IBM

eserver

iSeries

CL プログラミング

バージョン 5

SD88-5038-06
(英文原典：SC41-5721-06)

IBM

@server

iSeries

CL プログラミング

バージョン 5

SD88-5038-06
(英文原典：SC41-5721-06)

ご注意!

本書および本書で紹介する製品をご使用になる前に、 507 ページの『付録 E. 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 OS/400 (プロダクト番号 5722-SS1) のバージョン 5、リリース 3、モディフィケーション 0 に適用されます。また、改訂版で断りがない限り、それ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。

本書は、SD88-5038-05 の改訂版です。 本書は、RISC システムにのみ適用されます。製品のレベルに合った版であることを確かめてご使用ください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

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: SC41–5721–06
iSeries
CL Programming
Version 5

発 行: 日本アイ・ビー・エム株式会社

担 当: ナショナル・ランゲージ・サポート

第1刷 2004.4

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

図

CL プログラミング (SD88-5038) について

本書では、OS/400 のプログラムに関する次の事項について、その詳細を説明しています。

- 制御言語 (CL) プログラミング
- OS/400 プログラミングの概念
- オブジェクトおよびライブラリー
- メッセージの処理
- ユーザー定義のコマンド
- ユーザー定義のメニュー
- テスト機能

本書の対象読者

本書は、OS/400 のプログラマーおよびアプリケーション・プログラマーを対象として書かれています。 CL プログラミングについては特に詳細に説明していますが、本書の内容の多くは、システムに一般的に適用され、 iSeries サーバーでサポートされている高水準言語のプログラマーにとっても参考になります。

前提情報および関連情報

iSeries Information Center を iSeries 技術情報の開始点として使用します。

以下の 2 つの方法で Information Center にアクセスすることができます。

- 以下の Web サイトから

  `http://www.ibm.com/eserver/iseries/infocenter`

- 「*iSeries Information Center および PDF ライブラリー CD パッケージ* (SK88-8055-03)」CD-ROM から。この CD-ROM は、新規 iSeries ハードウェアまたは IBM Operating System/400 ソフトウェアのアップグレード・オーダーに付属しています。また、この CD-ROM は以下の IBM Publications Center でもオーダーできます。

  `http://www.ibm.com/shop/publications/order`

iSeries Information Center には、ソフトウェアとハードウェアのインストール、Linux、WebSphere、Java、高可用性、データベース、論理区画、CL コマンド、システム・アプリケーション・プログラミング・インターフェース (API) などの新規および更新済み iSeries 情報が収められています。さらに、iSeries ハードウェアおよびソフトウェアの計画、トラブルシューティング、および構成を支援するアドバイザーおよび検索機能も備わっています。

ハードウェアを新規にオーダーするたびに、「*iSeries セットアップおよびオペレーション CD* (SK88-8058-02)」が送付されます。この CD-ROM には、IBM **@server** IBM e(logo)server iSeries Access for Windows および EZ セットアップ・ウィザードが含まれています。 iSeries Access Family は、iSeries サーバーに PC を接続す

る、クライアント / サーバー機能の強力なセットを備えています。EZ セットアップ・ウィザードは、多くの iSeries セットアップ・タスクを自動化します。

関連した情報については、 511 ページの『参照文献』を参照してください。

第 1 章 序

この章ではOS/400 の主要な概念について説明しています。個々の概念の詳細については、第 2 章以降で説明しています。

システムの操作は、以下によって制御されます。

- CL コマンド。 CL コマンドは、バッチ・ジョブおよび対話式ジョブ (コマンド入力画面からのジョブなど) で単独で使用され、また CL プログラムおよびプロシージャーでも使用されます。
- メニュー・オプション。システムの操作は、メニュー・オプションを選択することによって制御することができます。対話方式でタスクを行うユーザーは、iSeries サーバーのメニューを用いて多くのシステム・タスクを実行することができます。
- システム・メッセージ。システム・メッセージは、プログラムとプロシージャーとの間、およびプログラムとプロシージャーとユーザーとの間の通信に使用されます。メッセージによって、状況情報やエラー状態などを知ることができます。

制御言語 (CL)

制御言語 (CL) とは、ユーザーとオペレーティング・システムとの間の最も重要なインターフェースであり、異なるワークステーションのユーザーが同時に CL を使用することができます。また、単一の制御言語ステートメントのことを、**コマンド**と呼びます。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。

特定のコマンドについては、iSeries™ Information Center の『CL』トピックを参照してください。

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、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。

モジュール

モジュールとは、ILE (統合言語環境®) 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、CL モジュールの作成 (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 (たとえば C) では、複数のユーザーが作成した複数のプロシージャーが 1 つのモジュールに含まれることもあります。

プログラム

OS/400® は 2 つのタイプのプログラムをサポートしています。 **ILE プログラム**は、統合言語環境 (ILE) 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 OS/400 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 CL プログラムの作成 (CRTBNDCL) コマンドを使用して作成できます。プログラムの作成 (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。 ILE モジュール、ILE プログラム、サービス・プログラム、およびバインド・ディレクトリーの詳細については、


「ILE 概念」 を参照してください。

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。

サービス・プログラム

サービス・プログラムとは、1 つまたは複数のモジュールを含む OS/400 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。

コマンドの構文

各コマンドは、コマンド名とパラメーターから成っています。コマンド名は通常、動詞 (つまり処置) と、その後にある名詞または語句 (処置の対象となるもの) で構成されています。コマンド名を構成する語は略語 (通常 1 ～ 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。たとえば、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が、このコマンド名は SNDMSG で、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。

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。

CL コマンドおよびキーワードで使用する略語のリストについては、 473 ページの『付録 D. CL コマンドおよびキーワードの省略形』を参照してください。

CL プロシージャー

CL プログラムとプロシージャーは、CL コマンドから成っています。コマンドはコンパイルされて OPM プログラムまたは ILE モジュールとなり、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することができます。 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。たとえば、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、たとえば、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。たとえば、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。

コマンド定義

コマンド定義機能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。

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。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
- 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

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、たと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。

コマンド定義の詳細については、『第 9 章 コマンドの定義』を参照してください。コマンドのオンライン・ヘルプ情報の作成については、『第 10 章 コマンドの資料作成』を参照してください。

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、様々な機能を実行することができます。メニューを使用したシステム・タスクの実行には、次のような利点があります。

- 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。

システム提供のメニューのように使用できるメニューを作成するための手順については、「Application Display Programming」  を参照してください。

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、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。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 (アポストロフィや括弧など) 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。

iSeries 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。 iSeries 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。

OS/400 は、コマンド・ラインから 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・ライン・ウィンドウの表示 (QUSCMDLN) API を使用して、アプリケーションからコマンド・ラインを表示できます。

オブジェクトおよびライブラリー

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。たとえば、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。さらに、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。

オブジェクト

ほとんどの CL コマンドの持つ機能は、オブジェクトを対象として実行されます。コマンドには、任意のタイプのオブジェクトに使用できるものもあれば、特定のオブジェクトにしか使用できないものもあります。

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。たとえば、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。たとえば、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。たとえば、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。

ライブラリー

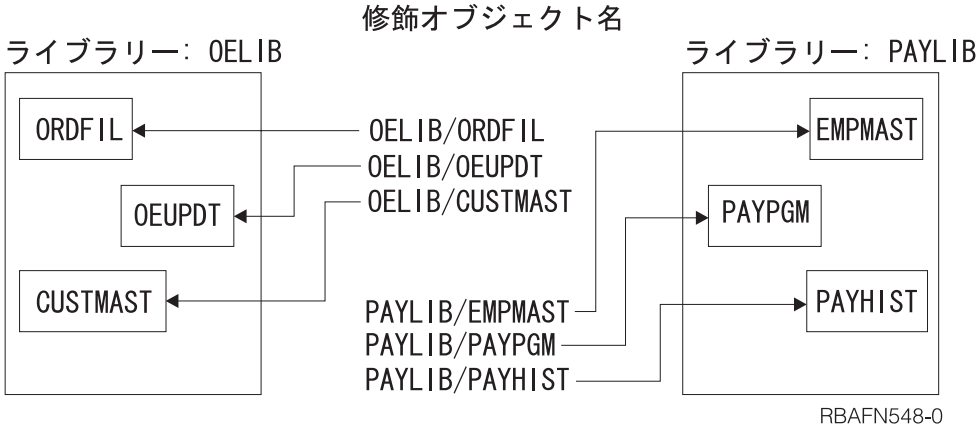
ライブラリーとは、関連するオブジェクトをグループ化し、それらのオブジェクトの使用時に名前で見つけられるようにするために使用されるオブジェクトです。したがって、ライブラリーはオブジェクトのグループの登録簿のようなものです。ライブラリーを使用することによって、オブジェクトを何らかの意味を持つグループにグループ化することができます。たとえば、セキュリティー上の要件、バックアップ上の要件、または処理上の要件などによって類別し、それぞれのオブジェクトのグループを作ることができます。 1 つのライブラリーに入れることのできるオブジェクトの数、およびシステム上のライブラリーの数については、使用可能なディスク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。独立 ASP の詳細については、iSeries Information Center の『独立ディスク・プール』トピックを参照してください。

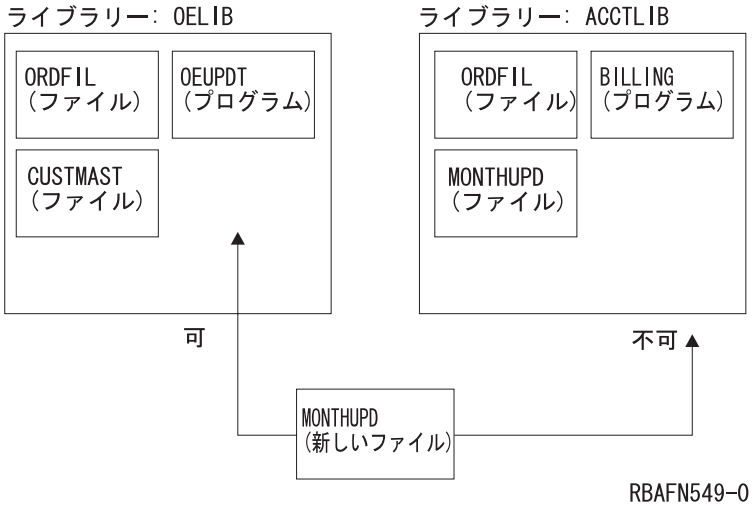
ほとんどのタイプのオブジェクトは、その作成時点でライブラリーに入れられます。CRTLIB コマンドの AUT パラメーターは、ライブラリーの共通認可を定義します。CRTAUT パラメーターは、ライブラリーに作成されるオブジェクトのデフォルト権限を指定します。オブジェクトを作成するコマンドが AUT パラメーターに ***LIBCRTAUT** を指定している場合、オブジェクトの共通認可はライブラリーに指定されていた作成権限になります。ほとんどのタイプのオブジェクトはライブラリーから別のライブラリーに移すことができますが、この場合、1 つのオブジェクトを同時に複数のライブラリーの中に存在させることはできません。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。すでに述べたように、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。たとえば、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すで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。

ライブラリーを使用してオブジェクトを探し出す方法の詳細については、 125 ページの『ライブラリーの使用法』を参照してください。

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。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。 OS/400 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、 OS/400 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。

メニュー、メッセージ、およびメッセージ記述で使用するコード化文字セット識別コード (CCSID) についての情報は、 iSeries Information Center の『**システム概要、計画およびインストール**』カテゴリー内の『*グローバリゼーション (Globalization)*』のトピックを参照してください。

メッセージ記述

メッセージ記述は、 OS/400 に対してメッセージを定義するためのものです。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

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メッセージは、そのプロシージャー、プログラムまたはユーザーの**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。

OS/400 には次のようなメッセージ待ち行列があります。

- 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。

テスト機能

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。この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。このテスト機能は、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。

プロシージャーのソース・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、調べなければならない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。 OS/400 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。
- プログラムの停止が可能な任意の位置で、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。

統合言語環境 (ILE) プログラムのデバッグに関する詳細は、405 ページの『ILE プログラムのデバッグ』を、また OPM プログラムのデバッグに関する詳細は、430 ページの『OPM プログラムのデバッグ』を参照してください。

他の 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。

第 2 章 CL プログラミング

この章で焦点を当てているのは、OPM ではなく ILE です。したがってこの章では、「プロシージャー」が「プログラム」の代わりに使われています。ただし、一般的な CL コマンドについての解説では、やはり「プログラム」という語が使用されている場合もあります。

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。この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呼び出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 プロシージャーの場合もソース・ステートメントを入力し、コンパイルし、バインドして初めて実行可能なプロシージャーになります。

CL コマンドを個別に (たと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には、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。そして、入力したすべてのコマンドは 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。さらに、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実行することができるので、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は向上します。

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 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、または入力装置からのバッチ入力ストリームによって、データベース・ソース・メンバー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って、ILE プログラムを作成することができます。

CL プロシージャーは、たと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 プロシージャーを使用すれば、次のような種々の利点が得られます。以下にその例を示します。

- プログラム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、プログラムは個々のコマンドを個別に入力して実行するよりも実行速度が速くなります。
- CL プロシージャーには柔軟性があります。 CL プロシージャーにパラメーターを渡すことにより、使用時点の特定の要件に合わせてプロシージャーの機能を制御することができます。
- 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 プロシージャーおよびプログラムには、コマンドを個別に入力した場合には使用できない、条件に基づくロジックや特殊な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。

次の目的に CL プロシージャーを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。

CL プログラムの作成方法

CL プログラムはすべて次のステップで作成されます。

1. ソースの作成。 CL プロシージャーは CL コマンドによって構成されます。多くの場合、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・ファイルに入力されます。
2. モジュールの作成。 CL モジュールの作成 (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 HLL 言語では、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。プログラムの作成 (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 CL プログラムの作成 (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 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。

対話式入力

OS/400 には、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、プログラマーを支援するさまざまなメニューと画面があります。 「機密保護解説書」 で説明されている OS/400 セキュリティー機能を使用する場合には、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。

ソースの入力に通常使用されるのは、 WebSphere Development Studio の一部である、原始ステートメント入力ユーティリティー (SEU) 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。

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、ジョブ投入 (SBMJOB) 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。ソースをインラインで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。たとえば、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、CL モジュールの作成 (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。

CL プロシージャーの構成要素

CL プロシージャーのソースとして入力する各ソース・ステートメントは実際には CL コマンドであり、典型的な CL プロシージャーの場合、これらのソース・ステートメントは次のような基本的な構成要素に類別されます。

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、プロシージャー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)

プロシージャー変数の宣言であり、変数を使用する場合には必須です。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。

CL 処理コマンド (通常複数のコマンドが使用される)

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

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、および %BINARY (%BIN)

算術式、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

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、RETURN

他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。この区別については、165 ページの『CL プログラムでのオブジェクトへのアクセス』の項、および各種オブジェクトに関する項で説明します。プロシージャーを正しく実行するには、次のもの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。詳細については、169 ページの『CL プロシージャーでのファイルの処理』を参照してください。
- 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。詳細については、169 ページの『CL プロシージャーでのファイルの処理』を参照してください。
- 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。呼び出しモジュールのコンパイル時にはなくても構いません。詳細については、165 ページの『CL プログラムでのオブジェクトへのアクセス』 および第 3 章を参照してください。

- 他のプロシージャー。 CALLPRC コマンドを使用する場合は、呼び出されるプロシージャーは CRTPGM の実行時には存在している必要があります。 CRTCLMOD の実行時には存在している必要はありません。

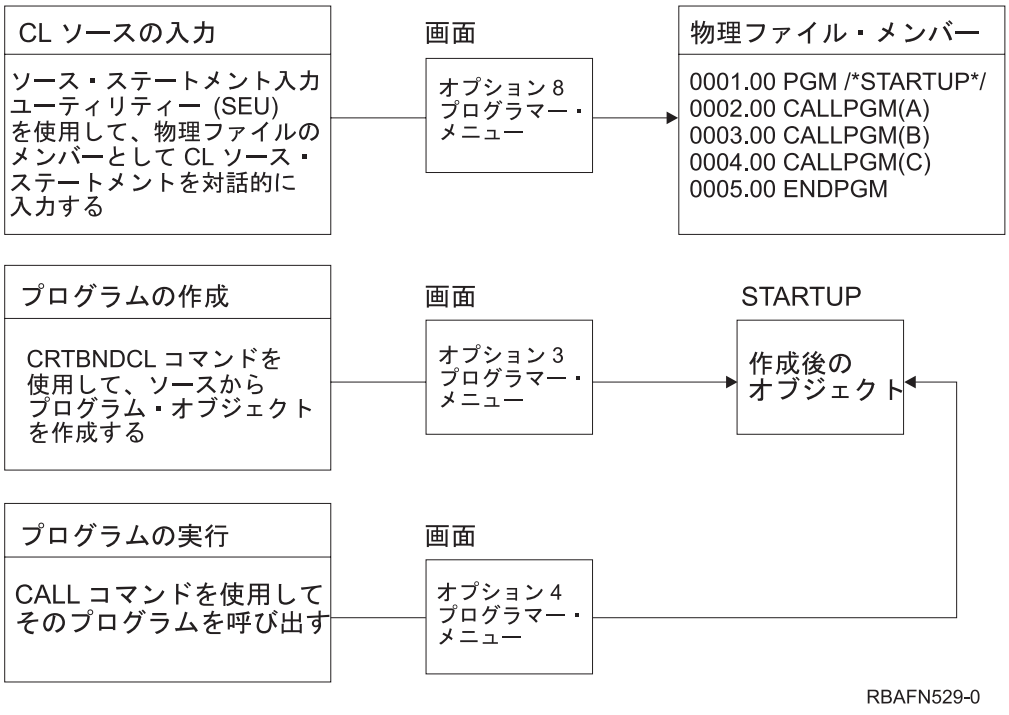
簡単な CL プログラムの例

CL プログラムは、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。たとえば、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 の一部である、プログラム開発管理機能 (PDM) を使用することもできます。

このプログラムを入力、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 ソースの入力は次のように行います。

- 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。)
- 「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。

- 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、次の操作を行ってください。

- F3 キーを押して SEU を終了する。
- 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。
- 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ためには、参照されるプログラムが、CALL コマンドの処理の時点で存在していなければなりません。

CL プロシージャーの中で使用するコマンド

CL プロシージャーに含めることができるのは、CL コマンドだけです。 IBM 提供のコマンド、および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 プロシージャーで使用できないものもあります。個々のコマンドの説明と CL プロシージャーでの適用度については、特定のコマンドのオンライン・ヘルプを参照するか、**iSeries Information Center** の『プログラミング』カテゴリーの『*CL*』セクションを参照してください。

RQSDTA および CMD パラメーターに指定するコマンド

ジョブ転送 (TFRJOB) やジョブ投入 (SBMJOB) コマンドなど、ある特定の CL コマンドには RQSDTA または 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。

CL コマンド

次の表は、CL プロシージャーの中でよく使われるコマンドのリストです。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。必要なコマンドを判別する方法については、**iSeries Information Center** の『プログラミン

グ』カテゴリーにある『*CL*』セクションを参照してください。これらのコマンドの機能に精通しておくと、この章の内容を理解するのに役立ちます。肩文字 1 の付いているコマンドは、CL プログラムおよびプロシージャー内で**だけ**使用できるコマンドです。

システムの機能	コマンド	コマンドの機能
プロシージャー制御の変更	CALL (呼び出し)	プログラムを呼び出す。
	CALLPRC (プロシージャーの呼び出し) [1]	プロシージャーを呼び出す。
	RETURN (戻り)	あるプログラムまたはプロシージャーを実行させたコマンドの次のコマンドに戻る。
CL プロシージャー境界	PGM (プログラム) [1]	CL プロシージャー・ソースの始めを示す。
	ENDPGM (プログラム終了) [1]	CL プロシージャー・ソースの終わりを示す。
CL プロシージャー・ロジック	IF[1]	論理式の値に基づいてコマンドを処理する。
	ELSE[1]	IF コマンドにおける ELSE (偽) 条件の場合にとる処置を定義する。
	DO (DO グループ)[1]	DO グループの始めを示す。
	DOWHILE (Do While)[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。
	DOUNTIL (Do Until)[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。
	DOFOR (Do For)[1]	コマンドを指定した値に基づいてゼロ回以上処理する DO グループの始めを示す。
	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。
	ITERATE (繰り返し) [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。
	ENDDO (DO グループ終了) [1]	DO グループの終わりを示す。
	GOTO (指定先に進む) [1]	他のコマンドに分岐する。
	SELECT (SELECT グループ)[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。
	OTHERWISE (他の場合) [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。
	ENDSELECT (SELECT グループ終了) [1]	SELECT グループの終わりを示す。
CL プロシージャー変数	CHGVAR (変数変更) [1]	CL 変数の値を変更する。
	DCL (宣言) [1]	変数を宣言する。
変換	CHGVAR (変数変更) [1]	CL 変数の値を変更する。
	CVTDAT (日付形式変換) [1]	日付の形式を変更する。
データ域	CHGDTAARA (データ域変更)	データ域を変更する。
	CRTDTAARA (データ域作成)	データ域を作成する。
	DLTDTAARA (データ域削除)	データ域を削除する。
	DSPDTAARA (データ域表示)	データ域を表示する。
	RTVDTAARA (データ域検索) [1]	データ域の内容を CL 変数にコピーする。
ファイル	ENDRCV (受信終了) [1]	RCVF、SNDF、または SNDRCVF コマンドによって前に出された入力要求を取り消す。

システムの機能	コマンド	コマンドの機能
	DCLF (ファイル宣言) [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。
	RCVF (ファイル受信) [1]	表示装置ファイルまたはデータベース・ファイルからレコードを読み込む。
	RTVMBRD (メンバー記述の検索) [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。
	SNDF (ファイル送信) [1]	表示装置ファイルにレコードを書き出す。
	SNDRCVF (ファイル送信 / 受信) [1]	表示装置ファイルにレコードを書き出し、ユーザーが応答した時点でそのレコードを読み取る。
	WAIT (待機) [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	MONMSG (メッセージ・モニター) [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。
	RCVMSG (メッセージ受信) [1]	メッセージ待ち行列から CL プロシージャー内の CL 変数にメッセージをコピーする。
	RMVMSG (メッセージ除去) [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。
	RTVMSG (メッセージ検索) [1]	メッセージ・ファイルから CL 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。
	SNDPGMMSG (プログラム・メッセージ送信) [1]	プログラム・メッセージをメッセージ待ち行列に送る。
	SNDRPY (応答送信) [1]	照会メッセージの送信元に応答メッセージを送る。
	SNDUSRMSG (ユーザー・メッセージ送信)	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かシステム・オペレーターに送る。
その他のコマンド	CHKOBJ (オブジェクト検査)	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。
	PRTCMDUSG (コマンド使用状況の印刷)	指定した CL 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。
	RTVCFGSRC (構成ソースの検索)	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。
	RTVCFGSTS (構成状況検索) [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。
	RTVJOBA (ジョブ属性検索) [1]	1 つまたは複数のジョブ属性の値を検索し、CL 変数に入れる。
	RTVSYSVAL (システム値検索) [1]	システム値を検索し CL 変数に入れる。
	RTVUSRPRF (ユーザー・プロファイル検索) [1]	ユーザー・プロファイルの属性を検索し CL 変数に入れる。
プログラム作成コマンド	CRTCLMOD (CL モジュールの作成)	CL モジュールを作成する。
	DLTMOD (モジュールの削除)	モジュールを削除する。
	DLTPGM (プログラム削除)	プログラムを削除する。
	CRTBNDCL (バインド CL プログラムの作成)	バインド CL プログラムを作成する。
	CRTCLPGM (CL プログラム作成)	OPM CL プログラムを作成する。
	CRTPGM (プログラムの作成)	1 つまたは複数のモジュールからプログラムを作成する。
	CRTSRVPGM (サービス・プログラムの作成)	1 つまたは複数のモジュールからサービス・プログラムを作成する。

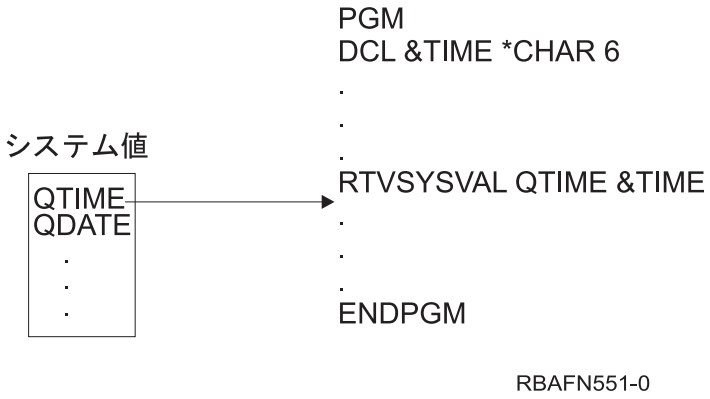
CL プロシージャーの使用法

CL プログラミングは、さまざまな操作を行うことのできる柔軟性の高いツールです。次に示す使用法については、この章の以後の各項で詳しく説明します。一般に次のようなことができます。

- 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。

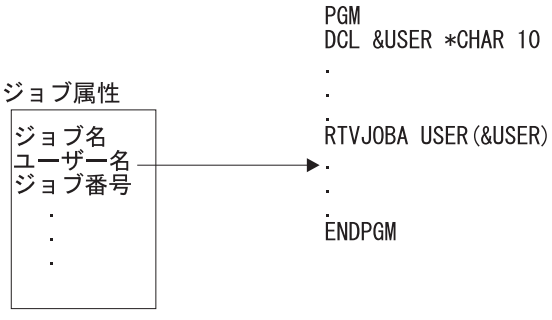
```
PGM
DCL &C *LGL
DCL &A *DEC VALUE(22)
DCL &B *CHAR VALUE(ABCDE)
•
•
•
CHGVAR &A (&A + 30)
•
•
•
IF (&A < 50) THEN(CHGVAR &C '1')
•
DSPLIB ('Q' || &B)
•
IF (%SST(&B 5 1)=E) THEN(CHGVAR &A 12)
•
•
•
       ENDPGM
```

- システム値を CL プロシージャーの変数として使用する。



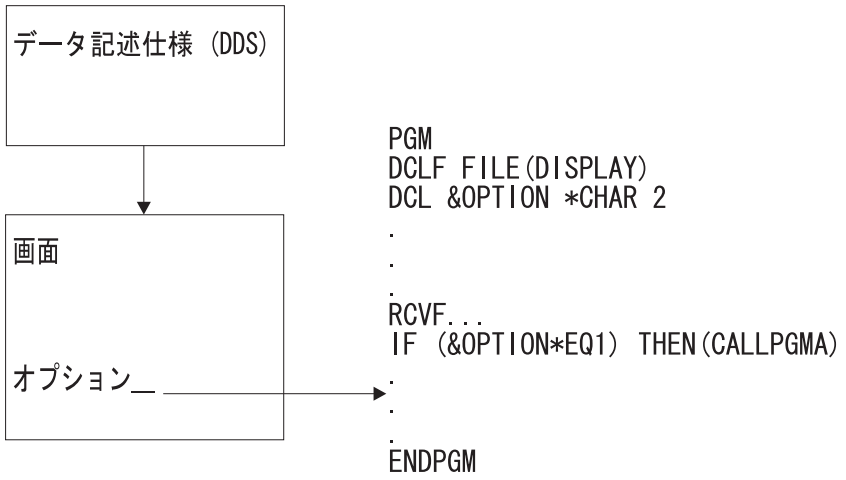
RBAFN551-0

- 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 プロシージャーとの間でデータのやりとりを行う。



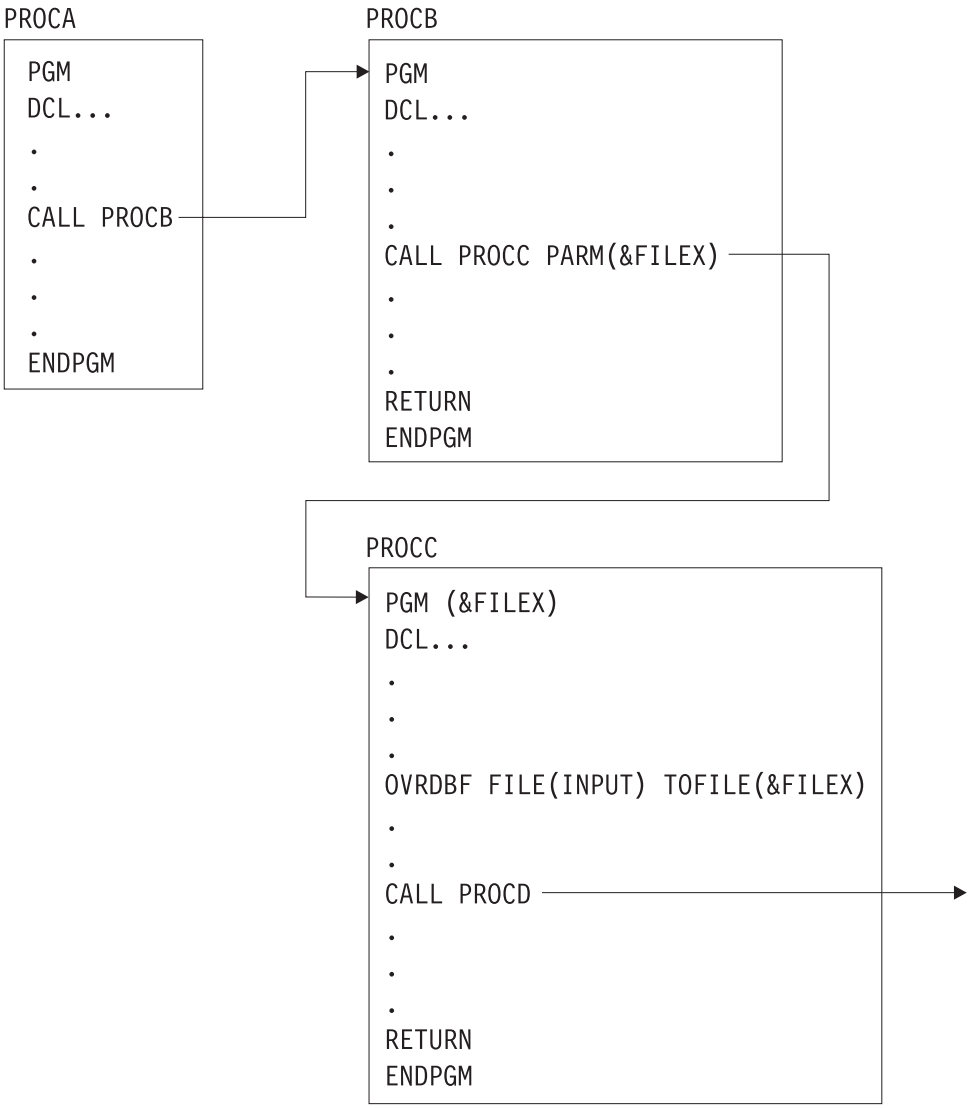
RBAFN553-0

- ジョブのエラー・メッセージを監視する 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

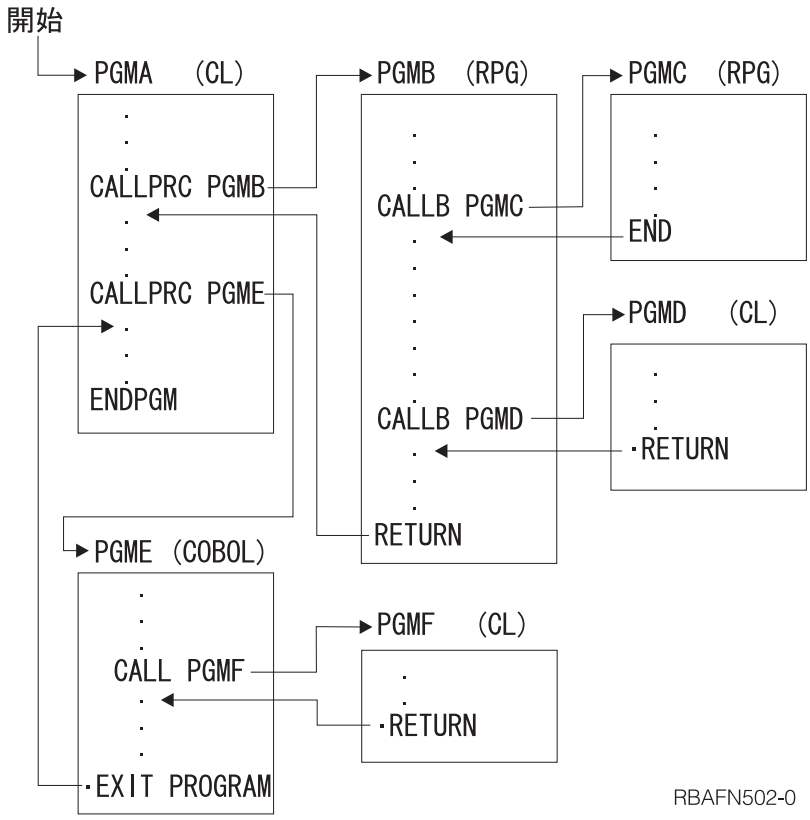
        MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
        CALL PROGA
        CALL PROGB
        RETURN
ERROR:  SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*ESCAPE)
        ENDPGM
```

- 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 プロシージャーから他のプロシージャーまたはプログラムにパラメーターを渡す。



CL プロシージャーを制御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の中における、CL プロシージャーと RPG IV* および 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 は次のものから成ります。

- 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)
```

変数の処理

CL プロシージャーは CL コマンドによって構成され、コマンド自体はコマンド・ステートメント、パラメーター、およびパラメーター値によって構成されます。

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。**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 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なく、変数を含むプロシージャーが活動状態ではなくなると、その変数の値は消失します。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。たとえば、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 プロシージャー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 プロシージャーの実行時に定義 (指定) することができます。

CL 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。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。

  たとえば、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。

この項で述べた用法に加えて、変数は次の目的で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。 77 ページの『第 3 章 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』を参照してください。
- 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。 180 ページの『複数装置表示装置ファイルの処理』を参照してください。
- 条件に基づくコマンドの処理。 34 ページの『CL プロシージャー内での処理の制御』を参照してください。
- 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。

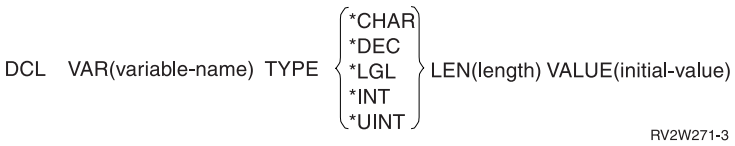
  ```
  CRTPF  FILE(DSTPRODLB/&FILE)
  CRTPF  FILE(&LIB/&FILE)
  ```

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。詳細については、 193 ページの『実行時の CL コマンド変更の許可』を参照してください。

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。詳細については、 187 ページの『QCAPCMD プログラムの使用』および 187 ページの『QCMDEXC プログラムの使用法』を参照してください。

変数の宣言

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 バイトまたは 4 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。
- 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'

  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。

変数によるリストまたは修飾名の指定

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。たとえば、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。 %SUBSTRING 関数の使用例については、53 ページの『%SUBSTRING 組み込み関数の使用法』および第 9 章を参照してください。

変数の値としての小文字

変数の値として使用できる予約値 (*LIBL など) は、特にアポストロフィ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。たとえば、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 パラメーターがアポストロフィ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: 上記の段落の記述では、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。**重要**: 値を大文字に変換するのをシステム任せにすることは、予期しない結果を生じさせる場合があります。

予約値または数値のパラメーター値の変数による置き換え

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。

たとえば、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 ～ 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ので、JOBSEP の値として文字変数を使用する CL プロシージャーを作成することができます。

```
      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 (アポストロフィの用法に注意してください)。 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。

コマンドのパラメーターに使用できる値のタイプについては、 第 9 章を参照してください。追加情報については、 **iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションを参照してください。

CL プロシージャー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。

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。変数の値は次のどれかに変更することができます。

- 定数:

  ```
  CHGVAR   VAR(&INVCMPLT)   VALUE(0)
  ```

  または

  ```
  CHGVAR   &INVCMPLT  0
  ```

  変数 &INVCMPLT の値は 0 に設定されます。
- 他の変数の値:

  ```
  CHGVAR   VAR(&A)   VALUE(&B)
  ```

  または

  ```
  CHGVAR   &A  &B
  ```

  変数 &A の値は変数 &B の値に設定されます。
- 計算後の式の値:

  ```
  CHGVAR   VAR(&A)   VALUE(&A + 1)
  ```

  または

  ```
  CHGVAR   &A  (&A + 1)
  ```

変数 &A の値が 1 ずつ増加します。

- 組み込み関数 %SST により生成された値 (詳細については、 53 ページの『%SUBSTRING 組み込み関数の使用法』を参照):

```
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 (詳細については、 55 ページの『%SWITCH 組み込み関数の使用法』を参照):

```
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 (詳細については、 51 ページの『%BINARY 組み込み関数の使用法』 を参照):

```
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。

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。たとえば、次の例で最初のコマンドは、ローカル・データ域の最初の 10 バイトをブランクにし、2 番目のコマンドはローカル・データ域の最初の 10 バイトを検索します。

```
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 1. 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。

たとえば、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 の全体の長さと小数部分の桁数がそれぞれ 5 と 2 であるとします。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。

     たとえば、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。

コマンド・パラメーターの後書きブランク

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、アポストロフィ (') 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、後書き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、パラメーターに後書きブランクがあり、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、このような状態を引き起こす形で定義され、使用されています。このようなパラメーターの例として、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可能性がある場合には、パラメーターの値をアポストロフィで区切ったコマンド・ストリングを作り、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、意図したとおりの結果を得ることができます。

次の例は、OVRDBF コマンドの実行の際に、後書き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、しかも後書き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。

```
            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。

CL プロシージャーにおける注記の指定方法

CL プロシージャーに注記を入れたい場合、またはプログラム中のコマンドに注記を付加したい場合には、 /* と */ とを一対にして使用し、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、次のど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/*
/**
```

したがって、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。

```
       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 プロシージャー内での処理の制御

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

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、多くの場合 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

GOTO コマンドおよびラベルの使用法

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、その時点でプロシージャー内の別の部分 (ラベルで指定された箇所) 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に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 ステートメント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 を使用して、プロシージャーの外のラベルに移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。

```
                PGM
          .
          .
          .
START:    SNDRCVF RCDFMT(MENU)
          IF (&RESP=1) THEN(CALL CUS210)
          .
          .
          .
          GOTO START
          .
          .
          .
          ENDPGM
```

この例ではラベルは 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。

IF コマンドの使用法

IF コマンドは条件を指定するためのものであり、その条件が真であれば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 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も指定しておくことができます。

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。論理式の構成に関する詳細については、 47 ページの『*AND、*OR、および *NOT 演算子の使用法』の項を参照してください。

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、コマンドのグループである場合もあります。 (41 ページの『DO コマンドおよび DO グループの使用法』の項を参照)。条件が真でなければ、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。

次に示すのは、IF コマンドによる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。たとえば、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。たとえば、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)。このように一連の単純な IF ステートメントを使用した場合と、IF と同時に ELSE を使用した場合または組み込み IF コマンドを使用した場合 (後で説明します) との間の、ロジック上および処理上の違いに注意してください (『ELSE コマンドの使用法』および 39 ページの『組み込み IF コマンドの使用法』を参照)。

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。たとえば、次の例では 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
```

ELSE コマンドの使用法

ELSE コマンドは、対応する IF コマンドで指定した条件が偽であった場合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
```

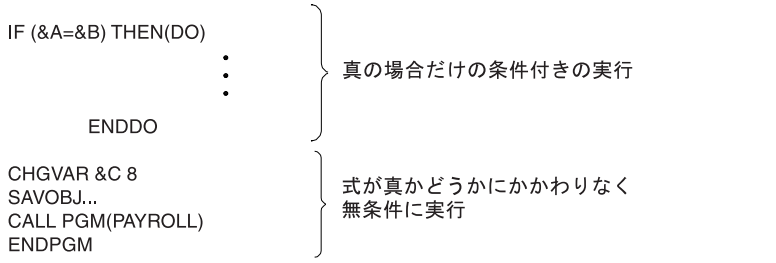
この例の場合、PROCA が呼び出されるのは &A=&B の場合だけですが、PROCB は常に呼び出されます。

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
```

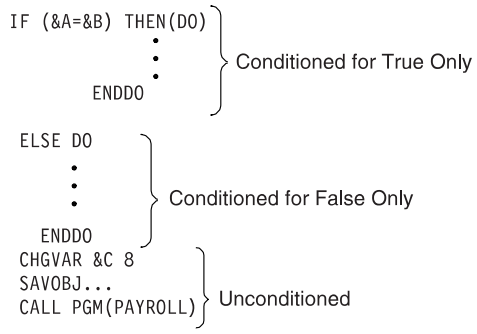
IF 式の評価の結果が偽となった場合に別の分岐 (すなわち二者択一的な処理) を行いたい場合には、ELSE コマンドを使用しなければなりません。

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。



RSLF157-0

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。



RV2W275-1

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、IF テストが最初に真となった時点でその組み込みコマンドが処理され、次に 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
```

組み込み IF コマンドの使用法

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。これは、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他の部分 (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。 47 ページの『*AND、*OR、および *NOT 演算子の使用法』を参照してください。

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
```

すべての条件が真であれば、SNDPGMMSG コマンドが処理され、続いて 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 コマンドがただちに処理されます。最後に、&RESP が 1 でなければ、CHGVAR コマンドがただちに処理されます。 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)))

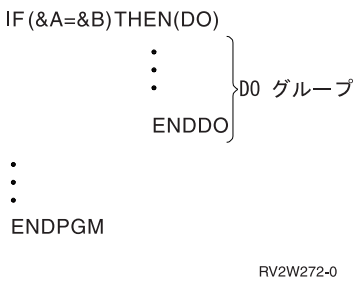
IF (&RESP=1) THEN +
           (IF (&A=5) THEN +
              (IF (&B=NO) THEN(DO)))

IF (&RESP=1) +
        (IF (&A=5) +
              (IF (&B=NO) THEN(DO)))
```

DO コマンドおよび DO グループの使用法

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 コマンドと対応する 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。

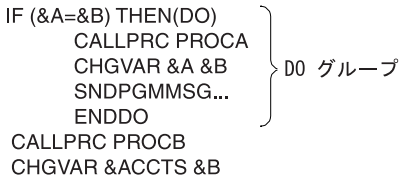
グループの処理は、通常関連するコマンドの評価の結果に基づいて条件付きで行われます。 DO グループは、IF、ELSE、または MONMSG コマンドと関連付けられるのが普通です。次の例をご覧ください。



RV2W272-0

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。

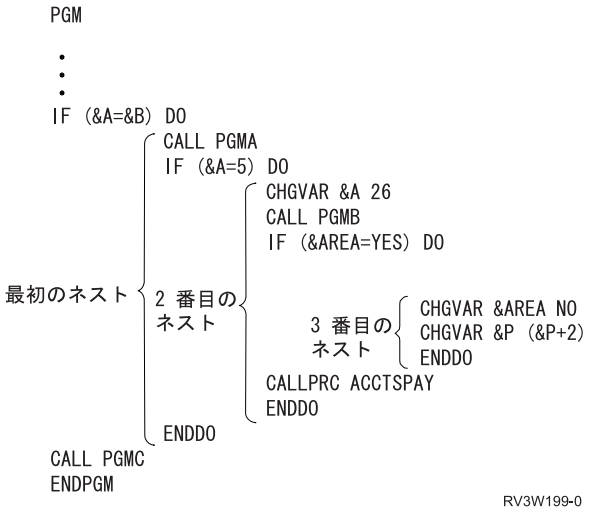
次のプロシージャーでは、&A が &B に等しくなければ、システムは PROCB を呼び出します。したがって、PROCA は呼び出されず、DO グループ内のその他のコマンドも処理されません。



RV3W198-0

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。

DOUNTIL コマンドの使用法

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 と対応する 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。論理式の構成に関する詳細については、 47 ページの『*AND、*OR、および *NOT 演算子の使用法』の項を参照してください。

次に示すのは、DOUNTIL コマンドによる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 コマンドを使用すれば、DOUNTIL 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て、宣言されている条件を即時に評価できます。

DOWHILE コマンドの使用法

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ためのものであり、その条件が真であれば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 と対応する 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。論理式の構成に関する詳細については、 47 ページの『*AND、*OR、および *NOT 演算子の使用法』の項を参照してください。

次に示すのは、DOWHILE コマンドによる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 コマンドを使用すれば、DOWHILE 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て、宣言されている条件を即時に評価できます。

DOFOR コマンドの使用法

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。

次に示すのは、DOFOR コマンドによる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。

ITERATE コマンドの使用法

ITERATE コマンドを使用すれば、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、最も近い活動状態の 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 コマンドの使用を示します。

```
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 以下であれば、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。真であれば、DOWHILE に続く DOFOR から処理が再開します。

LEAVE コマンドの使用法

LEAVE コマンドを使用すれば、活動状態にある DOWHILE、DOUNTIL、または DOFOR グループを終了できます。これは、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 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、最も近い活動状態の 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 コマンドの使用を示します。

```
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 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 に続くコマンドから処理が継続します。

SELECT コマンドおよび SELECT グループの使用法

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関連コマンドの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 または OTHERWISE コマンドで識別されるコマンドのグループのうち、各グループ内で 1 つだけが処理されます。

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。論理式の構成に関する詳細については、 47 ページの『*AND、*OR、および *NOT 演算子の使用法』の項を参照してください。

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 グループ内の次の WHEN コマンドで指定される条件が評価されます。次の 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。 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 が評価されます。 &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。

*AND、*OR、および *NOT 演算子の使用法

*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の論理演算子として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、比較式で構成されるか、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、¬<)

これらの演算子については、**iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションで参照できます。

次に論理式の例をいくつか示します。

```
((&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 (比較式など) です。 (比較式とは、>、<、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。) 以下に例を示します。

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

上記の例では、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比較式で、それぞれが同じく括弧で囲まれています。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比較式を使用することができます。

```
IF (&A=&B) THEN(DO)
              .
              .
              .
              ENDDO
```

この比較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、オペランドとして比較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
```

39 ページの『組み込み IF コマンドの使用法』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、比較式と比較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。

比較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'))
          (not true '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'

AND 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

Incorrect:   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、比較式間の 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。

論理式および関係式については、**iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションを参照してください。

%BINARY 組み込み関数の使用法

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。

2 進数組み込み関数の構文は次のとおりです。

```
%BINARY(文字変数名 開始位置 長さ)
```

または

```
%BIN(character-variable-name starting-position length)
```

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 および 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 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 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。

以下は、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。変数 &B4 には値 X'FFFFFFC7' が入ります。

- ```
  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。文字変数 &C5 には値 ‘-0229’ が入ります。

- ```
  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。変数 &B2 には、値 X'04E5' が入ります。
- ```
  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、SNDPGMMSG (プログラム・メッセージ送信) 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。

%SUBSTRING 組み込み関数の使用法

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する関数で、CL プロシージャーの中でだけ使用することができます。 CHGVAR コマンドでは、その値を変更したい変数 (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 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)
```

または

```
%SST(character-variable-name starting-position length)
```

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いくつか示します。

- 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 ～ 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 ～ 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。たとえば、ローカル・データ域の 1 ～ 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。

```
          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。

%SWITCH 組み込み関数の使用法

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、該当のジョブですでに設定されている 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、SBMJOB、CHGJOB、または JOB コマンドの SWS パラメーターによっ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ど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。マスクの各文字は、それぞれジョブの 8 個のスイッチの 1 つに対応しています。すなわち、1 文字目はジョブ・スイッチ 1 に対応し、2 文字目はスイッチ 2 に対応しています (以下同様)。マスクの各文字には、0、1、または X のどれかを指定することができます。

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。そのスイッチの値は %SWITCH の結果には影響しません。

たとえば、`%SWITCH (0X111XX0)` を指定したとすると、ジョブ・スイッチ 1 および 8 が 0 であるかがテストされ、スイッチ 3、4、および 5 が 1 であるかがテストされ、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 プロシージャーの中では、スイッチのテストによりプロシージャーのロジックを制御することができます。この関数は、CL プロシージャーの中で IF および CHGVAR コマンドとともに使用されます。また、CL 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 プロシージャーの場合、このような変更はただちに有効になります。

IF コマンドでの %SWITCH

IF コマンドでは、 %SWITCH はテストの論理式として 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、3、4、5、および 8 の値がそれぞれ 0、1、1、1 および 0 であれば、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。

```
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
```

CHGVAR コマンドでの %SWITCH

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。たとえば、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。

```
PGM
DCL &A *LGL
CHGVAR VAR(&A) VALUE(%SWITCH(10000001))
.
.
.
ENDPGM
```

実行の結果、変数 &A の値は '1' に設定されます。

メッセージ・モニター (MONMSG) コマンドの使用法

エスケープ・メッセージは、CL プロシージャーの中のコマンド、または CL 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シージャーに伝えるためのものです。 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。たとえば、ある CL プロシージャーがすで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。

メッセージ・モニター (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、あらかじめプロシージャーに指定しておくことができます。 MONMSG コマンドは、このコマンドが使用されている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を監視するために使用することができます。 MONMSG コマンドには次のような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

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シージャーの呼び出し待ち行列に送られるメッセージです。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 コマンドは、CL プロシージャー内のすべてのコマンドにより送られるメッセージ・モニターを行うこともできます。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 に対する監視を行うものであり、これらのメッセージは、プロシージャー内のそれ以降のどれかのコマンドから送られてくる可能性があります。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 に対する監視を行います。 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、たと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 パラメーターの指定された 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、またエラーが生じたコマンドの次の順番のコマンドに再び戻ることはできません。これを避けるためには、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 (コマンド・レベルの監視)、それぞれ特定のエラー訂正プロシージャーに分岐する方法をとることができます。

EXEC の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、次の順番のコマンドに移って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 プロシージャーの最初の部分に MONMSG コマンドを指定した場合には、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。

プロシージャー・レベルの MONMSG コマンドおよびコマンド・レベルの MONMSG コマンドの両方に、同じメッセージ識別コードを指定することもできます。その場合には、コマンド・レベルの MONMSG コマンドの方が、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 を受け取った場合には、プロシージャー EXIT2 に分岐します。 CMDB も含めて、プロシージャー内のどれかのコマンドの実行によってメッセージ CPF1999 を受け取った場合には、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、プロシージャーでエラーが生じた場合に限り送られるものですが、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。一般的に CL プロシージャーでは、その基本的な機能に関連し、しかもそれに対する適切な処置を行うことのできるメッセージについての監視を行うことが必要です。その他のメッセージについては、OS/400 は、エラーが起こったものと見なして適切なデフォルトの処置をとります。

CL プロシージャー内でのメッセージの処理に関する詳細については、第 7 章および第 8 章を参照してください。

変数として使用できる値

システム値の検索

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。たとえば、OS/400 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)
```

RTNVAR パラメーターには、指定のシステム値の値を検索して入れる CL 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。システム値の属性の定義については、iSeries Information Center の**『システム管理』**カテゴリーの下に定義されています。

システム値 QTIME

次の例では、QTIME の値を検索して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。

```
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
```

システム値のリスト、およびその変更方法と表示方法については、 iSeries Information Center の**『システム管理』**を参照してください。

システム値 QDATE

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。 CL 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。QDATFMT の出荷時の値は、国や地域によって異なります。たとえば、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、CVTDAT コマンドによって行うことができます。

CVTDAT コマンドの形式は次のとおりです。

```
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。

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。たとえば、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 CL プログラムの作成 (CRTBNDCL) コマンドを単独で使用するか、 CL モジュールの作成 (CRTCLMOD) コマンドとプログラムの作成 (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
```

ILE API の詳細については、iSeries Information Center の『**プログラミング**』カテゴリーを参照してください。

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)

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。

ネットワーク属性の検索

ネットワーク属性検索 (RTVNETA) コマンドを使用することによって、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すること、および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。ネットワーク属性についての詳細は、 iSeries Information Center の**『システム管理』**カテゴリーを参照してください。

RTVNETA の例

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ことができます。

ジョブ属性の検索は、ジョブ属性検索 (RTVJOBA) コマンドによって行います。 RTVJOBA 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一部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 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。 RTVJOBA コマンドは、一連の宣言コマンドの後に続いています。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。

RTVJOBA の例

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。 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。 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 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。

オブジェクト記述の検索

オブジェクト記述の検索 (RTVOBJD) コマンドを使用して、特定のオブジェクトの記述を CL プロシージャーで検索することができます。オブジェクト記述を検索するには変数を使用します。これらの記述から、不要なオブジェクトを判別することができます。オブジェクト記述の検索の詳細については、140 ページの『オブジェクト記述の検索』を参照してください。

特定のオブジェクトの記述をプロシージャーで検索するために、オブジェクト記述の検索 (QUSROBJD) アプリケーション・プログラム・インターフェース (API) を使用することもできます。システムは変数を使用して記述を戻します。詳細については、**iSeries Information Center** の『*プログラミング*』カテゴリーの『*API*』セクションを参照してください。

ユーザー・プロファイル属性の検索

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・プロファイルの属性 (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。

RTVUSRPRF コマンドの実行の後でエスケープ・メッセージ・モニターを行うこともできます。詳細については、 **iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションを参照してください。

RTVUSRPRF の例

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、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。

RTVMBRD の例

次の CL プロシージャーでは、RTVMBRD 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 変数に入れられる。
- 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 プロシージャーの処理

CL プロシージャーを実行するには、まず CL ソース・プロシージャーをコンパイルしてモジュールにし、プログラムにバインドしなければなりません。

CL プログラムを 1 つのステップで作成するには、バインド CL プログラムの作成 (CRTBNDCL) コマンドを使用し、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、CL モジュールの作成 (CRTCLMOD) コマンドによって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、プログラムの作成 (CRTPGM) またはサービス・プログラムの作成 (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。

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。

バインド CL プログラムの作成 (CRTBNDCL) コマンドでは、リスト・オプション、およびそのプログラム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。

ほとんどの場合、CL プロシージャーおよびプログラムは、プログラミング開発管理機能 (PDM) メニューやプログラマー・メニューのオプションを使用して作成されるので、CL モジュールの作成 (CRTCLMOD) またはバインド CL プログラムの作成 (CRTBNDCL) を直接入力する必要はありません。

CL プロシージャーのコマンドのロギング

CL プロシージャーをコンパイルする場合、CL モジュールの作成 (CRTCLMOD) コマンドかバインド CL プログラムの作成 (CRTBNDCL) コマンドの LOG パラメーターに次の値を指定することにより、その CL プロシージャーの実行時点で実行されたプロシージャー中のほとんどの CL コマンドをジョブ・ログに書き込む (記録する) ことができます。

***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 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CL モジュールの作成 (CRTCLMOD) またはバインド CL プログラムの作成 (CRTBNDCL) コマンドの一部なので、値の変更が必要な場合にはプログラムをコンパイルし直さ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 プロシージャーだけであり、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。記録されないコマンドには次のものがあります。

CALLPRC	**CHGVAR**	**DCL**
DCLF	**DO**	**DOFOR**
DOUNTIL	**DOWHILE**	**ELSE**
ENDDO	**ENDPGM**	**ENDSELECT**
GOTO	**IF**	**ITERATE**
LEAVE	**MONMSG**	**OTHERWISE**
PGM	**SELECT**	**WHEN**

ロギング・オプションがオンの状態にある場合、ロギング・メッセージは CL プロシージャーのメッセージ待ち行列に送られます。 CL プロシージャーが対話式に呼び出され、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には、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することができます。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。

CL モジュールのコンパイラー・リスト

CL モジュールを作成する時点で、CL モジュールの作成 (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。

```
          1                                                                               2                    3
5722SS1 V5R3M0  040430                    制御言語                  MYLIB/DUMPERR        01/02/15 13:47:32        ページ   1
 プログラム . . . . . . . . . . . . . . . . . :   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   01/02/15 13:46:02 4
 ソースの印刷オプション . . . . . . . . . . . :   *XREF  *NOSECLVL  *NOEVENTF
 ユーザー・プロファイル . . . . . . . . . . . :   *USER
 プログラム・ロギング . . . . . . . . . . . . :   *JOB
 省略時の活性化グループ . . . . . . . . . . . :   *YES
 プログラムの置き換え . . . . . . . . . . . . :   *NO
 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 AS/400 制御言語コンパイラー 5
    6                          ソース制御言語
SEQ NO  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.   日付 8
   100- PGM                                                                                       01/02/15
   200- DCL &ABC *CHAR 10 VALUE('THIS')                                                           01/02/15
   300- DCL &XYZ *CHAR 10 VALUE('THAT') 7                                                         01/02/15
   400- DCL &MNO *CHAR 10 VALUE('OTHER')                                                          01/02/15
   500- CRTLIB LB(LARRY)                                                                          01/02/15
* 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                                                                            01/02/15
* CPD0013 30  対応する括弧が見つからない。
   700- MONMSG CPF0000 EXEC(GOTO ERR)                                                             01/02/15
   800- ERROR:                                                                                    01/02/15
   900- CHGVAR &ABC 'ONE'                                                                         01/02/15
  1000- CHGVAR &XYZ 'TWO'                                                                         01/02/15
  1100- CHGVAR &MNO 'THREE'                                                                       01/02/15
  1200- DMPCLPGM                                                                                  01/02/15
  1300- ENDPGM                                                                                    01/02/15
                                            * * * * *   ソースの終わり   * * * * *
5722SS1 V5R3M0  040430                    制御言語                  MYLIB/DUMPERR        01/02/15 13:47:32        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        参照                                            11
ERR              ******        700
* CPD0715 30  ラベル 'ERR       ' が存在していない。
ERROR               800
                  * * * * *   相  互  参  照  表  の  終  わ  り   * * * * *
5722SS1 V5R3M0  040430                    制御言語                  MYLIB/DUMPERR        01/02/15 13:47:32        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
 プログラム DUMPERR がライブラリー 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 OS/400 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。

前書き:

4 CL モジュールの作成 (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。

コンパイル・エラーの詳細については、 71 ページの『コンパイルで検出されたエラー』の項を参照してください。

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。

コンパイルで検出されたエラー

モジュールのコンパイラー・リストには、特定のコマンドに直接関係のあるエラー状態は、そのコマンドの直後に印刷されます。このようなメッセージの例については、69 ページの『CL モジュールのコンパイラー・リスト』の項を参照してください。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。

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。次のタイプのエラーがある場合、プロシージャーは作成されません (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。

プロシージャー・ダンプの取り方

CL プロシージャーの処理の過程で、そのプロシージャー・ダンプを取ることができます。 CL プロシージャー・ダンプには、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 プロシージャー・ダンプを取るには、次のどれかの方法を使用してください。

- 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 プロシージャーの中でだけ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。このジョブ属性についての説明は、iSeries Information Center の**『システム管理』**を参照してください。 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 プロシージャー内で機能エラーが発生するたびにプログラム・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 040430                        CL プログラムのダンプ                        1/02/15  14:06:44 2  ページ 1

 ジョブ名 . . . . . . . . :   DSP04 3  ユーザー名 . . . . . . . :   SMITH 3  ジョブ番号 . . . . . . . :   01329 3
 プログラム名 . . . . . . :   DUMP 4   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  送信元

           メッセージ 6                          メッセージ         プログラム      命令      プログラム      命令
 時刻        ID            重大度     タイプ     テキスト           QLICRLIB        *N        DUMP            *N
140644     CPC2102          00        COMP      ライブラリー LARRY が
                                                作成された。       QLICLLIB        *N        DUMP            *N
140644     CPF2110          40        ESC       ライブラリー MOE が見
                                                つかりません。

                                                   変数       7

 変数              タイプ     長さ               値                               値 (16 進数)
                                                *...+....1....+....2....+       * . . . + . . . . 1 . . . . + . . . . 2 . . . . +
&ABC              *CHAR              10        'ONE        '                   D6D5C540404040404040
&XYZ              *CHAR              10        'TWO        '                   E3E6D640404040404040

                                   * * * * *  ダ ン プ の 終 わ り  * * * * *
```

1 OS/400 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。

モジュール属性の表示

モジュールの表示 (DSPMOD) コマンドを使用して、モジュールの属性を表示することができます。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。

このコマンドについては、 **iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションを参照してください。

プログラム属性の表示

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。 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。

このコマンドについては、 **iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションを参照してください。

戻りコードの要約

RTVJOBA コマンドの戻りコード (RTNCDE) パラメーターは、5 文字の 10 進数で、小数点以下がありません (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、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、OS/400 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～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/400® プログラム

  COBOL/400 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/400 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。戻りコードの値が 2 の場合にはメッセージ CBE9001 が送られ、 *ABNORMAL オプションの指定された資源再利用 (RCLRSC) コマンドが実行されます。

- C/400* プログラム

  C/400® プログラムの最後の C/400 戻りステートメントにより戻された、整数戻りコードの現行値。

ソース・プログラムを前のリリース用にコンパイルする

CL プログラム作成 (CRTCLPGM) 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、作成する CL プログラム・オブジェクトをどのリリースの OS/400 ライセンス・プログラムの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 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 以外の特定の値を指定してコンパイルした CL 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。

前リリース (*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)。たとえば、QUSRV4R5M0 ライブラリーは、 OS/400 ライセンス・プログラムのバージョン 4 リリース 5 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。

別個のリリース上でオブジェクトを保管する方法については、該当する CL 参照コマンド (オブジェクト保管 (SAVOBJ)、変更オブジェクトの保管 (SAVCHGOBJ)、またはライブラリー保管 (SAVLIB)) を参照してください。

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。

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・サポートと QSYSVxRxMx ライブラリーは以下の手順に従って導入します。

1. 次のように入力します。

   ```
   GO LICPGM
   ```

   ライセンス・プログラム・メニューが表示されます。
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、次のように入力してこのサポートを削除します。

   ```
   DLTLICPGM LICPGM(5722SS1) 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) コマンドを使ってこれを明示的に削除する必要があります。

第 3 章 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) コマンド、プロシージャーの呼び出し (CALLPRC) コマンド、および戻り (RETURN) コマンドを使用することにより、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。ユーザー・プロファイルについての詳細は、「機密保護解説書」

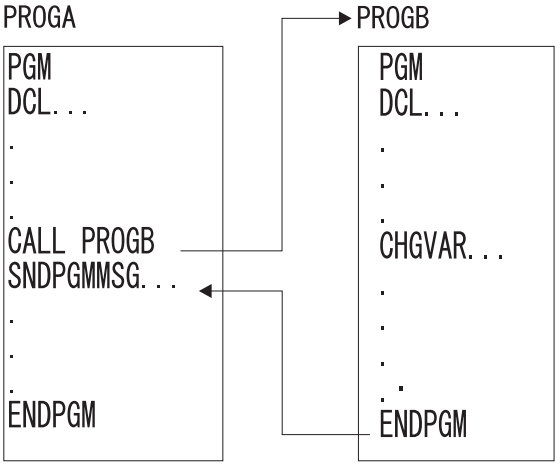

を参照してください。

CALL コマンド

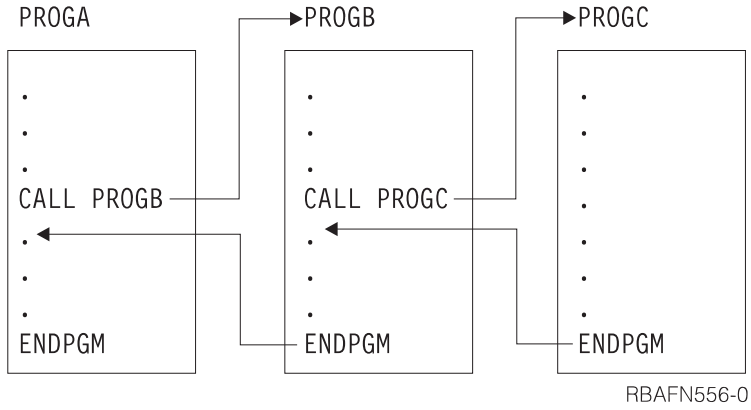
プログラム呼び出し (CALL) コマンドは、指定したプログラムを呼び出し、そのプログラムに制御を渡す場合に使用します。 CALL コマンドの形式は次のとおりです。

```
CALL PGM(ライブラリー名 / プログラム名) PARM(パラメーター値)
```

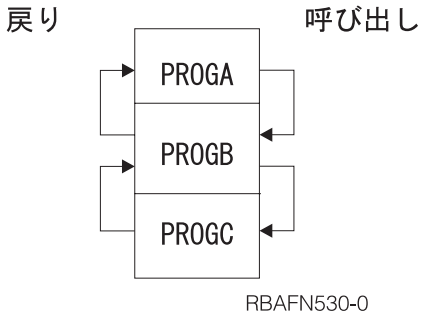
プログラム名およびライブラリー名は、どちらも変数にすることができます。呼び出されたプログラムが、ライブラリー・リスト上にないライブラリーの中にある場合には、PGM パラメーターにプログラムの修飾名を指定しなければなりません。PARM パラメーターについては、 80 ページの『プログラムおよびプロシージャー間のパラメーターの受け渡し』の項で説明します。呼び出されたプログラムの実行が終わると、呼び出し側プログラムの中の次の順番のコマンドに制御が戻ります。



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。たとえば、次のような一連の呼び出しがあるとします。



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どうかにかかわらず行われます。

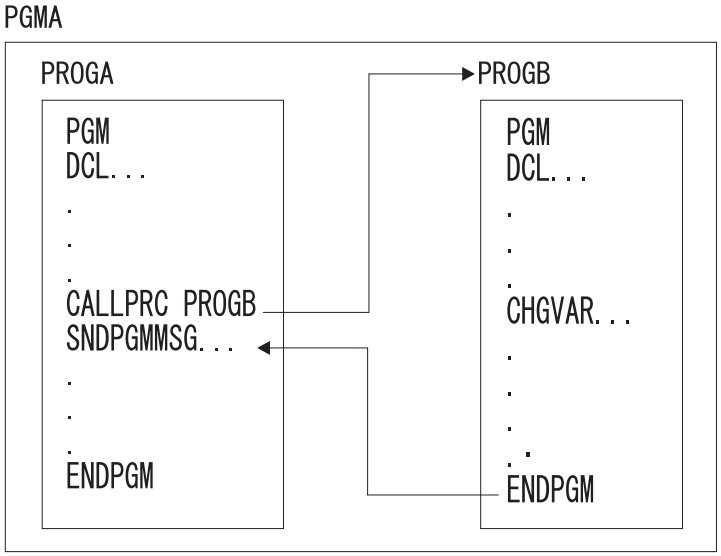
CL プログラムはその CL プログラム自身を呼び出すこともできます。

CALLPRC コマンド

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 コマンドは次のような形式です。

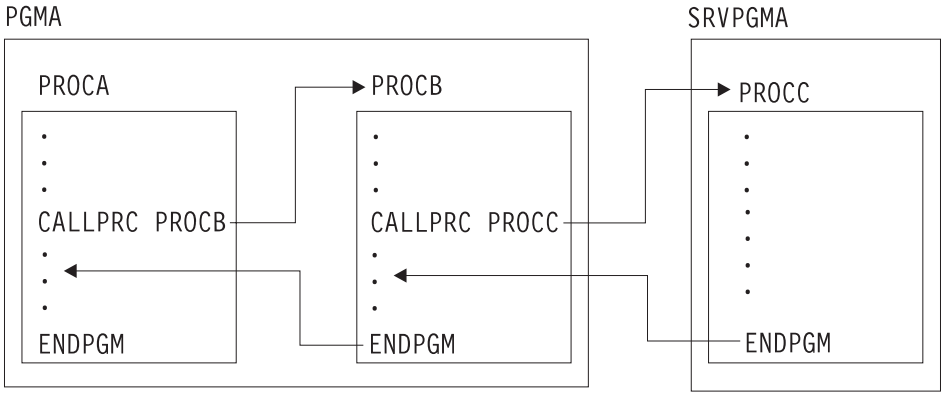
```
CALLPRC PRC(procedure-name) PARM(parameter-values) RTNVAL(return-value-variable)
```

プロシージャー名は変数であってはなりません。 PARM パラメーターについては、 80 ページの『プログラムおよびプロシージャー間のパラメーターの受け渡し』の項で説明します。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。



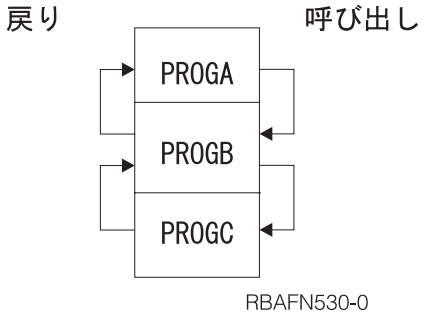
RBAFN539-0

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。たとえば、次のような一連の呼び出しがあるとします。



RBAGN540-0

呼び出しスタックは次のようになります。



RBAFN530-0

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どう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。

RETURN コマンド

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 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 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。

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。

プログラムおよびプロシージャー間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。これについては、84 ページの『CALL コマンドの使用法』の項を参照してください。プログラムまたはプロシージャーに渡す情報は、CALL コマンドまたは 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。

たとえば、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 */
```

CALL コマンドまたは CALLPRC コマンドのいずれの場合にも、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しなければなりません。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、CALL コマンドまたは 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。たとえば、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。

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さなければなりません。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。 API CEEDOD については、**iSeries Information Center** の『プログラミング』にある『*API (APIs)*』セクションを参照してください。

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 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。

OPM CL プログラムを呼び出す場合、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 プロシージャーを呼び出したり、CL 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・リストを持つプロシージャーを作成できるので、ILE CL プロシージャーの作成がさらに柔軟に行えます。たとえば、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。

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、プロシージャーは適切な処置を実行しなければなりません。

プロシージャーを呼び出す際、引き数を参照または値によって渡すことができます。値による引き渡しの詳細については、「ILE概念」  の『Calls to Procedures and Programs』の章を参照してください。

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
```

CALL コマンドの使用法

CL プロシージャーで 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 プログラム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 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、アポストロフィ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、後書きブランク (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、変数を引き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。プログラム呼び出し (CALL) コマンドのオンライン・ヘルプ情報では、CALL コマンドを対話式に使用してパラメーターを渡す方法が説明されています。このコマンド・ヘルプを参照するか、 **iSeries Information Center** の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。

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。たとえば、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アポストロフィ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 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。たとえば、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、引き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 の最初の 6 文字だけが設定されています (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アポストロフィは不要です。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。

DCL コマンドでのデフォルトの長さについては、このコマンド・ヘルプを参照するか、 **iSeries Information Center** の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。

プログラムおよびプロシージャーの呼び出しに関する一般的なエラー

この項では、CALL コマンドまたは CALLPRC コマンドによってプログラムまたはプロシージャーに値を渡す場合に起こる可能性の高いエラーについて説明します。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 コマンドを使用する日付タイプのエラー

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アポストロフィ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。変数の詳細については、25 ページの『変数の処理』を参照してください。

SBMJOB コマンドの CMD パラメーターに CALL コマンドを指定して使用すると、予測でき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。 CMD パラメーターを指定して使用すると、CALL コマンドはコマンド・ストリングに変換されます。そしてバッチ・サブシステムが後でコマンドを開始するときに、そのコマンド・ストリングが実行されます。 CALL コマンドが CALL コマンド自身を使用すると、CL コンパイラーがコードを生成して呼び出しを実行します。

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、通常は後書き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、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 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。コマンド・ストリングの投入には、SBMJOB コマンドのデータ要求 (RQSDTA) パラメーターを使用します。

- CL 文字変数が 32 文字より多くて後書き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。 CALL コマンドを使用する代わりに、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。

データ・タイプ・エラー

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。数値定数をアポストロフィで囲んだ場合には、文字ストリングとして渡されます。しかし定数をアポストロフィで囲まない場合、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 の変数 &A の値は、16 進数 001234567800000F になります。

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 進数の長さおよび精度に関するエラー

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、 (長す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。たとえば、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るだけです。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

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ただ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ただ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 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、その後に後書き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。CALLPRC コマンドに対しては、指定された文字数だけが渡されます。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
```

プログラムとプロシージャー間の通信のためのデータ待ち行列の使用法

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ただ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。

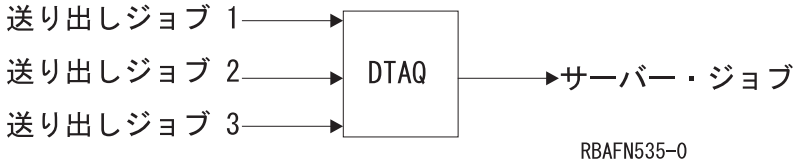
- データ待ち行列を利用すれば、ジョブによる作業の一部を軽減することができます。ジョブが対話式ジョブである場合には、これにより応答時間が短縮され、対話式プログラムおよびその処理アクセス・グループ (PAG) のサイズを縮小することができます。これは、システム全体のパフォーマンスの向上にもつながります。たとえば、複数のワークステーション・ユーザーが、いくつかのファイルに対する更新操作または追加操作を含む同じトランザクションを入力するような場合に、対話式ジョブがそのトランザクションに対する要求を 1 つのバッチ処理ジョブに投入することによって、システムのパフォーマンスは大幅に向上します。
- データ待ち行列は、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は、QSNDDTAQ、QRCVDTAQ、QMHRDQM、QCLRDTAQ、および QMHQRDQD の各プログラムを呼び出すこと

によりデータ待ち行列への送信、受信、記述の検索が可能です。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、送信、受信、消去、または記述を検索したりするための 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。たとえば、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)、またはキー順のどれかにより、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。

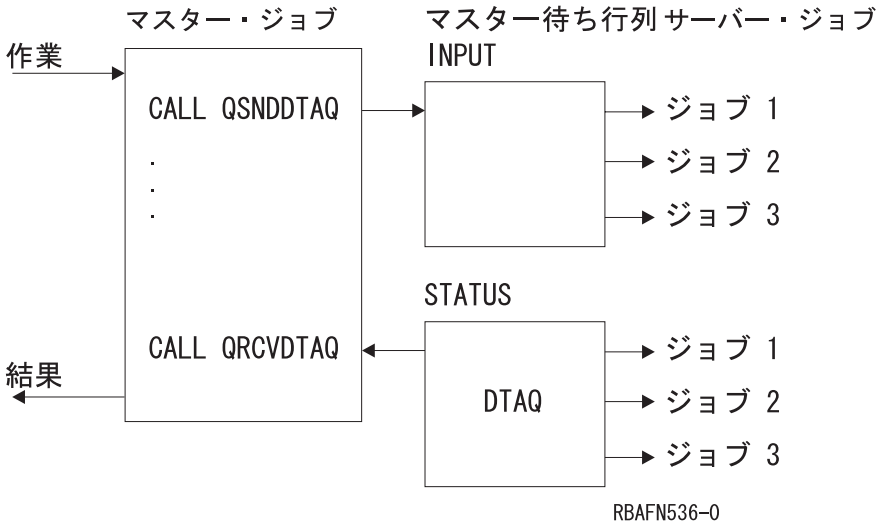
上記の利点に加え、データ待ち行列をジャーナルできます。これにより、異常な初期プログラム・ロード (IPL) または破損が起こったときにオブジェクトが変更アクションの途中であった場合であっても、オブジェクトを一貫性のある状態に回復することができます。また、ジャーナル処理により、リモート・システムのデータ待ち行列ジャーナルの複製も行えます (たとえば、リモート・ジャーナルを使用する)。これにより、システムは、類似の環境にアクションを複製して、アプリケーション作業を複製することができます。 iSeries サーバーのジャーナル・サポートについての詳細は、Information Center の『ジャーナル管理』を参照してください。

次の例はデータ待ち行列の仕組みを示したものです。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、ただちに制御を返す。

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。

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 を使っ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。 出力待ち行列作成 (CRTOUTQ) コマンドについては、 **iSeries Information Center** の**『プログラミング』**にある*『CL』*セクションを参照してください。出力待ち行列におけるデータ待ち行列の詳細については、「印刷装置プログラミング」  を参照してください。

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。詳細については、99 ページの『例 2: 表示装置ファイルおよび ICF ファイルからの入力の待機』および 102 ページの『例 3: 表示装置ファイルおよびデータ待ち行列からの入力の待機』を参照してください。

リモート・データ待ち行列

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 つのサーバーに存在するプログラムがリモート・サーバー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。

DDM データ待ち行列については、**iSeries Information Center** の『*プログラミング*』カテゴリーにある『*CL*』セクションを参照してください。

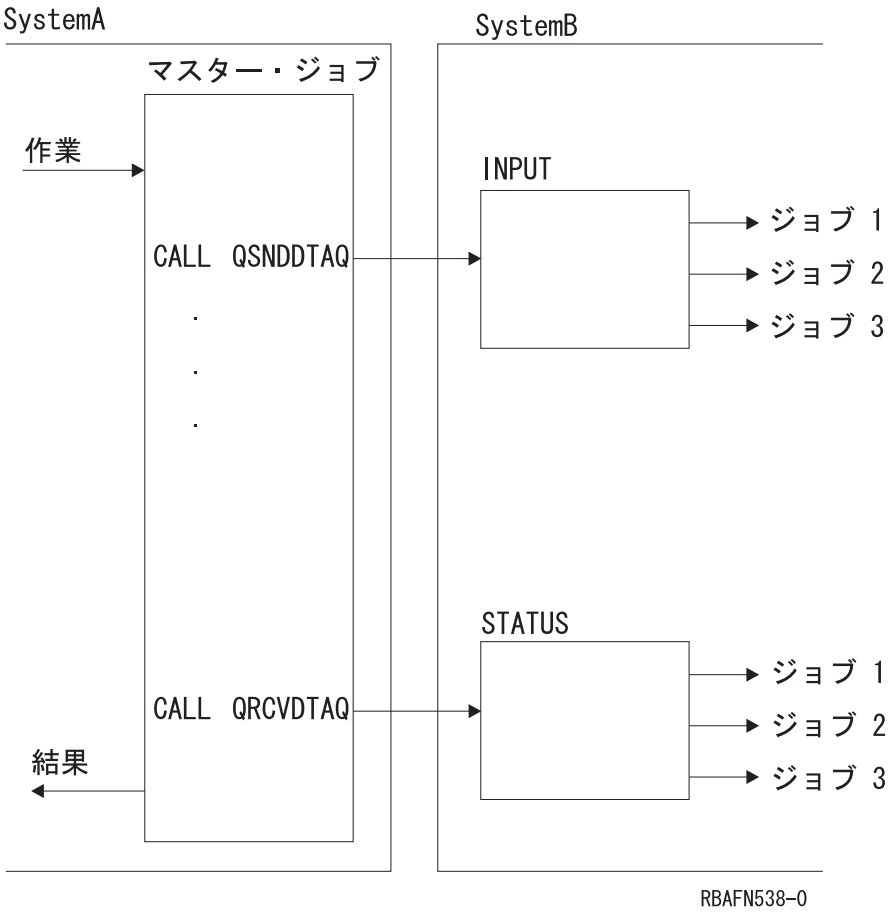


図 1. リモート・データ待ち行列へのアクセス例

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との相違点は次のとおりです。

- 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。したがって、データの保管が主な目的である場合には、データベース・ファイルを待ち行列として使用してください。
- 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、作成し直すのは良い方法です。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、 『データ待ち行列に使用する記憶域の管理』で説明されている自動再利用機能を使用することです。

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。

データ待ち行列の使用上の前提条件

データ待ち行列を使用する場合には、まずデータ待ち行列作成 (CRTDTAQ) コマンドを使用してそれを作成しておかなければなりません。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 ～ 64,512 文字) を指定します。

データ待ち行列に使用する記憶域の管理

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、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、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。

データ待ち行列の割り振り

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必ず ALCOBJ コマンドを使用することにより、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。すでに他のアプリケーション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るからです。

データ待ち行列がすで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。このメッセージを監視し、それに対する処置をとるために、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) コマンドを使用することができます。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。詳細については、272 ページの『CL プログラムまたは CL プロシージャーによるメッセージの監視』の項を参照してください。

データ待ち行列の使用例

以下の例では、データ待ち行列のファイルを処理する 3 つの方法を説明します。

例 1: データ待ち行列からデータを受け取るのに 2 時間待機する

以下の例では、プログラム B でデータ待ち行列から項目を受け取るのに 2 時間 (7200 秒) まで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 がここで項目を 2 時間以内に送る場合、プログラム B は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には、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生じるまでは、項目の受信を継続します。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。

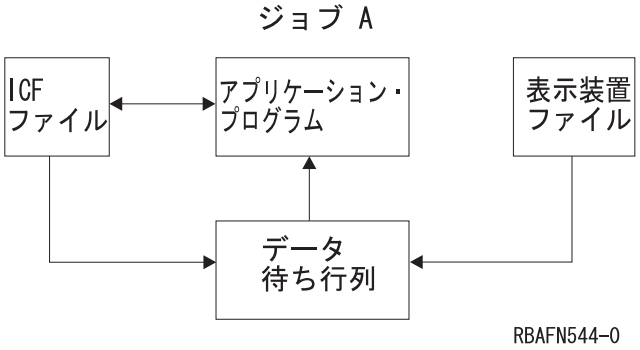
```
PGM
DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
DCL  &FIELD  *CHAR  LEN(80)
.
.(determine data to be sent to the queue)
.
CALL  QSNDDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD)
.
.
.
```

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 2: 表示装置ファイルおよび ICF ファイルからの入力の待機

以下の例ではジョブが 1 つしかないので、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 ～ 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 ～ 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 ～ 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 ～ 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 ～ 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 ～ 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.        */

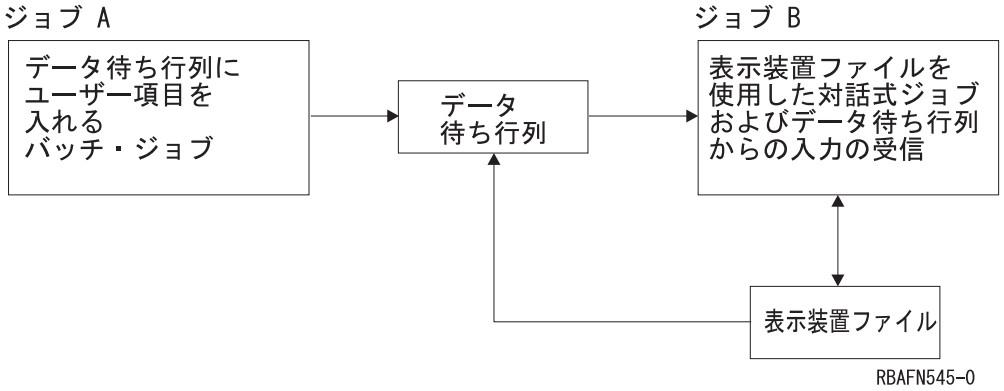
 .
 .
 DO
   WRITE DSPFILE   /* Write with Invite for the Display file           */
   WRITE ICFFILE   /* Write with Invite for the ICF file               */

   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
```

例 3: 表示装置ファイルおよびデータ待ち行列からの入力の待機

以下の例では、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
```

出力待ち行列に関連したデータ待ち行列の作成

データ待ち行列は出力待ち行列に関連付けることができます。出力待ち行列上にあるスプール・ファイルが READY 状況になると、データ待ち行列項目がデータ待ち行列に送られます。データ待ち行列を作成するには、データ待ち行列作成 (CRTDTAQ) コマンドを使用します。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 にしてください。

出力待ち行列に関連したデータ待ち行列については、iSeries Information Center (http://www.iseries.ibm.com/infocenter) の『印刷』カテゴリーの『スプール・ファイルのデータ待ち行列サポート (Data queue support for spooled files)』を参照してください。この情報には、サンプルのデータ待ち行列項目「レコード・タイプ 01 データ待ち行列の項目の形式 (Record type 01 data queue entry format)」も含まれています。

プロシージャーとプログラム間の通信のためのデータ域の使用法

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 (たとえば、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。 iSeries サーバーのジャーナル・サポートについての詳細は、「バックアップおよび回復の手引き」


を参照してください。

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。

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。たとえば、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。

プログラム初期設定パラメーター (PIP) データ域

ジョブの開始の時点で、開始されるジョブごとに 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。

リモート・データ域

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。あるサーバーに存在するアプリケーション・プログラムがリモートのサーバー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・サーバー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。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。

DDM データ域については、**iSeries Information Center** の『*プログラミング*』カテゴリーにある『*CL*』セクションを参照してください。

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')
```

データ域のロックと割り振り

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。 RTVDTAARA コマンドおよび DSPDTAARA コマンドの場合には、コマンドの処理の過程で、データ域に対して *SHRRD (読み取り共用) が適用されます。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。たとえば、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。オブジェクトの割り振りについては、第 4 章を参照してください。

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 HLL 言語の解説書を参照してください。

データ域の表示

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) や、データ域の値を表示することができます。データ域表示 (DSPDTAARA) コマンドについての詳細は、 **iSeries Information Center** の『*プログラミング*』にある『*CL*』セクションを参照してください。

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。

データ域の変更

データ域変更 (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 プロシージャーに入れる場合には、プログラムの作成の時点では、該当のデータ域は存在しなくても構いません。

データ域の検索

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。

データ域の検索の例

例 1

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 ～ 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 2

次の例の RTVDTAARA コマンドは、5 文字のデータ域から指定された内容を検索し、それを 3 文字の変数に入れます。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 の中に) 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)
```

上記の結果、&CLVAR1 には 'CDE' が入ります。

例 3

次の 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、小数部分の長さが 2 文字、全体の長さが 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、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)
```

上記の結果、&CLVAR2 には 0012.3 という値が入ります (小数部分に切り捨てが生じます)。

データ域の変更と検索の例

次に示すのは、文字サブストリング操作を行うための CHGDTAARA コマンドおよび RTVDTAARA コマンドの使用例です。

この例では次のことが行われます。

- DA1 という名前の 10 文字のデータ域が (ライブラリー MYLIB 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 桁付けたもの) 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)
```

上記の結果、変数 &CLVAR1 には 'EFG I' という値が入ります。

第 4 章 オブジェクトおよびライブラリー

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。たとえば、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、iSeries サーバーのデータの検索、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。

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。) この章に含まれる情報は、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。ディレクトリーについては、iSeries Information Center の『**データベースおよびファイル・システム**』カテゴリーにある『*統合ファイル・システム*』トピックを参照してください。

オブジェクトのタイプと共通属性

サーバーの各オブジェクト・タイプはシステム固有の使用目的を持ち、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。 オブジェクト・タイプ、オブジェクト・タイプ・パラメーターのパラメーター値として使用される省略形、およびそのタイプに属するオブジェクトの定義の完全なリストについては、**iSeries Information Center** の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。

各オブジェクト・タイプには、オブジェクトを記述する 1 組の共通の属性があります。この共通属性のリストは、 136 ページの表 3 に示されています。オブジェクト記述の表示 (DSPOBJD) コマンドのオンライン・ヘルプ情報には、これらの属性が説明されています。 **iSeries Information Center** の『**プログラミング**』カテゴリーの『*CL*』セクションにあるコマンド資料を参照してください。

オブジェクトに対して実行可能な機能

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。これらの機能には、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を使用して要求できる機能があります。

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。たとえば、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を持ってい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。たとえば、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、次の 2 つのタイプのものがあります。

- 各オブジェクト・タイプごとに異なる特定の機能。たとえば、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。
- 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 2. オブジェクトに対する共通機能

機能	ページ
ライブラリー中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	124
ライブラリー中のオブジェクトの権限の指定	126
ライブラリーへのオブジェクトの収容	130
オブジェクト記述	135
オブジェクト記述の表示	135
オブジェクト記述の検索	140
システム内の不要オブジェクトの検出	143
ライブラリー相互間のオブジェクトの移動	149
オブジェクトの複製	152
オブジェクト名の変更	154
オブジェクトの削除	159
オブジェクトの割り振りと割り振り解除	159
オブジェクトのロック状態の表示	163
オブジェクトの存在の検査	168

ライブラリー

iSeries サーバーでは、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。詳細については、128 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』および 126 ページの『ライブラリーに対する権限の指定』を参照してください。

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。独立 ASP の詳細については、『独立 ASP』を参照してください。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。

ライブラリー・リスト

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・ライブラリーのどれかに作成されます。たとえば、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。

OS/400 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 (たとえば、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。

たとえば、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。

システムとともに出荷されるライブラリー、および必要に応じてシステムに導入可能なライブラリーのリストについては、467 ページの『付録 C. ライセンス・プログラム (LP) 内の IBM 提供ライブラリー』を参照してください。

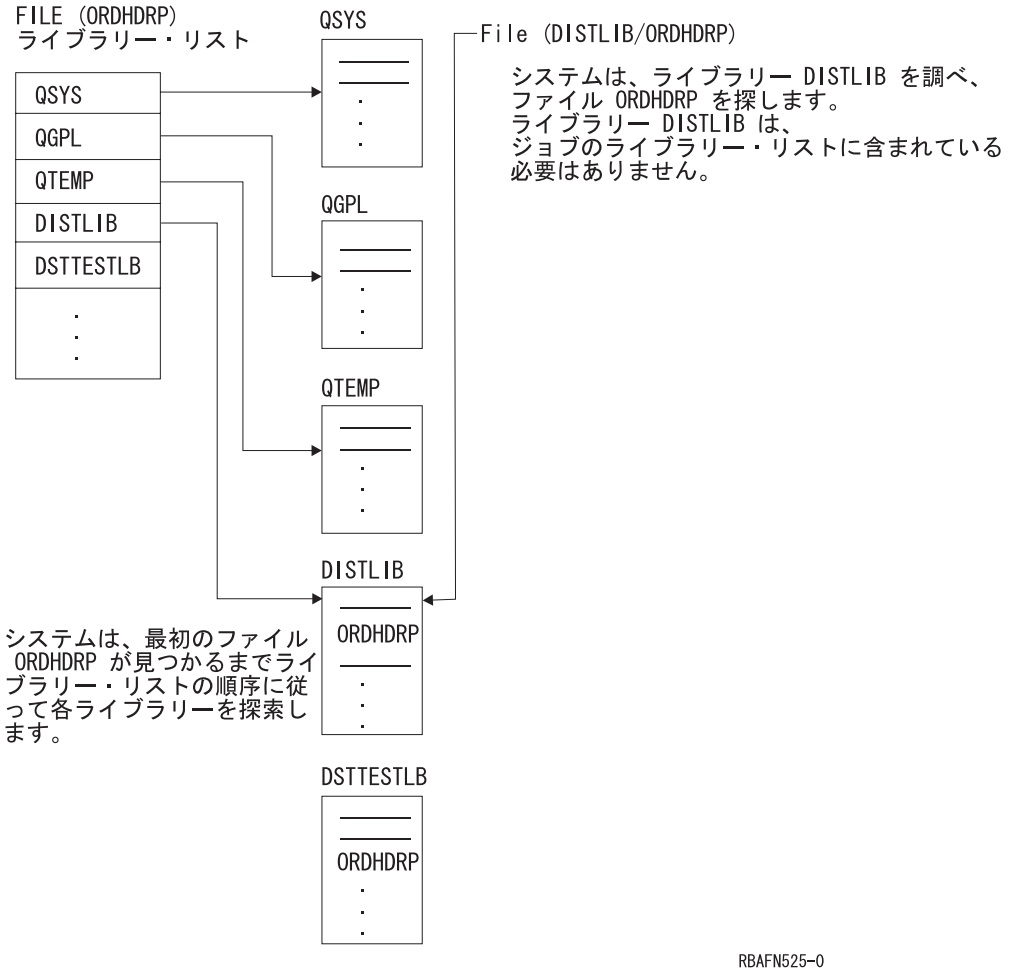
次の図は、ライブラリー・リストの構成の一例です。

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。たとえば、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。

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。

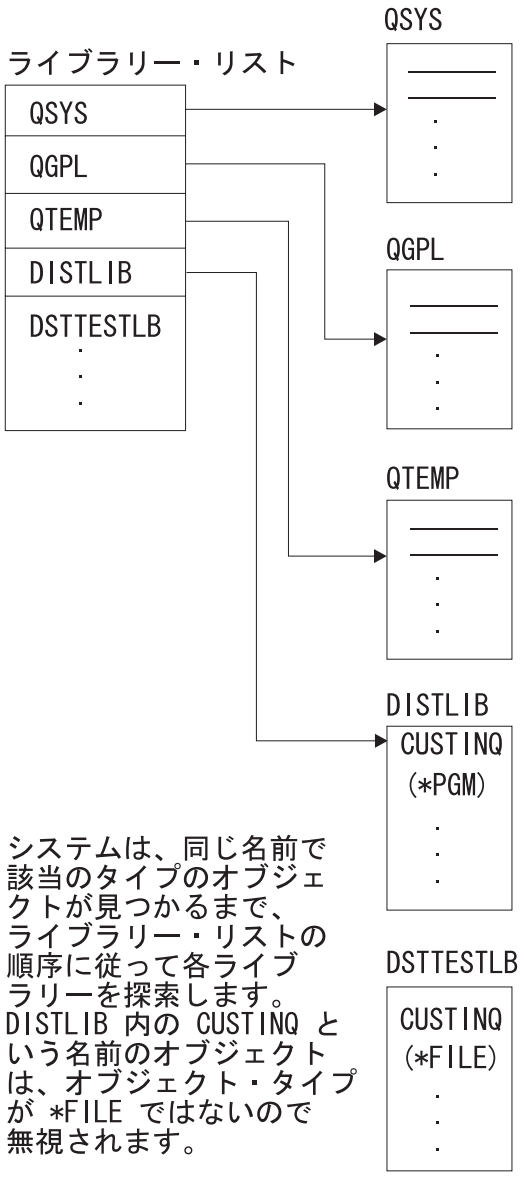
注: 別の方法として、 *LIBL ではなく *NLVLIBL を使用してコマンドを修飾することもできます。 CL プログラム、コマンド行、または他の普段コマンドを入力しているところでコマンドを入力します。システムは *NLVLIBL を使用することによって、 *CMD オブジェクトを探索するのに使用するライブラリーを決定します。 *NLVLIBL を指定すると、ライブラリー・リスト内の各国語サポート・ライブラリーだけが探索されます。

*CMD オブジェクトの署名と検査についての詳細は、 iSeries Information Center の情報の『**セキュリティー**』カテゴリーで、『オブジェクトの署名および検証』を参照してください。



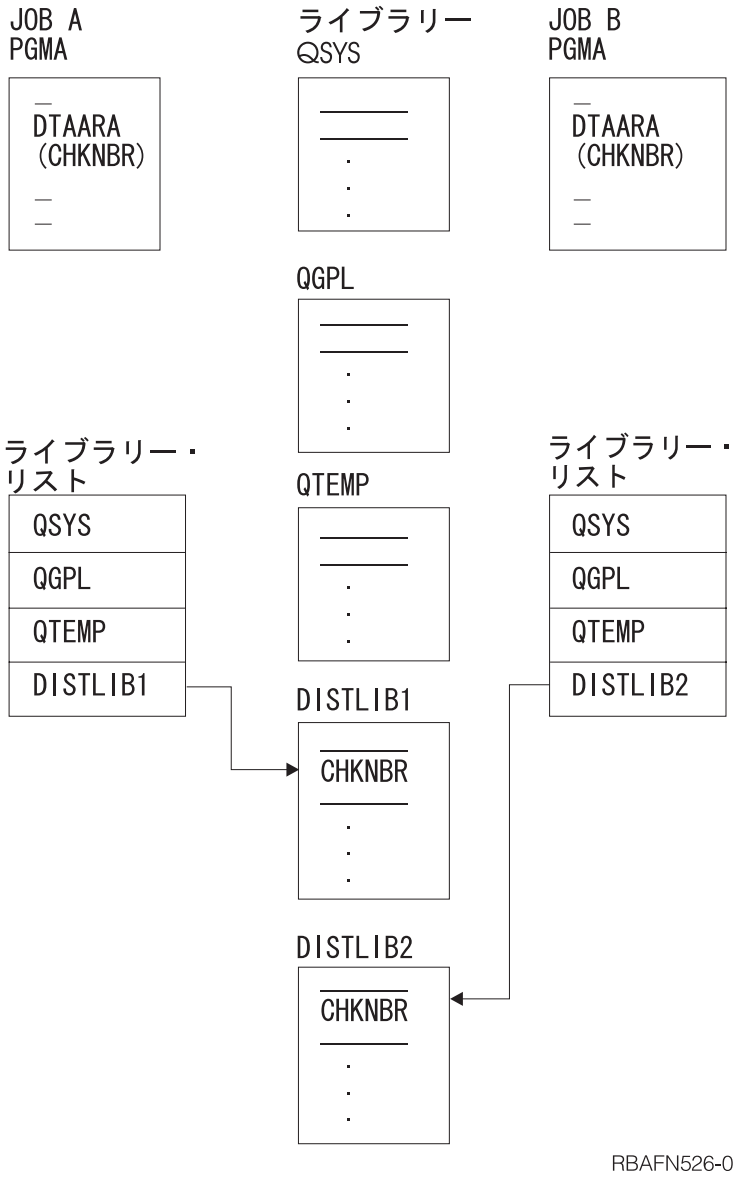
次の図は、名前が同じでタイプの異なる 2 つのオブジェクトがライブラリー・リストに入っている場合にどうなるかを示しています。次のように指定すると、システムはライブラリー・リストからタイプが *FILE の CUSTINQ を探索します。

```
DSPOBJD OBJ(*LIBL/CUSTINQ) OBJTYPE(*FILE)
```



RBAFN541-0

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別々のライブラリー・リストを使用するだけでよく、アプリケーション自体を変更する必要がないということです。たとえば、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。

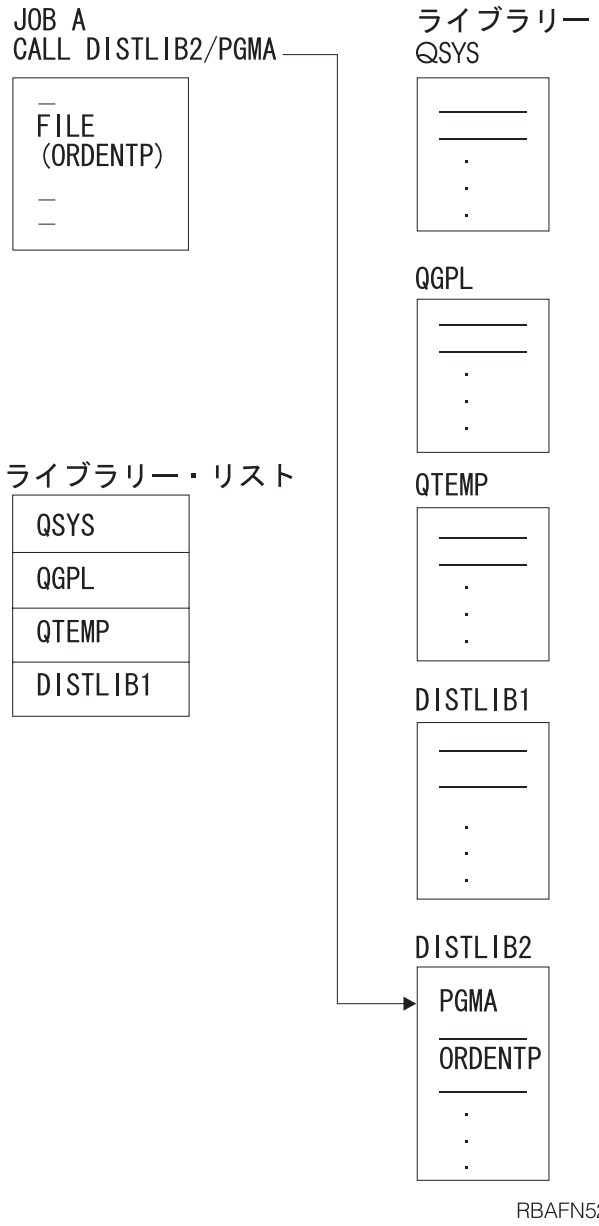


RBAFN526-0

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。



RBAFN527-0

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。

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。

ライブラリー・リストの変更

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 CHGLIBL コマンドや 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 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。ジョブ記述と、バッチ・ジョブ (BCHJOB) コマンドまたは 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。 (ジョブ記述についての詳細は、iSeries Information Center の『**システム管理**』カテゴリーの『*実行管理機能*』トピックを参照してください。)
- BCHJOB コマンドまたは 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。
- 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、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。たとえば、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。

ライブラリー・リストの設定に関する考慮事項

ライブラリー・リストを設定し、使用する場合には、次の点を考慮しなければなりません。

- 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。 OS/400 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、OS/400 はそのライブラリーを除外して開始されます。 OS/400 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。ユーザーはライブラリー・リストを、たとえばジョブ投入 (SBMJOB) 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、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なければなりません。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。

ライブラリー・リストの表示

ライブラリー・リストの表示 (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。

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、活動状態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。

総称オブジェクト名の使用法

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。たとえば、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。**iSeries Information Center** の『**プログラミング**』の『*API (APIs)*』にある『汎用ライブラリー名 (Generic library names)』でリストされています。
- 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。

IBM は、汎用機能を使用する操作についての情報を提供しています。**iSeries Information Center** の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

複数のオブジェクトまたは単一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法

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。

ライブラリーは次の目的のために使用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。たとえば、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。たとえば、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。たとえば、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。これについての詳細は、430 ページの『OPM プログラムのデバッグ』を参照してください。
- 複数の実動ライブラリーの使用。たとえば、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。たとえば、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。 (テスト・ライブラリーの使用法の詳細については、 430 ページの『OPM プログラムのデバッグ』を参照してください。)

ライブラリーの作成方法

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。たとえば、次の例の 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。詳細については、 123 ページの『総称オブジェクト名の使用法』の項を参照してください。

ライブラリーに対する権限の指定

ライブラリーに対しユーザーに認可することのできる各権限について、次に説明します。詳細については、「機密保護解説書」  を参照してください。

オブジェクト権限

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

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

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。

「機密保護解説書」 は、オブジェクトに認可できる権限のタイプと、ユーザーがその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について詳しく説明しています。ライブラリーに対して認可できる権限については、 126 ページの『ライブラリーに対する権限の指定』の項を参照してください。

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。これらのプログラムを作成するための情報は、「機密保護解説書」 を参照してください。

監査ジャーナル項目の表示 (DSPAUDJRNE) コマンド

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。この報告書は、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。

このコマンドで使用するパラメーターおよび値の説明については、オンライン・ヘルプを参照してください。

新たに作成されるオブジェクトに対するデフォルトの共通認可

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。

たとえば、次のように指定したとします。

```
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。

たとえば、次のように指定したとします。

```
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。

たとえば、次のように指定したとします。

```
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。

たとえば、次のように指定したとします。

```
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。 CRTPF コマンドの AUT パラメーターに値として *CHANGE が指定されているので、ファイル PAYACC に対する共通認可は *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。

***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。

***ALL** すべての共通認可

***CHANGE**
変更権限

***USE** 使用権限

***EXCLUDE**
排他権限

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。

新たに作成されるオブジェクトに対するデフォルトの監査属性

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。

たとえば、次のように指定したとします。

```
CRTLIB LIB(PAYROLL) AUT(*EXCLUDE) CRTAUT(*EXCLUDE) CRTOBJAUD(*ALL)
```

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。監査についての詳細は、「機密保護解説書」


を参照してください。

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。たとえば、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。たとえば、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。すで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。 OS/400 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。

ライブラリーの削除と消去

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)
```

または、

```
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. 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。

ライブラリー以外のオブジェクトの削除に関する考慮事項については、 159 ページの『オブジェクトの削除』の項を参照してください。

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。

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、それに対する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、該当の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得ることができます。

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

DSPLIB コマンドには、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、それによってライブラリー選択画面を回避することができます。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。

- DSPLIB コマンドに複数のライブラリー名を指定した場合には、コマンドに指定された順序に従ってライブラリーが表示されます。
- DSPLIB コマンドに *LIBL または *USRLIBL を指定した場合には、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。
- DSPLIB コマンドに *ALL または *ALLUSR を指定した場合には、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。

たとえば、次の DSPLIB コマンドによって、DISTLIB に入っているオブジェクトのリストが表示されます。

```
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。

DSPLIB コマンドの詳細およびサンプル画面については、 **iSeries Information Center** の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

ライブラリー記述の表示および検索

ライブラリー記述の表示 (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、ライブラリーの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

OS/400 グローバリゼーション

OS/400 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。

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。たとえば、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 OS/400* 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。たとえば、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。次のオプションのどれかを使用して、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。

```
PGM
  CHGSYSLIBL  LIB(QSYS2928)  /* Use French information */
ENDPGM
```

オブジェクトの記述

作成コマンドを使用してオブジェクトを作成する場合には、その作成コマンドの TEXT パラメーターの 50 文字のフィールドに、そのオブジェクトについての記述を指定することができます。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 3. 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 • 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 • 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 • オブジェクト監査の値 • ディジタル署名 • ディジタル署名システム・トラステッド・ソース • ディジタル署名複数署名 • 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 • 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 • ジャーナル開始日時 • 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	• 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 • 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どう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 API (QLIRLIBD) 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。

DSPOBJD コマンドまたは 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 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

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= 最上部    F18= 最下部
```

*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        1 次グループ . . . . :   *NONE
タイプ . . . . . :   *FILE

ユーザー定義の情報 :
  属性 . . . . . :
  テキスト . . . . . :

作成情報 :
  作成日／時刻 . . . . . :   06/08/89  10:17:03
  作成ユーザー . . . . . :   QSECOFR
  作成システム . . . . . :   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        1 次グループ . . . . :   *NONE
タイプ . . . . . :   *FILE

変更／使用状況情報 :
  変更日／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
  使用日数カウント . . . . . :   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
  ディジタル署名 . . . . . :   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        1 次グループ . . . . :   *NONE
タイプ . . . . . :   *FILE

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 . . . . . :    不適格
  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        1 次グループ . . . . :   *NONE
タイプ . . . . . :   *FILE

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
```

オブジェクト記述の検索

オブジェクト記述の検索 (RTVOBJD) コマンドを使用して、特定のオブジェクトの記述を CL プロシージャーで検索することができます。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。

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
- ライブラリー ASP 装置
- オブジェクトが、常駐している補助記憶域プール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)
- 最後の保管時のオブジェクトのサイズ (記憶域バイト数)
- 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
- オブジェクトがディジタル署名であるかどうか
- ディジタル署名システム・トラステッド・ソース

- ディジタル署名複数署名
- 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
- ジャーナルの最終開始日時

RTVOBJD の例

次に示す CL プロシージャーにおいて、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(13)
DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
DCL  &RESET    TYPE(*CHAR) LEN(13)
.
.
.
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

次の情報は、オブジェクト記述中に提供され、オブジェクトの作成時に設定されます。これらの情報は、オブジェクトの管理と保守に有用です。

- 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 の導入前にすで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した場合、作成システムは、媒体上のオブジェクトを作成したシステムです。
  - IBM 提供のオブジェクトの場合、作成システムは 00000000 です。
  - バージョン 1、リリース 3.0 の導入前にすでにシステムに存在しているオブジェクトの場合、作成システムはブランクです。

システム上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**。たとえば、CRTDUPOBJ を使用してコピーを作成したが、データベース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。

表 4 は、オブジェクト・タイプごとに、その最終使用日を更新させる操作について、さらに情報を示します。

表 4. 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	コマンド変更要求活動の変更 (CHGCMDCRQA)
図表様式	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
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

表 4. 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
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
図形記号セット	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
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。
媒体定義	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

表 4. 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
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。
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

表 4. 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
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
検索見出し	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。
サブシステム記述	サブシステムが開始された場合
スペル援助辞書	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
テーブル	変換用にプログラムで使用された場合
時間帯の記述	• 時間帯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、時間帯記述を参照する。
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: iSeries サーバーは、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すでに存在する場合、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: アプリケーション・プログラミング・インターフェース (API) である QUSROBJD は、オブジェクト記述の検索コマンドと同じ情報を提供します。詳細については、**iSeries Information Center** の『**プログラミング**』カテゴリーの『*API*』セクションを参照してください。

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)

ライブラリー相互間のオブジェクトの移動

オブジェクト移動 (MOVOBJ) コマンドを使用して、ライブラリー相互間でオブジェクトを移すことができます。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。たとえば、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。

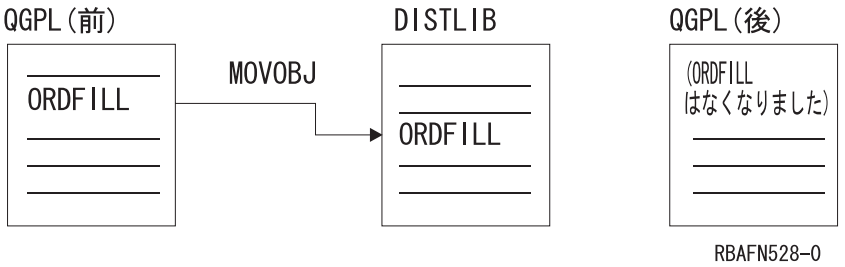
ジャーナルおよびジャーナル・レシーバーは、それが本来作成されたライブラリーに戻す場合に限り移動が可能です。ジャーナル・オブジェクトが記憶域再利用

(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)
- 構造化照会言語 (SQL) パッケージ (*SQLPKG)
- 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。



RBAFN528-0

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。たとえば、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 が指定されていればライブラリー・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。

MOVOBJ コマンドはオブジェクトを 1 度に 1 つだけ移動します。

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・サーバー記述 (*NWSD)
- 参照コード変換テーブル (*RCT)
- サーバー記憶域 (*SVTSTG)
- スペル援助辞書 (*SPADCT)
- SQL パッケージ (*SQLPKG)
- 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だけを複製したい場合には、CRTDUPOBJ コマンドに続けて、選択値を指定した 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。

オブジェクト名の変更

オブジェクト名変更 (RNMOBJ) コマンドを使用して、オブジェクトの名前を変更することができます。ただし、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・サーバー記述 (*NWSD)
- SQL パッケージ (*SQLPKG)
- 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。 RNMOBJ コマンドを発行した時点で、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ず、オブジェクトの名前は変更されません。その結果、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。たとえば、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れば、それらの参照名もすべて変更しなければなりません。他のオブジェクトを参照するオブジェクトのリストについては、149 ページの『ライブラリー相互間のオブジェクトの移動』の項を参照してください。

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。

オブジェクトの圧縮または圧縮解除

システムのディスク・スペースを節約する目的で、オブジェクトの圧縮 (CPROBJ) コマンドを使用して選択したオブジェクトを圧縮することができます。また圧縮したオブジェクトを、オブジェクトの圧縮解除 (DCPOBJ) コマンドを使用してもとに戻すことができます。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。 OS/400 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。

オブジェクトの圧縮

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 5. 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

OS/400 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、以下に示すど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。 (つまり、ENDSYS または ENDSBS *ALL が実行された)
  - 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ためには、オブジェクトのタイプに対応する削除 (DLTxxx) コマンドを使用することができます。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) で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、ユーザーはすで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、これらの再作成されたプログラム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、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だけ であることを確認してください。総称名によるオブジェクトの指定の詳細については、123 ページの『総称オブジェクト名の使用法』を参照してください。

ライブラリーの削除の詳細については、131 ページの『ライブラリーの削除と消去』の項を参照してください。

資源の割り振り

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。たとえば、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。

OS/400 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。 (メッセージ・モニターの詳細については、本書の第 7 章および第 8 章を、例外処理の詳細については、該当する高水準言語の解説書を参照してください。)

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。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、オブジェクトがすで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 6. 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	他のジョブが取り得るロック状態
*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 7. 特定のオブジェクト・タイプに有効なロック状態

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
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				

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ただ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。

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。

```
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。

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。

WAITRCD パラメーターについては、「バックアップおよび回復の手引き」  を参照してください。

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。

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・ファイルの場合には、WRKOBJLCK コマンドによってファイル・レベル (オブジェクト・レベル) のロックは表示されますが、レコード・レベルのロックは表示されません。たとえば、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ついてのロックもまた、WRKOBJLCK コマンドを使用して表示することができます。

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

 OPT      ジョブ          ユーザー        ロック       状況       有効範囲    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
                                         *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
```

第 5 章 CL プロシージャーおよびプログラムでのオブジェクトの処理

CL プログラムでのオブジェクトへのアクセス

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。例外については、166 ページの『例外: 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』の項で説明します。

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、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、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
```

CRTCLPGM や 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・リスト内のどれかのライブラリーに入っていなければなりません。)

例外: 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

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・リスト内のどれか 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。コマンド属性のうち、コマンドを再作成せずに変更できる属性については、 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』トピックのコマンド変更 (CHGCMD) コマンド説明を参照してください。

ファイルへのアクセス

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。 CL モジュールまたは 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。ファイルの詳細については、 169 ページの『CL プロシージャーでのファイルの処理』の項を参照してください。

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。

- プログラムの作成 (CRTPGM) または 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。 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。以下にその例を示します。

```
          CHKOBJ  OBJ(OELIB/PGMA)  OBJTYPE(*PGM)
          MONMSG  MSGID(CPF9801)  EXEC(GOTO NOTFOUND)
          CALL OELIB/PGMA
          .
          .
          .
NOTFOUND: CALL FIX001 /*PGMA Not Found Routine*/
          ENDPGM
```

この例では、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。 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、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。メッセージ・モニターの詳細につ

いては、 57 ページの『メッセージ・モニター (MONMSG) コマンドの使用法』、第 7 章、および第 8 章を参照してください。

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。

CL 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 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、

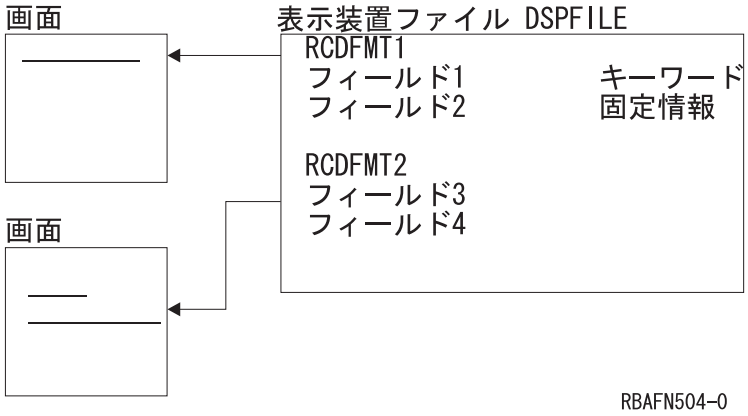
同じコマンドが使用されるので、この 2 つのファイルに対するサポートは類似しています。ただし、次に示すような多少の相違があります。

- 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、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 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 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、他の高水準言語で固有性が要求されることがあるからです。



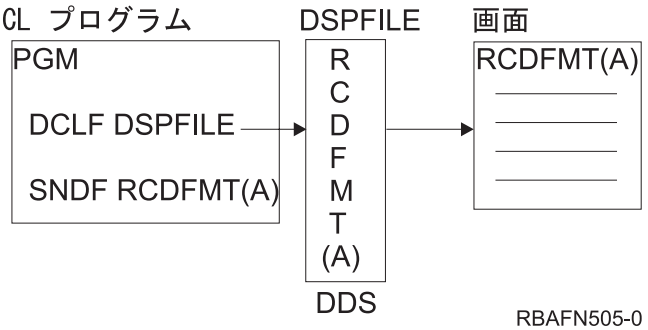
「Application Display Programming」 で説明されている方法、または画面設計機能 (SDA) を使用して、表示装置ファイルのレコードおよびフィールドの DDS ソースを入力することができます。対話式画面設計についての詳細は、「*ADTS/400: 画面設計機能 (SDA)*」を参照してください。

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っ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 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。 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、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 ～ &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。

CL プロシージャーでのファイルの参照

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 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・リスト内のどれかのライブラリーに入っていなければなりません。

CL プロシージャーでのファイルのオープンとクローズ

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 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時点で、事前に 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、CRTDSPF、CRTPF、および CRTLF の各コマンドでの SHARE パラメーターの使用に関する説明をオンラインで提供しています。 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。 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ただちにクローズされ、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。 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 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。 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。

たとえば、INPUT という名前のフィールドおよび標識 10 が DDS 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
  •
  •
  •
```

表示装置ファイルによるデータの送信および受信 (書き込み / 読み取り)

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、SNDF、RCVF、および SNDRCVF の 3 つのコマンドだけです。

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 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。

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 SNDRCVF コマンドまたは 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 キーワードが入ったレコード様式を用いる SNDF コマンドも、これと同じことを行います。

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。 CL プロシージャーおよびプログラムのメッセージおよびサブファイルに関する詳細については、第 8 章を参照してください。

次の例は、典型的なオペレーター・メニューを作成し、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 OS/400 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。

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。

メニューを制御する CL プログラムの作成

次の例では、メニューを表示して制御するための CL プロシージャーの作成方法を示しています。メニューの作成および制御する別の方法については、「Application Display Programming」  を参照してください。

この例では、CL プロシージャー ORD040CD 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。

```
|
|       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、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。 SNDF、RCVF、または SNDRCVF のど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。たとえば、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、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 CALL コマンドを使用して実行されます。 GO コマンドで実行されるメニューについての説明は、「Application Display Programming」を参照してください。

CL プロシージャーでの表示装置ファイルの一時変更

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。

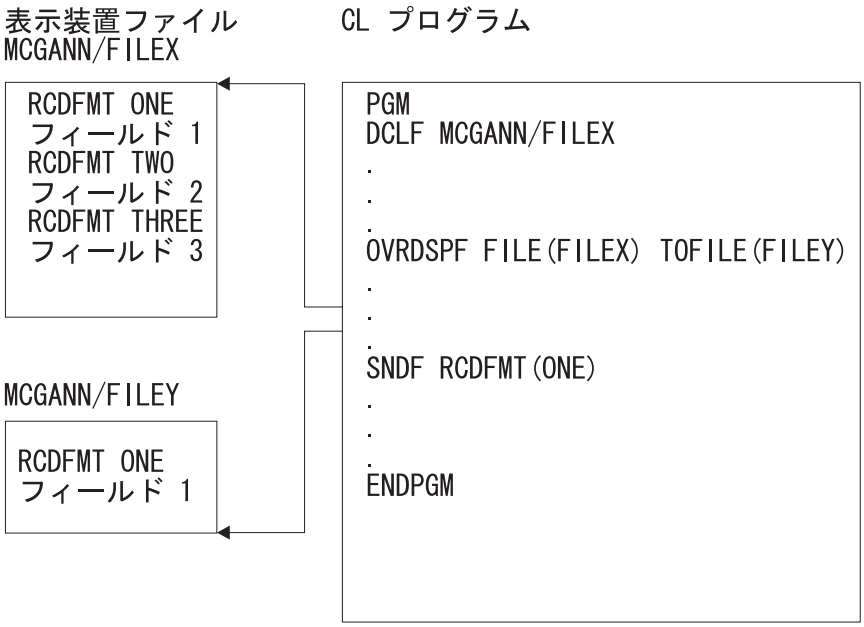
```
OVRDSPF   FILE(一時変更ファイル名) TOFILE(新規ファイル名)
          DEV(device-name)
```

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しなければなりません。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、SNDF コマンド、RCVF コマンド、または 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。

考慮すべきもう 1 つの事項は、OVRDSPF コマンドを適用する場合の SNDF、RCVF、および SNDRCVF の各コマンドの DEV パラメーターに関するものです。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、操作は一時変更ファイルに対応する正しい装置に対して、システムにより自動的に行われます。 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、エラーが起こります。
- 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、エラーが起こります。

複数装置表示装置ファイルの処理

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。各ユーザーは、プロシージャー内の表示装置ファイルも含めて、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。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 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 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、SNDRCVF および RCVF コマンドに影響を与え、WAIT または ENDRCV コマンドを使用しなければならないこともあります。複数装置表示装置ファイルについて RCVF または 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。

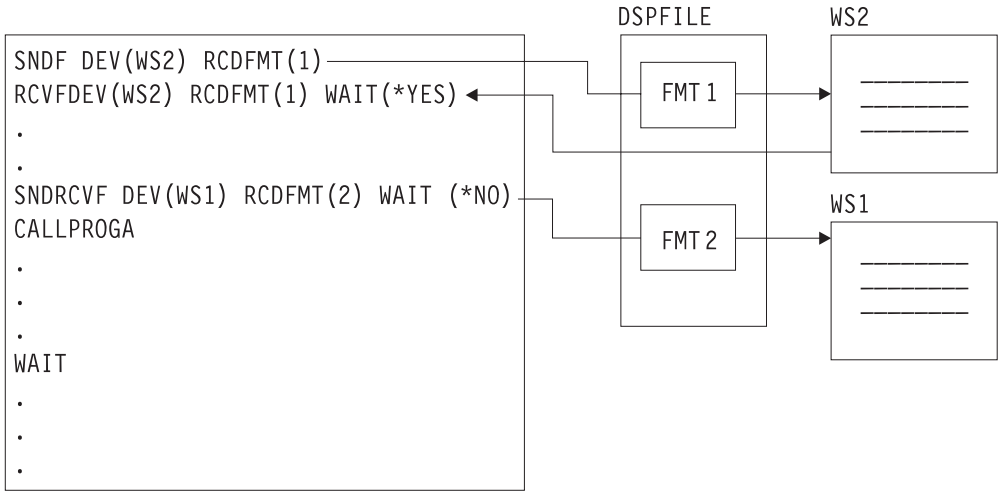
RCVF または SNDRCVF コマンドを使用し、WAIT(*NO) を指定した場合には、WAIT コマンドを実行するまで CL プロシージャーまたはプログラムの処理が続行されます。

DDS の INVITE キーワードを含むレコード様式に対して SNDF コマンドを使用するのは、WAIT(*NO) を指定して SNDRCVF コマンドを使用するのと同様です。SNDRCVF および RCVF コマンドの場合には、DDS の INVITE キーワードは無視されます。

データ・レコードにアクセスするためには、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 のどれかのコマンドの 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。

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。



この例では、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、プログラムで WAIT コマンドに出会った後で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、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。SNDF または 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、SNDF または 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 または RCVF コマンドを使用してデータを受け取るか、あるいは 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、SNDF または SNDRCVF コマンドを実行することはできません。

データベース・ファイルからのデータの受信

データベース・ファイルからデータを受信するために使用できるコマンドは RCVF コマンドだけです。

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 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。

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

データベース・ファイル一時変更 (OVRDBF) 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に指定されているデータベース・ファイルの置き換えや、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うことができます。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)
```

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 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。

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しなければなりません (オープンは、最初に 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、OVRDBF コマンドを含むプロシージャーまたは 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、あるいは 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。

表示コマンドからの出力ファイルの参照

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・ファイルに入れることができます (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。

```
     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、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) 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) 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、DCLF コマンドにより暗黙的に宣言されたフィールドに対応する 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、QSYS/QADSPOBJ の代わりに 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) 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。

第 6 章 上級プログラミングに関する説明

この章では、さらに進んだプログラミング技法を紹介します。説明する事項は次のとおりです。

- 高水準言語プログラム (CL プログラムも含む) から呼び出すことのできる特殊機能。
- プロンプトおよびプログラマー・メニューを用いたプログラム・ソースの入力。

拡張機能コマンド処理については、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

この章の終わりには、サンプル・プログラムがいくつか記載されています。

QCAPCMD プログラムの使用

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・ストリングを実行する前に、その構文をチェックする。
- 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。

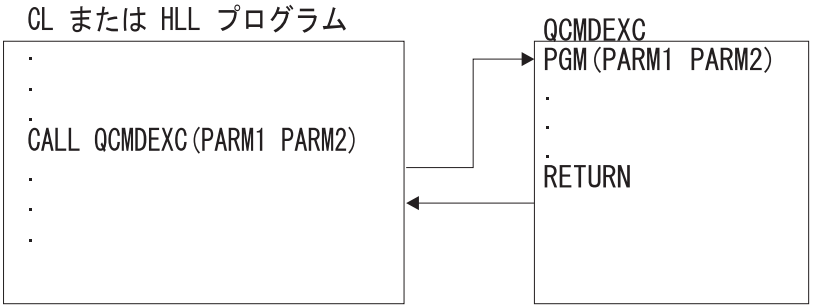
QCAPCMD API については、iSeries Information Center の『**プログラミング**』にある『*API*』セクションを参照してください。

QCMDEXC プログラムの使用法

コマンド実行 (QCMDEXC) は、1 つのコマンドを実行する IBM 提供のプログラムです。このコマンドを使用すると、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。呼び出されるコマンドは、CALL コマンド上のパラメーターとして渡され、実行されます。



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、コマンドをアポストロフィで囲まなければなりません。 文字ストリングの最大長は 6000 文字ですが、区切り文字 (アポストロフィ) 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。

次の例をご覧ください。

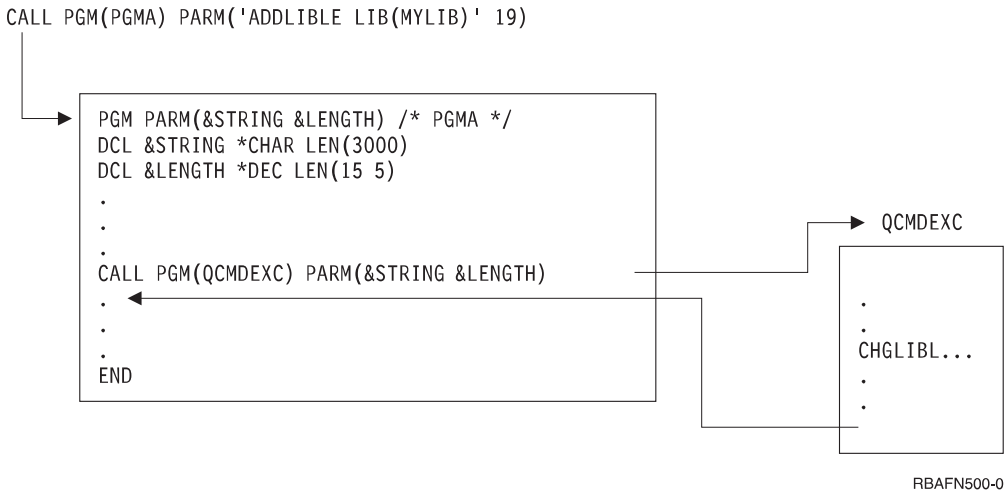


図 2. 呼び出し PGM の例

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

QCMDEXC プログラムへの呼び出しで CL コマンドを渡すことができるかどうかを判別するのに役立つ情報については、 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションにあるコマンド資料を参照してください。コマンドが実行できる環境を判別するためには、 CL コマンドに関する Information Center 資料ファイルの最初にある「Where allowed to run」値を参照してください。また、コマンド表示 (DSPCMD) コマンドを使用すれば、コマンドを使用できる場所を参照できます。

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。メッセージ・モニターに関する詳細については、第 7 章および第 8 章を参照してください。

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。

QCMDEXC プログラムでの DBCS データの使用

QCMDEXC 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。たとえば、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・プロンプトを表示するようシステムに指示します。
- 値 7 は、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
```

QCMDCHK プログラムの使用法

QCMDCHK は、1 つのコマンドの構文検査を行い、オプションでそのコマンドのプロンプトを表示する 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、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、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。

CL プログラムまたはプロシージャー内でのメッセージ・サブファイルの使用

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。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 SNDF コマンドまたは 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 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。

メッセージ・サブファイルによって、制御プロシージャーまたはプログラムは、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。

実行時の CL コマンド変更の許可

ほとんどの CL プロシージャーおよびプログラムでは、ワークステーション・ユーザーが、プロシージャーまたはプログラムに渡すパラメーター値を指定するか、または表示プロンプトの入力可能フィールドに入力を行って、プロシージャーまたはプログラムに入力を行います。

さらに、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もできます。

- CL プロシージャーまたはプログラム・ソース内の 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。プロシージャーまたはプログラム内ですでに指定しているパラメーター値はプロンプトに組み込まれるため、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。 『CL プロシージャーまたはプログラム内での OS/400 プロンプターの使用』の項を参照してください。
- 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 プログラム・ソースで指定する必要はありません。 QCMDEXC プログラムの詳細については、 187 ページの『QCMDEXC プログラムの使用法』の項を参照してください。

CL プロシージャーまたはプログラム内での OS/400 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。たとえば、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。 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・プロシージャーに指定された値は、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。たとえば、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。この制約事項の例外については、 199 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』の項を参照してください。 CL プロシージャー内での処理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 プロシージャー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)
  ```

- 次のコマンドに対しては、プロンプトを使用することができません。

CALL	**CALLPRC**	**CHGVAR**	**COPYRIGHT**
DCL	**DCLF**	**DO**	**DOFOR**
DOUNTIL	**DOWHILE**	**ELSE**	**ENDDO**
ENDPGM	**ENDRCV**	**ENDSELECT**	**GOTO**
IF	**ITERATE**	**LEAVE**	**MONMSG**
OTHERWISE	**PGM**	**RCVF**	**RETURN**
SELECT	**SNDF**	**SNDRCVF**	**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 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。

CL コマンドの選択プロンプト

コマンド内の選択したパラメーターについてプロンプトを表示するよう要求することができます。これは、長い構文のコマンドを使用するさいに、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) を付けなければなりません。
  - 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。
- プロンプト一時変更プログラムで **?-** を使用することはできません。
- パラメーターが必須の場合は、**??** 選択プロンプトを使用しなければなりません。

  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 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

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	有	デフォルト値	無
?/KEYWORD(VALUE)	値	有	デフォルト値	無
?-KEYWORD()	なし	適用外	デフォルト値	適用外
?-KEYWORD(VALUE)	なし	適用外	値	適用外
?&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
?*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。たとえば、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 使用のために予約されています。
?/KEYWORD(VALUE)	値	IBM 使用のために予約されています。
?&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

入力	F5 を押すか消去した時に 表示される値	説明
?&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。たとえば、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)
```

次の表は選択プロンプト文字について要約しています。

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 使用のために予約されています。
?-	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。
?&	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。

QCMDEXC または QCMDCHK の詳細については、 187 ページの『QCMDEXC プログラムの使用法』および 191 ページの『QCMDCHK プログラムの使用法』を参照してください。

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

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。しかし、次のように、QCMDEXC プログラムを用いたプログラムの処理時にコマンドが呼び出される場合は、

```
CALL QCMDEXC PARM('?OVRDBF FILE(FILEX)' 19)
```

FILE を含めて任意のパラメーターに値を指定できます。この例では、FILEX がデフォルト値です。

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)
```

プログラマー・メニューの使用

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 コマンドも、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、他のオプションをサポートします。

STRPGMMNU コマンドとそのパラメーターの詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

プログラマー・メニュー開始 (STRPGMMNU) コマンドの使用

プログラマー・メニュー開始コマンドは、次の目的で使用できます。

- QPGMMENU 呼び出しと同じ機能の実行
- 標準入力フィールドの記入

  4 つのコマンド・パラメーターにより、メニューの下部の標準入力フィールドに記入することができます。これらのパラメーターは、次のとおりです。

  – 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。

```
      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。たとえば、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。これは、メニューを使用するユーザーが STRPGMMNU コマンドを呼び出して、別個の呼び出しにおける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 QPGMMENU プログラムを直接呼び出すのを防ぐ (許可によって) 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することは防げません。

- 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。 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。 次に、EXITPGM パラメーターの一般的な用途をいくつか説明します。

EXITPGM パラメーター

EXITPGM パラメーターは、次の目的で使用できます。

- 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。

  たとえば、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、それがすでに使用されており、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。

- SBMJOB コマンド上の他のパラメーターを変更するため。

たとえば、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。 RTVJOBA コマンドを使用し、特定の値として属性を入力して、1 つまたは複数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。

  たとえば、すべての物理ファイルに P で終わる 7 文字の名前を付けることが必要な命名規則がある場合、出口プログラムで、この規則に従わない名前を指定した 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 1 つのコマンドの出口点を 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、コマンドに対して妥当性検査プログラム (VCP) を実行した後、かつコマンド処理プログラム (CPP) を実行する前に、これらの出口プログラムを呼び出します。

これらの出口点についての詳細は、iSeries Information Center の『**プログラミング**』カテゴリーにある『*API*』セクションを参照してください。

DBCS データ用のアプリケーション・プログラミング

2 バイト・データを処理するアプリケーション・プログラムを設計する場合、または英数字アプリケーション・プログラムを 2 バイト・プログラムに変換する場合には、特殊な考慮事項があります。

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しますが、以下の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。 「AS/400 適用業務開発ツール・セット 文字作成ユーティリティー」 では、DBCS がサポートされている国々に対して 2 バイト文字を定義する方法について説明しています。

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・プログラムを変換するには、次のことを行います。

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 プログラムでの DBCS データの使用

次のプログラムでは、CL プログラム内での異なるキーボード・シフトの使用法が示されています。このプログラムでは、2 バイト・データがテキスト値としてのみ使用されています。コマンド自体は英数字です。

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。

```
PGM

      DCLF    IGCTEST

START: CHGVAR &OUTPUTA  'ABCDEFGHIJ'

       CHGVAR &OUTPUTJ  'ABCD'
       CHGVAR &BOTHJ    'ABCD'
       CHGVAR &OUTPUTE  'EFGH'
       CHGVAR &OUTPUTO  'A B C  D  F'


LOOP:  SNDRCVF

      IF &INO1 RETURN
       CHGVAR &OUTPUTA   &INPUTA
       CHGVAR &OUTPUTJ   &INPUTJ
       CHGVAR &OUTPUTE   &INPUTE
       CHGVAR &BOTHE     &INPUTE
       CHGVAR &OUTPUTO   INPUTO


    GOTO LOOP

ENDPGM
```

RV3W197-0

サンプル CL プログラム

次のサンプル・プログラムは、CL プログラムの柔軟性、簡略性、用途の広さを示します。 次のプログラムは、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。 スレッドの追加情報については、iSeries Information Center にアクセスし、『**プログラミング**』カテゴリーの下にあるトピックをオープンしてください。

セットアップの初期プログラム (プログラマー)

```
            PGM
CHGLIBL LIBL(TESTLIB QGPL QTEMP)
CHGJOB OUTQ(WSPRTR)
TFRCTL QPGMMENU
    ENDPGM
```

テスト・ライブラリーは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。

オブジェクトのテスト・ライブラリーから実動ライブラリーへの移動 (プログラマー)

```
PGM PARM(&OBJ &OBJTYPE &OPER)
DCL &OBJ *CHAR LEN(10)
DCL &OBJTYPE *CHAR LEN(7)
DCL &OPER *CHAR LEN(1)  /* R=Replace M=Move */
IF ((&OPER *NE 'M') *AND (&OPER *NE 'R')) THEN(DO)
        SNDPGMMSG MSG('Operation code must be "R" or "M" ')
        RETURN
        ENDDO
IF ((&OBJTYPE *NE *PGM) *AND (&OBJTYPE *NE *FILE) *AND (&OBJTYPE +
    *NE *DTAARA)) THEN(DO)
        SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' must be *PGM, +
                  *FILE, or *DTAARA')
        RETURN
        ENDDO
CHKOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE)
        MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not exist +
                  in BLDLIB')
        RETURN
        ENDDO
IF (&OPER *EQ 'M') THEN(DO)
        MOVOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE) TOLIB(PRODLIB)
        MONMSG MSGID(CPF3208) EXEC(DO)
                        SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' +
                                  already exists in PRODLIB')
                        RETURN
                        ENDDO
        CHKOBJ PRODLIB/&OBJ OBJTYPE(&OBJTYPE)
        MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not +
                  exist in PRODLIB')
                  RETURN
                  ENDDO
        ENDDO
RETURN
ENDPGM
```

オブジェクト名、オブジェクト・タイプ、および操作コードが別のプログラムまたはプロシージャーに渡されます。操作コードとオブジェクト・タイプが正しいかどうか、およびオブジェクトがテスト・ライブラリー内に存在するかどうかを調べるための検査が実行されます。オブジェクトは、実動ライブラリーにすでに存在している場合を除き、移動されます。その際には移動が確認されます。オブジェクトへの追加権限を認可するため、または追加例外と追加オブジェクト・タイプを処理するために、新たにコマンドを追加できます。

アプリケーション内の特定のオブジェクトの保管 (システム・オペレーター)

例

```
PGM
SAVOBJ OBJ(FILE1 FILE2) LIB(LIBA) OBJTYPE(*FILE) DEV(TAP01) +
  CLEAR(*YES)
SAVOBJ OBJ(DTAARA1) LIB(LIBA) OBJTYPE(*DTAARA) DEV(TAP01)
SNDPGMMSG MSG('Save of daily backup of LIBA completed') +
  MSGTYPE(*COMP)
ENDPGM
```

このプログラムは、定期的に繰り返されるプロシージャーについて一貫性のあるコマンド入力を保証します。

もちろん、このほか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ディスケットまたは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ディスケットまたはテープ・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。たとえば、給与支払いファイルを 4 週間、週別に保管したい場合は、各ディスケットまたはテープに別々の名前を付け、ディスケットまたは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。

異常終了からのリカバリー (システム・オペレーター)

```
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
```

ジョブの投入 (システム・オペレーター)

```
PGM  /*DAILYAC*/
SBMJOB JOB(DAILYACCRC) JOBD(ACCRC2) +
             CMD(CALL ACCRC305 PARM(DAILY))
SNDPGMMSG    MSG('Daily Accounts Receivable job DAILYACCRC +
             submitted to batch') MSGTYPE(*COMP)
ENDPGM
```

ジョブ投入のためのすべてのパラメーターに入力する必要はなく、システム・オペレーターが DAILYAC を呼び出します。

ディスプレイ装置からの入力待ち時のタイムアウト

```
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
```

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。

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。 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.')

                          * * * * * *  E N D  O F  S O U R C E  * * * * *
```

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、'Invalid option Selected.' 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、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 SNDF コマンドは、SNDRCVF WAIT(*NO) の代わりに使用できます。機能は同じです。

日付算術の実行

```
/* 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
```

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。

プログラム属性の検索

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 API (QCLRPGMI) を使用できます。

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) コマンドにより、ユーザーは、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、実行することができます。

LODRUN コマンドを実行すると、

- 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 プログラムがすで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。
- RSTOBJ コマンドを使用して、QINSTAPP プログラムを QTEMP ライブラリーに復元します。
- 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、たとえば、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。

アプリケーション作成者の責任

QINSTAPP プログラムを提供するユーザーは、それを作成し、サポートする責任があります。 IBM* は QINSTAPP プログラムを提供しません。このプログラムは、多くの異なるタスクを実施するように作成できます。たとえば、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。

209 ページの図 3 は、QINSTAPP プログラムの例を示しています。このプログラムを、プログラム作成者がテープまたは光メディア上に保管し、LODRUN コマンドを用いてシステムにロードします。 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。

```
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 3. LODRUN コマンドを使用するアプリケーションの例

第 7 章 メッセージの定義

iSeries サーバーでは、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      06/12/94   10:50:54
  System going down at 11:00; please sign off
  ```

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。たとえば、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。オンライン・ヘルプまたは iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションにあるメッセージ記述追加 (ADDMSGD) コマンドの説明を参照してください。

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、いくつかの IBM 提供のメッセージ待ち行列があります。これらのメッセージ待ち行列は、この章の終わりで解説します (231 ページの『メッセージ待ち行列のタイプ』)。

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。ログ記録の詳細については、 309 ページの『メッセージのロギング』を参照してください。

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)
- メッセージおよびメッセージ記述の表示 (メッセージ表示 [DSPMSG] コマンド、メッセージ記述表示 [DSPMSGD] コマンド、メッセージの処理 [WRKMSG] コマンド、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。 (たとえば、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シージャーまたはプログラム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。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。

この章では、次のことを説明します。

- 独自のメッセージ・ファイルの作成方法
- メッセージ記述をメッセージ・ファイルに追加する方法
- メッセージ待ち行列のタイプ
- メッセージ待ち行列の作成方法

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。

CRTMSGF コマンドでは、SIZE パラメーター上に K 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、

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

S、I、および N のデフォルト値はそれぞれ、10、2、および *NOMAX です。

たとえば、S は 5、I は 1、N は 2 として指定します。ファイルが 5K の初期記憶量に達すると、システムは自動的に新たに 1K を初期記憶域に追加します。追加される量 (1K) は、記憶域に 2 回追加して、合計 7K 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 16M 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。

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・ファイルのサイズ判別

次の式を使用すると、メッセージのサイズを判別することができます。
(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・オンライン・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 (DFTPBM、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。

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、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 OS/400 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。

ファイルへのメッセージの追加

メッセージ記述追加 (ADDMSGD) コマンドを使用して、事前定義メッセージを記述し、それらを作成したメッセージ・ファイルに追加します。 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)

メッセージ記述に入れることができる項目はそれぞれ、次のページで詳しく解説します。

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

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 ～ 9 の 10 進数、および A ～ F の文字で構成されます。

以下にその例を示します。

```
CPF1234
```

これは、CPF のメッセージ 1234 です。

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 ～ 9 の 10 進数、および A ～ 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 ～ F は文字として処理されることに注意してください。

たとえば、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。詳細については、272 ページの『CL プログラムまたは CL プロシージャーによるメッセージの監視』の項を参照してください。

メッセージおよびメッセージ・ヘルプの定義

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。

重大度コードの割り当て

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、無効な場合があります。機能は部分的にしか完了されていないことがあります。たとえば、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。エラーがソース・データ内にあった場合、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。操作の結果は無効です。

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。これは、装置が誤動作しているか、または何らかの理由で操作不能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、ただ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。

SEV パラメーターについての詳細は、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

置換変数の定義

ADDMSGD コマンドの FMT パラメーターに、1 次レベルまたは 2 次レベルのメッセージ置換変数を指定できます。以下にその例を示します。

```
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。以下にその例を示します。

```
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、1〜99 の任意の数字です。たとえば、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。たとえば、&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。

たとえば、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、ADDMSGD コマンドの FMT パラメーターで &1、&2、&3 を定義しなければなりません。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。たとえば、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、たと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)
```

ADDMSGD コマンドには、データ・タイプおよび長さ (オプション) を指定することによって、置換変数に対応するメッセージ・データ・フィールドの形式を指定しなければなりません。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。

- 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アポストロフィで囲んだものです。この場合、先行ブランクまたは後書き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。
- 文字ストリング (*CHAR)。これは、アポストロフィで囲んでいない文字データのストリングです。 後書き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、アポストロフィで囲んでいない文字データのストリングです。 後書き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 に関するメッセージ処理プログラムの詳細については、iSeries Information Center の『**プログラミング**』カテゴリーにある『*グローバリゼーション*』トピックを参照してください。
- 16 進数 (*HEX)。アポストロフィで囲み、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 ～ 9 および A ～ 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。
- 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。
- 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。
- 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。
- 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。
- 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、ADDMSGD コマンドの DMPLST パラメーターの一部としてだけです。

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。

- 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)。
- 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。

応答に対する妥当性検査の指定

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

次の例では、プロシージャーが次のことを行います。

- 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、その文字が大文字または小文字の Y か N かどうかが、プロシージャー・ロジックによって検査されます。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、プログラムに応答は送られません。ブランクがプログラムに返された場合は、 (オペレーターが応答を返さなかったことになり) タイムアウトが生じます。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は 'Y' として示されます)。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、またはすで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。

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

ADDMSGD には、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。たとえば、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・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。既定時のメッセージ処理出口プログラムとこのパラメーターのフィールドの詳細については、 **iSeries Information Center** の『**プログラミング**』カテゴリーの『*API (APIs)*』セクションにある『メッセージ処理 API (Message Handling APIs)』を参照してください。
  - 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、次のものもダンプできます。
  - 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。デフォルト値処理の詳細については、276 ページの『デフォルトの処理』を参照してください。

デフォルト・プログラムの例

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、OPM CL プログラムまたは ILE プログラムにすることができます。

```
            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

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。警報の作成が可能な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。 OS/400 警報サポートの詳細については、「DSNX Support」


を参照してください。

メッセージの記述例

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、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 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、ADDMSGD コマンドを用いる 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、次のことを行います。

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。

   MONMSG コマンドを使用する場合には、比較データ (CMPDATA) パラメーターを 6 個の 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。

システム・メッセージ・ファイルの検索

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。

   詳細については、『メッセージ・ファイルの一時変更』の項を参照してください。

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。

   詳細については、『メッセージ・ファイルの検索』の項を参照してください。

メッセージ・ファイルの検索

メッセージ・ファイルが一時変更されていない場合、指定された (メッセージ・ファイル送信時に) メッセージ・ファイル名とライブラリーを使用して、メッセージ記述が取り出されるメッセージ・ファイルの検索が行われます。

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識別コードが入っていない場合も、指定されたメッセージ・ファイル名とライブラリーを使用してメッセージ・ファイルの検索が行われます。

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

プロシージャーまたはプログラムで使用されるメッセージ・ファイルを一時変更することができます。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) は、他のタイプの一時変更と類似しています。 ただし、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。

たとえば、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)
```

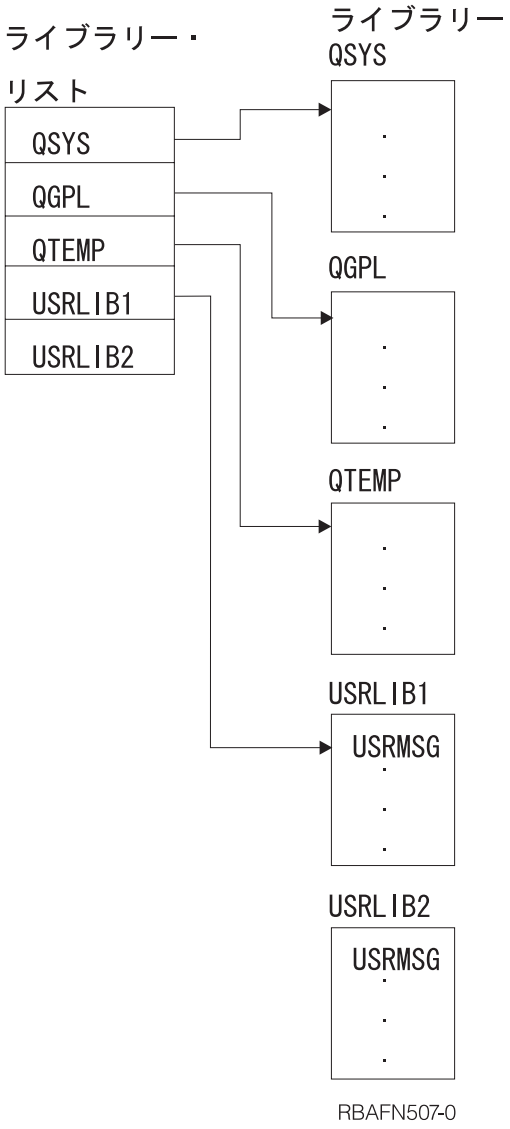
事前定義されたメッセージが検索または表示されるときは、メッセージ記述を見つけるために一時変更後のファイルが検索されます。そのファイルでメッセージ記述が検出されない場合は、一時変更前のファイルが検索されます。

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。たとえば、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。

たとえば、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。

メッセージ・ファイルの一時変更の例

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。たとえば、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。

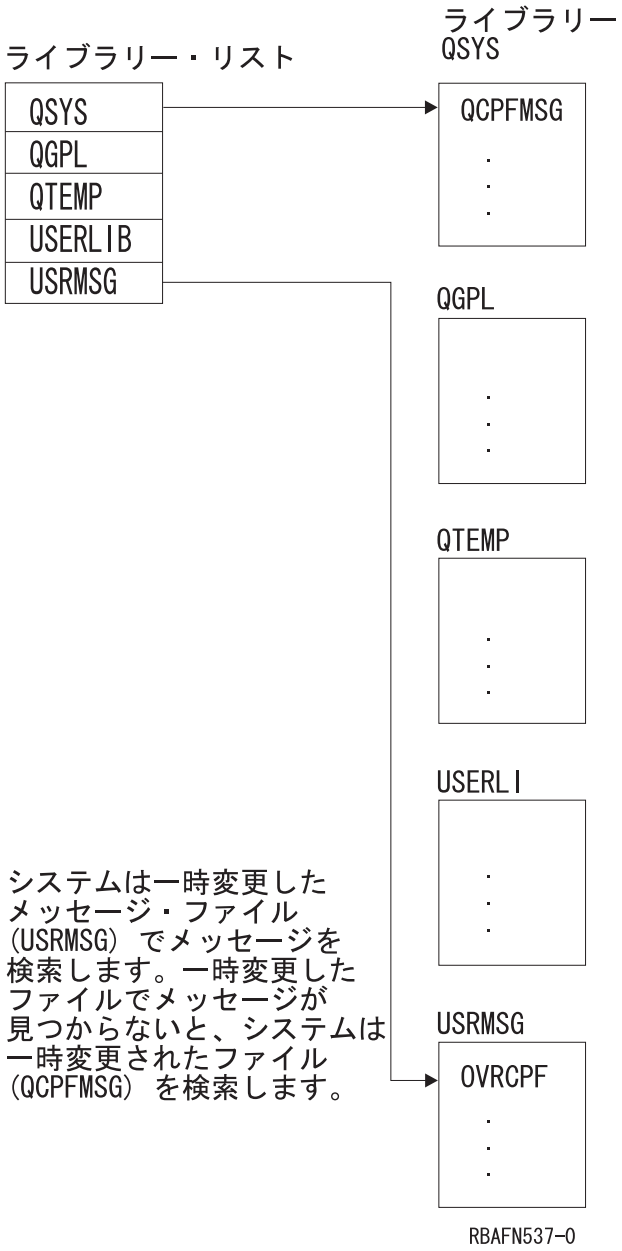
```
CRTMSGF MSGF(USRMSG/OVRCPF)
```

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('&2 のオブジェクト &1 が削除された。') +
         SEV(00) FMT((*CHAR 10) (*CHAR 10))
```

次に、そのジョブを実行する時点で、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。たとえば、1 つのジョブ内で次のような 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。

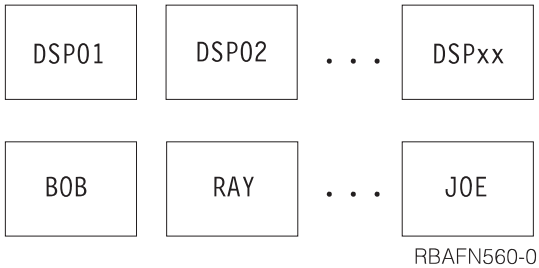
1. MSGFILE3
2. MSGFILE2
3. MSGFILE1

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のタイプ

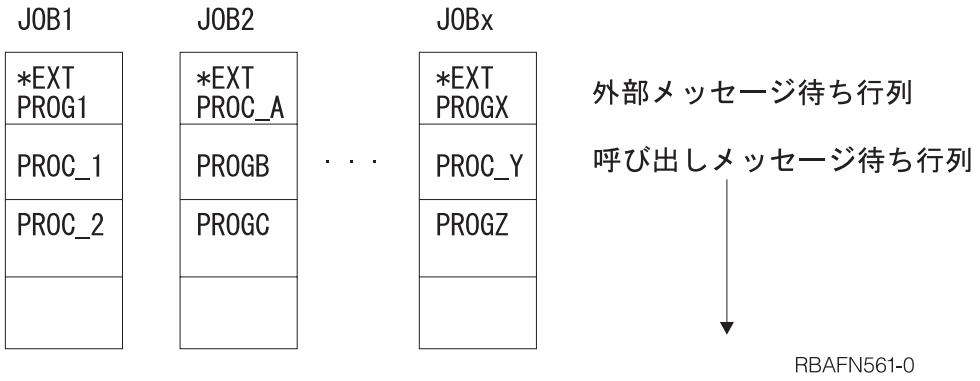
システム上の全メッセージは、メッセージ待ち行列に送信されます。そのメッセージ待ち行列に関連するシステム・ユーザーまたはプログラムは、その待ち行列からメッセージを受信します。同様に、メッセージに対する応答も、その応答を要求しているユーザーまたはプログラムのメッセージ待ち行列に送り返されます。

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各ディスプレイ装置ごと、および各ユーザー・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)。

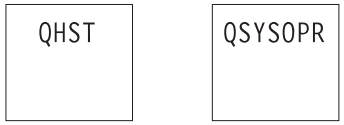


RBAFN560-0

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。各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 OPM プログラムまたは 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。



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。詳細については、237 ページの『ジョブ・メッセージ待ち行列』の項を参照してください。
- 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からの情報を活動記録ログ (QHST) に送るために使用されます。

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。

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ただちにディスクへ書き込まなければならないかどうか。変更がただ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、メッセージの表示 (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、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 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 コマンドによって要求するまで、メッセージはメッセージ待ち行列に保留されます。
  - 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。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。 出荷時の QSYSOPR では、この属性は元々折り返すように設定されています。
  - 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。

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、代替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。

代替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代替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、以上のような状況が生じることがあります。

独立 ASP 内のメッセージ待ち行列

独立 ASP の詳細については、『独立 ASP』を参照してください。

独立 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されています。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。

中断処理プログラム

中断処理プログラムは、メッセージの重大度が着信した中断転送モードのメッセージ待ち行列の重大度コード・フィルターと同じか、またはそれよりも高い場合に呼び出されます。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 CHGMSGQ コマンドに指定しなければなりません。メッセージ処理プログラムは、メッセージ受信 (RCVMSG) コマンドによってメッセージを受信しなければなりません。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。メッセージ受信および中断処理プログラムの詳細については、『第 8 章 メッセージ処理』を参照してください。

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。

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。詳細については、305 ページの『システム応答リストの使用法』の項を参照してください。

転送モードの変更の例

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。

```
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、他のユーザーがすでに QSYSOPR を中断モードにしている場合です。

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 ILE プロシージャーおよび 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。

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。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 (詳細については、309 ページの『ジョブ・ログ』を参照してください)。

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。

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。たとえば、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。以下にその例を示します。

```
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
```

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、その呼び出しメッセージ待ち行列は存在しなくなり、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) 256 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一番最近呼び出されたオカレンスを識別します。

呼び出しスタック項目を識別する方法はほかにもあります。これらの方法の詳細は、 249 ページの『SNDPGMMSG の呼び出しスタック項目識別コード』に記載されています。

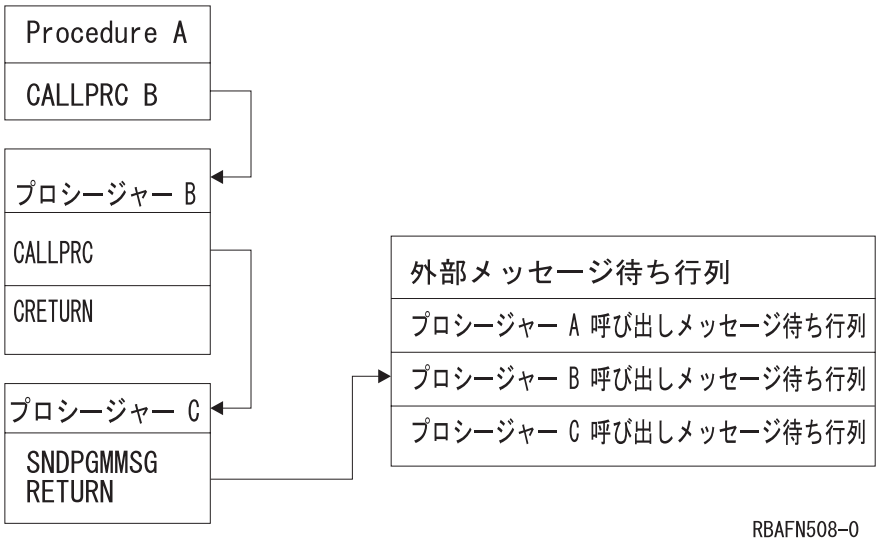
OPM または ILE プログラムが呼び出しスタック上にあるときにコンパイルされ、置換された場合は、プログラム名を使用して呼び出しスタック項目を参照するときに注意が必要です。置換操作が実行された時点より前からスタック上にある呼び出しスタック項目については、名前参照は置換されたオブジェクト (現在 QRPLOBJ 内に存在している) に解決されます。これらの名前参照は、置換されたオブジェクトが引き続き QRPLOBJ ライブラリー内に存在している限りは有効です。置換操作が実行された時点よりも新しいスタック項目については、名前参照はプログラムの

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。

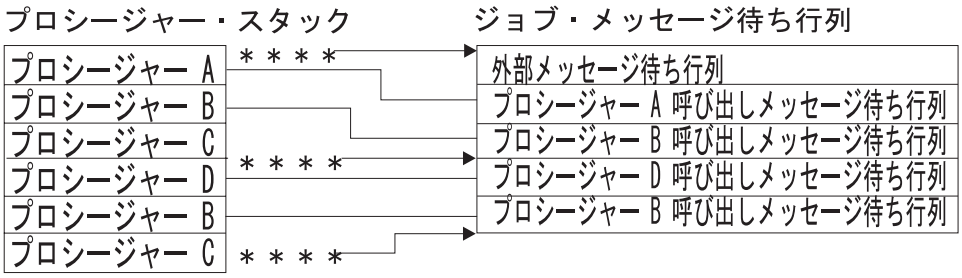
たとえば、プロシージャー A がプロシージャー B を呼び出し、プロシージャー B がプロシージャー C を呼び出し、プロシージャー C はプロシージャー B にメッセージを送信して終了するとします。 メッセージはプロシージャー B で使用可能です。しかし、プロシージャー B が終了すると、その呼び出しメッセージ待ち行列は使用できなくなるため、メッセージがジョブ・ログ内に示されている場合でも、プロシージャー A からプロシージャー B へはアクセスできません。プロシージャー A は、プロシージャー B に送信されるメッセージに対してメッセージ参照キーを持っていなければ、そのメッセージにアクセスできません。



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。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。

第 8 章 メッセージ処理

この章では、メッセージを使用してユーザーとプログラムの間における通信を行ういくつかの方法を説明しています。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 OPM プログラムまたは 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。

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・ユーザーへのメッセージの送信

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)

SNDPGMMSG および SNDUSRMSG は、バッチまたは対話式の OPM プログラム、または ILE プロシージャーの中でだけ使用することができます。これらのコマンドはコマンド入力行に入力できません。 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 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)
```

SNDBRKMSG コマンドでは、ワークステーション、プログラム、またはジョブから 1 つまたは複数のディスプレイ装置に即時メッセージを送るのに使用され、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、そのメッセージは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。
SNDBRKMSG コマンドは、ディスプレイ装置ユーザーの注意をただ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使用します。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 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。

CL プログラムからのメッセージの送信

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 (237 ページの『ジョブ・メッセージ待ち行列』を参照)
- 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 (237 ページの『ジョブ・メッセージ待ち行列』を参照)

- 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 8. メッセージ待ち行列のタイプ別の有効なメッセージ・タイプ

メッセージ・タイプ	メッセージ待ち行列のタイプ				
	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。

225 ページの『メッセージの記述例』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
```

また、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 (TOPGMQ パラメーター) を正しく指定できない場合もあります。このような場合には、外部メッセージ待ち行列 *EXT を指定します。

メッセージ

照会メッセージと情報メッセージ

SNDUSRMSG コマンドを使用すれば、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情報メッセージを送ることができます。 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。 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、SNDUSRMSG コマンドで以下の事項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。

完了メッセージと診断メッセージ

SNDPGMMSG コマンドを使用して、診断メッセージおよび完了メッセージを送ることができます。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、要求された機能がすで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、システム・オペレーター・メニューを使用して 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だけが含まれています。このプロシージャーはオブジェクトを保管してから以下の完了メッセージを出します。

```
PGM
SAVOBJ OBJ(PAY1 PAY2) LIB(PAYROLL) CLEAR(*YES)
SNDPGMMSG MSG('Payroll objects have been saved') MSGTYPE(*COMP)
ENDPGM
```

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 (これについては後で説明します)。 SAVOBJ コマンドが正常に完了した場合には、システム・オペレーター・メニュー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

状況メッセージは、SNDPGMMSG コマンドを使用して、CL プロシージャーまたは CL プログラムから呼び出しメッセージ待ち行列に、またはジョブの外部メッセージ待ち行列 (*EXT) に送ることができます。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。 272 ページの『CL プログラムまたは CL プロシージャーによるメッセージの監視』を参照してください。

エスケープ・メッセージと通知メッセージ

エスケープ・メッセージは SNDPGMMSG コマンドを使用して、CL プロシージャーまたは CL プログラムから呼び出し元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 プロシージャーまたは CL プログラムから、呼び出し元の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。 呼び出し側が統合言語環境 (ILE) のプロシージャーである場合、そのプロシージャーは次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 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。 272 ページの『CL プログラムまたは CL プロシージャーによるメッセージの監視』を参照してください。

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)
```

メッセージの送信の例

例 1: 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: 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変更します。その後、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: 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 文字を入力しなければなりませんが、プロシージャーはその応答値を使用していません。

```
PGM
DCL &MSGKEY TYPE(*CHAR) LEN(4)
SNDPGMMSG   MSG('Load special form') TOUSR(*SYSOPR) +
            KEYVAR(&MSGKEY) MSGTYPE(*INQ)
RCVMSG MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120)
.
.
.
ENDPGM
```

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、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 RCVMSG コマンドの次のコマンドに移って処理を続けます。 RCVMSG コマンドの MSGKEY パラメーターは、プロシージャーが特定のメッセージに対する応答を受け取れるようにするために使用されています。 SNDPGMMSG コマンド内の変数 &MSGKEY は、RCVMSG コマンドで使用されるように、プロシージャーに戻されます。

例 4: 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 (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: 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: 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・オペレーターがプログラムを呼び出すと、このプロシージャー (呼び出されるプログラムの中にある) 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、それを使用して、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
           CHGVAR &WORKHR &HOUR
           CHGVAR &WORKHR (&WORKHR - 12)
           CHGVAR &HOUR &WORKHR /* Change from military time */
           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
```

SNDPGMMSG の呼び出しスタック項目識別コード

CL プロシージャーが OPM プログラムまたは他の ILE プロシージャーにメッセージを送る予定がある場合には、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。

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

  TOPGMQ(*PRV *) 仕様は、基本項目が 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。

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。

251 ページの図 4 を参照して、CLPROC2 が CLPROC4 を呼び出した時点の呼び出しスタックの構造を調べてください。この図では以下の考慮事項が説明されています。

- 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。

基本呼び出しスタック項目を指定する様々な方法を次に説明します。

コマンドを基本項目として使用するプロシージャー

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。プログラムの呼び出し側 (この例では 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。

252 ページの図 5 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戻した場合の結果が図示されています。

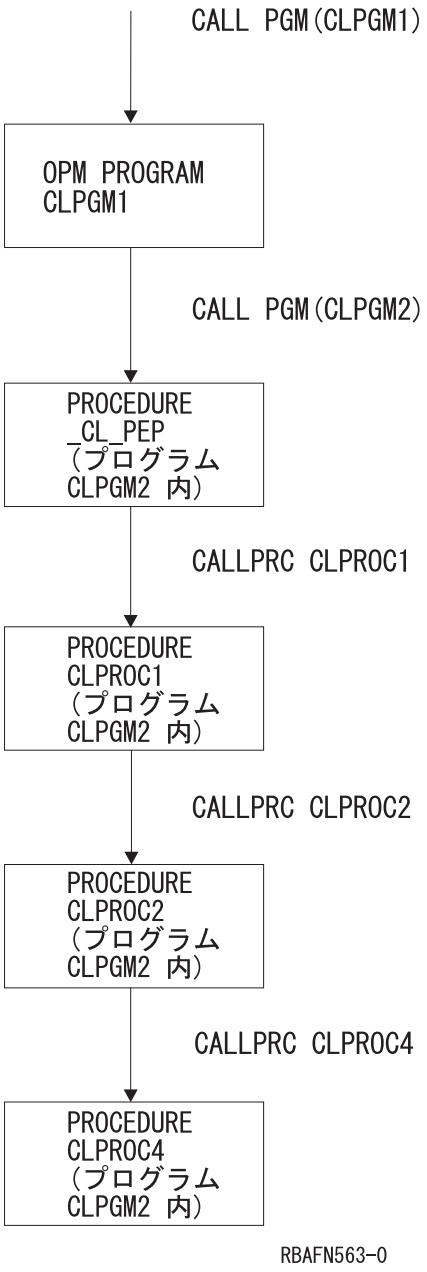


図 4. 実行時呼び出しスタックの例

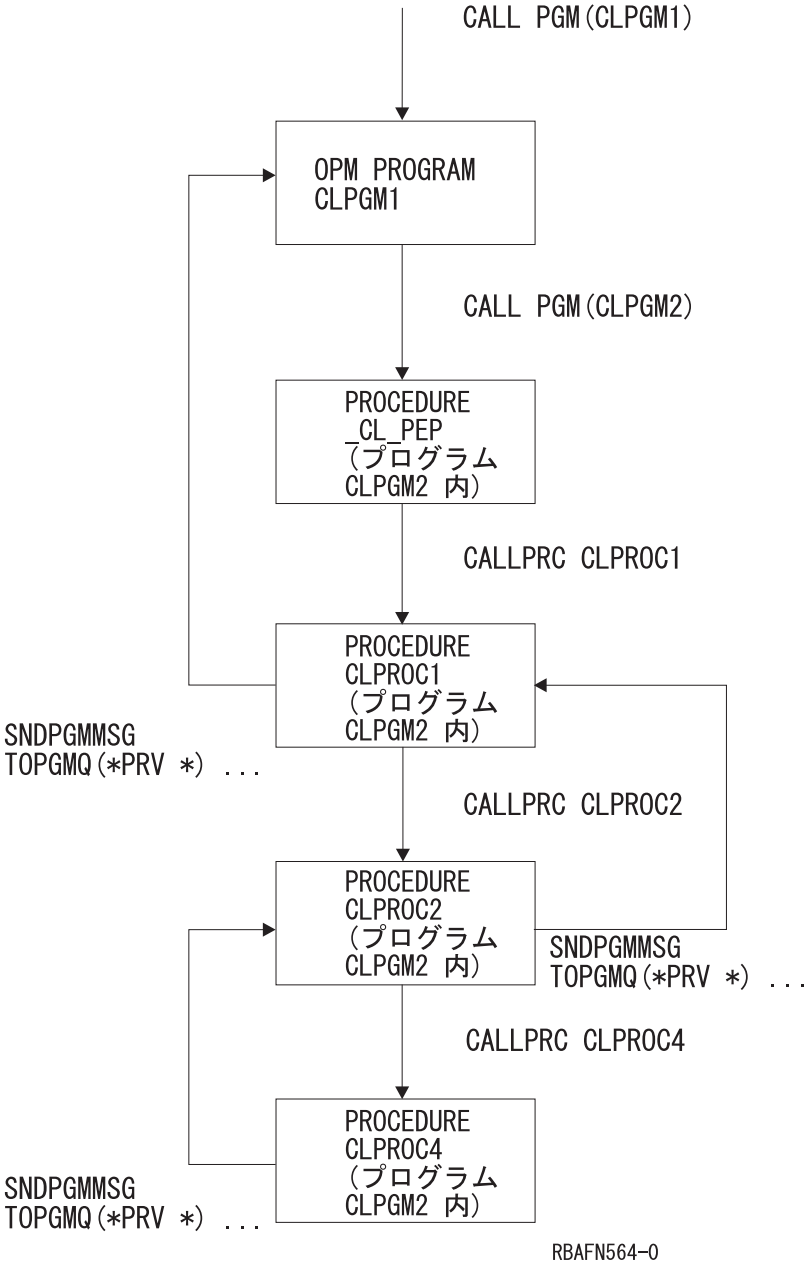


*図 5. TOPGMQ(*PRV *) の例*

名前による基本項目の識別

基本項目中で実行している OPM プログラムか ILE プロシージャーの名前を付けることによって、その項目を識別できます。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、254 ページの図 6 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、255 ページの図 7 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。 IBM は、部分呼び出しスタック項目名を指定する方法に関するオンライン情報を提供しています。 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。

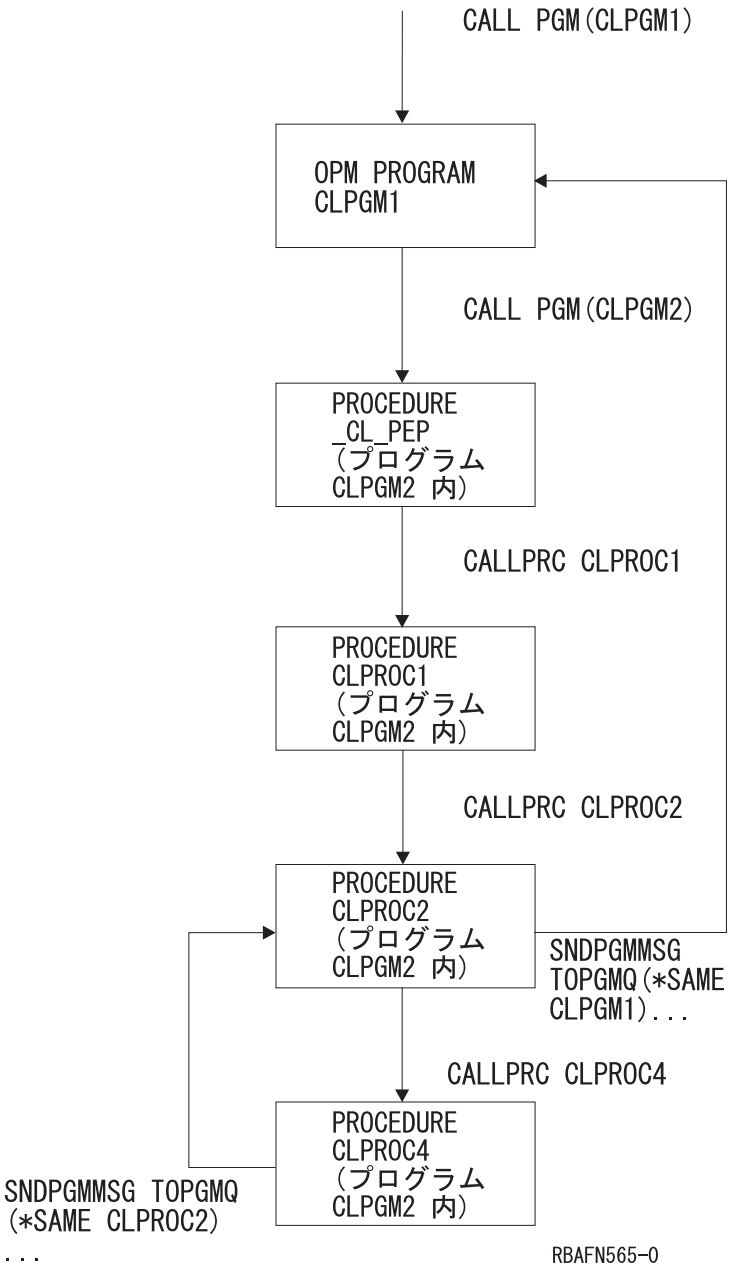

CALL PGM(CLPGM1)
OPM PROGRAM
CLPGM1
CALL PGM(CLPGM2)
PROCEDURE
_CL_PEP
(プログラム
CLPGM2 内)
CALLPRC CLPROC1
PROCEDURE
CLPROC1
(プログラム
CLPGM2 内)
CALLPRC CLPROC2
PROCEDURE
CLPROC2
(プログラム
CLPGM2 内)
SNDPGMMSG
TOPGMQ(*SAME
CLPGM1)...
CALLPRC CLPROC4
PROCEDURE
CLPROC4
(プログラム
CLPGM2 内)
SNDPGMMSG TOPGMQ
(*SAME CLPROC2)
...
RBAFN565-0

図 6. 単純名の使用例

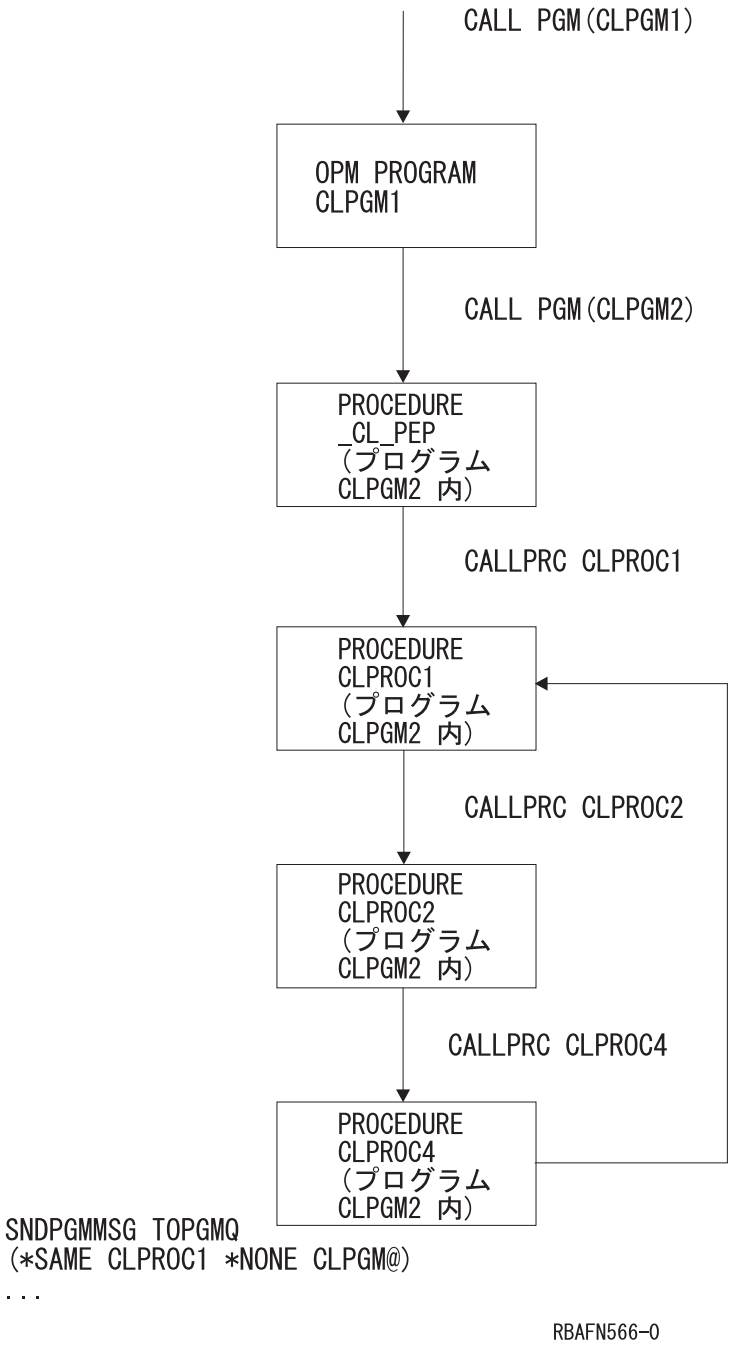


図 7. 複合名の使用例

基本項目としてのプログラム境界

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信する例については、257 ページの図 8 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、 258 ページの図 9 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 258 ページの図 9 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 ILE プログラムの 1 つです。
- 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。

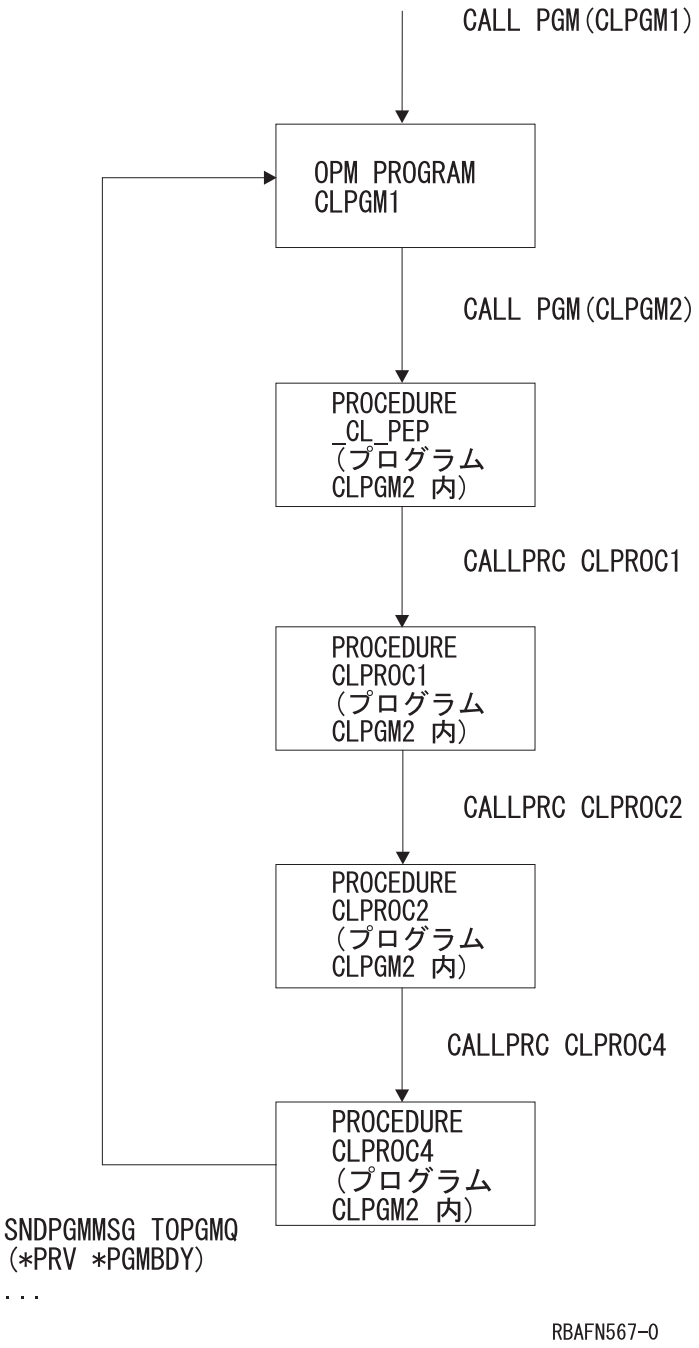


*図 8. *PGMBDY の使用例 1*

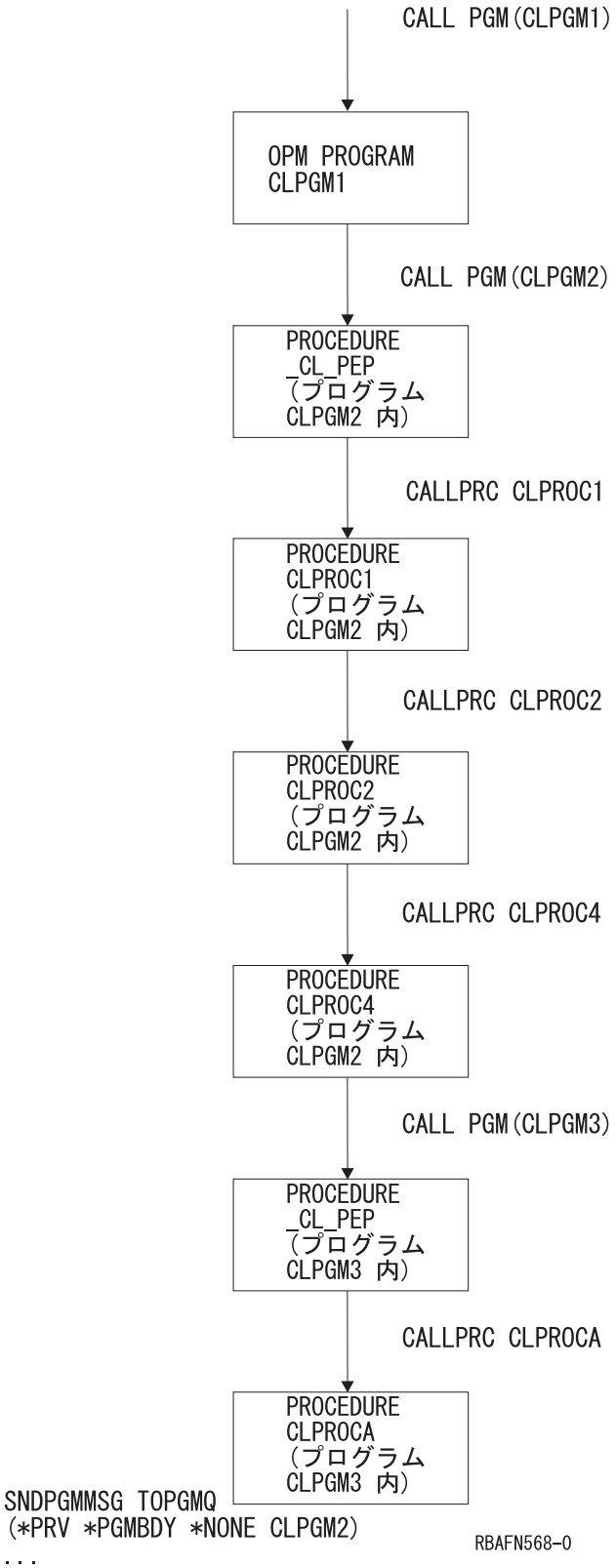


*図 9. *PGMBDY の使用例 2*

OPM プログラムを使用する場合も特殊値 *PGMBDY を使えます。 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。たとえば 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。 図 10 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。

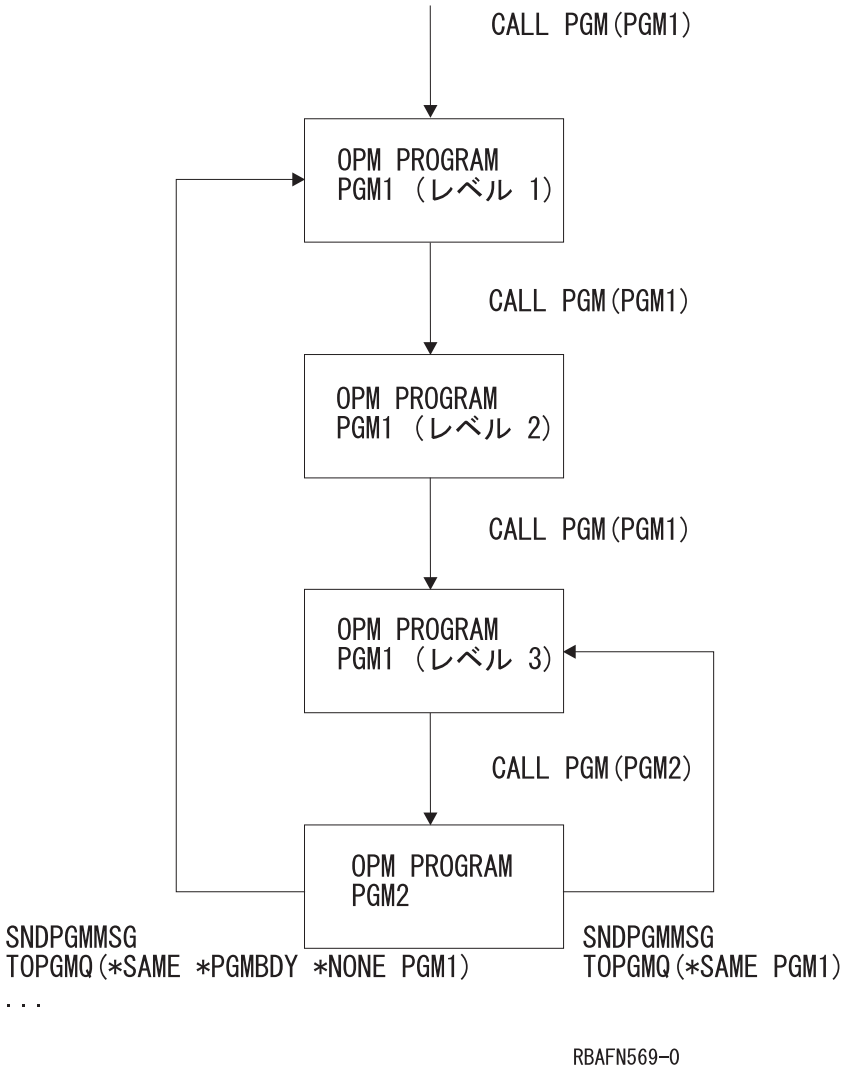


*図 10. *PGMBDY の使用例 3*

基本項目として最後に呼び出されたプロシージャー

プロシージャーの名前はわからないが、最後に呼び出された ILE プログラムのプロシージャーにメッセージを戻したい場合があります。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。

- CLPGM1。これはプロシージャー PROCA と PROCB を含む ILE プログラムです。
- CLPGM2 と CLPGM3。これは両方とも OPM プログラムです。
- CLPGM3。これはメッセージを CLPGM1 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 CLPGM1 を使用して送信を行います。

特殊値 *PGMNAME を使用してメッセージを送信する例については、 261 ページの図 11 を参照してください。

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。たとえば、CLPGM1 は OPM CL プログラムです。CLPGM3 はメッセージを CLPGM1 に送り、TOPGMQ(*SAME CLPGM1) を指定します。 CLPGM1 が ILE に変換されると、CLPGM3 (OPM) 内の SNDPGMMSG コマンドだけが実行可能です。 CLPGM1 は、CLPGM1 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 CLPGM1) に変更すると、CLPGM3 は、プロシージャー名に使用した名前にかかわらず、CLPGM1 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。たとえば、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。

基本項目としての制御境界の使用

特殊値 *CTLBDY を使用すると、基本項目を最も近い制御境界の項目として識別できます。 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、 262 ページの図 12 を参照してください。この例の 3 つのプログラム (CLPGM1、CLPGM2、および CLPGM3) はすべて ILE プログラムです。 CLPGM1 は活動化グループ AG1 で実行し、CLPGM2 と CLPGM3 の両方は活動化グループ AG2 で実行します。この例では、PROC3A は AG2 の境界の直前にある項目にメッセージを送ります。

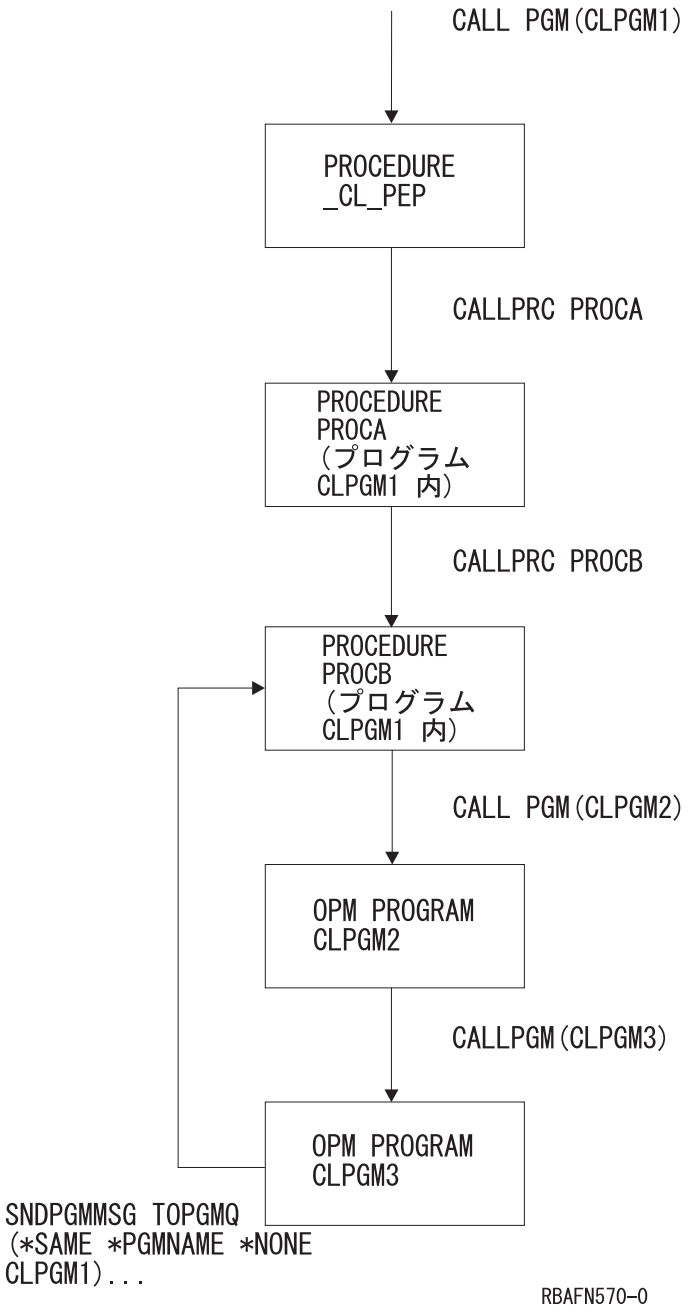


図 11. 実行時呼び出しスタックの例

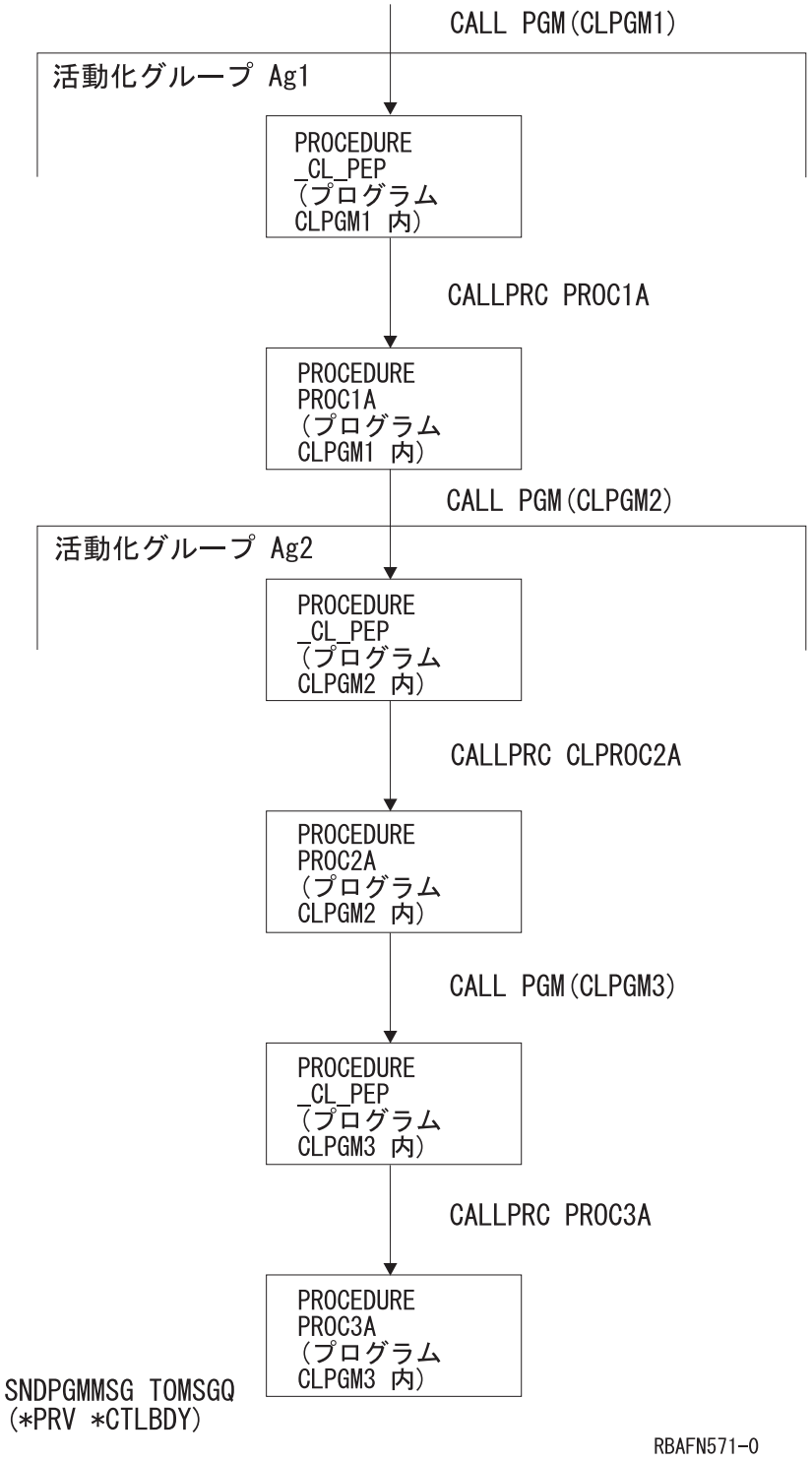


*図 12. *CTLBDY の使用例*

サービス・プログラムに関する考慮事項

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。たとえば、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 CL プログラムによるメッセージの受信

ユーザーのプロシージャーまたはプログラムでメッセージ待ち行列からメッセージを受け取る場合には、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、RCVMSG コマンドにこの CL 変数を指定することによってキーを渡さなければなりません (MSGKEY パラメーター)。
  - 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・タイプを指定したり、全タイプを指定できます。
- 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、RCVMSG コマンドを実行しているプロシージャーか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、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を新しいメッセージにリセットすることができ、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るようになります。ただし、すで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。 (詳細については、 270 ページの『メッセージ待ち行列からのメッセージの除去』を参照してください。)
- 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 (各事項がそれぞれ 1 つの変数に対応します)。
  - 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 (80 文字の文字変数)
  - 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。

```
RCVMSG MSGQ(QGPL/INVN) MSGTYPE(*ANY) MSG(&MSG)
```

受信されたメッセージは、変数 &MSG に入れられます。 *ANY は、MSGTYPE パラメーターのデフォルト値です。

CL 以外の言語で作成された 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。 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のシステムへの通知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。

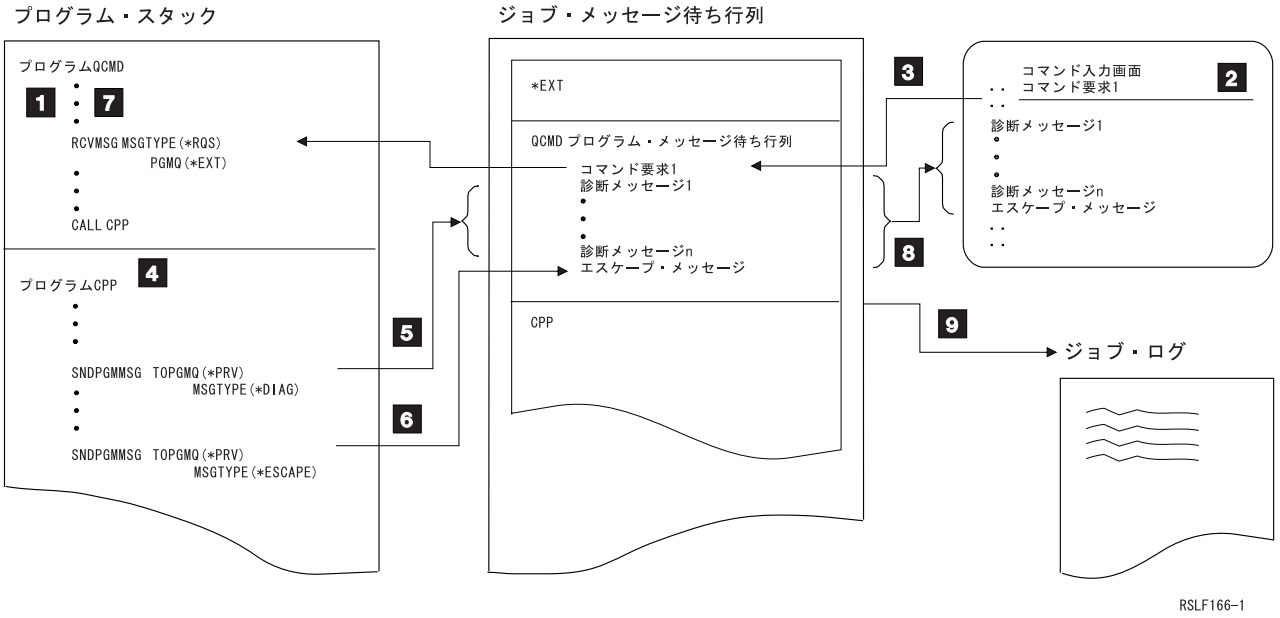
要求メッセージ

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。たとえば、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を受け取り、プログラム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。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。たとえば、CL コマンドは IBM 提供の CL 処理プログラム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 OPM プログラムまたは 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 次に、そのコマンドは、QCMD の呼び出しメッセージ待ち行列の終わりに移され、そこから QCMD に渡されます。

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。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。詳細は、 309 ページの『メッセージのロギング』を参照してください。

要求処理プロシージャーおよびプログラムの作成

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。

- 265 ページの『要求メッセージ』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。

ユーザーの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、その中にプログラム・メッセージ送信 (SNDPGMMSG) コマンドおよびメッセージ受信 (RCVMSG) コマンドを組み込まなければなりません。たとえば、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れは、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されます (実際には、移動され、リセットされる)。したがって、メッセージが除去されるときに正しい呼び出しメッセージ待ち行列が使用されていることが必要です。

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。 (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 コマンドに指定しなければなりません。

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 (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 RMVMSG 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てしまうまで、要求

処理プログラムとして機能します。たとえば、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)
```

OS/400 コマンド分析プログラムに OS/400 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、メッセージ検索キーを指定してください。メッセージ検索キーは、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 にメッセージ検索キーが提供されていない。

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確認するには、そのジョブの呼び出しスタックを表示します。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に表示される要求レベル列に番号がある場合、番号のあるプログラムまたは 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


         プログラム
 REQ     または
 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    F10= スタックの更新    F11= 活動化グループの表示    F12= 取り消し
F17= 最上部    F18= 最下部
```

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、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。 RMVMSG コマンドにより要求処理は終了するので、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、RCVMSG コマンドは要求プロセッサーになるために必要なパラメーターの最小の数を使用しています。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。

CL プロシージャーによるメッセージの検索

メッセージ検索 (RTVMSG) コマンドを使用すると、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、それを変数に入れることができます。 RTVMSG は、事前定義メッセージの記述に操作を行います。以下の事項に加えて、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。これは、置換変数に入れるメッセージ・データです。
- 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)
  - 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。

たとえば、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)
```

&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。 CL 変数 &MSG には、次のメッセージが入れられます。

```
File INVENT not found in library QGPL
```

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れられます。

```
File not found in library
```

変数 &MSG にメッセージが入れられた後、次のことを行うことができます。

- SNDPGMMSG コマンドを使っ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。

注: テキスト内に変数名が含まれているメッセージ・テキストを検索することはできません。 システムは、RTVMSGD で送信可能なメッセージを返そうとします。

メッセージ待ち行列からのメッセージの除去

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、メッセージ受信 (RCVMSG) または応答送信 (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の消去オプションによって除去されるまで、メッセージはメッセージ待ち行列に保持されます。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

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 RCVMSG 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、その除去したいメッセージのメッセージ参照キーを指定してください。

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への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しようとした場合には、そのメッセージの送信側にデフォルトの応答が送られた上で、その照会メッセージとそのデフォルトの応答が除去されます。すで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ユーザーの応答の両方が除去されます。

すべての非活動プログラムおよびプロシージャーに関する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したい場合には、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。

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には、処理されていない例外に対する例外メッセージが RMVMSG コマンドの実行時に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、例外は処理されません。

以下の RMVMSG コマンドによって、ユーザー・メッセージ待ち行列 JONES からメッセージが 1 つ除去されます。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 RMVMSG コマンドは、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)
```

注: ユーザー・メッセージ待ち行列またはワークステーション・メッセージ待ち行列でのメッセージ最大数は、送信されるメッセージのタイプごとに 65 535 です。たとえば、待ち行列には診断メッセージを 65 535 だけ入れることができ、完了メッセージも 65 535 までというようになります。ただし、呼び出しメッセージ待ち行列や *EXT 待ち行列については、タイプごとのメッセージ最大数の制限はありません。

CL プログラムまたは CL プロシージャーによるメッセージの監視

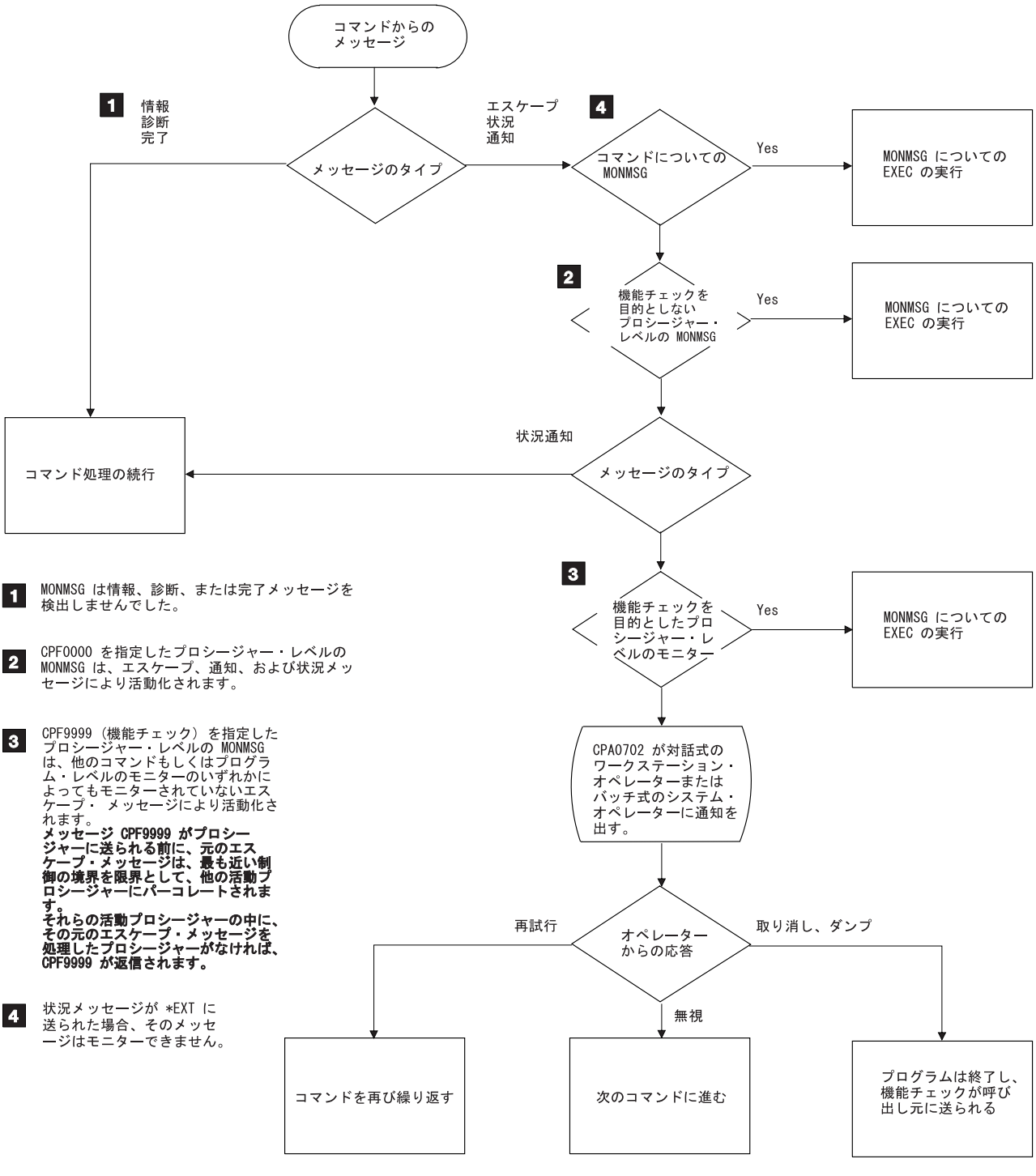
ユーザーの CL プロシージャーまたは CL プログラムの呼び出しメッセージ待ち行列に送られてくるエスケープ、通知、および状況の各メッセージを、別のプロシージャーまたはプログラムのコマンドによって監視することができます。メッセージ・モニター (MONMSG) コマンドは、そのコマンドに指定されている条件に応じて、呼び出しメッセージ待ち行列に送られてくるメッセージを監視するためのものです。該当する条件が存在すれば、MONMSG コマンドで指定した CL コマンドが実行されます。 MONMSG コマンドによるロジックは以下のとおりです。

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、そのユーザー・プロシージャーまたはそのプログラム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することができます。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 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。



RV3W196-1

- 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。たとえば、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。たとえば、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。たとえば、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

  すべての総称メッセージ識別コードについて機能チェック・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、そのメッセージのデータと比較することもできます。たと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。ファイルの名前はメッセージ・データとして送られてきます。

```
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ますが、その場合には、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 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、CALL コマンドが正しく実行され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、プロシージャーまたはプログラム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、プロシージャーまたはプログラムはその 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箇所でエスケープ・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。そのためには 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。 IBM は、CL コマンドに関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るオンライン・ヘルプを提供しています。 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。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: 上記の段落は、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。

デフォルトの処理

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。これらのすべてのメッセージを監視して処理する必要はないが、ユーザーの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。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。

デフォルトの処置では、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、以下のことを行います。

- 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。
- エスケープ用の MONMSG コマンドがプロシージャーにあるかどうか調べる。

プロシージャーにエスケープ用の MONMSG コマンドがある場合には、メッセージ・パーコレーション処置は停止し、システムは MONMSG コマンドで指定した処置をとります。 メッセージ・パーコレーションは 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 MONMSG コマンドを含む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例外に関する処置は完了したと見なします。 それからシステムは、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となっていた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 MONMSG がこの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操作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 プロシージャーを取り消して、機能チェックを呼び出しスタック上の直前のプロシージャーにパーコレートします。

D 失敗したプロシージャーの呼び出しスタック項目をダンプし、プロシージャーを取り消し、機能チェックを呼び出しスタック上の直前のプロシージャーにパーコレートします。 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、次のいずれかの処置をとることができます。

- プロシージャーを終結処置し、終了する。
- プロシージャーの他の部分に移って処理を続行する。

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。

通知メッセージ

エスケープ・メッセージ・モニターに加えて、ユーザーの CL プロシージャーまたは CL プログラムのコマンドあるいはそれらが呼び出した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することができます。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。活動化グループ境界に達したが境界用の 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の応答が自動的に戻され、送信側は処理を継続できます。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

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または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 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。たとえば、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。 OS/400 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。

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。 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。メッセージ受信 (RCVMSG) コマンドのオンライン・ヘルプ情報では、中断処理出口プログラムのパラメーターについて説明されています。 **iSeries Information Center** の『**プログラミング**』カテゴリーの『*CL*』セクションにあるコマンド資料を参照してください。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が終了すると、元の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。
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。メッセージ受信 (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。

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上記の例で、CPA5316 メッセージが 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。詳細は、 305 ページの『システム応答リストの使用法』を参照してください。

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 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。

QSYSMSG メッセージ待ち行列

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 (次の『QSYSMSG メッセージ待ち行列に送られるメッセージ』の項を参照) 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に送られるメッセージ

このトピックでは、QSYSMSG メッセージ待ち行列に送られる特定のメッセージについて説明しています。 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 ～ CPI96C7

システムは、特定のメッセージを、共に送られる SRC (system reference code) とその SRC が重大メッセージ処理でログに記録されているかどうかに応じて、QSYSMSG と QSYSOPR の両方、またはその一方だけに送信します。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。詳細については、「バックアップおよび回復の手引き」


を参照してください。

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。

- 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。
- 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 (たと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能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。

再び要求を実行してください。プログラム・リカバリー時の勧告については、「Communications Management」  を参照してください。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)、または電源オフの通知 (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。センス・データおよび関連エラーの詳細については、「Finance Communications Programming」


を参照してください。

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。

装置をオンに構成変更して、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 OS/400 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。

再び要求を実行してください。 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)。 プログラム・リカバリーについての詳細は、「Communications Management」  を参照してください。

CPF5344

制御装置 &9、制御記述 &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。

再び要求を実行してください。 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。たとえば、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

オペレーティング・システム/400 の猶予期間が満了した。

オペレーティング・システム/400® 用のソフトウェア・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。

新しいオペレーティング・システム/400 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た。

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。上記のディスク装置情報を使用して、ハードウェア・サービス・マネージャー・サービス・ツールで装置を見付け、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、システムを電源遮断せず、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、システムを電源遮断せず、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。システムの電源をオフにしないでください。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るときはいつも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、圧縮リカバリー方針を変更してください。

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。

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。削除される前のユーザー・プロファイルはオブジェクトを所有していることがあります。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・エラーが検出された。

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。 ただちに、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 &27 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 &27 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 &27 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 &27 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

9406 システム装置に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 ただ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 ただ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。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ていることを示しています。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。そうすることで、低下したシステム・パフォーマンスを障害からすくうことができるでしょう。

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 ただちに、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 ただちに、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。

QSYSMSG からメッセージを受け取るためのサンプル・プログラム

次に示すのは、QSYSMSG メッセージ待ち行列からメッセージを受け取るサンプル・プログラムです。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、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・オペレーターにも何が起こったのか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 ENDJOB コマンドによって終了させることができます。

```
/*********************************************************************/
/*                                                                   */
/*  Sample 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
```

システム応答リストの使用法

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応答するように指定することができ、ディスプレイ装置ユーザーが応答する手間を省くことができます。この自動応答は、照会メッセージに対してだけ行うことができます。

システム応答リストにはメッセージ識別コード、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識別コード、および必要があれば応答メッセージの比較データとの比較によって対応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	ダンプ
10	CPA0700	*NONE	D	*YES
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 ～ CPA0799、RPG0000 ～ RPG9999、CBE0000 ～ CBE9999、または PLI0000 ～ 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、ジョブでシステム応答リストの使用を指定しておかなければなりません。

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。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	比較開始位置	応答	ダンプ
10	CPA0700		1	D	*YES
15	RPG1241		1	C	*NO
18	RPG1200		1	*DFT	*YES
20	RPG0000		1	D	*YES

順序番号	メッセージ ID	比較値	比較開始位置	応答	ダンプ
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。メッセージ RPG1241 については、この項目よりも前に追加されている項目の方が優先されます。
- 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 &3 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。

&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。 IBM は、メッセージ記述追加 (ADDMSGD) コマンド *CCHAR データに関するオンライン情報を提供しています。 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。

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。

応答処理

応答処理出口プログラムを使用すれば、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。応答処理出口プログラムの詳細については、 **iSeries Information Center** の『**プログラミング**』カテゴリーの『*API (APIs)*』セクションにある『メッセージ処理 API (Message Handling APIs)』を参照してください。

メッセージのロギング

メッセージについてのログは 2 種類あります。

- ジョブ・ログ
- 活動記録ログ

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。活動記録ログ (QHST) には、システムにおけるジョブの開始や終了の活動の記録などのシステム・データが入ります。

ジョブ・ログ

各ジョブにはジョブ・ログがそれぞれ 1 つずつあり、そのジョブに関する次のような事項が記録されます。

- 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 (CL プログラムのコマンドのロギングに関する詳細については、67 ページの『CL プロシージャーのコマンドのロギング』の項を参照)。
- 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。

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。ジョブが正しく完了した場合に、ジョブ・ログが書き込まれないように指定することができます。この章で後述する、ジョブ・ログの生成の抑止についての説明を参照してください。

QMHCTLJL API を使用すれば、データベース・ファイルにジョブ・ログを書き込めます。 QMHCTLJL API については、**iSeries Information Center** の『**プログラミング**』カテゴリーにある『*API (APIs)*』セクションを参照してください。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。 1 次ファイルと 2 次ファイルの様式に関する情報は、457 ページの『付録 B. ジョブ・ログ出力ファイル』を参照してください。

ジョブ・ログに記録する情報を、ユーザーが制御することができます。 そのためには、ジョブ記述作成 (CRTJOBD) コマンドの LOG パラメーターに必要な値を指定します。 また、これらの値はジョブ変更 (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。これらのコマンドの詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

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・コマンド」属性によって許可される場合に 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・コマンド」属性によって許可される場合に 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。 たとえば、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 ～ 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。

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。**メッセージをフィルターにかける**とは、ジョブに対して設定されるメッセージ・ロギング・レベルに基づいて、ジョブ・ログからのメッセージを除去する処理のことです。

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。

以下の例は、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る場合、各コマンドの後でフィルターがかけられることを示しています。

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 として識別され、詳細メッセージは*詳細*なメッセージ として識別されています。

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、それ自身の呼び出しメッセージ待ち行列と、指定された呼び出しの前に呼び出されるプログラムの呼び出しメッセージ待ち行列 (*PRV) に、重大度コードが 20、50、および 60 の 3 つの情報メッセージを送ります。 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。また、PGMB は、*PRV に重大度コードが 10 の情報メッセージを 1 つ送ります。

   この時点で CHGJOB コマンドが画面には表示されていない点に注意してください。ロギング・レベル 2 に従って、指定された重大度と同じかまたはそれより大きい重大度を持つメッセージが出された要求だけがジョブ・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。このようなメッセー

ジが出されていた場合、詳細なメッセージがあればそれもジョブ・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
```

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、入力された要求に関連するすべてのメッセージが表示されます。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。

```
                                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 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。

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、指定された呼び出しの前に呼び出されるプログラムの呼び出しメッセージ待ち行列 (*PRV) に、重大度コードが 30 および 40 の 2 つのメッセージを送ります。

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。

   コマンド SIGNOFF *LIST は、ジョブ・ログを印刷するために入力されます。

```
                              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、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。

```
 5722SS1 V5R3M0 040430                    ジョブ・ログ                        SYSAS727 02/01/18 07:58:53  ページ    1
  ジョブ名 . . . . . . . . . . :   QPADEV0007     ユーザー  . . . . . :   JOHNDOE     番号. . . . . . . . . . . :   004201
  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
MSG1124   INFORMATION      00   02/01/18  07:57:16     QWTPIIPP    QSYS       04FC    *EXT                  *N
                                 Message    . . . . :   QSYS のサブシステム QINTER のジョブ004201/JOHNDOE/QPADEV0007 が
                                   07:57:16 に開始された。ジョブは 02/01/18 07:57:16 にシステムに入れられました。
*NONE     REQUEST                02/01/18  07:57:50      QMHGSD      QSYS       0322    QCMD  QSYS          00B6
                                 Message    . . . . :  -CALL PGMA
MSG1001   INFORMATION      20   02/01/18  07:57:50      PGMA        JOHNDOE    000C    PGMA  JOHNDOE       000C
                                 Message    . . . . :   詳細なメッセージ重要度 20 - PGMA
                                 MSG1001 2 次レベル・テキスト - PGMA
MSG1002   INFORMATION      50   02/01/18  07:57:50      PGMA        JOHNDOE    0010    PGMA  JOHNDOE       0010
                                 Message    . . . . :   詳細なメッセージ重要度 50 - PGMA
                                 MSG1002 2 次レベル・テキスト - PGMA
MSG1003   INFORMATION      60   02/01/18  07:57:50      PGMA        JOHNDOE    0014    PGMA  JOHNDOE       0014
                                 Message    . . . . :   詳細なメッセージ重要度 60 - PGMA
                                 MSG1003 2 次レベル・テキスト - PGMA
MSG1004   INFORMATION      20   02/01/18  07:57:50      PGMA        JOHNDOE    0018    QCMD  QSYS          00DE
                                 Message    . . . . :   メッセージ重要度 20 - PGMA
                                 MSG1004 2 次レベル・テキスト - PGMA
MSG1005   INFORMATION      50   02/01/18  07:57:50      PGMA        JOHNDOE    001C    QCMD  QSYS          00DE
                                 Message    . . . . :   メッセージ重要度 50 - PGMA
                                 MSG1005 2 次レベル・テキスト - PGMA
MSG1006   INFORMATION      60   02/01/18  07:57:50      PGMA        JOHNDOE    0020    QCMD  QSYS          00DE
                                 Message    . . . . :   メッセージ重要度 60 - PGMA
                                 MSG1006 2 次レベル・テキスト - PGMA
*NONE     REQUEST                02/01/18  07:58:31      QMHGSD      QSYS       0322    QCMD  QSYS          00B6
                                 Message    . . . . :  -CHGJOB LOG(3 40 *SECLVL)
*NONE     REQUEST                02/01/18  07:58:34      QMHGSD      QSYS       0322    QCMD  QSYS          00B6
                                 Message    . . . . :  -CALL PGMC
MSG100F   INFORMATION      30   02/01/18  07:58:34      PGMC        JOHNDOE    *STMT   QCMD  QSYS          00DE
                                 Message    . . . . :   メッセージ重要度 30 - PGMC
                                 MSG100F 2 次レベル・テキスト - PGMC
MSG1010   INFORMATION      40   02/01/18  07:58:34      PGMC        JOHNDOE    *STMT   QCMD  QSYS          00DE
                                 Message    . . . . :   メッセージ重要度 40 - PGMC
                                 MSG1010 2 次レベル・テキスト - PGMC
*NONE     REQUEST                02/01/18  07:58:38      QMHGSD      QSYS       0322    QCMD  QSYS          00B6
                                 Message    . . . . :  -CALL PGMD
*NONE     REQUEST                02/01/18  07:58:45      QMHGSD      QSYS       0322    QCMD  QSYS          00B6
                                 Message    . . . . :  -CALL PGME
*NONE     REQUEST                02/01/18  07:58:52      QMHGSD      QSYS       0322    QCMD  QSYS          00B6
                                 Message    . . . . :  -SIGNOFF *LIST
MSG1164   COMPLETION       00   02/01/18  07:58:52      QWTMCEOJ    QSYS       01EE    *EXT  0000
                                 Message    . . . . :   07:58:52 に 02/1/18 でジョブ 004201/JOHNDOE/QPADEV0007 が終了した。
                                   1 秒が使用され、終了コードは 30 です。
                                 原因 . . . . . :   1 秒の処理装置時間を使用した後で、07:58:52に 02/01/18 でジョブ
                                  004201/JOHNDOE/QPADEV0007 が完了しました。ジョブの終了コードは 30 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。上記のサンプル・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た。
- 送信側が 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。追加情報については、『プログラムまたはプロシージャーの送信または受信』を参照してください。上記のサンプル・ジョブ・ログでは、メッセージ MSG100F において、 PGMC は 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。

プログラムまたはプロシージャーの送信または受信

送信側または受信側が 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。

追加のメッセージ・フィルター

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に、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。

- ジョブが終了し、ジョブ・ログがまだ印刷されていない場合には、次のようなスプール・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。
- ジョブ (バッチ・ジョブまたは対話式ジョブ) がまだ活動状態にあるか、またはジョブ待ち行列にあってまだ開始されていない場合には、ジョブ・ログ表示 (DSPJOBLOG) コマンドを使用します。たとえば、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。

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。

ジョブ・ログ表示 (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。たとえば、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すで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、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。

ジョブ・ログの生成の抑止

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・ログに関する考慮事項

ジョブ・ログの使用については以下の事項を考慮に入れてください。

- 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。操作援助機能の終結処置機能の詳細については、iSeries Information Center の『**システム管理**』カテゴリーにある『*iSeries スタートアップ・ガイド*』トピックを参照してください。
- 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。

対話式ジョブのジョブ・ログに関する考慮事項

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、CHGJOB コマンドまたは 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。たとえば、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。

- CL プログラム作成 (CRTCLPGM) コマンド、CL モジュールの作成 (CRTCLMOD) コマンド、またはバインド CL プログラムの作成 (CRTBNDCL) コマンドで LOG(*YES) を指定した場合。
- CL プログラム作成 (CRTCLPGM) コマンド、CL モジュールの作成 (CRTCLMOD) コマンド、またはバインド CL プログラムの作成 (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・ジョブのジョブ・ログに関する考慮事項

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したい場合には、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 BCHJOB コマンドまたは 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、309 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 *常に* 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。

- CL プログラム作成 (CRTCLPGM)
- CL モジュールの作成 (CRTCLMOD)
- バインド CL プログラムの作成 (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。

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 (QHST)。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高いレベルのトレース情報が含まれます。このログのメッセージ待ち行列は QHST です。

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、

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 ～ Z または 0 ～ 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。

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 ～ 13 桁目です。最後のメッセージの位置は、14～26 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。

活動記録ログ・レコードを処理するプログラムを作成することもできます。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。たとえば、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 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。たとえば、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 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いっぱいになるか、または 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バージョンの物理ファイルに書き込みます。現在バージョンを最新のものにしたい場合には、DSPLOG コマンドに架空のメッセージ識別コード (たとえば ###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、これによってログ・バージョン物理ファイルは最新のものとなります。

ログ表示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) コマンドを使っ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

システム上のログに送られたメッセージを保管するためにデータベース・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、たと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	文字	26	11 ～ 36
変換された日付と時刻[1]	文字	13	37 ～ 49
メッセージ識別コード	文字	7	50 ～ 56
メッセージ・ファイル名	文字	10	57 ～ 66
ライブラリー名	文字	10	67 ～ 76

内容	タイプ	長さ	レコード内の桁
メッセージ・タイプ[2]	文字	2	77 ～ 78
重大度コード	文字	2	79 ～ 80
送信側のプログラム名[3]	文字	12	81 ～ 92
送信側のプログラム命令番号[4]	文字	4	93 ～ 96
受信側のプログラム名[3]	文字	10	97 ～ 106
受信側のプログラム命令番号[4]	文字	4	107 ～ 110
メッセージ・テキストの長さ	2 進数	2	111 ～ 112
メッセージ・データの長さ	2 進数	2	113 ～ 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 ～ 118
送信側のユーザー・プロファイル	文字	10	119 ～ 128
未使用	文字	14	129 ～ 142

:

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	文字	可変[1]

内容	タイプ	長さ
メッセージ・データ	文字	可変[2]
:		
[1] この長さは最初のレコード (桁 111 ～ 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 ～ 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 バージョンに入れられます。

メッセージのメッセージ・データに関する説明については、 218 ページの『置換変数の定義』を参照してください。

QHST ファイルの処理

HLL 言語を使用して QHST ファイルを処理する場合には、同じメッセージでもそれが出されるごとにその長さが変わるという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 のジョブの開始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 BCHJOB コマンドまたは 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 (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。たとえば、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。
  - メッセージは常に 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、たと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。

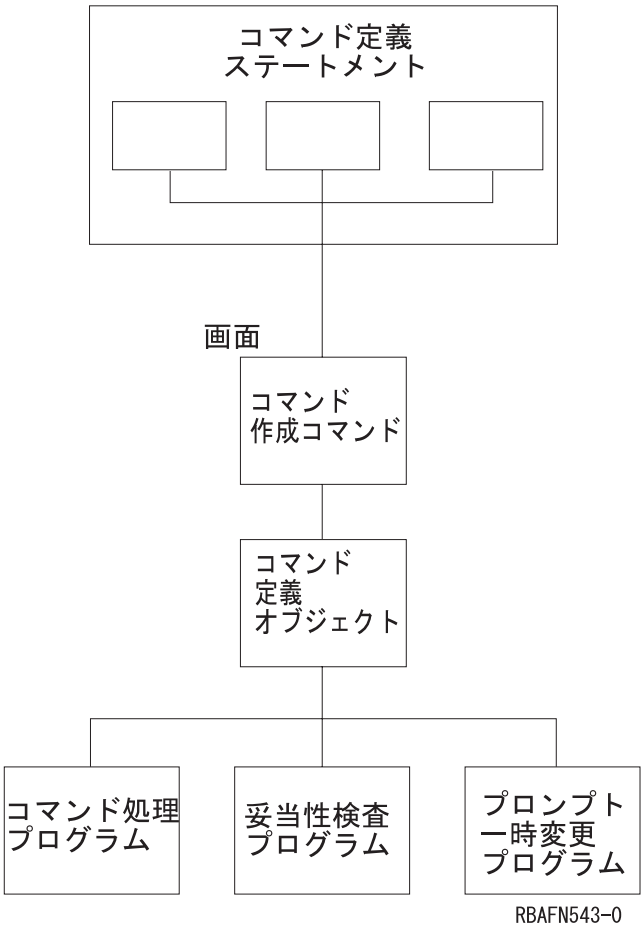
第 9 章 コマンドの定義

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。この章では、ユーザー独自のコマンドの定義と作成の方法について説明しています。 CL コマンドおよびキーワードの省略形については、 473 ページの『付録 D. CL コマンドおよびキーワードの省略形』を参照してください。

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。独自のコマンドに関するオンライン・ヘルプを作成する方法や、そのヘルプの HTML ページを作成する方法については、 399 ページの『第 10 章 コマンドの資料作成』を参照してください。

コマンド定義方法の概要

以下の図は、コマンドを作成する手順を示しています。また図の後に、各手順についての説明文があります。



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。

手順の説明

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はワークステーション・ユーザーが入力する時点でのプロンプト、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。

コマンド定義のステートメントは、CRTCMD コマンドに対する入力としてサポートされるファイルであればどのようなファイルにでも入れることができます。 表 9 には、コマンドの定義に使用するステートメントが記載されています。

表 9. 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	説明
コマンド	CMD	コマンド名のプロンプト・テキストが必要な場合に、そのプロンプト・テキストを指定する。
パラメーター	PARM	コマンドのパラメーターを定義する。
要素	ELEM	パラメーター値として使用されるリストの要素を定義する。
修飾子	QUAL	パラメーターとして使用される名前の修飾子を定義する。
依存関係	DEP	1 つのパラメーターまたはパラメーター間の依存関係を定義する。
プロンプト制御	PMTCTL	パラメーターのプロンプトを表示する条件を定義する。

コマンド作成 (CRTCMD) コマンド

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。 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。

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。

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って、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 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。 QCMDEXC プログラムの詳細については、第 6 章を参照してください。
- CL モジュールまたは 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 (390 ページの『妥当性検査プログラムの作成方法』の項を参照) 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制御言語プログラムまたはプロシージャー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。 妥当性検査プログラムを記述するための詳細については、 390 ページの『妥当性検査プログラムの作成方法』を参照してください。

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。 たとえば、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。 詳細については、 369 ページの『キー・パラ

メーターとプロンプト一時変更プログラムの使用法』を参照してください。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。

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 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。 たとえば、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。

コマンド出口プログラムおよび独立 ASP

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 (たとえば、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。

ユーザー定義のコマンドに必要な権限

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。

簡略コマンドの作成例

たとえば、システム・オペレーターがシステムを開始するプログラムを呼び出すコマンドを作成したい場合には、以下のようにします。 (この例では、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 CL プログラムの作成 (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
   ```

上記の例において、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。これでシステム・オペレーターは、S を入力してコマンドを呼び出すことも、あるいは 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もできます。

コマンドの定義方法

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。詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。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・ファイルにステートメントを入力することについての詳細は、「適用業務開発ツールセット AS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照」 を参照してください。

CMD ステートメントの使用法

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 つだけ含めなければなりません。

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。

パラメーターの定義

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。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。

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 ～ 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)。

パラメーターのキーワードの命名

パラメーターには、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。たとえば、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。

パラメーター・タイプ

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、文字ストリングでアポストロフィ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、アポストロフィ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〜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 〜 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。 その値は左寄せにされ、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う回します。
- 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、文字ストリングでアポストロフィ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、アポストロフィ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。たとえば、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 〜 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 〜 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。指定する文字は 0 ～ 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。値をアポストロフィで囲む場合には、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。たとえば、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(*UNIT) 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。 たとえば、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、それに含まれているのが数字、正符号か負符号、小数点だけであった場合には、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。その値は左寄せにされ、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 (たとえば、&PARM)。変数名が OS/400 で使用されている命名規則に従っていない場合は、その名前をアポストロフィ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。 たとえば、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。

パラメーター値の長さ

以下の各タイプのパラメーター値に対しては、長さ (LEN パラメーター) を指定できます。ただし、日付および時刻のパラメーター・タイプの場合には、日付は常に 7 文字であり、時刻は常に 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	20K	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、1〜12 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。プロンプターは、512 文字を超える可能性のあるパラメーターについて、512 文字入力フィールドを表示します。

デフォルト値

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 パラメーターで定義できます。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
*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

パラメーター定義の例

以下の例では、受注アプリケーションを呼び出す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。

データ・タイプおよびパラメーターに関する制約事項

以下の図は、パラメーター・タイプごとに有効なパラメーターの組み合わせを示したものです。 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

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL
*CNAME	X		X	X	X	X	X	X	3	1
*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					

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	
*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。

次の図は、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。たとえば、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。これは表の後に示す、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										
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										

	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY
SPCVAL										
SNGVAL		7								
MIN		6								
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					
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			

	CHOICE	CHOICEPGM	PMTCTL	PMTCTLPGM	PROMPT	INLPMTLEN
MAX						
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。

パラメーターに対するリストの定義

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')

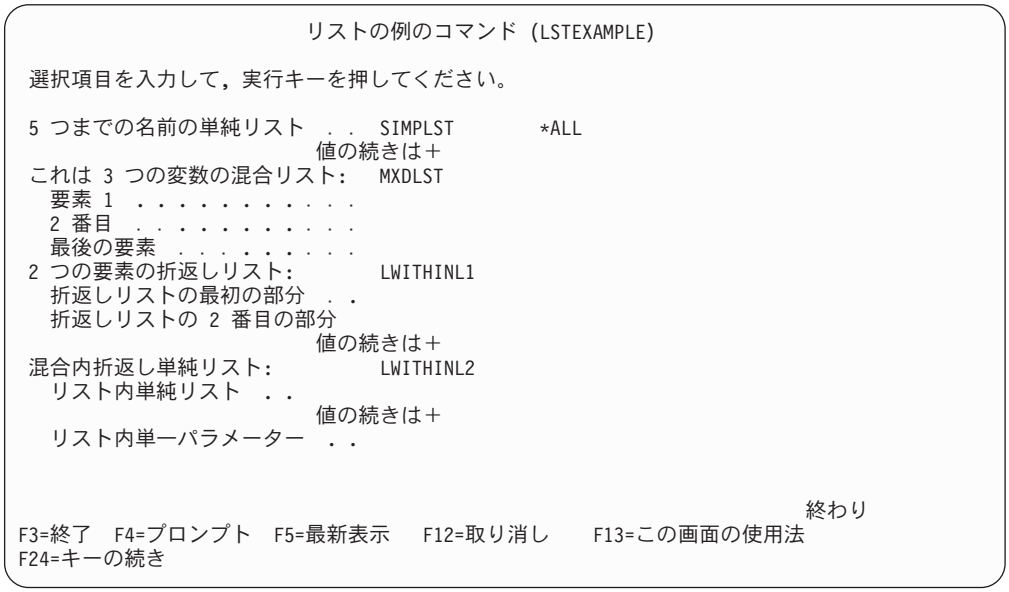

   /* THE FOLLOWING PARAMETER IS A SIMPLE LIST. IT WILL ACCEPT UP TO   */
   /* 5 NAMES.                                                         */
             PARM       KWD(SIMPLST) TYPE(*NAME) LEN(10) DFT(*ALL) +
                         SPCVAL((*ALL)) MAX(5) PROMPT('5 つまでの名前の +
                         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 つの要素の折返しリスト')
LWLSPECA:    ELEM         TYPE(*CHAR) LEN(10) PROMPT('折返しリストの +
                            最初の部分')
             ELEM         TYPE(*DEC) LEN(5 0) PROMPT('折返しリストの +
                            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
                         within a list')
```

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。



単純リストの定義

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。たとえば、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。

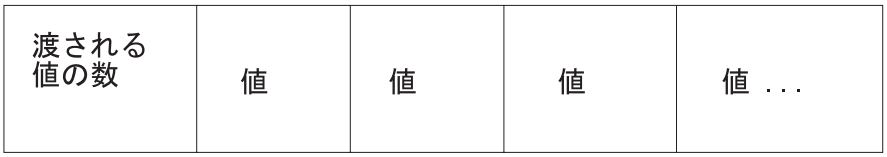
```
PARM     KWD(USER) TYPE(*NAME) LEN(10) MIN(0) MAX(5) +
         SPCVAL(*ALL) DFT(*ALL)
```

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、HLL、REXX のどれを使用しているかによって異なります。以下の項では、前の例で使用した要素を CL または HLL を使用して渡す方法について説明します。 REXX を使用した場合との違いについては、352 ページの『REXX 使用時の単純リスト』の項を参照してください。

CL または HLL 使用時の単純リスト

CL または HLL を使ってコマンドを実行すると、単純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。



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 (335 ページの『パラメーターの定義』の項を参照) 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。たとえば、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。

351 ページの図 13 は、2 進数組み込み関数を使用して単純リストを処理する CL プロシージャーの例を示しています。

```
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 13. 単純リストの例

これと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。

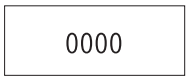
単純リストの場合、リストの代わりに *ALL または *NONE などの単一の値をコマンドに指定できます。単一の値は、個々の値として渡されます。同様に、パラメーターに対して値をまったく指定しなかった場合、デフォルト値が定義されていれば、そのデフォルト値がリストの唯一の値として渡されます。たとえば、USER パラメーターでデフォルト値 *ALL を使用した場合には、以下のように渡されます。

0001	*ALL

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

REXX 使用時の単純リスト

同じコマンドを実行した場合、単純リストの要素は、以下の形式の引き数リストとして REXX プロシージャーに渡されます。

```
. . . USER(value1 value2  . . . valueN) . . .
```

ここで valueN は単純リスト内の最後の値です。

たとえば、2 つのユーザー名 (BJONES および TBROWN) を USER パラメーターに指定している場合には、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。

図 14 の REXX の例は、 351 ページの図 13 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 14. 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。たとえば、USER パラメーターに対してデフォルト値 *ALL が使用されている場合、以下のものが渡されます。

```
. . . 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() . . .
```

REXX プロシージャーの詳細については、「REXX/400 Programmer's Guide」 
および 「REXX/400 Reference」  を参照してください。

混合リストの定義

混合リストは、個別に定義された一連の値を受け入れるもので、それぞれの値は、通常意味もタイプも異なり、リスト内で一定の位置を占めています。たとえば、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、ALWUNPRT、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。ディスプレイ装置ユーザーは、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、HLL、REXX のどれを使用しているかによって異なります。以下の項では、前の例で使用した要素を CL または HLL を使用して渡す方法について説明します。 REXX を使用した場合の違いについては、 355 ページの『REXX 使用時の混合リスト』の項を参照してください。

CL または HLL 使用時の混合リスト

CL または HLL を使用するコマンドを実行する場合、混合リスト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。

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 (335 ページの『パラメーターの定義』) と同様に、タイプに応じて行われます。たとえば、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。たとえば、ディスプレイ装置ユーザーがパラメーターの単一値として *ANY を入力した場合には、以下のものが渡されます。

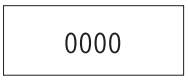
RBAFN515-0

*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 (第 2 章を参照)。

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

REXX 使用時の混合リスト

REXX を使用する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。

```
. . . CMPVAL(value1 value2  . . . valueN) . . .
```

ここで、valueN は混合リストの最後の値です。

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。たとえば前の例で、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。たとえば、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() . . .
```

リスト内リストの定義

リスト内リストとは、次のようなリストのことです。

- 1 つのパラメーターに対し複数回指定できるリスト (単純リストまたは混合リスト)
- 混合リスト内の 1 つの値として指定できるリスト

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、たとえば次のようになります。

```
PARM2((NAME1 NAME2 NAME3) 123.0)
```

リスト内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、HLL、REXX のどれを使用しているかによって異なります。以下の項では、 CL または HLL を使用して要素を渡す方法について説明します。 REXX を使用した場合との違いについては、 359 ページの『REXX 使用時のリスト内リスト』の項を参照してください。

CL または HLL 使用時のリスト内リスト

CL または HLL を使用するコマンドを実行すると、リスト内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。

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))
```

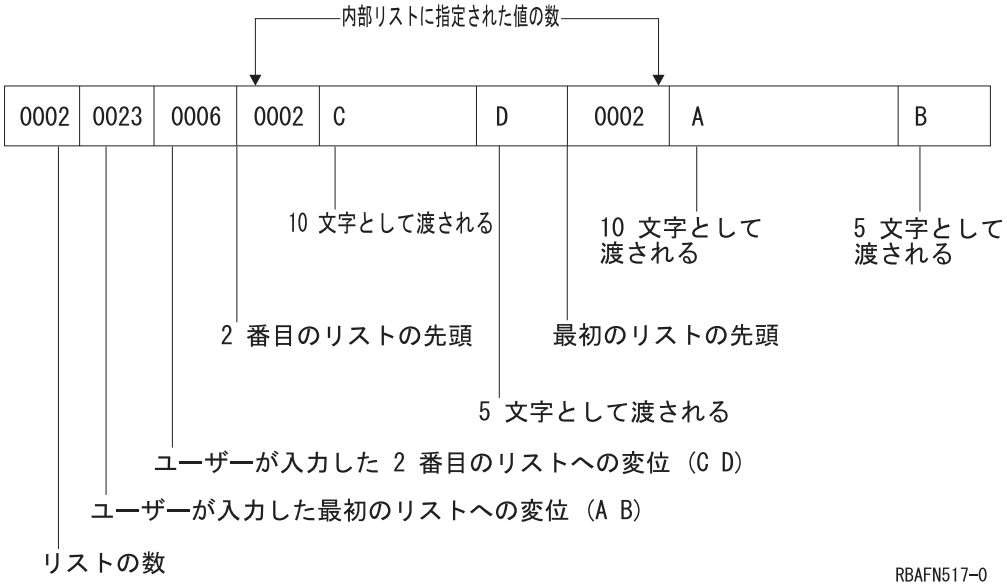
以下のものがコマンド処理プログラムに渡されます。

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))
```

この場合には、以下の形式で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。

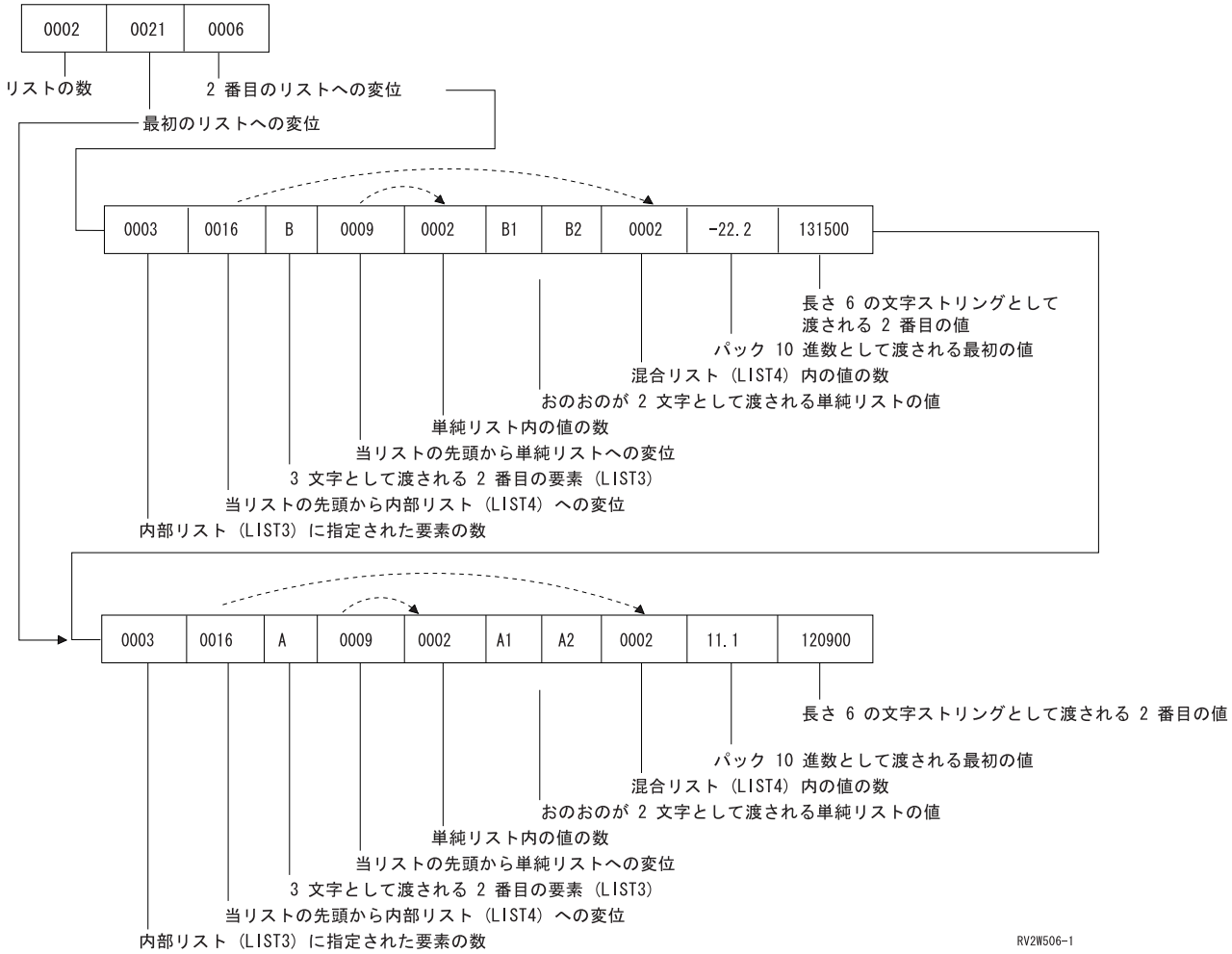
以下の例は、リスト内リストのさらに複雑な使用例です。パラメーターの定義は以下のとおりです。

```
        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 PARM1 パラメーターを次のように入力したとします。

```
PARM1(((11.1  120900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

この場合には、以下の形式でコマンド処理プログラムに渡されます。



REXX 使用時のリスト内リスト

REXX を使用するコマンドを実行した場合、リスト内リストがその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。

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)))
```

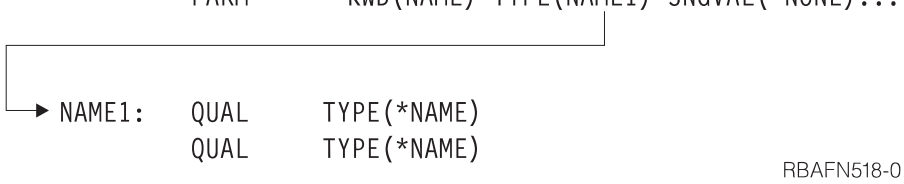
以下のものがコマンド処理プログラムに渡されます。

```
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

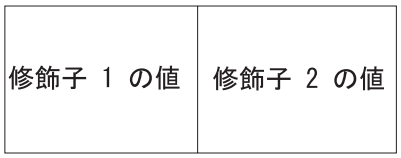
QUAL ステートメントに指定できるパラメーターの多くは、PARM ステートメントの場合と同じです (335 ページの『パラメーターの定義』を参照)。ただし、TYPE パラメーターには以下の値だけを指定できます。

- *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、HLL、REXX のどれを使用するかによって異なります。以下の項では、CL または HLL を使用して修飾名を渡す方法について説明します。 REXX を使用した場合との違いについては、362 ページの『REXX 使用時の修飾名』の項を参照してください。

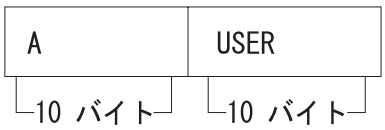
CL または HLL 使用時の修飾名

CL または HLL を使用している場合、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。



RBAFN519-0

たとえば、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 (335 ページの『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あてはまります。

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。たとえば、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

以下の例に示すように、修飾名はサブストリング組み込み関数を使用して、CL プログラムで処理できます。

サブストリング組み込み関数 (%SUBSTRING または %SST) は、修飾名を 2 つの値に分けるために使用します。

```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
DCL &OBJ TYPE(*CHAR) LEN(10)
DCL &LIB TYPE(*CHAR) LEN(10)
CHGVAR &OBJ %SUBSTRING(&QLFDNAM 1 10) /* First 10 */
CHGVAR &LIB %SST(&QLFDNAM 11 10) /* Second 10 */
```

```
 .
 .
 .
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。 (OBJ(&LIB/&OBJ) など)。

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

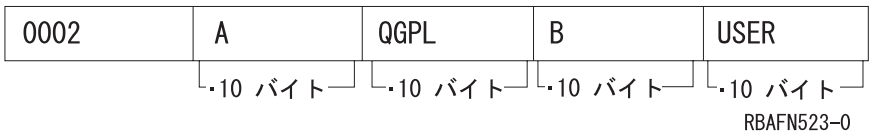
たとえば、以下の例の NAME パラメーターの PARM ステートメントに、次のように MAX(3) を指定したとします。

```
        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。

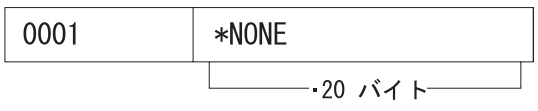
```
NAME(QGPL/A USER/B)
```

この場合、パラメーターは以下の形式で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

REXX 使用時の修飾名

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。

たとえば、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 (335 ページの『パラメーターの定義』を参照)。

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)
```

従属関係の定義

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)
```

選択可能な項目と指定可能な値

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、または 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、たと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 2000 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 2000 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 2000 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 ～ 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。

プロンプト制御の使用法

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。

条件プロンプト

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、値の前に長さ値は付きません。制御パラメーターが PASSATR(*YES) の場合には、属性バイトは含まれません。
- 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。

```
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
```

追加のパラメーター

使用頻度の低いパラメーターについてのプロンプトは、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、表示されないようにすることができます。このためには、そのパラメーターに対応する 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。

キー・パラメーターとプロンプト一時変更プログラムの使用法

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。

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。キー・パラメーターは、たと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。

  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。

  たとえば、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。

  たとえば、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。追加のパラメーターの詳細については、368 ページの『追加のパラメーター』の項を参照してください。

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。

プロンプト一時変更プログラムの使用手順

プロンプト一時変更プログラムを使用する場合には、以下のことを行ってください。

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。 KEYPARM パラメーターの詳細については、370 ページの『キー・パラメーターの識別』の項を参照してください。
2. プロンプト一時変更プログラムを作成します。プロンプト一時変更プロンプト作成の詳細については、370 ページの『プロンプト一時変更プログラムの作成方法』の項を参照してください。
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。プロンプト一時変更プログラムを使

用するコマンドの作成または変更の詳細については、 373 ページの『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方法』の項を参照してください。

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、以下のことを行ってください。

- 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。

プロンプト一時変更プログラムの作成方法

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。

プロンプト一時変更プログラムの CL ソースの例については、 374 ページの『プロンプト一時変更プログラムの CL サンプル』の項を参照してください。

プロンプト一時変更プログラムに渡すパラメーター: プロンプト一時変更プログラムに対し以下のパラメーターを渡します。

- 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。

たとえば、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。選択プロンプト文字の使い方については、195 ページの『CL コマンドの選択プロンプト』の項を参照してください。

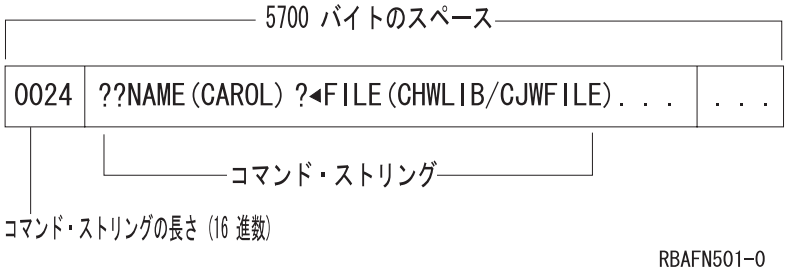
  選択プロンプトを使用する場合、以下のことを行ってください。

  - 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、ユーザーは選択プロンプト文字 ?? を指定しなければなりません。
  - 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。

  たとえば、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。たとえば、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。たとえば、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。テキストの検索に使用するフィールドの長さの定義方法については、 374 ページの『プロンプト一時変更プログラムの CL サンプル』の項を参照してください。

  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。
- 内部に含まれるアポストロフィ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。

プロンプト一時変更プログラム内でのエラー対策: 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、プロンプト一時変更プログラム自体で以下のことを行わなければなりません。

- 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。
- 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。
- エスケープ・メッセージ CPF0011 を送る。

たとえば、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' || &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。

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方法

ユーザーが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) コマンドの使用時点で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。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。

コマンド定義ソースにキー・パラメーターを指定していない場合に、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ことがあります。このような場合には、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。

プロンプト一時変更プログラムの CL サンプル

以下の例は、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。このコマンドにより、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。

こ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。

コマンド・ソースの例

```
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の例

```
   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 wish 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 wish 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
```

コマンドの作成方法

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。(QCMDEXC および QCAPCMP API の詳細については、第 6 章を参照。)
- 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、 340 ページの『パラメーター定義の例』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')
```

生成されるコマンドは以下のとおりです。

```
ORDENTRY  OETYPE(value)
```

ここで、value には、DAILY、WEEKLY、MONTHLY のいずれかを指定できます。

コマンドが作成されると、以下の操作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。

コマンド定義のソース・リスト

コマンドを作成する場合には、ソース・リストが生成されます。次に、ソース・リストの例を示します。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。

```
5722SS1 V5R3M0  040430 1       コマンド定義          DSTPRODLB/ORDENTRY    11/20/04 14:53:32 2 ページ 1 3

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     11/20/04  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
図形ユーザー・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  . . . . . :  Calls order entry application
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
ヘルプ識別コード  . . . . . :  *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 AS/400 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')                                    11/20/04
7
   200-              PARM      KWD(OETYPE) TYPE(*CHAR) RSTD(*YES) +                             11/20/04
   300                           VALUES(DAILY WEEKLY MONTHLY) MIN(1) +                          11/20/04
   400                           PROMPT('Type of order entry:')                                 11/20/04
                                      * * * * *   ソースの終わり   * * * * *

5722SS1 V5R3M0  040430               コマンド定義              DSTPRODLB/ORDENTRY     11/20/04 14:54:32        ページ    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   * * * * *

5722SS1 V5R3M0  040430               コマンド定義              DSTPRODLB/ORDENTRY     11/20/04 14:54:32        ページ    3
                                       最終メッセージ

メッセージ              順序
識別コード              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 OS/400 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。

前書き:

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。たとえば、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。

コマンド定義エラーの詳細については、 380 ページの『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』の項を参照してください。

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れば、ラベルの相互参照リスト (ラベル表) が印刷されます。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。

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

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・ステートメントで引き起こしたエラーを訂正するには、 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 については、「適用業務開発ツールセット AS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照」 を参照してください。

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。このようなメッセージについては、 378 ページの『コマンド定義のソース・リスト』の項を参照してください。 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。

コマンド定義の表示

コマンド表示 (DSPCMD) コマンドを用いて、CRTCMD コマンドにパラメーターとして指定されていた値を表示または印刷できます。 DSPCMD コマンドによって、ユーザー作成または IBM 提供のコマンドに対して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、コマンド作成時に妥当性検査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、コマンド作成 (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。これらの API の詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*API (APIs)*』セクションを参照してください。

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の変更の影響

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。たとえば、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。たとえば、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。

コマンド・デフォルト値の変更

コマンド・デフォルト値変更 (CHGCMDDFT) コマンドを使用することによって、コマンドのキーワードのデフォルト値を変更できます。詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーの『*CL*』セクションを参照してください。デフォルト値を新たな値に変更できるのは、キーワードに既存のデフォルト値がある場合だけです。 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。デフォルト値の変更に関する推奨事項は以下のとおりです。

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、システム・ライブラリー・リスト変更 (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。

```
PGM
DLTCMD USRQSYS/SIGNOFF
CRTDUPOBJ OBJ(SIGNOFF) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS)  NEWOBJ(*SAME)
CHGCMDDFT CMD(USRQSYS/SIGNOFF) NEWDFT('LOG(*LIST)')
 .
 .
Repeat the DLTCMD, CRTDUPOBJ and CHGCMDDFT for each
command you want changed
 .
 .
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。詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーの『*API*』セクションにあるコマンド分析プログラムの検索出口プログラムの記述を参照してください。

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 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 の値の一部としてアポストロフィ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。

例 1

コマンド 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)')
```

例 2

コマンド 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)')
```

例 3

以下の指定により、コマンド 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)')
```

例 4

以下の指定により、コマンド 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 キーワードにはアポストロフィが組み込まれているため、これらのアポストロフィを二重にする必要があります。二重にしないと、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'')')
```

例 5

以下の指定により、コマンド CRTLF の 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)))')
```

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。

例 6

ジョブのスプール・ファイルを表示するコマンドを作成します。

```
CRTDUPOBJ OBJ(WRKJOB) FROMLIB(QSYS) +
          TOLIB(MYLIB) NEWOBJ(WRKJOBSPLF)
WRKJOBSPLF  OPTION(*SPLF)
  .
  .
CHGCMDDFT CMD(MYLIB/WRKJOBSPLF) +
          NEWDFT('OPTION(*SPLF)')
```

コマンド処理プログラムまたはコマンド処理プロシージャーの作成方法

コマンド処理プログラム (CPP) としては CL プログラム、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、CALL CL 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。

CL または HLL コマンド処理プログラムの作成方法

図 15 は、コマンド作成 (CRTCMD) コマンド、コマンド定義ステートメント、およびコマンド処理プログラムの相互関係を示しています。

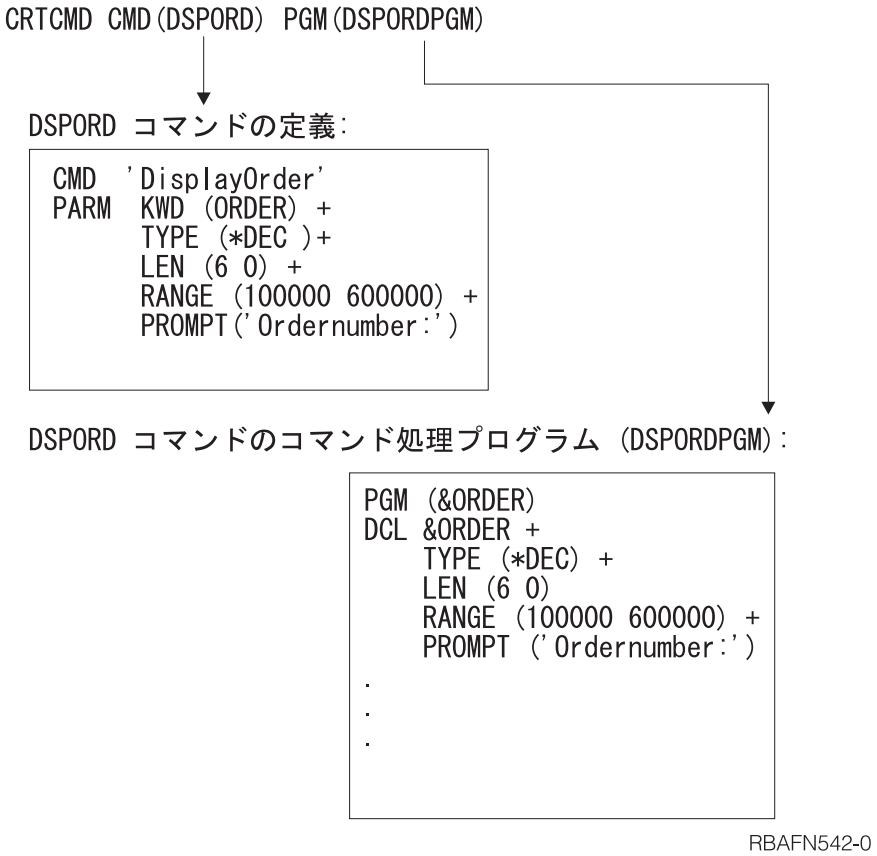


図 15. CL および HLL の場合のコマンドの相互関係

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。(図 15 のパラメーター ORDER の宣言に注意してください。)

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

:

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。

コマンド処理プログラムの例については、 391 ページの『コマンドの定義および作成の例』を参照してください。

REXX コマンド処理プロシージャーの作成方法

図 16 は、REXX を使用する場合のコマンド作成 (CRTCMD) コマンド、コマンド定義ステートメント、およびコマンド処理プロシージャーの相互関係を示しています。

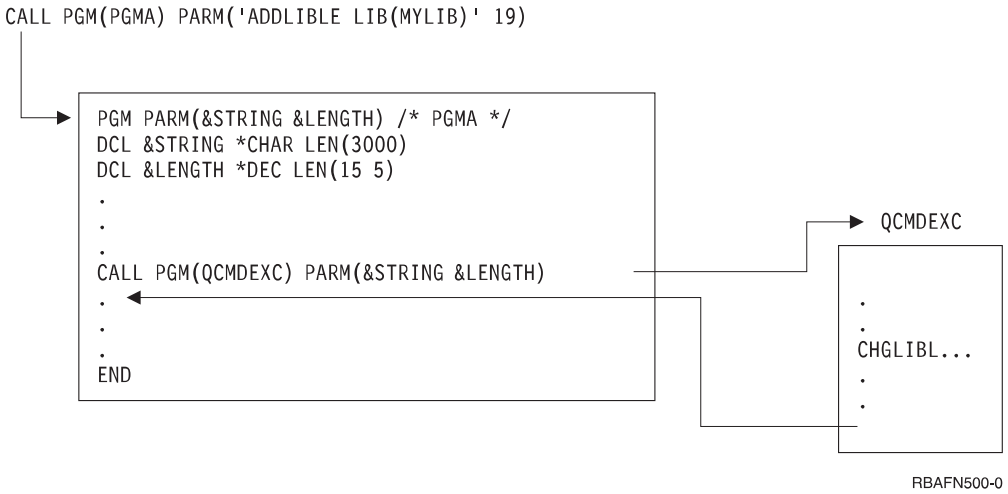


図 16. REXX の場合のコマンドの相互関係

妥当性検査プログラムの作成方法

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場合には、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、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。たとえば、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' || &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
```

コマンドの定義および作成の例

この項では、コマンドの定義および作成の例を示しています。

アプリケーション・プログラムの呼び出し

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 OS/400 により行われます。 ただし、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。

たとえば、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 10〜150 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 OS/400 プログラムの場合には、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
```

CRTCMD コマンドは以下のとおりです。

```
CRTCMD CMD(LBLWRT) PGM(LBLWRT) SRCMBR(LBLWRT)
```

デフォルト値の置き換え

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。たとえば、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し、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。

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
```

CRTCMD コマンドは以下のとおりです。

```
CRTCMD CMD(SAVLIBTAP) PGM(SAVLIBTAP) SRCMBR(SAVLIBTAP)
```

出力待ち行列の表示

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。次に示すコマンド DSPOQ は、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したい場合にディスプレイ装置ユーザーが使用でき、印刷オプションも用意されています。

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。

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
```

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 (たとえば、A〜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 OS/400 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。

IBM 提供のコマンドからのメッセージの再表示

CLROUTQ コマンドは、処理の完了時に完了メッセージ 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。 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。たとえば、ユーザー定義の 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。

たとえば、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')
```

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。

```
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。

簡略コマンドの作成

例 1

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。たとえば、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 CJ コマンドを作成し実行するための 3 つ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: CRTCMD コマンド

```
CRTCMD  CMD(CJ)  PGM(CJ)  SRCMBR(CJ)
```

例 2

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
```

CRTCMD コマンドは以下のとおりです。

```
CRTCMD CMD(DW1) PGM(DW1) SRCMBR(DW1)
```

ファイルおよびソース・メンバーの削除

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コマンドを作成できます。

DFS 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 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。

コマンド処理プログラムは以下のとおりです。

```
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
```

プログラム・オブジェクトの削除

高水準言語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。 コマンド処理プログラムは以下のとおりです。

```
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
```

第 10 章 コマンドの資料作成

329 ページの『第 9 章 コマンドの定義』で説明されているように独自の CL コマンドの定義と作成を選択した場合は、コマンドを説明するオンライン・コマンド・ヘルプの作成も決定できます。この章では、独自のコマンドに関するオンライン・ヘルプを作成する方法や、そのオンライン・コマンド・ヘルプの HTML コマンド資料を作成する方法について説明します。

コマンドとコマンド・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。コマンド・ヘルプについても同じことが言えます。ヘルプを作成することにより、コマンドをユーザーに説明することができます。

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。
- CRTCMD (コマンド作成) 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 (たとえば、コマンド・プロンプトで F1 (ヘルプ) キーを押す) と、 OS/400 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、 CHGCMD (コマンド変更) 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。 OS/400 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。たとえば、STRPAY が HLPID(STRPAY) を指定して作成された場合、OS/400 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。たとえば、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、 OS/400 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。たとえば、コマンド STRPAY の HLPID 値が STRPAY の場合、 OS/400 は、STRPAY/COMMAND/EXAMPLES という名前のヘルプ・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。たとえば、コマンド STRPAY の HLPID 値が STRPAY の場合、 OS/400 は、STRPAY/ERROR/MESSAGES という名前のヘルプ・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・グループで見付からないと、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・モジュールが見付からない場合、診断メッセージは表示されません。

コマンド・ヘルプの作成

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・テキストを実際に作成できます。iSeries のコマンドのオンライン・ヘルプは、UIM (ユーザー・インターフェース・マネージャー) というタグ言語で作成します。 UIM ソースを CRTPNLGRP (パネル・グループの作成) 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。ヘルプ・パネル・グループと UIM パネル・グループ定義言語についての詳細は、 「Application Display Programming」  の付録 A を参照してください。

コマンド・ヘルプ用の UIM ソースの生成

UIM 構文を学ぶ単純な代わりとして、コマンド資料の生成 (GENCMDDOC) コマンドを使用できます。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。このファイルを、オンライン・コマンド・ヘルプのテンプレートにできます。 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。 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。

GENCMDDOC の詳細については、 iSeries Information Center の『CL』トピックにあるコマンド資料を参照してください。

次に示すのは、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 (たとえば、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、 CRTPNLGRP (パネル・グループの作成) 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。 CRTPNLGRP コマンドの使用例を次に示します。

```
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付からなければ、 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。

共通ヘルプの共用

前のセクションでは、単一のコマンドのヘルプ・モジュールを含むパネル・グループを作成する方法について説明しました。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。たとえば、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 タグの使用です。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。

:IMHELP. および :IMPORT. の UIM タグの詳細については、 「Application Display Programming」  の付録 A を参照してください。

ヘルプ・モジュールへのヘルプ・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。

コマンド資料の HTML ソースの生成

コマンドのヘルプ情報の作成に加え、 iSeries システムに接続していないときに表示または印刷できるコマンド資料も作成できます。 OS/400 では、コマンド資料をハイパーテキスト・マークアップ言語 (HTML) ソースとして生成し、インターネット・ブラウザーを使用して表示したり、多くのワード・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。

GENCMDDOC の使用の詳細については、 iSeries Information Center の『CL』トピックにあるコマンド資料を参照してください。

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、iSeries 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、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、DSPF (ファイルの表示) 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。

第 11 章 プログラムのデバッグ

ILE プログラムのデバッグ

デバッグを行うと、プログラム内のエラーの検出、診断、および除去が行えます。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。この章では、ILE ソース・デバッガーの使用法について説明します。

この章では、以下の操作を行う方法について説明します。

- 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 (停止点) および無条件ブレークポイントの設定や除去を行う。
- 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。

詳細については、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

ILE ソース・デバッガー (プログラムまたはサービス・プログラムのデバッグに必要な権限および最適化レベルの効果を含む) は、「ILE 概念」  の『デバッグに関する考慮事項』の章を参照してください。

ILE ソース・デバッガー

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の設定や除去を行う。
- 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箇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。この箇所でデバッグ・コマンドを追加入力できます。

CL モジュールの作成 (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。

デバッグ・コマンド

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。

表 10 にこれらのデバッグ・コマンドをまとめてあります。 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 10. 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 *position* WHEN *expression* を使用してください。

表 10. 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、ストリング、またはテキストを見つける。
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™	現行のデバッグ・セッション中の後続の FIND 要求を、大文字小文字を区別して探索するか、または区別せずに探索するかを指定する。また、実動ファイルの値の更新の変更することもできます。
WATCH	現在活動状態のウォッチの状況のリストを表示する。

プログラム・オブジェクトのデバッグ・セッションの準備

ILE ソース・デバッガーを使用する場合には、その前に CRTCLMOD コマンドか CRTBNDCL コマンドを使用し、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ど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

ルート・ソース・ビューの使用法

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。

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、CRTCLMOD コマンドか 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 モジュールの作成 (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。

リスト・ビューの使用法

リスト・ビューは、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・コード部分と同じです。

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、モジュールを作成する際に CRTCLMOD コマンドか CRTBNDCL コマンドの DBGVIEW パラメーターに *LIST か *ALL オプションを使用してください。

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)
```

ステートメント・ビューの使用法

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 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 CRTCLMOD コマンドか 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)
```

ILE ソース・デバッガーの開始

デバッグ・ビューの作成後に、ユーザーの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、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)
```

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

セッションを開始した後で、さらにプログラム・オブジェクトをデバッグ・セッションに追加できます。

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。 ILE ソース・デバッガー・コマンドのリストについては、 406 ページの表 10を参照してください。「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。

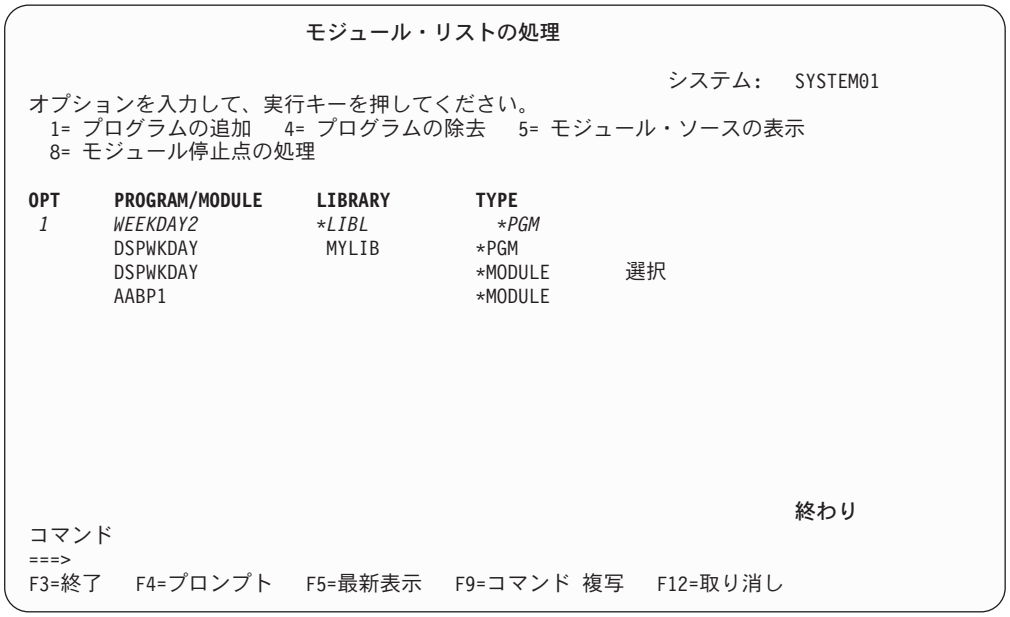


図 17. ILE プログラム・オブジェクトのデバッグ・セッションへの追加: 実行キーを押すと、プログラム WEEKDAY2 がデバッグ・セッションに追加される。

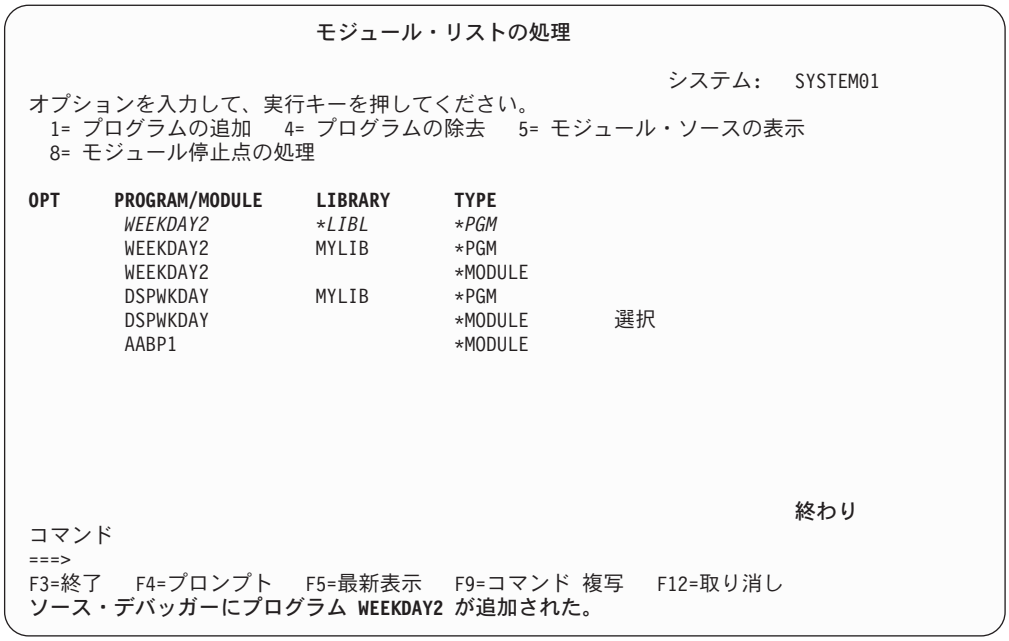


図 18. ILE プログラム・オブジェクトのデバッグ・セッションへの追加: 画面の最下部の情報メッセージは、プログラム WEEKDAY2 がデバッグ・セッションに追加されたことを示している。

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。

OPM プログラムをデバッグ・セッションに追加する場合は、プログラム追加 (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、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 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。

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

セッションを開始した後で、プログラム・オブジェクトをデバッグ・セッションから除去できます。

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。 412 ページの図 19 を参照してください。「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
                           *LIBL        *PGM
 4        WEEKDAY2         MYLIB        *PGM
          WEEKDAY2                       *MODULE
          DSPWKDAY          MYLIB        *PGM
          DSPWKDAY                       *MODULE     選択
          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 複写    F12=取り消し
```

図 19. 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
                           *LIBL        *PGM
          DSPWKDAY         MYLIB        *PGM
          DSPWKDAY                      *MODULE     選択
          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 複写    F12=取り消し
プログラム WEEKDAY2 はソース・デバッガーから除去された。
```

図 20. 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。

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

「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(*LISTING)

「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、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・リストの処理」画面でオプション 5 (モジュール・ソースの表示) を使用すると、「モジュール・ソースの表示」画面に表示されるモジュール・オブジェクトを変更できます。「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。 414 ページの図 21 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

PROGRAM:   DSPWKDAY       LIBRARY:   MYLIB          MODULE:   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= 終了 プログラム    F6= 停止点の追加／消去    F10= ステップ    F11= 変数の表示
F12=再開              F17=ウォッチ変数       F18=ウォッチの処理      F24=キーの続き
```

図 21. モジュール・ソースの表示

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・オブジェクトのビューの変更

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・オブジェクトの複数のビューを使用できます。以下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

「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・オブジェクトのビューを、「ビューの選択」画面を用いて変更できます。「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。 415 ページの図 22 に「ビューの選択」画面が記載されています。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。

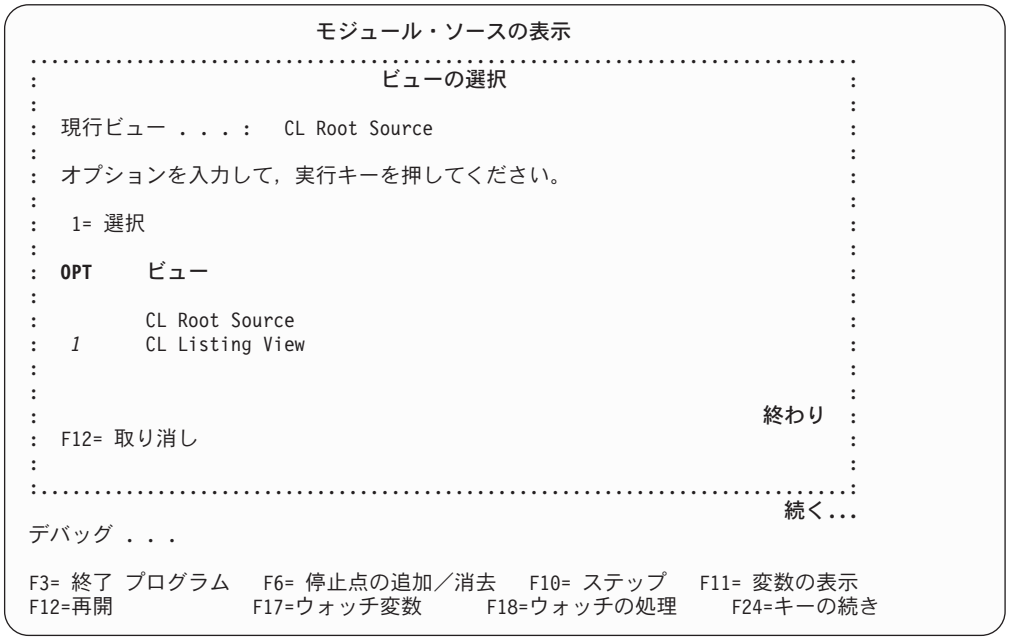


図 22. モジュール・オブジェクトのビューの変更

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 (停止点) の設定と除去

ブレークポイントを使用して、プログラム・オブジェクトを実行中に特定の箇所で停止することができます。**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 (たとえば 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と除去

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したいモジュール・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したいモジュール・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、停止させたい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と除去

条件付き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・ブレークポイントの処理」画面
- 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)

「モジュール・ブレークポイントの処理」画面の使用法:

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。

条件付きブレークポイントの設定と除去を行う 1 つの方法は、「モジュール・ブレークポイントの処理」画面を使用することです。「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。 418 ページの図 23 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実行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

たとえば、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、418 ページの図 23 に記載されているとおり、以下のように入力して実行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、除去したい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、実行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 23. 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・コマンドの使用法: 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したいモジュール・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。 NLSS に関する詳細は、 419 ページの『各国語分類順序 (NLSS)』を参照してください。

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したいモジュール・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、停止させたいステートメントの番号です。

各国語分類順序 (NLSS): 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)。

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式の有効な組み合わせについては、420 ページの表 11 を参照してください。

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、CRTCLMOD か 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 11. 数値以外の条件付きブレークポイントの式

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]

:

1 文字リテラルの形式は ‘abc’ です。

2 16 進数リテラルの形式は X‘16 進数字’ です。

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。

条件付きブレークポイントの例:

```
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。最初の引き数はストリングの識別コード、2 番目の引き数は開始位置、および 3 番目の引き数は 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き数を区切ります。

すべてのブレークポイントの除去

CLEAR PGM デバッグ・コマンドを使用すると、「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) を除去できます。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。

プログラム・オブジェクトのステップスルー

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て「モジュール・ソースの表示」画面に戻ることができます。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。

以下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スルーできます。

- 「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)
- STEP デバッグ・コマンド

「モジュール・ソースの表示」画面での F10 または F22 の使用法

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。 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。

STEP デバッグ・コマンドの使用法

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。 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。 たとえば、デバッグ・コマンド行に次のように入力したとします。

```
STEP 5
```

この場合、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。

ステップオーバーとステップイントゥ

デバッグ・セッション中に他のプログラム・オブジェクトに対する CALL ステートメントが検出された場合は、次のどちらかを行えます。

- 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。
- 呼び出されるプログラム・オブジェクトにステップイントゥする。

呼び出されるプログラム・オブジェクトの**ステップオーバー**を選択すると、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。ステップオーバーはデフォルトのステップ・モードです。

呼び出されるプログラム・オブジェクトの**ステップイントゥ**を選択すると、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。次のステップ (実行中のプログラム・オブジェクトが停止するステップ) が、呼び出されたプログラム・オブジェクト中で失敗すると、呼び出されたプログラム・オブジェクトはこの箇所で停止します。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。

プログラム・オブジェクトのステップオーバー

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

F10 (ステップ) の使用法

「モジュール・ソースの表示」画面の F10 (ステップ) を使用すると、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。次に実行される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、F10 を押すと、呼び出されたプログラム・オブジェクトが実行が完了した後で呼び出されたプログラム・オブジェクトは再停止します。

STEP OVER デバッグ・コマンドの使用法

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
```

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。 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、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。

プログラム・オブジェクトのステップイントゥ

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

F22 (ステップ・イン) の使用法

「モジュール・ソースの表示」画面の F22 (ステップ・イン) を使用すると、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。次に実行される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、F22 (ステップ・イン) を押すと、呼び出されたプログラム・オブジェクト中の最初のステートメントが実行されます。それから、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。

注: 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」画面に表示されるようにするには、関連するデバッグ・データがそのオブジェクトに含まれている必要があります。

STEP INTO デバッグ・コマンドの使用法

別の方法として、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
```

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。 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。たとえば、デバッグ・コマンド行に次のように入力したとします。

```
STEP 5 INTO
```

この場合、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

次のものを使用すると、変数の値を表示できます。

- 「モジュール・ソースの表示」画面の F11 (変数の表示)
- EVAL デバッグ・コマンド

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。

F11 (変数の表示) の使用法

F11 (変数の表示) を使用して変数を表示する場合は、表示したい変数の上にカーソルを置き、F11 を押してください。「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

PROGRAM:   DSPWKDAY       LIBRARY:   MYLIB          MODULE:   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= 終了 プログラム   F6= 停止点の追加／消去   F10= ステップ   F11= 変数の表示
F12=再開              F17=ウォッチ変数       F18=ウォッチの処理      F24=キーの続き
&DAYOFWK = 3.
```

図 24. F11 (変数の表示) の使用による変数の表示： EVAL デバッグ・コマンドの使用法

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・コマンド行に次のように入力してください。

```
EVAL 変数名
```

変数名 は、表示したい変数の名前です。 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。

たとえば、図 24 のように、モジュール・オブジェクトの行 7 にある変数 *&DAYOFWK* の値を表示する場合は、次のように入力してください。

```
EVAL &DAYOFWK
```

「モジュール・ソースの表示」画面のメッセージ行には、 424 ページの図 24 のように &DAYOFWK = 3. と表示されます。

ロジック変数の表示例

```
CL 宣言:                DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')

デバッグ・コマンド:      EVAL &LGL1

結果:                   &LGL1 = '1'
```

文字変数の表示例

```
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。最初の引き数はストリングの識別コード、2 番目の引き数は開始位置、および 3 番目の引き数は 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き数を区切ります。

10 進変数の表示例

```
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
```

16 進値での変数の表示

EVAL デバッグ・コマンドを使用して変数の値を 16 進形式で表示することができます。値を 16 進形式で表示する場合は、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: x  バイト数
```

変数名 は、16 進形式で表示したい変数の名前です。 'x' は変数が 16 進形式で表示されるように指定し、バイト数 は表示されるバイト数を示します。 'x' の後に長さを指定しないと、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

EVAL コマンドと代入演算子 (=) を使用して、変数の値を変更することができます。

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ます。

```
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。たとえば *&COUNTER*; の値を 3.0 に変更するには、次のように入力します。

```
EVAL &COUNTER = 3.0
```

「モジュール・ソースの表示」画面のメッセージ行に次のように表示されます。

```
&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 データ<si>’ の形式の図形リテラル
- X'16 進数字' 形式の 16 進リテラル

ロジック変数の変更例

```
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'
```

文字変数の変更例

```
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。最初の引き数はストリングの識別コード、2 番目の引き数は開始位置、および 3 番目の引き数は 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き数を区切ります。

10 進変数の変更例

```
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
```

変数の属性の表示例

属性 (ATTR) デバッグ・コマンドを使用すると、変数の属性を表示できます。 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 (バイト単位) とタイプです。

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
```

名前の変数、式、またはコマンドとの等価

EQUATE デバッグ・コマンドを使用して、省略した名前を変数、式、またはデバッグ・コマンドと等しくすることができます。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場合は、デバッグ・コマンド行に次のように入力します。

```
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。

たとえば、*DC* という省略名 (*&COUNTER* という変数の内容を表示する) を定義する場合には、次のように入力します。

```
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・コマンドを用いてデバッグ・セッション用に定義されている名前を表示する場合は、デバッグ・コマンド行に次のように入力してください。

```
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。

ILE CL 用のソース・デバッグ各国語サポート

ILE CL 用のソース・デバッグ各国語サポートで処理している場合には、以下の条件が当てはまります。

- 「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るときに、ソース・デバッガーは引用符付きリテラル (たとえば 'abc') での 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、以下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・デバッガーはさまざまな文字 (#、@、$) 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。

デバッグに関する各国語サポートの制約事項についての詳細は、「ILE 概念」  の『第 10 章 デバッグに関する考慮事項』を参照してください。

デバッグ時の COPY、SAVE、RESTORE、CRTDUPOBJ、および CHKOBJITG の使用

特定の CL (制御言語) コマンドを使用してライブラリーやプログラムを指定する場合、ブレークポイントやステップがデバッグ中にプログラムから**一時的に除去されます**。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。

OPM プログラムのデバッグ

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。ユーザーはこれによってプログラムを特殊なテスト環境で実行させ、その環境下でのプログラムの処理過程を綿密に観察し、制御することができます。ユーザーはこの章で述べる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。テスト機能により次の処理が可能です。

- 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。

デバッグ・モード

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。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: iSeries 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 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 iSeries 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。 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、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 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)
```

ILE ソース・デバッガーを使用すると、OPM プログラムのデバッグを行えます。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)
```

ILE ソースのデバッグの詳細については、 405 ページの『ILE プログラムのデバッグ』を参照してください。

デバッグ・モードへのプログラムの追加

任意のプログラムをデバッグ・モードで実行できますが、デバッグを行うにはまず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。プログラムをデバッグ・モードにするには、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、STRDBG または 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。たとえば、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) コマンドまたは 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 (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、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 (ADDPGM コマンド)。

実動 (プロダクション) ライブラリーのデータベース・ファイルの更新の防止

デバッグ・モードになっている間でも、実動ライブラリーのファイルを使用することができます。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、STRDBG コマンドで UPDPROD(*NO) を明示的に指定するか、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

デバッグ表示 (DSPDBG) コマンドを使用して、呼び出しスタックを表示することができます。呼び出しスタックは次のことを示します。

- 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。呼び出しスタックについて詳しくは、第 3 章を参照してください。

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。

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。次のど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。

RPG/400® の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。

非監視メッセージの処理

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、

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、システムはデフォルトの処置をとりプログラムを停止せずにユーザーに制御権を渡します。システムは、非監視メッセージ画面に以下の情報を表示します。

- 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。







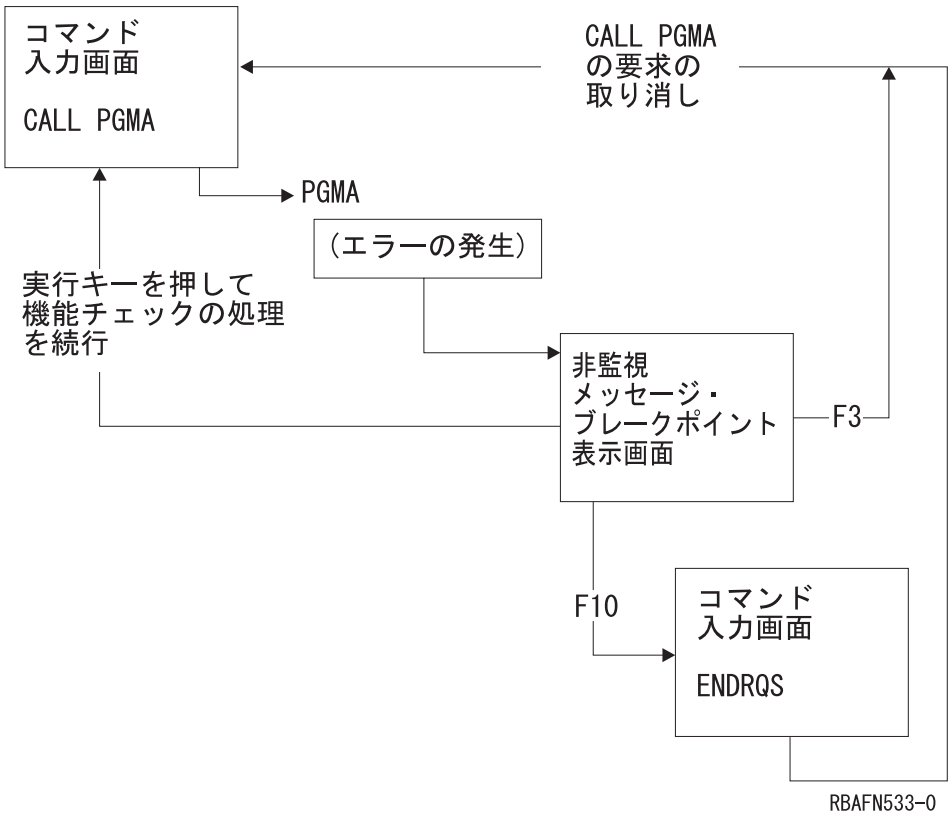


 続行するためには、実行キーを押してください。

 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。

次の図は ENDRQS コマンドの機能を示しています。



ENDRQS コマンドが入力されると、プログラム呼び出しは打ち切られます。 (この図では、PGMA のプログラム呼び出しが打ち切られます。)

ブレークポイント (停止点)

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)。

プログラムへのブレークポイントの追加

デバッグしたいプログラムにブレークポイントを指定 (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だけ適用されます。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。

ブレークポイント・コマンドについての詳細は、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。

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、次のど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。たとえば、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。

以下の ADDBKP コマンドは、プログラム CUS310 にブレークポイントを追加します。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。 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 変数はアポストロフィで囲んで入力しなければなりません。

CUS310 のソースは以下のとおりです。

```
5728PW1 R01M00  880101                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
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= 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。

条件付きブレークポイント

デバッグ中のプログラムに条件付きブレークポイントを追加することができます。該当のステートメントおよび条件の指定には、ブレークポイント追加 (ADDBKP) コマンドを使用します。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。たとえば、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 コマンドでより一般的なブレークポイントの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。たとえば、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。すべての有効な演算子のリストは、 iSeries Information Center の『**プログラミング**』カテゴリーにある『*CL*』セクションを参照してください。
- 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。たとえば、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 ')
```

プログラムからのブレークポイントの除去

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。

トレース (追跡)

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しなければなりません。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。

プログラムへのトレースの追加

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。

STRDBG コマンドまたは 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) 次のど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 (RMVTRC コマンド)、トレース・データを消去するか (CLRTRCDTA コマンド)、あるいは最大数を変更することができます (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、すで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、次のど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。 (450 ページの『マシン・インターフェース・レベルでのデバッグ』 を参照してください)。

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。

```
                        追跡データの表示

                  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

  開始桁 . . . . . . . . . . . . . . . . . :   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
```

```
                        追跡データの表示


開始桁 . . . . . . . . . . . . . . . . . :   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            2100                      1                     6
CUS310            2200                      1                     7
CUS310            2600                      1                     8  +

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
```

```
                              追跡データの表示

CUS310          2700                          1                          9

開始桁 . . . . . . . . . . . . . . . . . . . :   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
```

命令ごとのステップ処理

STRDBG コマンドまたは CHGDBG コマンドを使用し、MAXTRC パラメーターに 1 を、また TRCFULL パラメーターに *STOPTRC を指定することによってプログラムを 1 命令ごとに実行させること (ステップスルー) ができます。トレースの範囲を ADDTRC コマンドで指定し、プログラムがその範囲内の 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。

トレース範囲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 (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

DSPTRCDTA コマンドでは、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、それとも残しておくかを指定できます。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。

プログラムからのトレースの除去

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、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。

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。

- 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 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。

RMVPGM コマンドと 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。

表示機能

デバッグ・モードでは、テスト情報を表示して、デバッグ・ジョブをどのようにセットアップしたかの確認を行うことができます。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。

変数の値の表示

ブレークポイントに達した時点でプログラム変数の値を表示できます。これをブレークポイント表示画面に自動的に表示させるには、ブレークポイント追加 (ADDBKP) コマンドに変数の名前を指定しておきます。あるいは、ブレークポイントで F10 キーを押してコマンド入力画面を表示し、そこ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 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、次のど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、必ずアポストロフィで前後を囲みます。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 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
```

HLL (高水準言語) 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。

変数の値の変更

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) コマンドを使用してください。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。たとえば、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。

他のジョブのデバッグのためのジョブの使用

他の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が、それには次のような利点があります。

- 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。 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) コマンドを使っ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) コマンドを使ってデバッグを終了させ、次にサービス・ジョブ終了 (ENDSRVJOB) コマンドを使ってジョブのサービスを終了させてください。

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ジョブには、ジョブ待ち行列に投入されないジョブもあります。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。たとえば、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。このプログラム中で、ジョブの遅延 (DLYJOB) コマンドを使って 1 分遅延させ、CALL コマンドを使って名前変更後の該当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。

実行中のジョブのデバッグ

すで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。たとえば、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 (たとえば、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。

他の対話式ジョブのデバッグ

ジョブを他のディスプレイ装置からデバッグすることができますが、これは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行うことができます。他の対話式ジョブのデバッグを行うためには、次のようにしてください。

1. デバッグしたいジョブの修飾ジョブ名を入手する。デバッグしたい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、あるいは活動ジョブ処理 (WRKACTJOB) コマンドを実行することにより、修飾ジョブ名が得られます。
2. 入手した修飾ジョブ名を使ってサービス・ジョブ開始 (STRSRVJOB) コマンドを入力する。
3. デバッグ開始 (STRDBG) コマンドと必要な他のすべてのデバッグ・コマンドを入力する。ジョブがすで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表示させる必要があります。

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。

ジョブを他のジョブからデバッグする場合の考慮事項

ほとんどのジョブが他のジョブからデバッグできますが、次の点に考慮しなければなりません。

- 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 (たとえば、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を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。たとえば、実行中のジョブをデバッグしている時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ただ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、HLL ステートメント番号で停止するのと同様に、該当の 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、それをアポストロフィで囲まなければなりません ('/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。たとえば、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。

セキュリティーに関する考慮事項

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。

デバッグ時の COPY、SAVE、RESTORE、CRTDUPOBJ、および CHKOBJITG の使用

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ートメント・トレースは、デバッグ機能の実行中にプログラムから**一時的に除去されます**。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。

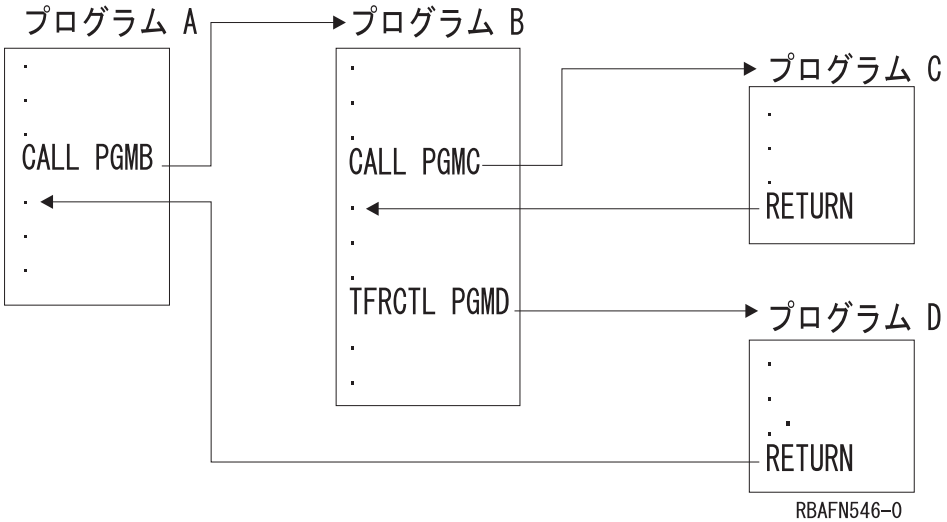
付録 A. TFRCTL コマンド

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。 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 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 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 CALL コマンドの次の順次命令を開始します。

注: TRFCTL コマンドは ILE CL プロシージャーでは無効です。

TFRCTL コマンドの使用法

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。



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き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き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、TFRCTL コマンドを実行するプログラムに渡されていない変数は 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 TRFCTL コマンドは有効です。 2 番目の 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。

```
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
```

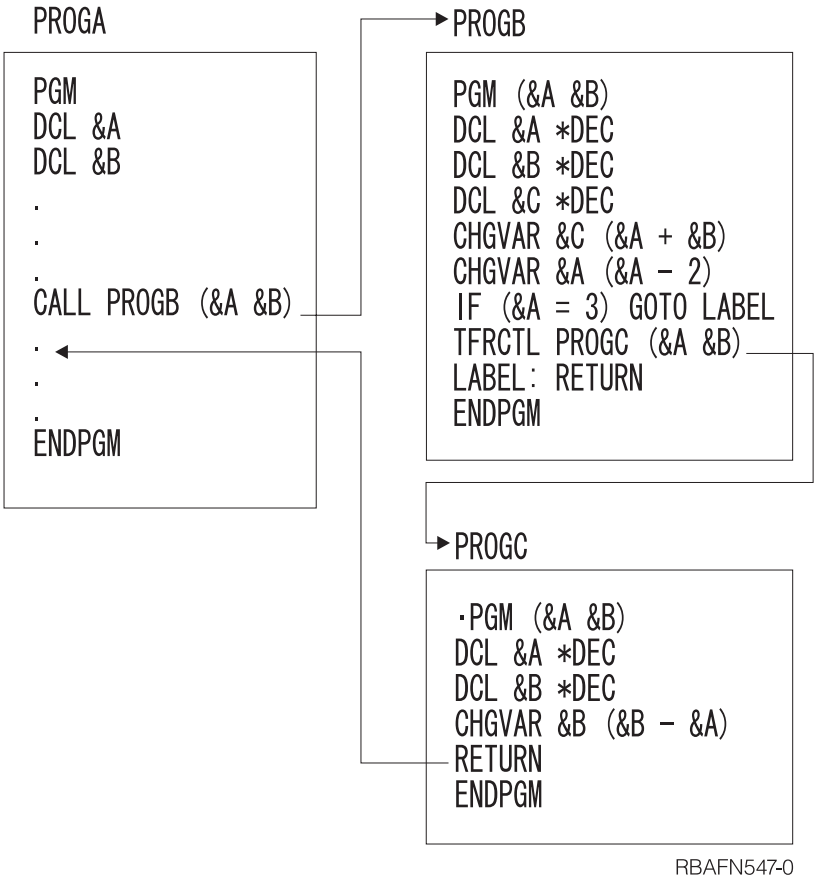
PARM パラメーターについては、 80 ページの『プログラムおよびプロシージャー間のパラメーターの受け渡し』の項で説明します。

パラメーターの引き渡し

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 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。


PROGA
PGM
DCL &A
DCL &B
.
.
.
CALL PROGB (&A &B)
.
.
.
ENDPGM
PROGB
PGM (&A &B)
DCL &A *DEC
DCL &B *DEC
DCL &C *DEC
CHGVAR &C (&A + &B)
CHGVAR &A (&A - 2)
IF (&A = 3) GOTO LABEL
TFRCTL PROGC (&A &B)
LABEL: RETURN
ENDPGM
PROGC
·PGM (&A &B)
DCL &A *DEC
DCL &B *DEC
CHGVAR &B (&B - &A)
RETURN
ENDPGM

RBAFN547-0

付録 B. ジョブ・ログ出力ファイル

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、ジョブ用のジョブ・ログを 1 つまたは 2 つのデータベース・ファイルに作成できます。 1 つ目のファイルは 1 次ジョブ・ログ・ファイルです。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。たと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・ログのモデル

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

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
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

1 次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、次のど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 ～ 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 OPM プログラムと ILE プログラムのどちらだったかを示す標識。

- 標識がゼロ ('F0'X) に設定されている場合、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。送信プログラムの完全名は 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

メッセージ・データの 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 ～ 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 または 2 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。プロシージャー名はそれぞれ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 ('0F'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 ～ 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；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)	フィールド記述
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。

2 次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。

付録 C. ライセンス・プログラム (LP) 内の IBM 提供ライブラリー

iSeries サーバーには、数多くのライブラリーの定義が含まれています。 これらの定義は、システム上に保管されるオブジェクトの大部分を編成するためのメソッドを提供します。

以下の表では、IBM 提供のライブラリーが、ライブラリーを提供したライセンス・プログラム (LP) の下にアルファベット順で示されています。

- 『OS/400 ライセンス・プログラム用の IBM 提供ライブラリー』では、基本オペレーティング・システムであるオペレーティング・システム/400 (Operating System/400) (OS/400) ライセンス・プログラムとともに使用するために提供されているライブラリーを示しています。
- 469 ページの『その他の iSeries ライセンス・プログラム用の IBM 提供ライブラリー』では、その他すべての iSeries ライセンス・プログラムに提供されているライブラリーを示しています。
  - これらのライセンス・プログラムは、各プログラムの全記述名によってアルファベット順に表示されています。これらのライセンス・プログラムは、システムの「ライセンス・プログラムの導入」メニューに表示されます。
  - また、ライセンス・プログラムに複数の機能がある場合、機能名と機能番号がその機能に含まれているライブラリーの前に示されています。

OS/400 ライセンス・プログラム用の IBM 提供ライブラリー

表 12. OS/400 プログラム用の IBM 提供ライブラリー

プログラム名	ライブラリー名	ライブラリーの目的
オペレーティング・システム/400 (5722-SS1)		
	QCCA	CCA cryptographic service provider
	QCLUSTER	HA スイッチャブル・リソース
	QDB2MS	DB2 マルチ・システム
	QDNS	ドメイン・ネーム・システム
	QDOC	システム補助記憶域プール (ASP)、ASP 1 内にある文書ライブラリー・オブジェクトを含む。文書ライブラリー・サービス (DLS) システム・オブジェクトはこの中に含まれている。システムに付属していない。IPL 時に作成される。
	QDOC*nnnn*	ユーザー補助記憶域プール (ASP) 内にある文書ライブラリー・オブジェクトを含む。 *nnnn* は、ASP の 0002 ～ 0016 までの数である。システムに付属していない。IPL 時に作成される。
	QDSNX	システムに付属していない。インストール時に作成される。
	QFNTCPL	AFP* 互換フォント
	QFNTWT	追加フォント

表 12. OS/400 プログラム用の IBM 提供ライブラリー (続き)

プログラム名	ライブラリー名	ライブラリーの目的
	QFPNTWE	Netware 拡張導入機能
	QGDDM	図形データ表示管理プログラム (GDDM*) および図形表示ルーチン (PGR) のサポート
	QGPL	ユーザーの汎用目的ライブラリー
	QGPLTEMP	システムに付属していない。インストール時に作成される。
	QHLPSYS	一部のシステム機能のオンライン文書
	QICSS	ディジタル証明書マネージャー
	QICU	ユニコード用国際コンポーネント
	QIWS	ホスト・サーバー
	QJRNL	HA ジャーナル・パフォーマンス
	QMSE	メディア・ストレージ拡張機能 (MSE) ライブラリー
	QPASE	ポータブル・アプリケーション・ソリューション環境
	QPFRDATA	システム収集パフォーマンス・データのライブラリー
	QQALIB	質疑応答ユーティリティー・ライブラリー
	QRCL	RCLSTG コマンドによるオブジェクトの再利用のためのライブラリー
	QRECOVERY	システム回復ライブラリー
	QRPLOBJ	置換済みオブジェクトのためのライブラリー
	QSCxxxxxxx	APAR データ xxxxxxx (問題識別コードの最後の 7 文字) を収集するためのデータ・ライブラリー
	QSHELL	Qshell
	QSMP	DB2 マルチ・プロセス (SMP)
	QSOC	OptiConnect
	QSPL	スプーリング・ライブラリー
	QSR	オブジェクト接続
	QSRV	システム・サービス・ライブラリー
	QSYS	システム・ライブラリー
	QSYSCGI	拡張ベース・ディレクトリー・サポート
	QSYSDIR	拡張ベース・ディレクトリー・サポート
	QSYSINC	オープンネス
	QSYSLOCALE	*LOCALE オブジェクトを作成するために使用するためのロケール・ソース・メンバーを含む。
	QSYSNLS	拡張 NLS サポート
	QSYSVxRxMx	iSeries CL コンパイラー・ライブラリー、前のリリースのサポート。 x は、前に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・レベルを表す。
	QSYS2	拡張基本サポート。オブジェクト名が文字 Q で始まっていないオブジェクト用の補足システム・ライブラリー。たとえば、CPI 用のオブジェクトを含む。
	QSYSxxxx	オンライン情報
	QTEMP	ユーザーの一時ライブラリー
	QUSRSYS	追加の IBM 提供オブジェクト

表 12. OS/400 プログラム用の IBM 提供ライブラリー (続き)

プログラム名	ライブラリー名	ライブラリーの目的
	QUSRTEMP	システムに付属していない。インストール時に作成される。
	QUSRTOOL	例題ツール
システム/36 実行環境用のライブラリー		
	QSSP	システム/36 実行環境ライブラリー
	QS36F	システム/36 実行環境ライブラリー。システムにより作成される。
	#CGULIB	システム/36 文字作成ユーティリティー (CGU)
	#DFULIB	システム/36 データ・ファイル・ユーティリティー (DFU)
	#DSULIB	システム/36 開発サポート・ユーティリティー (DSU)
	#LIBRARY	システム/36 実行環境、ユーザーの汎用目的のライブラリー
	#SDALIB	システム/36 画面設計機能 (SDA)
	#SEULIB	システム/36 原始ステートメント入力ユーティリティー (SEU)
システム/38 実行環境用のライブラリー		
	QSYS38	システム/38 実行環境ライブラリー

その他の iSeries ライセンス・プログラム用の IBM 提供ライブラリー

表 13. iSeries サーバー上のその他の LP 用のライブラリー

プログラム名	ライブラリー名	ライブラリーの目的
IBM Advanced DBCS Printer Support for iSeries (5722-AP1)		
	QAPS	多機能 (DBCS) 印刷装置サポート
	QAPS2	拡張 DBCS 印刷装置サポート AS/400 用
IBM Advanced Function Printing DBCS Fonts for AS/400 (5769-FN1)		
	QFNT60	AFP DBCS フォント AS/400 用 - ベース・サポート
	ライブラリー: 日本語フォント用	
	QFNT61	AFP DBCS フォント AS/400 用 - 日本語
	ライブラリー: 韓国語フォント用	
	QFNT62	AFP DBCS フォント AS/400 用 - 韓国語
	ライブラリー: 中国語 (繁体字) フォント用	
	QFNT63	AFP DBCS フォント AS/400 用 - 中国語
	ライブラリー: 中国語 (簡体字) フォント用	
	QFNT64	AFP DBCS フォント AS/400 用 - 中国語 (簡体字)
	ライブラリー: タイ語フォント用	
	QFNT65	AFP DBCS フォント AS/400 用 - タイ語
IBM Advanced Function Printing Fonts for AS/400 (5769-FNT)		
	QFNT00	AFP™ フォント用の OS/400 ベース・サポート

表 13. iSeries サーバー上のその他の LP 用のライブラリー (続き)

プログラム名	ライブラリー名	ライブラリーの目的
	QFNT01	OS/400 Sonoran Serif フォント・サポート (Sonoran Serif は Monotype Times New Roman** と機能的に同等)
	QFNT02	OS/400 Sonoran Serif Headliner フォント
	QFNT03	OS/400 Sonoran Sans Serif** フォント・サポート (Sonoran Sans Serif は Monotype Arial** と機能的に同等)
	QFNT04	OS/400 Sonoran Sans Serif Headliner フォント
	QFNT05	OS/400 Sonoran Sans Serif Condensed フォント
	QFNT06	OS/400 Sonoran San Serif Expanded フォント
	QFNT07	OS/400 Monotype Garamond** フォント
	QFNT08	OS/400 Century Schoolbook** フォント
	QFNT09	OS/400 外字および特殊文字フォント
	QFNT10	OS/400 ITC Souvenir** フォント
	QFNT11	OS/400 ITC Avant Garde** フォント
	QFNT12	OS/400 Math および Science フォント
	QFNT13	OS/400 DATA1 フォント
	QFNT14	OS/400 APL2® フォント
	QFNT15	OS/400 OCR A および OCR B フォント
Advanced Job Scheduler for iSeries (5722-JS1)		
	QIJS	ジョブ・スケジューラー
IBM Advanced Function Printing Utilities for iSeries (5722-AF1)		
	QAFP	IBM Advanced Function Printing Utilities for iSeries
IBM Application Program Driver for AS/400 (5722-PD1)		
	QAPD	アプリケーション・プログラム・ドライバー/400
IBM Backup Recovery and Media Services for iSeries (5722-BR1)		
	QBRM	BRM サービス/400 サポート
	QUSRBRM	BRM サービス/400 ユーザー・データ
	Q1ABRMSFnn	ASP(nn) ごとの保管ファイル・ライブラリー
IBM Business Graphics Utility for AS/400 (5722-DS1)		
	QBGU	AS/400 ビジネス・グラフィックス・ユーティリティー (BGU)
IBM CICS Transaction Server for iSeries (5722-DFH)		
	QCICS	CICS/400
	QCICSSAMP	サンプル CICS® アプリケーション・プログラム
IBM Communications Utilities for iSeries (5722-CM1)		
	QRJE	リモート・ジョブ入力 (RJE) および VM/MVS ブリッジ
Content Manager for iSeries (5722-VI1)		
	QVI	Content Manager for iSeries
Content Manager OnDemand for iSeries (5722-RD1)		
	QRDARS	OnDemand

表 13. iSeries サーバー上のその他の LP 用のライブラリー (続き)

プログラム名	ライブラリー名	ライブラリーの目的
	QUSRRDARS	OnDemand ユーザー・データ
	QUSROND	OnDemand ユーザー・データ
Cryptographic Access Provider 128-bit (5722-AC3)		
	QCAP3	Cryptographic Access Provider 128-bit
IBM Cryptographic Support for AS/400 (5722-CR1)		
	QCRP	IBM Cryptographic Support for AS/400
DataPropagator Relational V8 for iSeries (5722-DP4)		
	QDPR	DB2 DataPropagator
IBM DB2 Query Manager and SQL Development Kit for iSeries (5722-ST1)		
	QSQL	構造化照会言語/400 (SQL/400®)
DB2 Universal Database Extenders for iSeries V7.2 (5722-DE1)		
	QDBEX	DB2 UDB エクステンダー
	QDBXM (オプション 2)	XML エクステンダー
	QDB2TX (オプション 1)	テキスト・エクステンダー
	QIMO (オプション 3)	テキスト検索エンジン
DCE Base Services for AS/400® (5769-DC1)		
	QDCE2	DCE Base Services for AS/400
DCE DES Library Routines for AS/400 (5769-DC3)		
	QDCEE	DCE DES Library Routines
	QDCE2	DCE Base Services for AS/400
Developer Kit for Java (5722-JV1)		
	QJAVA	Developer Kit for Java
HTTP Server for iSeries (5722-DG1)		
	QHTTPSVR	HTTP Server
	QTCM	トリガー・キャッシュ・マネージャー
ILE C (5722-WDS)		
	QCLE	ILE C
ILE C++ (5722-WDS)		
	QCPPLE	ILE C++
	QCXXN	ILE C++、前のリリースのサポート
ILE COBOL (5722-WDS)		
	QCBLLE	ILE COBOL ライブラリー
	QCBLLEP	ILE COBOL、前のリリースのサポート
	QLBL	OPM COBOL
	#COBLIB	システム/36 - 互換 COBOL
	QCBL	システム/38 - 互換 COBOL
ILE RPG (5722-WDS)		
	QRPG	RPG/400
	QRPGLE	ILE RPG/400
	QRPGLEP	ILE RPG/400、前のリリースのサポート
	#RPGLIB	システム/36 - 互換 RPG II
	QRPG38	システム/38 - 互換 RPG III
Infoprint Server for iSeries (5722-IP1)		
	QIPS	Infoprint Server
IBM e(logo)server iSeries Access Family (5722-XW1)		
	QCA400W	iSeries Access Family ベース
iSeries Access for Windows® (5722-XE1)		
	QCAEXP	iSeries Access for Windows
iSeries Access for Web (5722-XH1)		
	QIWA	iSeries Access for Web
iSeries Access for Wireless (5722-XP1)		
	QIWR	iSeries Access for Wireless

表 13. iSeries サーバー上のその他の LP 用のライブラリー (続き)

プログラム名	ライブラリー名	ライブラリーの目的
iSeries Client Encryption (128 ビット) (5722-CE3)		
	QCE3	Client Encryption 128 ビット
iSeries Integration for Windows Server (5722-WSV)		
	QNTAP	Integration for Windows Server
IBM Managed System Services for iSeries (5722-MG1)		
	QSVMSS	SystemView 分散管理/400
	QSVDSTRPS	ユーザー・オブジェクト用の SystemView 配布リポジトリー、 LP インストール時に作成される
IBM Performance Tools for iSeries (5722-PT1)		
	QPFR	IBM Performance Tools for iSeries
IBM Query for iSeries (5722-QU1)		
	QQRYLIB	IBM Query for iSeries
IBM System/38 Utilities for AS/400 (5722-DB1)		
	QIDU	システム/38 Query、テキスト管理、およびデータ・ファイル・ユーティリティー (DFU)
IBM System Manager for iSeries (5722-SM1)		
	QSMU	SystemView システム・マネージャー/400
IBM TCP/IP Connectivity Utilities for iSeries (5722-TC1)		
	QTCP	TCP/IP 接続ユーティリティー/400
IBM Toolbox for Java (5722-JC1)		
	QJT400	IBM Toolbox for Java
VisualAge Generator Server for AS/400 (5769-VG1)		
	OVGEN	VisualAge Generator Server
	QGPL	ユーザーの汎用目的ライブラリー
IBM WebSphere Studio Development Suite for iSeries (5722-WDS)		
	QPDA	以下のツールおよびユーティリティーが含まれる。 • APF (拡張印刷機能) • CGU (文字作成ユーティリティー)、DBCS システム専用 • DFU (データ・ファイル・ユーティリティー) • ISDB (対話式ソース・デバッガー) • PDM (プログラム開発管理機能) • RLU (報告書設計ユーティリティー) • SDA (画面設計機能) • SEU (原始ステートメント入力ユーティリティー)

付録 D. CL コマンドおよびキーワードの省略形

このセクションでは、 IBM OS/400 およびその他の IBM iSeries ライセンス・プログラムに含まれている CL コマンドで使用する省略形を、アルファベット順に記載しています。

この情報を活用することにより、コマンド定義を使用する際にコマンドおよびキーワードの命名に一貫性を持たせることができます。 (329 ページの『第 9 章 コマンドの定義』および iSeries Information Center の『**プログラミング**』の『*CL*』のセクションにあるコマンド定義ステートメントの章を参照してください。)

CL コマンド動詞の省略形

ほとんどの CL コマンド名は一貫性のある命名スタイルに従っています。コマンド名の最初の 3 文字は、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。

この 3 文字のコマンド接頭部を、コマンド '動詞' ともいいます。 CL コマンドのほとんどは、以下の共通コマンド動詞のいずれかを使用します。

動詞の省略形	意味
ADD	追加
CHG	変更
CRT	作成
DLT	削除
DSP	表示
END	終了
RMV	除去
STR	開始
WRK	処理

次の表は、コマンド動詞として使用されるすべての省略形をリストしたものです。

動詞の省略形	意味
ADD	追加
ALC	割り振り
ALM	警報
ANS	応答
ANZ	分析
APY	適用
ASK	質問
BLD	組み立て
CFG	構成
CHG	変更
CHK	検査
CLO	クローズ
CLR	消去
CMP	比較

動詞の省略形	意味
CNL	取り消し
CPH	暗号
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
DUP	複製
EDT	編集
EJT	拒否
EML	エミュレート
ENC	暗号化
END	終了
EXP	エクスポート
EXT	抽出
FIL	ファイル
FMT	形式設定、様式
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
OVR	一時変更
PAG	ページ編集
PKG	パッケージ
POS	配置
PRM	プロモート
PRT	印刷
PWR	電源
QRY	照会
RCL	再利用
RCV	受け取り
RGZ	再編成
RLS	解放
RMV	除去
RNM	名前変更
RPL	置換
RQS	要求

動詞の省略形	意味
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 (追跡)
UPD	更新
VFY	検証
VRY	変更
WRK	処理

CL コマンドの省略形

次の表は、CL コマンド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) をリストしたものです。

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
ACNE	処理項目
ACT	活動状態、活動、活動化
ADD	追加
ADM	アプリケーション開発管理機能、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
ADR	アドレス
ADSM	ADSTAR 分散ストレージ管理
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

コマンドの省略形	意味
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 (停止点)
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
CAL	カレンダー
CALL	呼び出し
CAP	取り込み
CBL	COBOL 言語
CCS	変更コントロール・サーバー
CCT	IPX 回線
CCTRTE	回線経路
CCTSRV	回路サービス
CDE	コード、コード化
CDS	コード化データ保管
CFG	構成
CFGL	構成リスト
CFGLE	構成リスト項目
CGY	カテゴリー

コマンドの省略形	意味
CHG	変更
CHK	検査
CHT	図表
CICS	顧客情報管理システム
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
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
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

コマンドの省略形	意味
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
DFT	デフォルト
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
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ウイング
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
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
DO	実行
DOC	文書
DOM	Domino
DPCQ	DSNX/PC 待ち行列
DPR	DataPropagator Relational
DSC	切断
DSK	ディスク
DSP	表示
DSPF	表示装置ファイル
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
DWN	ダウン
E (接尾部)	項目

コマンドの省略形	意味
EDT	編集
EDTD	編集記述
EDU	研修
EJT	拒否
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、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
FMT	形式設定、様式
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
FTN	FORTRAN 言語
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	図形記号セット
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

コマンドの省略形	意味
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
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
IMG	イメージ
IMP	インポート
INF	情報
INP	入力
INS	インストール
INT	内部マシン
INTR	システム間
INZ	初期設定
IPI	IPX より上のインターネット・プロトコル
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
IPX	インターネットワーク・パケット交換
IPXD	IPX 記述
ISDB	対話式ソース・デバッガー
ISDN	統合サービス・ディジタル網
ITF	端末対話機能
ITG	保全性
ITM	項目
IWS	高機能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
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
KBD	キーボード
KEY	キー
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	ローカル
LCLA	ローカル属性

コマンドの省略形	意味
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
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
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
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
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
NDSCTX	NetWare ディレクトリー・サービス・コンテキスト

コマンドの省略形	意味
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
NTF	NetFinity
NTS	Notes
NTW	NetWare
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
NWSD	ネットワーク・サーバー記述
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
OFF	オフ
OMC	オブジェクト管理サイクル
OPC	OptiConnect
OPN	オープン
OPT	光ディスク
ORD	順序
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
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	PC 文書
PCL	プロトコル
PCO	PC 編成プログラム
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

コマンドの省略形	意味
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
PRD	プロダクト
PRF	プロファイル
PRFL	プロファイル・リスト
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
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
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	受け取り
RCY	回復
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

コマンドの省略形	意味
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
RJE	RJE
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
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
RSI	リモート・システム情報
RSM	再開
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

コマンドの省略形	意味
SFW	ソフトウェア
SHD	シャドウ、シャドウ機能
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
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
SVR	サーバー
SWA	活動状態での保管
SWL	停止語リスト
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

コマンドの省略形	意味
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	タイ・ファイル
TIMZON	時間帯
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 (追跡)
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
UBC	アルチメディア・ビジネス会議
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
USF	アルチメディア・システム機能
USG	使用状況
USR	ユーザー
USRIDX	ユーザー索引
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
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
X25	X.25

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。たとえば 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
ACN	処理
ACP	受け入れ
ACQ	獲得
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
ADP	借用、適応
ADPT	アダプター
ADR (または ADDR)	アドレス
ADV	拡張
AFN	類縁性
AIX	AIX オペレーティング・システム
AJE	自動開始ジョブ項目
AFP	拡張機能印刷
ALC	割り振り
ALM	警報
ALR	警報
ALRD	警報記述

キーワードの省略形	意味
ALS	別名
ALW	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
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
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 (停止点)
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

キーワードの省略形	意味
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
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
CGU	文字作成ユーティリティー
CHAR (または CHR)	文字
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
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	コミットメント、注記
CNG	輻輳 (ふくそう)

キーワードの省略形	意味
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
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

キーワードの省略形	意味
CTN	競合
CTS	送信可
CTX	コンテキスト
CUR	現行
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
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	送達
DEP	従属
DEPT	非依存
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
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	送達
DLY	遅延
DMN	ドメイン
DMP	ダンプ
DN	識別名
DOC	文書
DPR	DataPropagator Relational

キーワードの省略形	意味
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
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
E (接尾部)	項目
EBCDIC	拡張 2 進化 10 進コード
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	拒否
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、実行
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
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
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

キーワードの省略形	意味
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	形式設定、様式
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
GPH	グラフ
GRP	グループ
GRT	権限認可
GSS	図形記号セット
GVUP	放棄
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

キーワードの省略形	意味
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
IDD	対話式データ定義
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
IN	入力
INAC (または INACT)	非活動
INACT (または INAC)	非活動
INC	組み込み
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
IPDS	高機能印刷装置データ・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・ディジタル網
ISP	インターネット・サービス・プロバイダー

キーワードの省略形	意味
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
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	ローカル
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
LST	リスト、最後の
LTR	文字
LVL	レベル

キーワードの省略形	意味
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
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
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

キーワードの省略形	意味
NAM	名前
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
NEP	非終止プログラム
NET	ネットワーク
NFY	通知
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	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

キーワードの省略形	意味
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
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)

キーワードの省略形	意味
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
PRB	問題
PRC	プロシージャー、プロシージャーの、プロセス
PRD	プロダクト、生産
PRJ	プロジェクト
PREBLT	事前作成
PRED	先祖
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
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
PWD	パスワード
PWR	電源

キーワードの省略形	意味
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
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	受け取り
RCY	回復
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
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

キーワードの省略形	意味
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
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	シャドウ、シャドウ機能
SHF	シフト
SHM	短期保留モード
SHR	共用
SHUTD	シャットダウン

キーワードの省略形	意味
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
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
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

キーワードの省略形	意味
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
STS	状況
STT	状態
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	サブネットワーク
SUBPGM	サブプログラム
SUBST	置換
SUCC	継承者
SUN	日曜日
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
SYS	システム
SYSLIBL	システム・ライブラリー・リスト
S36	System/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
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

キーワードの省略形	意味
TIMMRK	刻時
TIMO	タイムアウト
TIMOUT (または TIMO)	タイムアウト
TIMZON	時間帯
TKN	トークン
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 (追跡)
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

キーワードの省略形	意味
URL	URL
USG	使用状況
USR	ユーザー
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
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

付録 E. 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あり、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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

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。

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 49XA
3605 Highway 52 N

Rochester, MN 55901
U.S.A.

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。

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。お客様は、IBM 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。

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_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

CL プログラミング資料には、ユーザーが CL プログラムを作成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

以下は、IBM Corporation の商標です。

Advanced 36
AFP
Application System/400
CICS
CICS/400
Client Access
COBOL/400
C/400
DataPropagator
DB2
DB2 Universal Database
e (logo)
IBM
Integrated Language Environment
IPDS
iSeries
Operating System/400
OS/400
RPG/400
System/36
System/38

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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

参照文献

OS/400 プログラミングに関する詳細については、次の資料と iSeries Information Center トピックを参照してください。

- 「Application Display Programming」 
  この資料は、画面の作成と保持のための DDS の使用法、表示装置ファイルの作成と処理、オンライン・ヘルプ情報の作成、画面の定義のための UIM の使用法、およびパネル・グループ、レコード、および文書の使用法について説明しています。

- 「バックアップおよび回復の手引き」 
  本書には、iSeries サーバーのリカバリーおよび可用性オプションに関する一般情報が記載されています。サーバーで使用可能なオプションが説明され、比較対比され、詳細情報が存在する場所が示されます。
- iSeries Information Center の『**システム管理**』カテゴリーの『**バックアップおよび回復**』情報トピックには、バックアップおよび回復方針を計画する方法、サーバーをバックアップする方法、テープ・ライブラリーを管理する方法、およびデータ用にディスク保護をセットアップする方法に関する情報が記載されています。また、iSeries(TM) ナビゲーターの Backup Recovery and Media Services プラグインに関する情報、サーバーの回復に関する情報、およびバックアップと回復に関してよく尋ねられるいくつかの質問に対する答えが記載されています。
- iSeries Information Center の『**プログラミング**』情報カテゴリーの『**CL**』トピックには、OS/400 コマンドや他の IBM 提供コマンドを使用してプログラムを作成するシステム・プログラマーやシステム管理者のための情報が記載されています。

- 「ILE 概念」 
  この資料は、OS/400 ライセンス・プログラムの統合言語環境 (ILE) 体系に関する概念と用語を説明しています。モジュールの作成、バインド、プログラムの実行、プログラムのデバッグ、および例外の処理に関するトピックが記載されています。
- iSeries Information Center の『**データベース**』情報カテゴリーには、 DB2 Universal Database™ (UDB) for iSeries に関する情報が記載されています。これは、アプリケーションやユーザー・インターフェースを介したサーバー・データのアクセスと管理を可能にするとともに、参照保全や並列データベース処理などの拡張機能を提供します。
- iSeries Information Center の『**ファイルおよびファイル・システム (Files and File Systems)**』情報カテゴリーには、IBM iSeries サーバーのデータベース・ファイル管理機能と統合ファイル・システム機能の両方に関する情報が記載されています。
- iSeries Information Center の『**プログラミング**』情報カテゴリーの『**グローバリゼーション (Globalization)**』トピックは、データ処理管理者、システム・オペレーターおよび管理者、アプリケーション・プログラマー、エンド・ユーザー、 IBM 営業担当員、およびシステム・エンジニアを対象として、 iSeries サーバーの各国語サポート機能の理解や使用に必要な事項を説明しています。 この手引きは、ユーザーが、iSeries サーバーのグローバリゼーションおよび多国語サポート機能の計画、導入、構成、および使用を準備するためのものです。また、多国語データのデータベース管理、および多国語システムのアプリケーションの考慮事項についても説明しています。

- 「印刷装置プログラミング」 
  この資料は、印刷の要素と iSeries サーバーの概念、印刷装置ファイルおよび印刷スプーリング・サポート、およびプリンターの接続性について説明しています。
- iSeries Information Center の『**情報**』情報カテゴリーには、印刷機能を計画および構成する方法に関する情報や、印刷に関する基本情報が記載されています。

- 「機密保護解説書」 

  この資料は、システムのセキュリティーに関する一般的な概念、およびその計画について説明しています。また、すべてのユーザーを対象に資源保護に関する情報も示しています。
- iSeries Information Center の『**セキュリティー**』情報カテゴリーには、 iSeries セキュリティーをセットアップおよび計画する方法、ネットワークおよび通信アプリケーションを保護する方法、および高度に保護された暗号処理機能を iSeries サーバーに追加する方法に関する情報が記載されています。また、オブジェクト署名と署名検査、ID マッピング、およびインターネット・セキュリティー・リスクの解決策に関する情報が記載されています。
- iSeries Information Center の『**システム管理**』カテゴリーは、実行管理環境の設定と変更、システム値の取り扱い、およびシステムのパフォーマンスの向上のためのパフォーマンス・データの収集と使用法について説明しています。
- iSeries Information Center の『**プログラミング**』情報カテゴリーの『**API**』トピックは、熟練したアプリケーションやシステムのプログラマーが、 OS/400 アプリケーション・プログラム・インターフェース (API) を使用したい場合の情報について説明しています。

本書で説明されている OS/400 プログラミング・ユーティリティーの詳細については、次の資料を参照してください。

- 「AS/400 V3 適用業務開発ツール (ADTS)・セット/400: 文字作成ユーティリティー (CGU)」
- 「AS400 プログラム開発管理機能」 
- 「適用業務開発ツールセットAS/400 用: 画面設計機能」 
- 「適用業務開発ツールセットAS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照」 

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

D

E

G

H

I

L

M

O

P

Q

R

S

T

U

W

[特殊文字]

Printed in Japan

SD88-5038-06

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12