンドに STG(*FREE) を指定します。

CPI099C

重大な記憶域の下限値に達しました。

システム補助記憶域プールで使用されている記憶域量が、下限値の限界に達しました。ここでシステムは、QSTGLOWACN システム値 &5 で指定されている処置を実行します。可能な処置には、次のものがあります。

- *MSG - システムはこれ以上処置をとりません。
- *CRITMSG - システムはメッセージ CPI099B を、CRITMSGUSR サービス属性によって指定されたユーザーに送信します。
- *REGFAC - システムは、QIBM_QWC_QSTGLOWACN 出口点に登録されている出口プログラムを実行するように、ジョブに実行依頼します。
- *ENDSYS - システムは終了し、制限状態になります。
- *PWRDWNSYS - システムは即時に電源遮断を行い、再始動します。

記憶域の使用を削減するには、次の処置を行ってください。

- 未使用のオブジェクトをすべて削除します。
- STG(*FREE) を指定して、オブジェクトを保管します。
- QHST の古いバージョンの使われていないログを保管してから、それらを削除します。
- システム上のスプール・ファイルを印刷するか、または削除します。

記憶域の使用量を削減することに失敗すると、補助記憶装置の初期化が必要になり、ユーザー・データが失われるという事態になることがあります。 WRKSYSSTS コマンドを使用して、使用中の記憶域の量を監視します。 PRTDSKINF コマンドを使用して、記憶域の使用量に関する情報を印刷します。 WRKSYSVAL コマンドを使用すると、補助記憶域下限値 (QSTGLOWLMT) と処置 (QSTGLOWACN) を表示して、変更することができます。

CPI099D

システムが記憶域制限状態で始動中です。

利用可能な記憶域が補助記憶装置の下限を下回っているため、システムが制限状態で開始しました。記憶域の使用量を削減することに失敗すると、補助記憶装置の初期化が必要になり、ユーザー・データが失われるという事態になることがあります。 コンソールが唯一のアクティブ装置です。

記憶域の使用を削減するには、次の処置を行ってください。

- 未使用のオブジェクトをすべて削除します。
- STG(*FREE) を指定して、オブジェクトを保管します。
- QHST の古いバージョンの使われていないログを保管してから、それらを削除します。
- システム上のスプール・ファイルを印刷するか、または削除します。

記憶域の使用量を削減することに失敗すると、補助記憶装置の初期化が必要になり、ユーザー・データが失われるという事態になることがあります。 WRKSYSSTS コマンドを使用して、使用中の記憶域の量を監視します。 PRTDSKINF コマンドを使用して、記憶域の使用量に関する情報を印刷します。 WRKSYSVAL コマンドを使用すると、補助記憶域下限値 (QSTGLOWLMT) と処置 (QSTGLOWACN) を表示して、変更することができます。

CPI099E

記憶域下限出口プログラム・エラーが起こりました。

出口点 QIBM_QWC_QSTGLOWACN のユーザー出口プログラムを呼び出しているときに、エラーが生じました。理由コードは &1 です。理由コードとその意味を次に示します。

1. ユーザー出口プログラムの実行中にエラーが生じました。
2. システムが、ユーザー出口プログラムを見つけられませんでした。
3. システムが登録済みユーザー出口プログラムを見つけられませんでした。
4. ユーザー出口プログラムが、30 分内で終了しませんでした。
5. ユーザー出口プログラムを実行しているジョブが終了しました。
6. システムが終了するため、システムはユーザー出口プログラム・ジョブを実行依頼しませんでした。
7. エラーが生じたため、システムはユーザー出口プログラム・ジョブを実行依頼しませんでした。
8. システムは、ユーザー出口プログラム・ジョブを実行依頼しましたが、警告も発行しました。
9. システムは、出口点の登録情報を検索できませんでした。
10. 失敗した出口プログラム・ジョブの最大数を超えたため、システムはユーザー出口プログラム・ジョブを実行依頼しませんでした。
11. ユーザー出口プログラム・ジョブに予期しないエラーが生じました。

CPI099F

PWRDWNSYS &1 コマンドが進行中である。

このメッセージは、1 次区画が電源遮断するときに 2 次区画に送信されます。

CPI116A

ロード・ソース・ディスク装置でミラー保護が保留されました。

ディスク装置 1 でミラー保護の保留が生じました。データは失われていません。ディスク装置 1 は、多機能 I/O プロセッサー (MFIOP) に接続されています。ディスク装置をできる限り早期に修理してください。ディスク装置 1 の修理を完了するまでは、システムをスイッチ・オフせず、システムの IPL を行わず、システムの IPL を行うことになる操作を実行しないでください。

次の情報は、保留されている装置を識別するためのものです。

- ディスク製造番号: &5
- ディスク・タイプ: &3
- ディスク型式: &4
- 装置資源名: &26

システムは、エラーの修正が行われた後で、自動的にミラーリングを再開します。

CPI116B

ミラー保護はロード・ソース・ディスク装置でまだ保留されています。

ディスク装置 1 でミラー保護が引き続き保留されています。データは失われていません。ディスク装置 1 は、多機能 I/O プロセッサー (MFIOP) に接続されています。ディスク装置をできる限り早期に修理してください。ディスク装置 1 を修理し終えるまでは、システムをスイッチ・オフせず、システムの IPL を行わず、システムの IPL を行うことになる操作を実行しないでください。

次の情報は、保留されている装置を識別するためのものです。

- ディスク製造番号: &5
- ディスク・タイプ: &3
- ディスク型式: &4
- 装置資源名: &26

このメッセージ待ち行列中の、上記にリストされているメッセージを参照して、ミラー保護の延期の原因となる障害を判別してください。推奨されているリカバリー手順を実行してください。

CPI116C

圧縮されたディスク装置 &1 がいっぱいです。

圧縮されたディスク装置 &1 が一時的にいっぱいです。記憶域サブシステム制御装置がこの状態を検出し、圧縮されているディスク装置のデータを位置変更します。システムがこのことを行うのは、ディスク装置上の保管可能データの量を最大にするためです。この操作は、完了するまでに数分かかります。記憶域サブシステム制御装置がデータの位置変更を完了したら、システムは通常の操作を再開します。

次の情報は、いっぱいになっている装置を識別するためのものです。

- ディスク製造番号: &5
- ディスク・タイプ: &3
- ディスク型式: &4
- 装置資源名: &26

記憶域サブシステム制御装置が、圧縮されているディスク装置のデータを位置変更するまで待ってください。システムをスイッチ・オフしないでください。圧縮されているディスク装置がいっぱいであることを示すこのメッセージを頻繁に受け取る場合、次の 1 つまたは複数を行ってください。

1. 補助記憶域プールから必要のないオブジェクトを保管するために、 **オブジェクト保管 (SAVOBJ)** コマンドに STG(*FREE) を指定します。
2. 補助記憶域プールから必要のないオブジェクトを削除します。
3. 1 つまたは複数のフォルダーを別の補助記憶域プールに移動するには、フォルダーを保管し、そのフォルダーを削除し、そしてそのフォルダーを別の補助記憶域プールに復元します。
4. 記憶容量を増やすには、ディスク装置を補助記憶域プールに追加します。システムに、ユーザー補助記憶域プールからシステム補助記憶域プールへと、データを直ちにオーバーフローさせることができます。このことによって、圧縮されているディスク装置がいっぱいになるたびに、記憶域サブシステム制御装置がそのディスク装置のデータを位置変更するまで待つ必要がなくなります。ASP 属性の変更 (CHGASPA) コマンドを使用して、圧縮されているディスク装置がいっぱいになった場合は常に、直ちにデータをシステム補助記憶域プールにオーバーフローさせるように、圧縮回復方針を変更してください。

CPI116F

ホット・スペア・ディスク装置が有効なロケーションにない。

このメッセージは、ロード・ソース・ディスク装置に障害が起こり、このロード・ソース・ディスク装置を保護するホット・スペア・ディスク装置が有効なロケーションになく、システムが初期プログラムをロードできないときに送られます。

CPI1117

ライブラリー &2 の損傷のあるジョブ・スケジュール &1 が削除された。

このメッセージは、ライブラリーにあるジョブ・スケジュールが損傷のために削除された場合に送られます。

CPI1136

ミラー保護はまだ保留されている。

このメッセージは、1 つまたは複数のディスク装置でミラー保護が保留されている場合に、1 時間ごとに送られます。

CPI1138

ASP &1 の記憶域オーバーフローが回復した。

このメッセージは、理由 &2 のためにシステム ASP にオーバーフローされているオブジェクトが ASP &1 になくなったときに送られます。

CPI1139

ASP &1 の記憶域オーバーフローの回復が正常に行われなかった。

このメッセージは、記憶域のオーバーフローの回復の試みが失敗した場合に送られます。

CPI1153

システム・パスワード・バイパス期間が終了した。

このメッセージは、システムがシステム・パスワードの有効なバイパス期間の間稼働していた場合に送られます。そのバイパス期間は終了しました。正しいシステム・パスワードが入力されない限り、次の IPL は正常に完了できません。

CPI1154

システム・パスワード・バイパス期間は &5 日で終了する。

このメッセージは、システム・パスワード (直前の IPL の間に) が入力されなかったかあるいは入力が正しくない場合に送られて、システムのバイパス期間が選択されます。

CPI1159

システム固有識別コードはあと &1 回の導入で満了する。

システム固有識別コードが満了になりそうな場合にこのメッセージが送られます。 IBM サービス技術員に連絡してください。

CPI1160

システム固有識別コードが満了したか、あるいは無効である。

システム固有識別コードが満了になると、このメッセージが送られます。 IBM サービス技術員に連絡してください。

CPI1161

装置パリティー保護のある装置 &1 が完全には作動可能でない。

装置 &1 は、装置パリティー保護のあるディスク装置サブシステムの一部です。装置 &1 には保守が必要です。データは保存されています。この状態が訂正されないと、パフォーマンスの低下、マシン・チェック、データの破損が発生する場合があります。

CPI1162

装置パリティー保護のある装置 &1 が完全には作動可能でない。

装置 &1 は、装置パリティー保護のあるディスク装置サブシステムの一部です。装置 &1 は、次のいずれかの理由のために完全には作動可能でありません。

- サービス技術員が装置を修理中である。
- 装置が作動していないが、問題分析を実行するのに十分な情報がない。

CPI1163

入出力アダプターにホット・スペア・ディスク装置が残っていない。

このメッセージは、入出力アダプターのホット・スペア・ディスク装置がすべて消費されたか、障害が起こった場合に送られます。 データは失われていません。

CPI1164

ホット・スペア・ディスク装置が作動していない。

このメッセージは、ホット・スペア・ディスク装置が作動を停止した場合に送られます。データは失われていません。

CPI1165

1 つ以上の装置パリティー保護された装置が完全には作動可能でない。

装置パリティー保護のあるディスク装置サブシステム内の 1 つまたは複数の装置が、エラーのために完全には作動可能でありません。

CPI1166

装置パリティー保護のある装置が完全に作動可能である。

装置パリティー保護を提供するすべての IOP サブシステムの装置は完全に作動可能です。

CPI1167

一時入出力プロセッサー・エラーが起こった。

ディスク装置の入出力プロセッサーでエラー状態が発生しました。

CPI1168

ディスク装置 &1 でエラーが起こった。

ディスク番号 &1 でエラーが検出されました。オブジェクトの破損が生じる可能性があります。問題が悪化した場合、マシン・チェックが生じる可能性があります。ディスク装置を識別する情報が続きます。

CPI1169

ディスク装置 &1 が作動していない。

ディスク装置 &1 の作動が停止しました。 データは失われていません。

CPI1171

内部システム・オブジェクトを回復することができない。

システムのジョブの索引を含む、内部のシステム・オブジェクトが損傷しています。システムは、&1 回の試行の後でも、オブジェクトを回復できませんでした。

リカバリー処置は必要ありませんが、ジョブの検索パフォーマンスが影響を受けることがあります。

CPI1468

システム・ジョブ・テーブルが容量に近づいている。

このメッセージは、システム・ジョブ・テーブル内の項目数が、許可されている最大数に近づいている場合に送られます。ジョブ・テーブルが完全にいっぱいになるのを許可すると、ジョブの正常な投入またはそれ以後の IPL の完了の妨げになる場合があります。

CPI22AA

監査レコードを QAUDJRN に書き込めません。

タイプ &3 の監査レコードを QAUDJRN 監査ジャーナルに書き込もうとする際に、QSYSAUDR プログラムの命令 &2 で予期しない例外 &1 が発生しました。監査終了処置 (QAUDENDACN) システム値によって指定された処置が実行されます。

CPI2209

ユーザー・プロファイル &1 が損傷しているために削除された。

このメッセージは、ユーザー・プロファイルが損傷を受けたことが原因で削除された場合に送られます。このユーザー・プロファイルは、削除される前にオブジェクトを所有していた可能性があります。そのようなオブジェクトには所有者が存在しなくなります。記憶域再利用 (RCLSTG) コマンドを使用して、そのようなオブジェクトの所有権を QDFTOWN ユーザー・プロファイルに移すことができます。

CPI2239

QAUDCTL システム値が &1 に変更されました。

インストールの際に、機密保護監査機能が利用不能であったため、QAUDCTL は *NONE に変更されました。現在は機密保護監査機能が利用可能であるため、QAUDCTL システム値は元の値に変更されています。

CPI2283

QAUDCTL システム値が *NONE に変更された。

このメッセージは、監査が失敗したためシステムが監査を中止した後、1 時間ごとに送られます。監査を再開するか、または監査が失敗した理由を確認するには、システム値 QAUDCTL の値を *NONE 以外に変更します。

CPI2284

QAUDCTL システム値が *NONE に変更された。

このメッセージは、監査が失敗したためシステムが監査を中止した場合に IPL の間に送られます。監査を再開するか、または監査が失敗した理由を確認するには、システム値 QAUDCTL の値を *NONE 以外に変更します。

CPI8A13

QDOC ライブラリーがシステム・オブジェクトの限界に近づいている。

このメッセージは、ライブラリー QDOC にあるオブジェクトの数が、1 つのライブラリーに保管することのできるオブジェクトの数としてシステムがサポートしている限界に近づいている場合に送られます。

CPI8A14

QDOC ライブラリーが保管活動記録の限界を超えた。

このメッセージは、ライブラリー QDOC にあるオブジェクトの数が、1 つのライブラリーに入れることのできるオブジェクトの数としてシステムがサポートしている限界を超えた場合に送られます。

CPI9014

装置 &1 から受け取ったパスワードが正しくない。

このメッセージは、文書変換セッションで正しくないパスワードを受け取った場合に送られます。無認可でシステムにアクセスしようとしていることを示す場合もあります。

CPI9490

装置 &25 でディスク・エラー。

このメッセージはディスク・エラーが検出された場合に送られます。

CPI94A0

装置 &25 でディスク・エラー。

このメッセージはディスク・エラーが検出された場合に送られます。

CPI94CE

バス拡張アダプター、バス延長アダプター、システム・プロセッサー、またはケーブルでエラーが検出されました。

このメッセージは、システムが主記憶装置に障害を検出したときに送られます。 システム・パフォーマンスは低下することがあります。 問題分析を実行して、障害を起こしているカードを判別してください。

CPI94CF

主記憶装置カードの障害が検出されました。

このメッセージは、システムが主記憶装置に障害を検出したときに送られます。 システム・パフォーマンスは低下することがあります。 問題分析を実行して、障害を起こしているカードを判別してください。

CPI94FC

装置 &25 でディスク・エラー。

このメッセージは、9336 ディスク装置の一部がエラーの限界値を超過し、始動はしたが失敗した場合に送られます。

CPI96C0

保護パスワードを妥当性検査できなかった。

APPC サインオン・トランザクション・プログラムによってユーザー・プロファイルのために受信された保護パスワードが正しくない場合、システムはこのメッセージを送信します。このメッセージには、エラーを識別するための理由コードが含まれています。適切な処置をとるために、理由コードを調べてください。

CPI96C1

サインオン要求 GDS 変数が正しくなかった。

APPC サインオン・トランザクション・プログラムによって受信されるサインオン要求 GDS 変数が正しくないと、システムはこのメッセージを送信します。 リモート・プログラムは、正しいサインオン・データを送信しなければなりません。

CPI96C2

ユーザー・パスワードを変更することができなかった。

このメッセージは、セキュリティーの問題が生じた時に送信されます。

CPI96C3

システム・コールでメッセージ &4 が戻された。

システムがこのメッセージを送信するのは、APPC サインオン・トランザクション・プログラムによってメッセージがシステム・コールで戻されるときです。

CPI96C4

ユーザー・プロファイルのパスワードが正しくない。

システムがこのメッセージを送信するのは、指定されているパスワードが正しくないときです。

CPI96C5

ユーザー &4 が存在していない。

システムがこのメッセージを送信するのは、受信したユーザーがシステム上に存在しないときです。

CPI96C6

CPI 通信に対する呼び出しで戻りコード &4 を受け取った。

システムがこのメッセージを送信するのは、CPI 通信への呼び出しが戻りコードを送信したときです。戻りコードの説明、および障害の原因判別については、「共通プログラミング・インターフェース コミュニケーション・インターフェース解説書」を参照してください。

CPI96C7

APPC サインオン・トランザクション・プログラムでシステムの障害。

システムがこのメッセージを送信するのは、予期しないエラーを受信したときです。 問題分析を実行して、障害を判別してください。

CPP0DD9

システム・プロセッサーの障害が検出された。

このメッセージは、システム・プロセッサーまたはシステム・プロセッサー・キャッシュに障害があった場合に送られます。 システム・パフォーマンスは低下することがあります。

CPP0DDA

スロット 9 でシステム・プロセッサーの障害が検出された。

このメッセージは、システム・プロセッサーまたはシステム・プロセッサー・キャッシュに障害があった場合に送られます。 システム・パフォーマンスは低下することがあります。

CPP0DDB

スロット 10 でシステム・プロセッサーの障害が検出された。

このメッセージは、システム・プロセッサーまたはシステム・プロセッサー・キャッシュに障害があった場合に送られます。 システム・パフォーマンスは低下することがあります。

CPP0DDC

システム・プロセッサーの障害が検出された。

このメッセージは、システムがシステム・プロセッサーでエラーを検出した場合に送られます。システム・パフォーマンスは低下することがあります。

CPP0DDD

システム・プロセッサー診断コードによってエラーが検出された。

このメッセージは、システム・プロセッサー診断が IPL の間に障害を検出した場合に送られますが、システムは機能することができます。 システム・パフォーマンスは低下することがあります。

CPP0DDE

システム・プロセッサー・エラーが検出された。

このメッセージは、システム・プロセッサーで制御の障害が検出された場合に送られます。ハードウェア ECC がその障害を修正中です。しかし、初期プログラム・ロード (IPL) を実行すると制御を初期化することができず、システムはそのプロセッサーを使用せずにシステム自身を再構成することになります。

CPP0DDF

システム・プロセッサーが脱落している。

このメッセージは、あるプロセッサーが複数プロセッサー・システム上で脱落している場合に送られます。

CPP1604

重要 DASD の障害が差し迫っている。直ちに、ハードウェア保守提供者に連絡する。

このメッセージは、データ損失につながる回復不能エラーがディスク装置上で発生しようとしているときに送られます。

CPP29B0

制御装置 &25 の装置で回復限界値を超えた。

このメッセージは、9337 ディスク装置のある部分が障害を起こし始めた場合に送られます。

CPP29B8

制御装置 &25 で RAID 保護が中断された。

このメッセージは、9337 ディスク・アレイのある部分が障害を起こしている場合に送られます。RAID 5 方式の実施に対する保護がそのディスク・アレイで中断しています。

CPP29B9

制御装置 &25 で電源保護が中断された。

このメッセージは、9337 ディスク・アレイにある電源モジュールの 1 つが障害を起こしている場合に送られます。電源保護がそのディスク・アレイで中断しています。

CPP29BA

制御装置 &25 でハードウェア・エラー。

このメッセージは、9337 ディスク・アレイのある部分が障害を起こした場合に送られます。保守処置が必要です。

CPP951B

バッテリー電源装置の障害。

このメッセージは、バッテリー電源装置が障害を起こした場合に送られます。

CPP9522

バッテリー電源装置の障害。

このメッセージは、5042 拡張装置または 5040 拡張装置のバッテリー電源装置が障害を起こした場合に送られます。

CPP955E

バッテリー電源装置が取り付けられていない。

このメッセージは、9406 システム装置の電源供給のバッテリー電源装置が導入されていない場合に送られます。

CPP9575

9406 のバッテリー電源装置を交換する必要がある。

このメッセージは、9406 システム装置のバッテリー電源装置が障害を起こし、交換が必要な場合に送られます。引き続き作動することはできますが、推奨される回数以上の充電・放電のサイクルが起こっています。

CPP9576

9406 のバッテリー電源装置を交換する必要がある。

このメッセージは、9406 システム装置のバッテリー電源装置が障害を起こし、交換が必要な場合に送られます。引き続き作動することはできますが、推奨されるよりも長く導入されています。

CPP9589

バッテリー電源装置のテストが完了した。

このメッセージは、バッテリー電源装置のテストが完了し、結果がログに記録された時点で送られます。

CPP9616

バッテリー電源装置が取り付けられていない。

このメッセージは、バッテリー電源装置が 5042 拡張装置または 5040 拡張装置の電源供給に導入されていない場合に送られます。

CPP9617

バッテリー電源装置を交換する必要がある。

このメッセージは、5042 拡張装置または 5040 拡張装置のバッテリー電源装置の交換が必要な場合に送られます。引き続き作動することはできますが、推奨される回数以上の充電・放電のサイクルが起こっています。

CPP9618

バッテリー電源装置を交換する必要がある。

このメッセージは、5042 拡張装置または 5040 拡張装置のバッテリー電源装置の交換が必要な場合に送られます。引き続き作動することはできますが、推奨されるよりも長く導入されています。

CPP961F

DC 電源モジュール 3 の障害。

このメッセージは、9406 システム装置の DC 電源モジュール 3 が障害を起こした場合に送られます。

CPP9620

DC 電源モジュール 2 の障害。

このメッセージは、9406 システム装置の DC 電源モジュール 2 が障害を起こした場合に送られます。

CPP9621

DC 電源モジュール 1 の障害。

このメッセージは、9406 システム装置の DC 電源モジュール 1 が障害を起こした場合に送られます。

CPP9622

DC 電源モジュール 1 の障害。

このメッセージは、5042 拡張装置または 5040 拡張装置の DC 電源モジュール 1 が障害を起こした場合に送られます。他の DC 電源モジュールもこの障害を起こすことがあります。

CPP9623

DC 電源モジュール 2 の障害。

このメッセージは、5042 拡張装置または 5040 拡張装置の DC 電源モジュール 2 が障害を起こした場合に送られます。他の DC 電源モジュールもこの障害を起こすことがあります。

CPP962B

DC 電源モジュール 3 の障害。

このメッセージは、5042 拡張装置または 5040 拡張装置の DC 電源モジュール 3 が障害を起こした場合に送られます。他の DC 電源モジュールもこの障害を起こすことがあります。

CPPEA02

重要 直ちにハードウェア保守提供者に連絡する。

このメッセージは、例外の内部分析でハードウェアの保守が必要であることが示されたときに送られます。

CPPEA04

重要 ハードウェア保守提供者に連絡する。

このメッセージは、例外の内部分析で、永続的な障害のためにハードウェアの冗長度が失われたことが示されたときに送られます。

CPPEA05

重要 ハードウェア保守提供者に連絡する。

このメッセージは、例外の内部分析で、永続的な障害のためにデータ保護機能が失われたことが示されたときに送られます。

CPPEA12

重要 直ちにハードウェア保守提供者に連絡する。

このメッセージは、例外の内部分析で、I/O カードが削減されたパフォーマンス・レベルで作動していることが示されたときに送られます。

CPPEA13

重要 ハードウェア保守提供者に連絡する。

このメッセージは、例外データの内部分析で、システム・パフォーマンスを維持するためにハードウェアの保守を勧めることを示すときに送られます。 I/O カードのキャッシュ・バッテリー・パックを交換するため、ハードウェア保守提供者に連絡することを勧めます。それができなかった場合は、システム・パフォーマンスが低下します。

CPPEA26

重要 ハードウェア保守提供者に連絡する。

このメッセージは、例外の内部分析で、システム使用可能性を維持するためにハードウェアの保守を勧めることを示すときに送られます。

CPPEA32

ストレージ・サブシステム構成エラー。

このメッセージは、I/O カードの下で構成されている装置の数が多すぎるか種類が正しくないときに送られます。

CPPEA38

重要 直ちにハードウェア保守提供者に連絡する。システム・エラーである。

CPPEA39

重要 直ちにハードウェア保守提供者に連絡する。重大なシステム・エラーである。システムは冗長なリソースを使用して自動的に IPL します。

関連情報:

バックアップおよび回復


Communications Management PDF


Finance Communications Programming (PDF)

例: QSYSMSG からのメッセージの受信

このサンプル・プログラムは、QSYSMSG メッセージ待ち行列からメッセージを受け取ります。

このプログラムは、メッセージを受け取ってメッセージ CPF1269 を処理する単一のプロシージャーで構成されています。 CPF1269 メッセージ中の理由コードは 2 進数形式で示されます。理由コードが 704 または 705 であるかどうかの比較のために、この理由コードを 10 進数値に変換しなければなりません。このプロシージャーでは、**モード終了 (ENDMOD)** コマンドを使用して、事態が判明するまでは新しいジョブが開始されないようにします。その後、ユーザー定義のメッセージ待ち行列にも同じメッセージを送って、機密保護担当者がそれを検討できるようにしています。さらにこのプログラムは、システム・オペレーターにも状況を知らせるメッセージを送ります。別のメッセージを受け取った場合には、そのメッセージはシステム・オペレーターに送られます。

このサンプル・プログラムを呼び出すために、独立した 1 つのジョブが開始されます。このジョブは、活動状態を持続してメッセージの到着を待ちます。また、このジョブは、**ジョブ終了 (ENDJOB)** コマンドによって終了することができます。

注: コーディング例を使用すると、 669 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
/*****************************************************************/
/*                                                               */
/*  Example program to receive messages from QSYSMSG              */
/*                                                               */
/*****************************************************************/
/*                                                               */
/*  Program looks for message CPF1269 with a reason code of 704  */
/*  or 705.  If found then notify QSECOFR of the security failure.  */
/*  Otherwise resend the message to QSYSOPR.                     */
/*                                                               */
/*  The following describes message CPF1269                      */
/*                                                               */
/*      CPF1269:  Program start request received on communications  */
/*                device &1 was rejected with reason codes &6,; &7;  */
/*                                                               */
/*      Message data from DSPMSGD CPF1269                        */
/*                                                               */
/*        Data  type   offset length Description                 */
/*                                                               */
/*          &1   *CHAR      1     10  Device                     */
/*          &2   *CHAR     11      8  Mode                       */
/*          &3   *CHAR     19     10  Job - number               */
/*          &4   *CHAR     29     10  Job - user                 */
/*          &5   *CHAR     39      6  Job - name                 */
/*          &6   *BIN      45      2  Reason code - major        */
/*          &7   *BIN      47      2  Reason code - minor        */
/*          &8   *CHAR     49      8  Remote location name       */
/*          &9   *CHAR     57  *VARY  Unit of work identifier     */
/*                                                               */
```

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

          PGM

          DCL        &MSGID   *CHAR LEN(  7)
          DCL        &MSGDTA  *CHAR LEN(100)
          DCL        &MSG     *CHAR LEN(132)

          DCL        &DEVICE  *CHAR LEN( 10)
          DCL        &MODE    *CHAR LEN(  8)
          DCL        &RMTLOC  *CHAR LEN(  8)

          MONMSG     CPF0000  EXEC(GOTO PROBLEM)
          /*********************************************************/
          /* Fetch messages from QSYSMSG message queue             */
          /*********************************************************/
LOOP:     RCVMSG     MSGQ(QSYS/QSYSMSG) WAIT(*MAX) MSGID(&MSGID) +
                       MSG(&MSG) MSGDTA(&MSGDTA)

          IF         ((&MSGID *EQ 'CPF1269') /* Start failed msg */ +
            *AND      ((%BIN(&MSGDTA 45 2) *EQ 704)                +
              *OR      (%BIN(&MSGDTA 45 2) *EQ 705)) )             +
          THEN(DO)
            /*******************************************************/
            /* Report security failure to QSECOFR                  */
            /*******************************************************/

            CHGVAR     &DEVICE %SST(&MSGDTA 1 10) /* Extract device */
            CHGVAR     &MODE   %SST(&MSGDTA 11 8) /* Extract mode   */
            CHGVAR     &RMTLOC %SST(&MSGDTA 49 8) /* Get loc name   */

            ENDMOD     RMTLOCNAME(&RMTLOC) MODE(&MODE)

            SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
                         TOMSGQ(QSECOFR)

            SNDPGMMSG  MSG('Device ' *CAT &DEVICE *TCAT ' Mode '  +
                         *CAT &MODE *TCAT ' had security failure, +
                         session max changed to zero')            +
                         TOMSGQ(QSYSOPR)
          ENDDO
          ELSE DO
            /*******************************************************/
            /* Other message - Resend to QSYSOPR                   */
            /*******************************************************/

            SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
                         TOMSGQ(QSYSOPR)

            /* SNDPGMMSG would fail if the message does           */
            /*   not have a MSGID or is not in QCPFMSG            */

            MONMSG     MSGID(CPF0000)  +
                       EXEC(SNDPGMMSG MSG(&MSG) TOMSGQ(QSYSOPR))
          ENDDO

          GOTO       LOOP    /* Go fetch next message            */


          /*********************************************************/
          /* Notify QSYSOPR of abnormal end                        */
          /*********************************************************/

PROBLEM:  SNDPGMMSG  MSG('QSYSMSG job has abnormally ended') +
                       TOMSGQ(QSYSOPR)
          MONMSG     CPF0000
```

```
SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
             MSGDTA('Unexpected error occurred')
MONMSG     CPF0000

ENDPGM
```

コードに関するライセンス情報および特記事項

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

いかなる場合においても、IBM および IBM のサプライヤーならびに IBM ビジネス・パートナーは、その予見の有無を問わず発生した以下のものについて賠償責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

© Copyright IBM Corp. 1998, 2018

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プログラミング・インターフェース情報

この「CL 概説および概念」資料には、プログラムを作成するユーザーが IBM i のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、『www.ibm.com/legal/copytrade.shtml』 をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用**:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用**:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

プログラム番号: 5770-SS1

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21