Enterprise COBOL for z/OS and OS/390

コンパイラーおよび実行時プログラム移行ガイド

バージョン 3.1

GC88-9118-00
(英文原典：GC27-1409-00)

Enterprise COBOL for z/OS and OS/390

コンパイラーおよび実行時プログラム移行ガイド

バージョン 3.1

GC88-9118-00
(英文原典：GC27-1409-00)

お願い

本書および本書がサポートする製品をご使用になる前に、367 ページの 『付録 N. 特記事項』を必ずお読みください。

本書は GC88-7054-03 の改訂版です。本書の技術的な変更は、「変更の要約」で要約されており、変更個所の左側の縦線によって示されています。

本書は、IBM Enterprise COBOL for z/OS and OS/390 バージョン 3 リリース 1 (5655-G53)、および新版やテクニカル・ニュースレターで特に断りがない限り、以降のリリースにも適用されます。製品のレベルに応じた正しい版を使用していることを確認してください。

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)

原　典： GC27-1409-00
Enterprise COBOL for z/OS and OS/390
Compiler and Run-Time Migration Guide
Version 3 Release 1

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 2002.2

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

本書について

本書には、言語環境プログラム以前のランタイム・ライブラリーを IBM 言語環境プログラム (z/OS および OS/390 版) に移行するとき、およびソース・プログラムを IBM Enterprise COBOL for z/OS and OS/390 にアップグレードするときに役立つ情報が記載されています。

用語の説明

本書では、Enterprise COBOL という用語は以下のものを一般的に指しています。

- IBM Enterprise COBOL for z/OS and OS/390

本書では、IBM COBOL という用語は以下のものを一般的に指しています。

- COBOL/370 バージョン 1 リリース 1
- COBOL (MVS および VM 版) バージョン 1 リリース 2
- COBOL (OS/390 および VM 版) バージョン 2 リリース 1
- COBOL (OS/390 および VM 版) バージョン 2 リリース 2

表 1. COBOL コンパイラー名、バージョン、リリース、およびプロダクト番号

コンパイラー	リリース・レベル	プロダクト番号
COBOL/370	バージョン 1 リリース 1	5688-197
COBOL (MVS および VM 版)	バージョン 1 リリース 2	5688-197
COBOL (OS/390 および VM 版)	バージョン 2 リリース 1	5648-A25
COBOL (OS/390 および VM 版)	バージョン 2 リリース 2	5648-A25
COBOL for z/OS and OS/390	バージョン 3 リリース 1	5655-G53

本書では、ランタイムを言語環境プログラムへ移行するときに役立つように、既存の VS COBOL II および OS/VS COBOL ロード・モジュールを言語環境プログラムのもとで実行する方法 (サポートのためのリンク・エディット要件および互換性のある動作のための推奨されるランタイム・オプションを含む) を説明します。

本書では、ソース・プログラムを Enterprise COBOL によってサポートされる COBOL 85 標準にアップグレードするときに役立つように、COBOL 74 標準と COBOL 85 標準間の言語の違いを説明します。さらに、ソース・プログラムを Enterprise COBOL プログラムに移行するのに役立てることができる IBM 移行ツールを説明します。

ランタイムとソースの両方のタイプの移行について、本書にはサンプルを使った戦略とシナリオが示されています。

謝辞

IBM では、*OS/VS COBOL to VS COBOL II Migration Guide* の準備における GUIDE COBOL Migration Task Force の援助に感謝の意を表します。Task Force からは、OS/VS COBOL から VS COBOL II への移行に関して、さまざまなアイデア、経験に基づく情報、および明敏な解説の提供を受けました。

この以前の移行の経験から得た情報と、OS/VS COBOL および VS COBOL II の経験を積んだ多くのお客様から得た情報は、このコンパイラーおよび実行時プログラム *移行ガイド* の開発に役立ちました。これらのご支援がなければ、本書の開発はもっと困難であったと思われます。

資料の使い方

Enterprise COBOL および言語環境プログラムと共に提供される資料は、z/OS または OS/390 のもとで COBOL プログラミングを行う際に役立ちます。

Enterprise COBOL for z/OS and OS/390

表 2. COBOL for z/OS and OS/390 の資料

作業	資料	オーダー番号
プロダクトを評価する	*Fact Sheet*	GC27-1407
保証情報を理解する	*Licensed Program Specifications*	GC27-1411
z/OS のもとでコンパイラーをインストールする	*プログラム・ディレクトリー*	GI88-8573
OS/390 のもとでコンパイラーをインストールする	*プログラム・ディレクトリー*	GI88-8573
プロダクトの変更点を理解する — ソースを COBOL for z/OS and OS/390 に、ランタイムを言語環境プログラムにアップグレードする	*コンパイラーおよび実行時プログラム 移行ガイド*	GC88-9118
Enterprise COBOL for z/OS and OS/390 をカスタマイズする	*カスタマイズ・ガイド*	GC88-9119
プログラムを作成およびテストし、コンパイラー・オプションに関する詳細を入手する	*プログラミング・ガイド*	SC88-9121
COBOL 構文および言語エレメントの仕様に関する詳細を入手する	*言語解説書*	SC88-9117

z/OS バージョン 1 リリース 1 以上の言語環境プログラム・エレメント

表 3. z/OS の言語環境プログラム・エレメントの資料

作業	資料	オーダー番号
プロダクトを評価する	*概念*	SA88-8555
言語環境プログラムのインストール	*z/OS Program Directory*	GI10-0670
言語環境プログラムのプログラム・モデルおよび概念を理解する	*プログラミング・ガイド*	SA88-8549
言語環境プログラムのランタイム・オプションおよび呼び出し可能サービスの構文を見つける	*プログラミング・リファレンス*	SA88-8550

表 3. z/OS の言語環境プログラム・エレメントの資料 (続き)

作業	資料	オーダー番号
言語環境プログラムのもとで実行されるアプリケーションをデバッグし、ランタイム・メッセージの詳細を入手し、言語環境プログラムに関する問題を診断する	*デバッグのガイド* *ランタイム・メッセージ*	GA88-8548 SA88-8554
アプリケーションを言語環境プログラムにマイグレーションする	*ランタイム マイグレーション・ガイド*	GA88-8553
言語間通信 (ILC) アプリケーションを開発する	*ILC (言語間通信) アプリケーションの作成*	SA88-8551

OS/390 バージョン 2 リリース 10 の言語環境プログラム・エレメント

表 4. OS/390 の言語環境プログラム・エレメントの資料

作業	資料	オーダー番号
プロダクトを評価する	*概念*	GC88-7592
言語環境プログラムのインストール	*OS/390 Program Directory*	GI10-4001-08
言語環境プログラムをカスタマイズする	*OS/390 版 カスタマイズ*	SC88-7595
言語環境プログラムのプログラム・モデルおよび概念を理解する	*プログラミング・ガイド*	SC88-7593
言語環境プログラムのランタイム・オプションおよび呼び出し可能サービスの構文を見つける	*プログラミング・リファレンス*	SC88-7594
言語環境プログラムのもとで実行されるアプリケーションをデバッグし、ランタイム・メッセージの詳細を入手し、言語環境プログラムに関する問題を診断する	*デバッグのガイドおよびランタイム・メッセージ*	SC88-7596
アプリケーションを言語環境プログラムにマイグレーションする	*ランタイム 移行ガイド*	SC88-7598
言語間通信 (ILC) アプリケーションを開発する	*ILC (言語間通信) アプリケーションの作成*	SC88-7597

変更の要約

移行ガイドに対する変更

ここでは、COBOL コンパイラーおよびランタイム・ライブラリーに対する変更の結果として、本書 (*コンパイラーおよび実行時プログラム 移行ガイド*) に対して行われた主要な変更を示します。

技術上の変更については、左側の余白にリビジョン・バーが付いています。

2001 年 11 月 — GC88-9118-00 (英文原典: GC27-1409-00)

コンパイラー

- CMPR2 コンパイラー・オプションを含むいくつかのコンパイラー・オプションが除去されました。
- 新しい予約語が追加されました。
- 新しい組み込みの CICS トランスレーターに関する情報が追加されました。
- SOM ベースの COBOL 構文およびプログラミング・モデルが除去されました。
- Enterprise COBOL コンパイラーへのマイグレーションに関する情報が追加されました。

ランタイム

- DATA(31) プログラムの動作の変更に関する情報が追加されました。
- DUMP マクロを使用するアセンブラー・プログラムを持つアプリケーションでの CEEDUMP の欠如に関する情報が追加されました。
- RECORDING MODE U を指定した COBOL プログラムにおけるファイル処理方法の変更に関する情報が追加されました。
- アセンブラーと COBOL との間の呼び出しに関する情報が追加されました。

2000 年 9 月 — GC88-7054-03 (英文原典: GC26-4764-05) における変更

コンパイラー

- 131 ページの『第 10 章 OS/VS COBOL ソース・プログラムのアップグレード』では、新しく発見された文書化されていない拡張が追加され、また、多数の既存の項目が改良されました。
- 新しい予約語が追加されました。
- V2R2 コンパイラーへのマイグレーションに関する情報が追加されました。

ランタイム

- ランタイム・オプション ABTERMENC (OS/390 V2R9 以上の言語環境プログラムの場合は ABEND) の新しいデフォルトおよび OS/390 V2R7 以上の言語環境プログラムで使用可能な新しいサブオプション TERMTHDACT の説明が追加されました。
- 言語環境プログラムの領域全体ランタイム・オプションに関する情報が追加されました。
- 仮想記憶要件が更新されました。
- CICS 考慮事項が更新されました。
  - パフォーマンス
  - SORT インターフェースの変更
  - DISPLAY ステートメント
- 言語環境プログラムのリリース・レベルのアップグレードに関する情報が更新されました。

各種の保守上および編集上の変更が加えられました。

COBOL コンパイラーに対する変更の要約

ここでは、IBM ホスト COBOL コンパイラーに対して行われた主要な変更を示します。

技術上の変更については、左側の余白にリビジョン・バーが付いています。

IBM Enterprise COBOL for z/OS and OS/390 における変更

- マルチスレッド化のサポート: POSIX スレッドおよびシグナルの許容により、COBOL プログラムが含まれるアプリケーションを 1 つのプロセス内でマルチスレッドで実行できるようになりました。
- オブジェクト指向の構文による COBOL と Java の相互運用性 (インターオペラビリティー) により、COBOL プログラムで Java クラスのインスタンスを生成したり、Java オブジェクトのメソッドを呼び出すことができるようになりました。また、Java または COBOL でインスタンスを生成することができ、かつ、Java または COBOL で呼び出すことができるメソッドを持つ Java クラスを COBOL プログラムで定義することができます。
- Java Native Interface (JNI) によって提供されるサービスを呼び出すことで、Java の追加機能を取得することができます。また、JNI へのアクセスを容易にするためのコピーブック JNI.cpy および特殊レジスター JNIENVPTR が提供されています。
- Unicode の基本的なサポートを提供するために、国別データ型および国別 (N、NX) リテラル、文字変換用の組み込み関数 DISPLAY-OF および NATIONAL-OF、およびコンパイラー・オプション NSYMBOL および CODEPAGE が追加されました。
  - 国別リテラル、英数字および DBCS のデータ項目とリテラルのエンコードに使用するコード・ページを指定するためのコンパイラー・オプション CODEPAGE が追加されました。

- N 記号を使用するリテラルおよびデータ項目に対して国別または DBCS の処理を適用するかどうかを制御するコンパイラー・オプション NSYMBOL が追加されました。

- 基本的な XML サポートが追加されました。これには、高速の XML パーサーが含まれます。この XML パーサーにより、プログラムではインバウンド XML メッセージを利用し、メッセージが整形式 (well-formed) かどうかを検査して COBOL データ構造に変換することができます。また、Unicode UTF-16 または複数の単一バイト EBCDIC コード・ページでエンコードされた XML 文書もサポートされます。
- 個別の変換ステップなしで、CICS ステートメントを含むプログラムをコンパイルできるようになりました。
  - コンパイラー・オプション CICS が追加され、組み込みの CICS トランスレーターの使用と CICS オプションの指定が可能になりました。
- BINARY データ項目のための VALUE 文節が追加されました。これにより、数字リテラルは、PICTURE 文節内での 9 の数で暗黙的に指定される値に制限されることなく、本来のバイナリー表現の容量と同じ大きさまでの値を持つことができます。
- 4 バイトのデータ項目 FUNCTION-POINTER に COBOL または COBOL 以外の入り口点のアドレスを指定することができ、これにより、C の関数ポインターとのインターオペラビリティーが向上しました。
- *移行ガイド* に記載されているように、以下のサポートは中止されました。
  - SOM ベースのオブジェクト指向構文およびサービス
  - コンパイラー・オプション CMPR2、 ANALYZE、 FLAGMIG、 TYPECHK、および IDLGEN

COBOL (OS/390 および VM 版) バージョン 2 リリース 2 における変更

- 10 進データのサポートが拡張され、10 進数の最大桁数が 18 から 31 に引き上げられたことにより、算術計算に対して拡張精度のモードが提供されるようになりました。
- コンパイル・フックではなくオーバーレイ・フックを使用する拡張実動デバッグ機能が追加されました。シンボリック・デバッグ情報は、必要に応じて別個のファイルに入れることができます。
- 階層ファイル・システム (HFS) に COBOL ファイルを常駐させた状態で、OS/390 UNIX システム・サービス環境におけるコンパイル、リンク、および実行がサポートされるようになりました。
- fork()、exec()、および spawn() の許容が追加され、UNIX/POSIX 関数を呼び出すことができるようになりました。
- 入出力機能が拡張され、SELECT. . . ASSIGN で指定された環境変数による、ファイルの動的割り振りが可能になりました。また、ACCEPT や DISPLAY などを使用して、順次編成の HFS ファイルにアクセスできるようになりました。
- レコードが改行文字で区切られているテキスト・データが格納された HFS ファイルにアクセスするための行順ファイル編成がサポートされるようになりました。

- COMP-5 データ型がホスト COBOL に新たに追加されました。これにより、本来のバイナリー表現の容量と同じ大きさまでの値を持つことができます。
- TRUNC(BIN) コンパイラー・オプションを指定したバイナリー・データの処理におけるパフォーマンスが大幅に改善されました。
- OS/390 DFSMS バインダーのみを使用する COBOL アプリケーションのリンクがサポートされるようになりました。プリリンカーは、CICS での例外的なケースでのみ必要になります。
- コンパイラー・オプション DIAGTRUNC を指定することで、数値の切り捨てを招く (暗黙的または明示的な) 移動の診断が可能になりました。
- BLKSIZE=0 を指定することで、リスト・データ・セット用として、システムで判別されたブロック・サイズを使用できるようになりました。
- QSAM テープ・ファイルのブロック・サイズ最大値が 2GB になりました。
- CICS のもとで、システム論理出力装置宛ての DISPLAY および日時の取得のための ACCEPT がサポートされるようになりました。
- SQL コンパイラー・オプションを使用して DB2 coprocessor がサポートされるようになりました。これにより、個別のプリコンパイル・ステップが不要になり、ネストされたプログラムおよびコピーブックで SQL ステートメントを使用できるようになりました。
- 2000 年言語拡張のサポートが基本 COBOL プロダクトに追加されました。

COBOL (OS/390 および VM 版) V2 R1 モディフィケーション 2 における変更

- 新しいコンパイラー・オプション ANALYZE が追加され、組み込み SQL ステートメントおよび CICS ステートメントの構文を検査できるようになりました。
- Working Draft for Proposed Revision of ISO 1989:1985 Programming Language COBOl に記載されている勧告に準拠するため、ACCEPT ステートメントが拡張されました。
- 新しい組み込み日付関数が追加され、4 桁の年を持つ日付を変換できるようになりました。
- 2000 年言語拡張が追加され、2 桁または 4 桁の年を持つ日付についてコンパイラー援助日付処理が可能になりました。

使用中のコンパイラーに IBM VisualAge Millennium Language Extensions for OS/390 & VM (プログラム番号 5648-MLE) をインストールする必要があります。

COBOL (OS/390 および VM 版) V2 R1 モディフィケーション 1 における変更

- 金融データの表示についての通貨サポートに対して、以下の拡張が行われました。
  - 複数文字からなる通貨符号のサポート
  - 単一プログラムにおける複数の通貨符号タイプのサポート
  - 経済通貨同盟 (EMU) により定義されたユーロ通貨符号のサポート

COBOL (OS/390 および VM 版) バージョン 2 リリース 1 における変更

- ダイナミック・リンク・ライブラリー (DLL) のサポートが追加されました。
- OS/390 リリース 3 で提供された SOMobjects プロダクトの変更により、オブジェクト指向 COBOL アプリケーションの作成に使用する JCL の変更が必要になりました。
- INTDATE コンパイラー・オプションがインストール・オプションに限定されなくなり、コンパイラーの呼び出し時に、オプションとして指定できるようになりました。

第 1 部 概要

第 1 章 再コンパイルする必要がありますか ?

プログラムを、サポートされるコンパイラー (IBM Enterprise COBOL for z/OS and OS/390 をお勧めします) でコンパイルし、サポートされるランタイム・ライブラリー (言語環境プログラム) を使用して実行することが理想的です。次のようにすれば、完全にサポートされる方法で、この理想的な状態に段階的に近づけることができます。

1. 最初にランタイム・マイグレーションを行い、その後*任意に*
2. コンパイラー・マイグレーションを行う

ほとんどのプログラムの場合、プログラムが正しく機能し続けること、またはプログラムがサービス・サポートを得ることを確実にするために再コンパイルする必要はありません。アップグレードする必要のあるプログラムの詳細については、以下を参照してください。

- OS/VS COBOL ランタイムを使用して実行される OS/VS COBOL プログラムの場合、75 ページの 『アップグレードが必要なプログラムの判別』
- VS COBOL II ランタイムを使用して実行される OS/VS COBOL および VS COBOL II プログラムの場合、90 ページの 『アップグレードが必要なプログラムの判別』
- IBM COBOL プログラムの場合、191 ページの 『Enterprise COBOL を使用してコンパイルする前にアップグレードが必要なプログラムの判別』

この章では、いつ、どのような理由で、アプリケーション (ランタイムまたはソース) をマイグレーションするかについて説明します。以下のトピックが記載されています。

- マイグレーションの基本
- OS/VS COBOL および VS COBOL II プログラム用のサービス・サポート

マイグレーションの基本

マイグレーション作業としては、ランタイム・マイグレーション (アプリケーションを新しいランタイムに移行する) とソース・マイグレーション (ソース・プログラムをアップグレードする) があります。マイグレーション作業の一部として、アプリケーションの目録の評価およびテストを行うことも必要になります。すでに述べたように、ランタイムとソースを並行してマイグレーションする必要はありません。

マイグレーション作業の詳細については、19 ページの『一般的な移行作業』を参照してください。

ランタイム・マイグレーション

どの COBOL プログラムも、実行するためにはランタイム・ライブラリーを必要とします。ランタイム・ライブラリーは、ロード・モジュールに静的にリンクされる

(NORES コンパイラー・オプションを指定してコンパイルする) か、または実行時に動的にアクセスされます (RES コンパイラー・オプションを指定してコンパイルする)。

言語環境プログラムへの移行

NORES オプションを指定してコンパイルし、OS/VS COBOL ランタイム・ライブラリーまたは VS COBOL II ランタイム・ライブラリーを使用してリンク・エディットしたプログラムから構成されたロード・モジュールの場合は、REPLACE リンケージ・エディター制御ステートメントを使用して、既存のランタイム・ライブラリー・ルーチンを言語環境プログラム・バージョンで置き換える必要があります。オブジェクト・プログラム (リンク済みでない) の場合は、言語環境プログラムを使用してリンク・エディットするだけです。

RES オプションを指定してコンパイルしたプログラムの場合は、実行時に、LNKLST、LPALST、JOBLIB、または STEPLIB を使用して、OS/VS COBOL ライブラリー・ルーチンまたは VS COBOL II ライブラリー・ルーチンの代わりに言語環境プログラム・ライブラリー・ルーチンを使用可能にしてください。

実行時にアプリケーションで複数の COBOL ランタイム・ライブラリーを使用可能にしてはなりません。たとえば、LNKLST には、COBOL ランタイム・ライブラリーが 1 つ (言語環境プログラムの SCEERUN など) しかあってはなりません。複数の COBOL ランタイム・ライブラリーがあると、検出が困難なエラーが発生するか、または使用されないロード・ライブラリーが連結中に存在することになります。また、連結中に複数のランタイム・ライブラリーがあると、IBM でサポートされない無効な構成になります。

ランタイムを言語環境プログラムに移行する方法の追加情報については、以下の章を参照してください。

- 59 ページの『第 5 章 言語環境プログラムのもとでの既存のアプリケーションの実行』
- 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』
- 87 ページの『第 7 章 VS COBOL II ランタイムからの移行』

ソース・マイグレーション

ソース・マイグレーションは、ほとんどのプログラムの場合は必要ありません。OS/VS COBOL プログラムまたは VS COBOL II プログラムを言語環境プログラムのもとで実行するために移行したあとで行うことができます。

ソース・マイグレーションとは、通常、使用するソース言語レベルをアップグレード (たとえば、OS/VS COBOL でサポートされる COBOL 74 標準から Enterprise COBOL でサポートされる COBOL 85 標準へ) することです。また、アプリケーションを言語環境プログラムのもとで実行できるようにするためにソース・マイグレーションが必要になることもあります。

ソース・コードをアップグレードするときに役立つ移行ツールがいくつかあります。詳細については、 293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

OS/VS COBOL および VS COBOL II プログラム用のサービス・サポート

OS/VS COBOL および VS COBOL II コンパイラーでコンパイルされたプログラムが COBOL ライブラリー・ルーチンの言語環境プログラム・ランタイム・ライブラリー・バージョンを使用するときは、IBM はそのプログラムの実行のためのサービス・サポートを引き続き提供します。

たとえば、OS/VS COBOL プログラム用のライブラリー・ルーチンは OS/VS COBOL、VS COBOL II、および言語環境プログラム・ランタイム・ライブラリーに存在します。OS/VS COBOL ライブラリーまたは VS COBOL II ランタイム・ライブラリーを使用して実行される OS/VS COBOL プログラムは、IBM サービスでサポートされません。言語環境プログラム・ランタイム・ライブラリーのサポートされるリリースを使用して実行される OS/VS COBOL プログラムは、IBM サービスでサポートされます。

CICS のもとで実行される OS/VS COBOL プログラムには、実行する CICS プロダクトからの特別なサポートと、言語環境プログラムからのサポートが必要となります。CICS TS バージョン 2 リリース 2 以降の CICS TS (Transaction Server) では、この特別なサポートを利用できなくなります。CICS TS 2.2 以降の CICS では、COBOL ランタイム・ライブラリーとして言語環境プログラムを使用しても、OS/VS COBOL プログラムを実行することはできません。CICS のもとで実行されるすべての OS/VS COBOL プログラムは、できるだけ早く Enterprise COBOL にアップグレードしてください。

OS/VS COBOL プログラムの変更

OS/VS COBOL コンパイラーはサポートされなくなりましたが、このコンパイラーによって生成されたプログラムは、言語環境プログラムのもとで実行される場合はサポートされます。ランタイムを言語環境プログラムに移行したあと、ソース移行ツール (COBOL および CICS 移行援助プログラム (CCCA) など) を使用してソース・コードを実行し、次いで Enterprise COBOL コンパイラーを使用してコンパイルすることができます。

CCCA の詳細については、293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

第 2 章 新しいコンパイラーおよびランタイムの紹介

この章では、Enterprise COBOL コンパイラー (IBM Enterprise COBOL for z/OS and OS/390) および共通のランタイム (言語環境プログラム) の概要を示し、本書で使用する用語を紹介します。この章には、以下のトピックに関する情報が記載されています。

- プロダクトの関係 — コンパイラー、ランタイム、デバッグ
- 各種の COBOL コンパイラーの比較
- 他のプログラムのための言語環境プログラムのランタイム・サポート
- 新しいコンパイラーおよびランタイムの利点
- 新しいコンパイラーおよびランタイムへのアップグレードの障害となるもの
- 新しいコンパイラーおよびランタイムに関する主要な変更点
- 一般的な移行作業

用語の説明

本書では、Enterprise COBOL という用語は以下のものを一般的に指しています。

- IBM Enterprise COBOL for z/OS and OS/390 バージョン 3 リリース 1

本書では、IBM COBOL という用語は以下のものを一般的に指しています。

- COBOL/370 バージョン 1 リリース 1
- COBOL (MVS および VM 版) バージョン 1 リリース 2
- COBOL (OS/390 および VM 版) バージョン 2 リリース 1
- COBOL (OS/390 および VM 版) バージョン 2 リリース 2

プロダクトの関係 — コンパイラー、ランタイム、デバッグ

IBM Enterprise COBOL for z/OS and OS/390 は、IBM の zSeries プラットフォームおよび System 390 用の戦略的な COBOL コンパイラーです。Enterprise COBOL は、IBM COBOL、VS COBOL II、および OS/VS COBOL の機能はもとより、マルチスレッド使用可能性、Unicode、XML 構文解析機能、Java とのインターオペラビリティーのためのオブジェクト指向 COBOL 構文、組み込みの CICS トランスレーター、および組み込みの SQL coprocessor および DB2 coprocessor などの追加機能を備えています。Enterprise COBOL、IBM COBOL、および VS COBOL II は、いずれも COBOL 85 標準をサポートします。IBM COBOL でサポートされていた CMPR2 コンパイラー・オプションおよび SOM ベースのオブジェクト指向 COBOL 構文など一部の機能は、Enterprise COBOL ではサポートされません。

言語環境プログラムは、COBOL、PL/I、C、および FORTRAN に単一の言語ランタイム環境を提供します。既存のアプリケーションのサポートに加えて、言語環境プログラムは、共通の条件処理、向上した言語間通信 (ILC)、再使用可能ライブラリー、およびより効率的なアプリケーション開発も提供します。アプリケーション開

発は、共通の規則、共通のランタイム機能、および一連の共用呼び出し可能サービスの使用により単純化されます。Enterprise COBOL プログラムを実行するには、言語環境プログラムが必要です。

デバッグ能力はデバッグ・ツールによって提供されます。デバッグ・ツールは、以前の COBOL デバッグ・ツールと比較してデバッグ機能が大幅に改善されており、言語環境プログラムのもとで実行される Enterprise COBOL プログラム、IBM COBOL プログラム、VS COBOL II プログラム、およびその他の言語環境プログラム準拠言語プログラム (PL/I および C/C++ を含む) のデバッグに使用することができます。

デバッグ・ツールは、コンパイラーの全機能バージョンと共に組み込まれています。

図 1 は、IBM の以前の COBOL プロダクトと Enterprise COBOL、言語環境プログラム、およびデバッグ・ツールとの関係を示しています。

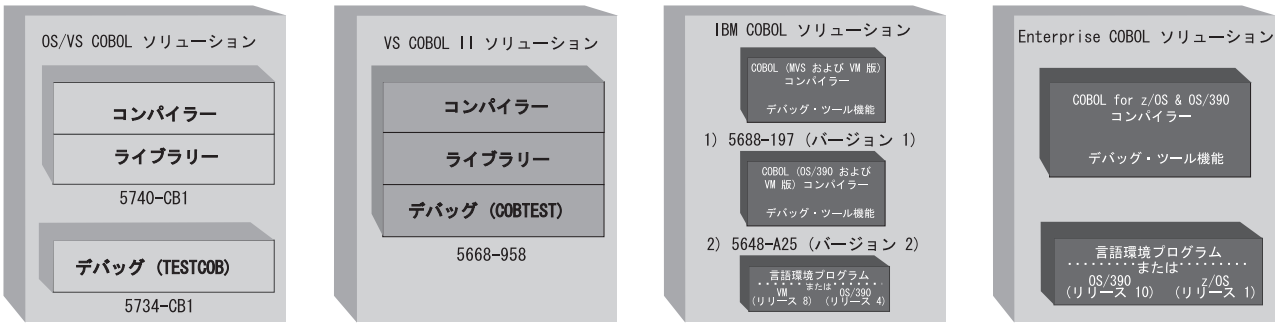


図 1. プロダクトの関係 — コンパイラー、ランタイム、およびデバッグ

各種の COBOL コンパイラーの比較

表 5 に、OS/VS COBOL、VS COBOL II、COBOL (MVS および VM 版) および COBOL (OS/390 および VM 版) の最新リリースでサポートされる機能の概要と、Enterprise COBOL コンパイラーでサポートされる新機能を示します。

表 5. 各種の COBOL コンパイラーの比較

OS/VS COBOL	VS COBOL II	COBOL (MVS および VM 版)	COBOL (OS/390 および VM 版)	Enterprise COBOL for z/OS and OS/390
				Java とのインターオペラビリティー、XML サポート、組み込みの CICS トランスレーター、マルチスレッド化サポート、Unicode

表 5. 各種の COBOL コンパイラーの比較 (続き)

OS/VS COBOL	**VS COBOL II**	**COBOL (MVS および VM 版)**	**COBOL (OS/390 および VM 版)**	**Enterprise COBOL for z/OS and OS/390**
			以下のサポート: DLL 31 桁 DB2 coprocessor OS/390 UNIX 以下の拡張サポート: デバッグ・ツール	以下のサポート: DLL 31 桁 DB2 coprocessor OS/390 UNIX 以下の拡張サポート: デバッグ・ツール
		以下のための拡張: オブジェクト指向 COBOL、 C 言語とのインターオペラビリティー、 組み込み関数、 85 標準への改訂、 以下のサポート: 言語環境プログラム デバッグ・ツール	以下のための拡張: オブジェクト指向 COBOL、 C 言語とのインターオペラビリティー、 組み込み関数、 85 標準への改訂、 以下のサポート: 言語環境プログラム デバッグ・ツール	以下のための拡張: C 言語とのインターオペラビリティー、 組み込み関数、 85 標準への改訂、 以下のサポート: 言語環境プログラム デバッグ・ツール
	COBOL 85 標準、組み込み関数非サポート、構造化プログラミング、DBCS の各国語、改善された CICS インターフェース、31 ビット・アドレッシング、再入可能性、Fast Sort Optimizer、SAA フラグ設定、対話式デバッグ (フルスクリーン・モード)	COBOL 85 標準、構造化プログラミング、DBCS の各国語、改善された CICS インターフェース、31 ビット・アドレッシング、再入可能性、Fast Sort Optimizer、SAA フラグ設定、対話式デバッグ (フルスクリーン・モード)	COBOL 85 標準、構造化プログラミング、DBCS の各国語、改善された CICS インターフェース、31 ビット・アドレッシング、再入可能性、Fast Sort Optimizer、SAA フラグ設定、対話式デバッグ (フルスクリーン・モード)	COBOL 85 標準、構造化プログラミング、DBCS の各国語、改善された CICS インターフェース、31 ビット・アドレッシング、再入可能性、Fast Sort Optimizer、SAA フラグ設定、対話式デバッグ (フルスクリーン・モード)
COBOL 74 標準、74 標準の FIPS フラグ設定、動的ロード、バッチ・デバッグ、対話式デバッグ (ライン・モード)	COBOL 74 との互換性、85 標準の FIPS フラグ設定、動的ロード、バッチ・デバッグ、対話式デバッグ (ライン・モード)	COBOL 74 との互換性、85 標準の FIPS フラグ設定、動的ロード、バッチ・デバッグ、対話式デバッグ (ライン・モード)	COBOL 74 との互換性、85 標準の FIPS フラグ設定、動的ロード、バッチ・デバッグ、対話式デバッグ (ライン・モード)	

| 10 ページの表 6 に、それぞれのオペレーティング・システムについてサポートされる言語環境プログラムのリリース・レベルを示します。

表 6. Enterprise COBOL でサポートされる言語環境プログラムのリリース・レベル

Enterprise COBOL コンパイラー	言語環境プログラムのリリース	
COBOL for z/OS and OS/390	z/OS	言語環境プログラム バージョン 1 リリース 1 以上
	OS/390	言語環境プログラム バージョン 2 リリース 10

注: ホストのバージョンおよびリリースの完全なリストについては、言語環境プログラムおよび使用中のコンパイラーの *Licensed Program Specifications* を参照してください。

その他のプログラムのための言語環境プログラムのランタイム・サポート

図 2 は、言語環境プログラムのもとで稼働することができるプログラムを示しています。言語環境プログラムは、現在 OS/VS COBOL および VS COBOL II ライブラリーを用いて稼働しているプログラムをサポートします。言語環境プログラムは、他の高水準言語もサポートします。

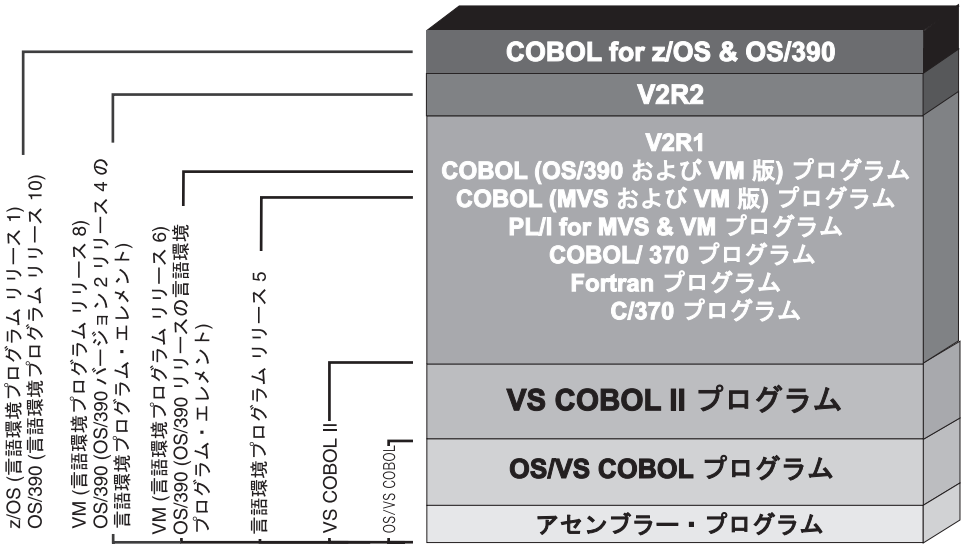


図 2. 他のプログラムのための言語環境プログラムのランタイム・サポート

使用する言語環境プログラムのリリースは、使用中のホスト・システムによって決まります。表 7 に、それぞれのオペレーティング・システムで必要な言語環境プログラムのバージョン、および Enterprise COBOL 用として最低限必要なオペレーティング・システム・レベルを示します。

表 7. 言語環境プログラムのリリース・レベル

ホスト・システム	言語環境プログラムのリリース
OS/390 バージョン 2 リリース 10	OS/390 の言語環境プログラム・エレメント
z/OS バージョン 1 リリース 1 以上	z/OS の言語環境プログラム・エレメント

注: ホストのバージョンおよびリリースの完全なリストについては、言語環境プログラムの *Licensed Program Specifications* を参照してください。

新しいコンパイラーおよびランタイムの利点

Enterprise COBOL コンパイラーおよび言語環境プログラムのランタイムは、OS/VS COBOL、VS COBOL II、および IBM COBOL に対する追加の機能を提供します。表 8に、新しいコンパイラーおよびランタイムの利点を示し、それらが VS COBOL II、OS/VS COBOL、IBM COBOL のいずれに適用されるかについても示します。

表 8. Enterprise COBOL および言語環境プログラムの利点

		適用対象		
利点	**注**	**OS/VS COBOL**	**VS II COBOL**	**IBM COBOL**
Java との相互運用性 (インターオペラビリティー)	Enterprise COBOL はオブジェクト指向 COBOL 構文をサポートするので、COBOL と Java を相互に連動させて運用することができます。	✓	✓	✓
マルチスレッドでの実行のサポート	Enterprise COBOL は、POSIX スレッドおよびシグナルを許容レベルでサポートします。Enterprise COBOL を使用すると、1 つのプロセス内でマルチスレッドで実行される COBOL プログラムをアプリケーションに組み込むことができます。	✓	✓	✓
Unicode サポート	COBOL Unicode サポートは、*OS/390 Unicode* サポート 製品を使用します。	✓	✓	✓
XML サポート	Enterprise COBOL は XML 文書を構文解析するための新しいステートメントをサポートしているので、プログラムでは、XML コンテンツを COBOL のデータ構造体に変換することができます。	✓	✓	✓
組み込みの CICS トランスレーター	Enterprise COBOL コンパイラーは、ソース・プログラム内の COBOL 固有のステートメントおよび CICS 組み込みステートメントの両方を処理します。	✓	✓	*
使用可能度に関する機能強化	以下の機能強化: • COMP-5 項目または TRUNC(BIN) を指定した BINARY 項目での VALUE 文節における大きなリテラル • 関数ポインター・データ型 • ADDRESS OF を指定した引き数	✓	✓	✓
COBOL 言語の向上	組み込み関数を使用して、COBOL で数学関数および金融関数を実行することができます。FORTRAN または C で書かれた現行ルーチンを固有 COBOL コードで置き換えて、アプリケーション論理を単純化することができます。	✓	✓	
境界より上のサポート	仮想記憶制約解放 (VSCR) により、プログラムは 16MB 境界より下または上で常駐したり、コンパイルしたり、プログラムにアクセスしたりすることができます。	✓		
	QSAM バッファーは、DFSMS およびデータ・ストライピングを最適にサポートするために 16MB 境界より上に置くことができます。	✓	✓	
	COBOL 外部データは、境界より上に置くことができます。	✓	✓	

表 8. Enterprise COBOL および言語環境プログラムの利点 (続き)

		適用対象		
利点	**注**	**OS/VS COBOL**	**VS II COBOL**	**IBM COBOL**
31 桁のサポート	Enterprise COBOL では、ARITH(EXTEND) オプションを使用したときは最大 31 桁の数がサポートされるようになりました。	✓	✓	*
OS/390 UNIX システム・サービスのサポート	cob2 コマンドを使用して、OS/390 UNIX シェル内の COBOL プログラムをコンパイルおよびリンクすることができます。COBOL プログラムは、POSIX 標準で定義されたほとんどの C 言語機能を呼び出すことができます。	✓	✓	
戦略的 IBM コンパイラー	z/OS プラットフォームに関する将来のパフォーマンス向上や言語拡張は、Enterprise COBOL および言語環境プログラムでのみサポートされます。	✓	✓	✓
エラー・リカバリー・オプション	プログラマーは、プログラム割り込み、異常終了、およびその他のソフトウェア生成条件をエラー・リカバリーのために代行受信する、アプリケーション固有のエラー処理ルーチンを持つことができるようになりました。これは、Enterprise COBOL プログラムを言語環境プログラム呼び出し可能サービスと共に使用して、ユーザー作成条件ハンドラーを登録することによって行われます。言語環境プログラムは、すべての条件管理を処理します。	✓	✓	
コストの節減	インストール先で複数の言語を使用する場合、複数の言語ランタイムを単一の言語環境プログラム・ランタイムに置き換えると、コストを節減することができます。IBM 担当員と相談して、インストール先で使用している現在のライセンスおよび言語の数に基づいて可能なコストの節減を評価してください。	✓	✓	
高精度の数学ルーチン	言語環境プログラム呼び出し可能サービスを使用すると、プログラムは最も正確な結果を戻すことができます。	✓	✓	
複数の MVS タスクのサポート	RES アプリケーションは、複数の MVS タスクのもとで独立して実行できるようになりました。(たとえば、2 つの Enterprise COBOL プログラムを ISPF 分割画面から同時に実行します。)	✓	✓	

表 8. Enterprise COBOL および言語環境プログラムの利点 (続き)

		適用対象		
利点	**注**	**OS/VS COBOL**	**VS II COBOL**	**IBM COBOL**
パフォーマンス	より高速の算術計算。	✓		
	より高速の動的および静的 CALL ステートメント。		✓	
	可変長 MOVE の向上したパフォーマンス。		✓	
	言語環境プログラムの CBLPSHPOP ランタイム・オプションを使用して CALL ステートメントの PUSH HANDLE および POP HANDLE を防止する場合の、より高速の CICS パフォーマンス。	✓		
	TRUNC(BIN) でコンパイルされたプログラムの向上したパフォーマンス。COBOL (OS/390 および VM 版) リリース 2 は、TRUNC(BIN) コンパイラー・オプションが使用されたときはさらに効率の良いコードを生成するためのサポートを追加しました。	✓	✓	
向上した ILC	共通の言語環境プログラム・ライブラリーにより、COBOL と他の言語環境プログラム準拠言語との ILC が向上します。たとえば、言語環境プログラムのもとでは、COBOL と他の言語との言語間呼び出しがより高速になります。これは、各 CALL ステートメントのオーバーヘッドが大幅に低減されるからです。さらに、CICS では、EXEC CICS LINK の代わりに CALL ステートメントを使用して PL/I または C プログラムを呼び出すことができます。	✓	✓	
文字操作	16 進数リテラルの使用により、ビットおよび文字操作が向上します。参照変更の使用により、文字操作の柔軟性が向上します。	✓		
トップダウン・モジュラー・プログラム開発	プログラムのネストおよび向上した CALL および COPY 機能を介するトップダウン・モジュラー・プログラム開発のサポート。	✓		
構造化プログラミングのサポート	次のものを介する構造化プログラミング・コーディングのサポート。 • インライン PERFORM ステートメント • CONTINUE プレースホルダー・ステートメント • EVALUATE ステートメント • 明示範囲終了符号 (たとえば、 END-IF、END-PERFORM、 END-READ)	✓		
COBOL 85 標準への適合性	COBOL 85 標準のサポート。	✓		
	COBOL 85 標準の改訂 1 (組み込み関数モジュール) のサポート。	✓	✓	
サブシステムのサポート	IMS、ISPF、DFSORT、DB2 の向上したサポート。	✓		
DB2 機能の改善	Enterprise COBOL では、DB2 ストアード・プロシージャーのサポートが組み込まれています。	✓	✓	
	SQL coprocessor のサポート (DB2 バージョン 7 で使用可能)。	✓	✓	*

表 8. Enterprise COBOL および言語環境プログラムの利点 (続き)

		適用対象		
利点	**注**	**OS/VS COBOL**	**VS II COBOL**	**IBM COBOL**
向上した CICS インターフェース	Enterprise COBOL では、CALL ステートメントのサポート (EXEC CICS LINK を使用する場合よりも高速の CICS パフォーマンスのための) が組み込まれており、BLL セルの必要性が除去されます。241 ページの『OS/VS COBOL プログラムについての基底アドレス可能度の考慮事項』を参照してください。	✓		
	DATA(24) および DATA(31) プログラム用の WORKING-STORAGE スペースの増加。DATA(31) の場合、限界は 128M です。DATA(24) の場合、限界は 16MB 境界より下の使用可能スペースです。	✓	✓	
再入可能性のサポート	すべての OS/VS COBOL プログラムは再入不能です。再入可能プログラムのみを共用ストレージ (LPA または共用セグメント) にロードできます。	✓		
デバッグ・ツールのサポート	デバッグ・ツールには以下の利点があります。 • CICS および非 CICS アプリケーションの対話式デバッグ • バッチ・アプリケーションの対話式デバッグ • CICS および非 CICS アプリケーションのフルスクリーン・デバッグ • 同じデバッグ・セッションでの混合している言語のデバッグ • ホストで実行されるプログラムをデバッグする能力 • VisualAge for COBOL と共に作動し、ワークステーションからグラフィカル・ユーザー・インターフェースを使用してホスト・プログラムをデバッグする能力	✓	✓	
	以下は、COBOL (OS/390 および VM 版) 以降のプログラムでのみ使用可能です。 • フックをデバッグせずに COBOL (OS/390 版) プログラムをコンパイルするための動的デバッグ機能。 • TEST コンパイラー・オプションに追加された SEPARATE サブオプション。このオプションを指定すると、COBOL プログラムのデバッグ時にデバッグ・ツールが使用する独立したデバッグ・ファイルが生成されます。	✓	✓	

表 8. Enterprise COBOL および言語環境プログラムの利点 (続き)

		適用対象		
利点	**注**	**OS/VS COBOL**	**VS II COBOL**	**IBM COBOL**
ランタイム・オプション	ABTERMENC および TERMTHDACT — エラー動作を制御するために使用できます。	✓	✓	
	CBLQDA — QSAM ファイルの動的割り振りを制御するために使用できます。		✓	
	LANGUAGE — エラー・メッセージの言語を変更するために使用できます。	✓		
	RPTSTG — ストレージ使用報告書を入手するために使用できます。	✓		
	ストレージ・オプション — ストレージが取得される場所および使用されるストレージの量を制御するために使用できます。	✓	✓	
コンパイラー・オプション	以下のコンパイラー・オプションは、Enterprise COBOL プログラムでのみ使用可能です。 • CICS — 組み込みの CICS トランスレーター機能を使用可能にし、CICS オプションを指定します。NOCICS がデフォルトです。 • CODEPAGE — 実行時に英数字データ項目および DBCS データ項目からなるコンテンツのエンコードに使用されるコード・ページ、および COBOL ソース・プログラム内の英数字リテラル、国別リテラル、および DBCS リテラルのエンコードに使用されるコード・ページを指定します。 • NSYMBOL(NATIONAL、DBCS) — リテラルおよび PICTURE 文節で使用される N 記号の解釈を制御し、国別処理や DBCS 処理が必要かどうかを指定します。 • THREAD — 複数の POSIX スレッドまたは PL/I タスクを含む言語環境プログラムのエンクレーブで COBOL プログラムを実行できるようにすることを指定します。デフォルトは NOTHREAD です。	✓	✓	✓
	以下のコンパイラー・オプションは、COBOL (OS/390 および VM 版) 以降のプログラムでのみ使用可能です。 • DLL — コンパイラーは、ダイナミック・リンク・ライブラリー (DLL) サポートに使用可能であるオブジェクト・モジュールを生成することができます。 • EXPORTALL — オブジェクト・デックをリンク・エディットして DLL を作成するときに特定の記号を自動的にエクスポートするようにコンパイラーに指示します。	✓	✓	

表 8. Enterprise COBOL および言語環境プログラムの利点 (続き)

		適用対象		
利点	注	OS/VS COBOL	VS II COBOL	IBM COBOL
	以下のコンパイラー・オプションは、COBOL (MVS および VM 版) 以降のプログラムで使用可能です。 • CURRENCY — COBOL プログラム用のデフォルト通貨記号を定義するために使用できます。 • OPTIMIZE(FULL) — 新しいサブオプションである FULL を指定した OPTIMIZE は、オブジェクト・プログラムを最適化し、OS/VS COBOL と VS COBOL II の両方の OPTIMIZE オプションと比較して向上したランタイム・パフォーマンスをもたらします。コンパイラーは、未使用のデータ項目を廃棄し、廃棄されたデータ項目について VALUE 文節のコードを生成しません。 • PGMNAME(COMPAT,LONGUPPER,LONGMIXED) — 長さおよび大文字小文字に関してプログラム名の処理を制御します。 • RMODE(AUTO,24,ANY) — このオプションを使用すると、NORENT プログラムを 16MB 境界より上に置くことができます。	✓	✓	
	Enterprise COBOL は、IBM COBOL および VS COBOL II と同様に、コンパイラー出力をさらに制御するためのコンパイラー・オプションを提供します。 • オブジェクト・コードの生成 • コンパイラーによる仮想記憶域の使用 • リスト、マップ、および診断 • ランタイム・デバッグ情報 • カスタマイズされた予約語リスト • COPY または BASIS ステートメントの処理 • エラー・メッセージのテキスト • エラー・メッセージの言語	✓		

注: * COBOL (OS/390 および VM 版) バージョン 2 リリース 2 に対する新機能として、組み込みの SQL coprocessor、組み込みの CICS トランスレーター、および 31 桁のサポートが追加されました。

段階的移行に関する提案

何らかの理由で Enterprise COBOL および言語環境プログラムにアップグレードできないことがわかった場合でも、それらの障害がなくなったときに移行できるようにするための準備として、段階的ステップを取ることができます。以下に例を示します。

- 言語環境プログラムに移行するための処置を評価します。
- 移行計画および長期スケジュールを開発します。
- マクロ・レベル CICS プログラムをコマンド・レベル・プログラムに変換します。(ガイダンス情報については、 300 ページの『CICS アプリケーション・マイグレーション・エイド』を参照してください。)

- 将来の移行を容易にするために、Enterprise COBOL および言語環境プログラムの要件に基づいてアプリケーションをコーディングします。たとえば、NORES コンパイラー・オプションではなく RES コンパイラー・オプションを指定します。すべての Enterprise COBOL プログラムは RES です。「RES 移行」は「NORES 移行」よりも簡単です。NORES アプリケーションを言語環境プログラムとリンク・エディットすると、アプリケーション内のプログラムがどのようにコーディングされているかによって、異なる動作が生じる可能性があります。詳細については、 121 ページの『第 8 章 アプリケーションと言語環境プログラムとのリンク・エディット』を参照してください。
- C と COBOL 間の ILC を含んでいるアプリケーションを判別します。使用中の COBOL および C が言語環境プログラムでサポートされていることを確認し、*C/370 移行の手引き* に記載されている内容に従って必要な変更を行います。詳細については、 112 ページの『COBOL と C/370』を参照してください。
- PL/I と VS COBOL II 間の ILC を含んでいるアプリケーションを判別します。使用している COBOL および PL/I が言語環境プログラムによってサポートされることを確認し、PL/I マイグレーション・ツールを用いて PL/I プログラムをリンク・エディットします。詳細については、 111 ページの『COBOL と PL/I』を参照してください。

  PL/I マイグレーション・ツールを使用すると、PL/I プログラムを、PL/I ランタイムで実行しながら、段階的にリンク・エディットすることができます。マイグレーション・ツールを用いてリンク・エディットする場合は、言語環境プログラムのもとで実行する前にリンク・エディットする必要はありません。
- すべての COBOL ソース・コードを COBOL 85 標準に変換します。

新しいコンパイラーおよびランタイムに関する変更点

Enterprise COBOL には、除去されたコンパイラー・オプション、デフォルト・コンパイラー・オプションの変更、サポートされない SOM ベースのオブジェクト指向 COBOL、および組み込みの CICS トランスレーターなど、既存の COBOL アプリケーションに影響する部分がいくつかあります。除去または改善されたエレメントを以下に示し、互換性を保つために必要な処置について説明します。

CMPR2 コンパイラー・オプションの非サポート

Enterprise COBOL は CMPR2 コンパイラー・オプションを提供しません。CMPR2 を使用してコンパイルされた既存のプログラムの場合、Enterprise COBOL を使用してコンパイルするには、NOCMPR2 (1985 COBOL 標準) に変換する必要があります。

詳細については、以下の章を参照してください。

- 131 ページの『第 10 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 181 ページの『第 12 章 VS COBOL II ソース・プログラムのアップグレード』
- 191 ページの 『第 14 章 IBM COBOL ソース・プログラムのアップグレード』

FLAGMIG コンパイラー・オプションの非サポート

Enterprise COBOL は FLAGMIG コンパイラー・オプションを提供しません。FLAGMIG には CMPR2 が必要ですが、これは Enterprise COBOL ではサポートさ

れません。Enterprise COBOL にマイグレーションするには、FLAGMIG および CMPR2 をサポートする従来の COBOL コンパイラーを使用して、変換する必要があるステートメントにフラグを設定するか、CCCA を使用してください。

詳細については、以下の章を参照してください。

- 131 ページの『第 10 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 181 ページの『第 12 章 VS COBOL II ソース・プログラムのアップグレード』
- 191 ページの『第 14 章 IBM COBOL ソース・プログラムのアップグレード』

ANALYZE コンパイラー・オプションの非サポート

Enterprise COBOL は、ANALYZE コンパイラー・オプションをサポートしません。IBM COBOL では、ANALYZE オプションを指定することにより、SQL ステートメントや CICS ステートメントを含むプログラムの構文チェックを行うことができました。ANALYZE オプションと ADATA オプションの両方を指定したときは、SYSADATA ファイルを生成して、program understanding tool を使用してこのファイルを分析することができました。しかし、VisualAge COBOL では program understanding tool をサポートしなくなったので、Enterprise COBOL では、SQL コンパイラー・オプションおよび CICS コンパイラー・オプションを介して SQL ステートメントと CICS ステートメントをサポートします。ADATA コンパイラー・オプションを使用すると、似たような SYSADATA ファイルを生成することができ、このファイルは program understanding tool で分析する必要がありません。

SOM ベースのオブジェクト指向 COBOL の非サポート

Enterprise COBOL は SOM ベースのオブジェクト指向 COBOL をサポートしていませんが、COBOL プログラムと Java プログラムの間の相互運用性のためのオブジェクト指向構文をサポートしています。Enterprise COBOL から SOM ベースのオブジェクト指向 COBOL が除去されたことにより、コンパイラー・オプション TYPECHK および IDLGEN も除去されました。その理由は、これらのオプションでは、実行のために SOM ベースのオブジェクト指向 COBOL が必要になるからです。SOM ベースのオブジェクト指向 COBOL を使用しているアプリケーションは、Java ベースのオブジェクト指向 COBOL 構文へアップグレードするために設計し直すか、プロシージャー型 (非オブジェクト指向) の COBOL に設計し直す必要があります。

詳細および互換性に関する考慮事項については、以下を参照してください。

- 191 ページの 『SOM ベースのオブジェクト指向 (OO) COBOL プログラムのアップグレード』

組み込みの CICS トランスレーターのサポート

Enterprise COBOL は組み込みの CICS トランスレーターをサポートしています。このため、Enterprise COBOL コンパイラーでは、ソース・プログラム内の COBOL 固有のステートメントと CICS 組み込みステートメントの両方を処理することができます。CICS Transaction Server バージョン 2 がインストールされている場合は、必要に応じて、分離型の CICS トランスレーターから組み込みの CICS トランスレーターにマイグレーションすることができます。

CICS ステートメントを含む COBOL ソース・プログラムを CICS トランスレーターで処理できるようにするには、CICS コンパイラー・オプションを指定する必要があります。詳細および互換性に関する考慮事項については、以下の章を参照してください。

• 237 ページの『第 17 章 COBOL ソースに関する CICS の移行の考慮事項』

一般的な移行作業

インストール先の要件に応じて、おそらく、以下の一般的な移行作業を 1 つ以上完了する必要があります。

- 戦略を計画する
- 言語環境プログラム・ランタイム・ライブラリーに移行する
- ソースを Enterprise COBOL にアップグレードする
- Enterprise COBOL プログラムを既存のアプリケーションに追加する

戦略を計画する

言語環境プログラム・ランタイム・ライブラリーに移行してソース・プログラムを Enterprise COBOL にアップグレードする前に、移行戦略を作成してください。徹底的戦略は、新しいコンパイラーおよびランタイムへのスムーズな移行に役立ちます。

移行戦略としては、言語環境プログラムに移行し、その後、必要に応じて段階的に既存のアプリケーションを Enterprise COBOL で再コンパイルします。本書では、新しいランタイムに移行するための戦略とソースをアップグレードするための戦略を別々に説明します。詳細については、以下の章を参照してください。

- 25 ページの『第 3 章 言語環境プログラムへの移行の計画』
- 41 ページの『第 4 章 ソース・プログラムのアップグレードの計画』

言語環境プログラム・ランタイムに移行する

既存のロード・モジュールを言語環境プログラムのもとで実行した場合、言語環境プログラムへ移行前のライブラリーを使用した場合と同じ結果を得ることができます。重要な互換性情報については、 59 ページの『第 5 章 言語環境プログラムのもとでの既存のアプリケーションの実行』を参照してください。

OS/VS COBOL ランタイムのもとで稼働しているアプリケーションの移行については、 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』を参照してください。

VS COBOL II ランタイムのもとで稼働しているアプリケーションの移行については、 87 ページの『第 7 章 VS COBOL II ランタイムからの移行』を参照してください。(OS/VS COBOL プログラムを VS COBOL II ランタイムのもとで稼働している場合もあるので、この章の内容は、VS COBOL II ランタイムのもとで稼働している OS/VS COBOL プログラムについて記載している第 5 章の内容と重複しています。)

場合によっては、既存のアプリケーションを言語環境プログラムとリンク・エディットしたり、プログラムを Enterprise COBOL にアップグレードしたりすることが

必要になります。言語環境プログラムとリンク・エディットする必要があるプログラムを判別するには、以下を参照してください。

- 74 ページの『リンク・エディットが必要なプログラムの判別』 (OS/VS COBOL ランタイムのもとで稼働しているプログラムの場合)
- 88 ページの『リンク・エディットが必要なプログラムの判別』 (VS COBOL II ランタイムのもとで稼働しているプログラムの場合)

プログラムを言語環境プログラムとリンク・エディットすると、異なる動作が生じる場合があります。詳細については、121 ページの『第 8 章 アプリケーションと言語環境プログラムとのリンク・エディット』を参照してください。

Enterprise COBOL にアップグレードする必要があるプログラムを判別するには、以下を参照してください。

- 75 ページの『アップグレードが必要なプログラムの判別』 (OS/VS COBOL ランタイムのもとで稼働しているプログラムの場合)
- 90 ページの『アップグレードが必要なプログラムの判別』 (VS COBOL II ランタイムのもとで稼働しているプログラムの場合)
- 191 ページの 『Enterprise COBOL を使用してコンパイルする前にアップグレードが必要なプログラムの判別』 (IBM COBOL プログラムの場合)

ソースを Enterprise COBOL にアップグレードする

ソース・プログラムをアップグレードするために必要な処置は、使用したコンパイラーおよび使用した言語レベルによって異なります。

OS/VS COBOL

LANGLVL(1) または LANGLVL(2) を指定してコンパイルされた OS/VS COBOL プログラムは、COBOL 68 標準または COBOL 74 標準エレメントを含んでいる可能性があります。これらのプログラムを Enterprise COBOL でコンパイルするためには、移行が必要です。この移行を援助する移行ツールを使用してください。詳細については、135 ページの『移行ツールを使用したプログラムの COBOL 85 標準への移行』を参照してください。

VS COBOL II

移行の観点からは、VS COBOL II リリース 4 と Enterprise COBOL 間の言語上の唯一の違いは、新しい予約語が追加されたことです。ただし、NOOO 代替予約語テーブルを選択すると、コンパイラーはオブジェクト指向 COBOL 用の新しい予約語を無視します。予約語 (オブジェクト指向 COBOL 用に予約された予約語を含む) の完全なリストは、275 ページの『付録 B. COBOL 予約語の比較』に示されています。

VS COBOL II リリース 3 からアップグレードする場合は、ANSI 解釈の変更に起因する言語上の 3 つの小さな違いがあります。これらの小さな違いは別として、変更を行わずに Enterprise COBOL でコンパイルして、同じ結果を得ることができます。詳細については、181 ページの『第 12 章 VS COBOL II ソース・プログラムのアップグレード』を参照してください。

VS COBOL II リリース 2 プログラムは、CMPR2 コンパイラー・オプションを指定してコンパイルされたその他の VS COBOL II プログラムと同様に、COBOL 74

標準に合わせてコーディングされています。CMPR2 コンパイラー・オプションは、Enterprise COBOL ではサポートされなくなりました。このため、CMPR2 を使用してコンパイルされた VS COBOL II リリース 1 または 2 のプログラム、および VS COBOL II リリース 3 または 4 のプログラムは、すべてソースの変換を行う必要があります。ソース・プログラムを COBOL 85 標準にアップグレードするには、移行ツールが役立ちます。CMPR2 と NOCMPR2 間の言語の違いの詳細については、199 ページの 『第 16 章 CMPR2 から NOCMPR2 へのマイグレーション』に記載されています。

ソース・プログラムのアップグレードに使用できる移行ツールの詳細については、293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

IBM COBOL

Enterprise COBOL では、SOM ベースのオブジェクト指向 COBOL アプリケーションはサポートされなくなりました。オブジェクト指向 COBOL 構文として、Java ベースのオブジェクト指向プログラミングが新たに採用され、COBOL と Java の間の相互運用が容易になりました。以下のコンパイラー・オプションの実行には SOM ベースのオブジェクト指向 COBOL が必要なので、Enterprise COBOL では、これらのコンパイラー・オプションは除去されました。

- IDLGEN
- TYPECHK

Enterprise COBOL プログラムを既存のアプリケーションに追加する

新しい Enterprise COBOL プログラムを作成し (または既存のプログラムを Enterprise COBOL で再コンパイルし)、それを既存のアプリケーションと一緒に言語環境プログラムのもとで実行することができます。

Enterprise COBOL プログラムを既存のアプリケーションに追加するときは、言語環境プログラムとのリンク・エディットの影響、16MB 境界の上または下でプログラムを実行する場合の制約事項、コンパイラー・オプションの変更の影響、予約語の変更の影響、および Enterprise COBOL でのその他の動作の違いの影響を理解していることが必要です。詳細については、251 ページの『第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加』を参照してください。

第 2 部 移行戦略

第 3 章 言語環境プログラムへの移行の計画

この章では、ランタイム環境を言語環境プログラムに移行するための一般的な戦略を説明します。以下の作業が必要です。作業は、次の順序を参考に行ってください。

1. 言語環境プログラム・ランタイム・ライブラリーに移行するための準備。
2. アプリケーションの目録の作成。
3. 言語環境プログラムを段階的に取り入れる方法の決定。
4. レグレッション・テスト・プロシージャーの設定。
5. 実動使用への切り替え。

重要

- OS/390 の場合、Enterprise COBOL プログラムは、OS/390 バージョン 2 リリース 10 の言語環境プログラム・エレメントでのみ実行することができます。
- z/OS の場合、Enterprise COBOL プログラムは z/OS バージョン 1 リリース 1 以上の言語環境プログラム・エレメントで実行することができます。

言語環境プログラム・ランタイム・ライブラリーに移行するための準備

言語環境プログラムに移行するための準備では、以下の作業を行う必要があります。これらの作業は並行して行うことができます。

- 言語環境プログラムのインストール
- 言語環境プログラムについてのプログラマー教育
- ストレージ要件の評価

言語環境プログラムのインストール

z/OS の場合

z/OS (言語環境プログラム・エレメントを含む) をインストールするには、*z/OS Program Directory* または *ServerPac: Installing Your Order* を参照してください。

OS/390 の場合

OS/390 (言語環境プログラム・エレメントを含む) をインストールするには、*OS/390 Program Directory* または *ServerPac: Installing Your Order* を参照してください。

重要: 言語環境プログラム・ランタイムの結果が言語環境プログラム以前のランタイムの結果と互換性を保つようにするには、デフォルトのランタイム・オプションを変更する必要があります。推奨されるランタイム・オプションのリストについては、59 ページの『推奨されるデフォルト言語環境プログラム・ランタイム・オプションの設定』を参照してください。

ストレージ要件の評価

OS/VS COBOL および VS COBOL II 互換ルーチン、新しい機能、および他の言語のサポートのために、言語環境プログラムのストレージ要件は言語環境プログラム以前の COBOL ライブラリーよりも大きくなります。

DASD ストレージ要件

移行時には、言語環境プログラム・ランタイム用の DASD ストレージと、言語環境プログラム以前のランタイム・ライブラリー用の DASD ストレージが必要です。言語環境プログラムへの移行が完了したら、OS/VS COBOL ランタイム・ライブラリーおよび VS COBOL IIランタイム・ライブラリー用に確保したストレージを解放することができます。

言語環境プログラムが必要とする DASD ストレージの量を判別するには、以下の資料を参照してください。

- z/OS の場合: *z/OS Program Directory*
- OS/390 の場合: *OS/390 Program Directory*

仮想記憶域要件

COBOL プログラムを言語環境プログラムのもとで実行する場合の仮想記憶域要件は、OS/VS COBOL ランタイムと VS COBOL II ランタイムの両方に比べて増加します。CICS アプリケーションと非 CICS アプリケーションのどちらの場合も、増加する量は次のような要因に左右されます。

- 言語環境プログラムのランタイム・ストレージ・オプション (STACK、LIBSTACK、 HEAP、 ANYHEAP、 BELOWHEAP) に使用された値。

  注: 言語環境プログラムの RPTSTG ランタイム・オプションによって生成された情報を、ストレージ・オプションの調整に役立てることができます。詳細については、*言語環境プログラム プログラミング・リファレンス* を参照してください。

- 言語環境プログラムのランタイム・オプション ALL31 に使用された値。
- LPA (リンク・パック域) や ELPA (拡張リンク・パック域) に置かれるランタイム・ルーチンの種類
- さらに、VS COBOL II ランタイムから移行するときは、
  - VS COBOL II ランタイム・オプションの指定
  - 16MB 境界より上での実行のために COBPACK が変更されているかどうか

重要

以下のセクションに示す仮想記憶域データは、特定のハードウェアおよびソフトウェアの構成で、選択された一連のテストを使用してサンプル・プログラムを実行することにより得られたものであり、図示する目的でのみ示しています。ユーザーの環境に適用できる構成でユーザー独自のプログラムを実行して、言語環境プログラムが仮想記憶に与える影響を調べることをお勧めします。

非 CICS の場合の仮想記憶域: 表 9 は、RES を指定してコンパイルされた単純な VS COBOL II プログラムを z/OS および OS/390 で実行したときに使用される仮

想記憶域の量を示しています。各ケースにおいて、ランタイム・ライブラリー・ルーチンは STEPLIB から獲得されました。

表 9. 非 CICS で VS COBOL II RES プログラムが使用した仮想記憶域

ランタイム・ライブラリー	IBM デフォルトの変更	16MB より下の仮想記憶域	16MB より上の仮想記憶域
VS COBOL II リリース 4	なし	276K	4K
VS COBOL II リリース 4	16MB 境界より上での実行のために COBPACK が変更された	32K	192K
OS/390 バージョン 2 リリース 10 の言語環境プログラム・エレメント[1]	なし	284K	2436K
z/OS バージョン 1 リリース 2 の言語環境プログラム・エレメント	なし[2]	88K	2488K
z/OS バージョン 1 リリース 2 の言語環境プログラム・エレメント	ランタイム・オプション ALL31(OFF) および STACK(BELOW)[2]	280K	2356K

注:

1. OS/390 バージョン 2 リリース 10 の言語環境プログラム・エレメントの場合、仮想記憶域に関する情報は、z/OS バージョン 1 リリース 1 の言語環境プログラム・エレメントの場合と同じです。
2. ALL31 と STACK のデフォルト値は、言語環境プログラム バージョン 2 リリース 10 と z/OS バージョン 1 リリース 2 の言語環境プログラム・エレメントでは異なります。

言語環境プログラム・ランタイムと言語環境プログラム以前のランタイムによって使用された仮想記憶域の量を比較する場合、ジョブで使用された仮想記憶域には、LPA や拡張 LPA からアクセスされたランタイム・ルーチンは含まれていないことに注意してください。

CICS の場合の仮想記憶域: 言語環境プログラムは、VS COBOL II ランタイムに比べて、より多くの CICS 動的ストレージ域を使用します。使用される CICS 動的ストレージ域の量およびストレージが CICS UDSA (16MB 境界より下の動的ストレージ域) と CICS EUSDA (16MB 境界より上の動的ストレージ域) のどちらから割り振られるかは、多くの要因に左右されます。

最も重要な要因は次のとおりです。

- 言語環境プログラムのランタイム・オプション ALL31 に使用された値。ALL31(OFF) を CICS でのインストール・デフォルトとして使用すると、言語環境プログラムは境界より下の CICS ユーザー動的ストレージ域 (UDSA) のかなりの量を使用する可能性があります。
- 言語環境プログラムのランタイム・ストレージ・オプション (STACK、LIBSTACK、 HEAP、 ANYHEAP、 BELOWHEAP) に使用された値。
- CICS LINK のネスト・レベル。各 CICS LINK が新しい実行単位を開始します。そのため、ネスト・レベルが深くなればなるほど、言語環境プログラムはより多くのストレージを使用します。

- 使用された COBOL コンパイラー。OS/VS COBOL コンパイラーでコンパイルされたプログラムの実行に使用される CICS 動的ストレージ域は増加しません。OS/VS COBOL プログラムが CICS で実行されるときは、互換性ランタイム・ライブラリー・ルーチンによって実行単位用に確立された環境は OS/VS COBOL をサポートするからです。

言語環境プログラムが VS COBOL II ランタイムに比べてより多くの CICS 動的ストレージ域を使用する理由は、次のとおりです。

1. 言語環境プログラムの場合、実行単位ごとにストレージ管理が行われます。そのため、STACK、LIBSTACK、および別々のヒープが実行単位ごとに割り振られます。VS COBOL II の場合、STACK (SRA) およびヒープはトランザクション・レベルで管理されます。
2. 追加の制御ブロックが、言語環境プログラムの共通コンポーネントによって割り振られます。

次の表に、CICS バージョン 4 での VS COBOL II ランタイムおよび言語環境プログラム・ランタイムのストレージ使用量に関するデータを示します。このデータは、補助トレースをオンにした状態で、ある VS COBOL II プログラムが別の VS COBOL II プログラムに対して EXEC CICS LINK を実行するという、単純なトランザクションにより収集されたものです。使用されたストレージの量は、補助トレース出力の GETMAIN トレース記入項目を調べることにより決定されました。

表 10. CICS アプリケーションの場合のストレージ割り振り

ストレージ割り振り	VS COBOL II ランタイム	言語環境プログラム・ランタイム
トランザクションごとに	2040 バイト トランザクションが TASKDATALOC(BELOW) で定義されている場合、ストレージは 16MB 境界より下です。トランザクションが TASKDATALOC(ANY) で定義されている場合、ストレージは 16MB 境界より上です。 制御ブロックには固定サイズが使用されます。	14148 バイト トランザクションが TASKDATALOC(BELOW) で定義されている場合は、ストレージは 16MB 境界より下です。トランザクションが TASKDATALOC(ANY) で定義されている場合は、ストレージは 16MB 境界より上です。APAR PN85951 が適用されている CICS TS バージョン 1 リリース 3、または CICS Transaction Server では、ストレージは常に 16MB 境界より上に割り振られます。 制御ブロックには固定サイズが使用されます。

表 10. CICS アプリケーションの場合のストレージ割り振り (続き)

ストレージ割り振り	VS COBOL II ランタイム	言語環境プログラム・ランタイム
実行単位ごとに	16MB 境界より下で 740 バイト (GETMAIN 用の 400 バイトと制御ブロック用の 340 バイト) 最初の実行単位では、16MB 境界より上のヒープ・ストレージおよび境界より下のヒープ・ストレージが割り振られます。このストレージは、トランザクション時に作成されるすべての新しい実行単位によって使用されます。必要に応じて追加ストレージが割り振られます。IBM 提供のデフォルトを使用しているときは、最初の実行単位の開始時に、境界より下の 8168 バイトのストレージおよび境界より上の 16352 バイトのストレージが割り振られます。	
実行単位ごとに。 ALL31(ON)		16MB 境界より上の 27344 バイト。これには、制御ブロック用のストレージと、STACK、LIBSTACK、HEAP、および ANYHEAP 用のストレージが含まれます。16MB 境界より下のヒープが必要な場合は、要求時に割り振られます。
実行単位ごとに。 ALL31(OFF) および STACK(4K,4K,BELOW,KEEP)		16MB 境界より上の 12512 バイト。これには、制御ブロック用のストレージと、HEAP および ANYHEAP 用のストレージが含まれます。 16MB 境界より下の 18928 バイト。これには、制御ブロック用のストレージと、LIBSTACK、STACK、および BELOWHEAP 用のストレージが含まれます。

注: このシナリオは、言語環境プログラム バージョン 2 リリース 10 で、ランタイム・オプション STACK、LIBSTACK、HEAP、ANYHEAP、および BELOWHEAP に IBM 提供の値を使用して実行されました。

言語環境プログラムについてのプログラマー教育

言語環境プログラムに移行する前に、言語環境プログラムの機能および言語環境プログラム以前のランタイムと言語環境プログラム・ランタイムとの間の違いを、アプリケーション・プログラマーがよく理解しておく必要があります。

プログラマーは、言語環境プログラムに精通していると、言語環境プログラムへの移行の準備をより的確に行うことができます。たとえば、アプリケーションの目録の作成を援助することができます。また、16 ページの 『段階的移行に関する提案』にリストされている提案に従ってコーディングを行うことができます。

弊社を通じて使用できる Enterprise COBOL および言語環境プログラムの教育については、弊社営業担当員にお尋ねください。言語環境プログラムの資料、ユーザー・グループ (SHARE など)、および Web サイト (www.ibm.com/s390/le) から直接情報を入手することもできます。

アプリケーションの目録の作成

言語環境プログラム・ランタイムへの移行を計画するときは、言語環境プログラムで実行する予定のアプリケーションをすべて網羅した目録を作成する必要があります。この目録には、次のものを組み込んでください。

- 取引先のツール、パッケージ、および製品
- COBOL アプリケーション

Edge Portfolio Analyzer は、既存のロード・モジュールの目録を作成するのに役立ちます。詳しくは、 301 ページの『Edge Portfolio Analyzer』を参照してください。

WebSphere Studio Asset Analyzer for z/OS を使用すると、アプリケーションのコード変更による影響を分析することができます。詳しくは、297 ページの『WebSphere Studio Asset Analyzer』を参照してください。

取引先のツール、パッケージ、および製品

ランタイムを言語環境プログラムに移行する前に、取引先のツール、パッケージ、および製品が言語環境プログラムのもとで稼働するように設計されているかどうかを調べる必要があります。以下のことを確認してください。

- すべてのパッケージが言語環境プログラムのもとで稼働すること (特に、パッケージのソース・コードを入手しない場合)。
- パッケージのソース・コードを入手する場合は、それが Enterprise COBOL と互換性のあるソース・コード (1985 標準レベル COBOL) であること。
- コード生成プログラムが、Enterprise COBOL と互換性のあるソース・コード (1985 標準レベル COBOL) を生成すること。
- 独自の ESPIE または ESTAE を出す開発ツールおよびデバッガーが言語環境プログラムと調和的に機能すること。

言語環境プログラムの場合に使用可能な取引先製品のリストを入手する方法については、 301 ページの『取引先製品』を参照してください。

COBOL アプリケーション

COBOL アプリケーションの目録を作成するときは、言語環境プログラムへの移行に影響を与えるプログラム属性に関する情報を収集する必要があります。この情報は、テストすべきものとその方法、および言語環境プログラムのもとでパフォーマンスに影響を与えるものが含まれます。目録のために、以下のことを判別してください。

アプリケーションの言語環境プログラムへの移行について:

- VS COBOL II でコンパイルされたプログラムと OS/VS COBOL でコンパイルされたプログラム
- RES を指定してコンパイルされたプログラムと NORES を指定してコンパイルされたプログラム
- 使用されているランタイム・オプション (および指定方法)。具体的には、以下のオプションです。
  - MIXRES (VS COBOL II)
  - RTEREUS (VS COBOL II)

  - LIBKEEP (VS COBOL II)
  - FLOW (OS/VS COBOL)
- アセンブラー・プログラムを呼び出すか、またはアセンブラー・プログラムによって呼び出される COBOL プログラム
- 言語間通信 (PL/I、C、または FORTRAN) を使用する COBOL プログラム
- CICS、IMS、DB2、またはその他のサブシステムのもとで使用される COBOL プログラム
- 使用される制御ステートメント
- 異常終了の頻度およびタイプ

レグレッション・テストについて:

- 必要で、使用可能なテスト・ケース

パフォーマンス測定について:

- 使用されたストレージの量
- 再使用可能 / 共通モジュールの実行の頻度
- プログラム実行時間 (CPU と経過の両方)

複雑度の割り当て

ここでは、言語環境プログラム・ランタイムへの移行時に変更することが必要になる可能性のあるプログラム属性をリストします。以下の説明では、1 つの範囲内のそれぞれの処置は「または」を意味していますが、1 つの属性に対して実際には 1 つ以上の処置が適用される可能性があります。

各プログラム属性には、以下の表の定義に基づいて、複雑度が割り当てられています。

表 11. プログラム属性についての複雑度

複雑度	定義
0 ～ 2	最小限のテストが必要であるか、または 変更なし、かつ、言語環境プログラムとのリンク・ エディットなしで、言語環境プログラムのもとで稼動する
3 ～ 6	中程度のテストが必要であるか、または 中程度の調整が必要であるか、または 言語環境プログラムとのリンク・エディットが必要
7 ～ 10	中程度から高度のテストが必要であるか、または 中程度から高度の調整が必要であるか、または モジュールの書き直しが必要であるか、または 言語環境プログラムのもとで稼働しない

OS/VS COBOL ランタイムから移行する場合

32 ページの表 12 に、特定のプログラム属性の移行についての複雑度の見積もりを示します。

表 12. OS/VS COBOL ランタイムのもとで稼働しているプログラムについての複雑度

プログラム属性	複雑度
NORES を指定してコンパイルされ、OS/VS COBOL とリンク・エディットされたプログラム	0
CICS オンライン	1
ILBOSTP0 を用いてのプログラムのアセンブラー・ドライバーとのリンク・エディット	3
IMS オンライン	5
RES プログラムと NORES プログラムの混合	5
ISPF プログラム	7
アセンブラー・ルーチンによる、LINK SVC を用いての呼び出し	7
CICS (OS/VS COBOL プログラムから動的 CALL ステートメントを使用する場合)	7
通常の保管域規則に従っていないアセンブラー・プログラム。詳細については、303 ページの 『呼び出し元および呼び出し先アセンブラー・プログラムについての要件の判別』を参照してください。	8
OS/VS COBOL プログラムと PL/I プログラム間の ILC	8
OS/VS COBOL プログラムと FORTRAN プログラム間の ILC	8
複数ロード・モジュール・アプリケーション (メイン・モジュールに複数の OS/VS COBOL NORES プログラムが含まれ、Enterprise COBOL や VS COBOL II プログラムは含まれていない場合)	10
QUEUE ランタイム・オプションの使用	10
STAE または SPIE を発行するアセンブラー・プログラム	10
保管域変更のための有効な 31 ビット・アドレスをレジスター 13 に入れていないアセンブラー・ルーチンの呼び出し	10
ILBO ルーチンの内容に基づいてコーディングされたアセンブラー・プログラム	10

詳細については、以下の章および付録を参照してください。

- 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』
- 303 ページの 『付録 D. COBOL とアセンブラーを含んでいるアプリケーション』

VS COBOL II ランタイムから移行する場合

32 ページの表 13 に、特定のプログラム属性の移行についての複雑度の見積もりを示します。

表 13. VS COBOL II ランタイムのもとで稼働しているプログラムについての複雑度

プログラム属性	複雑度
NORES を指定してコンパイルされ、VS COBOL II のもとで稼働している OS/VS COBOL プログラム	0
NORES を指定してコンパイルされ、VS COBOL II のもとで稼働している VS COBOL II プログラム	0
アセンブラー・ドライバーからの IGZERRE のロード	0
IGZEOPT オブジェクト・モジュールの使用 (非 CICS アプリケーションの場合)	1

表 13. VS COBOL II ランタイムのもとで稼働しているプログラムについての複雑度 (続き)

プログラム属性	複雑度
VS COBOL II プログラムと PL/I プログラム間の ILC (PL/I マイグレーション・ツールを使用してプログラムをリンク・エディットする場合)	1
CICS オンライン	2
RTEREUS ランタイム・オプションの動作への依存	2
IGZEOPT オブジェクト・モジュールの使用 (CICS アプリケーションの場合)	2
アセンブラー・ルーチンによる、LOAD および分岐を用いての呼び出し	2
ISPF プログラム	3
IGZETUN オブジェクト・モジュールの使用	3
NORES を用いてのコンパイル、および MIXRES ランタイム・オプションの指定	4
IMS オンライン	4
VS COBOL II プログラムと C/370 プログラム間の ILC	4
VS COBOL II プログラムと PL/I プログラム間の ILC	4
FORTRAN プログラムとの ILC	4
IGZERRE を用いてのプログラムのアセンブラー・ドライバーとのリンク・エディット	4
アセンブラー・ドライバーによる ILBOSTP0 の使用	4
OS/VS COBOL プログラムについての MIXRES ランタイム・オプションの指定 (IGZBRDGE を使用しない場合)	6
BLDL ユーザー出口の使用	8
通常の保管域規則に従っていないアセンブラー・プログラム。詳細については、303 ページの 『呼び出し元および呼び出し先アセンブラー・プログラムについての要件の判別』を参照してください。	8
OS/VS COBOL プログラムと PL/I プログラム間の ILC	8
アセンブラー・プログラムの COBOL プログラムへの LINK (再使用可能環境で稼働する場合)	9
STAE または SPIE を発行するアセンブラー・プログラム	10

詳細については、以下の章および付録を参照してください。

- 87 ページの『第 7 章 VS COBOL II ランタイムからの移行』
- 303 ページの 『付録 D. COBOL とアセンブラーを含んでいるアプリケーション』

移行 / 非移行カテゴリーの設定

既存のロード・モジュールを言語環境プログラムに移行するのに必要な処置が判明したら、プログラムの重要度および実行の頻度を考慮した上で、プログラムを言語環境プログラムへの移行の必要順にリストすることができます。

以下のような、移行をまったく行いたくないプログラムがある場合もあります。

- 言語環境プログラムの PL/I、C、または FORTRAN 部分の使用を必要とするロード・モジュール
- FORTRAN または PL/I との ILC を使用する OS/VS COBOL プログラム

これらのケースでは、アプリケーションを書き直すことができるまで、既存のランタイムに STEPLIB で指定してください。

言語環境プログラムを段階的に実動モードに移す方法の決定

言語環境プログラムを実動モードで使用する準備ができたら、以下のことを行う必要があります。

- 複数言語の移行を処理する方法の決定
- アプリケーションがライブラリーにアクセスする方法の決定

複数言語の移行

ILC を使用する COBOL アプリケーションがある場合は、関係のあるそれぞれの言語を移行してから、そのアプリケーションを言語環境プログラム・ランタイムに移行してください。たとえば、COBOL のみのアプリケーションおよび PL/I のみのアプリケーションを 言語環境プログラムに移行した後で、COBOL-PL/I アプリケーションを言語環境プログラムに移行してください。

注: LNKLST/LPALST には、1 つの言語について 2 つの異なるライブラリーをインストールしないでください。たとえば、言語環境プログラムを COBOL コンポーネントと共に LNKLST/LPALST にインストールする場合は、OS/VS COBOL ライブラリーまたは VS COBOL II ライブラリーが LNKLST/LPALST にインストールされていてはなりません。

言語環境プログラムを LNKLST にインストールした後は、デフォルトでは、すべての COBOL アプリケーションが言語環境プログラムのもとで実行されます。

アプリケーションがライブラリーにアクセスする方法の決定

言語環境プログラムを実動モードに移す場合、2 つの一般的な方式があります。つまり、言語環境プログラムを LNKLST/LPALST に追加するか、または STEPLIB アプローチを使用します。これらのシナリオは、z/OS または OS/390 で稼働するプログラムを対象としています。

LNKLST/LPALST

言語環境プログラムを LNKLST/LPALST に追加すると、言語環境プログラムはすべてのアプリケーションで使用可能になります。言語環境プログラムを LNKLST/LPALST に追加する前に、すべてのアプリケーションが言語環境プログラムのもとで正しく機能することを確認するために、言語環境プログラムを一時的に LNKLST/LPALST にインストールするか、STEPLIB を使用することができます。

実行時にアプリケーションで複数の COBOL ランタイム・ライブラリーを使用可能にしてはなりません。たとえば、LNKLST には、COBOL ランタイム・ライブラリーが 1 つ (言語環境プログラムの SCEERUN など) しかあってはなりません。複数の COBOL ランタイム・ライブラリーがあると、検出が困難なエラーが発生するか、または使用されないロード・ライブラリーが連結中に存在することになりま

す。言語環境プログラムを LNKLST/LPALST に追加するときは、他の COBOL ランタイム・ライブラリー (COBLIB および COB2LIB など) を除去してください。

LNKLST/LPALST への一時的なインストールまたは STEPLIB の使用: 言語環境プログラムを LNKLST/LPALST に一時的にインストールする場合の提案としては、次のものがあります。

- まず、テストまたは開発マシン上で言語環境プログラムを LNKLST/LPALST にインストールします。
- MVS システム・コマンド SETPROG を使用して LNKLST または LPA を一時的に変更します (システムを IPL する必要はありません)。SETPROG コマンドの使用法については、*z/OS MVS* システム・コマンド、SA88-8593 または *OS/390 OS/390 MVS* システム・コマンド、GC88-6592 を参照してください。
- 週末に IPL し、言語環境プログラムを LNKLST/LPALST にインストールします。その週末に、アプリケーションが言語環境プログラムのもとで稼働することを確認します。

注: z/OS および OS/390 の多数のエレメントは言語環境プログラム・ランタイム・ライブラリーに依存していますが、z/OS の場合も OS/390 の場合も、言語環境プログラムを LNKLST にインストールする必要はありません (ただし、言語環境プログラムを z/OS や OS/390 と同じゾーンにインストールする必要があります)。言語環境プログラムを LNKLST に入れないことを選択する場合は、言語環境プログラムを必要とするそれぞれの z/OS PROC または OS/390 PROC で言語環境プログラムを STEPLIB に指定しなければなりません。どのエレメントが言語環境プログラムを必要とするかについては、以下の資料を参照してください。

- *z/OS Program Directory for z/OS Version 1 Release 1* または *OS/390 Program Directory for OS/390 Version 2 Release 10*

STEPLIB

STEPLIB アプローチを使用して言語環境プログラムを段階的に取り入れることを選択することができます。言語環境プログラム・ランタイムに STEPLIB で指定するときには、一度に 1 つの領域 (CICS または IMS)、バッチ (アプリケーションのグループ)、またはユーザー (TSO) を段階的に取り入れます。

STEPLIB の使用は JCL の変更を意味しますが、段階的な移行はすべてのアプリケーションを一度に移行するよりも簡単です。さらに、STEPLIB を使用するときは、プログラムの実行速度は LNKLST/LPALST を通してランタイム・ライブラリーにアクセスするときよりも遅くなり、より多くの仮想記憶域が使用されることに注意してください。

注: 複数のプロセッサーがチャネル間接続によってリンクされている場合は、システム全体を 1 つのプロセッサーとして扱い、サブシステムごとに移行を行う必要があります。初期セットアップ時に、言語環境プログラム・ランタイムに STEPLIB で指定するために JCL を更新することのほかに、CEEDUMP のデフォルト割り振りがインストール先の要件に合わない場合は、CEEDUMP DD を指定することが必要になる可能性もあります (CEEDUMP は、言語環境プログラムがダンプ出力を書き込む場所の DD 名です)。デフォルトの CEEDUMP 宛先については、104 ページの『言語環境プログラムの定様式ダンプ』を参照してください。

STEPLIB および IMS プログラムに関する問題

言語環境プログラム・ランタイムにアクセスするために STEPLIB を IMS/DC でオンラインで使用するときは、プリロードされた言語環境プログラム・ライブラリー・ルーチンは読取専用ストレージにロードされません。アプリケーションがエラーを含んでおり、アプリケーション以外のストレージを上書きすると、プリロードされたランタイム・ルーチンが破壊され、その結果、使用時に異常終了を引き起こす可能性があります。最新表示時には、再入可能としてマークされたこれらのプリロードされたルーチンは、LPA または LNKLST/LPALST からロードされた場合を除き、最新表示されません。したがって、異常終了が再発します。

注: これは、MVS (OS/390)、IMS、および STEPLIB に関する 20 年来の問題であり、ここで言及されているのは、言語環境プログラムへの段階的な移行について提案される STEPLIB アプローチのためです。

以下の方式のいずれかを使用すると、この問題を防ぐことができます。

- 言語環境プログラムを LNKLST/LPALST にインストールする。
- ランタイム・ルーチンをプリロードしない (これはパフォーマンスを低下させます)。

影響を最小限にとどめる方法:

- 言語環境プログラムを可能な限り短時間で証明します (それを迅速に証明すれば、それだけ迅速に LNKLST/LPALST にインストールすることができます)。
- 同じ領域で異常終了しているいくつかの異なるアプリケーションがないか調べます。ある場合には、リカバリー手順に従う必要があることを意味します。

回復方法: 同じ領域でいくつかの異なるアプリケーションが異常終了していることに気付いた場合には、以下の IMS コマンドを使用して領域を停止し、再始動してください。

1. `'/DISPLAY ACTIVE'` を発行することによって、領域番号を判別します。
2. `'/STOP REGION region#'` を発行することによって、領域を停止します。
3. `'/START REGION region-name'` を発行することによって、領域を再始動します。

STEPLIB の例

以下に、STEPLIB 方式を用いて言語環境プログラムを段階的に取り入れる方法の例を示します。中央開発センター (すべてのコンパイルおよびリンクが 1 つの場所で行われる) と個別の実動場所がある組織を想定してください。これは、非常に保守的なアプローチですが、実動アプリケーションに絶対に混乱がないことを必要とする多くのお客様によって使用されています。

1. 中央開発センターで言語環境プログラムおよび Enterprise COBOL を認証します。
   - 現行のランタイムで、収集したデータを用いてテストを実行し、すべての結果を保管します。
   - 言語環境プログラムを STEPLIB の環境にインストールします。これは、未変更のジョブが現行のランタイムで実行されることと、ユーザーが STEPLIB JCL を用いて言語環境プログラム・ランタイム・ライブラリーにアクセスすることによって言語環境プログラム・ランタイムを使用できることを意味します。

NORES アプリケーションに関する注: このセクションは、変更されていない NORES アプリケーションには適用されません。しかし、NORES アプリケーションを変更すると (たとえば、それらを言語環境プログラムとリンク・エディットすることによって)、それらの動作はリンク・エディット前と異なる場合があります。

- STEPLIB の環境を使用して、言語環境プログラム・ランタイムで、収集したデータを用いてテストを実行し、結果を現行のランタイムと比較します。証明のサイクル全体にわたって並行テストを実行して、アプリケーションが言語環境プログラムで稼働するときに現行のランタイムでの稼働時と同じ結果を出すことを確認します。
- 最後に、Enterprise COBOL を用いてテスト・アプリケーションをコンパイルします。言語環境プログラム・ランタイム・ライブラリーを STEPLIB で指定し、証明テストを再実行します。

2. 言語環境プログラムを中央開発センターのシステムにインストールし、テストします。
   - STEPLIB を使用して現行のランタイムにアクセスして、既存のアプリケーションの非移行バージョンの並行テストを実行します。
   - 実動実行に移す前に、すべての新しいアプリケーションを言語環境プログラム・ランタイム環境で実行します。
3. バックアウト戦略を作成します。
   - 言語環境プログラム・ランタイムをバックアウトすることが必要になった場合に備えて、現行のランタイムをインストールするためのプロシージャーを保管します。
4. 1 つの実動場所で言語環境プログラム・ランタイムをインストールします。
   - 引き続き、STEPLIB の環境で現行のランタイムを用いて既存のアプリケーションの非移行バージョンの並行テストを実行します。
   - この実動場所で言語環境プログラム・ランタイムを 1 か月間実行します。
5. すべての実動場所で言語環境プログラム・ランタイムをインストールします。
   - 引き続き、STEPLIB の環境で現行のランタイムを用いて既存のアプリケーションの非移行バージョンの並行テストを実行します (任意)。
   - すべての実動場所で言語環境プログラム・ランタイムを 1 か月間実行します。
   - 1 か月後、現行のランタイム・ライブラリーの内容をすべて削除します。
6. Enterprise COBOL を用いて、新しいアプリケーションまたは変更されたアプリケーションをすべてコンパイルします。

できるだけ大きい作業単位の移行を試みてください。オンライン領域、アプリケーション、または実行単位全体を一度に移行すると、アプリケーションまたは実行単位内のプログラム間の相互作用をテストすることができます。

レグレッション・テスト・プロシージャーの設定

大部分のアプリケーションは、言語環境プログラムのもとで稼働し、既存のランタイムの場合と同じ結果をもたらしますが、コーディング・スタイル、リソースの使用状況、パフォーマンス、異常終了動作、または言語環境プログラムにおける IBM 規則へのより厳密な準拠によって、結果が異なる場合もあります。結果が異なる可能性がある場合の 2 つの例としては、COBOL の動的 CALL ステートメントではなく SVC LINK を使用してアセンブラー・プログラムを使用するアプリケーションや、CICS のもとで OS/VS COBOL プログラムへの COBOL 固有の CALL ステートメントを持つアプリケーションがあります。

コーディング技法の可能な組み合わせは非常に多いため、アプリケーションが言語環境プログラムのもとで稼働し、予期された結果を受け取るかどうかを判別するための唯一の方法は、レグレッション・テスト用のプロシージャーを設定することです。アプリケーションをテスト環境に移し、言語環境プログラムのもとで実行したときに、予期された結果が得られることを確認してください。

レグレッション・テストは、次のものがあるかどうかを確認するために役立ちます。

- COBOL 動的 CALL ステートメントではなく、SVC LINK を使用してアセンブラー・スタブを使用する OS/VS COBOL プログラム。
- OS/VS COBOL プログラムまたは非 COBOL プログラムによってオープンされた、クローズされていないファイル。これは C03 異常終了の原因となります。
- CICS のもとで、動的 CALL ステートメントを使用する OS/VS COBOL プログラム。
- CICS のもとで、CALL ステートメントを用いて OS/VS COBOL プログラムを呼び出す VS COBOL II プログラム。
- 現行のランタイムと言語環境プログラム・ランタイム間のストレージ使用量の違い。
- 現行のランタイムと言語環境プログラム・ランタイム間の CPU 時間の違い。

テスト時には、既存のアプリケーションを現行のランタイムと言語環境プログラム・ランタイムの両方のもとで並行して実行して、結果が同じであることを確認してください。言語環境プログラムとの比較のために、既存のアプリケーションのパフォーマンス測定を行ってください。

プログラムが正しく稼働したら、それを単独でテストし、さらに実行単位内の他のプログラムと一緒にテストしてください。プログラムを各種のデータに対してテストすることによって、すべてのプログラム処理機能を働かせることができます。このことは、予期しない実行の違いが起こらないようにするために役立ちます。

プログラム出力を分析し、結果が正しくない場合は、デバッグ・ツールまたは言語環境プログラムのダンプ出力を使用して、エラーを突き止め、そのエラーを訂正してください。必要な変更をすべて加えたあとで再実行し、必要に応じて、引き続きデバッグを行ってください。

パフォーマンス測定の実行

アプリケーションがテスト環境の言語環境プログラムのもとで稼働したら、パフォーマンス測定を行ってください (特に、時間や応答が重要なアプリケーションについて)。

言語環境プログラムと現行のランタイム環境間のランタイム・パフォーマンスを比較し、パフォーマンスの向上が必要とされるアプリケーションがあれば、そのアプリケーションを識別した後、プログラムの調整およびパフォーマンスの向上のために使用できる方式を調べることができます。たとえば、言語環境プログラムのランタイム・オプションを使用してストレージ値を変更することができます。追加情報については、COBOL Web サイトにあるパフォーマンス情報を参照してください。Web サイト www.ibm.com/software/ad/cobol にアクセスし、「Library」セクションに移動してください。

実動使用への切り替え

テストによって、アプリケーション (あるいは、IMS 領域または TSO で複数のアプリケーションを実行する場合は、アプリケーションのグループ) 全体が予期どおりの結果を受け取ることが示されたら、単位全体を実動使用に移すことができます。ただし、予期しないエラーが発生した場合に備えて、すぐにリカバリーできる状態にしておいてください。

- z/OS および OS/390 のもとでは、最後の生産性チェックポイントから、古いバージョンを代わりに実行します。
- DB2、CICS、および IMS のもとでは、最後のコミット点に戻り、その点から、移行前の COBOL プログラムを使用して処理を継続します。(DB2 の場合、SQL ROLLBACK WORK ステートメントを使用してください。)
- バッチ・アプリケーションの場合は、インストール先のバックアップおよび復元機能を使用して回復を行います。

既存のアプリケーションを言語環境プログラム・ランタイムのもとでの実動使用に移した後、少しの間アプリケーションを監視して、それが正しく機能し続けることを確認してください。その後は、以前のランタイムの場合と同様に信頼して実行することができます。

第 4 章 ソース・プログラムのアップグレードの計画

この章では、ソース・プログラムを Enterprise COBOL にアップグレードするための一般的な戦略を説明します。以下の作業が必要です。作業は、次の順序を参考に行ってください。

1. ソースをアップグレードするための準備。
2. アプリケーションの目録の作成。
3. アプリケーション・プログラムの更新。

古い COBOL コンパイラーのサービス・サポートは中止されるため、最終的にはすべての COBOL ソース・プログラムをアップグレードしなければなりません。ただちにアップグレードする必要があるというわけではありませんが、古いコンパイラーやそれに対するフィックスは、将来的にはサポートされなくなります。サポートされなくなった時点で、「迅速な」マイグレーションが強いられますが、そのタイミングが非常に不都合な場合もあります。

ソース・プログラムをアップグレードする前に、アプリケーションを言語環境プログラムに移行する必要があります。

ソースをアップグレードするための準備

ソースを Enterprise COBOL にアップグレードするための準備では、以下の作業を行う必要があります。これらは同時に行うことができます。

- Enterprise COBOL のインストール
- ストレージ要件の評価
- 使用する移行ツールの決定
- 新しいコンパイラー機能についてのプログラマー教育

Enterprise COBOL のインストール

このコンパイラーをまだインストールしていない場合は、インストールしてください。

- z/OS または OS/390 については、ご使用の製品の*プログラム・ディレクトリー*を参照してください。

ストレージ要件の評価

Enterprise COBOL コンパイラーの大部分は、16MB 境界より上にロードすることができます。さらに、Enterprise COBOL オブジェクト・プログラムは、31 ビット・アドレッシング・モードで実行され、16MB 境界より上に常駐することができます。これにより、16MB 境界より下のストレージが解放されます。解放されたストレージは、16MB 境界より下に常駐しなければならないプログラムまたはデータのために使用することができます。

移行時には、Enterprise COBOL コンパイラー用の DASD ストレージと、現行の COBOL コンパイラー用の DASD ストレージが必要です。移行が完了した時点で、OS/VS COBOL、VS COBOL II、または IBM COBOL のすべてのプログラムを

Enterprise COBOL にアップグレードしてあれば、現行の COBOL コンパイラー用に確保したストレージを解放することができます。

Enterprise COBOL でコンパイルされたソース・コードから生成されるロード・モジュールは、OS/VS COBOL または VS COBOL II でコンパイルされた同じソース・コードから生成されるロード・モジュールよりも、おそらく大きくなります。

使用する移行ツールの決定およびインストール

使用可能な移行ツールを使用すると、アップグレードを非常に簡単なプロシージャーで行うことができます。ソース・プログラムを Enterprise COBOL プログラムにアップグレードするときは、以下の移行ツールが役立ちます。

COBOL 移行ツール (CCCA)
COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA) は、CICS 専用ではありません。CCCA はすべての古い COBOL を Enterprise COBOL に移行します。CCCA は、変更が必要とされるステートメントの報告書、または実際に移行されたプログラム自体を提供します。CCCA はプロダクト番号 5648-B05 です。

OS/VS COBOL の MIGR コンパイラー・オプション
MIGR オプションを指定すると、Enterprise COBOL のもとでコンパイルするために移行が必要なソース・ステートメントがリストされます。

CMPR2、FLAGMIG、および NOCOMPILE コンパイラー・オプション
COBOL の CMPR2、FLAGMIG、NOCOMPILE の各オプションを指定すると、Enterprise COBOL のもとでコンパイルするために移行が必要なソース・ステートメントがリストされます。CMPR2 オプションと FLAGMIG オプションは Enterprise COBOL ではサポートされませんが、古いコンパイラーでこれらのオプションを指定してコンパイルすることで、Enterprise COBOL でコンパイルするために変更が必要なステートメントにフラグを立てることができます。

使用可能なその他の移行ツールには、次のものがあります。

- CICS アプリケーション・マイグレーション・エイド (CAMA) — CICS マクロ・レベル・コードをコマンド・レベル・コードに変換するのに役立ちます。CAMA はプロダクト番号 5695-061 です。
- COBOL 報告書作成プログラム・プリコンパイラー — 報告書作成プログラム・コードを使用し続けるか、または報告書作成プログラム・コードを非報告書作成プログラム・コードに変換することができます。

  COBOL 報告書作成プログラム・プリコンパイラーはプロダクト番号 5798-DYR です。COBOL 報告書作成プログラム・プリコンパイラーのワークステーション・バージョンも VisualAge for COBOL のオプショナル機構として注文可能 (個別に注文可能) です。

これらの移行ツールについては、293 ページの『付録 C. ソース・プログラム用の移行ツール』で詳しく説明されています。

CCCA、CAMA、または COBOL 報告書作成プログラム・プリコンパイラーの使用を計画している場合は、この時点でそれをインストールしてください。インストールの説明については、使用する移行ツールの資料を参照してください。

新しいコンパイラー機能についてのプログラマー教育

移行処置の初期段階で、アプリケーション・プログラマーが Enterprise COBOL の機能を理解し、さらに、Enterprise COBOL、言語環境プログラム、およびデバッグ・ツールと、インストール先で使用するその他のアプリケーション生産性向上ツールとの間の関係および相互依存性について理解しておくことが大切です。

COBOL 68 標準、COBOL 74 標準、および COBOL 85 標準間のソース言語の違いのほかに、プログラマーは言語環境プログラム条件処理および言語環境プログラム呼び出し可能サービスに精通することが必要です。

弊社を通じて使用できる Enterprise COBOL および言語環境プログラムの教育については、弊社営業担当員にお尋ねください。言語環境プログラムの資料またはテクニカル・コンファレンス (GUIDE、SHARE、または IBM Technical Interchange など) から直接に情報を入手することもできます。

プログラマーは、Enterprise COBOL の機能に精通していると、プログラムの目録の作成 (『アプリケーションの目録の作成』で説明します) を援助することができます。

アプリケーションの目録の作成

Enterprise COBOL へのアップグレードを計画するときは、Enterprise COBOL でのコンパイルを意図しているプログラムを含んでいるアプリケーションの広範囲の目録を作成する必要があります。アプリケーションの目録を作成することによって、必要とされる作業を詳細に把握することができます。次のものの目録を作成する必要があります。

- 取引先のツール、パッケージ、および製品
- COBOL アプリケーション

Edge Portfolio Analyzer は、既存のロード・モジュールの目録を作成するのに役立ちます。詳細については、 301 ページの『Edge Portfolio Analyzer』を参照してください。

WebSphere Studio Asset Analyzer for z/OS を使用すると、アプリケーションのコード変更による影響を分析することができます。詳しくは、297 ページの『WebSphere Studio Asset Analyzer』を参照してください。

取引先のツール、パッケージ、および製品の目録の作成

ソースのアップグレードを開始する前に、取引先のツール、パッケージ、および製品が Enterprise COBOL と一緒に作動するように設計されているかどうかを調べる必要があります。以下のことを確認してください。

- COBOL コード生成プログラムが、Enterprise COBOL でコンパイルできる COBOL 85 標準プログラムを生成すること。
- COBOL パッケージが、Enterprise COBOL でコンパイルできる COBOL 85 標準言語で書かれていること。

COBOL アプリケーションの目録の作成

COBOL アプリケーション内のプログラムごとに、少なくとも以下の情報を目録に組み込んでください。

IBM COBOL、VS COBOL II、および OS/VS COBOL の場合:

- 担当プログラマー
- ソース・プログラムの COBOL 標準レベル (68、74、85)
- 使用したコンパイラー (ANS COBOL V4、OS/VS COBOL、VS COBOL II、IBM COBOL)
- 使用したコンパイラー・オプション (特に CMPR2)
- 使用されたプリコンパイラー・オプション
- 使用された後処理オプション
- COBOL モジュール
- COBOL プログラム内で使用されている COPY ライブラリー・メンバー
- 呼び出し先サブプログラム
- 呼び出しプログラム
- 実行の頻度
- 必要で、使用可能なテスト・ケース
- 報告書作成プログラム・ステートメントを含んでいるプログラム

OS/VS COBOL の場合のみ:

- Enterprise COBOL によってサポートされない以下の機能を使用するプログラムを判別してください。
  - ISAM ファイル
  - BDAM ファイル
  - 通信機能
- 他のプロダクトの購入を必要とする可能性がある以下の機能を使用するプログラムを判別してください。
  - 報告書作成プログラム・プリコンパイラーを必要とする報告書作成プログラム・ステートメント
- Enterprise COBOL のもとでは結果が異なる可能性がある以下の機能を使用するプログラムを判別してください。
  - 可変長データ項目 (OCCURS DEPENDING ON)
  - 浮動小数点数値項目
  - 指数
  - 簡略複合比較条件
  - レジスター 13 の高位ビットを使用するアセンブラー・ルーチン

この情報は、計画作業の次のステップ (『アプリケーションの優先順位付け』) で役立ちます。

アプリケーションの優先順位付け

完成した目録を使用して、移行処置を優先順位付けすることができます。

1. 完成した目録内の各項目に複雑度を割り当て、各プログラムまたはアプリケーションの総合的な複雑度を決定します。
2. 各プログラムまたはアプリケーションの移行優先順位を決定します。

複雑度の割り当て

複雑度は、構成またはプログラムを移行、テスト、および調整するのに必要な処置に基づいて定義されています。45 ページの表 14 で使用されている複雑度は、以下のように定義されています。

0	すべてのコードが CCCA によってエラーなしで移行され、Enterprise COBOL のもとで正しくコンパイルされる
1 〜 3	中程度のテストが必要 中程度の調整が必要 大部分のコードが CCCA によってエラーなしで移行される
4	CCCA と、おそらく手動移行が必要 特殊なテスト考慮事項が必要
5 〜 6	中程度から高度の調整が必要 機能の等価性を調べるための中程度から高度のテストが必要 CCCA のほかに移行が必要 (手動または自動)
7 〜 8	高度の調整が必要 機能の等価性を調べるための高度のテストが必要
9	非常に高度の調整が必要 機能の等価性を調べるための非常に高度のテストが必要
10	モジュールの書き直しが必要

上記の複雑度 (またはユーザーが独自に定義した複雑度) に基づいて、プログラム内の各属性に複雑度を割り当てることができます。示された複雑度の中で最も高いものを、そのプログラム全体の複雑度として使用してください。アプリケーションの場合は、そのアプリケーション内で最も高い複雑度を割り当てられたプログラムの複雑度が、アプリケーション全体の複雑度になります。

表 14 に、特定のプログラム属性の移行についての複雑度の見積もりを示します。

表 14. プログラム属性の移行についての複雑度

プログラム属性	属性の説明	複雑度		
ソース・コードの行数	1000 以下	0		
	5000 〜 10,000	3		
	10,000 〜 20,000 以上	5		
固定ファイル属性の不一致 (FS 39)[1]		4		
CMPR2 を指定して VS COBOL II 以上でコンパイルされたプログラム	コンパイラー・オプション CMPR2 はサポートされない	1		C
COBOL 74 標準の COPY ライブラリー・メンバー		1	M	C
ANS COBOL V4 の COPY ライブラリー・メンバー	1 〜 10	2	M	C
	10 〜 20	5	M	C
	20 以上	6	M	C
安定度	変更の計画がないプログラム	0		
	年 2 回変更されるプログラム	3		
	毎月またはより頻繁に変更されるプログラム	8+		

表 14. プログラム属性の移行についての複雑度 (続き)

プログラム属性	属性の説明	複雑度		
アクセスされるファイルの数	1 ～ 3	1	M	C
	3 ～ 5	2	M	C
	6 以上	3	M	C
ソース・コードなしのモジュール	書き直しが必要なモジュール	10[2]		
	アップグレードの必要がないモジュール	6		
CICS マクロ・レベル・プログラム		10[3]		
完全版 ANS COBOL V4 コンパイラー (以前のコンパイラー) でコンパイルしたプログラム		4		C
OS/VS COBOL リリース 2 コンパイラーによるコンパイル	LANGLVL(2) 手動変更なし	1	M	C
	LANGLVL(1) 手動変更なし	1	M	C
	LANGLVL(2) 手動変更あり	4	M	C
	LANGLVL(1) 手動変更あり	4	M	C
結果が変わる言語の使用	複合 OCCURS DEPENDING ON	4		C
	簡略複合比較条件	6	M	
	浮動小数点演算	6	M	
	指数	6	M	
	符号付きデータ	2		
	バイナリー・データ	2		
使用されるアクセス方式	ISAM	6	M	C
	BDAM	10		C[4]
	TCAM	10		
報告書作成プログラム言語の使用 (報告書作成プログラム・プリコンパイラーを使用しない場合)		6	M	C
報告書作成プログラム言語の使用 (報告書作成プログラム・プリコンパイラーを使用する場合)		0		
CICS		4		

注:

1. 詳細については、 343 ページの『付録 H. QSAM ファイルでのファイル状況 39 の防止』を参照してください。
2. IBM 以外の取引先が、オブジェクト・コードから COBOL ソース・コードを再作成することができます。
3. CICS マクロ・レベル・プログラムをコマンド・レベル・プログラムに変換するのに役立つように、CICS アプリケーション・マイグレーション・エイドを使用することができます。
4. これは部分的な移行です。

M の印が付いているカテゴリーについては、OS/VS COBOL の MIGR オプションを使用して情報を収集することができます。**C** の印が付いているカテゴリーについては、COBOL 移行ツール (CCCA) を使用して情報を収集することができます。

移行優先順位の決定

目録内のそれぞれのプログラムについて複雑度を決定したら、アップグレードする必要のあるプログラムとそれらをアップグレードする順序について、十分な情報に基づく決定を行うことができます。

表 15 に、プログラムの複雑度を移行優先順位に関係付ける 1 つの方式を示します。(最も高い優先順位は「1」で、最も低い優先順位は「6」です。)

表 15. プログラム移行優先順位の割り当て

移行優先順位	複雑度	その他の考慮事項
1	0 ～ 3	組織にとっての重要度が高い。移行ツールを用いての移行処置が小さい。
2	4 ～ 6	組織にとっての重要度が高い。移行ツールを用いての移行処置が中位。
	0 ～ 3	組織にとっての重要度が中位。移行ツールを用いての移行処置が小さい。
3	7 ～ 8	組織にとっての重要度が高い。移行ツールを用いての移行処置が大きい。
	3 ～ 6	組織にとっての重要度が中位。移行ツールを用いての移行処置が中位。
	0 ～ 3	組織にとっての重要度が低い。移行ツールを用いての移行処置が小さい。
4	9 ～ 10	組織にとっての重要度が高い。移行処置が非常に大きい。
	7 ～ 8	組織にとっての重要度が中位。移行ツールを用いての移行処置が大きい。
	3 ～ 6	組織にとっての重要度が低い。移行ツールを用いての移行処置が中位。
5	9 ～ 10	組織にとっての重要度が中位。移行処置が非常に大きい。
	7 ～ 8	組織にとっての重要度が低い。移行ツールを用いての移行処置が大きい。
6	9 ～ 10	組織にとっての重要度が低い。移行処置が非常に大きい。

移行優先順位を決定するときには、以下のことを考慮に入れてください。

- アプリケーションが 16MB 境界より下で使用可能なストレージの限界にある場合は、Enterprise COBOL への移行の第 1 候補となります。z/OS や OS/390 アーキテクチャーでは、仮想記憶域制約から解放されます。

- プログラムが言語環境プログラムのもとで稼働できない場合は、それを移行する必要があります。たとえば、PL/I プログラムを呼び出す OS/VS COBOL プログラムや、PL/I プログラムから呼び出される OS/VS COBOL は、アップグレードする必要があります。

アップグレードする必要がある各プログラムの優先順位と、それらのプログラムをアップグレードするために必要な処置が判明したら、アプリケーションおよびプログラムを移行する順序を決定することができます。

アップグレード / 非アップグレード・カテゴリーの設定

設定した移行優先順位を使用し、プログラムの重要度および実行の頻度を考慮に入れることによって、ほとんどのプログラムを Enterprise COBOL への移行の必要順にリストすることができます。

以下のような、移行をまったく行いたくないプログラムがある場合もあります。

- ソース・コードがないプログラムで、再コンパイルする必要がなく、言語環境プログラムのもとで正しく稼働するもの。
- 組織にとっての重要度が低いプログラムで、言語環境プログラムのもとで正しく稼働し、非常に大きな移行処置を要するもの。
- 実動から段階的に取り除かれているプログラム。

ただし、アップグレードされたプログラムと混合された既存のモジュールの実行については、制限が課せられる場合があることに注意してください。 251 ページの『第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加』を参照してください。

移行手順の設定

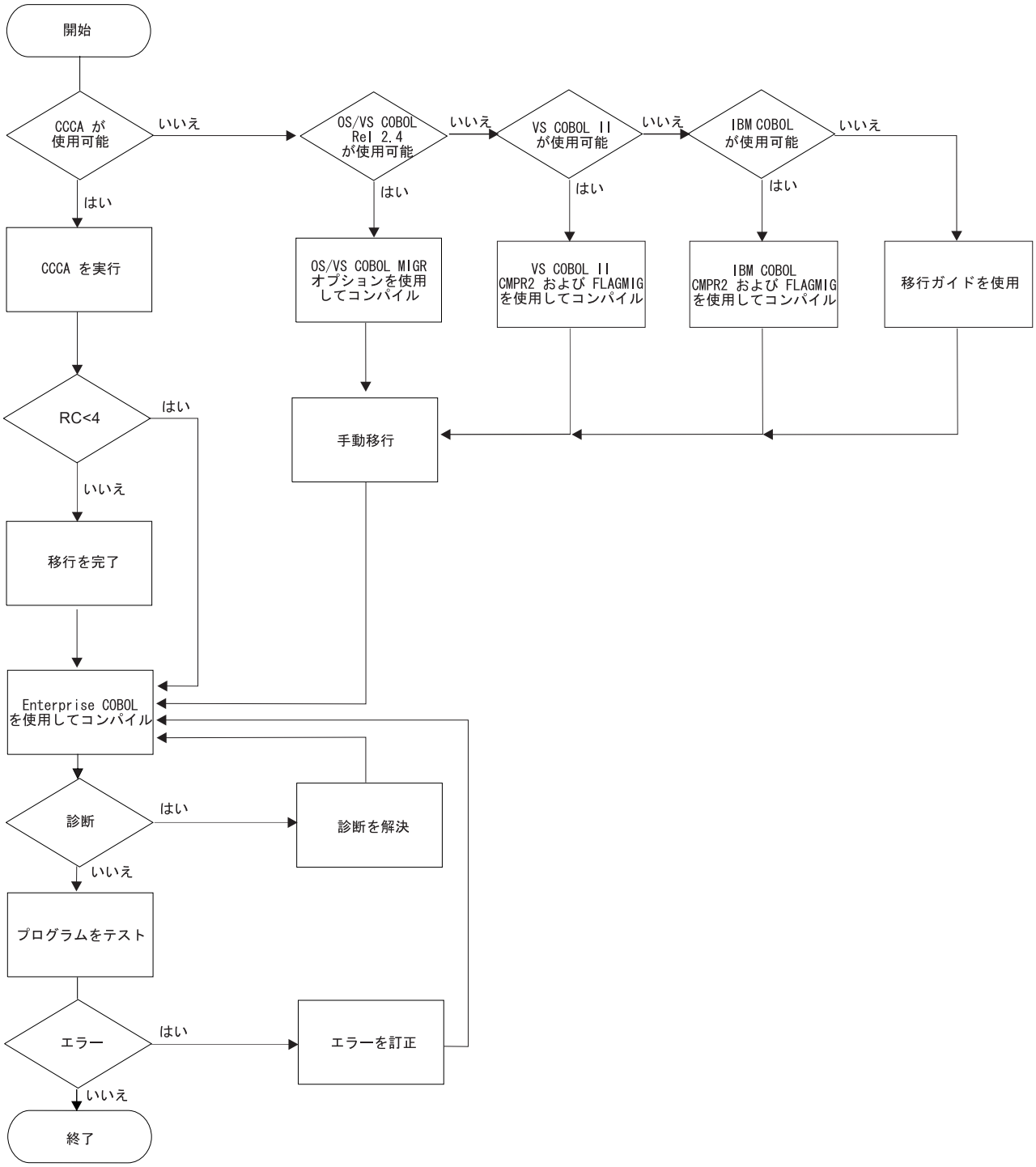
以下のページの要約および図では、5 つのタイプのプログラムをアップグレードするのに必要なステップの概要を示します。

- CICS も報告書作成プログラムも使用しないプログラム
- 構造化プログラミング・コードに変換されるプログラム
- CICS を使用するプログラム
- 廃棄される報告書作成プログラム・ステートメントを含んでいるプログラム
- 保持される報告書作成プログラム・ステートメントを含んでいるプログラム

以下のフローチャートでは、CCCA を使用しない場合は手動でプログラムをアップグレードするように指示されます。CCCA を使用しない場合は、手動の移行を行う前に、IBM 以外の取引先の移行ツールを使用することを検討してください。

CICS も報告書作成プログラムも使用しないプログラム

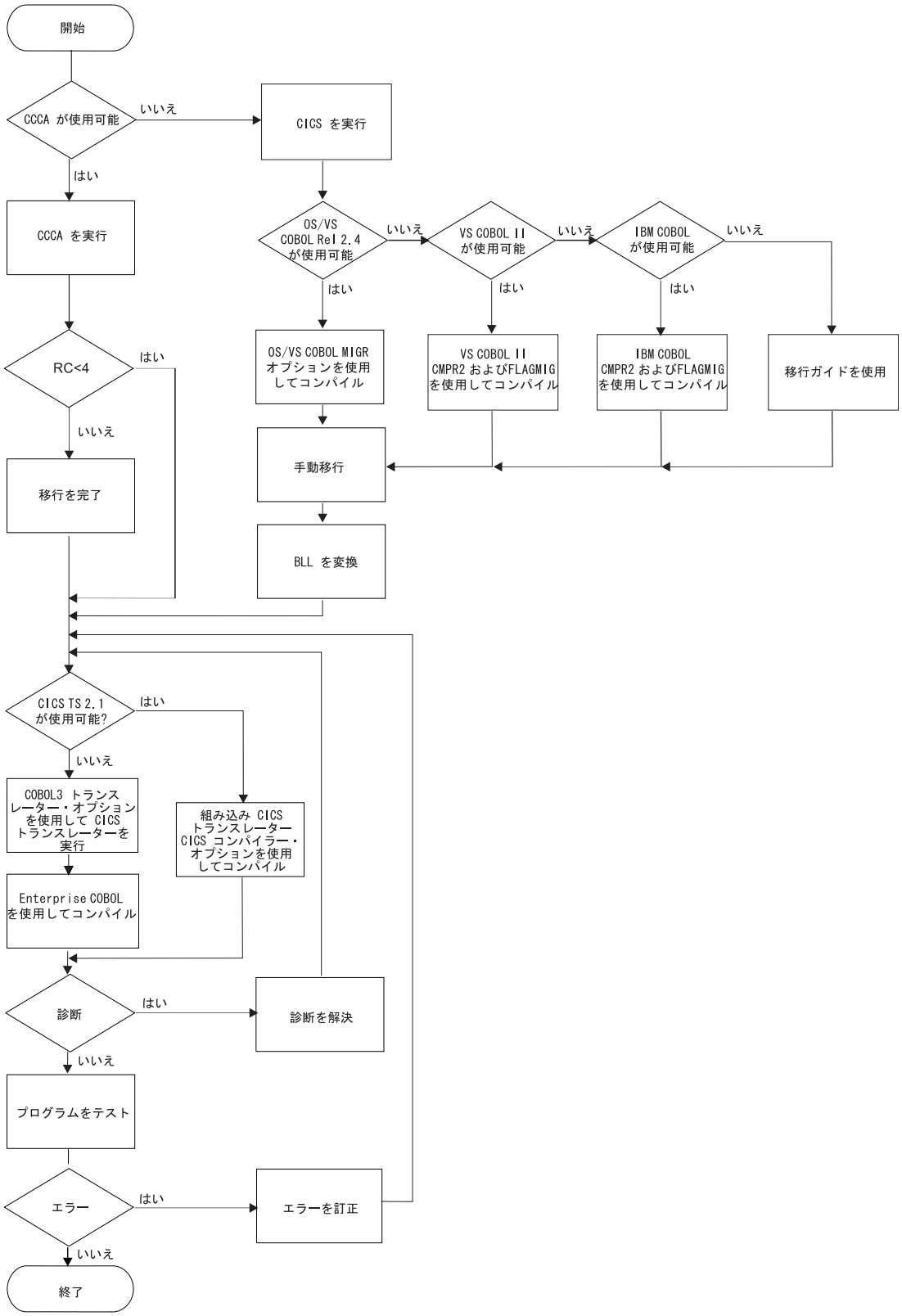
CICS コマンドも報告書作成プログラム・ステートメントも含んでいない OS/VS COBOL プログラムを Enterprise COBOL プログラムに移行するには、以下のステップに従ってください。



CICS を使用するプログラム

CICS コマンドが含まれている OS/VS COBOL プログラムを Enterprise COBOL プログラムに移行するには、以下のステップに従ってください。

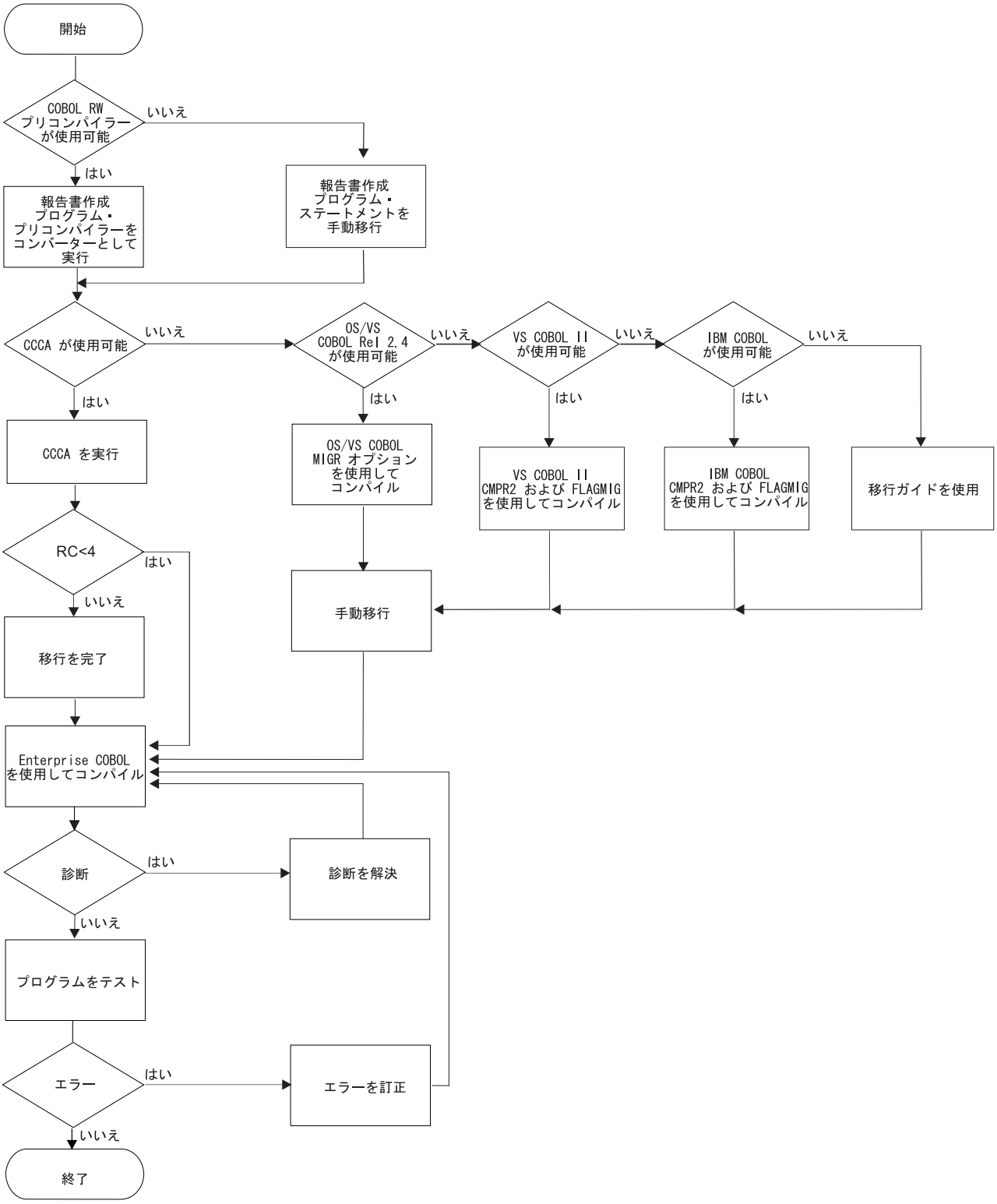
ソースのアップグレードの計画



廃棄される報告書作成プログラムのステートメントを含んでいるプログラム

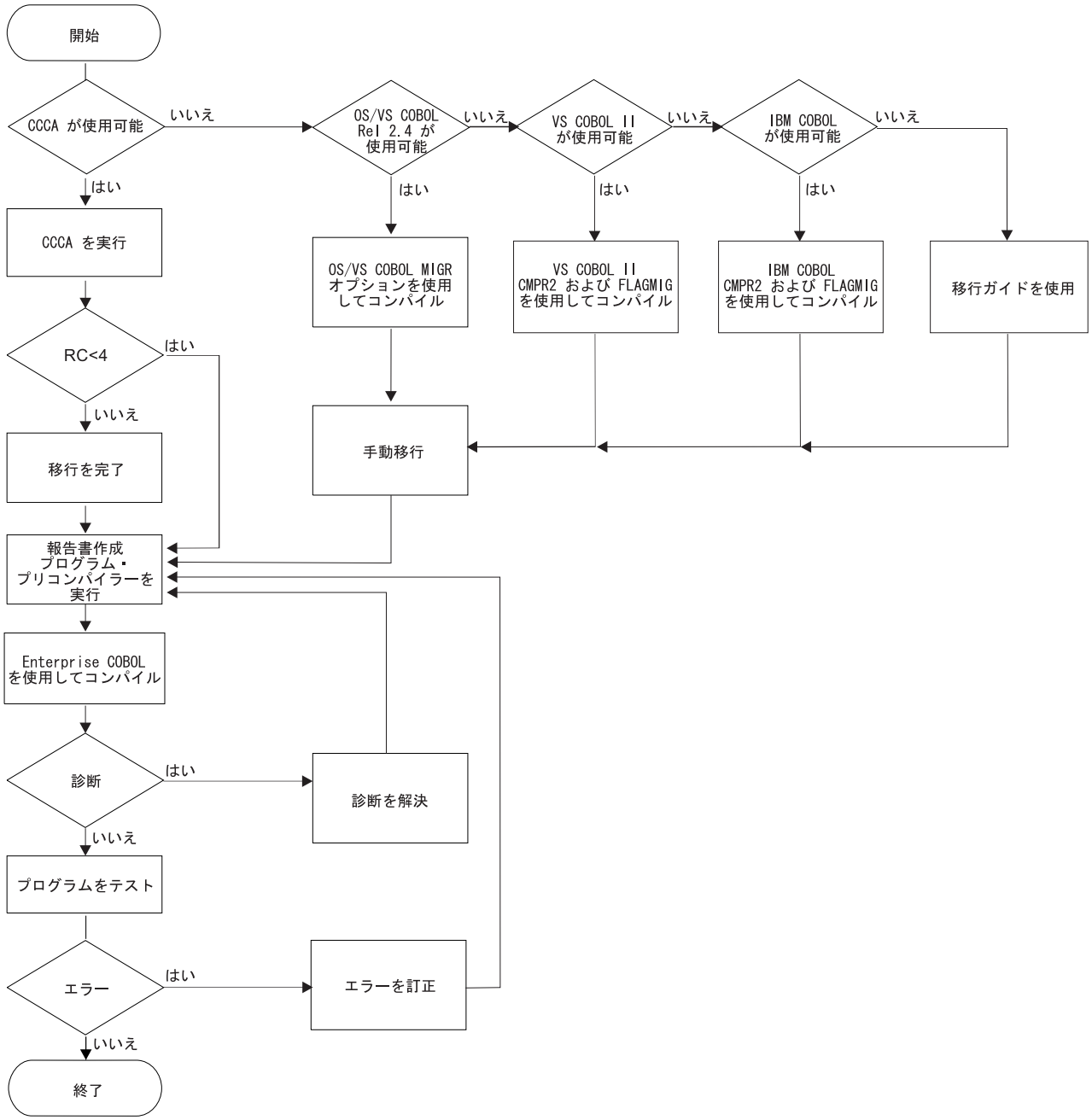
報告書作成プログラム・ステートメントを含んでいる OS/VS COBOL プログラムを Enterprise COBOL プログラムに移行し、すべての報告書作成プログラム・ステートメントを除去するには、以下のステップに従ってください。

ソースのアップグレードの計画



保持される報告書作成プログラムのステートメントを含んでいるプログラム

報告書作成プログラム・ステートメントを含んでいる OS/VS COBOL プログラムを Enterprise COBOL プログラムに移行し、ソース・コード内の報告書作成プログラム・ステートメントを保持するには、以下のステップに従ってください。



アプリケーション・プログラムの更新

ソースをアップグレードするときは、以下のアプリケーション・プログラミング作業が必要です。これらの作業は、だいたい以下の順序で行う必要があります。

既存のソースは、移行済みモジュールに問題がある場合のバックアップ (比較のためのベンチマークおよび回復のためのバージョン) として保管しておいてください。

1. ジョブおよびモジュールの文書の更新。

   すべての更新を適切に文書化することがきわめて重要になります。COBOL は、それ自体が適切に自己を文書化しています。しかし、指定するコンパイラー・オプションや、それらを指定する理由の記録を保管しておいてください。さらに、システムの特殊な考慮事項も文書化してください。これは、反復プロセスであり、移行プログラミング作業全体にわたって行う必要があります。

2. 使用可能なソース・コードの更新。

   可能であれば、 293 ページの『付録 C. ソース・プログラム用の移行ツール』に記述されている移行ツールを使用してください。移行ツールを使用しない場合は、ソース・コードを手動で更新してください。

3. コンパイル、リンク・エディット、および実行。

   ソースの更新が終わったプログラムは、新しく作成された Enterprise COBOL プログラムと同様に処理することができます (言語環境プログラム・ランタイムがインストールされている必要があります)。

4. デバッグ。

   プログラム出力を分析し、結果が正しくない場合は、デバッグ・ツールまたは言語環境プログラムのダンプ出力を使用して、エラーを突き止めてください。

5. 移行済みプログラムのテスト。

   ソースを Enterprise COBOL にアップグレードした後、レグレッション・テストのプロシージャーを設定してください。レグレッション・テストは、次のものがあるかどうかを確認するために役立ちます。

   - 固定ファイル属性の不一致 (ファイル状況 39 問題)。COBOL レコード記述、JCL DD ステートメント、および物理ファイル属性が一致していることを確認してください。詳細については、 343 ページの『付録 H. QSAM ファイルでのファイル状況 39 の防止』を参照してください。
   - 2 進ゼロに初期設定される WORKING-STORAGE への依存性。WORKING-STORAGE ゼロの依存性がある場合は、言語環境プログラムの STORAGE(00) ランタイム・オプションを指定してください。
   - パフォーマンスの違い。
   - 符号処理問題 — S0C7 異常終了。データの符号は、指定する NUMPROC コンパイラー・オプションのサブオプションによって許可される符号と一致しなければなりません。
   - DATA(24) 問題。AMODE 24 プログラムを 31 ビット・データと混合しないでください。

   レグレッション・テスト・プロシージャーを確立し、プログラムが正しく稼働したら、プログラムを各種のデータに対してテストしてください。

- ローカルに (各プログラムを別々に)。
- グローバルに (実行単位内のプログラムを相互に作用させて)。

このようにすれば、すべてのプログラム処理機能を働かせることができ、このことは、予期しない実行の違いが起こらないようにするために役立ちます。

6. 繰り返し (必要に応じて)。

   さらに必要な修正を加えた後で再コンパイル、再リンク、再実行し、必要に応じてデバッグを継続してください。

7. 実動モードへの切り替え。

   テストによって、アプリケーション全体が予期どおりの結果を受け取ることが示されたら、単位全体を実動モードに移すことができます。(これには、実動システムですでに言語環境プログラム・ランタイムを使用していることが前提となります。まだ使用していない場合は、言語環境プログラム・ランタイムに STEPLIB で指定してください。 34 ページの『言語環境プログラムを段階的に実動モードに移す方法の決定』を参照してください。)

   予期しないエラーが発生した場合に備えて、すぐにリカバリーできる状態にしておいてください。

   - z/OS または OS/390 のもとでは、最後の生産性チェックポイントから、古いバージョンを代わりに実行します。
   - DB2 および IMS のもとでは、最後のコミット点に戻り、その点から、移行前の COBOL プログラムを使用して処理を継続します。(DB2 の場合、SQL ROLLBACK WORK ステートメントを使用してください。)
   - 非 CICS アプリケーションの場合は、インストール先のバックアップおよび復元機能を使用して回復を行います。

8. 実動モードでの実行。

   切り替えの後、少しの間アプリケーションを監視して、予期どおりの結果が得られることを確認してください。これで、ソース移行作業が完了します。

第 3 部 既存アプリケーションの言語環境プログラムへの移行

第 5 章 言語環境プログラムのもとでの既存のアプリケーションの実行

アプリケーションの特性によっては、アプリケーションに変更を加え、以下の言語環境プログラム・カスタマイズ作業を行って、現行のアプリケーションが言語環境プログラムのもとで稼働するようにする必要もあります。

- 推奨されるデフォルト言語環境プログラム・ランタイム・オプションの設定。
- 既存のアプリケーションの呼び出し。
- 既存のアプリケーションのリンク・エディット。
- システム・ダンプまたは CICS トランザクション・ダンプの入手。
- 互換性のある異常終了動作の取得。
- 戻りコード値の互換性の確保。

OS/VS COBOL と VS COBOL II のどちらからランタイムを移行するかによって異なりますが、互換性を確保するためにその他の要因も適用されます。詳細については、以下の章を参照してください。

- 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』
- 87 ページの『第 7 章 VS COBOL II ランタイムからの移行』

推奨されるデフォルト言語環境プログラム・ランタイム・オプションの設定

言語環境プログラムの IBM 提供のデフォルト・ランタイム・オプション設定は、OS/VS COBOL または VS COBOL II のランタイム・オプションと同じランタイム動作をもたらさない可能性があります。このような違いのため、このセクションには 2 つの目的があります。第一は、変更する必要があるランタイム・オプション設定を判別できるように、COBOL プログラムについて推奨されるランタイム・オプション設定を示すことです。第二は、COBOL プログラムについて強く推奨されるデフォルト設定を不注意に変更することがないようにすることです。

非 CICS アプリケーションについて推奨されるランタイム・オプション

表 16 では、既存の非 CICS COBOL アプリケーションについて強く推奨される言語環境プログラム・ランタイム・オプションを記述します。言語環境プログラム・ランタイム・オプションの完全なリストについては、*言語環境プログラム プログラミング・リファレンス* を参照してください。

言語環境プログラムのもとでの実行

表 16. 非 CICS COBOL アプリケーションについて推奨される言語環境プログラム・ランタイム・オプション

オプション	LanEnv のデフォルト設定	推奨される COBOL 設定	説明
ABTERMENC	ABEND	ABEND	ABTERMENC(ABEND) を使用すると、異常終了、プログラム・チェック、または重大エラーが発生した場合に、アプリケーションは異常終了コードを出して終了します。これは、OS/VS COBOL や VS COBOL II で問題が発生した場合に終了するのと同様です。 VS COBOL II ランタイム・メッセージに関する追加の考慮事項については、 103 ページの『異常終了コード』を参照してください。 システム・ダンプを入手するには、 68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。
CBLOPTS	ON	ON	CBLOPTS(ON) を使用すると、既存の COBOL 形式の呼び出し文字ストリングが引き続き機能します (ユーザー・パラメーターの後にランタイム・オプションが続きます)。このオプションは、アプリケーションのメインプログラムが COBOL である場合にのみ有効です。
CBLQDA	OFF	OFF	CBLQDA(OFF) を使用すると、OPEN OUTPUT、OPEN I-O (オプショナル・ファイル)、または OPEN EXTEND (オプショナル・ファイル) ステートメントが QSAM ファイルに対して発行されたときに、使用可能でないファイルの QSAM 動的割り振りが抑制されます。CBLQDA(OFF) の動作は、OS/VS COBOL、VS COBOL II リリース 2、および CMPR2 を指定して VS COBOL II リリース 3 以上でコンパイルされたプログラムと互換性があります。また、IGZEQOC に対して APAR II04562 の ZAP を適用した VS COBOL II ランタイムのもとで実行される VS COBOL II NOCMPR2 プログラムとも互換性があります。CBLQDA(ON) は COBOL 85 標準に準拠します。

表 16. 非 CICS COBOL アプリケーションについて推奨される言語環境プログラム・ランタイム・オプション (続き)

オプション	LanEnv の デフォルト設定	推奨される COBOL 設定	説明
RTEREUS	OFF	OFF	RTEREUS は、インストール先のデフォルトとして推奨されません。RTEREUS を使用する場合、使用は特定のアプリケーションに限ってください。さらに、以下のような、可能性のある副次作用および制限を理解しておかなければなりません。 • CEEPIPI または DB2 ストアード・プロシージャーを使用する場合は、RTEREUS(ON) は無視されます。 • 言語環境プログラムのもとでは、IGZERREO CSECT を使用して動作を変更しない限り、RTEREUS(ON) は単一エンクレーブ環境でのみサポートされます。COBOL の再使用可能環境についての IBM 提供のデフォルト設定の場合、ネストされたエンクレーブを作成しようとするアプリケーションは終了し、エラー・メッセージ IGZ0168S が出されます。ネストされたエンクレーブは、SVC LINK または CMSCALL を使用してアプリケーション・プログラムを呼び出すアプリケーションによって作成される可能性があります。1 つの例としては、ISPF のもとで、ISPF サービス (CALL 'ISPLINK' および ISPF SELECT など) を使用しているアプリケーション・プログラムを呼び出すために SVC LINK が使用されたときです。 • 言語環境プログラムの再使用可能環境を確立した (RTEREUS を用いて) 場合は、言語環境プログラムのもとで C または PL/I メインプログラムを実行しようとしても失敗します。たとえば、RTEREUS(ON) を指定して ISPF で実行する場合、 – ISPF によって呼び出された最初のプログラムは COBOL プログラムです。言語環境プログラムの再使用可能環境が確立されます。 – その他のポイントで、ISPF が PL/I または C プログラムを呼び出します。PL/I または C プログラムの初期設定は失敗します。 • 多数の COBOL プログラムが同じ MVS タスクのもとで実行されると、領域不足の異常終了が起こる可能性があります。これは、COBOL プログラムを実行するために言語環境プログラムによって獲得されたすべてのストレージが、MVS タスクが終了するかまたは言語環境プログラム環境が終了するまで、ストレージ内で保持されるためです。 • エンクレーブを終了させるために STOP RUN が実行されない限り、言語環境プログラムの終了は行われません。その結果、次のようになります。 – 言語環境プログラムのストレージおよびランタイム・オプション報告書が言語環境プログラムによって生成されません。 – COBOL プログラムによってクローズされなかった COBOL ファイルが、言語環境プログラムによってクローズされません。その結果、ファイル内のデータを予測できません (たとえば、レコードが書き込まれなかったり、最後のレコードが 2 回書き込まれたりする可能性があります)。
ANYHEAP BELOWHEAP HEAP LIBSTACK STACK			これらのランタイム・オプションは、ストレージを管理するのに役立ちます。STACK については、アプリケーション内のいずれかのプログラムが AMODE 24 で稼働している場合は、サブオプション BELOW を指定してください (ALL31(OFF) も指定してください)。アプリケーション内のすべてのプログラムが AMODE 31 である場合は、サブオプション ABOVE を指定してください (ALL31(ON) も指定してください)。COBOL では、STACK の推奨設定値は、ALL31(OFF) を使用する場合は 64K、64K、BELOW、KEEP であり、ALL31(ON) を使用する場合は 64K、64K、ANY、KEEP です。

表 16. 非 CICS COBOL アプリケーションについて推奨される言語環境プログラム・ランタイム・オプション (続き)

オプション	LanEnv の デフォルト設定	推奨される COBOL 設定	説明
TERMTHDACT	TRACE	UADUMP または UATRACE	重大エラー (たとえば、プログラム・チェックまたは異常終了) が原因で環境が終了するときに言語環境プログラムのもとでシステム・ダンプを受け取るには、TERMTHDACT(UADUMP)、TERMTHADACT(UATRACE)、または TERMTHDACT(UAONLY) を使用してください。あるいは、異常終了出口を使用してください。詳細については、 68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。
TRAP	ON	ON	TRAP は、言語環境プログラム・ルーチンが異常終了およびプログラム・チェックを処理する方法を指定します。アプリケーションが正常に稼働するためには、TRAP(ON) を指定しなければなりません。さらに、TRAP(ON) を使用すると、言語環境プログラムは VS COBOL II (STAE ランタイム・オプション) と OS/VS COBOL (STATE、FLOW、COUNT、および SYMDMP の各デバッグ・オプション) の両方によって提供される既存の条件処理機構をサポートすることができます。

非 CICS アプリケーションに影響を与えるその他のランタイム・オプション

以下のランタイム・オプションも、既存のアプリケーションが言語環境プログラムのもとで稼働し、予期どおりの結果をもたらすかどうかに影響を与えます。デフォルト設定はそれぞれのインストール先の要件によって異なるため、特定のサブオプションは推奨できません。

ALL31

次のような AMODE 24 プログラムを含んでいるアプリケーションの場合、ALL31(OFF) が必要です。

- OS/VS COBOL プログラム
- VS COBOL II NORES プログラム
- その他の AMODE 24 非 COBOL プログラム

ALL31(ON) を使用すると、外部データを 31 ビット・アドレッシング範囲内のどこにでも割り振ることができ、ランタイム・パフォーマンスが向上します。ALL31 に対する IBM 提供のデフォルト値は、OS/390 言語環境プログラム バージョン 2 リリース 10 の場合は OFF であり、z/OS 言語環境プログラム バージョン 1 リリース 2 の場合は ON です。

MSGFILE

言語環境プログラム・ランタイム・オプション MSGFILE(ddname) を使用してメッセージの宛先を指定するときには、出力メッセージ・ファイルの DD 名に使用できる名前に関する制約事項があります。以下の DD 名を使用しないでください。

SYSABEND	SYSCOUNT	SYSDBOUT	SYSDTERM
SYSIN	SYSLIB	SYSLIN	
SYSPUNCH	SYSUDUMP	SYSLOUT	

IBM 提供のデフォルトは MSGFILE(SYSOUT) です。

STORAGE

2 進ゼロに初期設定される WORKING-STORAGE に依存するプログラムの場合、このオプションを使用して、RENT を指定してコンパイルされたプログラムによって獲得されたストレージを 2 進ゼロに初期設定することができます (VALUE 文節が指定されている場合を除きます)。さらに、このオプションを使用して、プログラムのすべての外部データ・レコードを 2 進ゼロに設定することもできます。ただし、パフォーマンスを向上させるためには、初期設定が必要なデータ項目だけを明示的に初期設定してください。

WSCLEAR および STORAGE(00) は、NORENT を指定してコンパイルされたプログラムには影響を与えません。

言語環境プログラムで VS COBOL II の WSCLEAR ランタイム・オプションと同等の影響を得るには、言語環境プログラムの STORAGE ランタイム・オプションの最初のサブオプションを 00 に設定してください。たとえば、STORAGE(00,NONE,NONE,8K)。

IBM 提供のデフォルトは STORAGE(NONE,NONE,NONE,8K) です。

CICS アプリケーションについて推奨されるランタイム・オプション

表 17 では、既存の COBOL CICS アプリケーションについて強く推奨される言語環境プログラム・ランタイム・オプションを記述します。CICS では、言語環境プログラムのデフォルト設定は非 CICS の場合とは異なります (言語環境プログラムのデフォルト設定の大部分は、推奨される COBOL 設定と同じです)。

言語環境プログラム・ランタイム・オプションの完全なリストについては、*言語環境プログラム プログラミング・リファレンス* を参照してください。

表 17. COBOL CICS アプリケーションについて推奨される言語環境プログラム・ランタイム・オプション

オプション	CICS での LanEnv のデフォルト設定	推奨される COBOL 設定	説明
ABTERMENC	ABEND	ABEND	このオプションを使用すると、異常終了、プログラム・チェック、または重大エラーの発生時に、VS COBOL II または OS/VS COBOL によって出されるのと類似した異常終了コードを受け取ることができます。 ABTERMENC(ABEND) を使用すると、システム異常終了コードを受け取ります。システム・ダンプを入手するには、 68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。

表 17. COBOL CICS アプリケーションについて推奨される言語環境プログラム・ランタイム・オプション (続き)

オプション	CICS での LanEnv のデフォルト設定	推奨される COBOL 設定	説明
ALL31	ON	ON	ALL31(ON) を使用すると、言語環境プログラムは制御ブロックを境界より上に割り振ることができます。ALL31(OFF) を使用すると、言語環境プログラムによる 16MB 境界より下のストレージの使用が増えます。ALL31(OFF) と ALL31(ON) の場合の CICS での言語環境プログラムによるストレージ使用の違いについては、 26 ページの『仮想記憶域要件』を参照してください。 ALL31(ON) は、COBOL アプリケーションについて推奨される設定値です。OS/VS COBOL プログラムを CICS 領域で実行する場合でも、VS COBOL II、IBM COBOL、および Enterprise COBOL のプログラムがすべて AMODE 31 の場合は、ALL31(ON) を使用することができます (OS/VS COBOL およびアセンブラー・プログラムだけを含んでいるロード・モジュールは AMODE 24 であり、ALL31 の設定による影響を受けません)。 **注:** ALL31(ON) で稼働するためには、ロード・モジュール内の各プログラム (アセンブラー・プログラムを含む) が 31 ビット使用可能でなければなりません。 ロード・モジュールが VS COBOL II、IBM COBOL、または Enterprise COBOL のプログラムを含んでいて、AMODE(24) を必要とするアセンブラー・プログラムも含んでいる場合は、ALL31(OFF) を使用してください。
ANYHEAP BELOWHEAP HEAP LIBSTACK STACK	*言語環境プログラム導入およびカストマイズ* を参照してください。	デフォルトの設定値	言語環境プログラムは、ストレージ管理に役立つようにこれらのランタイム・オプションを提供します。
TERMTHDACT	TRACE	UADUMP または UATRACE	重大エラー (プログラム・チェック、異常終了など) が原因で環境が終了する場合に言語環境プログラムのもとでトランザクション・ダンプを受け取るには、TERMTHDACT(UADUMP)、 TERMTHDACT(UATRACE)、または TERMTHDACT(UAONLY) を使用してください。あるいは、異常終了出口を使用してください。詳細については、 68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。 実動では、 TERMTHDACT(DUMP)、TERMTHDACT(TRACE)、 TERMTHDACT(UADUMP)、TERMTHDACT(UATRACE) の使用は避けてください。これらの TERMTHDACT サブオプションを使用すると、トランザクションの異常終了時に言語環境プログラムのダンプ・データを一時キュー・データ CESE に書き込むために多くの時間が費やされるからです。アプリケーション環境においてトレースバック CEEDUMP が不要な場合は、CESE CICS 一時データ・キューに定様式 CEEDUMP を書き込むことによるパフォーマンス・オーバーヘッドをなくすために、TERMTHDACT(MSG) を使用してください。
TRAP	ON	ON	TRAP は、言語環境プログラム・ルーチンが異常終了およびプログラム割り込みを処理する方法を指定します。アプリケーションが正常に稼働するためには、TRAP(ON) を指定しなければなりません。

CICS アプリケーションに影響を与えるその他のランタイム・オプション

以下のランタイム・オプションも、既存のアプリケーションが言語環境プログラムのもとで稼働し、予期どおりの結果をもたらすかどうかに影響を与えます。デフォルト設定はそれぞれのインストール先の要件によって異なるため、特定のサブオプションは推奨できません。

CBLPSHPOP

CBLPSHPOP は、VS COBOL II、IBM COBOL、または Enterprise COBOL サブルーチンが COBOL CALL ステートメントを用いて呼び出されるときに CICS の PUSH HANDLE および POP HANDLE コマンドが発行されるかどうかを制御するために使用されます。

CBLPSHPOP(ON) を使用すると、VS COBOL II プログラムを VS COBOL II ランタイムで実行する場合の動作と互換性のある動作を得ることができます。CBLPSHPOP(OFF) は、パフォーマンス上の利益をもたらします。詳細については、117 ページの『CICS HANDLE コマンドおよび CBLPSHPOP ランタイム・オプション』を参照してください。

STORAGE

2 進ゼロに初期設定される WORKING-STORAGE に依存するプログラムの場合、このオプションを使用して、RENT を指定してコンパイルされたプログラムによって獲得されたストレージを 2 進ゼロに初期設定することができます (VALUE 文節が指定されている場合を除きます)。さらに、このオプションを使用して、プログラムのすべての外部データ・レコードを 2 進ゼロに設定することもできます。ただし、パフォーマンスを向上させるためには、初期設定が必要なデータ項目だけを明示的に初期設定してください。

言語環境プログラムで VS COBOL II の WSCLEAR ランタイム・オプションと同等の影響を得るには、言語環境プログラムの STORAGE ランタイム・オプションの最初のサブオプションを 00 に設定してください。たとえば、STORAGE(00,NONE,NONE,0K)。

IBM 提供のデフォルト設定は STORAGE(NONE,NONE,NONE,0K) です。

既存のアプリケーションの呼び出し

言語環境プログラムにアクセスするには、アプリケーションの呼び出しに使用するプロシージャーを変更する必要があります。非 CICS アプリケーションの場合に必要なプロシージャーは、CICS アプリケーションの場合のアプリケーションと異なります。

注: プログラム名が AFH、CEE、EDC、IBM、IGZ、ILB、または FOR で始まっていないことを確認してください。これらの接頭部は、言語環境プログラム・ライブラリー・ルーチン・モジュール名のために予約されています。

非 CICS アプリケーションの場合

以下のセクションでは、非 CICS アプリケーションの場合に必要な変更を説明します。IMS でプログラムを実行する場合の考慮事項については、357 ページの『付録

L. IMS の考慮事項』を参照してください。言語環境プログラムを使用してプログラムを作成および実行する方法については、*言語環境プログラム プログラミング・ガイド* を参照してください。

正しいライブラリーの指定

言語環境プログラムのもとでの稼働時に既存のアプリケーションを呼び出すには、以下のことを行う必要があります。

z/OS および OS/390 の場合

現行ライブラリーを言語環境プログラムの SCEERUN ライブラリーで置き換えます。

代替 DD 名の指定 (オプショナル)

言語環境プログラムでは、言語環境プログラム出力の宛先を、MSGFILE ランタイム・オプション内の DD 名を必要な DD 名に変更することによって指定することができます。表 18に、言語環境プログラムからの出力用のデフォルト DD 名を示します。

表 18. 新しい DD 名 の指定

出力	デフォルト DD 名	動的割り振り
メッセージ	SYSOUT	はい
ランタイム・オプション報告書 (RPTOPTS)	SYSOUT	はい
ストレージ報告書 (RPTSTG)	SYSOUT	はい
ダンプ	CEEDUMP	はい

言語環境プログラムによって使用されるデフォルトがインストール先の要件に合わない場合を除き、言語環境プログラムのメッセージ、報告書、またはダンプのための DD 名を定義するために JCL、CLIST、または Rexx EXEC を変更する必要はありません。言語環境プログラムのデフォルト宛先は、次のとおりです。

- z/OS および OS/390 の場合: SYSOUT=*
- TSO の場合: ALLOC DD(SYSOUT) DA(*)

不要になった DD 名の除去 (オプショナル)

言語環境プログラムでの稼働時には、以下の DD 名は必要ではありません。

- SYSABOUT (VS COBOL II ランタイムによってのみ使用される)
- SYSDBIN (VS COBOL II ランタイムによってのみ使用される)
- SYSDBOUT

これらの DD 名を除去することは必須ではありません。この情報は、 JCL、CLIST、または Rexx EXEC 内の不要なコーディングを除去したい場合に備えて提供されています。

CICS アプリケーションの場合

CICS で言語環境プログラムを実行するには、いくつかの必須ステップを行う必要があります。CICS で言語環境プログラムのもとで実行する COBOL アプリケーションの呼び出し方法 (言語環境プログラム・ランタイム・ライブラリー SCEERUN の指定方法を含む) については、以下の資料を参照してください。

- z/OS の場合: *z/OS 言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

CICS で言語環境プログラムを使用する場合の出力の違い

CICS のもとでは、言語環境プログラム出力は CESE という名前の一時データ・キューに送られます。ファイルに書き込まれた各レコードには、端末 ID、トランザクション ID、日付、および時刻を示すヘッダーがあります。表 19 に、言語環境プログラムからの出力のタイプおよび位置を示します。

表 19. CICS のもとでの言語環境プログラム出力の位置

出力	一時データ・キュー
メッセージ	CESE
ランタイム・オプション報告書 (RPTOPTS)	CESE
ストレージ報告書 (RPTSTG)	CESE
ダンプ	CESE
DISPLAY UPON SYSOUT 出力	CESE

既存のアプリケーションのリンク・エディット

既存のアプリケーションのうち、言語環境プログラムとリンク・エディットすることが必要であるもの、またはそのことから利益を得るものを判別した後、正しいライブラリー名を指定する必要があります。言語環境プログラムのリンク・エディット・ライブラリーは、非 CICS アプリケーションの場合と CICS アプリケーションの場合とで同じです。

z/OS および OS/390 の場合

言語環境プログラムの SCEELKED を SYSLIB 連結に組み込んでください。

注: NCAL リンケージ・エディター・オプションを指定してリンク・エディットする場合は、SCEELKED からのすべての必須ランタイム・ルーチンがロード・モジュールに組み込まれるようにしてください。そうしなければ、予期しないエラーが発生します (一般に、プログラム・チェック)。

SCEELKED ライブラリーには、IBM 命名規則に従っていない名前、およびサブプログラム名と対立する可能性のある名前がいくつか入っています。たとえば、静的に呼び出されるサブルーチンの名前が DUMP であり、SCEELKED がリンク・エディット時の連結内で private サブルーチン・ライブラリーの前にある場合は、DUMP の参照は SCEELKED 内で解決されることになります。この例で、FORTRAN ルーチン AFHUDUMS がリンク・エディットされると、その結果、ユーザーは誤りの結果を得るか、機能が消失するか、またはパフォーマンスが低下する可能性があります。(別の共通名は ABORT で、これは EDC4$05C (C ランタイム・ライブラリー・ルーチン) の入り口点です。)

これらの問題を回避するための方法が 2 つあります。

- SCEELKED データ・セット内の名前を private サブルーチンの名前に照らして検査します。重複がある場合は、private サブルーチンの名前が SCEELKED データ・セット内の名前と同じにならないように名前変更してください。

- もう 1 つの方法は、SYSLIB 連結内で private サブルーチン・ライブラリーを SCEELKED の前に置くことです。ただし、アプリケーションが FORTRAN または C/C++ プログラムを含んでいる場合は、これを行うと、言語環境プログラムのもとで使用可能な機能が失われることがあります。言語環境プログラム・サブルーチンとの対立を避けるために private サブルーチンの名前を変更する方が、private サブルーチン・ライブラリーを SCEELKED の前に置くよりも好ましい方法です。

言語環境プログラムとのリンク・エディットが必要なアプリケーションを判別するには、以下のいずれかを参照してください。

- 74 ページの『リンク・エディットが必要なプログラムの判別』 (OS/VS COBOL ランタイムのもとで稼働している場合)
- 88 ページの『リンク・エディットが必要なプログラムの判別』 (VS COBOL II ランタイムのもとで稼働している場合)

システム・ダンプまたは CICS トランザクション・ダンプの入手

重大エラー (たとえば、プログラム・チェックまたは異常終了) が原因で環境が終了するときに言語環境プログラムのもとでシステム・ダンプまたは CICS トランザクション・ダンプを受け取るためには、2 つの方式があります。どちらの方式を選択するかは、言語環境プログラムに生成させたい診断情報の量に応じて異なります。

どちらの方式も、言語環境プログラムの TERMTHDACT ランタイム・オプションの影響を受けます。このランタイム・オプションは、重大エラーが原因で環境が終了するときに言語環境プログラムによって生成される診断情報のレベルを設定します。TERMTHDACT ランタイム・オプションの詳細については、*言語環境プログラム プログラミング・リファレンス* を参照してください。

方式 1: TERMTHDACT ランタイム・オプションを指定する

言語環境プログラムは 3 つの TERMTHDACT サブオプション (UADUMP、UATRACE、および UAONLY) を提供し、異なるタイプの言語環境プログラム・ダンプを生成します。

TERMTHDACT(UADUMP) を指定すると、言語環境プログラムは、言語環境プログラム定様式ダンプとシステム・ダンプを生成します。この設定値の場合、言語環境プログラム・ダンプには、スレッド / エンクレーブ / プロセス・レベルのストレージおよび制御ブロックのトレースバックとダンプが含まれます。

TERMTHDACT(UATRACE) を指定すると、言語環境プログラムは、終了の原因を示すメッセージ、活動化スタック上のアクティブ・ルーチンのトレース、および U4039 異常終了を生成します。この異常終了により、ユーザー・アドレス・スペースのシステム・ダンプが生成されます。

TERMTHDACT(UAONLY) を指定すると、言語環境プログラムは、U4039 異常終了を生成します。この異常終了により、ユーザー・アドレス・スペースのシステム・ダンプが生成されます。非 CICS のもとでは、適切な DD ステートメントを使用すれば、ユーザー・アドレス・スペースのシステム・ダンプを入手できます。CICS のもとでは、CICS トランザクション・ダンプを入手できます。

TERMTHDACT(UAIMM) を指定すると、言語環境プログラムは、U4039 異常終了を生成します。この異常終了により、ユーザー・アドレス・スペースのシステム・ダンプが生成されます。非 CICS のもとでは、適切な DD ステートメントを使用すれば、ユーザー・アドレス・スペースのシステム・ダンプを入手できます。

方式 2: 異常終了出口を指定する

異常終了出口を使用することは、TERMTHDACT ランタイム・オプションを DUMP または UADUMP 以外の値に設定したいときに、システム・ダンプまたは CICS トランザクション・ダンプを入手するための推奨されるアプローチです。

異常終了出口を指定すると、言語環境プログラムが獲得したリソースを解放する前に、システム・ダンプまたは CICS トランザクション・ダンプを入手することができます。そのため、システム・ダンプには影響を与えずに、定様式ダンプで受け取る診断情報の量が削減されます。

非 CICS のもとでの異常終了出口

非 CICS アプリケーションの場合、サンプル異常終了出口は SAMPDAT1 です (70 ページの図 3 を参照)。SAMPDAT1 はシステム異常終了ダンプを提供します。

CICS のもとでの異常終了出口

CICS アプリケーションの場合、サンプル異常終了出口は SAMPDAT2 です (71 ページの図 4 を参照)。SAMPDAT2 はトランザクション・ダンプを提供します。

異常終了出口の使用法については、以下の資料を参照してください。

- z/OS の場合: *z/OS 言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

異常終了出口 (非 CICS)

```
*********************************************************************
*                                                                   *
*  Do a system DUMP whenever an unhandled condition occurs.         *
*                                                                   *
*********************************************************************
SAMPDAT1 CEEENTRY PPA=ASMPPA,MAIN=NO
         L     2,0(,1)                    Put the pointer to the CIB
*                                         address in R2.
         L     2,0(,2)                    Put the CIB address in R2.
**********************************************************************
* Set up the ESTAE and force the abend with a dump.
**********************************************************************
         ESTAE ESHDLR
         ABEND 4039,REASON=0,DUMP         FORCE DUMP
RETRY    ESTAE 0
         CEETERM                          All done, return to LE/370
         DROP  11,13
         USING *,15
ESHDLR   STM   14,12,12(13)
NEXT     L     11,MODENT
         USING SAMPDAT1,11
         DROP  15
         SETRP RC=4,RETADDR=RETRY,RETREGS=YES,FRESDWA=YES
         LM    14,12,12(13)
         BR    14
MODENT   DC    A(SAMPDAT1)
ASMPPA   CEEPPA
         CEEDSA
         CEECAA
SDWA     IHASDWA
         END SAMPDAT1
```

図 3. 非 CICS 異常終了出口サンプル

異常終了出口 (CICS)

```
*ASM CICS(NOPROLOG NOEPILOG NOEDF SYSEIB)
*********************************************************************
*                                                                   *
*  Do a transaction DUMP whenever an unhandled condition occurs.    *
*                                                                   *
*********************************************************************
SAMPDAT2 CEEENTRY PPA=ASMPPA,MAIN=NO,AUTO=STORLEN
         USING DFHEISTG,DFHEIPLR
**************************************************************************
* Ask CICS to produce a transaction dump.
**************************************************************************
         EXEC CICS ADDRESS EIB(DFHEIBR)
         EXEC CICS DUMP TRANSACTION DUMPCODE('4039') TASK NOHANDLE
***************************************************************************
* To see if the dump was successful, add code here to check field EIBRESP.
***************************************************************************
         CEETERM                          All done, return to LE/370
ASMPPA   CEEPPA
         CEEDSA
         CEECAA
         DFHEISTG                         Extended save area for CICS
STORLEN  EQU   *-DFHEISTG
         COPY  DFHEIBLK
         EXTRN DFHEAI
DFHEIPLR EQU   13
DFHEIBR  EQU   10
         END SAMPDAT2
```

図 4. CICS 異常終了出口サンプル

互換性のある異常終了動作の取得

OS/VS COBOL および VS COBOL II と類似した異常終了動作を得るためには、言語環境プログラムの ABTERMENC ランタイム・オプションおよびアセンブラー・ユーザー出口を使用してください。

ABTERMENC(ABEND) を使用すると、異常終了、プログラム・チェック、または重大エラーの発生時に、VS COBOL II または OS/VS COBOL によって出されるのと類似したシステム異常終了コードを受け取ることができます。(システム・ダンプを入手するには、 68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。)

未処理の言語環境プログラム・ソフトウェア生成条件があるときに、VS COBOL II のもとで受け取ったのと類似したユーザー異常終了コード (ここで、xxx は IGZ メッセージ番号) を受け取るためには、以下のいずれかを行ってください。

- COBOL サンプル・アセンブラー・ユーザー出口 (CEEBX05A) からのコードをアセンブラー・ユーザー出口 (CEEBXITA) にコピーすることにより、CEEBXITA を変更してください。
- 103 ページの『CEEWUCHA の使用』で説明されているように、サンプルのユーザー条件処理ルーチン CEEWUCHA を使用してください。

戻りコード値の互換性の確保

言語環境プログラムは、OS/VS COBOL または VS COBOL II とは異なる方法で戻りコード値を計算します。言語環境プログラムのもとでの稼働時に異なる戻りコード値を受け取る可能性があるケースとしては、以下の 2 つがあります。

- ABTERMENC(RETCODE) ランタイム・オプションを指定する場合
- 言語環境プログラムのアセンブラー・ユーザー出口を変更して戻りコード値を操作する場合

第 6 章 OS/VS COBOL ランタイムからの移行

この章では、OS/VS COBOL プログラムを言語環境プログラムのもとで実行する方法を詳しく説明します。以下の情報が記載されています。

- リンク・エディットが必要なプログラムの判別
- アップグレードが必要なプログラムの判別
- ランタイム・オプションおよび指定方式の比較
- 非 COBOL および OS/VS COBOL プログラムでのファイルのクローズ
- 再使用可能実行時環境での実行
- ダンプ・サービスの管理
- ILBOABN0 を用いての異常終了の強制
- OS/VS COBOL プログラムでの SORT または MERGE の使用
- SYSOUT 出力の変更点の理解
- 他の言語との通信
- その他の CICS 考慮事項

ランタイムを言語環境プログラムに移行する場合の追加情報については、以下の付録で説明されています。

- 303 ページの 『付録 D. COBOL とアセンブラーを含んでいるアプリケーション』
- 357 ページの『付録 L. IMS の考慮事項』

この章の各セクションでは、以下の表記を使用することによって、そのセクションが既存の OS/VS COBOL プログラムのうち、RES コンパイラー・オプションを指定してコンパイルされたもの、NORES コンパイラー・オプションを指定してコンパイルされたもの (プログラムが言語環境プログラムとリンク・エディットされているかどうかを含む)、および CICS で実行されるアプリケーション用のものに適用されるかどうかを示します。

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	✓ NORES (リンク済み)

RES RES を指定してコンパイルされたプログラムから構成されるアプリケーション。

NORES

NORES を指定してコンパイルされたプログラムから構成されるアプリケーション。この NORES プログラムは、言語環境プログラムとリンク・エディットされていません。

または

NORES を指定してコンパイルされた OS/VS COBOL プログラムから構成されるアプリケーション。この NORES プログラムは、言語環境プログラムを使用してリンク・エディットされている が、以下のいずれも含んでいません。

- VS COBOL II プログラム

- COBOL/370 プログラム
- COBOL (MVS および VM 版) プログラム
- COBOL (OS/390 および VM 版) プログラム
- COBOL for z/OS and OS/390 プログラム
- IGZCBSN または IGZCBSO ブートストラップ・ルーチン

CICS CICS のもとで実行されるアプリケーション。

NORES (リンク済み)
NORES を指定してコンパイルされたプログラムから構成されるアプリケーション。この NORES プログラムは、言語環境プログラムとリンク・エディットされており、現在は RES と同様に動作します。アプリケーションは、以下の少なくとも 1 つを含んでいます。

- VS COBOL II プログラム
- COBOL/370 プログラム
- COBOL (MVS および VM 版) プログラム
- COBOL (OS/390 および VM 版) プログラム
- COBOL for z/OS and OS/390 プログラム
- IGZCBSN または IGZCBSO ブートストラップ・ルーチン

注: 複数ロード・モジュール・アプリケーションの場合、*最初に実行される*ロード・モジュールが上記の 1 つを含んでいると、アプリケーションは RES と同様に動作します。

リンク・エディットが必要なプログラムの判別

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	CICS	NORES (リンク済み)

言語環境プログラムとリンク・エディットすることが必要なプログラムを判別するには、以下のことを知る必要があります。

1. プログラムが RES と NORES のどちらのコンパイラー・オプションを指定してコンパイルされたか
2. プログラムが再使用可能環境 (ILBOSTP0 によって確立される) を使用するかどうか

リンク・エディットを必要とする OS/VS COBOL プログラムを含んでいるロード・モジュールの要約表については、256 ページの表 47 を参照してください。

COBOL ロード・モジュールを言語環境プログラムと再リンク・エディットする方法の追加情報については、349 ページの『付録 J. リンク・エディットの例』を参照してください。

RES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション

再使用可能環境を確立するために ILBOSTP0 を呼び出す (つまり、アセンブラー言語プログラムをメインプログラムとして、ILBOSTP0 とリンク・エディットして呼び出す) 場合は、アセンブラー・プログラムを言語環境プログラムを使用してリンク・エディットする必要があります。

NORES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション

NORES を指定してコンパイルされた既存の OS/VS COBOL プログラムは、変更なしで稼働し、以前と同じ結果をもたらします。これらのプログラムは、言語環境プログラムとリンク・エディットする必要はありません。ただし、これらのプログラムを言語環境プログラムとリンク・エディットしない場合は、これらの NORES アプリケーションのために IBM サービス・サポートを受けることはできません。

注: プログラムに CALL identifier ステートメントが含まれている場合は、コンパイラーは NORES 指定をオーバーライドして RES とします。その結果、NORES をインストール先のデフォルトとして指定した場合でも、アプリケーションには RES と NORES が混在することがあります。

NORES を指定してコンパイルされたプログラムを言語環境プログラムとリンク・エディットする場合は、可能性のある動作の変更について、121 ページの『第 8 章 アプリケーションと言語環境プログラムとのリンク・エディット』を参照してください。

RES および NORES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション

この組み合わせは OS/VS COBOL のもとではサポートされませんでしたが、場合によっては機能しました。これらのプログラムを言語環境プログラムとリンク・エディットし、以下の少なくとも *1* つを、COBOL を含んでいる最初に実行されるロード・モジュールに組み込む必要があります。

- VS COBOL II プログラム
- COBOL/370 プログラム
- COBOL (MVS および VM 版) プログラム
- COBOL (OS/390 および VM 版) プログラム
- COBOL for z/OS and OS/390 プログラム
- IGZCBSN ブートストラップ・ルーチン
- IGZCBSO ブートストラップ・ルーチン

アップグレードが必要なプログラムの判別

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	NORES (リンク済み)

注: この章で説明しているアップグレードについて、すべての OS/VS COBOL プログラムを Enterprise COBOL (または COBOL (OS/390 および VM 版) バージョン 2 リリース 2) にアップグレードすることをお勧めします。

CICS の場合

CICS での実行時に動的 CALL ステートメントを発行する OS/VS COBOL プログラムは、異常終了コード U3504 で異常終了します。実行単位内のすべての COBOL プログラムを Enterprise COBOL にアップグレードするか、または動的 CALL ステートメントを発行しないようプログラムを再コーディングしてください。

以下の場合は、OS/VS COBOL プログラムをアップグレードする必要があります。

- CICS TS バージョン 2 リリース 2 以降のリリースの CICS Transaction Server の使用を検討している場合

非 CICS の場合

以下の場合は、OS/VS COBOL プログラムをアップグレードする必要があります。

- プログラムが PL/I との ILC を使用する場合
- プログラムが FORTRAN との ILC を使用する場合
- OS/VS COBOL プログラムが複数のエンクレーブ (または実行単位) に含まれている場合

以下のアップグレードが必要な場合もあります。

- 言語環境プログラムまたは Enterprise COBOL の機能 (言語環境プログラム呼び出し可能サービスまたは組み込み関数など) を使用するアプリケーション
- 言語環境プログラムのもとで正しく稼働するために再コーディングが必要なアプリケーション

PL/I との ILC

PL/I を呼び出すかまたは PL/I によって呼び出される OS/VS COBOL プログラムは、アップグレードしなければなりません。

FORTRAN との ILC

FORTRAN を呼び出すかまたは FORTRAN によって呼び出される OS/VS COBOL プログラムは、アップグレードしなければなりません。

複数のエンクレーブに含まれている OS/VS COBOL プログラム

言語環境プログラムのもとでは、OS/VS COBOL プログラムが複数のエンクレーブに含まれていてはなりません。複数のエンクレーブは、いくつかの方法で作成される可能性があります。

- OS/VS COBOL プログラムがアセンブラー・プログラムを呼び出し、次にそれが別の OS/VS COBOL プログラムへの SVC LINK を発行する場合
- RES を指定してコンパイルされたプログラムから構成される OS/VS COBOL アプリケーションを言語環境プログラムのもとで実行し、そのアプリケーションが ISPF サービス (CALL 'ISPLINK' または ISPF SELECT など) を使用する場合

複数エンクレーブ・アプリケーションの場合、どのエンクレーブが OS/VS COBOL プログラムを含んでいるかを判別してください。1 つのエンクレーブだけが OS/VS

COBOL プログラムを含むことができます。その他のエンクレーブに含まれている OS/VS COBOL プログラムは、アップグレードしなければなりません。

ランタイム・オプションおよび指定方式の比較

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	CICS	✓ NORES（リンク済み）

ここでは、言語環境プログラム・ランタイム・オプションを指定するために使用できるメカニズムをリストし、それぞれの方式を簡単に説明します。

言語環境プログラム ランタイム・オプションの指定

言語環境プログラムでは、ランタイム・オプションを指定するための各種の方式があります。オプションを指定するための 3 つのメカニズムが使用可能です。

- ランタイム・オプション CSECT
  - CEEDOPT (インストール・システム全体のデフォルト用)
  - CEEROPT (領域全体のデフォルト用)
  - CEEUOPT (アプリケーション固有のデフォルト用)
- 呼び出し手順
- アセンブラー・ユーザー出口

既存のアプリケーションのために使用できる言語環境プログラム方式を判別するには、メインプログラムが RES と NORES のどちらを指定してコンパイルされたか、およびメインプログラムが言語環境プログラムとリンク・エディットされているかどうかを知る必要があります。表 20 に、言語環境プログラムのもとでランタイム・オプションを指定およびオーバーライドするのに使用できる指定方式をリストします。

表 20. OS/VS COBOL プログラム用のランタイム・オプションを指定するために使用できる方式

メインプログラム	CEEDOPT および CEEROPT	CEEUOPT	呼び出し	デフォルト・アセンブラー・ユーザー出口
言語環境プログラムとリンク・エディットされていない:				
OS/VS COBOL RES	X		X	X
OS/VS COBOL NORES			X	
言語環境プログラムとリンク・エディットされている:				
OS/VS COBOL RES	X		X	X
OS/VS COBOL NORES			X	
OS/VS COBOL NORES (リンク済み)[1]	X		X	X

表 20. OS/VS COBOL プログラム用のランタイム・オプションを指定するために使用できる方式 (続き)

メインプログラム	CEEDOPT および CEEROPT	CEEUOPT	呼び出し	デフォルト・アセンブラー・ユーザー出口

注:

1. このケースでは、NORES プログラムはリンク・エディット後に RES 動作を示します。言語環境プログラムとのリンク・エディットの影響については、121 ページの『第 8 章 アプリケーションと言語環境プログラムとのリンク・エディット』を参照してください。

インストール・システム全体のデフォルト

CEEDOPT (非 CICS) および CEECOPT (CICS) アセンブラー・ファイルには、すべての言語環境プログラム・ランタイム・オプションのデフォルト値が入っています。インストール時に、このファイルを編集し、共通環境で実行されるすべてのアプリケーション (CICS のもとでの OS/VS COBOL プログラムは除く) に適用されるデフォルトを選択することができます。

このモジュール内のオプションについては、オーバーライド不能 (固定) 属性を選択することができます。この属性により、インストール・システムでは、操作環境全体にとって重要であると思われるオプションを強制することができます。

言語環境プログラムでインストール・システム全体のデフォルト・ランタイム・オプションを設定する方法については、以下の資料を参照してください。

- z/OS の場合: *z/OS 言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

領域全体のデフォルト

CEEROPT を CEEDOPT および CEECOPT と同様の方法で使用して、ランタイム・オプションを指定することができます。CEEROPT を使用して、CICS 領域、IMS 領域、またはライブラリー・ルーチン保存を使用する MVS バッチ・ジョブ領域での言語環境プログラムのランタイム・オプション・デフォルトを設定することができます。CEEROPT はそれ自体のロード・モジュールに置かれるので、実行可能コードを含んでいるロード・モジュールに CEEROPT をリンクすることに関連した保守問題が回避されます。CEEROPT は、領域の初期設定時にロードされ、インストール・システム・ランタイム・オプション・デフォルトとマージされます。

CEEROPT はオプショナルです。環境の初期設定時に、CEEROPT を見つけることが試みられます。見つかった場合は、CEEROPT 内に指定されているランタイム・オプションが、CEEDOPT または CEECOPT 内に指定されているインストール・システム・デフォルトとマージされます。

言語環境プログラムで領域全体のデフォルトのランタイム・オプションを設定する方法については、*言語環境プログラム OS/390 版 カスタマイズ* を参照してください。

CEEUOPT (アプリケーション固有のデフォルト)

CEEUOPT は、OS/VS COBOL プログラムをメインプログラムとして含んでいるロード・モジュールでは使用できません。このことは、OS/VS COBOL プログラムを言語環境プログラムとリンク・エディットしているかどうかには関係なく該当します。

呼び出し手順

オペレーティング・システムから直接に呼び出される OS/VS COBOL プログラムの場合は、プログラム呼び出し時にオペレーティング・システムの適切なメカニズムを使用してランタイム・オプションを指定することができます。

z/OS および OS/390 の場合

JCL の EXEC ステートメントの PARM パラメーターで、ランタイム・オプションを指定することができます。

注: CICS では、PARM パラメーターを使用してランタイム・オプションを指定することはできません。

詳細については、*言語環境プログラム プログラミング・ガイド* を参照してください。

優先順位

1 つのアプリケーションについてのランタイム・オプションを指定するために上記のすべてのメカニズムを使用することができます。言語環境プログラムでは、以下の優先順位規則が適用されます。

1. インストール時にオーバーライド不能として指定されたインストール・システム全体のデフォルト
2. アセンブラー・ユーザー出口を使用して指定されたオプション
3. 呼び出し時に指定されたオプション
4. アプリケーション固有のオプション (アプリケーションとリンク・エディットされている)
5. CEEROPT を使用して定義された領域全体のデフォルト
6. インストール時に定義されたインストール・システム全体のデフォルト

OS/VS COBOL と言語環境プログラム・ランタイム・オプションの比較

次の表では、言語環境プログラムが OS/VS COBOL ランタイム・オプションについて提供するサポートを記述します。

表 21. OS/VS COBOL と言語環境プログラムのランタイム・オプションの比較

OS/VS COBOL オプション	説明
AIXBLD	AIXBLD は、言語環境プログラムのもとで OS/VS COBOL の場合と同じようにサポートされ、同じ構文で指定されます。
DEBUG	DEBUG は、言語環境プログラムのもとで OS/VS COBOL の場合と同じようにサポートされます。

表 21. OS/VS COBOL と言語環境プログラムのランタイム・オプションの比較 (続き)

OS/VS COBOL オプション	説明
FLOW FLOW=n	FLOW は、言語環境プログラムのもとで OS/VS COBOL の場合と同じようにサポートされます。FLOW は、CEEDOPT または CEEUOPT を用いて指定することはできません。アプリケーションの呼び出し時にのみ指定することができます。
QUEUE	QUEUE はサポートされません。
UPSI	UPSI は、言語環境プログラムのもとで OS/VS COBOL の場合と同じようにサポートされ、同じ構文で指定されます。

非 COBOL および OS/VS COBOL プログラムでのファイルのクローズ

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES (リンク済み)

z/OS または OS/390 の場合、ファイルをクローズしない OS/VS COBOL プログラムまたはアセンブラー・プログラムがあると、C03 異常終了が発生する可能性があります。バッチでは、COBOL メインプログラムがアセンブラー・プログラムから呼び出される場合、または COBOL がメインプログラムでない場合は、以下の変更のいずれかを行うことにより、C03 異常終了が発生しないようにする必要があります。

- ファイルをクローズするためのコードを追加する。
- IBM COBOL または Enterprise COBOL プログラムのいずれかを使用してファイルをオープンする。
- COBOL プログラムがアセンブラー・プログラムに呼び出され、そのアセンブラー・プログラムが z/OS または OS/390 のバッチ・イニシエーターによって起動される場合、(IBM COBOL または Enterprise COBOL でコンパイルされた) COBOL スタブをアプリケーションに追加してください。このようにすると、言語環境プログラムは動的に呼び出されたロード・モジュールを終了時に解放しません (アセンブラー・プログラムとそれに含まれている入出力制御ブロックはストレージに残ります)。
- アセンブラー・プログラムがファイルをオープンする場合は、そのプログラムを変更して、データ管理制御ブロック用の GETMAIN を実行してください (それらの制御ブロックをロード・モジュールに残さないようにします)。

その他の環境

その他の環境、たとえば、TSO、IMS、または (CEEPIPI または IGZERRE プログラムで) 事前初期設定済みの環境で稼働するプログラムの場合は、エンクレーブ終了の前に、すべてのファイルを OS/VS COBOL から明示的にクローズする (または OS/VS COBOL プログラムをアップグレードする) 必要があります。言語環境プログラムは、OS/VS COBOL プログラムによってオープンされたままになっているファイルを自動的にクローズすることはしません。

COBOL によって呼び出されたアセンブラー・プログラムは、以下の場合は、エンクレーブ終了の前にファイルをクローズする必要があります。

- アセンブラー・プログラムが、COBOL 動的 CALL ステートメントによってロードされたロード・モジュールに含まれている場合。
- アセンブラー・プログラムが、アセンブラー・プログラム内または COBOL WORKING-STORAGE データ項目内でファイル制御ブロック用のストレージを割り振る場合。

再使用可能実行時環境での実行

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

言語環境プログラムは、ILBOSTP0 によって提供されるアプリケーション用の再使用可能実行時環境のための互換性サポートを引き続き提供します。

言語環境プログラムのもとで再使用可能環境を既存のアプリケーションと共に使用する場合は、以下の制約事項が適用されます。

- IGZERREO CSECT を使用して動作を変更しない限り、アプリケーションは、単一エンクレーブ・アプリケーションでなければなりません (つまり、他の COBOL プログラムへの SVC LINK は許可されません)。

  注：IGZERREO CSECT を使用して動作を変更しても、OS/VS COBOL が複数のエンクレーブに含まれていてはならないという制約事項は除去されません。
- 再使用可能環境が確立された後、アセンブラー・ドライバーが呼び出せるのは、COBOL プログラムまたは言語環境プログラム準拠でないアセンブラー・プログラムだけです。次いで、COBOL は次のように、サポートされる言語を呼び出すことができます。
  - アセンブラー・ドライバーから呼び出された COBOL プログラムは、静的 CALL ステートメントを使用して別の言語を呼び出すことはできません。別の言語への動的呼び出しはサポートされます。
  - 別の COBOL プログラムから動的に呼び出された COBOL プログラムは、別の言語への静的 CALL ステートメントまたは動的 CALL ステートメントを使用することができます。
- 最初の高水準言語プログラムは COBOL プログラムでなければなりません (アセンブラーは高水準言語と見なされません)。
- アセンブラー・ドライバーが再使用可能環境を提供するために ILBOSTP0 と静的にリンクされており、それを呼び出す場合は、言語環境プログラムとリンク・エディットされていなければなりません。

STOP RUN を使用すると、再使用可能環境は、その初期設定に使用された方式に関係なく終了します。ILBOSTP0 がアセンブラー・プログラムによって呼び出された場合は、制御はアセンブラー・プログラムの呼び出し元に戻されます。

IGZERREO CSECT を変更すれば、COBOL 再使用可能環境を使用するときのパフォーマンスを向上させることができ、また、再使用可能環境での実行時にネストされたエンクレーブを使用することができます。詳細については、以下の資料を参照してください。

- z/OS の場合: *z/OS 言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

ILBOSTP0 の使用

非 COBOL プログラムをメインプログラムとして使用する既存のアプリケーションが引き続き稼働し、同じ結果をもたらすように、ILBOSTP0 が引き続きサポートされます。

注: ILBOSTP0 は AMODE 24 であるため、言語環境プログラムのもとで使用するときには、言語環境プログラムは自動的に ALL31(OFF) および STACK(,,BELOW) を設定します。

ダンプ・サービスの管理

このセクションは以下のものに適用されます。

✓ RES	NORES	✓ CICS	✓ NORES（リンク済み）

ダンプ・サービスは言語環境プログラムによって管理されます。ここでは、シンボリック・ダンプ、システム・ダンプ、およびトランザクション・ダンプのためのサポートを説明します。さらに、言語環境プログラムの定様式ダンプに関する情報が記載されています。

OS/VS COBOL シンボリック・ダンプ

NORES コンパイラー・オプションを指定してコンパイルされた OS/VS COBOL プログラムの場合、SYMDMP を用いて生成されたダンプ出力は同じままであり、引き続き SYSDBOUT に送られます。RES コンパイラー・オプションを使用した場合は、代わりに CEEDUMP が生成されます。

システム・ストレージ・ダンプおよび CICS トランザクション・ダンプ

ダンプに含まれる PSW (プログラム状況ワード) とレジスター情報は、言語環境プログラムのもとでの実行時と OS/VS COBOL ランタイムのもとでの実行時では異なります。

OS/VS COBOL では、ダンプはエラー発生時に生成されます。PSW およびレジスターの内容は、エラー発生時に使用可能な情報に基づきます。このため、ダンプ内の PSW 情報とレジスター情報を問題判別に使用することができます。

言語環境プログラムでは、ダンプは、言語環境プログラムの条件管理がエラーを処理した後で生成されます。PSW およびレジスターの内容は、エラー発生時に使用可能な情報に基づくのではなく、言語環境プログラムがエラーの処理を完了した時点で使用可能な情報に基づきます。エラー発生時に PSW とレジスターに入っていた

情報を判別するには、言語環境プログラムの定様式ダンプ出力を調べるか、ダンプ内で言語環境プログラム条件情報ブロックと言語環境プログラム・マシン状態情報ブロックを探して使用することができます。詳細については、*言語環境プログラム デバッグのガイドおよびランタイム・メッセージ* を参照してください。

言語環境プログラムのもとでシステム・ストレージ・ダンプまたは CICS トランザクション・ダンプを入手するには、68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。

OS/VS COBOL と言語環境プログラムのどちらのもとでも、システム・ストレージ・ダンプ出力は SYSUDUMP、SYSABEND、または SYSMDUMP に送られます。CICS トランザクション・ダンプ出力は DFHDUMPA または DFHDUMPB に送られます。

言語環境プログラムの定様式ダンプ

OS/VS COBOL は定様式ダンプを生成しませんでした。言語環境プログラムでは、レベル 2 以上の条件が発生し、未処理のままになっているときは、TERMTHDACT(DUMP) または TERMTHDACT(UADUMP) ランタイム・オプションを指定することによって定様式ダンプを生成することができます。

言語環境プログラムの定様式ダンプには、以下のものに関する情報など、各種の情報を含めることができます。

- 変数、引き数、およびレジスターのシンボリック・ダンプ
- ファイル制御ブロック
- ファイル・バッファー
- ランタイム制御ブロック
- プログラムによって使用されたストレージ

言語環境プログラムのダンプ出力の例については、*言語環境プログラム デバッグのガイドおよびランタイム・メッセージ* を参照してください。

非 CICS の場合のダンプ宛先

言語環境プログラムでは、JCL または FILEDEF で CEEDUMP DD 名を指定することによって、ダンプ出力の宛先を指示することができます。CEEDUMP ファイルの属性については、*言語環境プログラム プログラミング・ガイド* を参照してください。

CEEDUMP ファイルが必要であるが、定義されていない場合、言語環境プログラムは以下のデフォルトを使用してこのファイルを動的に割り振ります。

- z/OS および OS/390 の場合: SYSOUT=*

CICS の場合のダンプ宛先

言語環境プログラムからのランタイム出力はすべて、CESE という名前の CICS 一時データ・キューに送られます。ファイルに書き込まれた各レコードには、端末 ID、トランザクション ID、および日時を示すヘッダーがあります。

ILBOABN0 を用いての異常終了の強制

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

OS/VS COBOL では、ILBOABN0 への CALL を使用して、強制的に即時異常終了させ、システム・ダンプを取得することができます。言語環境プログラムでも引き続き、ILBOABN0 への CALL によって強制的な即時異常終了を引き起こすことができます。システム・ダンプを生成するには、68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。

言語環境プログラム版の ILBOABN0 を使用する場合、ILBOABN0 を発行するプログラムの保管域は、実際に異常終了が発生する保管域より 2 レベル後ろに置かれています。

ILBOABN0 への CALL を使用する OS/VS COBOL プログラムは、Enterprise COBOL でコンパイルされた場合、引き続き ILBOABN0 を呼び出すことができます。ただし、その代わりに言語環境プログラムの CEE3ABD 呼び出し可能サービスを使用することをお勧めします。

OS/VS COBOL プログラムでの SORT または MERGE の使用

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

言語環境プログラムのインプリメンテーションにより、OS/VS COBOL の SORT または MERGE を管理するランタイム・ルーチンが変更されました。OS/VS COBOL プログラムが SORT または MERGE を初期設定している場合、以下の変更が適用されます。

- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックまたは異常終了が発生した場合、言語環境プログラムは言語環境プログラム・ダンプを生成しません。
- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックまたは異常終了が発生した場合、OS/VS COBOL デバッグ・データは生成されません。
- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックまたは異常終了が発生した場合、言語環境プログラムは環境の終結処理を実行しません。
- 入力または出力プロシージャー内でアセンブラー・プログラムを呼び出した場合、そのアセンブラー・プログラムは COBOLプログラムを呼び出すことができません。
- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックが発生した場合、アプリケーションは異常終了コード U4036 で終了します。プログラム・チェック発生時のレジスターおよび PSW の状態を調べる方法については、*言語環境プログラム デバッグのガイド*および*ランタイム・メッセージ* を参照してください。

SYSOUT 出力の変更点の理解

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

言語環境プログラムの場合は、SYSOUT 出力は OS/VS COBOL の場合と異なります。違いは次のとおりです。

- RECFM=FB のファイルに送られるか、または他の宛先に送られる (DD に DCB=(RECFM=FB) が指定されているとき) SYSOUT 出力
- OS/VS COBOL トレース出力シーケンス

RECFM=FB を指定した SYSOUT 出力

言語環境プログラムの場合は、RECFM が FB であるファイルに送られるか、または他の宛先に送られる (DD に DCB=(RECFM=FB) が指定されているとき) DISPLAY/TRACE/EXHIBIT SYSOUT 出力の場合、以下に示されているように、先頭文字が出力に組み込まれません。

レコード・フォーマット	言語環境プログラムの場合	OS/VS COBOL の場合
RECFM=FBA	$01234	$01234
RECFM=FB	01234	$01234

「$」は、桁 1 にある制御文字を表します。

OS/VS COBOL トレース出力シーケンス

言語環境プログラムのもとでは、トレース出力シーケンスが変更されました。言語環境プログラムの場合は、OS/VS COBOL の場合のように 1 つのレコードにつき複数のトレース記入項目が現れるのではなく、1 つのレコードにつき *1* つのトレース記入項目だけが現れます。このことは、介在する DISPLAY SYSOUT 出力およびエラー・メッセージの場合に特に役立ちます。

図 5 に、OS/VS COBOL と言語環境プログラム間の違いを示します。

```
OS/VS COBOL :
     Record 1 -->  Section1(2), Paragraph1(2), Section2(2)

言語環境プログラム :
     Record 1 -->  Section1
     Record 2 -->  Section1
     Record 3 -->  Paragraph1
     Record 4 -->  Paragraph1
     Record 5 -->  Section2
     Record 6 -->  Section2
```

図 5. 言語環境プログラムにおけるトレース出力 (OS/VS COBOL との比較)

他の言語との通信

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	✓ NORES（リンク済み）

ここでは、既存の OS/VS COBOL プログラムに関する ILC の考慮事項の高度な概説を示します。言語環境プログラムのもとでの ILC に関する正確な詳細および最新の情報については、*言語環境プログラム ILC (言語間通信) アプリケーションの作成*を参照してください。

言語間通信とは、他の高水準言語を呼び出すかまたは他の高水準言語によって呼び出されるプログラムとして定義されています。アセンブラーは高水準言語と見なされません。このため、アセンブラー言語プログラムへの呼び出しおよびアセンブラー言語プログラムからの呼び出しは ILC と見なされません。COBOL プログラムとアセンブラー・プログラムに関係のある呼び出しに関する言語環境プログラムのサポートの詳細については、以下を参照してください。

- 305 ページの『非 CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』
- 307 ページの『CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』

CICS の場合、引き続き EXEC CICS LINK および EXEC CICS XCTL を使用して、他の言語と通信することができます。

表 22 に、ILC を使用する既存のアプリケーションのために必要な処置を示します。

表 22. ILC を使用する既存のアプリケーションのために必要な処置

高水準言語	説明
PL/I	PL/I を呼び出すかまたは PL/I によって呼び出される OS/VS COBOL プログラムは、アップグレードしなければなりません。
FORTRAN	FORTRAN を呼び出すかまたは FORTRAN によって呼び出される OS/VS COBOL プログラムは、アップグレードしなければなりません。
C	OS/VS COBOL プログラムと C プログラム間の ILC はサポートされません (今までサポートされたことがありません)。

その他の CICS 考慮事項

CICS のもとで言語環境プログラムを使用して稼働する OS/VS COBOL プログラムは、完全な言語環境プログラム・ランタイム環境のサブセット内で稼働します。OS/VS COBOL アプリケーションを CICS で実行する場合、実行単位のために言語環境プログラムによって確立された環境は、OS/VS COBOL のみサポートします。CICS プロダクトを使用する OS/VS COBOL プログラムのこの特殊なサポートは、CICS Transaction Server のバージョン 2 リリース 2 以降に出荷されるリリースの CICS Transaction Server では使用できなくなります。

第 7 章 VS COBOL II ランタイムからの移行

この章では、VS COBOL II ランタイムで実行されていた既存の OS/VS COBOL および VS COBOL II プログラムについて互換性および同等のランタイム結果を得るための方法を説明します。以下の情報が記載されています。

- リンク・エディットが必要なプログラムの判別
- アップグレードが必要なプログラムの判別
- ランタイム・オプションおよび指定方式の比較
- 非 COBOL および OS/VS COBOL プログラムでのファイルのクローズ
- 再使用可能実行時環境での実行
- メッセージ、異常終了コード、およびダンプ・サービスの管理
- ILBOABN0 を用いての異常終了の強制
- SORT または MERGE の使用
- SYSOUT 出力の変更点の理解
- 他の言語との通信
- ランタイム環境の初期設定
- ストレージ調整の変更点の判別
- その他の CICS 考慮事項
- 文書化されていない VS COBOL II 拡張

ランタイムを言語環境プログラムに移行する場合の追加情報については、以下の付録で説明されています。

- 303 ページの 『付録 D. COBOL とアセンブラーを含んでいるアプリケーション』
- 357 ページの『付録 L. IMS の考慮事項』

この章の各セクションでは、以下の表記を使用することによって、そのセクションが既存の VS COBOL II または OS/VS COBOL プログラムのうち、RES コンパイラー・オプションを指定してコンパイルされたもの、NORES コンパイラー・オプションを指定してコンパイルされたもの (プログラムが言語環境プログラムとリンク・エディットされているかどうかを含む)、および CICS で実行されるアプリケーションに適用されるかどうかを示します。

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	✓ NORES (リンク済み)

RES RES を指定してコンパイルされたプログラムから構成されるアプリケーション。

NORES
NORES を指定してコンパイルされたプログラムから構成されるアプリケーション。この NORES プログラムは、言語環境プログラムとリンク・エディットされていません。

または

NORES を指定してコンパイルされた OS/VS COBOL プログラムから構成され、かつ、言語環境プログラムとリンク・エディットされているが、以下のいずれも含んでいない アプリケーション。

- VS COBOL II プログラム
- IBM COBOL プログラム
- Enterprise COBOL プログラム
- IGZCBSN または IGZCBSO ブートストラップ・ルーチン
- IGZBRDGE ルーチンを使用するプログラム

CICS CICS のもとで実行されるアプリケーション。

NORES (リンク済み)

NORES を指定してコンパイルされたプログラムから構成されるアプリケーション。この NORES プログラムは、言語環境プログラムとリンク・エディットされており、現在は RES と同様に動作します。アプリケーションは、以下の少なくとも 1 つを含んでいます。

- VS COBOL II プログラム
- IBM COBOL プログラム
- Enterprise COBOL プログラム
- IGZCBSN または IGZCBSO ブートストラップ・ルーチン
- IGZBRDGE ルーチンを使用するプログラム

注: 複数ロード・モジュール・アプリケーションの場合、*最初*のロード・モジュールが上記の 1 つを含んでいると、アプリケーションは RES と同様に動作します。

リンク・エディットが必要なプログラムの判別

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	CICS	NORES (リンク済み)

言語環境プログラムとリンク・エディットすることが必要なプログラムを判別するには、以下のことを知る必要があります。

- プログラムが RES と NORES のどちらのコンパイラー・オプションを指定してコンパイルされたか
- プログラムが MIXRES ランタイム・オプションを指定しているかどうか
- プログラムが再使用可能環境 (ILBOSTP0 によって確立される) を使用するかどうか
- プログラムが ILC を使用するかどうか
- IGZCA2D または IGZCD2A を静的に呼び出すかどうか

リンク・エディットする必要がある、VS COBOL II プログラムを含むロード・モジュールに関する追加情報については、257 ページの 『VS COBOL II の考慮事項』を参照してください。

COBOL ロード・モジュールと言語環境プログラムとのリンク・エディットに関する追加情報については、349 ページの『付録 J. リンク・エディットの例』を参照してください。

RES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション

ILBOSTP0 を呼び出して再使用可能環境を確立する (つまり、アセンブラー言語プログラムを ILBOSTP0 とリンク・エディットし、ILBOSTP0 を呼び出すことによって、アセンブラー言語プログラムをメインプログラムとして確立する) 場合は、アセンブラー・プログラムを言語環境プログラムとリンク・エディットしなければならないか、またはアセンブラー・プログラムを変更して CEEPIPI を使用することができます。

NORES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション

NORES を指定してコンパイルされた既存の VS COBOL II プログラムは、変更する必要がなく、実行結果も以前と同じになります。これらのプログラムは、言語環境プログラムとリンク・エディットする必要はありません。ただし、これらのプログラムを言語環境プログラムとリンク・エディットしない場合は、これらの NORES アプリケーションのために IBM サービス・サポートを受けることはできません。

VS COBOL II の MIXRES または RTEREUS ランタイム・オプションを使用する、NORES を指定してコンパイルされたプログラムを含んでいるロード・モジュールは、言語環境プログラムとリンク・エディットしなければなりません。ロード・モジュールを言語環境プログラムとリンク・エディットすると、そのロード・モジュールは、以下の 2 つを除くすべてのケースで、すべての言語環境プログラム・サービス (呼び出し可能サービスを除く) にアクセスできるようになります。

1. **IGZBRDGE を使用しない OS/VS COBOL NORES:**

   プログラムが、IGZBRDGE マクロを用いて生成されたオブジェクト・モジュールとリンク・エディットされておらず、これがアプリケーション内の唯一のロード・モジュールであるか、または複数ロード・モジュール・アプリケーション内の最初のロード・モジュールである場合は、これは、言語環境プログラムとのリンク・エディット後に NORES 動作を示します。(つまり、このロード・モジュールは言語環境プログラム・サービスにアクセスできません。) このアプリケーションを言語環境プログラムとリンク・エディットしなければならない理由は、OS/VS COBOL 開始の終了 (STOP RUN など) が機能するようにするためです。

   このタイプのプログラムが言語環境プログラム・サービスにアクセスできるようにする必要がある場合は、アプリケーションに Enterprise COBOL プログラムを組み込むか、アプリケーションを言語環境プログラムとリンク・エディットするときに言語環境プログラムの IGZCBSN または IGZCBSO ブートストラップ・ルーチンを明示的に INCLUDE します。

2. **OS/VS COBOL RES と、IGZBRDGE を使用しない OS/VS COBOL NORES:**

   このロード・モジュールがアプリケーション内の唯一のロード・モジュールである場合、あるいは複数のロード・モジュールを持つアプリケーションにおける最初のロード・モジュールである場合は、このロード・モジュールを 言語環境プログラムとリンク・エディットするときに、言語環境プログラムの IGZCBSN または IGZCBSO ブートストラップ・ルーチンを明示的に INCLUDE する必要があります 。そうしなければ、このロード・モジュールはサポートされません。

さらに、マルチタスキング環境で実行される VS COBOL II プログラムを含んでいるロード・モジュールは、言語環境プログラムとリンク・エディットしなければなりません。

ILC を使用するプログラム

PL/I、C、または FORTRAN との ILC を使用する既存の VS COBOL II ロード・モジュールは、リンク・エディットする必要があります。詳細については、以下を参照してください。

- 111 ページの『COBOL と PL/I』
- 112 ページの『COBOL と C/370』
- 110 ページの『COBOL と FORTRAN』

IGZCA2D または IGZCD2A を静的に呼び出す場合

IGZCA2D または IGZCD2A を静的に呼び出す場合は、これらのモジュール用の REPLACE ステートメントを使用してアプリケーションを再リンクしなければなりません。

アップグレードが必要なプログラムの判別

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	NORES（リンク済み）

注: この章でアップグレードが必要と記載されている場合は、すべての OS/VS COBOL プログラムおよび VS COBOL II プログラムを Enterprise COBOL (または COBOL (OS/390 および VM 版) バージョン 2 リリース 2) にアップグレードすることを推奨します。

CICS

CICS のもとでは、CALL ステートメントを使用して OS/VS COBOL プログラムを呼び出す VS COBOL II プログラムがある場合は、その OS/VS COBOL プログラムをアップグレードする必要があります。

注: CICS では、OS/VS COBOL プログラムと VS COBOL II リリース 4 プログラムとの間の CALL ステートメントは、VS COBOL II ランタイムによって診断されます。この診断は、VS COBOL II リリース 3.2 およびリリース 3.1 の場合も提供されます (APAR PN18560 が適用されている場合)。これらのレベルで実行している場合は、CICS のもとでの COBOL CALL ステートメントを使用して OS/VS COBOL プログラムを呼び出す VS COBOL II プログラムはありません。

非 CICS

非 CICS では、既存の VS COBOL II プログラムを言語環境プログラムのもとで実行するために Enterprise COBOL にアップグレードする必要はありません。

以下の場合は、OS/VS COBOL プログラムをアップグレードする必要があります。

- OS/VS COBOL プログラムが PL/I との ILC を使用する場合
- OS/VS COBOL プログラムが FORTRAN との ILC を使用する場合
- OS/VS COBOL プログラムが複数のエンクレーブ (または実行単位) に含まれている場合

さらに、OS/VS COBOL プログラムや VS COBOL II プログラムで Enterprise COBOL または言語環境プログラムの機能 (言語環境プログラム呼び出し可能サービス、組み込み関数、または非 CICS の COBOL プログラムを z/OS または OS/390 のもとで同じアドレス・スペース内の複数のタスクで実行することなど) を使用する場合にも、そのプログラムのアップグレードが必要になります。

PL/I との ILC

OS/VS COBOL プログラムが PL/I を呼び出す場合、あるいは PL/I によって呼び出された場合は、そのプログラムをアップグレードする必要があります。

FORTRAN との ILC

RES を指定した OS/VS COBOL プログラムが FORTRAN を呼び出す場合、あるいは FORTRAN によって呼び出された場合は、そのプログラムをアップグレードする必要があります。

複数のエンクレーブに含まれている OS/VS COBOL プログラム

言語環境プログラムのもとでは、OS/VS COBOL プログラムが複数のエンクレーブに含まれていてはなりません。複数のエンクレーブは、いくつかの方法で作成される可能性があります。

- OS/VS COBOL プログラムがアセンブラー・プログラムを呼び出し、次にそれが別のプログラムへの SVC LINK を発行する場合
- RES を指定してコンパイルされたプログラムから構成される OS/VS COBOL アプリケーションを言語環境プログラムのもとで実行し、そのアプリケーションが ISPF サービス (CALL ’ISPLINK’ または ISPF SELECT など) を使用する場合

複数エンクレーブ・アプリケーションの場合、どのエンクレーブが OS/VS COBOL プログラムを含んでいるかを判別してください。1 つのエンクレーブだけが OS/VS COBOL プログラムを含むことができます。その他のエンクレーブに含まれている OS/VS COBOL プログラムは、すべてアップグレードする必要があります。

ランタイム・オプションおよび指定方式の比較

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	✓ NORES (リンク済み)

既存のアプリケーションを言語環境プログラムで実行するときには、VS COBOL II のランタイム・オプションが影響を受ける場合があります。以下のセクションでは、発生する可能性のある違いを説明します。言語環境プログラムの新しいランタイム・オプションおよび言語環境プログラムでサポートされる既存のランタイム・オプションに関する詳細については、*言語環境プログラム プログラミング・リファレンス* を参照してください。

注: 言語環境プログラムへの移行と並行して、Enterprise COBOL による VS COBOL II プログラムのコンパイルを行う場合は、251 ページの『第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加』を参照してください。

言語環境プログラム・ランタイム・オプションの指定

言語環境プログラムでは、ランタイム・オプションを指定するための各種の方式があります。オプションを指定するための 3 つのメカニズムが使用可能です。

- ランタイム・オプション CSECT
  - CEEDOPT (インストール・システム全体のデフォルト用)
  - CEEROPT (領域全体のデフォルト用)
  - CEEUOPT (アプリケーション固有のデフォルト用)
- 呼び出し手順
- アセンブラー・ユーザー出口

さらに、RES を指定してコンパイルされた VS COBOL II プログラムから構成されるロード・モジュールでは、VS COBOL II のアプリケーション固有ランタイム・オプション・モジュール (IGZEOPT) が使用可能です。IGZEOPT は、CICS のもとでは無視されます。

既存のアプリケーションのために使用できる言語環境プログラム方式を判別するには、メインプログラムが RES と NORES のどちらを指定してコンパイルされたか、およびメインプログラムが言語環境プログラムとリンク・エディットされているかどうかを知る必要があります。表 23 に、言語環境プログラムのもとでランタイム・オプションを指定およびオーバーライドするのに使用できる指定方式をリストします。

表 23. ランタイム・オプションを指定するために使用できる方式

メインプログラム	CEEDOPT および CEEROPT	CEEUOPT	呼び出し	デフォルト・アセンブラー・ユーザー出口	IGZEOPT
言語環境プログラムとリンク・エディットされていない:					
OS/VS COBOL RES	X		X	X	
OS/VS COBOL NORES			X		
VS COBOL II RES	X		X	X	X[1]
VS COBOL II NORES			X		
VS COBOL II NORES			X		X[2]
言語環境プログラムとリンク・エディットされている:					
OS/VS COBOL RES	X		X	X	
OS/VS COBOL NORES			X		
OS/VS COBOL NORES (リンク済み) [3]	X		X	X	
VS COBOL II NORES	X	X	X	X	
VS COBOL II RES	X	X	X	X	X[1,4]

表 23. ランタイム・オプションを指定するために使用できる方式 (続き)

メインプログラム	CEEDOPT および CEEROPT	CEEUOPT 呼び出し	デフォルト・アセンブラー・ユーザー出口	IGZEOPT

注:

1. IGZEOPT は、CICS での実行時には無視されます。
2. このケースでは、NORES プログラムはリンク・エディット後に「RES 動作」を示します。言語環境プログラムとのリンク・エディットの含意については、121 ページの 『第 8 章 アプリケーションと言語環境プログラムとのリンク・エディット』を参照してください。
3. IGZEOPT を用いて指定したランタイム・オプションが有効であるときの詳細については、95 ページの表 24 を参照してください。
4. MIXRES ランタイム・オプションを指定していないか、または言語環境プログラムとリンク・エディットされていない VS COBOL II NORES プログラムは、引き続き IGZEOPT を使用することができます。

インストール・システム全体のデフォルト

CEEDOPT (非 CICS) および CEECOPT (CICS) アセンブラー・ファイルには、すべての言語環境プログラム・ランタイム・オプションのデフォルト値が入っています。インストール時に、このファイルを編集し、共通環境で実行されるすべてのアプリケーションに適用されるデフォルトを選択することができます。

このモジュール内のオプションについては、オーバーライド不能 (固定) 属性を選択することができます。この属性により、インストール・システムでは、稼動環境全体にとって重要と見なされるオプションを適用することができます。

インストール・システム全体のデフォルト・ランタイム・オプションを設定する方法については、以下の資料を参照してください。

- z/OS の場合: *z/OS 言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

領域全体のデフォルト

CEEROPT を CEEDOPT および CEECOPT と同様の方法で使用して、ランタイム・オプションを指定することができます。CEEROPT を使用して、CICS 領域、IMS 領域、またはライブラリー・ルーチン保存を使用する MVS バッチ・ジョブ領域での言語環境プログラムのランタイム・オプション・デフォルトを設定することができます。CEEROPT はそれ自体のロード・モジュールに置かれるので、実行可能コードを含んでいるロード・モジュールに CEEROPT をリンクすることに関連した保守問題が回避されます。CEEROPT は、領域の初期設定時にロードされ、インストール・システム・ランタイム・オプション・デフォルトとマージされます。

CEEROPT はオプショナルです。環境の初期設定時に、 CEEROPT を検索します。見つかった場合は、CEEROPT 内に指定されているランタイム・オプションが、CEEDOPT または CEECOPT 内に指定されているインストール・システム・デフォルトとマージされます。

言語環境プログラムの領域全体のデフォルト・ランタイム・オプションを設定する方法については、*言語環境プログラム OS/390 版 カスタマイズ* を参照してください。

アプリケーション固有のデフォルト

言語環境プログラム提供のアセンブラー・ファイル CEEUOPT を編集して、特定のアプリケーションまたはプログラムに適用されるランタイム・オプションを選択することができます。

このファイルの編集後、アセンブルし、このファイルが適用される特定のアプリケーションまたはプログラムとリンク・エディットします。このモジュールで設定するオプションは、オーバーライド可能なインストール・システム全体のオプションに優先します。

CEEUOPT は、OS/VS COBOL プログラムをメインプログラムとして含んでいるアプリケーションでは使用できません。このことは、OS/VS COBOL プログラムを言語環境プログラムとリンク・エディットしているかどうかには関係なく該当します。

呼び出し手順

プログラム呼び出し時にオペレーティング・システムの適切なメカニズムを使用して、ランタイム・オプションを指定することもできます。(呼び出し時にランタイム・オプションを指定するには、COBOL プログラムがオペレーティング・システムから直接に呼び出されなければなりません。)

z/OS および OS/390 の場合

JCL EXEC ステートメントの PARM パラメーターを使用してランタイム・オプションを指定することができます。

注: CICS および IMS では、PARM パラメーターを使用してランタイム・オプションを指定することはできません。

詳細については、*言語環境プログラム プログラミング・ガイド* を参照してください。

優先順位

1 つのアプリケーションについてのランタイム・オプションを指定するために上記のすべてのメカニズムを使用することができます。言語環境プログラムでは、以下の優先順位規則が適用されます。

1. インストール時にオーバーライド不能として指定されたインストール・システム全体のデフォルト
2. アセンブラー・ユーザー出口を使用して指定されたオプション
3. 呼び出し時に指定されたオプション
4. アプリケーション固有のオプション (アプリケーションとリンク・エディットされている)
5. CEEROPT を使用して定義された領域全体のデフォルト
6. インストール時に定義されたインストール・システム全体のデフォルト

VS COBOL II ランタイム・オプションの指定

以下の情報は、既存のアプリケーションを Enterprise COBOL にアップグレードすることなく (または言語環境プログラムとリンク・エディットすることなく) 実行する場合に役立つように、互換性を援助する目的でのみ記載されています。アプリケーション内のプログラムが変更されるため、適切な言語環境プログラム・オプション・モジュールをアプリケーションに組み込むことをお勧めします。

RES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション (非 CICS)

言語環境プログラムのもとで実行される RES アプリケーションでは、VS COBOL II のアプリケーション固有ランタイム・オプション・モジュール IGZEOPT を使用できます。アプリケーションを言語環境プログラムとリンク・エディットした後は、IGZEOPT と CEEUOPT の両方を 1 つのアプリケーション内に共存させることができます。表 24 に、言語環境プログラムとリンク・エディットされたアプリケーションの場合の IGZEOPT と CEEUOPT 間の関係を示します。

表 24. VS COBOL II でコンパイルされ、言語環境プログラムとリンク・エディットされた既存のアプリケーションの場合のアプリケーション固有オプション

IGZEOPT が存在する	CEEUOPT が存在する	説明
いいえ	いいえ	ランタイム・オプションは影響を受けません。
はい	いいえ	IGZEOPT 内に指定されたランタイム・オプションが、適切な言語環境プログラム・ランタイム・オプションにマップされます。
いいえ	はい	CEEUOPT 内に指定されたランタイム・オプションが有効です。
はい	はい	CEEUOPT 内に指定されたランタイム・オプションが有効です。IGZEOPT は無視されます。エラー・メッセージは出されません。

このようなアプリケーションを言語環境プログラムとリンク・エディットするときに、IGZEOPT を CEEUOPT で置き換えることをお勧めします。

NORES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション (非 CICS)

VS COBOL II ランタイム・ライブラリーとリンク・エディットされた NORES アプリケーションでは、デフォルト VS COBOL II ランタイム・オプション・モジュール IGZEOPD を使用できます。MIXRES ランタイム・オプションが使用されていなければ、IGZEOPT も VS COBOL II NORES アプリケーションで使用できます。このようなアプリケーションが言語環境プログラムとリンク・エディットされると、IGZEOPD および IGZEOPT は無視されます。

CICS で実行されるアプリケーション

言語環境プログラムのもとでの実行時には、CICS で IGZEOPT は無視されます。CICS で実行中のアプリケーションで IGZEOPT が指定されると、言語環境プログラムは警告メッセージを出します。

VS COBOL II と言語環境プログラムのオプションの比較

次の表では、言語環境プログラムでの実行時に VS COBOL II ランタイム・オプションがサポートまたは変更される方法を説明します。動作の違いがある場合は、説明欄に記述します。

VS COBOL II ランタイムから言語環境プログラムへの移行

表 25. VS COBOL II と言語環境プログラムのランタイム・オプションの比較

VS COBOL II オプション	言語環境プログラム・オプション	説明
AIXBLD	AIXBLD	COBOL ルーチンについてのファイルおよび索引定義手順を完了するために、VSAM 索引付きおよび相対データ・セット用のアクセス方式サービス (AMS) を呼び出します。 言語環境プログラムの場合、OS/390 または z/OS での実行時に DD 名 SYSPRINT を指定する必要はありません。アクセス方式サービス (AMS) メッセージは、言語環境プログラムの MSGFILE ランタイム・オプションで指定された DD 名に送られます (デフォルトは SYSOUT です)。 AIXBLD は CICS では適用されません。
DEBUG	DEBUG	このオプションは、言語環境プログラムのもとで、VS COBOL II の場合と同じ構文で指定され、同じようにサポートされます。
LANGUAGE	NATLANG	VS COBOL II のインストール・オプションである LANGUAGE の代わりに、NATLANG が使用されます。言語環境プログラムでは、インストール時のほかに、NATLANG オプションを使用して実行時に各国語を選択することができます。
LIBKEEP		LIBKEEP はサポートされません。言語環境プログラムのライブラリー・ルーチン保存機能は、LIBKEEP と類似した機能を提供することができます。詳細については、 113 ページの『LIBKEEP を使用する既存のアプリケーション』を参照してください。 LIBKEEP オプションは CICS では適用されません。
MIXRES		MIXRES は、言語環境プログラムのもとではサポートされません。 MIXRES オプションは CICS では適用されません。
RTEREUS	RTEREUS	RTEREUS は、言語環境プログラムのもとで、VS COBOL II の場合と同じようにサポートされます。RTEREUS は、言語環境プログラムのもとではインストール先のデフォルトとして推奨されません。RTEREUS を使用する前の重要な考慮事項については、62 ページの 『非 CICS アプリケーションに影響を与えるその他のランタイム・オプション』および 99 ページの『IMS のもとで再使用可能環境を確立する場合の注意事項』を参照してください。 RTEREUS オプションは CICS では適用されません。
SIMVRD	SIMVRD	このオプションは、言語環境プログラムのもとで、VS COBOL II の場合と同じ構文で指定され、同じようにサポートされます。 SIMVRD オプションは CICS では適用されません。
SPOUT	RPTOPTS(ON) RPTSTG(ON)	SPOUT は、言語環境プログラム・オプション RPTOPTS および RPTSTG にマップされています。ストレージ報告書 (RPTSTG) とオプション報告書 (RPTOPTS) は両方とも、MSGFILE で指定された DD 名に送られます (デフォルトは SYSOUT です)。報告書の形式は、言語環境プログラムのもとでは VS COBOL II の場合と異なります。詳細については、 114 ページの『ストレージ調整の変更点の判別』を参照してください。
SSRANGE	CHECK(ON)	SSRANGE は、直接 CHECK にマップされています。

表 25. VS COBOL II と言語環境プログラムのランタイム・オプションの比較 (続き)

VS COBOL II オプション	言語環境プログラム・オプション	説明
STAE	TRAP(ON)	STAE は、直接 TRAP(ON) にマップされています。非 CICS アプリケーションの場合、TRAP(ON) によって ESTAE と ESPIE の両方が出されます。VS COBOL II のもとでは、STAE によって ESTAE のみが出されます。VS COBOL II のもとでは STAE はオプショナルであることに注意してください。言語環境プログラム リリース 1.9 (OS/390 V2R6) 以上のもとで、非 CICS では、TRAP(ON,NOSPIE) を指定することができ、その場合は ESTAE だけが出されます。
UPSI	UPSI	このオプションは、言語環境プログラムのもとで、VS COBOL II の場合と同じ構文で指定され、同じようにサポートされます。
WSCLEAR	STORAGE	WSCLEAR はサポートされません。類似した機能を得るためには、非 CICS アプリケーションの場合は STORAGE(00,NONE,NONE,8K) を使用し、CICS アプリケーションの場合は STORAGE(00,NONE,NONE,0K) を使用してください。

非 COBOL および OS/VS COBOL プログラムでのファイルのクローズ

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES (リンク済み)

z/OS または OS/390 の場合、ファイルをクローズしない OS/VS COBOL プログラムまたはアセンブラー・プログラムがあると、C03 異常終了が発生する可能性があります。バッチでは、COBOL メインプログラムがアセンブラー・プログラムから呼び出される場合、または COBOL がメインプログラムでない場合は、以下の変更のいずれかを行うことにより、C03 異常終了が発生しないようにする必要があります。

- ファイルをクローズするためのコードを追加する。
- VS COBOL II、IBM COBOL、または Enterprise COBOL プログラムを使用してファイルをオープンする。
- COBOL プログラムがアセンブラー・プログラムから呼び出され、そのアセンブラー・プログラムが z/OS または OS/390 バッチ・イニシエーターから起動される場合は、COBOL スタブ (IBM COBOL または Enterprise COBOL でコンパイルされたもの) をアプリケーションに追加してください。このようにすると、言語環境プログラムは、動的に呼び出されたロード・モジュールを終了時に解放しません (アセンブラー・プログラムおよびそれに含まれている入出力制御ブロックがストレージに残ります)。
- アセンブラー・プログラムがファイルをオープンする場合は、そのプログラムを変更して、データ管理制御ブロック用の GETMAIN を実行してください (それらの制御ブロックをロード・モジュールに残さないようにします)。

その他の環境

その他の環境 (TSO、IMS、または事前初期設定済み環境 (CEEPIPI プログラムや IGZERRE プログラム) など) で実行されるプログラムの場合は、エンクレーブ終了

の前に、OS/VS COBOL のすべてのファイルを明示的にクローズする (または、OS/VS COBOL プログラムをアップグレードする) 必要があります。再使用可能環境での実行時には、言語環境プログラムは、COBOL プログラムによってオープンされたままになっているファイルを自動的にクローズすることはしません。

COBOL によって呼び出されたアセンブラー・プログラムは、以下の場合は、エンクレーブ終了の前にファイルをクローズする必要があります。

- アセンブラー・プログラムが、COBOL 動的呼び出しによってロードされたロード・モジュールに含まれている場合。
- アセンブラー・プログラムが、アセンブラー・プログラム内または COBOL WORKING-STORAGE データ項目内でファイル制御ブロック用のストレージを割り振る場合。

再使用可能実行時環境での実行

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES (リンク済み)

言語環境プログラムは、再使用可能実行時環境を管理する目的で行われる、言語環境プログラム以前の方式の互換性サポートを引き続き提供します。その方式とは、次の 3 つです。

- アセンブラー・プログラムから IGZERRE を呼び出す。
- アセンブラー・プログラムから ILBOSTP0 を呼び出す。
- RTEREUS ランタイム・オプションを指定する。

言語環境プログラムのもとで再使用可能環境を既存のアプリケーションと共に使用する場合は、以下の制約事項が適用されます。

- IGZERREO CSECT を使用して動作を変更しない限り、アプリケーションは、単一エンクレーブ・アプリケーションでなければなりません。

  注: IGZERREO CSECT を使用して動作を変更しても、OS/VS COBOL が複数のエンクレーブに含まれていてはならないという制約事項は除去されません。
- 再使用可能環境の確立後は、アセンブラー・ドライバーは COBOL プログラムまたは言語環境プログラム準拠でないアセンブラー・プログラムだけを呼び出すことができます。次いで、COBOL はサポートされる言語を呼び出すことができます。
- 最初の高水準言語プログラムは COBOL プログラムでなければなりません (アセンブラーは高水準言語と見なされません)。
- アセンブラー・ドライバーが再使用可能環境を提供するために IGZERRE または ILBOSTP0 と静的にリンクされており、それを呼び出す場合は、言語環境プログラムとリンク・エディットされていなければなりません。
- RTEREUS ランタイム・オプションを有効にして COBOL NORES プログラムを実行する場合は、そのプログラムは言語環境プログラムとリンク・エディットされていなければなりません。

STOP RUN を使用すると、再使用可能環境は、その初期設定に使用された方式に関係なく終了します。IGZERRE または ILBOSTP0 がアセンブラー・プログラムによ

って呼び出された場合は、制御はアセンブラー・プログラムの呼び出し元に戻されます。言語環境プログラムの RTEREUS ランタイム・オプションを使用した場合は、制御は最初の COBOL プログラムの呼び出し元に戻されます。

IGZERREO CSECT を変更すれば、COBOL 再使用可能環境を使用するときのパフォーマンスを向上させることができ、また、再使用可能環境での実行時にネストされたエンクレーブを使用することができます。詳細については、以下の資料を参照してください。

- z/OS の場合: *z/OS 言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

IMS のもとで再使用可能環境を確立する場合の注意事項

再使用可能環境は、IMS のもとでは推奨されません。再使用可能環境を開始することを選択する場合は、次のことに注意してください。

- 再使用可能環境の使用は、アプリケーションが少数のプログラムから構成される場合に限ってください。そのようにしないと、ストレージが増えていき、IMS 領域を減少させ、最終的に IMS 領域を停止させます。
- IMS から制御を受け取るすべてのプログラム (COBOL メインまたはアセンブラーなど) をプリロードしてください。プリロードを行わないと、結果は予測できません。

RTEREUS の使用の詳細については、59 ページの『非 CICS アプリケーションについて推奨されるランタイム・オプション』を参照してください。

IGZERRE の使用

引き続き IGZERRE を使用して、COBOL の初期設定機能と終了機能を明示的に制御することができます。ただし、3 つの戻りコードの変更点を理解しておくことが必要です。

IGZERRE からの戻り時には、レジスター 15 に戻りコードが入っています。表 26 に、言語環境プログラムでの実行時の変更点を示します。

表 26. IGZERRE についての戻りコードの変更点

戻りコード	説明
0	無変更
4	言語環境プログラムがすでに初期設定されている場合に出されます (以前は、VS COBOL II がすでに初期設定されている場合に出されました)
8	無変更
12	出されなくなりました (NORES 環境の初期設定にのみ適用されていました)
16	無変更
20	出されなくなりました (COBTEST の使用に適用されていました)

ILBOSTP0 の使用

非 COBOL プログラムをメインプログラムとして使用する既存のアプリケーションが引き続き稼働し、同じ結果をもたらすように、ILBOSTP0 が引き続きサポートされます。

ILBOSTP0 を使用して COBOL 再使用環境を初期設定する場合、言語環境プログラムは、アプリケーション固有のルーチン CEEUOPT がロード・モジュールとリンクされていない限り、自動的に ALL31(OFF) および STACK(,,BELOW) を設定します。

CEEUOPT をロード・モジュールとリンクする場合は、それに ALL31(OFF) および STACK(,,BELOW) を指定するようにしなければなりません。

インストール・システム全体のデフォルト・ルーチン CEEDOPT で ALL31(ON) や STACK(,,ANYWHERE) がオーバーライド不能として指定されている場合、言語環境プログラムはメッセージ CEE3615I によってこの矛盾を診断します。このような矛盾があるアプリケーションを実行すると、結果は予測できません。

RTEREUS の使用

COBOL 再使用可能環境を初期設定するための RTEREUS ランタイム・オプションは言語環境プログラムによってサポートされますが、インストール先のデフォルトとしては推奨されません。重要な考慮事項については、59 ページの『非 CICS アプリケーションについて推奨されるランタイム・オプション』の RTEREUS の説明を参照してください。RES 環境では、RTEREUS の動作は、SVC LINK の使用 (たとえば、ISPF ″SELECT PGM()″) を除き、VS COBOL II の場合と同じです。

言語環境プログラムのもとでは、再使用環境での実行時にアセンブラー・プログラムが SVC LINK を発行すると、言語環境プログラムは、重大度 3 のメッセージ IGZ0168S が出されることになる条件を起こします。VS COBOL II では、エラーは検出されませんが、2 次実行単位の環境は再使用可能ではありません。IGZERREO CSECT を変更すれば、言語環境プログラムのもとでの動作を、VS COBOL II のもとでの動作に類似するように変更することができます。ただし、ネストされたエンクレーブ内の STOP RUN は、ネストされたエンクレーブだけを終了させ、親エンクレーブを終了させません。

最初の COBOL プログラムの呼び出し元がオペレーティング・システムである場合は、RTEREUS は無視されます。

RTEREUS は、インストール・システム全体のデフォルト・ルーチン CEEDOPT またはアプリケーション固有オプション・ルーチン CEEUOPT を用いて指定するか、またはアセンブラー・ユーザー出口で指定することができます。

IMS の重要な考慮事項については、99 ページの『IMS のもとで再使用可能環境を確立する場合の注意事項』を参照してください。

メッセージ、異常終了コード、およびダンプ・サービスの管理

このセクションは以下のものに適用されます。

✓ RES	NORES	✓ CICS	✓ NORES（リンク済み）

アプリケーションを言語環境プログラムのもとで実行する場合、メッセージ、異常終了コード、およびダンプ・サービスは、VS COBOL II ライブラリーや OS/VS COBOL ライブラリーのもとで実行する場合とは異なる方法で管理されます。

ランタイム・メッセージ

以下の 2 つの要因が、言語環境プログラムのもとで実行されるアプリケーションのランタイム・メッセージに影響を与えます。

- メッセージが OS/VS COBOL 互換ライブラリー・ルーチンから発行されるかどうか (接頭語 IKF)
- COBOL 特有のライブラリー・ルーチン (VS COBOL II 互換ルーチンを含む) によって発行されるメッセージ (接頭語 IGZ) の場合、言語環境プログラム・ランタイム初期設定プロセスの状況

接頭語 IKF を持つメッセージ

OS/VS COBOL メッセージの場合、違いはありません。メッセージの接頭語、番号、重大度、および内容は未変更のままです。さらに、プログラムが言語環境プログラムとリンク・エディットされていても、宛先は同じままです。メッセージは同期的に発行され、OS のプログラマー向け書き込みを用いて書き込まれます。これは、ランタイム初期設定プロセスの状態に関係なく該当します。

接頭語 IGZ を持つメッセージ

VS COBOL II メッセージは、言語環境プログラム・ランタイム初期設定プロセスの状況に応じて管理されます。

- 非 CICS では、ランタイム初期設定プロセスが完了していない場合は、メッセージは、OS のプログラマー向け書き込みによって指示されたとおりに送られます。
- 非 CICS では、ランタイム初期設定プロセスが完了している場合は、メッセージは、言語環境プログラムの MSGFILE ランタイム・オプションで指定されたファイル *ddname* (デフォルトは SYSOUT) に送られます。
  - z/OS および OS/390 の場合: SYSOUT=*

  MSGFILE ddname が定義されていない場合は、言語環境プログラム定義のファイル属性を用いて動的に割り振られます。詳細については、*言語環境プログラム プログラミング・ガイド* を参照してください。
- CICS では、言語環境プログラムからのランタイム出力はすべて、CESE という名前の CICS 一時データ・キューに送られます。ファイルに書き込まれた各レコードには、端末 ID、トランザクション ID、日付、および時刻を示すヘッダーがあります。ランタイム出力は一時記憶キューに書き込まれないようになりました。

  注: ランタイム出力が一時記憶キューに送られるようにするには、一時データ・キュー CESE を区画内データ・キューとして定義しておき、CESE からレコ

ードを読み取り、ヘッダー情報を取り除き、レコードの残りの部分を一時記憶キューに書き込むトランザクションを指定します。

すべての COBOL 特有メッセージは、IGZ の接頭語が付けられ、その後に 4 桁のメッセージ番号と、重大度のレベルが続きます。たとえば、VS COBOL II のメッセージ番号 IGZ020I は、言語環境プログラムのもとではメッセージ番号 IGZ0020W です。VS COBOL II の通知メッセージは、言語環境プログラムの重大度 1 警告メッセージになります。VS COBOL II の重大メッセージ (1xxx ユーザー異常終了になる) は、言語環境プログラムの重大度 3 重大条件または重大度 4 クリティカル条件になります。

重大度 1 (W)
警告メッセージ。言語環境プログラムのデフォルトの条件処理処置では、メッセージをフォーマット設定および発行し、プログラム実行を継続します。

重大度 2 (E)
エラー・メッセージ。言語環境プログラムのデフォルトの条件処理処置では、条件をパーコレートし、メッセージを発行し、エンクレーブを終了させます。

重大度 3 (S)
重大エラー・メッセージ。言語環境プログラムのデフォルトの条件処理処置では、条件をパーコレートし、メッセージを発行し、エンクレーブを終了させます。

重大度 4 (C)
クリティカル・エラー・メッセージ。言語環境プログラムのデフォルトの条件処理処置では、条件をパーコレートし、メッセージを発行し、エンクレーブを終了させます。

ランタイム検出エラーの場合の異常終了のタイミング

ランタイム検出エラー (IGZ0061S (0 除算) または IGZ0006S (範囲外の添え字) など) の場合、異常終了のタイミングは言語環境プログラムと VS COBOL II 間で異なります。異常終了のタイミングの違いは、CICS の HANDLE ABEND の動作に影響を与えます。

言語環境プログラムのもとでは、ランタイム検出エラーがあると、以下のイベントが発生します (ABTERMENC(ABEND) が有効の場合)。

1. 言語環境プログラムのソフトウェア生成条件が通知されます。
2. 登録されているユーザー条件処理ルーチンがある場合、言語環境プログラムの条件マネージャーは、そのユーザー条件処理ルーチンに制御を渡します。
3. 条件が処理されなかった場合は、エンクレーブが終了し、言語環境プログラムが 4038 異常終了を発行します。

VS COBOL II では、ランタイム検出エラーがあると、以下のイベントが発生します。

1. エラー・メッセージが書き込まれます。
2. 異常終了 1xxx が発行されます。

言語環境プログラムのもとでは、ランタイム検出エラーが発生すると、異常終了が発行される前に、そのコードを含んでいるエンクレーブ (実行単位) が終了します。そのため、CICS の HANDLE ABEND コマンドで参照されたラベルにあるコードは制御を取得しません。

VS COBOL II のもとでは、実行単位がまだ存在するときに異常終了が発行されるため、HANDLE ABEND のラベルにあるコードが実行されます。

VS COBOL II と互換性のある動作のためには、言語環境プログラムで SCEESAMP データ・セット内に提供されているサンプルのユーザー条件処理ルーチン・コード CEEWUCHA を使用してください。

言語環境プログラムのもとで CEEWUCHA を使用しているときは、ランタイム検出エラーがあると、以下のイベントが発生します (ABTERMENC(ABEND) が有効の場合)。

1. 言語環境プログラムのソフトウェア生成条件が通知されます。
2. 登録されているユーザー条件処理ルーチンがある場合、言語環境プログラムの条件マネージャーは、そのユーザー条件処理ルーチンに制御を渡します。
3. CEEWUCHA ユーザー条件処理ルーチンが制御を取得し、以下のことを発生させます。
   - エラー・メッセージが書き込まれます。
   - 言語環境プログラム・ダンプが生成されます。
   - 異常終了 1xxx が発行されます。

異常終了コード

未処理の言語環境プログラム・ソフトウェア生成条件があるときに、VS COBOL II のもとで受け取ったものと同様のユーザー異常終了 1xxx (xxx は IGZ メッセージ番号) を受け取るためには、以下のいずれかを行ってください。

- COBOL サンプル・アセンブラー・ユーザー出口 (CEEBX05A) からのコードをアセンブラー・ユーザー出口 (CEEBXITA) にコピーすることにより、CEEBXITA を変更してください。
- 『CEEWUCHA の使用』で説明されているように、サンプルのユーザー条件処理ルーチン CEEWUCHA を使用してください。

CEEWUCHA の使用

CEEWUCHA は、言語環境プログラムのデフォルト動作を変更して VS COBOL II に類似した動作を得るために使用できるサンプルのユーザー条件処理ルーチンです。

CEEWUCHA には、以下のことを行うコードが含まれています。

- ランタイム検出エラー (IGZ0061S (0 除算) など) の発生時に EXEC CICS HANDLE ABEND LABEL ステートメントが制御を取得するようにすることによって、CICS のもとで実行中の既存の VS COBOL II アプリケーションとの互換性を確保する。
- すべての未処理のランタイム検出エラーを、VS COBOL II によって発行される対応するユーザー異常終了 1xxx に変換する。

- すべての IGZ0014W メッセージ (これらは、IGZETUN または IGZEOPT が VS COBOL II アプリケーションとリンク・エディットされているときに生成される) を抑制する。

CEEWUCHA を使用するには、次のようにします。

1. サンプル SMP ジョブ CEEWWCHA を使用して、CEEWUCHA をアセンブルおよびリンク・エディットします。
2. CICS で、CICS 領域用の PPT 内に CEEWUCHA を定義します。
3. 以下のいずれかにおいて、USRHDLR(CEEWUCHA) を指定します。
   - CICS の場合、CEECOPT (CICS 領域全体に適用するため)
   - 非 CICS の場合、CEEDOPT (すべてのアプリケーションに適用するため)
   - CICS または 非 CICS の場合、CEEUOPT に指定し、それを個々のアプリケーションとリンク・エディットする

ダンプ・サービス

ダンプ・サービスは言語環境プログラムによって管理されます。ここでは、シンボリック・ダンプ、定様式ダンプ、およびシステム・ダンプについての変更点を説明します。

OS/VS COBOL シンボリック・ダンプ

NORES コンパイラー・オプションを指定してコンパイルされた OS/VS COBOL プログラムの場合、SYMDMP を用いて生成されたダンプ出力は同じままであり、引き続き SYSDBOUT に送られます。RES コンパイラー・オプションを使用した場合は、代わりに CEEDUMP が生成されます。

VS COBOL II の FDUMP

FDUMP オプションを指定してコンパイルされた VS COBOL II プログラムの場合、COBOL WORKING-STORAGE の定様式ダンプを入手することができます。VS COBOL II の FDUMP は、SYSDBOUT データ・セットに送られていました。言語環境プログラムのもとでのダンプ宛先の詳細については、104 ページの 『言語環境プログラムの定様式ダンプ』を参照してください。

OPT および FDUMP を指定してコンパイルされ、ODO データ構造を含んでいる VS COBOL II プログラムが言語環境プログラムのもとで実行され、異常終了すると、レベル 1 項目が言語環境プログラムの定様式ダンプ内のグループ項目としてダンプされます。

Test(SYM) の使用: Enterprise COBOL の TEST(SYM) コンパイラー・オプションは、VS COBOL II の FDUMP コンパイラー・オプションと同じ機能を提供します。TEST コンパイラー・オプションの SYM サブオプションを指定すると、言語環境プログラムは、FDUMP 出力と類似した出力を言語環境プログラムの定様式ダンプの一部として提供することができます。

言語環境プログラムの定様式ダンプ

ダンプ出力のダンプ形式および宛先は、言語環境プログラムのもとでは VS COBOL II の場合と異なります。

言語環境プログラムの定様式ダンプには、以下のものに関する情報など、さまざまな情報を含めることができます。

- 変数、引き数、およびレジスターのシンボリック・ダンプ
- ファイル制御ブロック
- ファイル・バッファー
- ランタイム制御ブロック
- プログラムによって使用されたストレージ

言語環境プログラムのダンプ出力の例については、*言語環境プログラム デバッグのガイドおよびランタイム・メッセージ* を参照してください。

言語環境プログラムのもとでは、レベル 2 以上の条件が発生し、未処理のままになっているときに言語環境プログラムの定様式ダンプを入手するためには、TERMTHDACT(DUMP) または TERMTHDACT(UADUMP) ランタイム・オプションを指定してください。

非 CICS の場合のダンプ宛先: 言語環境プログラムでは、JCL または FILEDEF で CEEDUMP の DD 名を指定することによって、ダンプ出力の宛先を指示することができます。CEEDUMP ファイルの属性については、*言語環境プログラム プログラミング・ガイド* を参照してください。

CEEDUMP ファイルが必要であるが、定義されていない場合、言語環境プログラムは以下のデフォルトを使用してこのファイルを動的に割り振ります。

- z/OS および OS/390 の場合: SYSOUT=*

CICS の場合のダンプ宛先: 言語環境プログラムからのランタイム出力はすべて、CESE という名前の CICS 一時データ・キューに送られます。ファイルに書き込まれた各レコードには、端末 ID、トランザクション ID、および日時を示すヘッダーがあります。

システム・ストレージ・ダンプ

システム・ストレージ・ダンプに含まれる PSW (プログラム状況ワード) 情報とレジスター情報は、言語環境プログラムのもとでの実行時と VS COBOL II ランタイムのもとでの実行時とでは異なります。

VS COBOL II では、ダンプはエラー発生時に生成されます。PSW およびレジスターの内容は、エラー発生時に使用可能な情報に基づきます。このため、ダンプ内の PSW 情報とレジスター情報を問題判別に使用することができます。

言語環境プログラムでは、ダンプは、言語環境プログラムの条件管理がエラーを処理した後で生成されます。PSW およびレジスターの内容は、エラー発生時に使用可能な情報に基づくのではなく、言語環境プログラムがエラーの処理を完了した時点で使用可能な情報に基づきます。エラー発生時に PSW およびレジスターに入っていた情報を判別するには、言語環境プログラムの定様式ダンプ出力を調べるか、または言語環境プログラム条件情報ブロックおよび言語環境プログラム・マシン状態情報ブロックを使用することができます。詳細については、*言語環境プログラム デバッグのガイドおよびランタイム・メッセージ* を参照してください。

システム・ストレージ・ダンプを入手するには、68 ページの『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。

z/OS および OS/390 では、ダンプ出力は SYSUDUMP、SYSMDUMP、または SYSABEND に送られます。その他のシステムでは、VS COBOL II ランタイム・ダ

ンプ出力は SYSABOUT に送られます (SYSABOUT は、言語環境プログラムのもとでは、VS COBOL II プログラムに対しても使用されません)。

ILBOABN0 による異常終了の強制

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

OS/VS COBOL および VS COBOL II のもとでは、ILBOABN0 への CALL を使用して即時異常終了を強制し、システム・ダンプを入手することができます。言語環境プログラムのもとでは、ILBOABN0 への CALL は引き続き即時異常終了を強制します。システム・ダンプを生成するには、68 ページの 『システム・ダンプまたは CICS トランザクション・ダンプの入手』を参照してください。

言語環境プログラム版の ILBOABN0 を使用すると、ILBOABN0 を発行するプログラムの保管域は、実際の異常終了が発行された保管域の 2 レベル後ろに置かれます。

ILBOABN0 への CALL を使用するプログラムを Enterprise COBOL に移行した場合、プログラムは引き続き ILBOABN0 を呼び出すことができます。ただし、その代わりに言語環境プログラムの CEE3ABD 呼び出し可能サービスを使用することをお勧めします。

SORT または MERGE の使用

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

言語環境プログラムのインプリメンテーションにより、OS/VS COBOL プログラムおよび VS COBOL II プログラムの SORT または MERGE を管理するルーチンが変更されました。

OS/VS COBOL プログラムの場合

OS/VS COBOL プログラムが SORT または MERGE を初期設定した場合は、以下の変更点が適用されます。

- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックまたは異常終了が発生した場合、言語環境プログラムは言語環境プログラム・ダンプを生成しません。
- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックまたは異常終了が発生した場合、OS/VS COBOL デバッグ・データは生成されません。
- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックまたは異常終了が発生した場合、言語環境プログラムは環境の終結処理を実行しません。
- 入力または出力プロシージャー内にあるときにアセンブラー・プログラムが呼び出された場合、そのアセンブラー・プログラムは COBOL プログラムを呼び出すことができません。

- SORT または MERGE ユーザー出口内にあるときにプログラム・チェックが発生した場合、アプリケーションは異常終了コード U4036 で終了します。プログラム・チェック発生時のレジスターおよび PSW の状態を調べる方法については、*言語環境プログラム デバッグのガイドおよびランタイム・メッセージ* を参照してください。

VS COBOL II サブプログラムの場合

VS COBOL II サブプログラムにおいて SORT または MERGE ステートメントと共に使用した入力または出力プロシージャーで GOBACK ステートメントを使用した場合、言語環境プログラムのもとでは、VS COBOL II ランタイムのもとでの場合と異なる動作を示します。

VS COBOL II プログラムを言語環境プログラムのもとで実行しているとき、動作の違いが発生するのは以下の場合です。

- COBOL サブプログラムが、入力プロシージャーまたは出力プロシージャーを指定して SORT または MERGE ステートメントを発行した。
- SORT または MERGE ステートメントの実行中に入力プロシージャーまたは出力プロシージャーが呼び出されたあと、SORT または MERGE ステートメントを開始したコンパイル単位の中で GOBACK が行われた。

言語環境プログラムのもとでは、重大条件 IGZ0012S が発行されます。VS COBOL II のもとでは、サブプログラムはエラーなしで呼び出し元に戻ります。

以下に例を示します。

```
⋮
    SORT SD01
       ASCENDING KEY A3
       USING INP1
       OUTPUT PROCEDURE OUTPRO1.
⋮
OUTPRO1 SECTION.
⋮
    GOBACK.
    * With Language Environment, this GOBACK statement will
    * cause condition IGZ0012S.
```

注: 上記の例で、VS COBOL II の場合 (言語環境プログラムの場合と同様に)、SORT または MERGE を発行する COBOL プログラムがメインプログラムであれば、エラー・メッセージが生成されます。

SYSOUT 出力の変更点の理解

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES (リンク済み)

言語環境プログラムのもとでは、DISPLAY UPON SYSOUT は、以下の点で OS/VS COBOL および VS COBOL II の場合と異なります。

- RECFM=FB のファイルに送られるか、または他の宛先に送られる (DD に DCB=(RECFM=FB) が指定されているとき) SYSOUT 出力
- OS/VS COBOL トレース出力シーケンス

DISPLAY UPON SYSOUT および DD 定義

言語環境プログラムは、DISPLAY UPON SYSOUT を VS COBOL II と異なる方法で処理します。

- VS COBOL II のもとでは、DD カードが欠落している場合は、VS COBOL II は IGZ メッセージおよび異常終了を発行します。言語環境プログラムのもとでは、言語環境プログラムが DD を割り振り、DISPLAY が正常に実行されます。
- VS COBOL II のもとでは、DISPLAY を使用する VS COBOL II プログラム内で OUTDD の値が異なる場合は、DISPLAY を含んでいる最初に検出されたプログラムからの DD が、アプリケーション全体を通じて使用される DD です。言語環境プログラムのもとでは、ユーザーが指定する固有 OUTDD のそれぞれについて DD が割り振られます。

RECFM=FB の場合の SYSOUT 出力

言語環境プログラムのもとでは、RECFM が FB であるファイルに送られるか、または他の宛先に送られる (DD に DCB=(RECFM=FB) が指定されているとき) TRACE/EXHIBIT 出力 (OS/VS COBOL プログラムの場合) および DISPLAY UPON SYSOUT 出力 (OS/VS COBOL プログラムおよび VS COBOL II プログラムの場合) の場合、以下に示されているように、先頭文字が出力に組み込まれません。

レコード・フォーマット	VS COBOL II の場合	言語環境プログラムの場合
RECFM=FBA	$01234	$01234
RECFM=FB	$01234	01234

「$」は、桁 1 にある制御文字を表します。DISPLAY のデフォルト属性は変更されません。

ファイルに送られる SYSOUT の場合、RECFM=FBA を使用することをお勧めします。

OS/VS COBOL トレース出力シーケンス

言語環境プログラムのもとでは、トレース出力シーケンスが変更されました。言語環境プログラムでは、VS COBOL II のもとで実行中の OS/VS COBOL プログラムの場合のように 1 つのレコードにつき複数のトレース記入項目が現れるのではなく、1 つのレコードにつき *1* つのトレース記入項目だけが現れます。この変更は、介在する DISPLAY SYSOUT 出力およびエラー・メッセージの場合に特に役立ちます。

109 ページの図 6 に、OS/VS COBOL と言語環境プログラム間の形式の違いを示します。

```
OS/VS COBOL (VS COBOL II のもとで実行) :

     Record 1 -->  Section1(2), Paragraph1(2), Section2(2)

言語環境プログラム :

     Record 1 -->  Section1
     Record 2 -->  Section1
     Record 3 -->  Paragraph1
     Record 4 -->  Paragraph1
     Record 5 -->  Section2
     Record 6 -->  Section2
```

図 6. 言語環境プログラムにおけるトレース出力 (OS/VS COBOL との比較)

他の言語との通信

このセクションは以下のものに適用されます。

✓ RES	✓ NORES	✓ CICS	✓ NORES (リンク済み)

ここでは、VS COBOL II ランタイムのもとで実行中の既存の OS/VS COBOL プログラムおよび VS COBOL II プログラムに関する ILC の考慮事項の高度な概説を示します。言語環境プログラムのもとでの ILC に関する詳細および最新情報については、*言語環境プログラム ILC (言語間通信) アプリケーションの作成* を参照してください。

言語間通信とは、他の高水準言語を呼び出すかまたは他の高水準言語によって呼び出されるプログラムとして定義されています。アセンブラーは高水準言語と見なされません。このため、アセンブラー言語プログラムへの呼び出しおよびアセンブラー言語プログラムからの呼び出しは ILC と見なされません。COBOL プログラムとアセンブラー・プログラムに関係のある呼び出しに関する言語環境プログラムのサポートの詳細については、以下を参照してください。

- 305 ページの『非 CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』
- 307 ページの『CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』

言語間通信 (ILC) を使用する既存のアプリケーションは、言語環境プログラムのもとでは実行できない場合があります。使用される言語を含め、その決定要因がいくつかあります。以下のセクションでは、ILC を使用する既存のアプリケーションについての含意を説明します。

CICS の場合、引き続き EXEC CICS LINK および EXEC CICS XCTL を使用して、他の言語と通信することができます。

ILC に関する一般的な考慮事項

COBOL の DISPLAY 出力 (非 CICS)

COBOL-C ILC アプリケーションを言語環境プログラムのもとで実行する場合、言語環境プログラムが終了するまで、SYSOUT DD はオープンされたままになります。このことは、以下のシナリオを含んでいるすべての ILC アプリケーションに影響を与えます。

1. C メインプログラムが COBOL プログラムを呼び出します。
2. COBOL プログラムが DISPLAY ステートメントを使用します。DISPLAY ステートメントからの出力が、SYSOUT DD に関連したデータ・セットに書き込まれます。COBOL プログラムが C プログラムに戻ります。
3. C プログラムが、SYSOUT DD に関連したデータ・セットをオープンし、書き込まれているレコードを読み取ります。

上記のシナリオは、VS COBOL II および言語環境プログラム以前の C ライブラリーと共に機能しました。VS COBOL II プログラムがメインプログラムであったからです (そのため、COBOL プログラムの終了時に VS COBOL II ランタイムが SYSOUT DD をクローズしたからです)。言語環境プログラムのもとでは、この COBOL プログラムはサブプログラムであり、COBOL プログラムが C プログラムに戻ったときに、言語環境プログラムは SYSOUT DD をクローズしません。

STOP RUN の影響 (非 CICS)

言語環境プログラムのもとでは、COBOL プログラムが C、PL/I、または FORTRAN プログラムから呼び出されたあとで STOP RUN を出すと、環境全体が終了します。VS COBOL II のもとでは、COBOL ランタイムだけが終了し、呼び出しプログラム (C、PL/I、または FORTRAN) は引き続き実行されます。

COBOL と FORTRAN

注: このセクションは CICS には適用されません。CICS で実行される FORTRAN プログラムは IBM でサポートされていません。

言語環境プログラムとリンク・エディットされている場合、以下の COBOL リリースと FORTRAN リリース間で ILC がサポートされます。

- VS COBOL II リリース 3 以上 (静的 CALL ステートメントのみ)
- IBM COBOL
- Enterprise COBOL
- VS FORTRAN バージョン 1 (リリース 1 またはリリース 2 でコンパイルされ、文字引き数を受け取るサブプログラムであるモジュールかまたは文字引き数をサブプログラムに渡すモジュールを除く)
- 以下の VS FORTRAN バージョン 2 (リリース 5 または リリース 6 でコンパイルされたモジュールを除く)
  - 並列言語構造体を含んでいる
  - PARALLEL コンパイラー・オプションを指定する
  - 並列呼び出し可能サービス (PEORIG、 PEPOST、 PEWAIT、 PETERM、 PLCOND、 PLFREE、 PLLOCK、 PLORIG、または PLTERM) を呼び出す
- FORTRAN IV G1

- FORTRAN IV H 拡張

言語環境プログラムでは、RES を指定した OS/VS COBOL プログラムが FORTRAN を呼び出す場合、あるいは FORTRAN によって呼び出される場合、このプログラムはサポートされません。

COBOL と FORTRAN 間の呼び出しを含んでいる既存の NORES アプリケーションは、既存の COBOL と FORTRAN との間の ILC 規則が守られる限り、引き続き以前と同様に稼働します。ただし、これらのアプリケーションは、FORTRAN プログラムが言語環境プログラムとリンク・エディットされるとサポートされません。

COBOL と PL/I

ユーザーが次のリンク・エディット要件に従う場合は、言語環境プログラムは以下の組み合わせの COBOL と PL/I 間の ILC をサポートします。

- VS COBOL II バージョン 1 リリース 3 以上
- IBM COBOL
- Enterprise COBOL
- OS PL/I バージョン 1 リリース 5.1
- OS PL/I バージョン 2
- PL/I for MVS & VM

既存のサポートされる PL/I-COBOL ILC アプリケーションは、OS PL/I バージョン 2 リリース 3 での実行時に PL/I 移行ツール (APAR PN46223) とリンク・エディットするか、言語環境プログラムとリンク・エディットする必要があります (PL/I 移行ツールが機能するためには、適切な USERMOD、COBOL サービス、および PL/I サービスもインストールする必要があります。詳細については、*IBM PL/I for MVS & VM コンパイラーと実行時の移行の手引き* を参照してください)。言語環境プログラムとリンク・エディットしなければならない以下のケースを除き、どちらの方式でも使用することができます。

- アプリケーションが COBOL NORES プログラムを含んでいる
- アプリケーションが COBOL プログラム特有のランタイム・オプションまたはスペース調整を必要とする
- アプリケーションが、SORT を使用する PL/I プログラムを含んでいる
- アプリケーションが PL/I 共用ライブラリーを使用する

注: OS/VS COBOL と PL/I 間の ILC はサポートされません。PL/I との ILC を含んでいる OS/VS COBOL プログラムは、アップグレードする必要があります。

動的に呼び出される RENT プログラムの場合の動作の違い

VS COBOL II プログラムでは、VS COBOL II のランタイムのもとで、それぞれの呼び出しごとに WORKING-STORAGE の別のコピーが使用されます。これに対し、言語環境プログラムでは、それぞれの呼び出しごとに WORKING-STORAGE の同じコピーが使用されます。このような状況の違いは、以下の条件がすべて該当する場合に発生します。

- RENT オプションを指定してコンパイルされた
- OS/VS COBOL、VS COBOL II、IBM COBOL、または Enterprise COBOL のプログラムから動的に呼び出された

- PL/I によって取り出され、呼び出された

さらに、VS COBOL II ランタイム・ライブラリーのもとで実行されるプログラムでは、初期状態が使用されます。これに対し、言語環境プログラムで実行されるプログラムの場合は、CANCEL による介入がない限り、最後に使われた状態が使用されます。

COBOL と PL/I の間の言語間通信に関する詳細については、*言語環境プログラム ILC (言語間通信) アプリケーションの作成* および *PL/I for MVS & VM コンパイラーと実行時の移行の手引き* を参照してください。

COBOL と C/370

ユーザーがリンク・エディット要件に従う場合は、言語環境プログラムは以下の組み合わせの COBOL プログラムと C プログラム間の ILC をサポートします。

- VS COBOL II バージョン 1 リリース 3 以上
- IBM COBOL
- Enterprise COBOL
- C/370 バージョン 1 (5688-040)
- C/370 バージョン 2 (5688-187)
- OS/390 C/C++

既存のサポートされる C-COBOL ILC アプリケーションは、以下のいずれかとリンク・エディットしなければなりません。

- 言語環境プログラム
- C/370 バージョン 2 で実行している場合は、C 移行ツール (APAR PN74931) (言語環境プログラムのリリース 5 で稼動する C 移行ツールの場合は、APAR PN77483 もインストールする必要があります。この機能は、言語環境プログラムリリース 6 以上では基本に含まれています。)

言語環境プログラムとリンク・エディットしなければならない以下のケースを除き、どちらの方式でも使用することができます。

- アプリケーションが COBOL NORES プログラムを含んでいる。
- アプリケーションが COBOL プログラム特有のランタイム・オプションまたはスペース調整を必要とする。

動的に呼び出される RENT プログラムの場合の動作の違い

VS COBOL II プログラムでは、VS COBOL II のランタイムのもとで、それぞれの呼び出しごとに WORKING-STORAGE の別のコピーが使用されます。これに対し、言語環境プログラムでは、それぞれの呼び出しごとに WORKING-STORAGE の同じコピーが使用されます。このような状況の違いは、以下の条件がすべて該当する場合に発生します。

- RENT オプションを指定してコンパイルされた
- OS/VS COBOL、VS COBOL II、IBM COBOL、または Enterprise COBOL のプログラムから動的に呼び出された
- C によって取り出され、呼び出された

さらに、VS COBOL II ランタイム・ライブラリーのもとで実行されるプログラムでは、初期状態が使用されます。これに対し、言語環境プログラムで実行されるプログラムの場合は、CANCEL による介入がない限り、最後に使われた状態が使用されます。

COBOL と C の間の言語間通信に関する詳細については、*言語環境プログラム ILC (言語間通信) アプリケーションの作成* および *IBM C/370 移行の手引き* を参照してください。

ランタイム環境の初期設定

このセクションは以下のものに適用されます。

✓ RES	NORES	CICS	✓ NORES（リンク済み）

ランタイム環境を初期設定するために使用できる方式は、言語環境プログラムと VS COBOL II とでは異なります。

LIBKEEP を使用する既存のアプリケーション

VS COBOL II の LIBKEEP ランタイム・オプションは、COBOL メインプログラムへの呼び出しと呼び出しの間でランタイム環境の区画レベルを維持することによってランタイム・パフォーマンスを向上させるために使用されていました。言語環境プログラム・ランタイム環境は LIBKEEP ランタイム・オプションをサポートしません。

言語環境プログラムのライブラリー・ルーチン保存 (LRR) 機能を使用することによって、メインプログラム環境について同様のパフォーマンスを得ることができます。 IMS 領域で LRR を使用するには、CEELRRIN を IMS PREINIT (事前初期設定) リストに入れる必要があります。これを行う方法の詳細については、*言語環境プログラム カスタマイズ* を参照してください。

アプリケーション・プログラム内で LRR を使用するには、CEELRR マクロを含んでいるアセンブラー・プログラムを呼び出します。

言語環境プログラムでは、2 つのサンプル・プログラム (LRR を初期設定するためのものと、LRR を終了するためのもの) が用意されています。これらのサンプル・プログラムの名前は CEELRRIN および CEELRRTR です。これらは SCEESAMP ライブラリーに入っています。

LRR の詳細については、*言語環境プログラム プログラミング・ガイド* および使用中のプラットフォーム用のカスタマイズ資料を参照してください。

言語環境プログラム事前初期設定に関する考慮事項

メインプログラム環境用の言語環境プログラム事前初期設定サービスにより、ユーザーは CEEPIPI(INIT_MAIN...) サービスを使用してメインプログラム用のランタイム環境を初期設定し、CEEPIPI(CALL...) サービスを使用してプログラムをメインとして呼び出し、CEEPIPI(TERM...) サービスを使用して事前初期設定済み環境を終了することができます。

VS COBOL II ランタイムから言語環境プログラムへの移行

言語環境プログラムの事前初期設定メインプログラム・サービスを使用するためには、VS COBOL II プログラムおよび OS/VS COBOL プログラムを CEEPIPI(CALL...) のターゲットにすることはできません。ただし、VS COBOL II プログラムおよび OS/VS COBOL プログラムを、事前初期設定済み環境内の COBOL CALL ステートメントのターゲットにすることはできます。

アプリケーションを VS COBOL II ランタイムのもとで実行するときは、ACCEPT SYSIN 機能、DISPLAY SYSOUT 機能、および DISPLAY SYSPUNCH 機能によって使用されるデータ・セットは、メインプログラムのそれぞれの呼び出しの完了後にクローズされます。したがって、データ・セットの内容は外部で使用できます。たとえば、z/OS および OS/390 バッチ環境では、SYSOUT をクローズすると、一般に、SYSOUT のデータが JOB 出力の一部になります。さらに、それぞれの実行の終了時に、ランタイム・メッセージおよびダンプを受け取ることができます。

メインプログラム環境用の言語環境プログラム事前初期設定機能を用いて実行するときは、これらのファイルと、MSGFILE ファイルおよびダンプ・ファイルは、事前初期設定済み環境を終了させるために CEEPIPI(TERM) が発行されるまでクローズされません。

言語環境プログラム事前初期設定サービスの詳細 (サブプログラム用のランタイム環境を初期設定する方法を含む) については、*言語環境プログラム プログラミング・ガイド* を参照してください。

ストレージ調整の変更点の判別

このセクションは以下のものに適用されます。

✓ RES	NORES	✓ CICS	✓ NORES (リンク済み)

スペース管理および調整のための既存の方式 (IGZTUNE マクロと SPOUT および WSCLEAR ランタイム・オプション) は、言語環境プログラムのもとでは使用可能ではありません。言語環境プログラム・サービスが RPTOPTS、 RPTSTG、 STORAGE、 ANYHEAP、 BELOWHEAP、 HEAP、 LIBSTACK、および STACK ランタイム・オプションを用いてスペース管理および調整を制御するようになりました。

VS COBOL IIでは、ストレージ調整はプロセス・レベルで行われます。言語環境プログラムのもとでは、ストレージ調整はエンクレーブ単位で行われます。言語環境プログラムでのスペース管理および調整の詳細については、*言語環境プログラム プログラミング・ガイド* を参照してください。

IGZTUNE に代わるもの

スペース調整 CSECT IGZETUN をアセンブルすることによって生成された CSECT がロード・モジュール内で検出されると、警告レベル・メッセージが発行され、この CSECT は使用されません。その他の言語環境プログラム機能は、リンク・エディット時ではなく実行時にストレージ管理能力を提供します。

注: IGZETUN を使用するアプリケーションの場合、警告レベル・メッセージを抑制する方法およびそのメッセージを言語環境プログラムの MSGFILE に書き込むときのオーバーヘッドについては、103 ページの『CEEWUCHA の使用』を参照してください。

これは、以下の 5 つのストレージ管理ランタイム・オプションを使用することによって行われます。

HEAP WORKING-STORAGE、EXTERNAL データ、および EXTERNAL ファイル情報のようなユーザー・データのためのヒープ・ストレージを管理します。

ANYHEAP
言語環境プログラムおよび COBOL ライブラリー・ルーチンによって使用されるヒープ・ストレージ (どこにでも置くことができ、制御ブロック用に使用することができる) を管理します。

BELOWHEAP
言語環境プログラムおよび COBOL ライブラリー・ルーチンによって使用されるヒープ・ストレージ (16MB 境界より下に置かれ、制御ブロックおよび入出力バッファー用に使用される) を管理します。

STACK
言語環境プログラム、LOCAL-STORAGE SECTION 内の COBOL データ項目、および COBOL ライブラリー・ルーチンによって使用されるスタック・ストレージ (動的ストレージ域 (DSA) 用に使用することができる) を管理します。

LIBSTACK
言語環境プログラムおよび COBOL ライブラリー・ルーチンによって使用されるスタック・ストレージ (16MB 境界より下に置かれ、DSA 用に使用される) を管理します。

RPTSTG ランタイム・オプションは、ストレージ管理オプションの指定時に使用すべき最適な値を提供します。(まず、RPTSTG オプションを使用して、プログラムまたは実行単位内で使用されたストレージの報告書を生成することが必要です。次に、この報告書を使用して、スペース調整の目的を達成するために言語環境プログラムの 5 つのストレージ・オプションに指定する必要のある値を決定することができます。)

言語環境プログラムの調整オプションを指定するときには、IGZTUNE からの値を使用しないでください。これらの値は、言語環境プログラム環境内のストレージ使用の調整には適さない可能性があります。推奨される言語環境プログラムのストレージ・オプション設定値については、59 ページの『推奨されるデフォルト言語環境プログラム・ランタイム・オプションの設定』を参照してください。

SPOUT 出力に関する考慮事項

SPOUT ランタイム・オプションを指定している既存のアプリケーションは、引き続き稼働します。SPOUT オプションは、言語環境プログラムの RPTOPTS および RPTSTG ランタイム・オプションにマップされています。

RPTSTG および RPTOPTS 報告書からの出力は、MSGFILE ランタイム・オプションで指定された DD 名に送られます。デフォルトは SYSOUT です。RPTOPTS お

よび RPTSTG オプションによって生成された情報を使用して、言語環境プログラムのストレージ・オプションによってストレージを調整するときに使用すべき値を決定することができます。

CICS に関するその他の考慮事項

ここでは、CICS で実行される COBOL プログラムに関するその他の考慮事項を示します。以下のトピックに関する情報が記載されています。

- パフォーマンス
- SORT インターフェースの変更
- WORKING-STORAGE の限界
- VS COBOL II NORENT プログラム
- IGZETUN または IGZEOPT と MSGFILE
- CICS HANDLE コマンドおよび CBLPSHPOP ランタイム・オプション
- DISPLAY ステートメント
- CLER トランザクション

パフォーマンスに関する考慮事項

CICS で言語環境プログラムを使用する場合にパフォーマンスを最大化するために取ることのできるいくつかの処置を、以下に示します。

- CICS SIT オプション RUWAPOOL を YES に設定してください。そうすれば、CICS LINK を使用したときの CICS GETMAIN と FREEMAIN アクティビティーを削減することができます。
- 言語環境プログラムのストレージ・ランタイム・オプションを調整してください。そうすれば、CICS GETMAIN と FREEMAIN アクティビティーを最小に保つことができ、増分が必要なときに発生するオーバーヘッドが回避されます。CICS Transaction Server バージョン 1 リリース 3 以上を使用している場合は、CICS SIT オプション AUTODST を YES に設定すると、自動ストレージ調整が実行されます。
- TERMTHDACT(DUMP) および TERMTHDACT(UADUMP) を実動で使用しないでください。TERMTHDACT のこれらのサブオプションを使用すると、トランザクションの異常終了時に言語環境プログラムのダンプ・データを一時データ・キュー CESE に書き込むために多くの時間が費やされるからです。
- 実動でランタイム・オプション RPTOPTS(ON) または RPTSTG(ON) を使用して実行しないでください。これらのオプションはパフォーマンスに著しい影響を与えます。
- 言語環境プログラムによって CESE に書き込まれた警告メッセージを受け取った場合は、その警告メッセージが出されないように変更を加えてください。

SORT インターフェースの変更点

CICS で VS COBOL II ランタイムのもとで SORT ステートメントが使用されると、SORT プログラムが EXEC CICS LINK によって呼び出されます。CICS で言語環境プログラムのもとで SORT ステートメントが使用されると、SORT プログラムが EXEC CICS LOAD によってロードされ、そのあと BASSM 命令によって呼び出されます。

WORKING-STORAGE の限界

表 27 に示されているように、言語環境プログラムのもとで実行される VS COBOL II プログラムについての WORKING-STORAGE の限界は、VS COBOL II ランタイムのもとで実行される場合と異なります。この違いは、言語環境プログラムを使用して開発され、VS COBOL II を使用して実動で実行されるアプリケーションに影響を与えます。

表 27. CICS での VS COBOL II プログラムについての WORKING-STORAGE の限界

	WORKING-STORAGE の限界	
DATA コンパイラー・オプションの指定	**VS COBOL II ランタイム**	**言語環境プログラム・ランタイム**
DATA(24)	64KB	16MB 境界より下の使用可能スペース
DATA(31)	1 MB	128 MB

VS COBOL II NORENT プログラム

VS COBOL II プログラムを CICS で実行するためには、RENT オプションを指定してコンパイルする必要があることが文書化されていますが、VS COBOL II リリース 3.0 以上では、この要件は強要されていませんでした。VS COBOL II APAR PN65736 は、VS COBOL II NORENT プログラムを CICS で実行する試みを診断するための機能を追加しました (VS COBOL II リリース 3.0 以上の場合)。

言語環境プログラムは、NORENT を指定してコンパイルされ、CICS で実行されている VS COBOL II プログラムを検出すると、メッセージ IGZ0018S を出します。PN65736 が適用されていない VS COBOL II で開発しているときに、CICS で実行される VS COBOL II アプリケーションが NORENT でコンパイルされていた場合、言語環境プログラムはそのアプリケーションを終了させます。

IGZETUN または IGZEOPT と MSGFILE

言語環境プログラムでの実行時に、IGZETUN または IGZEOPT のどちらかがメインプログラムとリンク・エディットされている場合、IGZETUN または IGZEOPT を含んでいるメインプログラムが実行されるたびに、言語環境プログラムは警告メッセージ IGZ0014W を言語環境プログラムの MSGFILE に書き込みます。メッセージ IGZ0014W が MSGFILE に書き込まれると、かなりの量の追加システム・リソースが使用されます。IGZ0014W の書き込みを抑制するには、103 ページの『CEEWUCHA の使用』を参照してください。

CICS HANDLE コマンドおよび CBLPSHPOP ランタイム・オプション

言語環境プログラムの CBLPSHPOP ランタイム・オプションは、VS COBOL II、IBM COBOL、または Enterprise COBOL のサブルーチンが呼び出されるときに、CICS の PUSH HANDLE コマンドおよび POP HANDLE コマンドが発行されるかどうかを制御するために使用されます。CBLPSHPOP の設定によって、より高速の CICS パフォーマンスまたは互換性のある動作を得ることができます (プログラムのコーディング方法に応じて異なります)。

CICS の PUSH HANDLE および POP HANDLE コマンドは、以下の CICS コマンドの現在の影響を延期 / 復元します。

- IGNORE CONDITION
- HANDLE ABEND
- HANDLE AID
- HANDLE CONDITION

VS COBOL II ランタイムは、COBOL サブルーチンの呼び出し時に PUSH HANDLE コマンドおよび POP HANDLE コマンドを自動的に発行します。言語環境プログラムでこれと同じ動作を得るためには、言語環境プログラムの CBLPSHPOP(ON) ランタイム・オプションを指定しなければなりません。

CICS のもとでアプリケーションが COBOL (VS COBOL II、IBM COBOL、または Enterprise COBOL) サブルーチンを呼び出す場合、CBLPSHPOP(OFF) を指定した方が、CBLPSHPOP(ON) の場合よりもパフォーマンスが向上します (CEEUOPT を使用することによって、エンクレーブ単位で CBLPSHPOP ランタイム・オプションを設定することができます)。

CBLPSHPOP(OFF) は、PUSH HANDLE および POP HANDLE コマンドの発行を防止します。これは、アップグレード時の互換性問題を避けるために注意して使用しなければなりません。COBOL サブルーチンへの呼び出しの前に CICS PUSH HANDLE コマンドを発行しない場合は、そのサブルーチンは呼び出し元から IGNORE CONDITION または HANDLE コマンドを継承します。そのサブルーチンが、その後、IGNORE CONDITION または HANDLE コマンドを発行すると、呼び出し元は戻り時にそれらの影響を継承します。

以下のいずれかの条件が存在する場合に、VS COBOL II ランタイムの場合と同じ動作を得るためには、CBLPSHPOP(ON) を指定してください。

- 呼び出し元に、CICS での条件処理に使用される以下のいずれかの CICS コマンドが含まれている。
  - CICS IGNORE CONDITION
  - CICS HANDLE ABEND PROGRAM(program)
- COBOL サブルーチンに、以下のいずれかの「PUSH 可能な」CICS コマンドが含まれている。
  - CICS IGNORE CONDITION
  - CICS HANDLE ABEND
  - CICS HANDLE AID
  - CICS HANDLE CONDITION

呼び出し元に含まれている CICS HANDLE...(label) コマンドは、プログラムに上記のいずれかのステートメントが含まれていない限り、CBLPSHPOP(OFF) の互換性問題を起こさないことに注意してください。

以下の 2 つの例では、CBLPSHPOP オプションが VS COBOL II の互換性に与える影響を示します。

例 1 — 互換性に影響を与えない場合

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PGM1
ENVIRONMENT DIVISION.
```

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.
     EXEC CICS HANDLE ABEND LABEL(LBL1) END-EXEC.
     CALL "PGM2" USING DFHEIBLK DFHCOMMAREA
     EXEC CICS RETURN END-EXEC.
 LBL1.
     EXEC CICS RETURN END-EXEC.


 IDENTIFICATION DIVISION.
 PROGRAM-ID. PGM2
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DATA1    PIC 9 VALUE 0.
 01  DATA2    PIC 9 VALUE 1.
 PROCEDURE DIVISION.
* Force a DIVIDE-BY-ZERO exception
     COMPUTE DATA1 = DATA2 / DATA1
     GOBACK.
```

この例では、CBLPSHPOP オプションの設定は VS COBOL II の互換性に影響を与えません。CBLPSHPOP(ON) を指定した場合、言語環境プログラムによって実行される CICS PUSH HANDLE が、PGM1 内で発行される CICS HANDLE ABEND の影響を延期するため、PGM2 について HANDLE ABEND が有効になりません。PGM2 内で 0 除算が発生すると、アクティブの HANDLE ABEND がないため、ASRA 異常終了が発生します。

CBLPSHPOP(OFF) を指定した場合、PGM2 内で 0 除算が発生すると、CICS は言語環境プログラムに PGM1 内の LBL1 へ分岐するように要請しますが、言語環境プログラムはプログラム境界を超えるラベルへの分岐を許可しません。CBLPSHPOP(ON) の場合と同様に、プログラムは ASRA 異常終了で終了します。

例 2 — 互換性に影響を与える場合

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PGM1
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DATA1    PIC 9 VALUE 0.
 01  DATA2    PIC 9 VALUE 1.
 PROCEDURE DIVISION.
     EXEC CICS HANDLE ABEND LABEL(LBL1) END-EXEC.
     CALL "PGM2" USING DFHEIBLK DFHCOMMAREA
* Force a DIVIDE-BY-ZERO exception
     COMPUTE DATA1 = DATA2 / DATA1
     EXEC CICS RETURN END-EXEC.
 LBL1.
     EXEC CICS RETURN END-EXEC.

 IDENTIFICATION DIVISION.
 PROGRAM-ID. PGM2
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.
     EXEC CICS HANDLE ABEND LABEL(LBL1A) END-EXEC.
     GOBACK.
 LBL1A.
     GOBACK.
```

VS COBOL II ランタイムから言語環境プログラムへの移行

この例では、CBLPSHPOP オプションは、VS COBOL II との互換性に影響を与えます。CBLPSHPOP(ON) を指定した場合、PGM1 の HANDLE ABEND ラベルが、PGM2 への呼び出しを超えて保管および復元され、0 除算例外は LBL1 への分岐によって処理されます。プログラムは正常に終了します。

CBLPSHPOP(OFF) を指定した場合、PGM2 の HANDLE ABEND コマンドは、PGM1 への戻り時に有効です。PGM1 内で 0 除算例外が発生すると、CICS は言語環境プログラムに PGM2 内の LBL1A へ分岐するように要請します。言語環境プログラムはこの分岐を妨げ、プログラムは ASRA 異常終了で終了します。

DISPLAY ステートメント

CICS で VS COBOL II ランタイムを使用する場合、VS COBOL II プログラムで DISPLAY ステートメントを使用しようとすると、トランザクション異常終了を引き起こします。

言語環境プログラムのもとで VS COBOL II、IBM COBOL、または Enterprise COBOL のプログラムから DISPLAY UPON SYSOUT (または、UPON 値の指定のない DISPLAY) を使用すると、表示出力が一時データ・キュー CESE に書き込まれます。

CLER トランザクション

APAR PQ38838 を適用すると、CICS トランザクション (CLER) で、ある領域について現行の言語環境プログラム・ランタイム・オプションをすべて表示することができ、これらのオプションのサブセットを変更することもできます。

文書化されていない VS COBOL II 拡張

COBOL の SORT ステートメントや MERGE ステートメントで参照されている入出力プロシージャーは、別の SORT ステートメントや MERGE ステートメントを直接呼び出すことも、間接的に呼び出すこともできません。VS COBOL II ランタイム・ライブラリーを使用する場合、このようなネストされた SORT ステートメントや MERGE ステートメントは、診断されませんでした。しかし、言語環境プログラム バージョン 2 リリース 7 (PQ39908 を適用) 以上では、VS COBOL II プログラムのネストされた SORT または MERGE を検出して、プログラムを終了する前にメッセージ IGZ0173S を発行します。

第 8 章 アプリケーションと言語環境プログラムとのリンク・エディット

この章では、OS/VS COBOL および VS COBOL II アプリケーションを言語環境プログラムとリンク・エディットすることの含意を説明します。アプリケーションを言語環境プログラムとリンク・エディットすると、アプリケーションが次のどちらのプログラムから構成されているかに応じて、動作に変更がある場合があります。

- NORES を指定してコンパイルされたプログラム
- RES を指定してコンパイルされたプログラム

COBOL ロード・モジュールを言語環境プログラムと再リンク・エディットする方法の追加情報については、349 ページの『付録 J. リンク・エディットの例』を参照してください。

NORES プログラムから構成されるアプリケーション

NORES プログラムから構成されるアプリケーションを言語環境プログラムとリンク・エディットするときは、以下のことを行う必要があります。

- ロード・モジュール内のすべてのプログラムを言語環境プログラムとリンク・エディットする
- 発生する可能性がある動作の変更を理解する

通常、OS/VS COBOL の NORES アプリケーションを言語環境プログラムとリンク・エディットする場合、影響はありません。アプリケーションは以前と同じ結果を提供し、言語環境プログラム・サービスにアクセスすることができません。

場合によっては、OS/VS COBOL の NORES アプリケーションを言語環境プログラムとリンク・エディットすると、アプリケーションが RES の動作を示すことがあります。表 28 に、この動作の変更を引き起こす可能性がある原因を示します。

表 28. アプリケーションの動作の変更を引き起こすプログラム属性

属性	説明
VS COBOL II プログラムを含んでいる	VS COBOL II プログラムを含んでいるアプリケーションは、言語環境プログラムとのリンク・エディット後に RES の動作を示します。
IBM COBOL または Enterprise COBOL プログラムを含んでいる	Enterprise COBOL プログラム、COBOL (OS/390 および VM 版) プログラム、COBOL (MVS および VM 版) プログラム、または COBOL/370 プログラムを含んでいるアプリケーションは、言語環境プログラムとのリンク・エディット後に RES の動作を示します。詳細については、251 ページの『第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加』を参照してください。
IGZCBSN ブートストラップ・ルーチンを含んでいる	IGZCBSN は、COBOL/370 のブートストラップ・ルーチンです。

表 28. アプリケーションの動作の変更を引き起こすプログラム属性 (続き)

属性	説明
IGZCBSO ブートストラップ・ルーチンを含んでいる	IGZCBSO は、COBOL (MVS および VM 版)、COBOL (OS/390 および VM 版)、および Enterprise COBOL のブートストラップ・ルーチンです。
IGZBRDGE マクロを用いて生成されたオブジェクト・モジュールを使用するプログラムを含んでいる	IGZBRDGE マクロを用いて生成されたオブジェクト・モジュールは、通常、静的 CALL ステートメントを動的 CALL ステートメントに変換するために使用されます。ただし、IGZBRDGE を用いて生成されたオブジェクト・モジュールが存在するだけで、それが使用される方法に関係なく、動作の変更が発生します。

注: 複数ロード・モジュール・アプリケーションの場合、最初のロード・モジュールが上記の属性の 1 つを含んでいると、アプリケーションは RES と同様に動作します。

RES と同様になることの含意

121 ページの表 28にリストされている属性の 1 つが存在すれば、NORES アプリケーションは RES アプリケーションとして動作するようになります。NORES アプリケーションが RES と同様になる場合、追加の考慮事項があります。

- RES を指定してコンパイルされたプログラムに関する考慮事項の多くが、言語環境プログラムとリンク・エディットされた NORES アプリケーションに適用されます。このことは、第 6 章と第 7 章の各セクションの先頭にあるマージン・アイコンに、「NORES (リンク済み)」カテゴリーで示されています。
- 追加の言語環境プログラム・サービスが使用可能です。たとえば、プログラムによって使用されたストレージの報告書を入手したい場合、
  - NORES アプリケーションを言語環境プログラムとリンク・エディットする前には、言語環境プログラムは初期設定されませんでした。ストレージ報告書を生成することはできませんでした。
  - NORES アプリケーションを言語環境プログラムとリンク・エディットした後では、言語環境プログラムが初期設定されます。RPTSTG ランタイム・オプションを指定することによって、ストレージ報告書を要求することができます。
- パフォーマンスに影響を与えます。

RES プログラムから構成されるアプリケーション

RES プログラムから構成されるアプリケーションを言語環境プログラムとリンク・エディットした場合、それは引き続き RES 動作を持ちます。

言語環境プログラムとリンク・エディットされた VS COBOL II RES プログラムは、言語環境プログラムのアプリケーション固有ランタイム・オプション CSECT である CEEUOPT を使用することができます。同じアプリケーション内に IGZEOPT と CEEUOPT が存在することができるため、95 ページの表 24で、どちらの方式が有効になるかを調べてください。

第 9 章 言語環境プログラムのリリース・レベルのアップグレード

言語環境プログラムは、前のリリースの言語環境プログラムで実行されていたアプリケーションのために、一般オブジェクトおよびロード・モジュール互換性を提供します。ただし、以下のケースでは、言語環境プログラムの特定のリリース・レベルにアップグレードするときに再リンクまたは再コンパイルする必要があります。

- OS/390 バージョン 2 リリース 9 以上において DATA(31) プログラムの動作が変更されている
- OS/390 バージョン 2 リリース 8 において DUMP マクロを使用するアセンブラー・プログラムを持つアプリケーションで CEEDUMP がない
- OS/390 バージョン 2 リリース 10 において RECORDING MODE U を指定した COBOL プログラムのファイル処理方法が変更されている
- OS/390 バージョン 2 リリース 9 以上において別の AMODE が指定されているアセンブラー・プログラムと COBOL プログラムの間で呼び出しが行われる
- 記号フィードバック・トークンを参照する

OS/390 バージョン 2 リリース 9 以上における DATA(31) プログラムの動作の変更

言語環境プログラム OS/390 版 バージョン 2 リリース 8 以前の場合、ユーザーが DATA(31) コンパイラー・オプションを指定して 16MB 境界より上の WORKING-STORAGE およびパラメーター・リストを要求した場合でも、16MB 境界より下のストレージが獲得されることがありました。これらのプログラムが AMODE 24 プログラムを動的に呼び出す場合は、言語環境プログラム OS/390 版 バージョン 2 リリース 8 以前を言語環境プログラム OS/390 版 バージョン 2 リリース 9 以上に移行すると、ランタイム・エラー・メッセージ IGZ0033S が発行されます。

言語環境プログラム OS/390 版 バージョン 2 リリース 8 以前では、DATA(31) を指定してコンパイルされた COBOL プログラムの WORKING-STORAGE は、BELOW ストレージから割り振られた HEAP セグメントから獲得されました。言語環境プログラム OS/390 版 バージョン 2 リリース 9 以上では、DATA(31) を指定してコンパイルされた COBOL プログラムの WORKING-STORAGE は、ANYWHERE ストレージ (16MB 境界より上にも下にも割り振ることができる) から割り振られた HEAP セグメントから獲得されました。この変更によって、アプリケーションがどのような影響を受けるかを説明する例を次に示します。

- エンクレーブ内の従来の HEAP 要求では、16MB 境界より下の HEAP ストレージを明確に要求する必要があります。たとえば、DATA(24) が指定された COBOL プログラムがこのような HEAP ストレージを要求すると、言語環境プログラムのヒープ・マネージャーは、BELOW ストレージから HEAP セグメントを割り振りました。

- DATA(31) が指定された後続の COBOL プログラムによる HEAP ストレージ要求は、BELOW ストレージから割り振られた HEAP セグメントによって満たされなければなりません。このためには、WORKING-STORAGE のサイズを制限する必要があります。

DATA(31) が指定された COBOL プログラムを言語環境プログラム OS/390 版 バージョン 2 リリース 8 以前で実行すると、HEAP は BELOW ストレージから獲得されます。同じプログラムを言語環境プログラム OS/390 版 バージョン 2 リリース 9 以上で実行すると、HEAP は ANYWHERE ストレージから獲得されます。

AMODE 24 プログラムに対する呼び出しが行われる場合は、言語環境プログラム OS/390 版 バージョン 2 リリース 9 以上に対して、次の考慮事項が適用されます。

- その呼び出しが AMODE 24 プログラムに対する COBOL 動的呼び出しである場合は、ランタイム・エラー・メッセージ IGZ0033S が出されます。
- AMODE 24 プログラムの呼び出しが COBOL 動的呼び出し以外の方法で行われた場合は、渡されたデータにプログラムがアクセスしようとすると、アドレッシング例外が発生します。

OS/390 バージョン 2 リリース 8 における DUMP マクロを使用するアセンブラー・プログラムを持つアプリケーションでの CEEDUMP の欠如

以下のバージョン間で移行する場合は、このセクションを参照してください。

- 言語環境プログラム OS/390 版 バージョン 2 リリース 7 以前から、APAR PQ38656 が適用された PTF を持つ言語環境プログラム OS/390 版 バージョン 2 リリース 8 以上への移行、または
- 言語環境プログラム OS/390 版 バージョン 2 リリース 7 以前から、z/OS 言語環境プログラムへの移行

ABEND マクロを使用するアセンブラー・プログラムは、DUMP パラメーター (ABEND、DUMP) が指定されていない限り、CEEDUMP を生成しません。前のリリースの言語環境プログラムでは、DUMP パラメーターを指定してもしなくても、CEEDUMP が生成されました。DUMP マクロのデフォルトは NODUMP です。この変更は、APAR PQ38656 が適用された言語環境プログラム OS/390 版 バージョン 2 リリース 8、9、および 10 で有効となりました。

OS/390 バージョン 2 リリース 10 における RECORDING MODE U を指定した COBOL プログラムのファイル処理方法の変更

言語環境プログラム OS/390 版 バージョン 2 リリース 9 以前から言語環境プログラム OS/390 版 バージョン 2 リリース 10 または z/OS 言語環境プログラムに移行する場合は、RECORDING MODE U を指定した RECFM=VB データ・セットを処理する COBOL プログラムの動作が変わる場合があります。RECFM=VB データ・セットを RECORDING MODE U として読み取るアプリケーション・プログラムの場合、ブロック全体ではなく個々のレコードを受け取る必要があるときは、アプリケーションを正しく動作させるために、変更を行わなければならないことがあります。この組み合わせによって不一致が起こり、この不一致が言語環境プログラム OS/390 版 バージョン 2 リリース 9 以前で偶発的に作用することによって、一

部のアプリケーションでこの動作が見られました。CMPR2 を指定してコンパイルされたプログラムではこの問題はありませんが、NOCMPR2 を指定してコンパイルした場合は、以下のセクションで解説している変更を JCL で行う必要があります。

アプリケーションにこの問題が存在するかどうかを判別するには、ソース・プログラムで RECORDING MODE U を検索します。RECORDING MODE U への参照がない場合、この問題に関して必要な変更は多くはありません。しかし、RECORDING MODE U を使用するプログラムがある場合は、DD ステートメントおよび (場合によっては) そのファイルのデータ・セット属性を確認する必要があります。

- RECORDING MODE U として定義されているファイルの DD ステートメントに RECFM=U が指定されている場合、おそらく、言語環境プログラム OS/390 版 バージョン 2 リリース 10 または z/OS 言語環境プログラムで実行するためにそのアプリケーションを変更する必要はありません。JCL RECFM=U によって、データ・セット・ラベルがオーバーライドされるので、そのファイルをオープンするために使用されるソース・プログラムおよび属性は整合性が保たれた状態になります。
- RECORDING MODE U として定義されているファイルの DD ステートメントに RECFM=VB (または、V、F、あるいは FB) が指定されている場合、言語環境プログラム OS/390 版 バージョン 2 リリース 10 または z/OS 言語環境プログラムで実行するためには、アプリケーションを変更する必要があります。
- RECORDING MODE U として定義されたファイルの DD ステートメントに RECFM= の指定がない場合は、そのファイルのデータ・セット属性を確認する必要があります。そのデータ・セットの RECFM が RECFM=U でない場合、アプリケーションを言語環境プログラム OS/390 版 バージョン 2 リリース 10 または z/OS 言語環境プログラムで実行するためには、そのアプリケーションを変更する必要があります。

変更を行う場合は、ファイルのオープン時にそのファイルの RECORDING MODE がデータ・セットの RECFM と一致するようにしなければなりません。アプリケーションを変更する場合は、以下の方法を使用することができます。

- ソースを変更して、可変長レコードの処理には、RECORDING MODE V および RECORD 文節のフォーマット 3 である "RECORD IS VARYING FROM n TO nn DEPENDING ON data-name-1" が使用されるようにすることができます。この RECORD 文節により定義されているファイルから読み取りを行うと、data-name-1 で読み取られた長さと同じ長さのレコードを受け取ります。
- そのファイルは引き続き RECORDING MODE U として処理することができますが、データ・セットのフォーマットが RECFM=U になるように設定する必要があります。このフォーマットを設定するには、実際の RECFM の JCL をオーバーライドするか、そのデータ・セットの実際の RECFM としてフォーマット U を定義します。この方法を使用する場合は、実際のフォーマット U の処理で戻されるレコードをアプリケーションが適切に処理できることを確認する必要があります。

実際のフォーマット U の処理では、そのファイルに対して読み取りを行うと、それぞれの読み取りからブロック全体を受け取り (フォーマット U のレコードの場合、各ブロックが 1 つのレコードとして書き込まれるため)、これがアプリケーションに戻されます。データ・セットが可変ブロック (RECFM=VB) または固定ブロック

(RECFM=FB) の場合、アプリケーションに戻されるブロックは複数のレコードで構成されるので、これをアプリケーション側で非ブロック化する必要があります。

ファイルの RECORDING MODE とデータ・セットの RECFM 間の不一致は、COBOL ソースに RECORDING MODE V または F がコーディングされていて、かつ、データ・セットに RECFM=U が設定されている場合にも起こることがあります。このコーディングは現在のところ動作しますが、RECORDING MODE と RECFM が一致するように変更することを推奨します。ファイルの RECORDING MODE とデータ・セットの RECFM の間に不一致がある場合、言語環境プログラムに対して今後行われる変更が原因でアプリケーションが失敗することが予想されます。

OS/390 バージョン 2 リリース 9 以上におけるアセンブラーと COBOL の間の呼び出し

言語環境プログラム OS/390 版 バージョン 2 リリース 9 で行われた変更は、COBOL プログラムからアセンブラー・プログラムへの戻り時に AMODE に影響を与えます。この変更は、APAR PQ05151 で報告された問題に対処するためのものです。この変更により、言語環境プログラムは VS COBOL II ランタイムと同様に動作し、COBOL プログラムのコンパイルに使用されたコンパイラーには関係なく、一貫性のある動作を提供します。

動作の変更がアセンブラーから COBOL への呼び出しに影響を与えるのは、AMODE 31 から AMODE 24 へのモード切り替えまたは AMODE 24 から AMODE 31 へのモード切り替えがあるときだけです。

言語環境プログラム OS/390 版 バージョン 2 リリース 8 以前では、COBOL およびアセンブラー間の CALL ステートメントの動作は次のとおりです。

- VS COBOL II または COBOL/370 サブプログラムがアセンブラー・プログラム呼び出し元に戻るときは、AMODE は、アセンブラー・プログラムの保管域の R14 スロットにある高位ビットに基づいて設定されます。このビットがオンの場合、制御は AMODE 31 で戻されます。オフの場合、制御は AMODE 24 で戻されます。
- OS/VS COBOL、COBOL (MVS および VM 版)、または COBOL (OS/390 および VM 版) サブプログラムがアセンブラー・プログラム呼び出し元に戻るときは、AMODE は、COBOL プログラムに入ったときに有効であったのと同じ AMODE に設定されます。
- COBOL 再使用可能環境 (RTEREUS、IGZERRE、または ILBOSTP0) を使用している場合、アセンブラー・ドライバーから呼び出された COBOL プログラムが制御をそのアセンブラー・ドライバーに戻すとき、AMODE は、アセンブラー・ドライバーの保管域の R14 スロットにある高位ビットに基づいて設定されます。このビットがオンの場合、制御は AMODE 31 で戻されます。オフの場合、制御は AMODE 24 で戻されます。

言語環境プログラム OS/390 版 バージョン 2 リリース 9 以上では、COBOL およびアセンブラー間の CALL ステートメントの動作は次のとおりです。

- COBOL サブプログラムがアセンブラー・プログラム呼び出し元に戻るときは、AMODE は、COBOL プログラムに入ったときに有効であったのと同じ AMODE に設定されます。この動作は、使用されたコンパイラーには関係なく同じであることに注意してください。
- COBOL 再使用可能環境 (RTEREUS、IGZERRE、または ILBOSTP0) を使用している場合、アセンブラー・ドライバーから呼び出された COBOL プログラムが制御をそのアセンブラー・ドライバーに戻すとき、AMODE は、COBOL プログラムに入ったときに有効であった AMODE に設定されます。

記号フィードバック・トークンの参照

言語環境プログラムのリリース 3 とリリース 4 間で、CEEIGZCT 内の 11 個の条件トークンに変更が加えられました。CEEIGZCT 内の変更された記号フィードバック・トークンを以下に示します。これらの記号フィードバック・トークンをプログラムで参照している場合は、言語環境プログラム リリース 4 以上にアップグレードするときにプログラムを再コンパイルする必要があります。

CEE36U	CEE372	CEE58Q
CEE36V	CEE373	CEE58R
CEE37O	CEE374	CEE58S
CEE371	CEE375	

第 4 部 ソース・プログラムのアップグレード

第 10 章 OS/VS COBOL ソース・プログラムのアップグレード

この章では、OS/VS COBOL 言語と Enterprise COBOL 言語間の違いについて説明します。この章の情報は、Enterprise COBOL へのアップグレードに適した COBOL アプリケーションを言語の観点から評価するためにも役立ちます。

将来の考慮事項

CICS のもとで稼働する OS/VS COBOL プログラムが言語環境プログラムを使用するには、CICS プロダクト側からの特別なサポートが必要となります。CICS TS バージョン 2 リリース 2 の後継である CICS TS (Transaction Server) リリースでは、この特別なサポートは使用できません。OS/VS COBOL プログラムは、たとえ COBOL ランタイム・ライブラリーとして言語環境プログラムを使用しても、CICS TS 2.2 以降、CICS のもとで稼働することはできなくなります。CICS のもとで稼働する OS/VS COBOL プログラムをお持ちの場合、できるだけ早く Enterprise COBOL にアップグレードする必要があります。

Enterprise COBOL は COBOL 85 標準をサポートします。OS/VS COBOL プログラムを Enterprise COBOL にアップグレードする際、Enterprise COBOL でコンパイルするためには、そのプログラムを COBOL 85 標準に変換する必要があります。

この章は、構文のガイドではありません。関係のある COBOL 言語エレメントの詳細およびコーディング規則は、以下の資料で説明されています。

- *VS COBOL for OS/VS Reference GC26-3857-04*
- *Enterprise COBOL 言語解説書 SC88-9117*

注: *VS COBOL for OS/VS Reference* は、現在 IBM では提供していません。

注:

1. CICS のもとでの実行時には、新しい言語エレメント、変更された言語エレメント、またはサポートされない言語エレメントについての特別の考慮事項があります。詳細については、237 ページの『第 17 章 COBOL ソースに関する CICS の移行の考慮事項』を参照してください。
2. 以下のセクションでは、COBOL 68 標準への参照は、IBM 完全版米国標準規格 COBOL バージョン 4 (プログラム 5734-CB2) によってサポートされる COBOL 言語への参照、または OS/VS COBOL (プログラム 5740-CB1) の LANGLVL(1) への参照です。
3. この章 (および本書全体) に記載されている情報は、最新のサービス更新が適用された OS/VS COBOL リリース 2.4 を対象としています。

OS/VS COBOL と Enterprise COBOL の比較

OS/VS COBOL は、COBOL 68 標準 (LANGLVL(1)) および COBOL 74 標準 (LANGLVL(2)) をサポートしました。Enterprise COBOL は、COBOL 85 標準をサポートします。COBOL 74 標準と、Enterprise COBOL との間の言語の違いのほかに、OS/VS COBOL プログラムは、文書化されていない OS/VS COBOL 拡張を含んでいる場合があります。

変更が必要な言語エレメント — 早見表

表 29に、OS/VS COBOL と Enterprise COBOL とで異なる言語エレメントをリストします。この表には、移行を自動化するために使用できる移行ツールがあればそれもリストしています。

以下にリストされている言語項目は、この章で詳細に記述されており、以下のカテゴリーに従って分類および配列されています。

- 他のプロダクトを必要とする OS/VS COBOL 言語エレメント
- サポートされない OS/VS COBOL 言語エレメント
- 異なる方法でインプリメントされる OS/VS COBOL 言語エレメント
- サポートされない、また文書化されていない OS/VS COBOL 拡張

表 29. OS/VS COBOL と Enterprise COBOL 間の言語エレメントの違い

言語エレメント	移行ツール	ページ
簡略複合比較条件		148
ACCEPT ステートメント		148
ALPHABETIC クラスの変更	CCCA	157
ALPHABET 文節の変更 — ALPHABET キーワード	CCCA	157
区域 A、ピリオド	CCCA	153
算術ステートメントの変更		157
ASSIGN . . . OR	CCCA	140
ASSIGN TO *integer system-name*	CCCA	140
ASSIGN . . . FOR MULTIPLE REEL /UNIT	CCCA	140
ASSIGN 文節の変更 — *assignment-name* 形式	CCCA	158
PICTURE 文節内の B 記号 — 評価の変更		158
BDAM ファイル処理	CCCA[1]	139
BLANK WHEN ZERO 文節およびアスタリスク (*) のオーバーライド		149
CALL identifier ステートメント — PICTURE 文節内の B 記号		158
CALL ステートメントの変更 — USING 句内のプロシージャー名およびファイル名		158
CANCEL ステートメント — PICTURE 文節内の B 記号		158
CLOSE . . . FOR REMOVAL ステートメント		149
CLOSE ステートメント — WITH POSITIONING 句および DISP 句	CCCA	140
簡略複合比較条件の変更	CCCA	159
グループと数値パック 10 進項目の比較		149
関連した名前を指定した COPY ステートメント	CCCA	161
通信機能		140

表 29. OS/VS COBOL と Enterprise COBOL 間の言語エレメントの違い (続き)

言語エレメント	移行ツール	ページ
CURRENCY-SIGN 文節の変更 — ’/’、’=’、および ’L’ 文字		161
CURRENT-DATE 特殊レジスター	CCCA	141
DIVIDE . . . ON SIZE ERROR — 中間結果の変更		167
CICS での実行時の動的 CALL ステートメント		149
CANCEL を介在させずに代替入り口点を使用するプログラムへの動的 CALL ステートメント		161
EXAMINE ステートメント	CCCA	142
EXHIBIT ステートメント	CCCA	142
EXIT PROGRAM/GOBACK ステートメントの変更		161
FILE STATUS 文節の変更	CCCA	162
FILE-CONTROL 段落の FILE-LIMIT 文節	CCCA	143
制御のフロー (終了ステートメントなしの場合)		150
FOR MULTIPLE REEL /UNIT	CCCA	140
USE AFTER STANDARD ERROR 宣言の GIVING 句	CCCA	143
IF . . . OTHERWISE ステートメントの変更	CCCA	164
固有でない指標名		150
INSPECT ステートメント — PROGRAM COLLATING SEQUENCE 文節		169
オプショナル・ワードとしての IS		168
ISAM ファイル処理	CCCA	138
JUSTIFIED 文節の変更	CCCA	165
TOTALING/TOTALED AREA を持つ LABEL RECORDS 文節	CCCA	143
LABEL RECORD IS ステートメント		150
MOVE ステートメント — バイナリー値および DISPLAY 値		150
MOVE ステートメントおよび比較 — 位取りの変更		165
MOVE CORRESPONDING ステートメント	CCCA	151
MOVE ステートメント — 複数の TO 指定		151
MOVE ALL--TO PIC 99		152
MOVE ステートメント — 数値切り捨ての警告メッセージ		152
MULTIPLY...ON SIZE ERROR — 中間結果の変更		167
固有でない Program-ID 名	CCCA	154
NOTE ステートメント	CCCA	144
グループ項目に対する数値クラス・テスト		166
数値データの変更		166
数字編集の変更 (PICTURE 文節)		153
OCCURS 文節 (句の順序)		152
OCCURS DEPENDING ON — ASCENDING および DESCENDING KEY 句		166
OCCURS DEPENDING ON — 受け取り項目の値の変更	CCCA	166
ON ステートメント	CCCA	144
ON SIZE ERROR 句 — 中間結果の変更		167
QSAM ファイルについて失敗する OPEN ステートメント (ファイル状況 39)		145

OS/VS COBOL プログラムの変更

表 29. OS/VS COBOL と Enterprise COBOL 間の言語エレメントの違い (続き)

言語エレメント	移行ツール	ページ
VSAM ファイルについて失敗する OPEN ステートメント (ファイル状況 39)		144
LEAVE、REREAD、および DISP 句を指定した OPEN ステートメント	CCCA	145
OPEN REVERSED ステートメント		153
OTHERWISE 文節の変更		164
パラメーターとして使用できない段落名		158
PERFORM ステートメント — VARYING および AFTER 句の変更		168
PERFORM ステートメント — 2 番目の UNTIL		153
任意の部における連続したピリオド		153
区域 A におけるピリオド	CCCA	153
段落名で欠落しているピリオド	CCCA	153
SD、FD、または RD の終わりで欠落しているピリオド		153
PICTURE 文節 (数字編集の変更)		153
PROGRAM COLLATING SEQUENCE 文節の変更		169
固有でない Program-ID 名	CCCA	154
修飾 — 同じ句の反復使用		154
READ ステートメント — KEY 句内の再定義されたレコード・キー		154
READ および RETURN ステートメントの変更 — INTO 句		169
READY TRACE および RESET TRACE ステートメント	CCCA	145
RECORD CONTAINS n CHARACTERS 文節		154
RECORD KEY 句および ALTERNATE RECORD KEY 句		154
SD または FD 記入項目内の REDEFINES 文節	CCCA	154
テーブルを指定した REDEFINES 文節		155
比較条件	CCCA	155
REMARKS 段落	CCCA	146
RENAMES 文節 — 固有でない非修飾データ名		155
報告書作成プログラム・ステートメント	報告書作成プログラム・プリコンパイラー	136
RERUN 文節の変更		169
RESERVE 文節の変更	CCCA	169
予約語リストの変更	CCCA	170
SEARCH ステートメントの変更	CCCA	170
セグメント化の変更 — 独立セグメント内の PERFORM ステートメント		170
対応する FD のない SELECT ステートメント		155
SELECT OPTIONAL 文節の変更	CCCA	171
SORT 特殊レジスター		171
SORT 動詞		155
SORT または MERGE		156
ソース言語のデバッグの変更		171
START . . . USING KEY ステートメント	CCCA	146

表 29. OS/VS COBOL と Enterprise COBOL 間の言語エレメントの違い (続き)

言語エレメント	移行ツール	ページ
STRING ステートメント — PROGRAM COLLATING SEQUENCE 文節		169
STRING ステートメント — 送り出しフィールド ID		156
範囲外の添え字 — コンパイル時にフラグが立てられる		172
ステートメント結合子としての THEN	CCCA	147
TIME-OF-DAY 特殊レジスター	CCCA	147
LABEL RECORDS 文節内の TOTALING/TOTALED AREA 句	CCCA	143
TRANSFORM ステートメント	CCCA	147
UNSTRING ステートメント — PROGRAM COLLATING SEQUENCE 文節		169
UNSTRING ステートメント — ’OR’、’IS’、または数字編集項目を用いるコーディング	CCCA	156
UNSTRING ステートメント — 複数の INTO 句		157
UNSTRING ステートメント — 添え字評価の変更		172
UPSI スイッチ	CCCA	173
USE AFTER STANDARD ERROR — GIVING 句	CCCA	143
USE BEFORE STANDARD LABEL ステートメント	CCCA	148
VALUE 文節 — 符号付き値と PICTURE 文節との関係	CCCA	157
VALUE 文節 — 条件名	CCCA	174
WHEN-COMPILED 特殊レジスター	CCCA	174
WRITE AFTER POSITIONING ステートメント	CCCA	174

注:

1. これは BDAM ファイル処理の一部についての移行です。

移行ツールを使用したプログラムの COBOL 85 標準への移行

Enterprise COBOL へのアップグレード時に必要な変更を行うのに役立つように、以下のものを使用することができます。

- COBOL 移行ツール (CCCA)
- OS/VS COBOL の MIGR コンパイラー・オプション
- 本書 (*移行ガイド*)

以下に、これらの移行ツールについて簡単に説明します。詳細については、293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

注: IBM 以外のツールも、COBOL 85 標準への移行を自動化するために使用することができます。詳細については、301 ページの『取引先製品』を参照してください。

COBOL 移行ツール (CCCA)

COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA) は、CICS 専用ではありません。CCCA はすべての古い COBOL を Enterprise COBOL に移行します。CCCA は、変更が必要とされるステートメントの報告書、または実際に移行されたプログラム自体を提供します。

詳細については、 298 ページの『COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA)』 および *COBOL and CICS/VS Command Level Conversion Aid Program Description and Operations Manual* を参照してください。

OS/VS COBOL の MIGR コンパイラー・オプション

OS/VS COBOL の MIGR コンパイラー・オプションは、OS/VS COBOL プログラム内のステートメントのうち、Enterprise COBOL でサポートされないかまたは変更が加えられたものの大部分にフラグを設定します。MIGR コンパイラー・オプションを使用すれば、移行ツールを購入しなくても、移行処置を分析することができ、必要な変更を識別できます。したがって、それぞれのプログラムごとに、移行を始める前であっても、必要な移行処置を判断することができます。

293 ページの『MIGR コンパイラー・オプション』に、MIGR によってフラグが設定される項目をリストします。MIGR によってフラグが設定される項目の詳細は、*IBM VS COBOL for OS/VS* の付録 H に記載されています。

CMPR2 および FLAGMIG コンパイラー・オプション

重要

CMPR2/NOCMPR2 コンパイラー・オプションは Enterprise COBOL ではサポートされていません。Enterprise COBOL でコンパイルされたプログラムは、NOCMPR2 が有効であるかのように動作します。

Enterprise COBOL でサポートされないか変更された OS/VS COBOL および VS COBOL II リリース 2 ステートメントを識別するには、FLAGMIG コンパイラー・オプションと CMPR2 コンパイラー・オプションを指定して、IBM COBOL のような COBOL コンパイラーの前のリリースを使用してください。既存のアプリケーション・プログラムを CMPR2 および FLAGMIG を指定して以前のリリースのコンパイラーでコンパイルすると、Enterprise COBOL でコンパイルするために修正が必要なソース言語の一部を識別することができます。

サポートのために他のプロダクトを必要とする言語エレメント

一部の OS/VS COBOL 言語エレメントは Enterprise COBOL でサポートされませんが、他のプロダクトを使用することによって同等の機能を得ることができます。

報告書作成プログラム

報告書作成プログラム機能は、報告書作成プログラム・プリコンパイラーの使用によってサポートされます。既存の報告書作成プログラム・コードが Enterprise COBOL で作動するようにするために、以下の考慮事項があります。

既存の報告書作成プログラム・コードを保持し、報告書作成プログラム・プリコンパイラーを使用する

既存の報告書作成プログラム・アプリケーション (または新しく作成されたアプリケーション) を報告書作成プログラム・プリコンパイラーで再コンパイルし、その出力を Enterprise COBOL コンパイラーへの入力として使用すると、報告書作成プ

ログラム・アプリケーションは 16MB 境界より上で稼働することができます。Enterprise COBOL を使用することにより、それらの処理能力を拡張することもできます。

この方式では、報告書作成プログラム・プリコンパイラーと Enterprise COBOL コンパイラーの両方を使用することが必要です。

報告書作成プログラム・プリコンパイラーを使用して既存の報告書作成プログラム・コードを変換する

報告書作成プログラム・コードを非報告書作成プログラム・コードに永続的に変換する場合は、報告書作成プログラム・プリコンパイラーの使用をやめて、Enterprise COBOL コンパイラーだけを使用することができます。ただし、この場合は、保守の困難な COBOL コードが生成される可能性があります。

報告書作成プログラム・コードを非報告書作成プログラム・コードに変換する場合、プリコンパイラーは変数名および段落名を生成します。これらの名前には意味がない場合があり、このため、変換後にプログラムに変更を加えようとするときに識別が困難である場合があります。これらの名前を意味のあるものに変更することはできますが、これは困難で時間のかかる可能性があります。

OS/VS COBOL でコンパイルされた既存の報告書作成プログラムを言語環境プログラムのもとで実行する

既存の OS/VS COBOL 報告書作成プログラム・アプリケーションを、Enterprise COBOL でコンパイルせずに言語環境プログラムのもとで実行することができます。言語環境プログラム・ランタイム・ライブラリーを使用して既存の OS/VS COBOL プログラムを実行する方法の詳細については、73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』を参照してください。報告書作成プログラム・ステートメントを含んでいる OS/VS COBOL アプリケーションをコンパイルするためには、引き続き OS/VS COBOL コンパイラーを使用しなければなりません。

OS/VS COBOL 報告書作成プログラム・アプリケーションは、16MB 境界より上では稼働しません。

影響を受ける報告書作成プログラム言語項目

Enterprise COBOL で受け入れられなくなった報告書作成プログラム言語項目は次のとおりです。

- GENERATE ステートメント
- INITIATE ステートメント
- LINE-COUNTER 特殊レジスター
- 非数値リテラル IS 簡略名
- PAGE-COUNTER 特殊レジスター
- PRINT-SWITCH 特殊レジスター
- FD 記入項目の REPORT 文節
- REPORT SECTION
- TERMINATE ステートメント
- USE BEFORE REPORTING 宣言

報告書作成プログラム・プリコンパイラーについては、293 ページの『付録 C. ソース・プログラム用の移行ツール』で説明されています。

インプリメントされなくなった言語エレメント

以下の OS/VS COBOL 言語エレメントは、Enterprise COBOL ではサポートされません。

- ISAM ファイル処理
- BDAM ファイル処理
- 通信機能

Enterprise COBOL では、COBOL 68 標準の言語エレメントの大部分についてのサポートが除去されました。さらに、Enterprise COBOL でインプリメントされない各種の OS/VS COBOL 言語項目もあります。

以下のセクションでは、影響を受ける言語エレメントと、行うことができる移行処置を記述します。各項目の要旨のほかに、移行に関する提案を示し、さらに、役立つ場合にはコーディング例を示しています。

ISAM ファイル処理

Enterprise COBOL は ISAM ファイルの処理をサポートしません。ISAM ファイルは、仮想記憶アクセス方式 / キー順データ・セット (VSAM/KSDS) ファイルに移行してください。

影響を受ける ISAM ファイル処理言語項目

Enterprise COBOL で受け入れられなくなった ISAM 言語項目は、次のとおりです。

APPLY CORE-INDEX
APPLY REORG-CRITERIA
ISAM ファイルのファイル宣言
NOMINAL KEY 文節
編成パラメーター I
TRACK-AREA 文節
START ステートメントの USING KEY 文節

自動移行オプション: ISAM ファイルを VSAM/KSDS ファイルに移行する場合、2 つの移行ツールが役立ちます。アプリケーションの設計に応じて、IDCAMS REPRO または CCCA を使用することができます。IDCAMS REPRO 機能は、ファイルにハードウェア依存性がない限り、移行を行います。

COBOL 移行ツール (CCCA) は、ファイル定義および入出力ステートメントを ISAM COBOL 言語から VSAM/KSDS COBOL 言語に自動的に移行することができます。CCCA 移行ツールについては、293 ページの『付録 C. ソース・プログラム用の移行ツール』で説明されています。

手動による移行処置: アプリケーションの設計上、VSAM への移行が不可能である場合は、アプリケーションを再構成して、ISAM ステートメントを、OS/VS COBOL コンパイラーでコンパイルできる入出力プログラムに分離することができます。その後、アプリケーション論理の残りの部分を、Enterprise COBOL にアップグレードできるプログラムに分離することができます。

その後、OS/VS COBOL プログラムと Enterprise COBOL プログラムの両方から構成されるアプリケーションを言語環境プログラムのもとで実行することができます。既存のプログラムを言語環境プログラムのもとで実行する方法の詳細については、以下の章を参照してください。

73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』
251 ページの『第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加』

注: この方式は、短期間マイグレーション方針としてのみ提供されます。OS/VS COBOL 用のサービスは提供されなくなったので、プログラムを書き直して OS/VS COBOL および ISAM への依存性を除去してください。(サポートされないコンパイラーの使用が気にならない場合は、この方式を使用し続けることができます。)

BDAM ファイル処理

Enterprise COBOL は BDAM ファイルの処理をサポートしません。BDAM ファイルは、仮想記憶アクセス方式 / 相対レコード・データ・セット (VSAM/RRDS) ファイルに移行してください。

影響を受ける BDAM ファイル処理言語項目

Enterprise COBOL で受け入れられなくなった BDAM 言語項目は、次のとおりです。

ACTUAL KEY 文節
APPLY RECORD-OVERFLOW
BDAM ファイルのファイル宣言
編成パラメーター D、R、W
SEEK ステートメント
TRACK-LIMIT 文節

自動移行オプション: COBOL 移行ツール (CCCA) は、BDAM COBOL 言語を VSAM/RRDS COBOL 言語に自動的に移行することができます。ただし、キー・アルゴリズムを指定する必要があります。CCCA 移行ツールについては、 293 ページの『付録 C. ソース・プログラム用の移行ツール』で説明されています。

IBM 以外のツールも、BDAM ファイルを VSAM/RRDS ファイルに移行するために使用することができます。詳細については、 301 ページの『取引先製品』を参照してください。

手動による移行処置: アプリケーションの設計上、VSAM への移行が不可能である場合は、アプリケーションを再構成して、BDAM ステートメントを、OS/VS COBOL コンパイラーでコンパイルできる入出力プログラムに分離することができます。その後、アプリケーション論理の残りの部分を、Enterprise COBOL にアップグレードできるプログラムに分離することができます。

分離が完了したら、OS/VS COBOL プログラムと Enterprise COBOL プログラムの両方で構成されるアプリケーションを言語環境プログラムのもとで実行できます。既存のプログラムを言語環境プログラムのもとで実行する方法の詳細については、以下の章を参照してください。

- 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』

- 251 ページの『第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加』

注: この方式は、短期間マイグレーション方針としてのみ提供されます。OS/VS COBOL 用のサービスは提供されなくなったので、プログラムを書き直して OS/VS COBOL および BDAM への依存性を除去してください。(サポートされないコンパイラーの使用が気にならない場合は、この方式を使用し続けることができます。)

通信機能

通信機能は Enterprise COBOL ではサポートされません。

影響を受ける通信言語項目

Enterprise COBOL で受け入れられない通信言語項目は、次のとおりです。

ACCEPT MESSAGE COUNT ステートメント
COMMUNICATION SECTION
DISABLE ステートメント
ENABLE ステートメント
RECEIVE ステートメント
SEND ステートメント

通信の移行処置

OS/VS COBOL の SEND および RECEIVE ステートメントを使用する既存の TCAM アプリケーションは、OS/VS COBOL の QUEUE ランタイム・オプションがサポートされない点を除き、言語環境プログラムのもとで稼働します。 (QUEUE ランタイム・オプションは、CD ... FOR INITIAL INPUT に RECEIVE ステートメントが指定されている OS/VS COBOL プログラムでのみ使用されます。)

詳しくは、*IBM VS COBOL for OS/VS* および *IBM OS/VS COBOL コンパイラーおよびライブラリー プログラマーの手引き* を参照してください。

サポートされない言語エレメント

Enterprise COBOL は、以下の OS/VS COBOL 言語エレメントをサポートしません。Enterprise COBOL へのアップグレード時には、以下の説明で示されているように、これらの項目を除去または変更しなければなりません。

ASSIGN . . . OR

OS/VS COBOL は ASSIGN ... OR 文節を受け入れました。この文節を Enterprise COBOL のもとで使用するには、OR を除去してください。

ASSIGN TO *integer system-name*

OS/VS COBOL は ASSIGN TO *integer system-name* 文節を受け入れました。この文節を Enterprise COBOL のもとで使用するには、integer (整数) を除去してください。

ASSIGN . . . FOR MULTIPLE REEL/UNIT

OS/VS COBOL は、ASSIGN ... FOR MULTIPLE REEL/UNIT 句を受け入れて、それを文書として扱いました。Enterprise COBOL はこの句をサポートしません。

CLOSE ステートメント — WITH POSITIONING 句および DISP 句

OS/VS COBOL は、OS/VS COBOL で IBM 拡張として提供された CLOSE ステートメントの WITH POSITIONING 句および DISP 句を受け入れました。Enterprise COBOL では、これらの句は受け入れられません。

CURRENT-DATE 特殊レジスター

OS/VS COBOL は CURRENT-DATE 特殊レジスターを受け入れました。このレジスターは、MOVE ステートメントの送り出しフィールドとしてのみ有効です。CURRENT-DATE は 8 バイトの英数字形式です。

```
MM/DD/YY  (month, day, year)
```

Enterprise COBOL は DATE 特殊レジスターをサポートします。 このレジスターは、ACCEPT ステートメントの送り出しフィールドとしてのみ有効です。DATE は 6 バイトの英数字形式です。

```
YYMMDD  (year, month, day)
```

したがって、次のようなステートメントを含んでいる OS/VS COBOL プログラムは変更しなければなりません。

```
77  DATE-IN-PROGRAM  PICTURE X(8)
  ⋮
    MOVE CURRENT-DATE TO DATE-IN-PROGRAM.
```

これを変更する (2 桁の年の形式を保持して) 1 つの方法の例は、次のとおりです。

```
01  DATE-IN-PROGRAM.
    02 MONTH-OF-YEAR     PIC X(02).
    02 FILLER            PIC X(01)  VALUE "/".
    02 DAY-OF-MONTH      PIC X(02).
    02 FILLER            PIC X(01)  VALUE "/".
    02 YEAR              PIC X(02).

01  ACCEPT-DATE.
    02 YEAR              PIC X(02).
    02 MONTH-OF-YEAR     PIC X(02).
    02 DAY-OF-MONTH      PIC X(02).
  ⋮
    ACCEPT ACCEPT-DATE FROM DATE.
    MOVE  CORRESPONDING ACCEPT-DATE TO DATE-IN-PROGRAM.
```

これを変更し、4 桁の年を指定する方法の例は、次のとおりです。

```
01  DATE-IN-PROGRAM.
    02 MONTH-OF-YEAR     PIC X(02).
    02 FILLER            PIC X(01)  VALUE "/".
    02 DAY-OF-MONTH      PIC X(02).
    02 FILLER            PIC X(01)  VALUE "/".
    02 YEAR              PIC X(04).

01  CURRENT-DATE.
    02 YEAR              PIC X(04).
    02 MONTH-OF-YEAR     PIC X(02).
    02 DAY-OF-MONTH      PIC X(02).
  ⋮
    MOVE FUNCTION CURRENT-DATE(1:8) TO CURRENT-DATE.
    MOVE CORRESPONDING CURRENT-DATE TO DATE-IN-PROGRAM.
```

サポートされない OS/VS COBOL 言語エレメント

EXAMINE ステートメント

OS/VS COBOL は EXAMINE ステートメントを受け入れていましたが、Enterprise COBOL では受け入れません。

したがって、OS/VS COBOL プログラムに以下のようなコーディングが含まれている場合、

```
EXAMINE DATA-LENGTH TALLYING UNTIL FIRST " "
```

Enterprise COBOL ではこれを以下のように置き換えてください。

```
MOVE 0 TO TALLY
INSPECT DATA-LENGTH TALLYING TALLY FOR CHARACTERS BEFORE " "
```

整数値の WORKING-STORAGE 基本データ項目を指定できる個所であれば、TALLY 特殊レジスターを引き続き使用することができます。

EXHIBIT ステートメント

OS/VS COBOL は EXHIBIT ステートメントを受け入れていましたが、Enterprise COBOL では受け入れません。

Enterprise COBOL では、DISPLAY ステートメントを使用して、EXHIBIT ステートメントを置き換えることができます。ただし、DISPLAY ステートメントは EXHIBIT ステートメントのすべての機能を実行するわけではありません。

EXHIBIT NAMED に対する修正措置

EXHIBIT NAMED ステートメントは、直接 DISPLAY ステートメントで置き換えることができます。

```
    OS/VS COBOL                          Enterprise COBOL

WORKING-STORAGE SECTION.        WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).             77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).             77  DAT-2 PIC X(8).
        .                               .
        .                               .
    EXHIBIT NAMED DAT-1 DAT-2       DISPLAY "DAT-1 = " DAT-1
                                        "DAT-2 = " DAT-2
```

EXHIBIT CHANGED に対する修正措置

EXHIBIT CHANGED ステートメントは、以下のように、IF および DISPLAY ステートメントで置き換えることができます。

1. データ項目の新しい値が以前の値と異なるかどうかを調べるために、IF ステートメントを指定します。
2. IF ステートメントの *statement-1* として DISPLAY ステートメントを指定します。

この変更により、新しい値が以前の値と異なるときにだけ、指定されたデータ項目の値が表示されます。

```
    OS/VS COBOL                          Enterprise COBOL

WORKING-STORAGE SECTION.        WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).             77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).             77  DAT-2 PIC X(8).
                                77  DAT1-CMP PIC X(8).
                                77  DAT2-CMP PIC X(8).
        .                               .
        .                               .
        .                               .
```

```
EXHIBIT CHANGED DAT-1 DAT-2    IF DAT-1 NOT EQUAL TO DAT1-CMP
                                 DISPLAY DAT-1
                               END-IF
                               IF DAT-2 NOT EQUAL TO DAT2-CMP
                                 DISPLAY DAT-2
                               END-IF
                               MOVE DAT-1 TO DAT1-CMP
                               MOVE DAT-2 TO DAT2-CMP
```

EXHIBIT CHANGED NAMED に対する修正措置

EXHIBIT CHANGED NAMED ステートメントは、以下のように、IF および DISPLAY ステートメントで置き換えることができます。

1. データ項目の新しい値が以前の値と異なるかどうかを調べるために、IF ステートメントを指定します。
2. IF ステートメントの *statement-1* として DISPLAY ステートメントを指定します。

この変更により、新しい値が以前の値と異なるときにだけ、指定されたデータ項目の値が表示されます。

OS/VS COBOL

```
WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).
        .
        .
        .
  EXHIBIT CHANGED NAMED
      DAT-1 DAT-2
```

Enterprise COBOL

```
WORKING-STORAGE SECTION.
77  DAT-1    PIC X(8).
77  DAT-2    PIC X(8).
77  DAT1-CMP PIC X(8).
77  DAT2-CMP PIC X(8).
        .
        .
        .
 IF DAT-1 NOT EQUAL TO DAT1-CMP
     DISPLAY "DAT-1 = " DAT-1
 END-IF
 IF DAT-2 NOT EQUAL TO DAT2-CMP
     DISPLAY "DAT-2 = " DAT-2
 END-IF
 MOVE DAT-1 TO DAT1-CMP
 MOVE DAT-2 TO DAT2-CMP
```

FILE-CONTROL 段落の FILE-LIMIT 文節

OS/VS COBOL は、FILE-LIMIT 文節を受け入れて、それをコメントとして扱いました。Enterprise COBOL はこの文節を受け入れません。したがって、FILE-LIMIT 文節のすべてのオカレンスを除去しなければなりません。

USE AFTER STANDARD ERROR 宣言の GIVING 句

OS/VS COBOL では、USE AFTER STANDARD ERROR 宣言の GIVING 句を指定することができました。Enterprise COBOL はこの句をサポートしません。したがって、USE AFTER STANDARD ERROR 宣言の GIVING 句のすべてのオカレンスを除去しなければなりません。

GIVING 句を置き換えるには、FILE-CONTROL FILE STATUS 文節を使用してください。FILE STATUS 文節は、エラー発生後だけでなく、各入出力要求の後で情報を提供します。

TOTALING/TOTALED AREA 句を指定した LABEL RECORDS 文節

OS/VS COBOL では、LABEL RECORDS 文節の TOTALING および TOTALED 句を使用できました。

サポートされない OS/VS COBOL 言語エレメント

Enterprise COBOL はこれらの句をサポートしません。したがって、LABEL RECORDS 文節から TOTALING/TOTALED 句のすべてのオカレンスを除去しなければなりません。さらに、これらの句に関連した変数も調べてください。

Enterprise COBOL で同等の機能を得るためには、次のようにしてください。

- FD の LABEL RECORDS 文節から TOTALING または TOTALED への参照を除去します。同等のフィールドを WORKING-STORAGE 内に定義します。

  同等のフィールドで、ファイルへのそれぞれの WRITE の前に、キーまたはレコード・データを保管する (TOTALED AREA の場合) か、またはレコード・カウントを更新して保管します (TOTALING AREA の場合) (これは、特定のボリュームに実際に書き込まれた最後のレコードを反映します)。このカウンターを FD のレコード域に保管しないでください。
- 『RESERVE 1 AREA』 を SELECT 文節に組み込みます。
- FD またはオーバーライドする JCL でブロック当たり 1 レコードだけを考慮に入れます。
- 以下を組み込みます。

  ```
  DECLARATIVES.
          USE AFTER STANDARD ENDING FILE (または REEL)
           LABEL PROCEDURE ON filename.
  ```

  その後、OUTPUT ファイル用のラベルをフォーマット設定するか、または INPUT ファイル・ラベルからのデータを保管します (AFTER STANDARD BEGINNING)。
- NOAWO コンパイラー・オプションを指定します。
- DD DCB オプション OPTCD=T を指定しないでください。これはサポートされないため、結果は予測できません。

NOTE ステートメント

OS/VS COBOL は NOTE ステートメントを受け入れました。Enterprise COBOL は NOTE ステートメントを受け入れません。したがって、Enterprise COBOL の場合、すべての NOTE ステートメントを削除し、NOTE 段落全体に代えてコメント行を使用してください。

ON ステートメント

OS/VS COBOL は ON ステートメントを受け入れました。Enterprise COBOL は ON ステートメントを受け入れません。

ON ステートメントは、それに含まれるステートメントの選択実行を可能にします。Enterprise COBOL では、EVALUATE ステートメントと IF ステートメントによって同様の機能が提供されます。

QSAM ファイルについて失敗する OPEN ステートメント (ファイル状況 39)

OS/VS COBOL では、QSAM ファイルの固定ファイル属性は、COBOL プログラムまたは JCL と一致する必要はありませんでした。Enterprise COBOL では、以下のものが一致していないと、プログラム内の OPEN ステートメントが正常に実行されないことがあります。

- ファイルについての DD ステートメントまたはデータ・セット・ラベルで指定された固定ファイル属性

- COBOL プログラムの SELECT ステートメントおよび FD ステートメントでそのファイルについて指定された属性

ファイル編成、レコード・フォーマット (固定または可変)、コード・セット、またはレコード長の属性が一致していないと、ファイル状況コード 39 が発生し、OPEN ステートメントが失敗します。

よくあるファイル状況 39 の問題を防止するには、343 ページの『付録 H. QSAM ファイルでのファイル状況 39 の防止』を参照してください。

VSAM ファイルについて失敗する OPEN ステートメント (ファイル状況 39)

OS/VS COBOL では、IDCAMS に関連した VSAM ファイル内で定義された RECORDSIZE が COBOL プログラムと一致する必要はありませんでした。 Enterprise COBOL では、それらは一致しなければなりません。以下の規則が VSAM ESDS、KSDS、RRDS、および VRRDS ファイル定義に適用されます。

表 30. VSAM ファイル定義に関する規則

ファイル・タイプ	規則
ESDS および KSDS VSAM	RECORDSIZE(avg,m) が指定されます。ここで、avg は COBOL レコードの平均サイズであり、m よりも確実に小さい値です。m は COBOL レコードの最大サイズ以上です。
RRDS VSAM	RECORDSIZE(n,n) が指定されます。ここで、n は COBOL レコードの最大サイズ以上です。
KSDS VSAM を用いる VRRDS	RECORDSIZE(avg,m) が指定されます。ここで、avg は COBOL レコードの平均サイズであり、m よりも確実に小さい値です。m は COBOL レコードの最大サイズ + 4 以上です。

LEAVE、REREAD、および DISP 句を指定した OPEN ステートメント

OS/VS COBOL では、LEAVE、REREAD、および DISP 句を指定した OPEN ステートメントを使用できました。Enterprise COBOL では、これらの句を使用できません。

REREAD 機能を置き換えるには、WORKING-STORAGE SECTION で入力レコードのコピーを定義し、各レコードを、それが読み取られた後で WORKING-STORAGE に移動してください。

READY TRACE および RESET TRACE ステートメント

OS/VS COBOL では、READY TRACE および RESET TRACE ステートメントを使用できました。Enterprise COBOL はこれらのステートメントをサポートしません。

READY TRACE ステートメントと類似した機能を得るには、デバッグ・ツールを使用するか、または Enterprise COBOL コンパイラーで使用可能な COBOL 言語を使用することができます。

デバッグ・ツールを使用する場合は、TEST(ALL,SYM) オプションを指定してプログラムをコンパイルし、以下のデバッグ・ツール・コマンドを使用してください。

```
"AT GLOBAL LABEL PERFORM;
 LIST LINES %LINE; GO; END-PERFORM;"
```

COBOL 言語を使用する場合は、Enterprise COBOL の USE FOR DEBUGGING ON ALL PROCEDURES 宣言で、READY TRACE および RESET TRACE と同様の機能を実行することができます。

以下に例を示します。

```
ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER. IBM-370 WITH DEBUGGING MODE.
   ⋮
DATA DIVISION.
   ⋮
  WORKING-STORAGE SECTION.
  01 TRACE-SWITCH        PIC 9 VALUE 0.
     88  READY-TRACE           VALUE 1.
     88  RESET-TRACE           VALUE 0.
   ⋮
PROCEDURE DIVISION.
  DECLARATIVES.
  COBOL-II-DEBUG SECTION.
    USE FOR DEBUGGING ON ALL PROCEDURES.
  COBOL-II-DEBUG-PARA.
    IF READY-TRACE THEN
        DISPLAY DEBUG-NAME
    END-IF.
  END DECLARATIVES.
  MAIN-PROCESSING SECTION.
   ⋮
  PARAGRAPH-3.
   ⋮
    SET READY-TRACE TO TRUE.
  PARAGRAPH-4.
   ⋮
  PARAGRAPH-6.
   ⋮
    SET RESET-TRACE TO TRUE.
  PARAGRAPH-7.
```

ここで、DEBUG-NAME は DEBUG-ITEM 特殊レジスターのフィールドであり、デバッグ・プロシージャーの実行を引き起こすプロシージャー名を表示します。(この例では、オブジェクト・プログラムは、制御がプロシージャー PARAGRAPH-4 ～ PARAGRAPH-6 の範囲内の各プロシージャーに達すると、そのプロシージャーの名前を表示します。)

実行時には、EXEC ステートメントに PARM=/DEBUG を指定して、このデバッグ・プロシージャーを活動化しなければなりません。この方法を使用すれば、デバッグ宣言を活動化または非活動化するためにプログラムを再コンパイルする必要はありません。

REMARKS 段落

OS/VS COBOL は REMARKS 段落を受け入れました。

Enterprise COBOL は REMARKS 段落を受け入れません。この代わりとして、桁 7 の * から始まるコメント行を使用してください。

START . . . USING KEY ステートメント

OS/VS COBOL では、USING KEY 句を指定した START ステートメント

を許可していましたが、Enterprise COBOL では許可していません。Enterprise COBOL では、KEY IS 句を使用した START ステートメントを指定することができます。

ステートメント結合子としての THEN

OS/VS COBOL は、ステートメント結合子としての THEN の使用を受け入れました。

次の例は、OS/VS COBOL での使用法を示しています。

```
MOVE A TO B THEN ADD C TO D
```

Enterprise COBOL は、ステートメント結合子としての THEN の使用をサポートしません。したがって、Enterprise COBOL では、これを次のように変更してください。

```
MOVE A TO B
ADD C TO D
```

TIME-OF-DAY 特殊レジスター

OS/VS COBOL は TIME-OF-DAY 特殊レジスターをサポートしました。このレジスターは、MOVE ステートメントの送り出しフィールドとしてのみ有効でした。TIME-OF-DAY は 6 バイトの外部 10 進フォーマットです。

```
HHMMSS (hour, minute, second)
```

Enterprise COBOL は TIME-OF-DAY 特殊レジスターをサポートしません。

したがって、次のようなステートメントを含んでいる OS/VS COBOL プログラムは変更しなければなりません。

```
77  TIME-IN-PROGRAM   PICTURE  X(6).
  ⋮
    MOVE TIME-OF-DAY TO TIME-IN-PROGRAM.
```

これを変更する 1 つの方法の例は、次のとおりです。

```
MOVE FUNCTION CURRENT-DATE (9:6) TO TIME-IN-PROGRAM
```

注: CICS のもとでは、TIME-OF-DAY も TIME も有効ではありません。

TRANSFORM ステートメント

OS/VS COBOL は TRANSFORM ステートメントをサポートしました。Enterprise COBOL は、TRANSFORM ステートメントをサポートしませんが、INSPECT ステートメントをサポートします。したがって、OS/VS COBOL プログラム内の TRANSFORM ステートメントは、INSPECT CONVERTING ステートメントで置き換えなければなりません。

たとえば、次の OS/VS COBOL TRANSFORM ステートメント

```
77  DATA-T     PICTURE X(9) VALUE "ABCXYZCCC"
  ⋮
    TRANSFORM DATA-T FROM "ABC" TO "CAT"
```

を実行した場合、TRANSFORM は各文字を評価し、各 A を C に、各 B を A に、各 C を T に変更します。

TRANSFORM ステートメントの実行後、DATA-T には “CATXYZTTT” が入っています。

たとえば、次の INSPECT CONVERTING ステートメント (Enterprise COBOL でのみ有効)

```
77  DATA-T     PICTURE X(9) VALUE "ABCXYZCCC"
  ⋮
    INSPECT DATA-T
        CONVERTING "ABC" TO "CAT"
```

を実行した場合、INSPECT CONVERTING は TRANSFORM と同様に各文字を評価し、各 A を C に、各 B を A に、各 C を T に変更します。

INSPECT CONVERTING ステートメントの実行後、DATA-T には “CATXYZTTT” が入っています。

USE BEFORE STANDARD LABEL

OS/VS COBOL では USE BEFORE STANDARD LABEL ステートメントを受け入れていましたが、Enterprise COBOL では受け入れません。

したがって、USE BEFORE STANDARD LABEL ステートメントのすべてのオカレンスを除去しなければなりません。Enterprise COBOL は標準外ラベルをサポートしないので、ユーザーは Enterprise COBOL で標準外ラベル付きファイルを処理することはできません。

文書化されていないサポートされない OS/VS COBOL 拡張

このセクションは、主に、MIGR オプションによってフラグ設定されていない COBOL ステートメントから構成されています。これらのステートメントは、OS/VS COBOL コンパイラーによって受け入れられましたが、Enterprise COBOL によって受け入れらないステートメントもあります。

これらの言語エレメントは、OS/VS COBOL への文書化されていない拡張であるため、有効な OS/VS COBOL コードであると見なされません。このリストには、すべての文書化されていない拡張が含まれているとは限りませんが、IBM で認識している限りのものをすべて組み込んであります。

簡略複合比較条件および括弧の使用

OS/VS COBOL は、簡略複合比較条件内での括弧の使用を受け入れました。

Enterprise COBOL は、ほとんどの括弧の使用を IBM 拡張としてサポートします。ただし、2 つの違いがあります。

- 簡略複合比較条件の有効範囲内では、Enterprise COBOL は括弧の内側の関係演算子をサポートしません。以下に例を示します。

  ```
  A = B  AND ( < C OR D)
  ```

- 比較条件内での括弧の誤った使用の一部は、OS/VS COBOL では受け入れられましたが、Enterprise COBOL では受け入れられません。以下に例を示します。

  ```
  (A = 0 AND B) = 0
  ```

ACCEPT ステートメント

OS/VS COBOL は、ID と簡略名または関数名との間にキーワード FROM がない ACCEPT ステートメントを受け入れました。

Enterprise COBOL はそのような ACCEPT ステートメントを受け入れません。

BLANK WHEN ZERO 文節およびアスタリスク (*) のオーバーライド

OS/VS COBOL では、同じ記入項目について BLANK WHEN ZERO 文節と、ゼロ抑制記号としてのアスタリスク (*) を指定した場合、ゼロ抑制が BLANK WHEN ZERO をオーバーライドしました。

Enterprise COBOL は、これら 2 つの言語エレメントが同じデータ記述記入項目について指定された場合、これらを受け入れません。したがって、Enterprise COBOL では、1 つのデータ記述記入項目の中にこの文節と記号の両方が含まれていてはなりません。

OS/VS COBOL プログラムの中で BLANK WHEN ZERO 文節と、ゼロ抑制記号としてのアスタリスクの両方を指定している場合に、Enterprise COBOL で同じ動作を得るためには、BLANK WHEN ZERO 文節を除去してください。

CLOSE . . . FOR REMOVAL ステートメント

OS/VS COBOL では、順次ファイル用の FOR REMOVAL 文節を使用できました。この文節はプログラムの実行に影響を与えました。Enterprise COBOL はこのステートメントを構文検査しますが、このステートメントはプログラムの実行に影響を与えません。

グループと数値パック 10 進項目の比較

OS/VS COBOL では、グループと数値パック 10 進項目の比較を使用できましたが、誤りの結果を出すコードが生成されました。

たとえば、以下の比較の結果はメッセージ

```
"1 IS NOT > 0"
```

であり、数値的に正しい "1 > 0" ではありません。

```
    05  COMP-TABLE.
        10 COMP-PAY              PIC 9(4).
        10  COMP-HRS             PIC 9(3).
    05  COMP-ITEM                PIC S9(7) COMP-3.

PROCEDURE DIVISION.
    MOVE 0 TO COMP-PAY COMP-HRS.
    MOVE 1 TO COMP-ITEM.
    IF COMP-ITEM > COMP-TABLE
       DISPLAY '1 > 0'
    ELSE
       DISPLAY '1 IS NOT > 0'.
```

Enterprise COBOL ではこのような比較を許可しません。

CICS での実行時の動的 CALL ステートメント

CICS での実行時には OS/VS COBOL プログラムからの動的 CALL ステートメントはサポートされていないのに、OS/VS COBOL は、動的 CALL を出すプログラムの診断を行いませんでした。

言語環境プログラムのもとで CICS での実行時に OS/VS COBOL プログラムが動的 CALL ステートメントを発行すると、プログラムは異常終了コード U3504 で異常終了します。

文書化されていない OS/VS COBOL 拡張

Enterprise COBOL プログラムは動的 CALL ステートメントを使用して、CICS のもとで、他の Enterprise COBOL プログラムや、PL/I および C/C++ プログラムを呼び出すことができます。

制御のフロー (終了ステートメントなしの場合)

OS/VS COBOL では、アセンブラー・プログラムを OS/VS COBOL プログラムの終わりにリンク・エディットし、制御のフローを COBOL プログラムの終わりからアセンブラー・プログラムに移すことが可能でした。

Enterprise COBOL では、プログラムの終わりに終了ステートメント (STOP RUN または GOBACK) をコーディングしていない場合、プログラムは暗黙の GOBACK によって終了します。制御のフローは COBOL プログラムの終わりを越えて進むことはできません。

「終わりを越えて」別のプログラムに進むプログラムがある場合は、コードを、別のプログラムへの CALL インターフェースに変更してください。

指標名 OS/VS COBOL では、修飾された指標名を使用できました。

Enterprise COBOL では、修飾された指標名を使用できません。指標名は参照される場合、固有でなければなりません。

LABEL RECORD IS ステートメント

OS/VS COBOL は、ワード RECORD のない LABEL RECORD 文節を受け入れました。たとえば、LABEL RECORD IS OMITTED の代わりに LABEL IS OMITTED を使用できました。

Enterprise COBOL はそのような LABEL RECORD 文節を受け入れません。

MOVE ステートメント — バイナリー値 および DISPLAY 値

Enterprise COBOL の TRUNC(OPT) コンパイラー・オプションは、OS/VS COBOL の NOTRUNC コンパイラー・オプションとの互換性のために推奨されますが、フルワード・バイナリー項目 (USAGE COMP PIC 9(5) ～ PIC 9(9)) の移動に関して異なる結果をもたらす可能性があります。

以下に例を示します。

```
WORKING-STORAGE SECTION.
  01 WK1 USAGE COMP-4 PIC S9(9).
   ⋮
PROCEDURE DIVISION.
   ⋮
  MOVE 1234567890 to WK1
  DISPLAY WK1.
  GOBACK.
```

この例では、10 桁の値が 9 桁の項目に移動されているので無効な COBOL コーディングです。

たとえば、以下のコンパイラー・オプションを指定してコンパイルされた場合、結果は以下のようになります。

	OS/VS COBOL の NOTRUNC	Enterprise COBOL の TRUNC(OPT)
バイナリー値	x’499602D2’	x’0DFB38D2’
DISPLAY 値	234567890	234567890

OS/VS COBOL の場合、バイナリー・データ項目に含まれているバイナリー値は DISPLAY 値と同じではありません。DISPLAY 値は PICTURE 文節内の桁の数に基づいており、バイナリー値はバイナリー・データ項目のサイズ (このケースでは 4 バイト) に基づいています。バイナリー・データ項目の 10 進数での実際の値は 1234567890 です。

Enterprise COBOL の場合、バイナリー値と DISPLAY 値は同じです。これは、発生した切り捨てが PICTURE 文節内の桁の数に基づいていたためです。

この状態は、OS/VS COBOL では MIGR によって、Enterprise COBOL では TRUNC(OPT) を用いてのコンパイル時に、フラグが設定されます。

MOVE CORRESPONDING ステートメント

- OS/VS COBOL では、MOVE CORRESPONDING で複数の受け取り側を許可していましたが、Enterprise COBOL では許可していません。したがって、次の OS/VS COBOL ステートメント

  ```
  MOVE CORRESPONDING GROUP-ITEM-A TO GROUP-ITEM-B GROUP-ITEM-C
  ```

  は、以下の 2 つの Enterprise COBOL MOVE CORRESPONDING ステートメントに変更する必要があります。

  ```
  MOVE CORRESPONDING GROUP-ITEM-A TO GROUP-ITEM-B
  MOVE CORRESPONDING GROUP-ITEM-A TO GROUP-ITEM-C
  ```

- OS/VS COBOL のリリース 2.4 より前のリリースは、MOVE CORRESPONDING ステートメントの受け取り側内の固有でない従属データ項目を受け入れましたが、Enterprise COBOL では受け入れません。以下に例を示します。

  ```
  01  KANCFUNC.
      03  CL  PIC XX.
      03  KX9 PIC XX.
      03  CC  PIC XX.
  01  HEAD1-AREA.
      03  CL  PIC XX.
      03  KX9 PIC XX.
      03  CC  PIC XX.
      03  KX9 PIC XX.
  ⋮
      MOVE CORR KANCFUNC to HEAD1-AREA.
  ```

  Enterprise COBOL の場合、受け取り側内のデータ項目が固有の名前を持つように変更してください。

MOVE ステートメント — 複数の TO 指定

OS/VS COBOL では、MOVE ステートメントのそれぞれの受け取り側の前で予約語 TO を使用できました。以下に例を示します。

```
MOVE aa TO bb TO cc
```

Enterprise COBOL では、上記のステートメントは次のように変更しなければなりません。

```
MOVE aa TO bb cc
```

MOVE ALL--TO PIC 99

OS/VS COBOL では、固定数値受け取りフィールドへのグループ移動を使用できました。以下に例を示します。

```
MOVE ALL ' ' TO num1
```

ここで、num1 は PIC 99 です。

Enterprise COBOL では上記のケースを許可しません。

MOVE ステートメント — 数値切り捨ての警告メッセージ

OS/VS COBOL は、桁が失われることになるような数値受け取り側を持つ MOVE ステートメントの場合は警告メッセージを出しました。以下に例を示します。

```
77 A PIC 999.
77 B PIC 99.
⋮
   MOVE A TO B.
```

VS COBOL II、COBOL (MVS および VM 版)、および COBOL (OS/390 および VM 版) バージョン 2 リリース 1 は、このケースで警告メッセージを出しません。

COBOL (OS/390 および VM 版) バージョン 2 リリース 2 および Enterprise COBOL は、新しいコンパイラー・オプション DIAGTRUNC が有効であれば、警告メッセージを出します。

OCCURS 文節

OS/VS COBOL では、OCCURS 文節に続く句について標準以外の順序が許可されましたが、Enterprise COBOL では許可されません。

たとえば、OS/VS COBOL では以下のコード・シーケンスは許可されました。

```
01  D   PIC 999.
01  A.
    02  B OCCURS 1 TO 200 TIMES
          ASCENDING KEY C
          DEPENDING ON D
          INDEXED BY H.
    02  C PIC 99.
```

Enterprise COBOL では、上記の例は次のように変更しなければなりません。

```
01  D   PIC 999.
01  A.
    02  B OCCURS 1 TO 200 TIMES
          DEPENDING ON D
          ASCENDING KEY C
          INDEXED BY H.
    02  C PIC 99.
```

OPEN REVERSED ステートメント

OS/VS COBOL は、複数リール・ファイル用の REVERSED 句を受け入れていましたが、Enterprise COBOL では受け入れません。

PERFORM ステートメント — 2 番目の UNTIL

OS/VS COBOL では、次の例に示されているように、PERFORM ステートメントで 2 番目の UNTIL を使用できました。

```
PERFORM CHECK-FOR-MATCH THRU CHECK-FOR-MATCH-EXIT
    UNTIL PARM-COUNT = 7
    OR UNTIL SSREJADV-EOF.
```

Enterprise COBOL では、2 番目の UNTIL ステートメントを許可しません。以下の例に示すように、除去する必要があります。

```
PERFORM CHECK-FOR-MATCH THRU CHECK-FOR-MATCH-EXIT
    UNTIL PARM-COUNT = 7
    OR SSREJADV-EOF.
```

区域 A におけるピリオド

OS/VS COBOL では、区域 A で、無効な区域 A 項目 (または項目なし) の後にピリオドをコーディングすることができました。Enterprise COBOL では、区域 A におけるピリオドは有効な区域 A 項目の後になければなりません。

任意の部における連続したピリオド

OS/VS COBOL では、任意の部で 2 つの連続したピリオドをコーディングすることができました。

Enterprise COBOL では、1 つの行で 2 つのピリオドが検出されると、警告メッセージ (RC = 4) が出される (PROCEDURE DIVISION の場合) か、または重大メッセージ (RC = 12) が出されます (ENVIRONMENT DIVISION または DATA DIVISION の場合)。

以下の例の場合、OS/VS COBOL では受け入れられますが、Enterprise COBOL では重大 (RC = 12) エラーおよび警告 (RC = 4) が出されます。

```
WORKING-STORAGE SECTION.
01 A PIC 9. .
⋮
    MOVE 1 TO A. .
   ⋮
    GOBACK.
```

SD、FD、または RD の終わりで欠落しているピリオド

ソート記述、ファイル記述、または報告書記述の終わり (01 レベル標識の前) にピリオドが必要です。

OS/VS COBOL は、欠落しているピリオドを診断し、警告メッセージ (RC = 4) を出しました。

Enterprise COBOL は、エラー・メッセージ (RC = 8) を出します。

段落名で欠落しているピリオド

OS/VS COBOL のリリース 2.4 より前のリリースは、後にピリオドのない段落名を受け入れました。OS/VS COBOL リリース 2.4 は警告メッセージ (RC = 4) を出しました。Enterprise COBOL はエラー・メッセージ (RC = 8) を出します。

文書化されていない OS/VS COBOL 拡張

PICTURE ストリング

OS/VS COBOL は、暗黙の小数点の左側がすべて Z であり、暗黙の小数点のすぐ右側が Z であるが、9 または 9- で終わる PICTURE ストリングを受け入れました。以下に例を示します。

```
05 WEIRD-NUMERIC-EDITED PIC Z(11)VZ9.
```

Enterprise COBOL は上記の例のようなステートメントを受け入れません。Z9 を ZZ または 99 に変更しなければなりません。

固有でない PROGRAM-ID 名

OS/VS COBOL では、データ名または段落名が PROGRAM-ID 名と同じであることが許可されました。Enterprise COBOL では、PROGRAM-ID 名は固有であることが必要です。

修飾 — 同じ句の反復使用

```
A of B of B
```

OS/VS COBOL では句の繰り返しを許可していましたが、Enterprise COBOL では許可していません。

READ ステートメント — KEY 句内の再定義されたレコード・キー

OS/VS COBOL は、READ ステートメントの KEY 句内の暗黙的または明示的に再定義されたレコード・キーを受け入れました。

Enterprise COBOL は、読み取られるファイル用の SELECT 文節でレコード・キーとして指定されたデータ項目の名前だけを受け入れます。

RECORD CONTAINS n CHARACTERS 文節

COBOL 74 標準との相違点として、OS/VS COBOL プログラムの RECORD CONTAINS n CHARACTERS 文節は、FD 内で OCCURS DEPENDING ON 文節が指定されるとオーバーライドされ、固定長レコードではなく可変長レコードを含むファイルが生成されていました。

Enterprise COBOL では、RECORD CONTAINS n CHARACTERS 文節は固定長レコードを含むファイルを生成します。

RECORD KEY 句および ALTERNATE RECORD KEY 句

OS/VS COBOL では、ALTERNATE RECORD KEY data-name-4 の左端の文字位置が RECORD KEY または他の任意の ALTERNATE RECORD KEY 句の左端の文字位置と同じであることが許可されました。

Enterprise COBOL では、これは許可されません。

SD または FD 記入項目内の REDEFINES 文節

OS/VS COBOL リリース 2.4 以前のリリースでは、レベル 01 の SD または FD 内に記述した REDEFINES 文節を受け入れていました。Enterprise COBOL および OS/VS COBOL リリース 2.4 では受け入れません。

たとえば、以下の一連のコードは無効になります。

```
SD ...
01  SORT-REC-HEADER.
    05  SORT-KEY          PIC X(20).
    05  SORT-HEADER-INFO  PIC X(40).
    05  FILLER            PIC X(20).
01  SORT-REC-DETAIL REDEFINES SORT-REC-HEADER.
    05  FILLER            PIC X(20).
    05  SORT-DETAIL-INFO  PIC X(60).
```

Enterprise COBOL で類似した機能を得るためには、REDEFINES 文節を削除してください。

テーブルを指定した REDEFINES 文節

OS/VS COBOL では、REDEFINES 文節内でテーブルを指定することができました。たとえば、以下の例の場合、OS/VS COBOL は警告メッセージ (RC = 4) を出します。

```
01 E.
    03 F OCCURS 10.
        05  G PIC X.
    03 I REDEFINES F PIC X.
```

Enterprise COBOL は、テーブルの再定義を許可しないため、上記の例の場合は重大 (RC = 12) メッセージを出します。

比較条件

OS/VS COBOL リリース 2.4 以前のリリースでは、比較条件内の無効な演算子を受け入れていました。次の表に、OS/VS COBOL リリース 2.3 では受け入れられ、Enterprise COBOL では受け入れられない演算子をリストします。この表は、Enterprise COBOL プログラムの場合の有効なコーディングも示しています。

OS/VS COBOL R2.3	Enterprise COBOL
= TO	= または EQUAL TO
> THAN	> または GREATER THAN
< THAN	< または LESS THAN

RENAMES 文節 — 固有でない非修飾データ名

OS/VS COBOL プログラム内の RENAMES 文節で、固有でない非修飾データ名が参照されていても、MIGR メッセージは出されません。しかし、Enterprise COBOL は、固有でない非修飾データ名の使用をサポートしません。

対応する FD のない SELECT ステートメント

OS/VS COBOL は、対応する FD 記入項目のない SELECT ステートメントを受け入れていましたが、Enterprise COBOL では受け入れません。

SORT 動詞

以前の保守レベルでは、OS/VS COBOL コンパイラーは SORT 動詞内の UNTIL および TIMES 句を受け入れました。以下に例を示します。

```
SORT FILE-1
 ON ASCENDING KEY AKEY-1
  INPUT PROCEDURE IPROC-1
   OUTPUT PROCEDURE OPROC-1
    UNTIL AKEY-1 = 99.


SORT FILE-2
 ON ASCENDING KEY AKEY-2
  INPUT PROCEDURE IPROC-2
   OUTPUT PROCEDURE OPROC-2
    10 TIMES.
```

Enterprise COBOL は上記の例のようなステートメントを受け入れません。

SORT ステートメントの正しい構文では、ASCENDING KEY または DESCENDING KEY の後で、ソート・キーであるデータ名を使用することができます。ワード KEY はオプショナルです。

OS/VS COBOL は、ASCENDING KEY の後で使用された場合の IS を受け入れました。Enterprise COBOL はこのコンテキストでの IS を受け入れません。以下に例を示します。

```
SORT SORT-FILE
     ASCENDING KEY IS SD-NAME-FIELD
     USING INPUT-FILE
     GIVING SORTED-FILE.
```

SORT または MERGE

OS/VS COBOL では、SORT または MERGE 出力 PROCEDURE 内の最初の RETURN の前に実行された SD バッファーへの MOVE は、最初のレコードのデータをオーバーレイしません。

Enterprise COBOL では、同様の MOVE が最初のレコードのデータをオーバーレイします。SORT または MERGE 操作時に、SD データ項目が使用されます。OUTPUT PROCEDURE 内で、最初の RETURN ステートメントの実行前に、その SD データ項目を使用してはなりません。最初の RETURN ステートメントの実行前にデータがこのレコード域に移動されると、最初に戻されるレコードが上書きされます。

STRING ステートメント — 送り出しフィールド ID

OS/VS COBOL では、整数ではない数値送り出しフィールド ID を使用できました。Enterprise COBOL のもとでは、数値送り出しフィールド ID は整数でなければなりません。

UNSTRING ステートメント — 'OR'、'IS'、または数字編集項目を用いるコーディング

OS/VS COBOL では、UNSTRING ステートメントに以下のいずれかの無効なコーディングが含まれていても、診断エラー・メッセージは出されませんでした。

1. 次のように、literal-1 と literal-2 の間に必須のワード 『OR』 が欠落している場合。

   ```
   UNSTRING A-FIELD DELIMITED BY '-'  ','
       INTO RECV-FIELD-1
       POINTER PTR-FIELD.
   ```

2. 次のように、ポインターの指定の中に無関係なワード 『IS』 がある場合。

   ```
   UNSTRING A-FIELD DELIMITED BY '-' OR ','
       INTO RECV-FIELD-2
       POINTER IS PTR-FIELD.
   ```

3. 次のように、UNSTRING ステートメントのソースとして数字編集項目を使用している場合。

   ```
   01 NUM-ED-ITEM    PIC $$9.99+
      ⋮
       UNSTRING NUM-ED-ITEM DELIMITED BY '$'
            INTO RECV-FIELD-1
            POINTER PTR-FIELD
   ```

   Enterprise COBOL では、UNSTRING ステートメント内の送り出し側として非数値データ項目のみが許可されます。

Enterprise COBOL は、これらのエラーのいずれかを含んでいる UNSTRING ステートメントが検出されると、メッセージを出します。

UNSTRING ステートメント — 複数の INTO 句

OS/VS COBOL は、複数の INTO 句が指定されていると、警告 (RC = 4) メッセージを出しました。以下に例を示します。

```
UNSTRING ID-SEND DELIMITED BY ALL "*"
   INTO ID-R1 DELIMITER IN ID-D1 COUNT IN ID-C1
   INTO ID-R2 DELIMITER IN ID-D2 COUNT IN ID-C2
   INTO ID-R2 DELIMITER IN ID-D3 COUNT IN ID-C3
```

Enterprise COBOL では、UNSTRING ステートメントで複数の INTO 句を使用することはできません。

VALUE 文節 — 符号付き値と PICTURE 文節との関係

OS/VS COBOL では、PICTURE 文節が無符号である場合に、VALUE 文節のリテラルに符号を付けることができました。

Enterprise COBOL では、VALUE 文節のリテラルは PICTURE 文節と一致しなければならず、符号を除去しなければなりません。

OS/VS COBOL から変更された言語エレメント

Enterprise COBOL では、COBOL 85 標準に準拠するために、以下の OS/VS COBOL 言語エレメントが変更されました。いくつかの言語エレメントは、言語の構文が変更されています。また、言語の構文は変更されていなくても、実行結果が異なる可能性がある (セマンティクスが変更されている) 言語エレメントもあります。

リストされているそれぞれの言語エレメントごとに、結果の違いと必要な処置が簡単に説明されています。さらに、必要な場合には、説明内容を明確にするためのコーディング例も示されています。

ALPHABETIC クラスの変更

OS/VS COBOL では、大文字とスペース文字だけが ALPHABETIC であると見なされました。

Enterprise COBOL では、大文字、小文字、およびスペース文字が ALPHABETIC であると見なされます。

OS/VS COBOL プログラムで ALPHABETIC クラス・テストを使用しており、テストされるデータに大文字と小文字が混在していると、実行結果が異なる可能性があります。このような場合は、OS/VS COBOL の ALPHABETIC テストの代わりに Enterprise COBOL の ALPHABETIC-UPPER クラス・テストを使用すると、同じ結果を得ることができます。

ALPHABET 文節の変更 — ALPHABET キーワード

OS/VS COBOL では、ALPHABET 文節内でキーワード ALPHABET が使用できませんでした。

Enterprise COBOL では、ALPHABET キーワードは必須です。

算術ステートメントの変更

Enterprise COBOL では、以下の算術項目の精度が向上しました。

- 浮動小数点データ項目の使用
- 浮動小数点リテラルの使用
- 分数による指数の表記

したがって、これらの項目を含んでいる算術ステートメントの場合、Enterprise COBOL は OS/VS COBOL よりも正確な結果を提供する可能性があります。これらの変更がアプリケーションに悪影響を与えないことを確認するために、アプリケーションをテストする必要があります。

ASSIGN 文節の変更

Enterprise COBOL は、以下の形式の ASSIGN 文節だけをサポートします。

```
ASSIGN TO assignment-name
```

ここで、*assignment-name* は以下の形式にすることができます。

QSAM ファイル

[comments-][S-]name

VSAM 順次ファイル

[comments-][AS-]name

VSAM 索引付きまたは相対ファイル

[comments-]name

LINE SEQUENTIAL ファイル

[comments-]name

OS/VS COBOL プログラムで別の形式の ASSIGN 文節、または別の形式の *assignment-name* を使用している場合は、それを、Enterprise COBOL でサポートされる形式に準拠するように変更しなければなりません。

PICTURE 文節内の B 記号 — 評価の変更

OS/VS COBOL は、英字項目の定義における PICTURE 記号 A および B を受け入れました。

Enterprise COBOL は PICTURE 記号 A だけを受け入れます (記号 A と記号 B の両方を含んでいる PICTURE は、英数字編集項目を定義します)。

この変更により、以下のものの評価について、OS/VS COBOL と Enterprise COBOL の間で実行の違いが生じる可能性があります。

- CANCEL ステートメント
- CALL ステートメント
- クラス・テスト
- STRING ステートメント

CALL ステートメントの変更

OS/VS COBOL は、CALL ステートメントの USING 句における段落名、セクション名、およびファイル名を受け入れました。

Enterprise COBOL の CALL ステートメントの USING 句では、プロシージャー名は受け入れられず、QSAM ファイル名だけが受け入れられます。したがって、プロシージャー名は除去しなければならず、さらに、CALL ステートメントの USING 句で使用するファイル名が QSAM 物理順次ファイルの名前であることを確認する必要があります。

アセンブラー・プログラムを呼び出し、プロシージャー名を渡す OS/VS COBOL プログラムを移行するためには、アセンブラー・ルーチンを書き直す必要があります。OS/VS COBOL プログラムでは、アセンブラー・ルー

チンを、パラメーターとして渡された段落名からアドレス (またはアドレスのリスト) を受け取るように書くことができます。アセンブラー・ルーチンは、エラーが発生した場合、このアドレスを使用してメインプログラム内の代替位置に戻ることができます。

Enterprise COBOL では、アセンブラー・ルーチンを、数値を割り当てて起点に戻るようにコーディングしてください。アセンブラー・プログラム内でエラーが発生した場合、この数値を使用して呼び出しルーチン内の代替位置に進むことができます。

たとえば、OS/VS COBOL の以下のアセンブラー・ルーチンは Enterprise COBOL では無効です。

```
CALL "ASMMOD" USING PARAMETER-1,
                    PARAGRAPH-1,
                    PARAGRAPH-2,
NEXT STATEMENT.
   ⋮
PARAGRAPH-1.
   ⋮
PARAGRAPH-2.
```

上記のサンプル・コードは、Enterprise COBOL でコンパイルするには、以下の例のように書き換える必要があります。

```
CALL "ASMMOD" USING PARAMETER-1,
                    PARAMETER-2.
IF PARAMETER-2 NOT = 0
   GOTO PARAGRAPH-1,
        PARAGRAPH-2,
        DEPENDING ON PARAMETER-2.
```

この例では、アセンブラー・プログラム (ASMMOD) を、代替位置に分岐しないように変更します。その代わりに、エラーがなければ数値ゼロ、エラーが発生すればゼロ以外の戻り値を呼び出しルーチンに渡すようにします。ゼロ以外の戻り値は、COBOL プログラム内のどの段落がエラー条件を処理するのかを判別するために使用されます。

多くの COBOL プログラマーが、特定のエラーまたは条件が存在するときに制御を取得するために、390 SPIE 機構を使用するアセンブラー・プログラムをコーディングしています。これらのルーチンは、SPIE ルーチンに渡された名前を持つ段落で COBOL プログラムに制御を渡すことができます。これらのユーザー作成 SPIE ルーチンを使用するアプリケーションは、言語環境プログラムの条件処理を使用するように変換してください。

簡略複合比較条件の変更

以下の 3 つの考慮事項が、簡略複合比較条件に影響を与えます。

- NOT および論理演算子 / 関係演算子の評価
- 括弧の評価
- オプショナル・ワード IS

以下のセクションでこれらを説明します。

NOT および論理演算子 / 関係演算子の評価: LANGLVL(1) を用いる OS/VS COBOL は、以下のように、簡略複合比較条件内での NOT の使用を受け入れます。

- 比較条件のサブジェクトだけが暗黙指定されているときは、NOT は論理演算子であると見なされます。以下に例を示します。

  ```
  A = B AND NOT LESS THAN C OR D
  ```

  これは以下と同等です。

  ```
  ((A = B) AND NOT (A < C) OR (A < D))
  ```

- サブジェクトと関係演算子の両方が暗黙指定されているときは、NOT は関係演算子の一部であると見なされます。

  以下に例を示します。

  ```
  A > B AND NOT C
  ```

  これは以下と同等です。

  ```
  A > B AND A NOT > C
  ```

LANGLVL(2) を用いる OS/VS COBOL および Enterprise COBOLでは、簡略複合比較条件内の NOT は以下のように見なされます。

- NOT GREATER THAN、NOT >、NOT LESS THAN、NOT <、NOT EQUAL TO、および NOT = の形の場合は、関係演算子の一部であると見なされます。以下に例を示します。

  ```
  A = B AND NOT LESS THAN C OR D
  ```

  これは以下と同等です。

  ```
  ((A = B) AND (A NOT < C) OR (A NOT < D))
  ```

- その他の位置にある NOT は、論理演算子であると見なされます (したがって、否定比較条件になります)。以下に例を示します。

  ```
  A > B AND NOT C
  ```

  これは以下と同等です。

  ```
  A > B AND NOT A > C
  ```

LANGLVL(1) を用いる OS/VS COBOL から移行する場合、予期したとおりの実行結果を得るようにするためには、すべての簡略複合条件を、簡略化しない完全な形に展開してください。

括弧の評価: OS/VS COBOL は、簡略複合比較条件内での括弧の使用を受け入れました。

Enterprise COBOL は、ほとんどの括弧の使用を IBM 拡張としてサポートします。ただし、いくつかの違いがあります。

- 簡略複合比較条件の有効範囲内では、Enterprise COBOL は括弧の内側の関係演算子をサポートしません。以下に例を示します。

  ```
  A = B  AND ( < C OR D)
  ```

- 比較条件内での括弧の誤った使用の一部は、OS/VS COBOL では受け入れられましたが、Enterprise COBOL では受け入れられません。以下に例を示します。

  ```
  (A = 0 AND B) = 0
  ```

オプショナル・ワード IS: OS/VS COBOL は、簡略複合比較条件内のオブジェクトの直前にあるオプショナル・ワード IS を受け入れました。以下に例を示します。

```
A = B OR IS C AND IS D
```

Enterprise COBOL は、オプショナル・ワード IS のこの用法を受け入れません。Enterprise COBOL では、このように使用されているワード IS を削除してください。

注: Enterprise COBOL では、オプショナル・ワード IS を簡略複合比較条件内の関係演算子の一部として使用することが許可されます。以下に例を示します。

```
A = B OR IS = C AND IS = D
```

関連した名前を指定した COPY ステートメント

LANGLVL(1) を用いる OS/VS COBOL では、COPY ステートメントの前に 01 レベルの標識を付けることができました。この 01 レベルの名前は COPY メンバー内の 01 レベルの名前を置き換えることになります。たとえば、COPY メンバー MBR-A の内容が以下の場合、

```
01 RECORD-A.
   05 FIELD-A...
   05 FIELD-B...
```

次のような COPY ステートメントを使用すると、

```
01  RECORD1 COPY MBR-A.
```

結果のソースは次のようになります。

```
01  RECORD1.
    05 FIELD-A...
    05 FIELD-B...
```

Enterprise COBOL はこの COPY ステートメントを受け入れません。Enterprise COBOL でコンパイルするためには、以下のステートメントを使用してください。

```
01  RECORD1.
    COPY MBR-A REPLACING ==01 RECORD-A== BY == ==.
```

CURRENCY-SIGN 文節の変更 — ’/’、’=’、および ’L’ 文字

LANGLVL(1) を用いる OS/VS COBOL は、CURRENCY-SIGN 文節内の ’/’ (スラッシュ) 文字、’L’ 文字、および ’=’ (等号) を受け入れました。

Enterprise COBOL は、これらの文字を有効として受け入れません。さらに、Enterprise COBOL は、DBCS データ項目を使用するプログラムの場合に、文字 G を受け入れません。

CURRENCY SIGN 文節にこれらの文字がある場合は、これらの文字を除去しなければなりません。

ENTRY ポイントへの動的 CALL ステートメント

OS/VS COBOL では、場合によっては、CANCEL を介在させずにサブプログラムの代替入り口点への動的 CALL ステートメントを使用することができました。

Enterprise COBOL では、介在する CANCEL が常に必要です。これらのプログラムを移行するときは、サブプログラムの代替 ENTRY ポイントを参照する動的 CALL ステートメントの間に介在する CANCEL を追加してください。

OS/VS COBOL から変更された言語エレメント

EXIT PROGRAM/GOBACK ステートメントの変更

OS/VS COBOL では、EXIT PROGRAM または GOBACK ステートメントが実行されるときに、その中の PERFORM ステートメントが範囲の終わりに達していないと、その PERFORM ステートメントは未完了の状態のままになりました。

Enterprise COBOL では、EXIT PROGRAM または GOBACK ステートメントが実行されるときには、その中のすべての PERFORM ステートメントが範囲の終わりに達しているものと見なされます。

FILE STATUS 文節の変更

Enterprise COBOL では、状況キーの値が、OS/VS COBOL から受け取られるものから変更されました。

- QSAM ファイルについては、表 31を参照してください。
- VSAM ファイルについては、 163 ページの表 32を参照してください。

OS/VS COBOL プログラムで、実行の進路を判別するために状況キー値を使用している場合は、プログラムを、新しい状況キー値を使用するように変更しなければなりません。Enterprise COBOL ファイル状況コードの詳細については、*Enterprise COBOL 言語解説書* を参照してください。

表 31. 状況キーの値 — QSAM ファイル

OS/VS	Enterprise COBOL	意味
(未定義)	04	誤長レコード。正常終了。
(未定義)	05	オプショナル・ファイルがありません。正常終了。
(未定義)	07	OPEN または CLOSE に NO REWIND/REEL/UNIT/FOR REMOVAL が指定されましたが、ファイルがリール / 装置メディア上にありません。正常終了。
00	00	正常終了。
10	10	At END (次の論理レコードがありません)。正常終了。
30	30	永続エラー。
34	34	永続エラー。ファイル境界違反。
90	90	その他のエラー (これ以上の情報はありません)。
90	35	非オプショナル・ファイルがありません。
90	37	装置タイプの矛盾。
90	39	固定ファイル属性の矛盾。OPEN が失敗します。
90	96	ファイル識別がありません (ファイル用の DD ステートメントがありません)。
92	38	WITH LOCK でクローズされたファイルに対して OPEN が試みられました。
92	41	OPEN モードのファイルに対して OPEN が試みられました。

表 31. 状況キーの値 — QSAM ファイル (続き)

OS/VS	Enterprise COBOL	意味
92	42	OPEN モードでないファイルに対して CLOSE が試みられました。
92	43	最後の入出力ステートメントが READ でないときに REWRITE が試みられました。
92	44	異なるサイズのレコードで順次ファイル・レコードの再書き込みが試みられました。
92	46	有効な次のレコードがない状況で順次 READ が試みられました。
92	47	ファイルが OPEN INPUT または I-O モードでないときに READ が試みられました。
92	48	ファイルが OPEN OUTPUT、I-O、または EXTEND モードでないときに WRITE が試みられました。
00	48	ファイルが OPEN I-O モードのときに WRITE が試みられました。
92	49	ファイルが OPEN I-O モードでないときに DELETE または REWRITE が試みられました。
92	92	論理エラー。

表 32. 状況キーの値 — VSAM ファイル

OS/VS	Enterprise COBOL	意味
(未定義)	14	相対ファイルの順次 READ で、相対レコード番号のサイズが相対キーにとって大きすぎました。
00	00	正常終了。
00	04	誤長レコード。正常終了。
00	05	オプショナル・ファイルがありません。正常終了。
00	35	非オプショナル・ファイルがありません。ファイルが空のときに発生する可能性があります。
02	02	重複キーがあり、DUPLICATES が指定されています。正常終了。
10	10	At END (次の論理レコードがありません)。正常終了。
21	21	VSAM 索引付きまたは相対ファイルのキーが無効です。シーケンス・エラー。
22	22	VSAM 索引付きまたは相対ファイルのキーが無効です。重複キーおよび重複は許可されません。
23	23	VSAM 索引付きまたは相対ファイルのキーが無効です。レコードが見つかりません。

表 32. 状況キーの値 — VSAM ファイル (続き)

OS/VS	Enterprise COBOL	意味
24	24	VSAM 索引付きまたは相対ファイルのキーが無効です。ファイル境界を超える書き込みが試みられました。 Enterprise COBOL: 相対ファイルへの WRITE で、相対レコード番号のサイズが相対キーにとって大きすぎました。
30	30	永続エラー。
90	37	大容量記憶装置上にないファイルのオープンが試みられました。
90	90	その他のエラー (これ以上の情報はありません)。
91	91	VSAM パスワード障害。
92	41	OPEN モードのファイルに対して OPEN が試みられました。
92	42	OPEN モードでないファイルに対して CLOSE が試みられました。
92	43	最後の入出力ステートメントが READ または DELETE でないときに REWRITE が試みられました。
92	47	ファイルが OPEN INPUT または I-O モードでないときに READ が試みられました。
92	48	ファイルが OPEN OUTPUT、I-O、または EXTEND モードでないときに WRITE が試みられました。
92	49	ファイルが OPEN I-O モードでないときに DELETE または REWRITE が試みられました。
93	93	VSAM リソースが使用可能ではありません。
93 96	35	非オプショナル・ファイルがありません。
94	46	有効な次のレコードがない状況で順次 READ が試みられました。
95	39	固定ファイル属性の矛盾。OPEN が失敗します。
95	95	VSAM ファイル情報が無効または不完全です。
96	96	ファイル識別がありません (この VSAM ファイル用の DD ステートメントがありません)。
97	97	OPEN ステートメントの実行が正常に終了しました。ファイルの保全性が検査されました。

IF . . . OTHERWISE ステートメントの変更

OS/VS COBOL では、次のような非標準形式の IF ステートメントを使用できました。

```
IF condition THEN statement-1 OTHERWISE statement-2
```

Enterprise COBOL では、次のような標準形式の IF ステートメントだけが使用できます。

```
IF condition THEN statement-1 ELSE statement-2
```

したがって、非標準形式の IF...OTHERWISE ステートメントを含んでいる OS/VS COBOL プログラムは、標準形式の IF...ELSE ステートメントに変更する必要があります。

JUSTIFIED 文節の変更

LANGLVL(1) を用いる OS/VS COBOL では、データ記述記入項目で VALUE 文節と一緒に JUSTIFIED 文節が指定されると、初期データは右寄せされます。以下に例を示します。

```
77  DATA-1 PIC X(9) JUSTIFIED VALUE "FIRST".
```

この結果、“FIRST” は DATA-1 の右端の 5 つの文字位置を占めます。

```
bbbbFIRST
```

Enterprise COBOL では、JUSTIFIED 文節は、データ項目内のデータの初期配置に影響を与えません。英字または英数字項目について VALUE と JUSTIFIED の両方の文節が指定されると、初期値はデータ項目内で左寄せされます。以下に例を示します。

```
77  DATA-1 PIC X(9) JUSTIFIED VALUE "FIRST".
```

この結果、“FIRST” は DATA-1 の左端の 5 つの文字位置を占めます。

```
FIRSTbbbb
```

Enterprise COBOL で結果が変わらないようにするためには、DATA-1 の 9 つすべての文字位置を占めるリテラル値を指定することができます。以下に例を示します。

```
77  DATA-1 PIC X(9) JUSTIFIED VALUE "    FIRST".
```

これは、DATA-1 の値を右寄せしています。

```
bbbbFIRST
```

MOVE ステートメントおよび比較 — 位取りの変更

LANGLVL(1) を用いる OS/VS COBOL では、MOVE ステートメント内の送り出しフィールドまたは比較内のフィールドが位取りされた整数であり (つまり、右端の PICTURE 記号が文字 P であり)、受け取りフィールド (または比較されるフィールド) が英数字または数字編集である場合、後続ゼロ (0) は切り捨てられます。

たとえば、以下の MOVE ステートメント

```
05  SEND-FIELD     PICTURE 999PPP VALUE 123000.
05  RECEIVE-FIELD  PICTURE XXXXXX.
  ⋮
    MOVE SEND-FIELD TO RECEIVE-FIELD.
```

が実行されると、RECEIVE-FIELD には 値 123*bbb* (左寄せされた) が入ります。ここでは、’*b*’ は 1 つのブランク文字を表しています。

Enterprise COBOL の場合、MOVE ステートメントでは後続ゼロが転送され、比較ではそれらが組み込まれます。

たとえば、以下の MOVE ステートメント

```
05  SEND-FIELD     PICTURE 999PPP VALUE 123000.
05  RECEIVE-FIELD  PICTURE XXXXXX.
   ⋮
    MOVE SEND-FIELD TO RECEIVE-FIELD.
```

が実行されると、RECEIVE-FIELD には値 123000 が入ります。

グループ項目に対する数値クラス・テスト

OS/VS COBOL では、IF NUMERIC クラス・テストを、1 つ以上の符号付き基本項目を含んでいるグループ項目と共に使用できました。

たとえば、`IF grp1 IS NUMERIC`。ここで、`grp1` はグループ項目です。

```
01 grp1.
   03 yy PIC S99.
   03 mm PIC S99.
   03 dd PIC S99.
```

Enterprise COBOL では、IF NUMERIC クラス・テストを符号付き従属項目を持つグループ項目に対して使用すると、S レベルのメッセージが出されます。

数値データの変更

Enterprise COBOL は、10 進データ用に生成されたコードを変更するために NUMPROC コンパイラー・オプションを使用します。NUMPROC(MIG) は、OS/VS COBOL の場合と非常に類似した処理を引き起こしますが、すべてのケースで結果が同じであるとは限りません。MOVE ステートメント、比較、および算術ステートメントの結果は、OS/VS COBOL の場合と異なる可能性があります (特に、フィールドが初期設定されていない場合)。

たとえば、Enterprise COBOL では負のゼロを結果として生成しませんが、OS/VS COBOL は生成する可能性がありました。さらに、Enterprise COBOL では NUMPROC(MIG) を指定した入力データの無効な符号を修正しませんが、OS/VS COBOL プログラムでは、入力データの矛盾する符号を修正していました。

データ例外を利用して、無効な内容の 10 進データ項目を識別したり、異常終了させたりしているプログラムは、10 進データ項目内のデータを検証するクラス・テストを使用するように変更が必要な場合があります。

OCCURS DEPENDING ON 文節 — ASCENDING および DESCENDING KEY 句

OS/VS COBOL は、OCCURS DEPENDING ON 文節の ASCENDING および DESCENDING KEY 句内の可変長キーを IBM 拡張として受け入れていました。

Enterprise COBOL では、ASCENDING または DESCENDING KEY 句内に可変長キーを指定できません。

OCCURS DEPENDING ON 文節 — 受け取り項目の値の変更

OS/VS COBOL では、OCCURS DEPENDING ON (ODO) オブジェクトの現行値が送り出し項目と受け取り項目の両方について常に使用されます。

Enterprise COBOL では、送り出し項目については ODO オブジェクトの現行値が使用されます。受け取り項目については、以下の長さが使用されます。

- グループ項目に ODO のサブジェクトとオブジェクトの両方が含まれており、同じレコード内でそのグループ項目の後に非従属データ項目が続いていない場合は、項目の最大長が使用されます。
- グループ項目に ODO のサブジェクトとオブジェクトの両方が含まれており、同じレコード内でそのグループ項目の後に非従属データ項目が続いている場合は、受け取り項目の実際の長さが使用されます。
- グループ項目に ODO のサブジェクトが含まれているが、オブジェクトが含まれていない場合は、項目の実際の長さが使用されます。

最大長が使用されるときは、テーブルがデータを受け取る前に ODO オブジェクトを初期設定する必要はありません。位置が ODO オブジェクトの値によって異なる項目については、それらを CALL ステートメントの USING 句で使用する前に、OCCURS DEPENDING ON 文節のオブジェクトを設定することが必要です。Enterprise COBOL のもとでは、可変位置ではない可変長グループの場合、項目が CALL ステートメントの USING BY REFERENCE 句で使用されるときに、そのオブジェクトを設定する必要はありません。これは、上記の 2 番目の中黒で記述されているグループについても該当します。

以下に例を示します。

```
01  TABLE-GROUP-1
    05  ODO-KEY-1 PIC 99.
    05  TABLE-1 PIC X(9)
        OCCURS 1 TO 50 TIMES DEPENDING ON ODO-KEY-1.
01  ANOTHER-GROUP.
    05  TABLE-GROUP-2.
        10  ODO-KEY-2 PIC 99.
        10  TABLE-2 PIC X(9)
            OCCURS 1 to 50 TIMES DEPENDING ON ODO-KEY-2.
    05  VARIABLY-LOCATED-ITEM PIC X(200).
   ⋮
PROCEDURE DIVISION.
   ⋮
   MOVE SEND-ITEM-1 TO TABLE-GROUP-1
   ⋮
   MOVE ODO-KEY-X TO ODO-KEY-2
   MOVE SEND-ITEM-2 TO TABLE-GROUP-2.
```

TABLE-GROUP-1 が受け取り項目であるときには、Enterprise COBOL は、その項目についての最大数の文字位置 (TABLE-1 の 450 バイト + ODO-KEY-1 の 2 バイト) を移動します。したがって、SEND-ITEM-1 データを TABLE-1 に移動する前に、TABLE-1 の長さを初期設定する必要はありません。

しかし、レコード記述の中で、TABLE-GROUP-2 の後には非従属データ項目 VARIABLY-LOCATED-ITEM が続いています。この場合には、Enterprise COBOL は ODO-KEY-2 内の実際の値を使用して TABLE-GROUP-2 の長さを計算します。ユーザーは、SEND-ITEM-2 データをグループ受け取り項目に移動する前に、ODO-KEY-2 をその有効な現在の長さに設定しなければなりません。

ON SIZE ERROR 句 — 中間結果の変更

OS/VS COBOL の場合、DIVIDE および MULTIPLY ステートメントの SIZE ERROR 句は中間結果と最終結果の両方に適用されました。

OS/VS COBOL から変更された言語エレメント

Enterprise COBOL の場合、DIVIDE および MULTIPLY ステートメントの SIZE ERROR 句は最終結果にのみ適用されます。これは、COBOL 74 標準と COBOL 85 標準間の変更です。この変更は既存のプログラムに影響を与える場合と与えない場合があります。

したがって、OS/VS COBOL プログラムが中間結果の SIZE ERROR 検出に依存している場合は、プログラムを変更することが必要になる可能性があります。

オプショナル・ワード IS

OS/VS COBOL プログラムの場合、簡略複合比較条件内のオブジェクトの直前にオプショナル・ワード IS があっても、MIGR メッセージは出されませんでした。以下に例を示します。

```
A = B OR IS C AND IS D
```

Enterprise COBOL は、オプショナル・ワード IS のこの用法を受け入れません。Enterprise COBOL では、このように使用されているワード IS を削除してください。

注: Enterprise COBOL では、オプショナル・ワード IS を簡略複合比較条件内の関係演算子の一部として使用することが許可されます。以下に例を示します。

```
A = B OR IS = C AND IS = D
```

PERFORM ステートメント — VARYING/AFTER 句の変更

OS/VS COBOL では、VARYING/AFTER が指定された PERFORM ステートメントで、内部条件が TRUE としてテストされると、2 つのアクションが起こります。

1. 内部条件に関連した ID/ 指標が、その現在の FROM 値に設定されます。
2. 外部条件に関連した ID/ 指標が、その現在の BY 値だけ増大されます。

Enterprise COBOL では、そのような PERFORM ステートメントで、内部条件が TRUE としてテストされると、以下のアクションが起こります。

1. 外部条件に関連した ID/ 指標が、その現在の BY 値だけ増大されます。
2. 内部条件に関連した ID/ 指標が、その現在の FROM 値に設定されます。

次の例は、実行の違いを示しています。

```
PERFORM ABC VARYING X FROM 1 BY 1 UNTIL X > 3
            AFTER Y FROM X BY 1 UNTIL Y > 3
```

OS/VS COBOL では、ABC は以下の値で 8 回実行されます。

```
X:  1  1  1  2  2  2  3  3
Y:  1  2  3  1  2  3  2  3
```

Enterprise COBOL では、ABC は以下の値で 6 回実行されます。

```
X:  1  1  1  2  2  3
Y:  1  2  3  2  3  3
```

以下のように、ネストされた PERFORM ステートメントを使用することによって、OS/VS COBOL の場合と同じ処理結果を得ることができます。

```
MOVE 1 TO X, Y, Z
PERFORM EX-1 VARYING X FROM 1 BY 1 UNTIL X > 3
   ⋮
EX-1.
    PERFORM ABC VARYING Y FROM Z BY 1 UNTIL Y > 3.
    MOVE X TO Z.
ABC.
```

PROGRAM COLLATING SEQUENCE 文節の変更

OS/VS COBOL では、PROGRAM COLLATING SEQUENCE 文節の *alphabet-name* で指定された照合シーケンスは、INSPECT、STRING、および UNSTRING ステートメントの実行中に暗黙に実行される比較に適用されます。

Enterprise COBOL では、*alphabet-name* で指定された照合シーケンスは、これらの暗黙の比較には使用されません。

READ および RETURN ステートメントの変更 — INTO 句

送り出しフィールドを、READ または RETURN...INTO identifier ステートメントに関連付けられた移動に合わせて選択する場合、OS/VS COBOL および Enterprise COBOL では、送り出しフィールドとして FD または SD から異なるレコードを選択することができます。このことは、レコード記述が PICTURE 文節を持っているときに、暗黙の基本 MOVE にのみ影響を与えます。

RERUN 文節の変更

RERUN 文節が指定されると、OS/VS COBOL では最初のレコードでチェックポイントが取られますが、Enterprise COBOL では取られません。

RESERVE 文節の変更

OS/VS COBOL は、以下の形式の FILE CONTROL 段落 RESERVE 文節をサポートしました。

```
RESERVE NO ALTERNATE AREA
RESERVE NO ALTERNATE AREAS
RESERVE integer ALTERNATE AREA
RESERVE integer ALTERNATE AREAS
RESERVE integer AREA
RESERVE integer AREAS
```

Enterprise COBOL は、以下の形式の RESERVE 文節だけをサポートします。

```
RESERVE integer AREA
RESERVE integer AREAS
```

OS/VS COBOL プログラムで RESERVE integer ALTERNATE AREA または RESERVE integer ALTERNATE AREAS 形式を使用している場合、Enterprise COBOL のもとで同等の処理を得るためには、*integer + 1* AREA(S) を指定した RESERVE 文節を使用しなければなりません。つまり、OS/VS COBOL の `RESERVE 2 ALTERNATE AREAS` 句は、Enterprise COBOL の `RESERVE 3 AREAS` と同等です。

OS/VS COBOL から変更された言語エレメント

LANGLVL(1) を用いる OS/VS COBOL のもとでは、RESERVE integer AREAS 形式の解釈は、Enterprise COBOL におけるこの形式の解釈と異なります。

LANGLVL(1) を使用し、RESERVE integer AREA または RESERVE integer AREAS 形式を使用している場合、Enterprise COBOL のもとで同等の処理を得るためには、*integer + 1* AREA(S) を指定した RESERVE 文節を使用しなければなりません。

予約語リストの変更

Enterprise COBOL と OS/VS COBOL では予約語リストに違いがあります。 275 ページの『付録 B. COBOL 予約語の比較』に予約語の完全なリストが記載されています。

SEARCH ステートメントの変更

OS/VS COBOL では、ASCENDING および DESCENDING KEY データ項目を SEARCH ステートメントの WHEN 比較条件のサブジェクトまたはオブジェクトとして指定することができました。

Enterprise COBOL では、WHEN 句のデータ名 (WHEN 比較条件のサブジェクト) は、このテーブル・エレメント内の ASCENDING または DESCENDING KEY データ項目でなければならず、identifier-2 (WHEN 比較条件のオブジェクト) はこのテーブル・エレメントに対応する ASCENDING または DESCENDING のキー・データ項目であってはなりません。

OS/VS COBOL は以下のコードを受け入れましたが、Enterprise COBOL では受け入れません。

```
WHEN VAL = KEY-1 ( INDEX-NAME-1 )
        DISPLAY "TABLE RECORDS OK".
```

以下の SEARCH の例は、Enterprise COBOL と OS/VS COBOL の両方で実行することができます。

```
01  VAL  PIC X.
01  TABLE-01.
    05  TABLE-ENTRY
           OCCURS 100 TIMES
           ASCENDING KEY IS KEY-1
           INDEXED BY INDEX-NAME-1.
        10  FILLER PIC X.
        10  KEY-1 PIC X.
    SEARCH ALL TABLE-ENTRY
        AT END DISPLAY "ERROR"
      WHEN KEY-1 ( INDEX-NAME-1 ) = VAL
        DISPLAY "TABLE RECORDS OK".
```

セグメント化の変更 — 独立セグメント内の PERFORM ステートメント

LANGLVL(1) を用いる OS/VS COBOL では、独立セグメント内の PERFORM ステートメントで永続セグメントを参照している場合、その独立セグメントは、実行されたプロシージャーが終了するたびに初期設定されます。

LANGLVL(2) を用いる OS/VS COBOL では、独立セグメント内の PERFORM ステートメントで永続セグメントを参照している場合、PERFORM ステートメントのそれぞれの実行ごとに 1 回だけ、実行されるプロシージャーに制御が渡されます。

Enterprise COBOL では、コンパイラーはオーバーレイを実行しません。したがって、上記の規則は適用されません。

プログラム・ロジックが OS/VS COBOL におけるこれらのセグメント化規則のインプリメンテーションのいずれかに依存している場合は、プログラムを書き直さなければなりません。

SELECT OPTIONAL 文節の変更

LANGLVL(1) を用いる OS/VS COBOL では、ファイル制御記入項目に SELECT OPTIONAL 文節が指定されると、ファイルが使用可能でない場合にプログラムが失敗します。Enterprise COBOL では、ファイル制御記入項目に SELECT OPTIONAL 文節が指定されると、ファイルが使用可能でない場合にプログラムが失敗せず、ファイル状況コード 05 が戻されます。USERMOD は、VSAM の場合のこの動作に影響を与えることができます。詳細については、以下の資料を参照してください。

- z/OS の場合: *言語環境プログラム カスタマイズ*
- OS/390 の場合: *言語環境プログラム OS/390 版 カスタマイズ*

SORT 特殊レジスター

SORT-CORE-SIZE、SORT-FILE-SIZE、SORT-MESSAGE、および SORT-MODE-SIZE 特殊レジスターは、Enterprise COBOL のもとでサポートされ、デフォルト以外の値を持っている場合に SORT インターフェースで使用されます。ただし、実行時には、個々の SORT 特殊レジスターは、SORT-CONTROL ファイルに組み込まれている制御ステートメントの対応するパラメーターによってオーバーライドされ、メッセージが出されます。さらに、プログラム内で設定されたそれぞれの SORT 特殊レジスターごとに、コンパイラー警告メッセージ (W レベル) が出されます。

OS/VS COBOL では、SORT-RETURN 特殊レジスターに、SORT の正常終了 (RC=0)、USING または GIVING ファイルに関する OPEN または入出力エラー (RC=2 〜 RC=12)、および SORT の失敗 (RC=16) を表すコードが入る可能性があります。Enterprise COBOL では、SORT-RETURN 特殊レジスターに、SORT の正常終了 (RC=0) および失敗 (RC=16) を表すコードだけが入ります。

ソース言語のデバッグの変更

Enterprise COBOL および OS/VS COBOL では、USE FOR DEBUGGING 宣言を使用してソース言語をデバッグすることができます。有効なオペランドを表 33に示します。Enterprise COBOL で無効なオペランドは、OS/VS COBOL プログラムから除去しなければなりません。デバッグ・ツールを使用して、同じデバッグ結果が得られるようにしてください。

表 33. USE FOR DEBUGGING 宣言 — 有効なオペランド

デバッグ・オペランド		プロシージャーが実行されるとき
OS/VS COBOL	**Enterprise COBOL**	
procedure-name-1	procedure-name-1	指定されたプロシージャーのそれぞれの実行の直前。 指定されたプロシージャーを参照している ALTER ステートメントの実行の直後。

表 33. USE FOR DEBUGGING 宣言 — 有効なオペランド (続き)

デバッグ・オペランド		プロシージャーが実行されるとき
OS/VS COBOL	**Enterprise COBOL**	
ALL PROCEDURES	ALL PROCEDURES	最外部プログラム内のそれぞれの非デバッグ・プロシージャーの実行の直前。 最外部プログラム内のそれぞれの ALTER ステートメント (宣言型プロシージャー内の ALTER ステートメントを除く) の実行の直後。
file-name-n	(なし)	説明については、*IBM VS COBOL for OS/VS* を参照。
ALL REFERENCES OF identifier-n	(なし)	説明については、*IBM VS COBOL for OS/VS* を参照。
cd-name-1	(なし)	説明については、*IBM VS COBOL for OS/VS* を参照。

コンパイル時にフラグ設定される範囲外添え字

Enterprise COBOL は、許容される最大値よりも大きいかまたは 1 よりも小さいリテラル添え字または指標値がコーディングされている場合は、エラー (RC = 8) メッセージを出します。このメッセージは、SSRANGE オプションが指定されているかどうかに関係なく生成されます。

OS/VS COBOL は、同等のエラー・メッセージを出しませんでした。

UNSTRING ステートメント — 添え字評価の変更

OS/VS COBOL の UNSTRING ステートメントでは、DELIMITED BY、INTO、DELIMITER IN、および COUNT IN フィールドについて、関連した添え字付け、指標付け、または長さ計算の評価は、データが受け取り項目に転送される直前に行われます。

Enterprise COBOL の UNSTRING ステートメントでは、これらのフィールドについて、関連した添え字付け、指標付け、または長さ計算の評価は 1 回だけ (区切り文字送り出しフィールドの検査の直前に) 行われます。以下に例を示します。

```
01  ABC     PIC  X(30).
01  IND.
    02 IND-1 PIC 9.
01  TAB.
    02 TAB-1 PIC X OCCURS 10 TIMES.
01  ZZ      PIC  X(30).
⋮
    UNSTRING ABC DELIMITED BY TAB-1 (IND-1) INTO IND ZZ.
```

OS/VS COBOL では、添え字 IND-1 は、2 番目の受け取り項目 ZZ が充てんされる前に再評価されます。

Enterprise COBOL では、添え字 IND-1 は、UNSTRING ステートメントの実行の開始時に 1 回だけ評価されます。

LANGLVL(1) を用いる OS/VS COBOL では、UNSTRING の DELIMITED BY ALL 句が指定されると、任意の区切り文字の 2 つ以上の連続するオカレンスが、1 つのオカレンスであるかのように扱われます。最初のオカレンスは、収容可能な限り多く、現行の区切り文字受け取りフィールド (指定されている場合) に移動されます。それ以降の各オカレンスは、そのオカレンス全体が収容される場合にのみ移動されます。OS/VS COBOL におけるこの句の動作の詳細については、*IBM VS COBOL for OS/VS* を参照してください。

Enterprise COBOL では、任意の区切り文字の 1 つ以上の連続するオカレンスは、1 つのオカレンスであるかのように扱われ、この 1 つのオカレンスが区切り文字受け取りフィールド (指定されている場合) に移動されます。

たとえば、ID-SEND に 123**45678**90AB が入っている場合、

```
UNSTRING ID-SEND DELIMITED BY ALL "*"
    INTO ID-R1 DELIMITER IN ID-D1 COUNT IN ID-C1
         ID-R2 DELIMITER IN ID-D2 COUNT IN ID-C2
         ID-R3 DELIMITER IN ID-D3 COUNT IN ID-C3
```

LANGLVL(1) を用いる OS/VS COBOL では、以下の結果になります。

```
ID-R1  123       1D-D1  **        ID-C1  3
ID-R2  45678     1D-D2  **        ID-C2  5
ID-R3  90AB      1D-D3            ID-C3  4
```

Enterprise COBOL では、以下の結果になります。

```
ID-R1  123       1D-D1  *         ID-C1  3
ID-R2  45678     1D-D2  *         ID-C2  5
ID-R3  90AB      1D-D3            ID-C3  4
```

UPSI スイッチ

OS/VS COBOL では、UPSI スイッチおよび UPSI に関連した簡略名への参照を使用できました。Enterprise COBOL では、条件名だけを使用できます。

たとえば、SPECIAL-NAMES 段落で条件名が定義されている場合、以下のものは同等です。

```
OS/VS COBOL                      Enterprise COBOL

SPECIAL-NAMES.                 SPECIAL-NAMES.
    UPSI-0 IS MNUPO                UPSI-0 IS MNUPO
                                    ON STATUS IS UPSI-0-ON
                                    OFF STATUS IS UPSI-0-OFF
        .                            .
        .                            .
        .                            .
PROCEDURE DIVISION             PROCEDURE DIVISION
        .                            .
        .                            .
        .                            .
    IF UPSI-0 = 1 ...               IF UPSI-0-ON ...
    IF MNUPO = 0 ...                IF UPSI-0-OFF ...
```

OS/VS COBOL から変更された言語エレメント

VALUE 文節の条件名

OS/VS COBOL のリリース 2.4 より前のリリースでは、VALUE 文節の条件名について、英数字フィールドを数値で初期設定することができました。以下に例を示します。

```
01 FIELD-A.
    02   LAST-YEAR  PIC XX VALUE 87.
    02   THIS-YEAR  PIC XX VALUE 88.
    02   NEXT-YEAR  PIC XX VALUE 89.
```

Enterprise COBOL は、この言語拡張を受け入れません。したがって、上記の例を訂正するには、以下の例のように、VALUE 文節に英数字値をコーディングしなければなりません。

```
01 FIELD-A.
    02   LAST-YEAR  PIC XX VALUE "87".
    02   THIS-YEAR  PIC XX VALUE "88".
    02   NEXT-YEAR  PIC XX VALUE "89".
```

WHEN-COMPILED 特殊レジスター

Enterprise COBOL および OS/VS COBOL は、WHEN-COMPILED 特殊レジスターの使用をサポートします。この特殊レジスターの使用規則は、両方のコンパイラーで同じです。ただし、データの形式が異なります。

OS/VS COBOL では、形式は次のとおりです。

```
hh.mm.ssMMM DD, YYYY  (hour.minute.secondMONTH DAY, YEAR)
```

Enterprise COBOL では、形式は次のとおりです。

```
MM/DD/YYhh.mm.ss  (MONTH/DAY/YEARhour.minute.second)
```

WRITE AFTER POSITIONING ステートメント

OS/VS COBOL は AFTER POSITIONING 句を指定した WRITE ステートメントをサポートしていましたが、Enterprise COBOL ではサポートしていません。

Enterprise COBOL では、WRITE...AFTER ADVANCING ステートメントを使用すれば、WRITE...AFTER POSITIONING と類似した動作を得ることができます。以下の 2 つの例は、OS/VS COBOL の POSITIONING 句と、Enterprise COBOL の同等の句を示しています。

リテラルを指定して WRITE . . . AFTER ADVANCING を使用する場合:

```
   OS/VS COBOL                              Enterprise COBOL

AFTER POSITIONING 0                    AFTER ADVANCING PAGE
AFTER POSITIONING 1                    AFTER ADVANCING 1 LINE
AFTER POSITIONING 2                    AFTER ADVANCING 2 LINES
AFTER POSITIONING 3                    AFTER ADVANCING 3 LINES
```

リテラル以外を指定して WRITE...AFTER ADVANCING を使用する場合:

```
WRITE OUTPUT-REC AFTER POSITIONING SKIP-CC.
```

```
   OS/VS COBOL                              Enterprise COBOL
                            SKIP-CC


AFTER POSITIONING SKIP-CC      1         AFTER ADVANCING PAGE
```

```
AFTER POSITIONING SKIP-CC     ' '        AFTER ADVANCING 1 LINE
AFTER POSITIONING SKIP-CC      0         AFTER ADVANCING 2 LINES
AFTER POSITIONING SKIP-CC      -         AFTER ADVANCING 3 LINES
```

注: Enterprise COBOL では、チャネル・スキップは、SPECIAL-NAMES への参照によってのみサポートされます。

CCCA を使用すると、WRITE . . . AFTER POSITIONING ステートメントを自動的に変換できます。たとえば、次のステートメントが指定されているとします。

```
WRITE OUTPUT-REC AFTER POSITIONING n.
```

n がリテラルである場合は、CCCA は上記の例を `WRITE...AFTER ADVANCING n LINES` に変更します。n が ID である場合は、SPECIAL-NAMES が生成され、セクションがプログラムの終わりに追加されます。

第 11 章 移行済み OS/VS COBOL プログラムのコンパイル

この章では、Enterprise COBOL コンパイラーと OS/VS COBOL コンパイラー間の違いを説明します。以下のトピックに関する情報が記載されています。

- 移行済みプログラム用の主要なコンパイラー・オプション
- サポートされない OS/VS COBOL コンパイラー・オプション
- Prolog 形式の変更点

これらのプロダクトのそれぞれに固有の情報が注記されています。

移行済みプログラム用の主要なコンパイラー・オプション

表 34 に、移行済みプログラムに特に関係があるコンパイラー・オプションをリストします。

表 34. 移行済み OS/VS COBOL プログラム用の主要なコンパイラー・オプション

コンパイラー・オプション	説明
BUFSIZE	OS/VS COBOL では、BUF オプションの値は、バッファー用に予約されるバイトの合計数を指定します。Enterprise COBOL では、BUFSIZE は、それぞれのコンパイラー作業データ・セットごとに予約されるバッファー・ストレージの量を指定します。デフォルトは 4096 です。 OS/VS COBOL プログラムで BUF オプションを使用している場合は、Enterprise COBOL の BUFSIZE オプションで要求する量を調整しなければなりません。
DATA(24)	RENT を指定してコンパイルされ、OS/VS COBOL プログラムと一緒に使用されている Enterprise COBOL プログラムの場合は、DATA(24) を使用してください。
DIAGTRUNC	MOVE ステートメントについて数値切り捨てフラグを設定するには、DIAGTRUNC を使用してください。これは、OS/VS COBOL におけるフラグ設定の機能と同じでです。
NUMPROC(MIG)	NUMPROC(MIG) は、OS/VS COBOL と*類似*しているが、正確に同じではない方法で数値符号を処理します。
NUMCLS(ALT)	OS/VS COBOL と一緒に配布された USERMOD を使用していた場合は、NUMCLS(ALT) を使用してください。USERMOD の場合、文字 A、B、E (および C、D、F) が、COBOL の数値のクラス・テストで有効な数値記号と見なされます (さらに、NUMPROC(MIG) を指定してコンパイルしなければなりません)。記号表記に使用できるその他の文字については、*Enterprise COBOL プログラミング・ガイド* を参照してください。
OPT(STD)	WORKING-STORAGE 内に目印またはタイム / バージョン・スタンプとしての非参照データ項目がある場合は、OPT(STD) を使用してください。未使用データ項目が必要でない場合のみ、OPT(FULL) を使用してください。

表 34. 移行済み OS/VS COBOL プログラム用の主要なコンパイラー・オプション (続き)

コンパイラー・オプション	説明
OUTDD(ddname)	このオプションは、システム論理出力装置に送られる SYSOUT 出力についてのデフォルト DD 名 (SYSOUT) をオーバーライドするために使用します。DD 名が言語環境プログラムの MSGFILE DD 名と同じである場合、出力は MSGFILE 用に指定された DD 名に送られます。DD 名が言語環境プログラムの MSGFILE DD 名と同じでない場合は、DISPLAY ステートメントからの出力は OUTDD DD 名宛先に送られます。最初の参照時に DD 名が存在しない場合は、デフォルト名および言語環境プログラムによって指定された属性を使用して動的割り振りが行われます。
PGMNAME(COMPAT)	プログラム名が、OS/VS COBOL と互換性のある方法で処理されるようにするために、PGMNAME(COMPAT) を使用してください。
RMODE(24 または AUTO)	NORENT を指定してコンパイルされ、OS/VS COBOL プログラムと一緒に使用されている Enterprise COBOL プログラムの場合は、RMODE(24) または RMODE(AUTO) を使用してください。
TRUNC	TRUNC は、MOVE 時および算術演算時に算術フィールドがバイナリー受け取りフィールドに合わせて切り捨てられる方法を制御します。 インストール先で OS/VS COBOL でのデフォルトとして TRUNC を使用していた場合は、TRUNC(STD) を使用してください。 インストール先で OS/VS COBOL でのデフォルトとして NOTRUNC を使用していた場合は、TRUNC(OPT) を使用してください (バイナリー・データの切り捨てが起こらないことを保証する必要がある選択プログラムを除く)。バイナリー・データの切り捨てが起こらないことを必要とするプログラムの場合は、TRUNC(BIN) を使用してください (特に、バイナリー・データ項目に移動されるデータが、バイナリー・データ項目用の PICTURE 文節で定義された値より大きい値を持つ可能性がある場合)。 **注:** Enterprise COBOL の場合、TRUNC(OPT) を指定してコンパイルされたプログラムと、NOTRUNC を指定してコンパイルされた OS/VS COBOL プログラムとでは、結果が異なる場合があります。主として、プログラムが非ゼロ高位桁を失う可能性があります。高位桁の消失が起こる可能性のあるステートメントについて、Enterprise COBOL では、以下のことを確認する必要があることを示す診断メッセージを出します。 • 送り出し項目に大きな数値が含まれないこと。 • 受け取り項目が、PICTURE 文節で、最大の送り出しデータ項目を処理するのに十分な桁を指定して定義されていること。

サポートされない OS/VS COBOL コンパイラー・オプション

179 ページの表 35 に、Enterprise COBOL でサポートされない OS/VS COBOL コンパイラー・オプションを示します。

Enterprise COBOL コンパイラー・オプションの完全なリストについては、323 ページの『付録 F. コンパイラー・オプションの比較』を参照してください。

表 35. Enterprise COBOL でサポートされない OS/VS COBOL コンパイラー・オプション

OS/VS COBOL オプション	Enterprise COBOL で同等のオプション
BATCH/NOBATCH	バッチ環境は常に使用可能です (プログラムのシーケンス)。CBL ステートメントは常に Enterprise COBOL で処理されます。 **注:** Enterprise COBOL でのプログラムのシーケンスに関する考慮事項については、*Enterprise COBOL プログラミング・ガイド* を参照してください。
COUNT/NOCOUNT	デバッグ・ツールを使用して同等の機能を使用することができます。
ENDJOB/NOENDJOB	ENDJOB 動作は常に有効です。
LANGLVL(1/2)	LANGLVL オプションは使用不能です。Enterprise COBOL では、COBOL 85 標準のみサポートされます。
LVL=A|B|C|D/ NOLVL	FIPS のフラグ設定には FLAGSTD が使用されます。ANSI COBOL 74 FIPS はサポートされません。
RES/NORES	RES または NORES オプションは使用不能です。Enterprise COBOL では、オブジェクト・モジュールは、必ず、ライブラリー・サブルーチンが実行時に動的に位置指定されるように (COBOL プログラムとリンク・エディットされるのではなく) 作成されます。これは、OS/VS COBOL での RES 動作と同等です。
SUPMAP/NOSUPMAP	NOCOMPILE/COMPILE コンパイラー・オプションと同等です。
SYMDMP/ NOSYMDMP	異常終了ダンプと動的ダンプは、言語環境プログラム・サービスを介して入手することができます。シンボリック・ダンプは、TEST コンパイラー・オプションを介して入手することができます。
SXREF/NOSXREF	XREF オプションが、ソート済み SXREF 出力を提供します。
VBSUM/NOVBSUM	VBREF コンパイラー・オプションを介して同等の機能を使用することができます。
CDECK/NOCDECK	LISTER 機能はサポートされません。
FDECK/NOFDECK	LISTER 機能はサポートされません。
LCOL1/LCOL2	LISTER 機能はサポートされません。
LSTONLY/LSTCOMP NOLST	LISTER 機能はサポートされません。
L120/L132	LISTER 機能はサポートされません。
OSDECK	Enterprise COBOL では、オブジェクト・デックは z/OS 環境または OS/390 環境でのみ実行されます。OSDECK 機能はサポートされません。

Prolog 形式の変更点

オブジェクト・プログラムの Prolog は、コンパイラーがプログラムの入り口点で生成するコードです。Prolog には、プログラムを識別するデータも含まれています。

Enterprise COBOL によって生成されたオブジェクト・モジュールは言語環境プログラム準拠であるため、Prolog 形式が OS/VS COBOL の場合とは異なります。日付 / 時刻を走査する既存のアセンブラー・プログラムは、新しい形式に更新することが必要です。

移行済み OS/VS COBOL プログラムのコンパイル

Enterprise COBOL の LIST コンパイラー・オプションを指定してプログラムをコンパイルすることによって、OS/VS COBOL の Prolog 形式を Enterprise COBOL の Prolog 形式と比較するのに使用できるリストを生成することができます。

第 12 章 VS COBOL II ソース・プログラムのアップグレード

この章では、VS COBOL II 言語と Enterprise COBOL 言語間の違いについて説明します。この章の情報は、Enterprise COBOL でコンパイルするためにはソースの修正が必用となる VS COBOL II プログラムを判別する上でも役立ちます。たとえば、CMPR2 オプションを指定してコンパイルされた VS COBOL II プログラムは、Enterprise COBOL が CMPR2/NOCMPR2 コンパイラー・オプションをサポートしていないので、ソースの修正が必要です。

この章では、VS COBOL II ソース・プログラムを Enterprise COBOL へアップグレードする際に考慮が必用な以下の項目についての情報を掲載しています。

- Enterprise COBOL でコンパイルする前にアップグレードが必用なプログラムの判別
- CMPR2 コンパイラー・オプションでコンパイルされた VS COBOL II プログラムのアップグレード
- COBOL 85 標準の解釈の 3 つの小さな変更
- VS COBOL II リリース 3.0 からの ACCEPT ステートメントの変更
- 新しい予約語
- 文書化されていない VS COBOL II の拡張

Enterprise COBOL でコンパイル前にアップグレードが必要なプログラムの判別

VS COBOL II プログラムは、以下のいずれかの場合を除き、変更を加えずに Enterprise COBOL コンパイラーを使用してコンパイルできます。

- CMPR2 コンパイラー・オプションを指定してコンパイルされたプログラム
- VS COBOL II リリース 3.x でコンパイルされたプログラムで、COBOL 85 標準の解釈の変更対象となった比較的軽微な 3 つの COBOL 85 標準機能のうち 1 つまたは複数を使用しているもの
- VS COBOL II リリース 3.0 でコンパイルされたプログラムで、ACCEPT . . . FROM CONSOLE を使用しているもの
- Enterprise COBOL で予約語となっているワードを使用しているプログラム
- 文書化されていない VS COBOL II 拡張を使用しているプログラム

CMPR2 コンパイラー・オプションを指定してコンパイルされた VS COBOL II プログラムのアップグレード

VS COBOL IIソース・プログラムが CMPR2 コンパイラー・オプションを指定してコンパイルされている場合、Enterprise COBOL でコンパイルするには、そのソース・プログラムを NOCMPR2 プログラムへ変換する必要があります。CMPR2/NOCMPR2 コンパイラー・オプションは、Enterprise COBOL ではサポートされていません。ただし、Enterprise COBOL プログラムは、NOCMPR2 が有効であるかのように動作します。CMPR2 と NOCMPR2 (COBOL 85 標準) 間の言語の違

いについては、第 16 章の 199 ページの 『CMPR2 コンパイラー・オプションを指定してコンパイルされたプログラムのアップグレード』 を参照してください。

CMPR2 の NOCMPR2 への変換に役立つツールについては、 293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

COBOL 85 標準の解釈の変更

VS COBOL II リリース 3 (3.0、3.1、および 3.2 を含む) で NOCMPR2 を指定してコンパイルされたプログラムと、以降のリリース (VS COBOL II リリース 4、IBM COBOL、および Enterprise COBOL を含む) で NOCMPR2 を指定してコンパイルされたプログラムの間には、言語の違いがいくつかあります。これらの変更は、VS COBOL II リリース 3 で使用されていたものとは異なるインプリメンテーションを必要とした COBOL 標準解釈要求に応じた結果です。これらの軽微な違いは、その使用頻度から、お使いのプログラムでは影響がない場合がほとんどであると考えられます。ただし、以下の言語エレメントには影響があります。

- REPLACE およびコメント行
- ネストされたプログラムの場合の USE プロシージャーの優先順位
- 長さが指定されていない可変長グループ受け取り側の参照変更

REPLACE およびコメント行

この項目は、REPLACE ステートメントの pseudo-text-1 に一致するテキスト内に現れるブランク行およびコメント行の扱いに影響を与えます。

一致するテキスト内に散在するブランク行は、REPLACE ステートメントの出力に現れません。この変更によって行番号が変わることがあるため、生成されるプログラムのセマンティクスに影響がある可能性があります (たとえば、プログラムで USE FOR DEBUGGING 宣言を使用している場合、DEBUG-ITEM の内容が異なる可能性があります)。Enterprise COBOL によって生成されたプログラムが同等の VS COBOL II プログラムと異なる場合は、以下のメッセージが出されます。

IGYLI0193-I

一致する pseudo-text-1 にブランク行またはコメント行が入っています。実行結果が VS COBOL II リリース 3.x の場合とは異なる可能性があります。

USE プロシージャーの優先順位

この違いは、包含されるプログラムに関連する USE プロシージャーの優先順位に影響を与えます。

VS COBOL II リリース 3.x では、ファイル指定の USE プロシージャーがモード指定の USE プロシージャーより常に優先順位が高くなります。この優先順位は、適用できるモード指定の USE プロシージャーが現行プログラム内に存在する場合、または外部プログラム内の GLOBAL 属性を持つモード指定の USE プロシージャーがファイル指定のプロシージャーより「近い」場合にも当てはまります。

VS COBOL II リリース 4 および Enterprise COBOL では、USE プロシージャーの優先順位は、プログラムごとのレベル (現行プログラムから、それが包含しているプログラム、さらに最外部プログラムに向かって) に基づいて決まります。

Enterprise COBOL によって生成されたプログラムが VS COBOL II リリース 3.x プログラムで使用されていたものとは異なる USE プロシージャーを選択する場合は、以下のメッセージが出されます。

IGYSC2300-I
プログラム「program-name」内のファイル「file-name」について、モード指定の宣言が選択される可能性があります。実行結果が VS COBOL II リリース 3.x の場合と異なる可能性があります。

可変長グループ受け取り側の参照変更

参照変更された可変長グループにデータを MOVE するプログラムは、可変長グループを評価するのに使用された長さが実際の長さと最大長のどちらかであるかによって、異なる結果を生成する可能性があります。

可変長グループが以下のすべての基準を満たす場合は、結果が異なる可能性があります。

- 可変長グループが受け取り側である。
- 可変長グループが、それ自体の OCCURS DEPENDING ON オブジェクトを含んでいる。
- 可変長グループの後に非従属項目 (可変位置データ項目とも呼ばれる) がない。
- 可変長グループが参照変更され、長さが指定されていない。

たとえば、グループ *VAR-LEN-GROUP-A* は ODO オブジェクトと OCCURS サブジェクトを含んでおり、このグループの後に可変位置データ項目があります。

```
01 VAR-LEN-PARENT-A.
   02 VAR-LEN-GROUP-A.
      03 ODO-OBJECT PIC 99 VALUE 5.
      03 OCCURS-SUBJECT OCCURS 10 TIMES DEPENDING ON ODO-OBJECT.
         04 TAB-ELEM  PIC X(4).
   02 VAR-LOC-ITEM  PIC XX.
01 NEXT-GROUP.

MOVE ALL SPACES TO VAR-LEN-GROUP-A(1:).
```

グループ *VAR-LEN-GROUP-B* は ODO オブジェクトと OCCURS サブジェクトを含んでおり、このグループの後に可変位置データ項目がありません。
VAR-LOC-ITEM は、VAR-LEN-GROUP-B の後にあるのではなく、OCCURS サブジェクトの後にあります。

```
01 VAR-LEN-PARENT-B.
   02 VAR-LEN-GROUP-B.
      03 ODO-OBJECT PIC 99 VALUE 5.
      03 OCCURS-SUBJECT OCCURS 10 TIMES DEPENDING ON ODO-OBJECT.
         04 TAB-ELEM PIC X(4).
      03 VAR-LOC-ITEM PIC XX.
01 NEXT-GROUP.

MOVE ALL SPACES TO VAR-LEN-GROUP-B(1:).
```

上記の例では、`MOVE ALL SPACES TO VAR-LEN-GROUP-A (1:)` の実行結果は、どの NOCMPR2 プログラム (VS COBOL II リリース 3.x、VS COBOL II リリース 4、または Enterprise COBOL) の場合も同じになります。このケースでは、どの NOCMPR2 プログラムも実際の長さを使用します。

NOCMPR2 言語の変更

`MOVE ALL SPACES TO VAR-LEN-GROUP-B (1:)` の実行結果は、NOCMPR2 を指定してコンパイルされた以下のプログラムでは異なります。

- VS COBOL II リリース 3.x は、ODO オブジェクトの現行値によって定義された、グループの実際の長さを使用します (グループの実際の長さが、ODO オブジェクト値を使用してスペースに設定されました)。
- VS COBOL II リリース 4 および Enterprise COBOL は、グループの最大長を使用します (データ項目全体が、ODO オブジェクト値を使用してスペースに設定されました)。

プログラムに、参照変更された可変長グループの受け取り側が含まれており、グループがそれ自体の ODO オブジェクトを持ち、グループの後に可変位置データがなく、参照変更子で長さが指定されていない場合は、以下のメッセージが出されます。

IGYPS2298-I

可変長グループ ″data name″ への参照は、グループの最大長を使用して評価されます。実行結果が VS COBOL II リリース 3.x の場合とは異なる可能性があります。

ACCEPT ステートメント

VS COBOL II リリース 3.0 と以降のリリースの間には、もう 1 つの違いがありますが、これは、ACCEPT ステートメントの簡略名サブオプションについてのシステム入力装置に関連します。

VS COBOL II リリース 3.0 の場合のみ、80 文字以外の論理レコード長が指定されても、80 文字の入力レコードが想定されます。VS COBOL II リリース 3.1 からリリース 4.0 の場合、80 文字以外の論理レコード長が指定された場合、256 文字の入力レコードが想定されます。

Enterprise COBOL では、許容される最大論理レコード長は 32,760 文字です。

新しい予約語

COBOL (MVS および VM 版) および COBOL (OS/390 および VM 版) では、オブジェクト指向拡張用に追加された新しいワードを予約していない代替予約語テーブルが用意されています。該当するいずれかのワード (FUNCTION および PROCEDURE-POINTER を除く) を引き続き使用するためには、WORD(NOOO) コンパイラー・オプションを指定してください。

既存のプログラムで FUNCTION および PROCEDURE-POINTER を使用している場合は、それらを除去する必要があります。

Enterprise COBOL と IBM COBOL を合わせると、VS COBOL II と比較してかなり多くの予約語が追加されています。既存の VS COBOL II プログラムでこれらの予約語をユーザー定義語として使用している場合、Enterprise COBOL でコンパイルする前にそのプログラムを変更する必要があります。

CCCA を使用すると、予約語を自動的に変換することができます。CCCA ツールについての詳細は、 293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

図 35 は COBOL 以降の各リリースで追加された予約語を示しています。予約語の完全なリストについては、275 ページの 『付録 B. COBOL 予約語の比較』を参照してください。

表 36. 新しい予約語、コンパイラーの、VS COBOL II との比較

コンパイラー	予約語
COBOL/370 V1R1	FUNCTION、PROCEDURE-POINTER
COBOL (MVS および VM 版) V1R2	CLASS-ID、 METACLASS、 RECURSIVE、 END-INVOKE、 METHOD、 REPOSITORY、 INHERITS、 METHOD-ID、 RETURNING、 INVOKE、 OBJECT、 SELF、 SUPER、 LOCAL-STORAGE、 OVERRIDE
COBOL (OS/390 および VM 版) V2R1	COBOL (MVS および VM 版) と同様
COBOL (OS/390 および VM 版) V2R2	COMP-5、 COMPUTATIONAL-5、 EXEC、 END-EXEC、 SQL、 TYPE、 FACTORY
COBOL (OS/390 および VM 版) V2R2 (PQ49375 適用済み)	EXECUTE
Enterprise COBOL	JNIENVPTR、 NATIONAL、 XML、 END-XML、 XML-EVENT、 XML-CODE、 XML-TEXT、 XML-NTEXT、 FUNCTION-POINTER

文書化されていない VS COBOL II 拡張

VS COBOL II コンパイラーは、区域 A 内で、無効な区域 A 項目 (または項目なし) の後のピリオドを診断していませんでした。Enterprise COBOL では、区域 A 内のピリオドの前には、有効な区域 A 項目が必要です。

VS COBOL II コンパイラーは、SORT の INPUT/OUTPUT PRODECDURE 内や MERGE の OUTPUT PROCEDURE 内に STOP RUN、GOBACK、または EXIT PROGRAM が記述されていてもそれを検出しませんでした。Enterprise COBOL ではこれを診断し、メッセージ IGYPA3036 を出します。

第 13 章 VS COBOL II プログラムのコンパイル

この章では、Enterprise COBOL コンパイラーと VS COBOL II コンパイラー間の違いを説明します。以下のトピックに関する情報が記載されています。

- VS COBOL II プログラム用の主要なコンパイラー・オプション
- Prolog 形式の変更点

VS COBOL II プログラム用の主要なコンパイラー・オプション

Enterprise COBOL コンパイラーと VS COBOL II コンパイラーは類似しています。現行の VS COBOL II アプリケーションで指定しているのと同じコンパイラー・オプションを使用する場合、いくつかの内部的な変更によって影響を受ける可能性はありますが、基本的には動作は変わりません。

VS COBOL II で使用していたコンパイラー・オプションの設定を変更する場合は、アプリケーションに与える可能性のある影響について十分確認してください。ソース・プログラムを CMPR2 から NOCMPR2 に変換する方法については、199 ページの 『CMPR2 コンパイラー・オプションを指定してコンパイルされたプログラムのアップグレード』を参照してください。その他のコンパイラー・オプションについては、*Enterprise COBOL プログラミング・ガイド* を参照してください。

Enterprise COBOL でのコンパイル

表 37 に、移行済みプログラムに特に関係がある Enterprise COBOL コンパイラー・オプションをリストします。

表 37. VS COBOL II プログラム用の主要な Enterprise COBOL コンパイラー・オプション

Enterprise COBOL コンパイラー・オプション	説明
PGMNAME	Enterprise COBOL でコンパイルする場合、プログラム名が VS COBOL II (および COBOL/370) と互換性のある方法で処理されるようにするには、PGMNAME(COMPAT) オプションを使用してください。
RMODE	AMODE 24 で実行中のプログラムにデータを渡す Enterprise COBOLNORENT プログラムの場合、RMODE(AUTO) または RMODE(24) を使用してください。
TEST	TEST オプションの構文は、Enterprise COBOL では VS COBOL II の場合と異なります。TEST オプションには、現在 3 つのサブオプションがあります。TEST を指定する代わりにフック位置、記号テーブル位置、シンボリック・デバッグ情報用の宛先を指定できます。 サブオプションを指定しない TEST は、TEST(ALL,SYM,NOSEPARATE) になります。TEST オプションについては、*Enterprise COBOL プログラミング・ガイド* を参照してください。

Enterprise COBOL でサポートされないコンパイラー・オプション

表 38 に、Enterprise COBOL でサポートされない VS COBOL II コンパイラー・オプションの一覧を示します。場合によっては、説明欄で記述されているように、VS COBOL II コンパイラー・オプションの機能が Enterprise COBOL コンパイラー・オプションにマップされています。

表 38. Enterprise COBOL でサポートされないコンパイラー・オプション

VS COBOL II コンパイラー・オプション	説明
CMPR2	CMPR2 オプションはサポートされません。CMPR2 でコンパイルされたプログラムを Enterprise COBOL でコンパイルするために、COBOL 85 標準に変換する必要があります。
FDUMP/NOFDUMP	Enterprise COBOL は FDUMP コンパイラー・オプションを提供しません。既存のアプリケーションのために、FDUMP は Enterprise COBOLの TEST(SYM) コンパイラー・オプションにマップされています。TEST(SYM) は、同等以上の機能を提供することができます。 言語環境プログラムは、FDUMP オプションに関係なく、VS COBOL II よりも優れた定様式ダンプを生成します。しかし、TEST(SYM) を指定すると、言語環境プログラムはデータ項目について情報のシンボリック・ダンプを定様式ダンプ内に加えることができます。 異常終了時に言語環境プログラムの定様式ダンプを取得する方法については、*言語環境プログラム デバッグのガイドおよびランタイム・メッセージ* を参照してください。 NOFDUMP が検出されると、NOFDUMP はサポートされないので、Enterprise COBOL は警告メッセージを出します。
FLAGMIG	FLAGMIG オプションは、Enterprise COBOL ではサポートされていません。FLAGMIG オプションを使用するには CMPR2 が必要です。これは Enterprise COBOL ではサポートされません。マイグレーションに関する同様の識別情報を取得するには、CCCA、本書 (*移行ガイド*) を利用するか、または Enterprise COBOL 以前のリリースのコンパイラーで FLAGMIG を使用するプログラムをコンパイルしてください。
FLAGSAA	Enterprise COBOL は FLAGSAA オプションをサポートしません。FLAGSAA が指定されていると、Enterprise COBOL は警告メッセージを出します。
RES/NORES	Enterprise COBOL は RES/NORES コンパイラー・オプションを提供しません。RES が検出されると、Enterprise COBOL は情報メッセージを出します。NORES が検出されると、Enterprise COBOL は警告メッセージを出します。

Prolog 形式の変更点

オブジェクト・プログラムの Prolog は、コンパイラーがプログラムの入り口点で生成するコードです。Prolog には、プログラムを識別するデータも含まれています。

Enterprise COBOL によって生成されたオブジェクト・モジュールは言語環境プログラム準拠であるため、Prolog 形式が VS COBOL II の場合とは異なります。日付 / 時刻とユーザー・レベル情報を走査する既存のアプリケーションは、新規の書式に対応して更新する必要があります。

Enterprise COBOL の LIST コンパイラー・オプションを指定してプログラムをコンパイルすることによって、VS COBOL II の Prolog 形式を Enterprise COBOL の Prolog 形式と比較するのに使用できるリストを生成することができます。

第 14 章 IBM COBOL ソース・プログラムのアップグレード

この章では、IBM COBOL 言語と Enterprise COBOL 言語間の違いについて説明します。この章に記載されている情報は、Enterprise COBOL でコンパイルを実行するためにどの IBM COBOL プログラムのソース変更が必要かを判別するうえで役立ちます。たとえば、CMPR2 オプションを指定してコンパイルした IBM COBOL プログラムにはソース変更が必要です。これは、Enterprise COBOL では CMPR2/NOCMPR2 コンパイラー・オプションがサポートされないためです。

この章には、IBM COBOL ソース・プログラムを Enterprise COBOL にアップグレードする際に考慮しなければならない以下の項目に関する情報が記載されています。

- Enterprise COBOL を使用してコンパイルする前にアップグレードが必要なプログラムの判別
- SOM ベースのオブジェクト指向 COBOL プログラムのアップグレード
- Enterprise COBOL の新しい予約語
- 文書化されていない IBM COBOL 拡張

CMPR2 コンパイラー・オプションを指定してコンパイルしたプログラムのアップグレードに関する情報は、199 ページの 『第 16 章 CMPR2 から NOCMPR2 へのマイグレーション』を参照してください。

単独の CICS (顧客情報管理システム) トランスレーターから組み込みの CICS トランスレーターへのマイグレーションについては、239 ページの 『単独の CICS トランスレーターから組み込みのトランスレーターへのマイグレーション』を参照してください。

Enterprise COBOL を使用してコンパイルする前にアップグレードが必要なプログラムの判別

IBM COBOL プログラムは、以下のいずれか (または複数) が含まれない限り、Enterprise COBOL コンパイラーを使用して変更なしにコンパイルされます。

- CMPR2 コンパイラー・オプションを指定してコンパイルされたプログラム
- SOM ベースのオブジェクト指向 COBOL 構文を持つプログラム
- 現在 Enterprise COBOL に予約済みのワードを使用するプログラム
- 文書化されていない IBM COBOL 拡張機能を持つプログラム

SOM ベースのオブジェクト指向 (OO) COBOL プログラムのアップグレード

SOM ベースのオブジェクト指向 COBOL アプリケーションは、Enterprise COBOL ではサポートされなくなりました。OO COBOL 構文は、COBOL と Java の相互協調処理を容易にするために再び Java ベースのオブジェクト指向プログラミング (OOP) を目標としています。

新しい Java ベースの OO COBOL は SOM ベースの OO COBOL との互換性がなく、OO COBOL プログラムへのマイグレーション・パスとして使用するようには意図されていません。たいていの場合、Enterprise COBOL コンパイラーを使用するためには OO COBOL をプロシージャー型 COBOL に書き直す必要があります。既存の SOM ベースの OO 構文の代わりに新しい OO COBOL 構文を使用することができますが、この変換は容易ではありません。

SOM ベースの OO COBOL ステートメントを含む IBM COBOL プログラムを Enterprise COBOL にアップグレードする際に適用される考慮事項については、192 ページの 『サポートされない SOM ベースのオブジェクト指向 COBOL 言語エレメント』および 193 ページの 『変更された SOM ベースのオブジェクト指向 COBOL 言語エレメント』を参照してください。

サポートされない SOM ベースのオブジェクト指向 COBOL 言語エレメント

ここでは、Enterprise COBOL でサポートされていない SOM ベースの OO COBOL 言語エレメントについて説明します。以下の考慮事項は、SOM ベースの OO COBOL を使用するアプリケーションを、Enterprise COBOL でサポートされる Java ベースの OO COBOL 構文にマイグレーションする場合に適用されます。

SOM への呼び出し

SOM サービスへの呼び出しはサポートされません。

INHERITS 文節

- CLASS-ID 段落の INHERITS 文節に複数のクラス名を指定する (多重継承) ことはサポートされていません。
- COBOL クラスは最後に java.lang.Object クラス (SOMObject または SOMClass ではなく) から派生する必要があります。INHERITS 文節に基底クラスとして SOMObject を指定することはサポートされていません。
- INHERITS 文節に基底クラスとして SOMClass を指定する (メタクラスを定義する) ことはサポートされていません。Java ベースの OO COBOL クラスは、FACTORY セクションを指定して、論理的にクラスのクラス・オブジェクトの一部である静的メソッドを定義することができます。

INVOKE

- INVOKE ステートメントの引き数リストおよびメソッドのパラメーター・リスト (PLIST) は、Java 型にマップするデータ型に制限され、BY VALUE によって渡されます。
- INVOKE ステートメントに SUPER を修飾するクラス名を指定することはサポートされていません。たとえば、以下の構文を使用することはできません。

  ```
  INVOKE C OF SUPER "foo"
  ```

  ただし、以下の構文は Enterprise COBOL でも引き続きサポートされます。

  ```
  INVOKE SUPER "foo"
  ```

METACLASS 文節

- CLASS-ID 段落の METACLASS IS 文節はサポートされません。
- オブジェクト・リファレンスを定義する USAGE 文節の METACLASS OF 文節はサポートされません。

METHODS

- METHOD-ID 段落の OVERRIDE 文節はサポートされません。
- SOM 基底クラスのメソッド (somNew、somFree、somInit など) の使用はサポートされていません。

コンパイラー・オプション IDLGEN および TYPECHK

IDLGEN および TYPECHK オプションは利用できません。これらのコンパイラー・オプションにはいずれも SOM ベースの OO COBOL が必要ですが、これは Enterprise COBOL では利用できません。

変更された SOM ベースのオブジェクト指向 COBOL 言語エレメント

ここでは、Enterprise COBOL で変更された SOM ベースの OO COBOL 言語エレメントについて説明します。以下の考慮事項は、SOM ベースの OO COBOL を使用するアプリケーションを、Enterprise COBOL でサポートされる Java ベースの OO COBOL 構文にマイグレーションする場合に適用されます。

外部名

- REPOSITORY 段落で定義される外部クラス名は、クラス名の CORBA (Common Object Request Broker Architecture) 形成規則ではなく、完全修飾クラス名を対象とした Java 命名規則に従って定義する必要があります。
- リテラルとして指定されるメソッド名は、CORBA 命名規則ではなく Java 命名規則を使用します。

INVOKE

somNew の代わりに、次の構文を使用してオブジェクト・インスタンスが作成されます。

```
INVOKE classname NEW ...
```

METHODS

COBOL メソッドは継承されたメソッドをオーバーライドすることができ、Java 規則に従って多重定義できます。ただし、このような場合に OVERRIDE 文節は METHOD-ID 段落で必須ではなく、サポートもされません。

OBJECTS

- somNew の代わりに、次の構文を使用してオブジェクト・インスタンスが作成されます。

  ```
  INVOKE classname NEW ...
  ```

- オブジェクト・インスタンスは somFree ではなく Java 自動ガーベッジ・コレクションによって解放されます。
- オブジェクト・インスタンス・データは、somInit ではなく VALUE 文節またはユーザー作成の初期化メソッドによって初期化されます。

- OBJECT 構文と END OBJECT 構文は、クラスがオブジェクト・インスタンス・データまたはオブジェクト・インスタンス・メソッドを指定しない場合は指定する必要があります。

Enterprise COBOL の新しい予約語

予約語の完全なリストについては、275 ページの 『付録 B. COBOL 予約語の比較』を参照してください。

表 39. VS COBOL II と比較したコンパイラーごとの新しい予約語

コンパイラー	予約語
COBOL/370 V1R1	FUNCTION、PROCEDURE-POINTER
COBOL (MVS および VM 版) V1R2	CLASS-ID、 METACLASS、 RECURSIVE、 END-INVOKE、 METHOD、 REPOSITORY、 INHERITS、 METHOD-ID、 RETURNING、 INVOKE、 OBJECT、 SELF、 SUPER、 LOCAL-STORAGE、 OVERRIDE
COBOL (OS/390 および VM 版) V2R1	COBOL (MVS および VM 版) と同じ
COBOL (OS/390 および VM 版) V2R2	COMP-5、 COMPUTATIONAL-5、 EXEC、 END-EXEC、 SQL、 TYPE、 FACTORY
COBOL (OS/390 および VM 版) V2R2 (PQ49375 適用済み)	EXECUTE
Enterprise COBOL	JNIENVPTR、 NATIONAL、 XML、 END-XML、 XML-EVENT、 XML-CODE、 XML-TEXT、 XML-NTEXT、 FUNCTION-POINTER

文書化されていない IBM COBOL 拡張

IBM COBOL コンパイラーは、区域 A で、無効な区域 A 項目 (または項目なし) の後にピリオドがあるかどうか診断しませんでした。Enterprise COBOL では、区域 A におけるピリオドは有効な区域 A 項目の後になければなりません。

IBM COBOL コンパイラーは、SORT 入力または出力プロシージャー、または MERGE 出力プロシージャーで STOP RUN、GOBACK、OR EXIT PROGRAM を検出しませんでした。Enterprise COBOL ではこれらをメッセージ IGYPA3036 で診断できるようになりました。

第 15 章 IBM COBOL プログラムのコンパイル

この章では、Enterprise COBOL コンパイラーと IBM COBOL コンパイラー間の違いを説明します。以下のトピックに関する情報が記載されています。

- IBM COBOL プログラム用の主要なコンパイラー・オプション
- Enterprise COBOL で使用不能なコンパイラー・オプション

IBM COBOL プログラム用の主要なコンパイラー・オプション

Enterprise COBOL コンパイラーと IBM COBOL コンパイラーは非常に類似しています。現行の IBM COBOL アプリケーションで指定しているのと同じコンパイラー・オプションを使用する場合は、いくつかの内部的な変更が効力を生じる可能性はありますが、基本的には動作は変わりません。

IBM COBOL アプリケーション内のコンパイラー・オプションの設定を変更する場合は、アプリケーションに与える可能性のある影響を理解しておかなければなりません。既存のソース・プログラムを CMPR2 から NOCMPR2 へ変換する方法については、199 ページの 『第 16 章 CMPR2 から NOCMPR2 へのマイグレーション』を参照してください。その他のコンパイラー・オプションについては、*Enterprise COBOL プログラミング・ガイド* を参照してください。

Enterprise COBOL のコンパイラー・オプションは、IBM COBOL のコンパイラー・オプションと少し異なります。表 40 は IBM COBOL と Enterprise COBOL の間の互換性に影響を与えるオプションの一覧です。

表 40. IBM COBOL プログラム用の主要なコンパイラー・オプション

コンパイラー・オプション	説明
ARITH	算術ステートメントの中間結果について、COBOL/370 リリース 1 から COBOL (OS/390 および VM 版) バージョン 2 リリース 1 の場合と同じ結果を得るには、ARITH(COMPAT) を使用してください。

表 40. IBM COBOL プログラム用の主要なコンパイラー・オプション (続き)

コンパイラー・オプション	説明
INTDATE	日付組み込み関数について COBOL/370 リリース 1 の場合と同じ結果を得るには、INTDATE(ANSI) を使用してください。整数値を保管し、同じデータに対して他の言語を使用する場合は、INTDATE(LILIAN) を使用してください。INTDATE(LILIAN) を指定すると、日付組み込み関数は言語環境プログラムの開始日付を使用します。この開始日付は言語環境プログラムの日付呼び出し可能サービスを使用する PL/I または C プログラムによって使用される開始日付と同じです。 整数日付を 1 つのプログラム内でのみ使用する (たとえば、グレゴリオをリリアン日に変換し、同一プログラム内でグレゴリオに変換し戻す) 場合は、INTDATE を設定することは重要ではありません。 インストール時のデフォルトとして INTDATE(LILIAN) を選択する場合、すべてのコードが確実にリリアン整数日付標準を用いるようにするには、組み込み関数を使用する COBOL/370 リリース 1 のプログラム (また、もしあれば INTDATE(ANSI) を使用した IBM COBOL プログラム) をすべて再コンパイルする必要があります。この方法は、整数日付を保管し、その日付をプログラム間で (PL/I、COBOL、C の他言語プログラム間でも) 渡すことができ、日付処理の整合性も保たれるので最も安全です。
PGMNAME	プログラム名が COBOL/370 リリース 1 と同様の方法で処理されるようにするには、PGMNAME(COMPAT) を指定してください。
NSYMBOL	国別処理または DBCS 処理 が想定されるかどうかを指定し、リテラルと PICTURE 文節で使用する ″N″ 記号の解釈を制御します。 NSYMBOL(NATIONAL) は COBOL の前のリリースとの互換性を提供します。

表 40. IBM COBOL プログラム用の主要なコンパイラー・オプション (続き)

コンパイラー・オプション	説明
TRUNC	COBOL (OS/390 および VM 版) のバージョン 2 リリース 2 より前のリリースでは、TRUNC(BIN) を指定された符号なしバイナリー・データ項目は、バイナリー値が多くても 15 ビット (ハーフワードの場合)、31 ビット (フルワードの場合)、または 63 ビット (ダブルワードの場合) を含んでいるときにのみ正しくサポートされました。つまり、データ項目が符号なしのときは、符号ビットは数値の一部として使用されませんでした。Enterprise COBOL および COBOL (OS/390 および VM 版) バージョン 2 リリース 2 では、TRUNC(BIN) を指定すると、符号なし COMP-5 データ項目または符号なしバイナリー・データ項目の数値として、ハーフワードの 16 ビットすべて、フルワードの 32 ビットすべて、ダブルワードの 64 ビットすべてが使用されます。 たとえば、TRUNC(BIN) を指定してコンパイルされたプログラムで、次のように宣言されているデータ項目 `01 X pic 9(2) binary.` は、以前のリリースでは 0 ～ 32767 のバイナリー値のみを正しくサポートしましたが、バージョン 2 リリース 2 では 0 ～ 65535 の値をサポートします。 このサポートにより、非常に大きい符号なしバイナリー値が使用された場合は、以前のリリースで得られた算術結果とは異なる算術結果になります。

Enterprise COBOL で使用不能なコンパイラー・オプション

IBM COBOL で使用可能なコンパイラー・オプションの大部分は、以下のものを除いて、Enterprise COBOL のコンパイルでも使用できます。

表 41. Enterprise COBOL で使用不能なコンパイラー・オプション

コンパイラー・オプション	説明
ANALYZE	ANALYZE オプションは Enterprise COBOL では使用不能です。代わりに CICS、SQL、および ADATA オプションを使用してください。
CMPR2	CMPR2 オプションは使用不能です。CMPR2 でコンパイルされたプログラムを Enterprise COBOL でコンパイルし、COBOL 85 標準に変換してください。
EVENTS	EVENTS オプションは使用不能です。COBOL/370 の EVENTS コンパイラー・オプションをエミュレートするには、次のようにします。 1. ADATA コンパイラー・オプションを指定する。 2. SYSADATA および SYSEVENTS を割り振る。 3. EXIT コンパイラー・オプションの ADEXIT サブオプションを、サンプル出口プログラム IGYADXIT と共に使用する。

表 41. Enterprise COBOL で使用不能なコンパイラー・オプション (続き)

コンパイラー・オプション	説明
FLAGMIG	FLAGMIG オプションは使用不能です。FLAGMIG には CMPR2 が必要ですが、これは Enterprise COBOL では使用不能です。FLAGMIG を使用するプログラムをコンパイルするには、CCCA、本書 (*移行ガイド*)、または Enterprise COBOL 以前のリリースのコンパイラーを使用してください。
IDLGEN	IDLGEN オプションは使用不能です。IDLGEN には SOM ベースの OO COBOL が必要ですが、これは Enterprise COBOL では使用不能です。
TYPECHK	TYPECHK オプションは使用不能です。TYPECHK オプションを使用するには SOM ベースの OO COBOL が必要ですが、これは Enterprise COBOL では使用不能です。
WORD(NOOO)	WORD(NOOO) コンパイラー・オプションを使用してコンパイルされた既存の IBM COBOL プログラムについては、以下の予約語を使用している場合、変更が必要です。 CLASS-ID、 END-INVOKE、 INHERITS、 INVOKE、 LOCAL-STORAGE、 METACLASS、 METHOD、 METHOD-ID、 OBJECT、 OVERRIDE、 RECURSIVE、 REPOSITORY、 RETURNING、 SELF、 SUPER。 WORD オプションの NOOO サブオプションは、Enterprise COBOL ではサポートされていません。

第 16 章 CMPR2 から NOCMPR2 へのマイグレーション

この章では、CMPR2 と NOCMPR2 間のコンパイラー・オプションの違いについて説明します。COBOL ソース・プログラムが CMPR2 コンパイラー・オプションを指定してコンパイルされている場合、Enterprise COBOL でコンパイルするには、そのプログラムを NOCMPR2 プログラムへ変換する必要があります。CMPR2/NOCMPR2 コンパイラー・オプションは、Enterprise COBOL ではサポートされていません。ただし、Enterprise COBOL プログラムは、NOCMPR2 が常に有効であるかのように動作します。

この章には、CMPR2 オプションを指定してコンパイルされた COBOL ソース・プログラムを調べる際に、考慮が必要な以下の項目についての情報が掲載されています。

- CMPR2 コンパイラー・オプションを指定してコンパイルされたプログラムのアップグレード

CMPR2 コンパイラー・オプションを指定してコンパイルされたプログラムのアップグレード

Enterprise COBOL は、COBOL 85 標準をサポートしています。また、VS COBOL II リリース 2 は、(COBOL 85 標準の一部の機能を組み込みつつ) COBOL 74 標準をサポートしていました。COBOL 85 標準のインプリメンテーションにより、いくつかの言語エレメントが、COBOL 74 標準のインプリメンテーションとは異なる方法で動作します。

VS COBOL II リリース 3.0 から、NOCMPR2 コンパイラー・オプションを使用して COBOL 85 標準の動作 (組み込み関数モジュールなし) を選択するか、または CMPR2 コンパイラー・オプションを使用して COBOL 74 標準の動作を選択することができるようになりました。CMPR2 オプションを使用すると、VS COBOL II リリース 2 にインプリメントされた COBOL 74 標準の動作と、現在 COBOL 85 標準でインプリメントされている*非標準の* リリース 2 拡張をサポートします。NOCMPR2 オプションを使用すると、COBOL 85 標準準拠の動作および IBM 拡張を提供します。これと同じメカニズムは、VS COBOL II リリース 2 レベルのコードを COBOL 85 標準レベルのコードへアップグレードする猶予期間を設けるための補助として、IBM COBOL でも提供されていました。Enterprise COBOL では、このような措置をサポートしていません。プログラムは COBOL 85 標準レベルでないと、Enterprise COBOL でコンパイルできません。

VS COBOL II リリース 3 または以降のリリースと IBM COBOL を参照するときは、以下の用語が定義されています。

CMPR2

CMPR2 という用語は、以下のものでコンパイルされ、実行されるプログラムの言語および動作を参照するときに使用します。

- VS COBOL II リリース 2

- CMPR2 コンパイラー・オプションを使用する VS COBOL II リリース 3 または 4
- CMPR2 コンパイラー・オプションを使用する IBM COBOL

NOCMPR2

NOCMPR2 という用語は、以下のものでコンパイルされ、実行されるプログラムの言語および動作を参照するときに使用します。

- NOCMPR2 コンパイラー・オプションを使用する VS COBOL II リリース 3 または 4
- NOCMPR2 コンパイラー・オプションを使用する IBM COBOL
- Enterprise COBOL

FLAGMIG

FLAGMIG は、CMPR2 オプションおよび FLAGMIG オプションをサポートする Enterprise COBOL 以前のコンパイラー (VS COBOL II または IBM COBOL) の使用に言及する際に使用します。

注: Enterprise COBOL へのマイグレーションの補助として、FLAGMIG および CMPR2 をサポートする、以前の COBOL コンパイラーを使用して、変換が必要なステートメントを識別することができます。

以下にリストする言語エレメントは、CMPR2/NOCMPR2 コンパイラー・オプションの影響を受けます。違いについては、以下で説明します。

表 42. CMPR2 と NOCMPR2 との間で異なる言語エレメント

言語エレメント	ページ
SPECIAL-NAMES 段落の ALPHABET 文節	201 ページの『SPECIAL-NAMES 段落の ALPHABET 文節』
ALPHABETIC クラス	202 ページの『ALPHABETIC クラス』
CALL ... ON OVERFLOW	202 ページの『CALL . . . ON OVERFLOW』
位取り整数と非数字との比較	204 ページの『位取り整数と非数字との比較』
非 COBOL 文字を使用する COPY...REPLACING ステートメント	205 ページの『非 COBOL 文字を使用する COPY...REPLACING ステートメント』
国別拡張文字を使用する COPY ステートメント	207 ページの『国別拡張文字を使用する COPY ステートメント』
ファイル状況コード	208 ページの『ファイル状況コード』
暗黙の EXIT PROGRAM	209 ページの『暗黙の EXIT PROGRAM』
PERFORM から戻る方法	211 ページの『PERFORM から戻る方法』
PERFORM...VARYING...AFTER	213 ページの『PERFORM ... VARYING ... AFTER』
"A" および "B" を指定した PICTURE 文節	216 ページの『"A" および "B" を指定した PICTURE 文節』

表 42. CMPR2 と NOCMPR2 との間で異なる言語エレメント (続き)

言語エレメント	ページ
PROGRAM COLLATING SEQUENCE	218 ページの『PROGRAM COLLATING SEQUENCE』
READ INTO および RETURN INTO	219 ページの『READ INTO および RETURN INTO』
RECORD CONTAINS n CHARACTERS	221 ページの『RECORD CONTAINS n CHARACTERS』
予約語	222 ページの『予約語』
SET...TO TRUE	223 ページの『SET . . . TO TRUE』
MULTIPLY と DIVIDE の SIZE ERROR	225 ページの『MULTIPLY と DIVIDE の SIZE ERROR』
UNSTRING オペランドの評価	226 ページの『UNSTRING オペランドの評価』
UPSI スイッチ	233 ページの『UPSI スイッチ』
可変長グループ移動	234 ページの『可変長グループ移動』

SPECIAL-NAMES 段落の ALPHABET 文節

CMPR2

ALPHABET 文節でキーワード ALPHABET を使用しません。つまり、ALPHABET は予約語ではありません。

以下に例を示します。

```
SPECIAL-NAMES.
   ALPHA-NAME IS STANDARD-1.
```

NOCMPR2

ALPHABET 文節でキーワード ALPHABET を使用する必要があります。現在、ALPHABET は予約済みキーワードです。

以下に例を示します。

```
SPECIAL-NAMES.
   ALPHABET ALPHA-NAME IS STANDARD-1.
```

メッセージ

CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、SPECIAL-NAMES 段落の各 ALPHABET 文節ごとに以下のメッセージが生成されます。

IGYDS1190-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、英字名の前にキーワード "ALPHABET" がなければなりません。

SPECIAL-NAMES 段落の ALPHABET 文節に対する修正処置:

キーワード ALPHABET を ALPHABET 文節に追加してください。

ALPHABETIC クラス

CMPR2

ALPHABETIC クラス・テストで定義する ALPHABETIC クラスの文字は、26 個の英大文字とスペース文字で構成されます。26 個の英小文字は英字とは見なされません。

以下に例を示します。

```
MOVE "AbC dE" TO PIC-X6.
IF PIC-X6 IS NOT ALPHABETIC THEN DISPLAY "CMPR2".
```

NOCMPR2

ALPHABETIC クラス・テストで定義する ALPHABETIC クラスの文字は、26 個の英大文字、26 個の英小文字、およびスペース文字で構成されます。

以下に例を示します。

```
MOVE "AbC dE" TO PIC-X6.
IF PIC-X6 IS ALPHABETIC THEN DISPLAY "NOCMPR2".
```

メッセージ

CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、各 ALPHABETIC クラス・テストごとに以下のメッセージが生成されます。

IGYPS2221-W

MIGR 英字クラスは、"NOCMPR2" コンパイラー・オプションのもとでは小文字も含まれるように拡張されました。

ALPHABETIC クラスに対する修正処置:

NOCMPR2 のもとでは、ALPHABETIC-UPPER クラス・テストを使用して、CMPR2 のもとでの ALPHABETIC クラス・テストと同じ機能を取得してください。
NOCMPR2 のもとでの ALPHABETIC-UPPER クラスは、26 個の大文字とスペース文字で構成されます。

CALL . . . ON OVERFLOW

CMPR2

CMPR2 のもとでは、オブジェクト時メモリーの使用可能部分に、CALL ステートメントで指定されたプログラムを収容できない場合、ON OVERFLOW 条件になります。CMPR2 では、その定義を、「プログラムのロードに必要なストレージが使用可能でない」という条件だけを対象とするように解釈していました。

注: 呼び出し先プログラムの実際の LOAD 時にエラーが発生した場合にのみ、ON OVERFLOW 条件になります。プログラムがロードされ、実行が開始された後でエラーが発生しても、この条件にはなりません。

NOCMPR2

NOCMPR2 のもとでは、CALL ステートメントで指定されたプログラムをその時点で実行のために使用可能にできない場合、ON OVERFLOW 条件になります。

NOCMPR2 は COBOL 85 標準の規則をインプリメントしており、ON OVERFLOW 条件については、呼び出し先プログラムが使用可能になるのを妨げる可能性のある「回復可能」条件を処理するために定義されています。

注: 呼び出し先プログラムの実際の LOAD 時にエラーが発生した場合にのみ、ON OVERFLOW 条件になります。プログラムがロードされ、実行が開始された後でエラーが発生しても、この条件にはなりません。

メッセージ

CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、ON OVERFLOW 句を指定するすべての CALL ステートメントに対してメッセージが出されます。以下のメッセージが出されます。

IGYPS2012-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、"CALL" ステートメントの "ON OVERFLOW" 句は、オーバーフロー条件以外の場合も実行されることがあります。

この変更の影響を受ける事態の例を示すプログラム部分を以下に示します。

```
PERFORM UNTIL ALL-ACCOUNTS-SETTLED
   ⋮
  CALL "SUBPROGA" USING CURRENT-ACCOUNT
    ON OVERFLOW
       CANCEL "SUBPROGB"
       CALL "SUBPROGA" USING CURRENT-ACCOUNT
       END-CALL
  END-CALL
   ⋮
  CALL "SUBPROGB" USING CURRENT-ACCOUNT
    ON OVERFLOW
       CANCEL "SUBPROGA"
       CALL "SUBPROGB" USING CURRENT-ACCOUNT
       END-CALL
  END-CALL
   ⋮
END-PERFORM
```

上記のプログラムを実行すると、SUBPROGA と SUBPROGB を同時に使用可能ストレージに収容することができない場合があります。ON OVERFLOW 句は、このような事態に対処し、他のサブプログラムによって占有されているストレージを解放するために使用します。

CMPR2 の動作を行うプログラムのもとで実行する場合は、「ストレージ不足」エラーの場合にのみ ON OVERFLOW 条件になるため、上記のような指定は適切です。

NOCMPR2 のもとで実行する場合は、「ストレージ不足」エラー以外の場合でも ON OVERFLOW 条件になることがあるため、2 回目の呼び出し (ON OVERFLOW 句内部の) も失敗する可能性があります。

CALL . . . ON OVERFLOW に対する修正処置:

この技法またはこれと類似した技法を使用するプログラムに一般に適用できる修正処置はありません。「ストレージ不足」エラーのため ON OVERFLOW 条件になるのであれば、プログラムは以前と同じ動作をします。他のエラーのため ON

OVERFLOW 条件になるのであれば、リカバリー・コード (ON OVERFLOW 句で提供される) はそのエラーを訂正しないことがあり、後続の CALL も失敗します。

一般に、Enterprise COBOL プログラムで、ON OVERFLOW 条件を発生させたエラーの実際の原因を判別することはできません。

位取り整数と非数字との比較

非数字項目と数値項目 (整数のみ) とを比較する場合、比較で使用される数値項目の値は、それが位取りされているかどうかによって異なります。

CMPR2

CMPR2 のもとでは、非数字項目との比較演算では、位取りされた数値項目の数値または代数値が使用されます。代数値を判別するときには、PICTURE 文字ストリングのすべての記号 P が合計桁数に含まれ、P の位置にはゼロが使用されます。

NOCMPR2

NOCMPR2 のもとでは、非数字項目との比較演算では、位取りされた数値項目の実際の文字表現または文字値が使用されます。位取りされた数値項目の文字値には、記号 P で指定された桁位置は含まれません。これらの桁位置は無視され、オペランドのサイズとしては数えられません。

以下に例を示します。

```
01  NUM    PIC 99PP  VALUE 2300.
01  ALPHA1 PIC XX    VALUE "23".
01  ALPHA2 PIC XXX   VALUE "23".
01  ALPHA3 PIC XXXX  VALUE "2300".

    IF NUM EQUAL ALPHA1 DISPLAY "ALPHA1".
    IF NUM EQUAL ALPHA2 DISPLAY "ALPHA2".
    IF NUM EQUAL ALPHA3 DISPLAY "ALPHA3".

                     CMPR2              NOCMPR2

Results              ALPHA3             ALPHA1
displayed                               ALPHA2
```

この例では、NOCMPR2 のもとでは、NUM の文字値には 2 つの桁位置しかありません。ALPHA2 のような長さの異なる非数字項目と NUM を比較する場合、短いほうのオペランド (NUM) に、他方のオペランドと同じ長さになるようにブランクが埋め込まれます。

メッセージ

CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、位取り整数と非数字項目との間のすべての比較に対して以下のメッセージが出されます。

IGYPG3138-W

MIGR 位取り整数項目 " " と非数字項目 " " との比較は、"NOCMPR2" コンパイラー・オプションのもとでは異なる方法で実行されます。

位取り整数と非数字との比較に対する修正処置:

CMPR2 での動作を保持するために、位取りされた整数を構造内に定義することができます。FILLER は、整数の位取り位置のプレースホルダーとしての役割を果たすものであり、ゼロに初期設定する必要があります。FILLER には、NUM の位取り位置と同じ数だけ英数字位置を定義する必要があります。非数字項目との比較に NUM を使用するときには、NUM を CHARVAL に置き換えてください。

```
01  CHARVAL.
  05  NUM     PIC 99PP  VALUE 2300.
  05  FILLER  PIC XX    VALUE "00".

      IF CHARVAL EQUAL ALPHA1 DISPLAY "ALPHA1".
      IF CHARVAL EQUAL ALPHA2 DISPLAY "ALPHA2".
      IF CHARVAL EQUAL ALPHA3 DISPLAY "ALPHA3".
```

非 COBOL 文字を使用する COPY...REPLACING ステートメント

このセクションでは、ライブラリー・テキストまたは COPY ... REPLACING ステートメント内で使用する非 COBOL 文字のうち、CMPR2 と NOCMPR2 とでは扱い方が異なる 3 つの種類の文字について説明します。

「非 COBOL」文字とは、正式な COBOL 文字セットに含まれない EBCDIC 文字 (非数値リテラルを除く) のことです。非数値リテラルには、コンピューターの文字セット内の任意の文字を含めることができます。

CMPR2

CMPR2 のもとでは、ライブラリー・テキストと COPY...REPLACING ステートメントに、「非 COBOL」文字で構成されるオペランドを含めることができます。

NOCMPR2

COBOL 85 標準では、すべての非 COBOL 文字は使用することができません。小文字およびコロンが文字セットに追加されました。

英小文字

CMPR2 では非 COBOL 文字であった英「小」文字が、Enterprise COBOL の正式な COBOL 文字セットに追加されました。CMPR2 では、COPY を使用して小文字の置き換えを行うことができます。

```
COPY  A  REPLACING == abc ==  BY  == XYZ ==.
```

上記の例では、すべての "abc" が検出されて "XYZ" で置き換えられます。これに対して、Enterprise COBOL では、データ名に使われている小文字と英大文字は同等として扱われるため、"abc" と "ABC" がすべて "XYZ" で置き換えられます。メンバー A に以下のものが含まれている場合、

```
1  abc PIC X.
1  ABC PIC XX.
```

結果は次のようになります。

```
   CMPR2                          NOCMPR2

After COPYing & REPLACING     After COPYing & REPLACING
 1  XYZ PIC X.                    1  XYZ  PIC X.
 1  ABC PIC XX.                   1  XYZ  PIC XX.
```

メッセージ

FLAGMIG コンパイラー・オプションを指定すると、動作の違いにフラグが立てられます。

IGYLI0161-W

MIGR 桁 " " で検出された英小文字 " " は、"NOCMPR2" コンパイラー・オプションのもとでは対応する大文字と同じように扱われます。結果が異なる可能性があります。

英小文字に対する修正処置:

Enterprise COBOL のもとで CMPR2 プログラムをコンパイルして同じ結果を得るには、REPLACING 引き数が (大文字への変換後も) すべて固有になっているか検証する必要があります。

コロン (:) 文字

CMPR2 では、コロンは、COPY...REPLACING のオペランドの一部として使用できる非 COBOL 文字でした。この文字は、Enterprise COBOL のもとでは正式な COBOL 区切り文字です。

```
COPY  A  REPLACING  == A ==    BY  == X ==
                    == B ==    BY  == Y ==
                    == A:B ==  BY  == Z ==.
```

メンバー A に以下のものが含まれている場合、

```
MOVE  A:B  TO  ID2.
```

COPY と REPLACING の後、CMPR2 と Enterprise COBOL 間の違いは以下のようになります。

CMPR2	NOCMPR2
`MOVE  Z  TO  ID2.`	`MOVE  X:Y  TO  ID2.`

":" は Enterprise COBOL では区切り文字の 1 つなので、"A:B" という記述は 3 つの別々の字句 ("A"、":"、および "B") に分割されます。A と B がまず置き換えられます。

メッセージ

FLAGMIG を指定すると、2 つのリリース間でのこの動作の違いにフラグが立てられます。

IGYLI0160-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、コロンは区切り文字として使用されます。結果が異なる可能性があります。

コロン (:) 文字に対する修正処置:

上記のコードが CMPR2 の場合と同様に動作するようにするためには、REPLACING 文節を以下のように変更してください。

```
COPY  A  REPLACING  == A:B ==  BY  == Z ==
                    == A ==    BY  == X ==
                    == B ==    BY  == Y ==.
```

無効な文字

正式な COBOL 文字セットに含まれない文字がいくつかあります。以下の例を考えてください。

```
COPY  A  REPLACING  == % ==  BY  == 1 ==.
```

メンバー A に以下のものが含まれているとします。

```
%  XDATA  PIC  X.
```

ここで、「非 COBOL」文字は "%" 文字です。

CMPR2 と NOCMPR2 のどちらのもとでも、上記のメンバーは置き換えが実行された上でコピーされます。Enterprise COBOL コンパイラーは E レベルの診断を出します。

IGYLI0163-E
非 COBOL 文字 "%" が、8 桁目で検出されました。文字は受け入れられました。

いずれの場合にも、すべての COPY ステートメントが処理された後、正式な COBOL プログラムが生成されます。

メッセージ

FLAGMIG を指定すると、2 つのリリース間でのこの動作の違いにフラグが立てられます。

IGYLI0162-W
MIGR 桁 8 で検出された非 COBOL 文字 "%" は "NOCMPR2" コンパイラー・オプションのもとでは診断されます。結果が異なる可能性があります。

無効な文字に対する修正処置:

ソース・プログラムと COPY ライブラリーからすべての非 COBOL 文字を除去し、COBOL 文字と置き換えてください。

非 COBOL 文字を除去することによって、Enterprise COBOL の以後のリリースで新たに発生する問題を予防できます。以後のリリースでは、(コロンなどのように) これらの文字のいずれかに意味が指定される可能性があるため、結果が異なる可能性があります。

国別拡張文字を使用する COPY ステートメント

CMPR2

国別拡張文字 @、#、および $ は、COPY ステートメントのテキスト名およびライブラリー名で使用できます。たとえば、`COPY X$3.` では項目がコピーされます。

NOCMPR2

コンパイラーは E レベルの診断を出します。

IGYLI0025-E
名前 ″X$3″ が無効でした。″X03″ として処理されました。

Enterprise COBOL では、非数値リテラルの形式であれば、国別拡張文字 (@、#、$) をテキスト名およびライブラリー名に使用することができます。たとえば、Enterprise COBOL で X$3 をコピーするには、`COPY "X$3".` とコーディングします。

メッセージ

FLAGMIG を指定すると、動作の違いにフラグが立てられます。

IGYLI0115-W

MIGR 名前 ″X$3″ はプログラム名の形成規則に従っていません。この名前は ″NOCMPR2″ コンパイラー・オプションのもとでは診断されます。

国別拡張文字を使用する COPY ステートメントに対する修正処置:

ソース・プログラムと COPY ライブラリー内のすべての国別拡張文字を COBOL 文字に変更してください。

ファイル状況コード

CMPR2

CMPR2 ではファイル状況コードが戻されます。

NOCMPR2

NOCMPR2 ではファイル状況コードが拡張されました。新しいファイル状況コードおよび変更されたファイル状況コードが戻され、入出力操作の状況についてのより詳しい情報が示されます。さらに、場合によっては早期に問題が検出され、「欠落している」ファイルの場合は定義およびファイル状況条件が更新されています。

メッセージ

ファイル状況データ名を含んでいるプログラムを CMPR2 および FLAGMIG コンパイラー・オプションでコンパイルすると、以下のメッセージが出されます。

IGYGR1188-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、ファイル状況値が異なっています。

ファイル状況コードに対する修正処置:

CMPR2 の状況コードと Enterprise COBOL の状況コードの間には 1 対 1 の対応付けはありませんが、表 43 では CMPR2 とNOCMPR2 のファイル状況コードの一般的な関係を示しています。Enterprise COBOL ファイル状況コードの詳細については、*Enterprise COBOL 言語解説書* を参照してください。

表 43. CMPR2 と NOCMPR2 での QSAM および VSAM ファイル状況コード

VSAM ファイル状況コード		QSAM ファイル状況コード	
CMPR2	**NOCMPR2**	**CMPR2**	**NOCMPR2**
00	00 04 05 14 24 35 39 44	00	00 04 05 07 39 44
02	02		
10	10	10	10
21	21		
22	22		
23	23		
24	24		
30	30 39	30	30 39
		34	34
90	37 90	90	35 37 90
91	91		
92	38 41 42 43 44 47 48 49 92	92	38 41 42 43 46 47 48 49 92
93	93		
94	46		
95	39 95		
96	96		
97	97		

暗黙の EXIT PROGRAM

プログラムを終了するためには、EXIT PROGRAM、STOP RUN、または GOBACK ステートメントを使用しなければなりません。呼び出し先サブプログラムの場合は EXIT PROGRAM を使用でき、メインプログラムの場合は STOP RUN を使用できます。GOBACK (IBM 拡張) は、いずれのタイプのプログラムにも使用できます。

CMPR2

CMPR2 のもとでは、プログラムが上記のいずれのステートメントも含んでいない場合は、コンパイラー警告診断メッセージが出され、プログラムが終了できることを確認するためにプログラムを分析するように指示されます。

以下の行がプログラムの最後の行であるとします。

```
IF TALLY = 0 THEN STOP RUN.
```

この場合は、コンパイラー診断メッセージは出されず、IF 条件をテストした結果が偽であった場合にのみ、以下のランタイム・メッセージが出されます。

IGZ0037S

プログラム "program-name" での制御のフローが、プログラムの最後の行を越えました。

NOCMPR2

NOCMPR2 のもとでは、すべてのプログラムが EXIT PROGRAM ステートメントで終了すると想定されています。呼び出し先サブプログラムの場合、制御のフローがサブプログラムの最後の行を越えることはありませんが、サブプログラムが呼び出しプログラムに戻ります。上記の例で、サブプログラムが以下のステートメント

```
IF TALLY = 0 THEN STOP RUN.
```

で終了し、テストの結果が偽であると、制御は呼び出し元に戻されます。CMPR2 での動作の場合、結果は異常終了になります。

メインプログラムの場合、EXIT PROGRAM ステートメントは何の影響も与えません。そのため、コンパイラーによって生成された暗黙の EXIT PROGRAM は、プログラムの実行に影響を与えません。メインプログラムの最後の行を越えて実行されると、やはり異常終了します。

メッセージ

プログラムが STOP RUN、GOBACK、EXIT PROGRAM のいずれのステートメントも含んでいない場合は、以下の診断メッセージが出されます。

IGYPS2091-W

"STOP RUN"、"GOBACK"、または "EXIT PROGRAM" がプログラムで検出されませんでした。プログラム・ロジックを調べて、プログラムが終了することを確認してください。

また、CMPR2 および FLAGMIG コンパイラー・オプションを使用すると、以下のメッセージが出されます。

IGYPS2223-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、このプログラムの終わりで暗黙の "EXIT PROGRAM" が実行されます。

プログラムが STOP RUN、GOBACK、または EXIT PROGRAM ステートメントを含んでおり、NOOPTIMIZE コンパイラー・オプションが有効である場合、FLAGMIG コンパイラー・オプションを使用すると、以下のメッセージが出されます。

IGYPS2224-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、このプログラムの終わりで暗黙の "EXIT PROGRAM" が実行される可能性があります。"OPTIMIZE" および "FLAGMIG" コンパイラー・オプションを指定して再コンパイルしてください。暗黙の "EXIT PROGRAM" についての "MIGR" メッセージが出力されない場合は、暗黙の "EXIT PROGRAM" は実行されません。

OPTIMIZE コンパイラー・オプションを指定して再コンパイルしたときに、このようなメッセージのいずれも出されなければ、プログラムのために暗黙の EXIT PROGRAM が生成されないということを示していますが、以下のメッセージが出される場合には、暗黙の EXIT PROGRAM が生成されることを示しています。

IGYOP3210-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、このプログラムの終わりで暗黙の "EXIT PROGRAM" が実行されます。

暗黙の EXIT PROGRAM に対する修正処置:

CMPR2 での動作を保持するために、プログラムを変更して、新しいセクションとセクション名をプログラムの最後のセクションとして含めてください。その新しいセクションに、エラー処理コード (ILBOABN0 への呼び出しなど) を含めることができます。

EXIT PROGRAM が暗黙的に生成されることを示すメッセージが出されたプログラムを調べて、プログラムが正しく終了することを確認してください。

PERFORM から戻る方法

ある段落 (または、ある範囲の複数の段落) が PERFORM ステートメント (「行外 PERFORM」) によって実行されるとき、指定段落の終了時のメカニズムによって、制御が PERFORM ステートメントの直後に戻るようになっています。

以下の例を考えてください。

```
    PERFORM A
    STOP RUN.
A.  DISPLAY "Hi".
B.  DISPLAY "there".
```

コンパイラー生成コードは、メッセージ "Hi" を表示した後、段落 A の実行後に制御のフローが STOP RUN ステートメントへ戻るようにします。これが行われないと、制御が段落 B に移ることになります。

このコード・メカニズムは、プログラムが初めて呼び出されたとき、またはプログラムが取り消されたときに、初期状態にリセットされます。NOCMPR2 のもとでは、さらに、プログラムが呼び出されるたびにリセットされます。CMPR2 のもとでは、プログラムが取り消されずに連続して 2 回呼び出されたとき、このメカニズムは最後に使用された状態のままになります。これは、PERFORM ステートメントすべての実行が完了する前にプログラムが EXIT PROGRAM または GOBACK ステートメントを出すような場合に重要になります。

以下の例について考えてください。

```
    IF FIRST-TIME-CALLED THEN
      PERFORM A
      MOVE ZERO TO N
    ELSE
      SUBTRACT 1 FROM N
      GO TO A.
    GOBACK.
A.  IF N > 1 THEN
      GOBACK.
B.  DISPLAY "Processing continues...".
```

スイッチ FIRST-TIME-CALLED がプログラムに渡されました。このスイッチは、プログラムが取り消されずに呼び出されたかどうかをプログラムに通知します。変数 N もプログラムに渡されました。

CMPR2

プログラムが初めて呼び出されたときは、PERFORM ステートメントが実行されます。テスト "N > 1" の結果が真であると、プログラムは呼び出しプログラムに戻ります。

ただし、これは、段落 A が実行個所に戻らなかったために、PERFORM ステートメントが正常に完了しなかったことを意味します。段落 A の終わりにおけるコンパイラー生成メカニズムは、PERFORM ステートメントに戻るように「設定」されたままになっています。

したがって、プログラムが 2 回目に呼び出されたとき、ELSE 側に進み、N から 1 が引かれ、GO TO ステートメントによって制御が段落 A に渡ります。ただし、テスト "N > 1" の結果が偽である場合、PERFORM のメカニズムは設定されたままになっています。段落 A の終わりに達すると、段落 B に進むのではなく、PERFORM ステートメントの後の MOVE ステートメントに制御が戻されます。

このような結果は、意図したものではありません。この問題は、以下のことがすべて発生した場合に発生する可能性があります。

1. プログラムが、EXIT PROGRAM または GOBACK ステートメントによって呼び出しプログラムに戻る場合。
2. PERFORM ステートメントがある段落 (または、ある範囲の段落) を実行し、さらに GO TO ステートメントによって、またはその段落に制御が移ることによってもその段落に達する可能性がある場合。
3. EXIT PROGRAM または GOBACK ステートメントが実行される前に、このような PERFORM ステートメントに戻る機会がなかった場合。

NOCMPR2

NOCMPR2 のもとでは、プログラムが最初に呼び出されたとき、PERFORM ステートメントが実行され、制御が段落 A に移ります。その後、プログラムは、テスト "N > 1" の結果に応じて、ただちに呼び出しプログラムに戻る、PERFORM に戻り N にゼロを移送してから呼び出しプログラムに戻る、のいずれかになります。

プログラムが次に呼び出されたときは、ELSE 側に進み、N から 1 が引かれ、GO TO ステートメントによって段落 A に制御が渡されます。その後、プログラムは、テスト "N > 1" の結果に応じて、ただちに呼び出しプログラムに戻る、段落 B に進んでメッセージを表示して処理を続ける、のいずれかになります。

取られる進路に関係なく、PERFORM ステートメントを制御するメカニズムは、プログラムが呼び出されるたびにリセットされるので、不規則な制御フローが発生することはありません。

メッセージ

ライン外 PERFORM と EXIT PROGRAM または GOBACK のいずれかのステートメントを含んでいるプログラムを CMPR2、FLAGMIG、および NOOPTIMIZE コンパイラー・オプションでコンパイルすると、以下のメッセージが出されます。

IGYPA3205-W
: **MIGR** "NOCMPR2" コンパイラー・オプションのもとでは、"EXIT PROGRAM" または "GOBACK" ステートメントは "PERFORM" 範囲の終わりに達したものと想定しています。このプログラムをサブプログラムとして使用する場合、マイグレーション後は実行結果が異なる可能性があります。

IGYPA3206-W
: **MIGR** "PERFORM" 範囲の終わりに関する詳細を入手するには、"OPTIMIZE" および "FLAGMIG" コンパイラー・オプションを指定して再コンパイルしてください。"PERFORM" 範囲の終わりに関するメッセージが出されなかった場合は、このプログラムには "PERFORM" 範囲の終わりに関するマイグレーション上の問題はありません。

OPTIMIZE コンパイラー・オプションを指定して再コンパイルしたときに、このようなメッセージのいずれも出されなければ、プログラムには、ライン外 PERFORM ステートメントの範囲内で実行される EXIT PROGRAM または GOBACK ステートメントの問題がないということを示していますが、以下のメッセージが出される場合には、問題があることを示しています。

IGYOP3205-W
: **MIGR** "NOCMPR2" コンパイラー・オプションのもとでは、"EXIT PROGRAM" または "GOBACK" ステートメントは、"PERFORM" 範囲の終わりに達したものと想定しています。このプログラムをサブプログラムとして使用する場合、マイグレーション後は実行結果が異なる可能性があります。

IGYOP3092-W
: "PERFORM" ステートメントの範囲内の "PERFORM (LINE xx.xx)" で "EXIT PROGRAM" または "GOBACK" ステートメントが検出されました。プログラムに再入すると、予期しない制御フローが発生する可能性があります。

PERFORM の戻りのメカニズムに対する修正処置:

広範囲にわたる複雑なコーディングを再び行わなければ、影響を受けるプログラムの CMPR2 での動作を保持することはできません。このようなプログラムは、書き直して、CMPR2 での動作に依存しないようにしてください。

PERFORM ... VARYING ... AFTER

ID が、異なる方法で設定され、増分されます。以下に例を示します。

```
PERFORM PARA3 VARYING id-2 FROM id-3 BY id-4
              UNTIL condition-1
        AFTER id-5 FROM id-6 BY id-7
              UNTIL condition-2.
```

CMPR2

CMPR2 のもとでは、PERFORM ステートメントの VARYING...AFTER 句の中で、id-5 が設定されてから、id-2 が増加されます。

CMPR2 のもとで 2 つの変数を変更する場合、中間段階で内側の条件が真であれば、内側の変数 (id-5) が現在の FROM 値 (id-6) に設定され、その後、外側の変数 (id-2) が現在の BY 値 (id-4) だけ増加されます。

NOCMPR2

しかし、NOCMPR2 のもとでは、id-2 が増加されてから、id-5 が設定されます。id-6 が id-2 に依存している場合、この変更のため、互換性が保持されなくなります。

以下の例を考えてください。

```
PERFORM PARA3 VARYING X FROM 1 BY 1 UNTIL X IS GREATER THAN 3
          AFTER Y FROM X BY 1 UNTIL Y IS GREATER THAN 3.
```

この例では、id-6(X) と id-2(X) は同じであるため、id-6(X) は id-2(X) に依存しています。

CMPR2 のもとでは、以下の値で、PARA3 が 8 回実行されます。

```
X:    1   1   1   2   2   2   3   3
Y:    1   2   3   1   2   3   2   3
```

NOCMPR2 のもとでは、以下の値で、PARA3 が 6 回実行されます。

```
X:    1   1   1   2   2   3
Y:    1   2   3   2   3   3
```

最初の ID が、2 番目の ID と同じであるか、2 番目の ID によって添え字付けされているか、2 番目の ID を部分的または完全に再定義したものであるか、2 番目の ID に応じて位置指定が可変である場合に、ID 間の依存性が発生します。

メッセージ

まず、すべてのプログラムを、以前のバージョンの COBOL コンパイラーで CMPR2 および FLAGMIG コンパイラー・オプションを指定して再コンパイルしてください。これによって、以下の変数間に依存性がある PERFORM...VARYING ステートメントにフラグが立てられます。

- id-6 が (潜在的に) id-2 に依存している
- id-9 が (潜在的に) id-5 に依存している
- id-4 が (潜在的に) id-5 に依存している
- id-7 が (潜在的に) id-8 に依存している

注: AFTER 句を指定した PERFORM...VARYING だけが影響を受けます。

たとえば、id-6 が id-2 に依存している場合、以前のバージョンの COBOL コンパイラーで CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、コンパイラーは以下のメッセージを出します。

IGYPA3209-W

MIGR "PERFORM ... VARYING" オペランド "ID-6 (NUMERIC INTEGER)"は "ID-2 (NUMERIC INTEGER)" に依存していました。"NOCMPR2" コンパイラー・オプションのもとでは、"PERFORM ... VARYING" オペランドの増加 / 設定に関する規則が変更されたため、このステートメントは互換性のない結果になる可能性があります。

PERFORM . . . VARYING . . . AFTER に対する修正処置

FLAGMIG によってフラグが立てられた PERFORM...VARYING ステートメントは変換する必要があります。上記の *4 つの依存関係*がすべて該当する PERFORM...VARYING ステートメントの変換方法の 1 つを以下に示します。

```
PERFORM xx
   VARYING id-2 FROM id-3 BY id-4 UNTIL cond-1
     AFTER  id-5 FROM id-6 BY id-7 UNTIL cond-2
     AFTER id-8 FROM id-9 BY id-10 UNTIL cond-3.
```

これは、以下のように変換します。

```
MOVE id-3 TO id-2.
MOVE id-6 TO id-5
MOVE id-9 TO id-8

PERFORM UNTIL cond-1
  PERFORM UNTIL cond-2
    PERFORM UNTIL cond-3
      PERFORM xx
      ADD id-10 TO id-8
    END-PERFORM
    MOVE id-9 TO id-8
    ADD id-7 TO id-5
  END-PERFORM
  MOVE id-6 TO id-5
  ADD id-4 TO id-2
END-PERFORM
```

この例では、すべての id-x が ID であると仮定しています。いずれかが指標名である場合は、MOVE ステートメントではなく SET ステートメントを使用しなければなりません。

上記の例は、変換の最悪の例です。(潜在的に) 依存関係のある ID を使用するステートメントの一部を変更するだけで、改良することができます。たとえば、id-6 が id-2 に依存しているだけであり、他の依存関係はない場合は、上記の変換を以下のように減らすことができます。

```
MOVE id-3 TO id-2.
MOVE id-6 TO id-5.

PERFORM UNTIL cond-1
  PERFORM UNTIL cond-2
    PERFORM VARYING  id-8 FROM id-9 BY id-10 UNTIL cond-3
         PERFORM XX
    END-PERFORM
    ADD id-7 TO id-5
  END-PERFORM
  MOVE id-6 TO id-5
  ADD id-4 TO id-2
END-PERFORM
```

"A" および "B" を指定した PICTURE 文節

CMPR2

CMPR2 のもとでは、PICTURE 文節に記号 B が指定されたデータ項目は、英字データ項目です。

NOCMPR2

NOCMPR2 のもとでは、PICTURE 文節に記号 B が指定されたデータ項目は、英数字編集項目です。

この変更によって問題が発生する機能はほとんどありません。ただし、INITIALIZE、STRING、CALL、および CANCEL 動詞については、CMPR2 から Enterprise COBOL にアップグレードするときに注意する必要のある微妙な事柄がいくつかあります。

メッセージ

CMPR2 および FLAGMIG オプションを指定してプログラムをコンパイルすると、記号 B で定義された英字項目に関してメッセージが出されます。

IGYDS1105-W

MIGR 記号 "A" と "B" で構成されている "PICTURE" 文節が検出されました。この英字項目は、"NOCMPR2" コンパイラー・オプションのもとでは英数字編集項目として扱われます。

INITIALIZE 動詞

以下の例を考えてください。

```
01 ALPHA  PIC AABAABAA.

INITIALIZE ALPHA REPLACING ALPHABETIC DATA BY ALL "3".
```

CMPR2 のもとでは、上記のようにコーディングされたステートメントは有効であり、初期設定が行われます。しかし、NOCMPR2 のもとでは、このステートメントに関して以下の警告メッセージが出され、初期設定は行われません。

IGYPS2047-W

"INITIALIZE" ステートメントの受け取り側の "ALPHA" は、"REPLACING" オペランドのデータ・カテゴリーと互換性がありませんでした。"ALPHA" は初期設定されませんでした。

この非互換は、項目のグループが初期設定されるときも発生する可能性があります。NOCMPR2 のもとでは、上記の ALPHA は英数字編集として分類されます。初期設定されるグループに ALPHA が定義されている場合、初期設定される英字項目がなかったときにのみ、上記のようなメッセージが出されます。したがって、以下の例では、ALPHA は初期設定されませんが、その事実を警告するメッセージは出されません。

```
01  GROUP1.
    05 ALPHA  PIC AABAA.
    05 BETA   PIC AAA.

INITIALIZE GROUP1 REPLACING ALPHABETIC DATA BY ALL "5".
```

INITIALIZE 動詞に対する修正処置

このような再分類されたデータ項目を以前と同じ方法で初期設定するためには、上記の最初の例における元のステートメントを以下のように変更してください。

```
INITIALIZE ALPHA REPLACING
           ALPHANUMERIC-EDITED DATA BY ALL "3".
```

2 番目の例のように、グループに複数のタイプが含まれている可能性がある場合は、INITIALIZE に句を追加する必要があります。以下に例を示します。

```
INITIALIZE GROUP1 REPLACING
                  ALPHABETIC DATA BY ALL "5"
                  ALPHANUMERIC-EDITED DATA BY ALL "5".
```

注: すでにこの句を指定していたが、置き換えたデータが異なる場合、またはグループ内に初期設定したくない別の英数字編集項目がある場合に、この句を追加すると、矛盾が発生することがあります。

STRING 動詞

CMPR2 または NOCMPR2 のいずれでも、STRING...INTO の受け取りフィールドに英字項目を指定することができます。ただし、編集項目は指定できません。さらに、CMPR2 プログラムに英字項目が STRING 動詞のこの位置に記号 B を指定して定義されている場合、これらのステートメントに対して Enterprise COBOL から重大エラーのメッセージが出されます。これは、この項目が英数字編集項目として再分類されたためです。

IGYPA3104-S

"STRING INTO" の ID "ALPHA (ALPHANUMERIC-EDITED)" は、編集データ項目であるか、または "JUSTIFIED" 文節によって定義されていました。このステートメントは無視されました。

STRING 動詞に対する修正処置

CMPR2 の STRING ステートメントは、記号 B で表された位置を自動的にオーバーレイするので、元の INTO フィールドに新しい英字データ名を再定義することだけが必要です。以下に例を示します。

CMPR2 のもとでのステートメント:

```
01 ALPHA  PIC AABAABAA.
01 VARX   PIC A(3)  VALUE "XXX".
01 VARY   PIC A(3)  VALUE "YYY".

   STRING VARX VARY DELIMITED BY SIZE INTO ALPHA.
```

NOCMPR2 のもとでのステートメント:

```
01 ALPHA  PIC AABAABAA
01 BETA   REDEFINES ALPHA   PIC A(8).
01 VARX   PIC A(3)  VALUE "XXX".
01 VARY   PIC A(3)  VALUE "YYY".

   STRING VARX VARY DELIMITED BY SIZE INTO BETA.
```

ALPHA に BETA を再定義します。BETA の長さは ALPHA (すべての記号 B を含む) と同じです。次に BETA は STRING ステートメントで使用されます。STRING が実行された後の ALPHA の値は、CMPR2 の場合と同じ値になります。

CALL および CANCEL 動詞

IBM 拡張によって、CALL および CANCEL ステートメントの ID として英字データ項目を使用できるようになりました。ただし、英数字編集項目は使用できません。したがって、記号 B で定義された英字項目を使用している CMPR2 プログラムの場合は、重大エラー・メッセージが出されます。たとえば、以下のプログラムは CMPR2 では動作しましたが、現在は、重大エラー・メッセージが出されます。

```
01  CALLDN   PIC  AAAAABB.

      MOVE "PROG1" TO CALLDN.
      CALL CALLDN.
      CANCEL CALLDN.
```

IGYPA3063-S

"CALL" または "CANCEL" の ID "CALLDN (ALPHANUMERIC-EDITED)" が英数字項目でもなく、ゾーン 10 進数でもなく、英字でもありませんでした。ステートメントは無視されました。

Enterprise COBOL でコンパイルするには、CALLDN の定義をすべて英字または英数字に変更するか、以下に示すように、CALLDN を有効なデータ型で再定義する新しいデータ名を追加してください。

```
01 CALLDN PIC A(7).
        or
01 CALLDN PIC X(7).
        or

01 CALLDN PIC AAAAABB
01 CALLDN1 REDEFINES CALLDN PIC A(7).

    MOVE "PROG1" TO CALLDN1.
    CALL CALLDN1.
    CANCEL CALLDN1.
```

PROGRAM COLLATING SEQUENCE

CMPR2

OBJECT COMPUTER 段落で確立された PROGRAM COLLATING SEQUENCE は、以下のような非数値の比較の真理値を判別するために使用されます。

- 比較条件に明示的に指定された比較。
- 条件名条件に明示的に指定された比較。
- SORT および MERGE ステートメントの実行の一部として暗黙的に実行される比較 (それぞれの SORT または MERGE ステートメントの COLLATING SEQUENCE 句によってオーバーライドされる場合を除く)。
- STRING、UNSTRING、および INSPECT ステートメントの実行の一部として暗黙的に実行される比較。

NOCMPR2

OBJECT COMPUTER 段落で確立された PROGRAM COLLATING SEQUENCE は、以下のような非数値の比較の真理値を判別するために使用されます。

- 比較条件に明示的に指定された比較。
- 条件名条件に明示的に指定された比較。

- SORT および MERGE ステートメントの実行の一部として暗黙的に実行される比較 (それぞれの SORT または MERGE ステートメントの COLLATING SEQUENCE 句によってオーバーライドされる場合を除く)。

STRING、UNSTRING、および INSPECT ステートメントの実行の一部として暗黙的に実行される非数値の比較の真理値を判別する場合は、*固有照合シーケンス*が使用されます。

ほとんどのアプリケーションの場合、この違いは、これらのステートメントの結果に影響を与えません。STRING、UNSTRING、および INSPECT ステートメントの一部として行われる暗黙の比較は常に品質的に等しくなります。したがって、PROGRAM COLLATING SEQUENCE 内の文字の順序が固有シーケンスの順序と異なっていても、比較の結果は同じになります。

この変更の影響を受けるアプリケーションの場合は、OBJECT COMPUTER 段落で確立された PROGRAM COLLATING SEQUENCE で、ALSO 文節 (複数の異なる文字を同じ順序位置に割り当てる) を用いて定義された英字を識別する必要があります。

メッセージ

CMPR2 および FLAGMIG オプションを指定してプログラムをコンパイルすると、この変更の影響を受ける可能性のあるすべてのステートメントに対して以下のメッセージが出されます。

IGYPS3142-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、"PROGRAM COLLATING SEQUENCE" は "STRING" ステートメントに影響を与えません。

修正処置

同じ順序位置に割り当てられた複数の文字が PROGRAM COLLATING SEQUENCE に存在することが原因でこのメッセージが出された場合は、プログラムを修正する一般的な方法はありません。

広範囲にわたる複雑なコーディングを再び行わなければ、影響を受けるプログラムの CMPR2 での動作を保持することはできません。このようなプログラムは、書き直して、CMPR2 での動作に依存しないようにしてください。

READ INTO および RETURN INTO

ファイルに複数のレコード記述が含まれていて、その中に英数字ではないレコード記述がある場合、INTO 句を指定した READ (または RETURN) が異なるレコード記述を使用して実行されることがあります。

ファイルが影響を受けるのは、ファイルが固定形式であり、01 のレコード記述が複数あり、少なくとも 1 つのレコード記述が数値項目または数字編集項目の場合だけです。

コンパイラーは、暗黙の MOVE ステートメントの送り出しフィールドとして使用するレコード記述を決める場合、最も長い 01 のレコード記述を選択します。同じ長さのレコード記述が複数ある場合は、そのうちの最初のレコード記述が選択され

ます。CMPR2 と NOCMPR2 のいずれにおいても、このように行われます。ただし、最も長い 01 のレコード記述を判別する方法が異なります。

CMPR2

CMPR2 のもとでは、数値および数字編集のレコード記述の長さは、PICTURE 内の桁位置の数を合計することによって計算されます。ほかのタイプのレコード記述には、レコード記述が占有するバイトの数と等しい長さが割り当てられます。

NOCMPR2

NOCMPR2 のもとでは、各レコード記述の長さは、レコード記述が数値、数字編集、またはそれ以外であるかどうかに関係なく、レコード記述が占有するバイトの数として決められます。

メッセージ

FLAGMIG および CMPR2 コンパイラー・オプションを指定すると、影響を受ける可能性のある READ INTO または RETURN INTO ステートメントに対してメッセージが出されます。

規則の変更による影響を受けるプログラムの場合、以下のメッセージが出されます。

IGYPS2281-I

"READ" または "RETURN" ステートメントの "INTO" 句が、複数レコードが含まれている固定形式ファイル "file-name" に指定されました。レコード "record-name" が MOVE の送り出しフィールドとして選択されました。

このメッセージは、CMPR2 と NOCMPR2 のいずれのコンパイラー・オプションのもとでも出されます。したがって、プログラムを CMPR2 を指定してコンパイルした後、NOCMPR2 を指定してコンパイルし、メッセージを調べることによって、CMPR2 と NOCMPR2 のいずれのもとでも同じレコードが選択されたかどうかを判別することができます。同じレコードが選択された場合は、プログラムを変更する必要はありません。

さらに、FLAGMIG コンパイラー・オプションを指定すると、以下のメッセージが出されます。

IGYPS2283-W

MIGR "READ" または "RETURN" ステートメントの "INTO" 句が、複数のレコードを含んでいるファイル "file-name" に指定されました。"NOCMPR2" コンパイラー・オプションのもとでは、異なるレコードが移動の送り出しフィールドとして選択される可能性があります。

READ INTO と RETURN INTO 句に対する修正処置:

修飾されたファイルごとにレコード記述規則を適用するか、またはメッセージを調べることによって、NOCMPR2 のもとでは CMPR2 の場合とは異なるレコード記述が選択されるかどうかを判別することができます。たとえば、以下のレコード記述について考えてください。

```
01 RECORD-1 PIC X(9) USAGE DISPLAY.
01 RECORD-2 PIC 9(9) USAGE DISPLAY.
```

この場合、CMPR2 と NOCMPR2 のいずれのもとでも、各レコード記述の長さは "9" と計算されます。したがって、非互換性はありません。

しかし、レコード記述の長さの計算方法が異なる場合、以下の例について考えてください。

```
01 RECORD-3 PIC X(4) USAGE DISPLAY.
01 RECORD-4 PIC 9(9) USAGE COMP.
```

この場合、NOCMPR2 のもとでは、各レコード記述の長さは "4" と計算されます。一方、CMPR2 のもとでは、数値レコード記述 (RECORD-4) の長さは桁数によって計算されるので、この長さは "4" でなく "9" になります。したがって、各レコード記述のバイト長が 4 であっても、RECORD-4 が送り出しフィールドとして使用されます。

非互換性が検出された場合は、コードを変更して、CMPR2 での動作が保持されるようにしてください。READ INTO または RETURN INTO ステートメントを READ または RETURN ステートメントに変更して、その後ろに MOVE ステートメントを続けることができます。MOVE ステートメントでは、必要なレコード記述 (「最も長い」もの) を送り出しフィールドとして指定し、INTO 項目として指定されていた項目を受け取りフィールドとして指定します。

RECORD CONTAINS n CHARACTERS

RECORD CONTAINS n CHARACTERS の定義は既存のプログラムに影響を与え、また、その動作は CMPR2 のもとでは異なります。

以下の例を考えてください。

```
FD FILE1
   RECORD CONTAINS 40.
 01 F1R1 PIC X(20).
 01 F1R2 PIC X(40).


FD FILE2
   RECORD CONTAINS 20 TO 40.
 01 F2R1 PIC X(20).
 01 F2R2 PIC X(40).
```

CMPR2

CMPR2 のもとでは、FILE1 と FILE2 には可変長レコードが含まれます。

NOCMPR2

NOCMPR2 のもとでは、FILE1 には固定長レコードが、FILE2 には可変長レコードが含まれます。

メッセージ

CMPR2 および FLAGMIG オプションを指定してプログラムをコンパイルすると、FILE1 に関して以下のメッセージが出されます。

IGYPS1183-W

MIGR 整数が 1 つだけ指定された "RECORD CONTAINS" 文節は、"NOCMPR2" コンパイラー・オプションのもとでは異なる方法でサポートされます。

異なる記述を使用しているプログラムを Enterprise COBOL でコンパイルすると、OPEN 時にファイル状況 39 が発生する可能性があります。

RECORD CONTAINS n CHARACTERS 文節に対する修正処置:
現行の動作を保持するためには、RECORD CONTAINS 文節を除去してください。この変更により、FILE1 と FILE2 のいずれにも可変長レコードが含まれるようになります。

より明確にするために、あるいは新しいアプリケーションの場合は、固定長レコードには RECORD CONTAINS n CHARACTERS を使用し、可変長レコードには RECORD IS VARYING FROM integer-1 TO integer-2 を使用してください。RECORD CONTAINS n1 TO n2 CHARACTERS は使用しないようにしてください。

予約語

VS COBOL II リリース 3、IBM COBOL、および Enterprise COBOL で新しい予約語が追加されました。新しい予約語の完全なリストについては、表 44 を参照してください。

注: あるリリースで追加された予約語は、以降のどのリリースでも予約されています。

表 44. コンパイラーによる新しい予約語の VS COBOL II リリース 2 との比較

コンパイラー	予約語
VS COBOL II リリース 3 NOCMPR2	ALPHABET、 ALPHABETIC-LOWER、 ALPHABETIC-UPPER、 BINARY、 CLASS、 COMMON、 CONVERTING、 DAY-OF-WEEK、 DBCS、 END-RECEIVE、 EXTERNAL、 GLOBAL、 ORDER、 PACKED-DECIMAL、 PADDING、 PURGE、 REPLACE、 STANDARD-2
COBOL/370 V1R1	FUNCTION、 PROCEDURE-POINTER
COBOL (MVS および VM 版) V1R2	CLASS-ID、 METACLASS、 RECURSIVE、 END-INVOKE、 METHOD、 REPOSITORY、 INHERITS、 METHOD-ID、 RETURNING、 INVOKE、 OBJECT、 SELF、 SUPER、 LOCAL-STORAGE、 OVERRIDE
COBOL (OS/390 および VM 版) V2R1	COBOL (MVS および VM 版) と同様
COBOL (OS/390 および VM 版) V2R2	COMP-5、 COMPUTATIONAL-5、 EXEC、 END-EXEC、 SQL、 TYPE、 FACTORY
COBOL (OS/390 および VM 版) V2R2 (PQ49375 適用済み)	EXECUTE
Enterprise COBOL	JNIENVPTR、 XML、 END-XML、 XML-EVENT、 XML-CODE、 XML-TEXT、 XML-NTEXT、 FUNCTION-POINTER

メッセージ

CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、新しい予約語ごとにメッセージが出されます。

以下のメッセージは、新しい予約語 "ALPHABET" の場合の例です。

IGYPS0057-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、"ALPHABET" は新しい COBOL 予約語です。

新しい予約語に対する修正処置:

文字を追加または削除するか、またはワード全体を変更することによって、ワードを、予約されていない別の COBOL ワードに変更してください。

SET . . . TO TRUE

CMPR2

SET...TO TRUE ステートメントは MOVE ステートメントの規則に従って実行されます。

NOCMPR2

NOCMPR2 のもとでは、SET...TO TRUE は VALUE 文節の規則に従います。そのため、この変更によって、以下の 3 つの場合は異なる結果が生じる可能性があります。

- データ項目が JUSTIFIED 文節で記述されている場合
- データ項目が BLANK WHEN ZERO 文節で記述されている場合
- データ項目の PICTURE ストリングに編集記号がある場合

メッセージ

この変更の影響を受ける可能性のあるプログラムの場合、CMPR2 および FLAGMIG オプションを指定してコンパイルすると、以下のメッセージが出されます。

IGYPS2219-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、"TO TRUE" 句を指定した "SET" ステートメントは、"VALUE" 文節の規則に従って実行されます。

JUSTIFIED 文節

JUSTIFIED 文節で記述されたデータ項目が MOVE ステートメントの受け取り項目である場合、送り出しデータは受け取り項目の右端の文字位置に位置合わせされます。VALUE 文節では、初期設定は JUSTIFIED 文節による影響を受けません。つまり、VALUE 文節のデータは、受け取り項目の左端の文字位置に位置合わせされます。

以下は CMPR2 の場合の例です。

```
01 A PIC X(3) JUSTIFIED RIGHT VALUE "a". (Result = "a  ")
   88 V VALUE "a".

SET V TO TRUE                            (Result = "  a")
MOVE "a" TO A                            (Result = "  a")
```

以下は NOCMPR2 の場合の例です。

```
01 A PIC X(3) JUSTIFIED RIGHT VALUE "a". (Result = "a  ")
   88 V VALUE "a".
SET V TO TRUE                            (Result = "a  ")

MOVE "a" TO A                            (Result = "  a")
```

JUSTIFIED 文節に対する修正処置

NOCMPR2 を使用する場合に、CMPR2 の場合と同じ動作が必要であれば、88 レベル項目の VALUE 文節のデータを適宜調整してください。

```
01 A PIC X(3) JUSTIFIED RIGHT VALUE "a". (Result = "a  ")
   88 V VALUE "  a".

SET V TO TRUE                            (Result = "  a")
MOVE "a" TO A                            (Result = "  a")
```

BLANK WHEN ZERO 文節

BLANK WHEN ZERO 文節で記述されたデータ項目が MOVE ステートメントにおいてゼロの値を受け取る場合、データ項目にはスペースだけが入れられます。VALUE 文節では、初期設定は BLANK WHEN ZERO 文節による影響を受けません。つまり、VALUE 文節にゼロの値が指定されても、そのデータがそのまま項目に入れられ、項目の中はスペースではなく、すべてゼロになります。

以下は CMPR2 の場合の例です。

```
01 N PIC 9(3) BLANK WHEN ZERO VALUE ZERO. (Result = "000")
   88 V VALUE ZERO.

SET V TO TRUE                             (Result = "   ")
MOVE ZERO TO N                            (Result = "   ")
```

以下は NOCMPR2 の場合の例です。

```
01 N PIC 9(3) BLANK WHEN ZERO VALUE ZERO. (Result = "000")
   88 V VALUE ZERO.
SET V TO TRUE                             (Result = "000")

MOVE ZERO TO N                            (Result = "   ")
```

CMPR2 のもとでの動作が NOCMPR2 のもとで必要な場合は、88 レベル項目の VALUE 文節のデータを適宜調整してください。

```
01 N PIC 9(3) BLANK WHEN ZERO VALUE ZERO. (Result = "000")
   88 V VALUE "   ".

SET V TO TRUE                             (Result = "   ")
MOVE ZERO TO N                            (Result = "   ")
```

編集記号を指定した PICTURE ストリング

データ項目の PICTURE ストリングに編集記号が含まれている場合、データ項目にデータが移動されるときは、それらの編集記号で表された文字位置には編集文字が入れられます。VALUE 文節では、初期設定は編集記号による影響を受けません。つまり、VALUE 文節のデータがそのまま項目に入れられ、MOVE ステートメントで行われるような編集は行われません。

以下は CMPR2 の場合の例です。

```
01 E PIC X/X VALUE SPACE. (Result = "   ")
   88 V VALUE SPACE.

SET V TO TRUE             (Result = " / ")
MOVE SPACE TO E           (Result = " / ")
```

以下は NOCMPR2 の場合の例です。

```
01 E PIC X/X VALUE SPACE. (Result = "   ")
   88 V VALUE SPACE.
SET V TO TRUE             (Result = "   ")

MOVE SPACE TO E           (Result = " / ")
```

CMPR2 のもとでの動作が NOCMPR2 のもとで必要な場合は、以下のように、88 レベル項目の VALUE 文節のデータを編集形式で指定してください。

```
01 E PIC X/X VALUE SPACE. (Result = "   ")
   88 V VALUE " / ".

SET V TO TRUE             (Result = " / ")
MOVE SPACE TO E           (Result = " / ")
```

MULTIPLY と DIVIDE の SIZE ERROR

COBOL 74 標準 および COBOL 85 標準では、COMPUTE、DIVIDE、または MULTIPLY ステートメントに複数の受け取りフィールドがあるときは、中間結果がインプリメンターによって提供されることが明言されています。たとえば、`MULTIPLY A BY B GIVING C D` は次のように動作します。

```
MULTIPLY A BY B GIVING temp
MOVE temp TO C
MOVE temp TO D
```

ここで、*temp* は、インプリメンターによって提供された中間結果です。

Enterprise COBOL プログラミング・ガイド で中間結果の使用および定義について説明されています。中間結果は最高 30 桁である (ARITH(EXTEND) を指定すると 31 桁)、などの定義があります。

したがって、上記の例で、A、B、C、D がすべて PIC S9(18) と定義されている場合、A と B を乗算すると 36 桁の結果となりますが、30 桁 (または 31 桁) の中間結果 *temp* に移されます。その後、*temp* が C と D に移されます。

CMPR2

MULTIPLY ステートメントの例に SIZE ERROR が指定されている場合、COBOL 74 標準に従って、36 桁の (即時) 結果が 30 桁 (または 31 桁) の (中間) 結果に移されると SIZE ERROR が発生します。対応する COMPUTE の場合はこれとは異なり、36 桁の (即時) 結果が 30 桁 (または 31 桁) の (中間) 結果に移されても、SIZE ERROR は発生しません。

```
COMPUTE C D = A * B ON SIZE ERROR...
```

この動作は、DIVIDE ステートメントおよび対応する COMPUTE ステートメントにも適用されます。

NOCMPR2

しかし、NOCMPR2 のもとでは、SIZE ERROR は最終結果にだけ適用されます。MULTIPLY の例では、36 桁 (即時) 結果が 30 桁 (または 31) の (中間) 結果に移されても、SIZE ERROR は発生しません。したがって、この点については、MULTIPLY ステートメントと COMPUTE ステートメントは同等になります。この動作は DIVIDE ステートメントにも適用されます。

現在、このようなステートメントに対しては、以下のコンパイラー・メッセージが出されます。

IGYPG3113-W
: 中間結果の精度が 30 桁を超えていたため、上位桁が切り捨てられる可能性があります。

実行時に切り捨てが実際に行われた場合、以下のメッセージが出されます。

IGZ0036W
: プログラム "program-name" の行番号 "n" で、高位桁位置の切り捨てが発生しました。

メッセージ

この変更による影響を受ける可能性のあるプログラムの場合、CMPR2 および FLAGMIG オプションを指定してコンパイルすると、以下のメッセージが出されます。

IGYPG3204-W
: **MIGR** "NOCMPR2" コンパイラー・オプションのもとでは、中間結果に関して "ON SIZE ERROR" 句は実行されません。

MULTIPLY と DIVIDE の SIZE ERROR に対する修正処置:

広範囲にわたる複雑なコーディングを再び行わなければ、影響を受けるプログラムの CMPR2 での動作を保持することはできません。このようなプログラムは、書き直して、CMPR2 での動作に依存しないようにしてください。

UNSTRING オペランドの評価

以下の説明では、参照用に、以下の一般形式の UNSTRING ステートメントを使用します。

```
UNSTRING id-1
  DELIMITED BY id-2 OR id-3 ...
  INTO id-4 DELIMITER IN id-5 COUNT IN id-6
       id-7 DELIMITER IN id-8 COUNT IN id-9
  ⋮
  WITH POINTER id-10
  TALLYING IN id-11
  ON OVERFLOW imp-stmt-1
  NOT ON OVERFLOW imp-stmt-2
  END-UNSTRING
```

CMPR2

CMPR2 のもとでは、id-1、id-10、および id-11 に関連した添え字付け、指標付け、または長さ計算の評価は、UNSTRING ステートメントの実行の開始時に 1 回だけ行われます。しかし、id-2、id-3、id-4、id-5、id-6、id-7、id-8、および id-9 (また

は、これらの繰り返し) に関連した添え字付け、指標付け、または長さ計算の評価は、それぞれのデータ項目への転送の直前に行われます。

NOCMPR2

NOCMPR2 のもとでは、id-1 ～ id-11 までのいずれか (または、これらの繰り返し) に関連した添え字付け、指標付け、または長さの計算の評価は、UNSTRING ステートメントの実行の開始時に 1 回だけ行われます。この変更により、id-2 ～ id-9 までの間に依存関係がある場合は、異なる結果が生成される可能性があります。

注: id-1、id-10、および id-11 に関係のある依存関係は、この変更による影響を受けません。

メッセージ

3211 ～ 3214 までのメッセージでフラグが立てられた UNSTRING ステートメントのほとんどは、同じ結果を生成します。UNSTRING ステートメントのオペランド間に特定の依存関係がある場合にのみ、異なる結果を生成します。

たとえば、以下のような場合には、UNSTRING ステートメントの 2 つのオペランド (op-1 と op-2) の間に依存関係が存在する可能性があります。

1. op-1 が添え字付けされており、その添え字値は op-2 によって変更される。
   a. 添え字 ID が、INTO、DELIMITER IN、または COUNT IN オペランド内の受け取り側として使用されている。
   b. 添え字 ID は位置可変の項目であり、この項目の位置に影響を与える ODO オブジェクト が INTO、DELIMITER IN、または COUNT IN オペランドの受け取り側として使用されている。
2. op-1 が可変長のグループ項目であり、このグループの長さを左右する ODO オブジェクトは op-2 によって変更される。
   a. ODO オブジェクトが、INTO、DELIMITER IN、または COUNT IN オペランド内の受け取り側として使用されている。
3. op-1 は位置可変の項目であり、この項目の位置に影響を与える ODO オブジェクトが op-2 によって変更されている。
   a. ODO オブジェクトが、INTO、DELIMITER IN、または COUNT IN オペランド内の受け取り側として使用されている。

注: オペランドをオーバーラップすることによって、あるいは DELIMITED BY オペランドと同じ ID および送り出し側、INTO、または DELIMITER IN オペランドのいずれかと同じ ID を指定することによって生成された依存関係は、COBOL 74 標準と COBOL 85 標準のいずれでも不法であるため、ここでは説明しません。一般に、結果は予測できません。

CMPR2 および FLAGMIG オプションを指定してプログラムをコンパイルすると、すべての UNSTRING ステートメントに同様の依存性が存在する可能性があるため、コンパイラーがメッセージを出します。

これらのメッセージでフラグが立てられなかった UNSTRING ステートメントは、CMPR2 と NOCMPR2 のもとで同一の結果を生成します。

メッセージ 2222 でフラグが立てられた UNSTRING ステートメントは、変更して、同じ結果を生成するようにする必要があります。

UNSTRING OPERAND の評価に対する修正処置:

変更が必要なそれぞれの場合についての詳しい説明を、メッセージ番号順に以下に記載し、それと共に、依存関係および提案される変更を示す例を記載します。例には、プログラムの重要な部分だけを示しています。

IGYPS2222-W

このメッセージは、UNSTRING ステートメントの「受け取り側」ID の 1 つが可変長グループ項目を参照していて、この項目がそれ自身の ODO オブジェクトを含んでいる場合に出されます。すべての UNSTRING ステートメントに適用される構文規則および制約事項により、この状態は、id-2、id-3、id-4、id-5、id-7、および id-8 (または、これらの繰り返し) に対してのみ発生する可能性があります。

以下に例を示します。

```
01 VLG-1.
 02 VLG-1-ODOOBJ PIC 9 VALUE IS 5.
 02 VLG-1-GR.
  03 VLG-1-ODO PIC X OCCURS 1 TO 9 TIMES
                     DEPENDING ON VLG-1-ODOOBJ.
77 S-1 PIC X(20) VALUE IS ALL "123456789".

   UNSTRING S-1
            INTO VLG-1
            END-UNSTRING
```

IGYPS2222-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、受け取り側 "vlg-1" の最大長が使用されます。

注: Enterprise COBOL では、vlg-1 の最大長を使用して、送り出し項目 s-1 から抽出されたデータの量と受け取り区域 vlg-1 の長さの両方が決定されます。

メッセージ 2222 でどの ID にフラグが立てられたかに関係なく、以下のように、ID を参照変更バージョンで置き換える必要があります。

```
UNSTRING S-1
         INTO VLG-1(1:LENGTH OF VLG-1)
         END-UNSTRING
```

この形式では、UNSTRING ステートメントの実行の開始時での vlg-1 の実際の長さが必ず使用されます。

この修正は、UNSTRING ステートメントにオプショナルの句 (DELIMITED BY、WITH POINTER、ON OVERFLOW) があってもその影響を受けず、UNSTRING ステートメント内のフラグが立てられたすべての ID に同じように適用されます。

IGYPA3211-W

このメッセージは、UNSTRING ステートメント内の "DELIMITED BY" の ID の 1 つに添え字が指定されているか、可変長グループ項目を参照しているか、位置可変の項目を参照している場合に出されます。

この変更によって影響を受ける UNSTRING ステートメントの場合、フラグが立てられた DELIMITED BY オペランドは、INTO の受け取り側の 1 つに依存していなければなりません。

以下に例を示します。

```
01 DEL
 02 OCC-DEL-1 PIC X OCCURS 9 TIMES.
 02 VLEN-DEL-2-ODOOBJ PIC 9 VALUE IS 5.
 02 VLEN-DEL-2.
  03 VLEN-DEL-2-ODO PIC X OCCURS 1 TO 9 TIMES
                         DEPENDING ON VLEN-DEL-2-ODOOBJ.

77 S-1 PIC X(20) VALUE IS ALL "123456789".
77 R-1 PIC X(20) VALUE IS SPACES.
77 R-2 PIC X(20) VALUE IS SPACES.
77 SUB-5 PIC 99 VALUE IS 5.

   UNSTRING S-1
            DELIMITED BY OCC-DEL-1(SUB-5) OR VLEN-DEL-2,
            INTO R-1 DELIMITER IN OCC-DEL-1(SUB-5 + 1)
                     COUNT IN VLEN-DEL-2-ODOOBJ,
                 R-2,
            END-UNSTRING
```

IGYPA3211-W

MIGR この "UNSTRING" ステートメントでは、"DELIMITED BY" オペランドの添え字または "OCCURS DEPENDING ON" の計算は、"NOCMPR2" コンパイラー・オプションのもとでは 1 回だけ行われます。

メッセージ 3211 でフラグが立てられた項目は、修正する必要はありません。

IGYPA3212-W

このメッセージは、UNSTRING ステートメント内の INTO ID の 1 つに添え字が指定されているか、可変長グループ項目を参照しているか、位置可変の項目を参照している場合に出されます。

この変更の影響を受ける UNSTRING ステートメントの場合、フラグが立てられた INTO ID は、その前の INTO 句内の受け取り側に依存していなければなりません。

以下に例を示します。

```
01 REC.
 02 R-1 PIC X(20) VALUE IS SPACES.
 02 R-2-SUB PIC 9 VALUE IS 9.
 02 OCC-R-2-GR.
  03 OCC-R-2 PIC X OCCURS 9 TIMES.
 02 R-3-ODOOBJ PIC 9 VALUE IS 9.
 02 ODO-R-3.
  03 FILLER PIC X OCCURS 1 TO 9 TIMES
                  DEPENDING ON R-3-ODOOBJ.

77 S-3 PIC X(20) VALUE IS "12 345  6789   .....".

   UNSTRING S-3
            DELIMITED BY ALL SPACES,
            INTO R-1 COUNT IN R-2-SUB,
                 OCC-R-2(R-2-SUB) COUNT IN R-3-ODOOBJ,
                 ODO-R-3,
            END-UNSTRING
```

IGYPA3212-W

MIGR この "UNSTRING" ステートメントでは、"INTO" オペ

ランドの添え字または "OCCURS DEPENDING ON" の計算は、"NOCMPR2" コンパイラー・オプションのもとでは 1 回だけ行われます。

この UNSTRING ステートメントは、CMPR2 と NOCMPR2 とでは異なる結果を生成します。2 番目の INTO 受け取り側の添え字が最初の INTO 句の COUNT IN 受け取り側によって変更されるためです。さらに、3 番目の INTO 受け取り側の長さは、2 番目の INTO 句の COUNT IN 受け取り側によって変更されます。

CMPR2 のもとでは、COUNT IN の ID に移される値が後続の INTO 句に使用されます。NOCMPR2 のもとでは、UNSTRING ステートメントの実行の開始時に有効な値が、すべての INTO 句に使用されます。

メッセージ 3212 でフラグが立てられた UNSTRING ステートメントは、複数の UNSTRING ステートメントに分割しなければなりません。依存する INTO 句ごとに別々の UNSTRING ステートメントを使用してください。ただし、以下のことに注意してください。

- 元の UNSTRING ステートメントに WITH POINTER 句が指定されている場合は、変更後の UNSTRING ステートメントのすべてにこの句を含めなければなりません。元の UNSTRING ステートメントに WITH POINTER 句が指定されていない場合は、変更後の UNSTRING ステートメントのすべてにこの句を追加し、POINTER の ID を 1 に初期設定しなければなりません。
- 元の UNSTRING ステートメントに TALLYING IN 句が指定されている場合は、変更後の UNSTRING ステートメントのすべてにこの句を含めなければなりません。
- 元の UNSTRING ステートメントに ON OVERFLOW 句または NOT ON OVERFLOW 句が指定されている場合は、変更後の最後の UNSTRING ステートメントにだけこの句を含めなければなりません。

上記の例にこれらの変更を行うと、以下のようになります。

```
77 PTR PIC 99.

   MOVE 1 TO PTR
   UNSTRING S-3
            DELIMITED BY ALL SPACES,
            INTO R-1 COUNT IN R-2-SUB,
            WITH POINTER PTR,
            END-UNSTRING
   UNSTRING S-3
            DELIMITED BY ALL SPACES,
            INTO OCC-R-2(R-2-SUB) COUNT IN R-3-ODOOBJ,
            WITH POINTER PTR,
            END-UNSTRING
   UNSTRING S-3
            DELIMITED BY ALL SPACES,
            INTO ODO-R-3,
            WITH POINTER PTR,
            END-UNSTRING
```

IGYPA3213-W

このメッセージは、UNSTRING ステートメント内の DELIMITER IN の ID

の 1 つに添え字が指定されているか、可変長グループ項目を参照しているか、位置可変の項目を参照している場合に出されます。

この変更の影響を受ける UNSTRING ステートメントの場合、フラグが立てられた DELIMITER IN の ID は、その前の INTO 句内の受け取り側に依存していなければなりません。

注: 同じ INTO 句にある複数の ID の間の依存関係は、UNSTRING ステートメントの結果に影響を与えません。CMPR2 での動作では、これらの依存関係は次の INTO 句まで有効になりません。

以下に例を示します。

```
01 DEL.
 02 D-2-SUB PIC 9 VALUE IS 9.
 02 OCC-D-2-GR.
  03 OCC-D-2 PIC X OCCURS 9 TIMES.
 02 D-3-ODOOBJ PIC 9 VALUE IS 9.
 02 ODO-D-3.
  03 FILLER PIC X OCCURS 1 TO 9 TIMES
                DEPENDING ON D-3-ODOOBJ.

77 S-4 PIC X(20) VALUE IS "12 345  6789   .....".
77 R-1 PIC X(20) VALUE IS SPACES.
77 R-2 PIC X(20) VALUE IS SPACES.
77 R-3 PIC X(20) VALUE IS SPACES.

   UNSTRING S-4
            DELIMITED BY ALL SPACES,
            INTO R-1 COUNT IN D-2-SUB,
                 R-2 DELIMITER IN OCC-D-2(D-2-SUB)
                     COUNT IN D-3-ODOOBJ,
                 R-3 DELIMITER IN ODO-D-3,
            END-UNSTRING
```

IGYPA3213-W

MIGR この "UNSTRING" ステートメントでは、"DELIMITER IN" オペランドの添え字または "OCCURS DEPENDING ON" の計算は、"NOCMPR2" コンパイラー・オプションのもとでは 1 回だけ行われます。

この UNSTRING ステートメントは、CMPR2 と NOCMPR2 とでは異なる結果を生成します。2 番目の INTO 句の DELIMITER IN の ID の添え字が最初の INTO 句の COUNT IN 受け取り側によって変更されるためです。さらに、3 番目の INTO 句の DELIMITER IN の ID の長さは、2 番目の INTO 句の COUNT IN 受け取り側によって変更されます。

CMPR2 の動作では、COUNT IN の ID に移される値が後続の INTO 句に使用されます。NOCMPR2 のもとでは、UNSTRING ステートメントの実行の開始時に有効な値が、すべての INTO 句に使用されます。

メッセージ 3213 でフラグが立てられた UNSTRING ステートメントは、複数の UNSTRING ステートメントに分割しなければなりません。依存する INTO 句ごとに別々の UNSTRING ステートメントを使用してください。

上記の例にこれらの変更を行うと、以下のようになります。

```
77 PTR PIC 99.
   MOVE 1 TO PTR
   UNSTRING S-4
            DELIMITED BY ALL SPACES,
            INTO R-1 COUNT IN D-2-SUB,
            WITH POINTER PTR,
            END-UNSTRING
   UNSTRING S-4
            DELIMITED BY ALL SPACES,
            INTO R-2 DELIMITER IN OCC-D-2(D-2-SUB)
                     COUNT IN D-3-ODOOBJ,
            WITH POINTER PTR,
            END-UNSTRING
   UNSTRING S-4
            DELIMITED BY ALL SPACES,
            INTO R-3 DELIMITER IN ODO-D-3,
            WITH POINTER PTR,
            END-UNSTRING
```

IGYPA3214-W

このメッセージは、UNSTRING ステートメント内の COUNT IN の ID の 1 つに添え字が指定されているか、位置可変の項目を参照している場合に出されます。

この変更の影響を受ける UNSTRING ステートメントの場合、フラグが立てられた COUNT IN の ID は、その前の INTO 句内の受け取り側に依存していなければなりません。

注: 同じ INTO 句にある複数の ID の間の依存関係は、UNSTRING ステートメントの結果に影響を与えません。CMPR2 での動作では、これらの依存関係は次の INTO 句まで有効になりません。

以下に例を示します。

```
01 C-2.
 02 C-2-SUB PIC 9 VALUE IS 9.
 02 OCC-C-2-GR.
  03 OCC-C-2 PIC 9 OCCURS 9 TIMES.

77 S-5 PIC X(20) VALUE IS "12 345  6789........".
77 R-1 PIC X(20) VALUE IS SPACES.
77 R-2 PIC X(20) VALUE IS SPACES.

   UNSTRING S-5
            DELIMITED BY ALL SPACES,
            INTO R-1 COUNT IN C-2-SUB,
                 R-2 COUNT IN OCC-C-2(C-2-SUB),
            END-UNSTRING
```

IGYPA3214-W

MIGR この "UNSTRING" ステートメントでは、"COUNT IN" オペランドの添え字または "OCCURS DEPENDING ON" の計算は、"NOCMPR2" コンパイラー・オプションのもとでは 1 回だけ行われます。

この UNSTRING ステートメントは、CMPR2 と NOCMPR2 とでは異なる結果を生成します。2 番目の INTO 句の COUNT IN の ID の添え字が最初の INTO 句の COUNT IN 受け取り側によって変更されるためです。

CMPR2 での動作では、最初の INTO 句の COUNT IN の ID に移される値が、2 番目の INTO 句に使用されます。NOCMPR2 のもとでは、UNSTRING ステートメントの実行の開始時に有効な値が使用されます。

メッセージ 3214 でフラグが立てられた UNSTRING ステートメントは、複数の UNSTRING ステートメントに分割しなければなりません。依存する INTO 句ごとに別々の UNSTRING ステートメントを使用してください。

上記の例にこれらの変更を行うと、以下のようになります。

```
77 PTR PIC 99.

   MOVE 1 TO PTR
   UNSTRING S-5
            DELIMITED BY ALL SPACES,
            INTO R-1 COUNT IN C-2-SUB,
            WITH POINTER PTR,
            END-UNSTRING
   UNSTRING S-5
            DELIMITED BY ALL SPACES,
            INTO R-2 COUNT IN OCC-C-2(C-2-SUB),
            WITH POINTER PTR,
            END-UNSTRING
```

UPSI スイッチ

CMPR2

UPSI スイッチは、スイッチの ON および OFF の設定に条件名を指定することによって定義することができます。CMPR2 のもとでは、すべての UPSI スイッチ (UPSI-0 ～ UPSI-7 まで) の条件名を、以下のように、同じ名前で定義することができます。

```
SPECIAL-NAMES.
    UPSI-0  ON STATUS IS T  OFF STATUS IS F
    UPSI-1  ON STATUS IS T  OFF STATUS IS F
   ⋮
    UPSI-7  ON STATUS IS T  OFF STATUS IS F
```

名前に対する参照は、以下のように、UPSI 名を使用して修飾することができます。

```
IF T OF UPSI-0 DISPLAY "UPSI-0".
IF T OF UPSI-1 DISPLAY "UPSI-1".
   ⋮
IF T OF UPSI-7 DISPLAY "UPSI-7".
```

NOCMPR2

NOCMPR2 のもとでは、UPSI スイッチ (UPSI-0 ～ UPSI-7 まで) の名前を PROCEDURE DIVISION (手続き部) で参照できなくなりました。現在、上記のステートメントに対しては、以下の形式のメッセージが出されます。

IGYPS2121-S

"T OF UPSI-0" はデータ名として定義されていませんでした。このステートメントは無視されました。

メッセージ

CMPR2 および FLAGMIG を使用すると、UPSI スイッチを名前で参照している PROCEDURE DIVISION (手続き部) のステートメントに対して、以下のメッセージが出されます。

IGYPS0186-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、UPSI スイッチを手続き部で直接参照することはできません。

UPSI スイッチに対する修正処置:

プログラムを変更して、以下のように固有の条件名を定義し、

```
SPECIAL-NAMES.
  UPSI-0  ON STATUS IS T0  OFF STATUS IS F0
  UPSI-1  ON STATUS IS T1  OFF STATUS IS F1
   ⋮
  UPSI-7  ON STATUS IS T7  OFF STATUS IS F7
```

以下のように新しい条件名を参照する必要があります。

```
IF T0 DISPLAY "UPSI-0".
IF T1 DISPLAY "UPSI-1".
   ⋮
IF T7 DISPLAY "UPSI-7".
```

可変長グループ移動

CMPR2

グループ移動 (MOVE ステートメントなど) に関係する送り出しフィールドおよび受け取りフィールド内の ODO オブジェクトはすべて、そのステートメントが実行される前に設定されなければなりません。送り出し側および受け取り側の実際の長さは、データ移動ステートメントが実行される直前に計算されます。影響を受ける動詞のリストについては、以下のメッセージを参照してください。

NOCMPR2

受け取り側が可変長グループである場合、CMPR2 では実際の長さを使用しますが、NOCMPR2 では可変長グループの最大長を使用する場合があります。この動作は、受け取り側が可変長であり、それ自身の ODO オブジェクトを含んでおり、構造内の最後のグループである場合に行われます。以下に例を示します。

```
01  ODO-SENDER
    02  SEND-OBJ PIC 99.
    02  SEND-ITEM PIC X OCCURS 1 TO 20 DEPENDING ON SEND-OBJ.

01  ODO-RECEIVER.
    02  RECV-OBJ PIC 99.
    02  RECV-ITEM PIC X OCCURS 1 TO 20 DEPENDING ON RECV-OBJ.
   ⋮
MOVE 5 TO SEND-OBJ.
MOVE 10 TO RECV-OBJ.
MOVE ODO-SENDER TO ODO-RECEIVER.
   ⋮
CMPR2:
   Occurrences 1-5 of ODO-SENDER moved to ODO-RECEIVER.
   Occurrences 6-10 of ODO-RECEIVER become spaces.
   Occurrences 11-20 of ODO-RECEIVER are unchanged.
```

```
NOCMPR2:
   Occurrences 1-5 of ODO-SENDER moved to ODO-RECEIVER.
   Occurrences 6-20 of ODO-RECEIVER become spaces.
```

データ移動ステートメントの実行時に ODO オブジェクトの値を超えるテーブル・オカレンス数を参照するプログラムは、NOCMPR2 のもとで使用されると正しい結果を生成しません。

上記の例では、グループ移動の前にオカレンス 11 〜 20 にデータがあっても、NOCMPR2 のもとで実行されると、グループ移動後にデータが消失します。

メッセージ

CMPR2 および FLAGMIG コンパイラー・オプションを指定してプログラムをコンパイルすると、NOCMPR2 のもとでは動作が異なるそれぞれのデータ移動ステートメントごとに以下のメッセージが生成されます。

IGYPS2222-W

MIGR "NOCMPR2" コンパイラー・オプションのもとでは、受け取り側 "ODO-RECEIVER" の最大長が使用されます。

可変長グループ移動におけるこの違いは、データを移動する動詞すべてに影響を与えます。影響のある動詞は次のとおりです。

ACCEPT identifier (形式 1 または形式 2)
MOVE . . . TO identifier
READ . . . INTO identifier
RELEASE identifier FROM . . .
RETURN . . . INTO identifier
REWRITE identifier FROM . . .
STRING . . . INTO identifier
UNSTRING . . . INTO identifier DELIMITER IN identifier
WRITE identifier FROM . . .

可変長グループ移動に対する修正処置:

手順を以下に示します。

- CMPR2 および FLAGMIG コンパイラー・オプションを指定してコンパイルすることによって、COBOL プログラムに可変長データを移動するステートメントが含まれているかどうか調べてください。コンパイルが完了すると、それ自身の ODO オブジェクトを含んでいて、複雑な ODO 項目ではない受け取り側を使用するすべての可変長グループ移動にフラグが設定されます。
- 以前は未変更のままであったが、現在はブランクに設定されるデータが、データ移動ステートメントの後で参照されているかどうかを調べてください。上記の例で、ODO オブジェクトの値が 5 で、最大値が 10 であり、MOVE の後でオカレンス番号 6 〜 10 までのデータを使用するようにコーディングされていると、CMPR2 と NOCMPR2 とでプログラムの結果が異なります。
- データ移動ステートメントの受け取り側を変更して、参照変更を使用して受け取りフィールドの長さを明示的に指定してください。以下に例を示します。

  ```
  MOVE ODO-SENDER TO ODO-RECEIVER (1:LENGTH OF ODO-RECEIVER).
  ```

第 17 章 COBOL ソースに関する CICS の移行の考慮事項

この章では、CICS で実行されるプログラムについてのソース言語の考慮事項を説明します。ここでは、CICS ソースまたは OS/VS COBOL ソースのいずれかを使用し、以下の機能を含むアプリケーションに対して行う必要のある処置について説明します。

- CICS で実行されるプログラム用の主要なコンパイラー・オプション
- 単独の CICS トランスレーターから組み込みの CICS トランスレーターへのマイグレーション
- OS/VS COBOL プログラムについての基底アドレス可能度の考慮事項

CICS ランタイムの考慮事項は、以下の章に記載されています。

- 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』
- 87 ページの『第 7 章 VS COBOL II ランタイムからの移行』

CICS Transaction Server バージョン 1 リリース 3 以上では、CICS 環境は、Enterprise COBOL および言語環境プログラムの使用をサポートします。したがって、CICS コマンド・レベル・アプリケーション・プログラムをコンパイルおよび実行することができます。(CICS マクロ・レベル・プログラムは、Enterprise COBOL でコンパイルすることも、言語環境プログラムで実行することもできません。この移行については、CICS アプリケーション・マイグレーション・エイドが役立ちます。詳細については、 300 ページの『CICS アプリケーション・マイグレーション・エイド』を参照してください。)

将来の考慮事項

CICS で実行される OS/VS COBOL プログラムには、言語環境プログラムのサポートだけでなく CICS プロダクトによる特殊なサポートが必要です。CICS Transaction Server バージョン 2 リリース 2 以降の CICS Transaction Server リリースでは、この特殊サポートは使用不可能です。OS/VS COBOL プログラムは、CICS Transaction Server バージョン 2 リリース 2 以降では COBOL ランタイム・ライブラリーとして言語環境プログラムを使用しても CICS のもとでは実行されません。できるだけ早くすべての OS/VS COBOL プログラムを Enterprise COBOL にアップグレードする必要があります。

CICS で実行されるプログラム用の主要なコンパイラー・オプション

238 ページの表 45 には、CICS で実行される Enterprise COBOL プログラム用の主要なコンパイラー・オプションがリストされています。

CICS ソースの考慮事項

表 45. CICS で実行されるプログラム用の主要なコンパイラー・オプション

コンパイラー・オプション	説明
CICS	CICS コンパイラー・オプションは、組み込みの CICS トランスレーター機能を使用可能にします。ソース・プログラムに CICS ステートメントが含まれており、単独の CICS トランスレーターによってまだ処理されていない場合、CICS オプションを指定する必要があります。 CICS オプションを指定する場合は、LIB、NODYNAM、および RENT オプションも有効にする必要があります。Enterprise COBOL では、NOLIB、DYNAM、または NORENT が CICS オプションと同じレベルで指定された場合、これらのオプションが強制的に指定されます。 NOCICS オプションを指定すると、ソース・プログラムで検出された CICS ステートメントはすべて廃棄されます。
DATA	以下の 2 つのケースでは、DATA(24) を使用してください。 • CICS 共用データベース・サポート (DFHDRP) を使用するバッチ・プログラムの場合、CBLTDLI に渡されるパラメーターは 16MB 境界より下になければなりません。 • CALL CBLTDLI を使用してローカル DL/I データベースにアクセスする CICS オンライン・プログラムの場合、DL/I 呼び出しのパラメーター・リストと、パラメーター・リスト内で参照されるすべてのストレージ域は、16MB 境界より下になければなりません。 CICS/ESA の場合、CALL CBLTDLI を使用してローカル DL/I データベースにアクセスする CICS オンライン・プログラムを含んでいるトランザクションについての TRANSACTION 定義で、TASKDATALOC(BELOW) を指定しなければなりません。 **注:** DBCTL を使用しているときは、DATA(24) は必要ありません。詳細については、*CICS-IMS Database Control Guide for CICS/ESA* を参照してください。
LIB	CICS トランスレーターによって変換されるプログラムの場合、LIB が必要です。
NODYNAM	CICS トランスレーターによって変換されるプログラムの場合、NODYNAM が必要です。これは、CICS コマンド・レベル・スタブを動的に呼び出すことができないためです。
RENT	CICS プログラムの場合、RENT が必要です。RENT を指定すると、コンパイラーは再入可能コードを生成するため、COBOL モジュールを LPA (リンク・パック域) または ELPA (拡張リンク・パック域) に置くことができるので、それを CICS のもとで複数のアドレス・スペースで共用することができます。また、LPA/ELPA にはストレージ保護キーがあるため、モジュールを上書きすることはできません。
TRUNC	EXEC CICS コマンドを含んでいる CICS プログラムの場合に、プログラムによるバイナリー・データ項目の使用法がそのデータ項目についての PICTURE および USAGE 文節に従っている場合は、TRUNC(OPT) を使用してください。 プログラムによるバイナリー・データ項目の使用法がそのデータ項目についての PICTURE および USAGE 文節に従っていない場合は、TRUNC(BIN) を使用してください。たとえば、データ項目が PIC S9(8) BINARY として定義され、それが 8 桁よりも大きい値を受け取る可能性がある場合は TRUNC(BIN) を使用してください。

単独の CICS トランスレーターから組み込みのトランスレーターへのマイグレーション

多くの OS/VS COBOL CICS プログラムは、Enterprise COBOL でコンパイルするために変更する必要があります。CCCA ツールを使用して、これらの変更をすべて自動的に行うことができます。あるいは、ここに記載された情報を使用してご自分で変更を行うこともできます。CCCA の詳細については、293 ページの『付録 C. ソース・プログラム用の移行ツール』を参照してください。

以下の考慮事項は、組み込みの CICS トランスレーターを使用できるように COBOL アプリケーションをマイグレーションする場合に適用されます。

- コンパイル・プロセスから単独の変換ステップを削除する。
- XOPTS トランスレーター・オプションを CICS コンパイラー・オプションに変更する。サブオプション・ストリングは引用符またはアポストロフィで区切る必要があります。たとえば、単独の CICS トランスレーターによって変換するプログラムには、以下のような CBL ステートメントを含める必要があります。

  ```
  CBL TEST(NONE,SYM), XOPTS(LINKAGE,SEQ,SP)
  ```

  組み込みの CICS トランスレーターの場合、以下のように変更する必要があります。

  ```
  CBL TEST(NONE,SYM), CICS('LINKAGE,SEQ,SP')
  ```
- CICS ステートメントを含むプログラムをコンパイルするときは、SIZE(MAX) を使用しないでください。ストレージはトランスレーター・サービス用にユーザー領域に残しておく必要があります。
- すべての CBL/PROCESS ステートメントをソース・プログラムの先頭行に移動する。組み込みの CICS トランスレーターは、CBL/PROCESS ステートメントの前にあるコメント行を受け入れません。ソース・プログラムは Enterprise COBOL 規則に準拠する必要があります。
- DFHCOMMAREA を再定義するネストされたプログラムがあるかどうか調べてください。組み込みトランスレーターはネストされたプログラムで DFHCOMMAREA または DFHEIBLK の宣言を生成しません。DFHCOMMAREA および DFHEIBLK 宣言は、指定された GLOBAL 属性を使用して最外部のプログラムで生成されます。ネストされたプログラム内で生成されたこれらの宣言に依存する COBOL プログラムは、ソースを変更する必要があります。

組み込みの CICS トランスレーター

組み込みトランスレーターは、CICS ステートメントを含む COBOL プログラム用の単独変換ステップを除去します。

組み込みトランスレーターでは、COBOL コンパイラーでは、ソース・プログラム内のネイティブ COBOL ステートメントと組み込み CICS ステートメントの両方を処理できます。コンパイラーは、CICS ステートメントが検出された場合、組み込みの CICS トランスレーターとインターフェースをとります。組み込みの CICS トランスレーターは適切な処置を行ってから、生成するネイティブ言語ステートメントを指示してコンパイラーに制御を戻します。

単独の CICS トランスレーターは依然として Enterprise COBOL でサポートされますが、組み込みの CICS トランスレーターを使用することをお勧めします。組み込みの CICS トランスレーターは使用可能度を向上させ、最高レベルの機能性を実現します。組み込みの CICS トランスレーターを使用する利点は以下のとおりです。

- デバッグ・ツールを使用した COBOL アプリケーションの対話式デバッグ機能が強化される。CICS トランスレーターによって生成された拡張ソースのレベルではなく、元のソース・レベルでアプリケーションをデバッグすることができます。
- EXEC CICS ステートメントや EXEC DLI ステートメントはコピーブックに入れておくことができ、コンパイル前に外部トランスレーターを使用して変換する必要がない。
- ソース・プログラムの変換済みバージョンを格納する (プログラムがコンパイルされる前に) 中間データ・セットが不要である。
- 出力リストは 2 つではなく 1 つだけである。
- EXEC CICS ステートメントの入っているネストされたプログラムの使用が簡単である。DFHCOMMAREA および DFHEIBLK は、ネストされたプログラムの PROCEDURE DIVISION USING に指定された GLOBAL 属性を使用して最外部のプログラムで生成されます。
- EXEC CICS ステートメントの入っているネストされたプログラムを個別のファイルに保管し、COPY ステートメントを使ってこれをインクルードすることができる。
- REPLACE ステートメントを EXEC CICS ステートメントに影響させることができる。
- CICS 制御ブロック内の 2 進数フィールドは、BINARY ではなく USAGE COMP-5 を使って生成される。したがって、TRUNC コンパイラー・オプションとの依存関係がなくなりました。TRUNC オプションの設定はいずれも組み込みトランスレーターを使用する CICS アプリケーションで使用でき、アプリケーション内部のユーザー作成論理の要件にのみ従います。
- CBL/PROCESS ステートメントの前にあるコメント行が受け入れられる。

組み込みの CICS トランスレーター用の主要なコンパイラー・オプション

表 46 には、組み込みの CICS トランスレーターを使用する Enterprise COBOL プログラム用のコンパイラー・オプションがリストされています。

表 46. 組み込みの CICS トランスレーター用の主要なコンパイラー・オプション

コンパイラー・オプション	説明
CICS	CICS コンパイラー・オプションは、組み込みの CICS トランスレーター機能を使用可能にします。ソース・プログラムに CICS ステートメントが含まれており、組み込みの CICS トランスレーターによってまだ処理されていない場合、CICS オプションを指定する必要があります。 CICS オプションを指定する場合は、LIB、NODYNAM、および RENT オプションも有効にする必要があります。Enterprise COBOL では、NOLIB、DYNAM、または NORENT が CICS オプションと同じレベルで指定された場合、これらのオプションが強制的に指定されます。 NOCICS オプションを指定すると、ソース・プログラムで検出された CICS ステートメントはすべて廃棄されます。
LIB	LIB トランスレーターによって変換されるプログラムの場合、LIB が必要です。
NODYNAM	CICS トランスレーターによって変換されるプログラムの場合、NODYNAM が必要です。これは、CICS コマンド・レベル・スタブを動的に呼び出すことができないためです。
RENT	CICS プログラムの場合、RENT が必要です。RENT を指定すると、コンパイラーは再入可能コードを生成するため、COBOL モジュールを LPA (リンク・パック域) または ELPA (拡張リンク・パック域) に置くことができるので、それを CICS のもとで複数のアドレス・スペースで共用することができます。また、LPA/ELPA にはストレージ保護キーがあるため、モジュールを上書きすることはできません。

OS/VS COBOL プログラムについての基底アドレス可能度の考慮事項

OS/VS COBOL を使用する場合は、COBOL CICS プログラムの WORKING-STORAGE SECTION に含まれていないストレージ域へのアドレス可能度を保守する必要があります。プログラム要求を満たすため、OS/VS COBOL プログラムでは、アプリケーション・プログラム内で LINKAGE-SECTION (BLL) セル用のベース・ロケーターを操作するために CICS によって割り振られたストレージ域アドレスを記録しておかなければなりません。

Enterprise COBOL (および CICS 内の関連サポート) を使用する場合は、この操作を行う必要がなくなりました。したがって、OS/VS COBOL から Enterprise COBOL へのアップグレード時には、そのようなプログラムを移行する必要があります (COBOL CICS プログラムが BLL セルのアドレッシングを操作しない場合は、移行は必要ありません)。

注: この移行は、 293 ページの『付録 C. ソース・プログラム用の移行ツール』で説明されているように、CCCA を使用して自動的に行うことができます。

BLL セルを操作するプログラムの場合に変更が必要とされる 3 つの領域は、次のとおりです。

- SERVICE RELOAD ステートメント
- LENGTH OF 特殊レジスター
- BLL セルを使用するプログラム

SERVICE RELOAD ステートメント

OS/VS COBOL では、CICS のもとで実行されるプログラムの場合、LINKAGE SECTION で定義された項目のアドレス可能度を確保するために SERVICE RELOAD ステートメントが必要です。

Enterprise COBOL では、SERVICE RELOAD ステートメントは必要ありません。SERVICE RELOAD ステートメントが検出されると、それはコメントとして扱われます。

LENGTH OF 特殊レジスター

LENGTH OF 特殊レジスターを使用すると、長さが必要とされる CICS コマンドの多くで明示的な長さ引き数を渡す必要がなくなります。COBOL ステートメントでは、LENGTH OF 特殊レジスターを、明示的に定義された数値データ項目であるかのように使用することができます。これによって、あるデータ項目がプログラム内で占有する文字の数についての情報を入手することができます。

BLL セルを使用するプログラム

以下のステップは、OS/VS COBOL CICS プログラムから Enterprise COBOL CICS プログラムへの移行を要約したものです。詳しくは、*CICS/ESA アプリケーション・プログラミング・ガイド* を参照してください。

1. すべての SERVICE RELOAD ステートメントを削除します。Enterprise COBOL コンパイラーはこれらのステートメントをコメントとして扱います (削除するのが望ましいですが、必須ではありません)。
2. LINKAGE SECTION 内のサイズが 4K バイトよりも大きい構造のアドレッシングを扱うすべての操作を削除します。代表的なステートメントは次のとおりです。

   ```
   ADD +4096 D-PTR1 GIVING D-PTR2.
   ```

3. サイズが 4K よりも大きい COMMAREA のアドレッシングをサポートするすべてのプログラム・コードを削除します。
4. CICS プログラムが正しく最適化されるようにするために、OS/VS COBOL が使用する冗長な割り当ておよびラベルを削除します (これは、望ましいプログラミング手法ですが、必須ではありません)。

   冗長な割り当ておよびラベルには、以下のものがあります。

   - 人工的な段落名 (チェーン・ストレージ域をアドレッシングするために BLL セルを使用するもの)
   - OCCURS DEPENDING ON 文節のオブジェクトからそれ自体への人工的な割り当て
5. CICS コマンド内のすべての SET(P) オプションを SET(ADDRESS OF L) に変更します。ここで、“L” は、“P” BLL セルに対応する LINKAGE SECTION 構造です。
6. SET オプションを用いて複数のレコード・フォーマットが定義されている場合は、LINKAGE SECTION で REDEFINES 文節を指定します。
7. LINKAGE SECTION 内で基本マッピング・サポート (BMS) データ構造を使用しているプログラムを調べます (STORAGE=AUTO として定義されていないマッ

プがないか調べます)。STORAGE=AUTO として定義されていないすべてのマップを WORKING-STORAGE セクションに移動し、関連した EXEC CICS GETMAIN コマンドを削除します。

DL/I 呼び出しインターフェース

プログラムを移行して、DL/I 呼び出し用の BLL セルの使用も回避することができます。これを行うためには、プログラムを調べて、以下の変更を行ってください。

1. ユーザー・インターフェース・ブロック (UIB) およびプログラム連絡ブロック (PCB) をアドレッシングするための BLL セルを削除します。
2. LINKAGE SECTION で、DLIUIB 宣言と少なくとも 1 つの PCB 宣言を保持します。
3. PCB 呼び出しを、以下のように UIB を直接指定するように変更します。

```
CALL "CBLTDLI" USING PCB-CALL,
                     PSB-NAME,
                     ADDRESS OF DLIUIB
```

4. UIB 内のアドレス・リストから、必要な PCB のアドレスを入手します。

以下の例は、PCB のアドレスを取得して PCB1-ADDR および PCB2-ADDR に入れる方法を示しています。

```
 LINKAGE SECTION.
     COPY DLIUIB.
 01  OVERLAY-DLIUIB REDEFINES DLIUIB.
     02  PCBADDR USAGE IS POINTER.
     02  FILLER  PIC XX.

 01  PCB-ADDRESSES.
     02  PCB1-ADDR    USAGE IS POINTER.
     02  PCB2-ADDR    USAGE IS POINTER.

* VACATION PCB
 01  VAC-PCB.
 COPY RDLICVP.

* HOTEL PCB
 01 HTL-PCB.
 COPY RDLICHT.
⋮
       PROCEDURE DIVISION.
⋮
     CALL 'CBLTDLI' USING PCB
                          PSBNAME
                          ADDRESS OF DLIUIB

     IF UIBFCTR IS NOT EQUAL TO LOW-VALUES
         MOVE 'PCB CALL FAILED.' TO MSG-TEXT
         PERFORM WRITE-MESSAGE-AND-AMEND
     ELSE
         SET ADDRESS OF PCB-ADDRESSES TO PCBADDR
         SET ADDRESS OF HTL-PCB TO PCB1-ADDR
         SET ADDRESS OF VAC-PCB TO PCB2-ADDR
     END-IF
⋮
```

注: 上記のコードで、行 ″SET ADDRESS OF PCB-ADDRESSES TO PCBADDR″ から行 ″SET ADDRESS OF VAC-PCB TO PCB2-ADDR″ までが、必要な PCB のアドレスを UIB 内のアドレス・リストから取得するコマンドです。

例 1: 連絡域の受け取り

この例では、COBOL プログラムは LINKAGE SECTION 内でレコード域を定義しています。この手法は、WORKING-STORAGE SECTION の外側で構造を定義する多くの COBOL CICS プログラムで使用されています。

移行時には、以下のステップを実行してください。

1. BLL セルを定義しているアドレス・リスト定義を削除します。Enterprise COBOL では、BLL セルは LINKAGE SECTION 内で明示的に定義されなくなりました。
2. CICS コマンドの SET オプションでストレージ域を参照するときは、BLL セル名を指定するのではなく、ADDRESS OF 特殊レジスターを使用します。

OS/VS COBOL

```
LINKAGE SECTION.
01  PARAMETER-LIST.
  05  PARM-FILLER      PIC S9(8) COMP.
  05  PARM-AREA1-PTR   PIC S9(8) COMP.
  05  PARM-AREA2-PTR   PIC S9(8) COMP.
01  AREA1.
   05  AREA1-DATA        PIC X(100).
01  AREA2.
   05  AREA2-DATA        PIC X(100).
          .
          .
  PROCEDURE DIVISION.
          .
          .
    EXEC CICS READ DATASET("INFILE")
       RIDFLD(INFILE-KEY)
       SET(PARM-AREA1-PTR)
       LENGTH(RECORD-LEN)
    SERVICE RELOAD PARM-AREA1-PTR.
```

Enterprise COBOL

```
LINKAGE SECTION.

01  AREA1.
       05  AREA1-DATA    PIC X(100).
01  AREA2.
       05  AREA2-DATA    PIC X(100).
          .
          .
 PROCEDURE DIVISION.
          .
          .
 EXEC CICS READ DATASET("INFILE")
     RIDFLD(INFILE-KEY)
 SET(ADDRESS OF AREA1)
     LENGTH(RECORD-LEN).
```

例 2: 4K を超えるストレージ域の処理

OS/VS COBOL では、LINKAGE SECTION 区域が 4096 バイトよりも大きい場合は、ストレージ域全体へのアドレス可能度を提供するためのステートメントを組み込む必要があります。Enterprise COBOL プログラムの場合、これらのステートメントは必要ではありません。

以下の例は、OS/VS COBOL プログラムと Enterprise COBOL プログラムの両方のコーディングを示しています。

移行時には、以下のステップを実行してください。

1. アドレス可能度を保守するために使用された以下のステートメントを削除します。

   ```
   ADD +4096 TO RECORD-POINTER ...
   SERVICE RELOAD ...
   ```

2. CICS コマンドの SET オプションを、中間 BLL セルから、レコードに関連した ADDRESS OF 特殊レジスターに変更します。

OS/VS COBOL

```
LINKAGE SECTION.
 01 PARMLIST.
    .
    .
    05  RECORD-POINTERA  PIC S9(8) COMP.
    05  RECORD-POINTERB  PIC S9(8) COMP.
    .
    .
 01 FILE-RECORD.
    05  REC-AREA1         PIC X(2500).
    05  REC-AREA2         PIC X(2500).
    .
    .
PROCEDURE DIVISION.
    .
    .
    EXEC CICS READ DATASET("INFILE")
       RIDFLD(INFILE-KEY)
       SET(RECORD-POINTERA)
       LENGTH(RECORD-LEN)
    END-EXEC
    SERVICE RELOAD RECORD-POINTERA
    ADD +4096 TO RECORD-POINTERA
        GIVING RECORD-POINTERB
    SERVICE RELOAD RECORD-POINTERB.
```

Enterprise COBOL

```
LINKAGE SECTION.

01 FILE-RECORD.
   05  REC-DATA1       PIC X(2500).
   05  REC-DATA2       PIC X(2500).
       .
       .
PROCEDURE DIVISION.
       .
       .
    EXEC CICS READ DATASET("INFILE")
         RIDFLD(INFILE-KEY)
         SET(ADDRESS OF FILE-RECORD)
         LENGTH(RECORD-LEN)
    END-EXEC
```

例 3: チェーン・ストレージ域へのアクセス

OS/VS COBOL CICS プログラムでは、別のストレージ域を指すポインターを含んでいるストレージ域を LINKAGE SECTION で定義することによって、チェーン・ストレージ域にアクセスします。その後、次のチェーン区域のアドレスを関連 BLL にコピーすることによって、次のチェーン区域にアクセスします。さらに、段落名と、チェーン区域をアドレッシングするために使用される BLL セルの内容を変更するステートメントをコーディングすることも必要です。Enterprise COBOL では、このチェーニングは SET ステートメントを使用してチェーン区域のアドレスを設定することにより単純化されます。

移行時には、以下のステップを実行してください。

1. 次のアドレスをストレージ域から関連 BLL セルに移動するコードを変更します。Enterprise COBOL では、現行および次のストレージ域に関連した ADDRESS OF 特殊レジスターを使用することによって、同じ機能を実行してください。
2. 必要に応じて、BLL セルの内容を変更する参照の前にあるダミー段落名を削除します (これは、望ましいプログラミング手法ですが、必須ではありません)。

OS/VS COBOL

```
WORKING-STORAGE SECTION.
01 WSDATA-HOLD PIC X(100).
    .
    .
LINKAGE SECTION.
01 PARAMETER-LIST.
    .
    .
    05  CHAINED-POINTER PIC S9(8) COMP.
    .
    .
01 CHAINED-STORAGE.
    05  CHS-NEXT.       PIC S9(8) COMP.
    05  CHS-DATA        PIC X(100).
    .
    .
PROCEDURE DIVISION.
    .
    .
   MOVE CHS-NEXT. TO CHAINED-POINTER.
ANY-PARAGRAPH-NAME.
    MOVE CHS-DATA TO WSDATA-HOLD.
    .
    .
```

Enterprise COBOL

```
WORKING-STORAGE SECTION.
01 WSDATA-HOLD PIC X(100).
      .
      .
LINKAGE SECTION.
      .
      .
      .
     .
      .
      .
01 CHAINED-STORAGE.
    05 CHS-NEXT.    USAGE IS POINTER.
   05 CHS-DATA      PIC X(100)
      .
      .
PROCEDURE DIVISION.
      .
      .
  SET ADDRESS OF CHAINED-STORAGE TO CHS-NEXT.

    MOVE CHS-DATA TO WS-DATA-HOLD.
         .
         .
```

例 4: OCCURS DEPENDING ON 文節の使用

OS/VS COBOL では、LINKAGE SECTION が OCCURS DEPENDING ON 文節のオブジェクトを含んでいて、OCCURS 文節のサブジェクトのオカレンスの数が変わるときは、OCCURS DEPENDING ON 文節のオブジェクトをリセットして、グループの長さの更新を起動する必要があります。Enterprise COBOL では、構造が参照されるときは必ずグループの長さが計算されるため、このリセットは必要ありません。

移行時には、Enterprise COBOL プログラムで、OCCURS DEPENDING ON 文節のオブジェクトの内容をリセットするコードを除去してください (このような参照は必要なくなりました)。

OS/VS COBOL

```
LINKAGE SECTION.
  01 PARMLIST.
     05  FILLER          PIC S9(8).
     05  RECORD-POINTER  PIC S9(8).
     .
     .
  01  VAR-RECORD.
    05  REC-OTHER-DATA   PIC X(30).
    05  REC-AMT-CNT      PIC 9(4).
    05  REC-AMT          PIC 9(5)
          OCCURS 1 TO 100 TIMES
            DEPENDING ON REC-AMT-CNT.
     .
     .
PROCEDURE DIVISION.
     .
     .
     EXEC CICS READ DATASET("INFILE")
         RIDFLD(INFILE-KEY)
         SET(RECORD-POINTER)
         LENGTH(RECORD-LEN)
     END-EXEC.
    MOVE REC-AMT-CNT TO REC-AMT-CNT.
    MOVE VAR-RECORD TO WS-RECORD-HOLD.
     .
     .
```

Enterprise COBOL

```
LINKAGE SECTION.
01 VAR-RECORD.
 05  REC-OTHER-DATA     PIC X(30).
 05  REC-AMT-CNT        PIC 9(4).
 05  REC-AMT            PIC 9(5)
       OCCURS 1 TO 100 TIMES
            DEPENDING ON REC-AMT-CNT.
 .
 .
PROCEDURE DIVISION.
 .
 .
 EXEC CICS READ DATASET("INFILE")
      RIDFLD(INFILE-KEY)
      SET(ADDRESS OF VAR-RECORD)
              LENGTH(RECORD-LEN)
 END-EXEC.
 .
MOVE VAR-RECORD TO WS-RECORD-HOLD.
 .
 .
```

第 5 部 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加

第 18 章 Enterprise COBOL プログラムの既存 COBOL アプリケーションへの追加

Enterprise COBOL プログラムを既存のアプリケーションに追加する場合、既存のプログラムを Enterprise COBOL で再コンパイルするか、または新たに作成した Enterprise COBOL プログラムを組み込みます。Enterprise COBOL プログラムを既存のアプリケーションに追加すると、以下のことを行うことができます。

- 既存のプログラムをインストール先の要件に応じて漸進的にアップグレードする
- 言語環境プログラムの条件処理を使用する

制約事項: CICS では、同じ実行単位内で OS/VS COBOL プログラムと Enterprise COBOL プログラムを混在させることはできません (EXEC CICS LINK および EXEC CICS XCTL は別々の実行単位を作成します)。

重要

Enterprise COBOL プログラムを既存のアプリケーションに追加した後は、そのアプリケーションは言語環境プログラムのもとで実行しなければなりません。

この章には、以下のトピックに関する情報が記載されています。

- RES プログラムから構成されたアプリケーション
- NORES プログラムから構成されたアプリケーション
- 複数ロード・モジュールの考慮事項
- AMODE および RMODE の考慮事項
- ランタイムの考慮事項

Enterprise COBOL プログラムを既存のアプリケーションに追加する作業を開始したら、既存のアプリケーションを言語環境プログラムとリンク・エディットすることの含意を理解する必要があります。まず、SCEELKED リンク・エディット・ライブラリーを使用する必要があります。これはアプリケーション内の既存のプログラムに影響を与えます。それが既存のアプリケーションにどのような影響を与えるかは、アプリケーションが以下のいずれから構成されているかによって異なります。

- RES を指定してコンパイルされたプログラム
- NORES を指定してコンパイルされたプログラム
- 複数のロード・モジュール

RES プログラムから構成されるアプリケーション

Enterprise COBOL プログラムを、RES を指定してコンパイルされたプログラムから構成されたアプリケーションに追加するときは、以下のことが必要です。

- 変更モジュールを言語環境プログラムとリンク・エディットする
- 言語環境プログラムのもとでアプリケーションを実行する
- 動的および静的 CALL ステートメントの使用に関する要件を理解する

以下のセクションでは、非 CICS および CICS での CALL ステートメントの使用に関する要件をリストします。

静的 CALL ステートメントを使用する Enterprise COBOL プログラムの追加

CICS および非 CICS の両方で、静的 CALL ステートメントを使用して VS COBOL II プログラムを呼び出す Enterprise COBOL プログラムを作成できます。ただし、言語環境プログラムでロード・モジュールをリンク・エディットするときは、適切なレベルの IGZEBST、つまり以下の VS COBOL II ブートストラップが必要です。

- VS COBOL II プログラムが RES コンパイラー・オプションを指定してコンパイルされ、以下のいずれかの製品とリンク・エディットされていた場合は、リンク・エディット・ジョブに、IGZEBST を置き換えるための REPLACE ステートメントを含めなければなりません。
  - APAR PN74000 が適用されていない VS COBOL II リリース 4 ランタイム・ライブラリー
  - APAR PN74011 が適用されていない言語環境プログラム リリース 2 またはリリース 3

正しいレベルの IGZEBST ルーチンが存在しなければ、VS COBOL II プログラムが呼び出されたときにプログラム・チェックが発生します。

最良の CALL パフォーマンスを得るためには、リンク・エディット・ジョブに、次のものを置き換えるための REPLACE ステートメントを含めることをお勧めします。

- COBOL/370 プログラムの場合 — IGZCBSN ブートストラップ・ルーチン (COBOL/370 プログラムが、APAR PN74011 が適用されていない言語環境プログラム リリース 2、リリース 3、またはリリース 4 とリンク・エディットされていた場合)。
- RES コンパイラー・オプションを指定してコンパイルされた VS COBOL II プログラムの場合 — IGZEBST ブートストラップ (VS COBOL II RES プログラムが、APAR PN74011 が適用されていない言語環境プログラム リリース 4 とリンク・エディットされていた場合)。

リンク・エディット JCL の例、または言語環境プログラム・ライブラリー SCEESAMP 内のサンプル・ジョブ (メンバー IGZWRLKA、IGZWRLKB、および IGZWRLKC) については、349 ページの『付録 J. リンク・エディットの例』を参照してください。

非 CICSでの CALL ステートメント

OS/VS COBOL プログラムと Enterprise COBOL プログラム間の CALL ステートメントの場合、パラメーターは 16MB 境界より下になければなりません。以下のセクションでは、動的および静的 CALL ステートメントについて行う必要がある処置を説明します。

動的 CALL ステートメント

OS/VS COBOL プログラムを動的に呼び出す Enterprise COBOL プログラ

ムから渡されるパラメーターは、OS/VS COBOL プログラムからアドレッシング可能でなければなりません。適切な Enterprise COBOL コンパイラー・オプションを指定すれば、データが OS/VS COBOL プログラムからアドレッシング可能になります。

RENT を指定してコンパイルされた Enterprise COBOL プログラムの場合、DATA(24) コンパイラー・オプションを指定してください。

NORENT を指定してコンパイルされた Enterprise COBOL プログラムの場合、RMODE(24) または RMODE(AUTO) コンパイラー・オプションを指定してください。

静的 CALL ステートメント

OS/VS COBOL プログラムと Enterprise COBOL プログラム間で静的 CALL ステートメントを発行することによって単一のロード・モジュールを形成する場合、ロード・モジュールは 16MB 境界より下に置かなければなりません。ロード・モジュールは、RMODE 24、AMODE 24 としてマークされなければなりません。

Enterprise COBOL プログラムと OS/VS COBOL プログラムの両方を含んでいるロード・モジュールの場合、ロード・モジュールが、NORENT を指定してコンパイルされた Enterprise COBOL プログラムを含んでいるときは、デフォルト AMODE 設定を AMODE 24 にオーバーライドする必要があります (RENT を指定してコンパイルされたプログラムの場合は、処置は必要ありません。リンケージ・エディターが正しい AMODE 設定を自動的に割り当てます)。デフォルト AMODE 設定をオーバーライドする方法については、347 ページの『付録 I. リンケージ・エディターのデフォルトのオーバーライド』を参照してください。

CICS での CALL ステートメント

Enterprise COBOL では、VS COBOL II、IBM COBOL、Enterprise COBOL、アセンブラー、C、および PL/I プログラムの呼び出しに静的および動的 CALL ステートメントを使用することができます。ただし、言語環境プログラムは Enterprise COBOL プログラムと OS/VS COBOL プログラム間の静的または動的 CALL ステートメントをサポートしません (引き続き、EXEC CICS LINK によって OS/VS COBOL サブルーチンにアクセスしてください)。

一般的な考慮事項

Enterprise COBOL プログラムが CICS トランスレーター (単独または組み込みのどちらの場合でも) によって処理された場合は、そのプログラムの呼び出し元は CICS EXEC インターフェース・ブロック (DFHEIBLK) および連絡域 (DFHCOMMAREA) を CALL ステートメントの最初の 2 つのパラメーターとして渡す必要があります。Enterprise COBOL プログラムが CICS トランスレーターによって処理されなかった場合は、DFHEIBLK および DFHCOMMAREA を渡すことが必要なのは、呼び出し先サブプログラムでそれらが明示的にコーディングされている場合だけです。

CICS コマンド変換プロセスは、これらのパラメーターを、サブプログラム内の対応する PROCEDURE DIVISION USING ステートメントの最初の 2 つのパラメーターとして自動的に挿入します。

静的 CALL ステートメント

Enterprise COBOL では、COBOL CALL ステートメントを使用して、VS COBOL II、IBM COBOL、Enterprise COBOL、およびアセンブラー・プログラムを動的に呼び出すことができます。どのような場合に静的 CALL ステートメントがサポートされるかの詳細については、305 ページの表 51 を参照してください。さらに、言語環境プログラムは、CICS での実行時に PL/I と COBOL 間および C と COBOL 間の ILC をサポートします。詳細については、*言語環境プログラム ILC (言語間通信) アプリケーションの作成* を参照してください。

OS/VS COBOL では、複数の COBOL プログラムが別々にコンパイルされ、その後で 1 つにリンク・エディットされる場合、最初のプログラムだけが CICS ステートメントを含むことができます。Enterprise COBOL では、この制限が取り除かれたため、アプリケーション・プログラムの設計をより柔軟に行うことができます。

動的 CALL ステートメント

Enterprise COBOL では、COBOL CALL ステートメントを使用して、VS COBOL II、IBM COBOL、Enterprise COBOL、および アセンブラー・プログラムを動的に呼び出すことができます。どのような場合に動的 CALL ステートメントがサポートされるかの詳細については、305 ページの表 51 を参照してください。たとえば、動的 CALL ステートメントのターゲットであるプログラムには、CICS ステートメントを含めることができます。さらに、言語環境プログラムは、CICS での実行時に PL/I と COBOL 間および C と COBOL 間の ILC をサポートします。詳細については、*言語環境プログラム ILC (言語間通信) アプリケーションの作成* を参照してください。

CICS のもとで実行される Enterprise COBOL プログラムの場合、CALL ステートメントの ON EXCEPTION/OVERFLOW 文節を使用することができます。

NORES プログラムから構成されたアプリケーション

Enterprise COBOL プログラムを、NORES を指定してコンパイルされたプログラムから構成されたロード・モジュールに追加するときは、以下のことが必要です。

- Enterprise COBOL プログラムを言語環境プログラムとリンク・エディットする。
- アプリケーション内の他のすべてのプログラムを言語環境プログラムとリンク・エディットし、これらの NORES プログラムが言語環境プログラムとリンク・エディットされたときの動作の変更を理解する。
- リンク・エディット時に、すべての IGZ および ILBO CSECT を言語環境プログラムからのコピーで REPLACE する。ロード・モジュール内の現行ライブラリー・ルーチンを言語環境プログラム・ライブラリー・ルーチンで置き換える方法を示すリンク・エディット JCL の例については、349 ページの『付録 J. リンク・エディットの例』を参照してください。

言語環境プログラムとリンク・エディットする前の動作

言語環境プログラムとのリンク・エディットは、すべてのプログラムが NORES であるアプリケーションの場合は、Enterprise COBOL プログラムを追加する場合を除いて必要ありません。

NORES を指定してコンパイルされ、言語環境プログラムとリンク・エディットされていないプログラムは、73 ページの 『第 6 章 OS/VS COBOL ランタイムからの移行』で説明したトピックの大部分には影響されません。これは、NORES を指定してコンパイルされたプログラムが、言語環境プログラム・ライブラリーにアクセスせずに、そのプログラムがリンク・エディットされた環境で稼働し続けるためです。

言語環境プログラムとリンク・エディットした後の動作

Enterprise COBOL プログラムを追加し、残りのプログラムを言語環境プログラムとリンク・エディットしたあとは、NORES プログラムは RES プログラムと同様に動作します。RES を指定してコンパイルされたプログラムに関する考慮事項の多くが、言語環境プログラムとリンク・エディットされたこれらの NORES プログラムに適用されます。詳細については、122 ページの『RES と同様になることの含意』を参照してください。

リンク・エディットのオーバーライド要件

Enterprise COBOL プログラムと OS/VS COBOL プログラムの両方を含んでいるロード・モジュールの場合、ロード・モジュールが、NORENT を指定してコンパイルされた Enterprise COBOL プログラムを含んでいるときは、デフォルト AMODE 設定を AMODE 24 にオーバーライドする必要があります。デフォルト AMODE 設定をオーバーライドする方法については、347 ページの『付録 I. リンケージ・エディターのデフォルトのオーバーライド』を参照してください。

複数ロード・モジュールの考慮事項

複数のロード・モジュールから構成されたアプリケーションは、言語環境プログラムのもとでサポートされない場合があります。複数ロード・モジュール・アプリケーションがサポートされるかどうかを判別するには、以下のことが分かっている必要があります。

- メインモジュール
- メインモジュールのメインプログラム
- サブモジュール

注: 複数プログラム・ロード・モジュールの一部である OS/VS COBOL NORES プログラムまたは VS COBOL II NORES プログラムを言語環境プログラムとリンク・エディットするときは、ロード・モジュール内の COBOL ライブラリー・ルーチンを言語環境プログラム・ライブラリー・ルーチンで置き換えなければなりません。これを行わないと、予期しない結果が生じる可能性があります。ライブラリー・ルーチンを置き換えるためのコーディング例については、349 ページの『付録 J. リンク・エディットの例』を参照してください。

OS/VS COBOL の考慮事項

256 ページの表 47 に、OS/VS COBOL プログラムの場合の複数ロード・モジュールの可能なすべての組み合わせをリストします。以下の表では、ロード・モジュールに複数のプログラムがある場合は、メインプログラムが最初にリストされています。

Enterprise COBOL プログラムの既存アプリケーションへの追加

表 47. 複数のロード・モジュールから構成されたアプリケーションについてのサポート — OS/VS COBOL

メインモジュール	サブモジュール	サポート	アプリケーション全体を LanEnv とリンク・エディットすることが必要
OS/VS COBOL NORES [1]	Enterprise COBOL のみ	いいえ	適用外
	Enterprise COBOL と IBM COBOL	いいえ	適用外
	Enterprise COBOL および IBM COBOL または VS COBOL II RES のみ	いいえ	適用外
	OS/VS COBOL RES のみ	いいえ	適用外
	OS/VS COBOL NORES のみ	はい	いいえ[2]
	Enterprise COBOL および OS/VS COBOL RES	いいえ	適用外
	Enterprise COBOL および OS/VS COBOL NORES	いいえ	適用外
	OS/VS COBOL RES および OS/VS COBOL NORES	いいえ	適用外[4]
OS/VS COBOL RES	すべての組み合わせ	はい	はい[2]
Enterprise COBOL	すべての組み合わせ	はい	はい[2]
Enterprise COBOL および OS/VS COBOL RES	すべての組み合わせ	はい	はい[2]
OS/VS COBOL RES および Enterprise COBOL	すべての組み合わせ	はい	はい[2]
Enterprise COBOL および OS/VS COBOL NORES	すべての組み合わせ	はい	はい[2]
OS/VS COBOL NORES および Enterprise COBOL	すべての組み合わせ	はい	はい[2]
OS/VS COBOL RES および OS/VS COBOL NORES	すべての組み合わせ	いいえ[3]	適用外
OS/VS COBOL NORES および OS/VS COBOL RES	すべての組み合わせ	いいえ[3]	適用外
Enterprise COBOL および OS/VS COBOL RES および OS/VS COBOL NORES	すべての組み合わせ	はい	はい[2]
OS/VS COBOL RES および OS/VS COBOL NORES および Enterprise COBOL	すべての組み合わせ	はい	はい[2]
OS/VS COBOL NORES および OS/VS COBOL RES および Enterprise COBOL	すべての組み合わせ	はい	はい[2]

表 47. 複数のロード・モジュールから構成されたアプリケーションについてのサポート — OS/VS COBOL (続き)

メインモジュール	サブモジュール	サポート	アプリケーション全体を LanEnv とリンク・エディットすることが必要

注:

1. OS/VS COBOL NORES プログラムだけを含んでいるロード・モジュールは、アセンブラー・プログラムを使用してサブモジュールをロードする (またはサブモジュールにリンクする) 場合にのみ、サブモジュールにアクセスすることができます。
2. NORES を指定してコンパイルされた既存の OS/VS COBOL プログラムは、変更なしで稼働し、以前と同じ結果をもたらします。これらのプログラムは、言語環境プログラムとリンク・エディットする必要はありません。ただし、これらのプログラムを言語環境プログラムとリンク・エディットしない場合は、これらの NORES アプリケーションのために IBM サービス・サポートを受けることはできません。
3. サブモジュールが OS/VS COBOL RES プログラムだけを含んでいる場合は、言語環境プログラムとのリンク・エディットは必要ありません。
4. RES を指定してコンパイルされた OS/VS COBOL プログラムおよび NORES を指定してコンパイルされた OS/VS COBOL プログラムを含んでいるロード・モジュールは、Enterprise COBOL プログラムまたは特定の CSECT を組み込まない限り、サポートされません。詳細については、75 ページの 『RES および NORES を指定してコンパイルされた COBOL プログラムを含んでいるアプリケーション』を参照してください。

すべての組み合わせ とは、サブモジュールを構成することができるプログラムの異なる組み合わせ (表の先頭の OS/VS COBOL NORES の隣りにリストされている) を意味します。

VS COBOL II の考慮事項

VS COBOL II プログラムを含む複数のロード・モジュールを持つアプリケーションは、言語環境プログラムとリンク・エディットしなくてもサポートされます。ただし、VS COBOL II NORES プログラムと RES プログラム (VS COBOL II、IBM COBOL、または Enterprise COBOL) の組み合わせがある場合を除きます。VS COBOL II NORES プログラムと Enterprise COBOL または IBM COBOL RES プログラムの組み合わせが存在する場合は、アプリケーションを言語環境プログラムとリンク・エディットする必要があります。

メイン・モジュールとサブモジュールが NORES を指定してコンパイルされた VS COBOL II プログラムである場合は、変更なしで稼動し、以前と同じ結果をもたらします。これらのプログラムは、言語環境プログラムとリンク・エディットする必要はありません。ただし、これらのプログラムを言語環境プログラムとリンク・エディットしない場合は、これらの NORES アプリケーションのために IBM サービス・サポートを受けることはできません。モジュールを言語環境プログラムとリンク・エディットしたあとは、IBM サービスによるサポートを受けることができます。

一般に、VS COBOL II NORES プログラムを含んでいる複数ロード・モジュール・アプリケーションは、言語環境プログラムとリンク・エディットしなければなりません。モジュールがいずれも VS COBOL II RES、IBM COBOL、または Enterprise COBOL のプログラムのみを含む場合は、アプリケーションはサポートされますので、言語環境プログラムとリンク・エディットする必要はありません。

その他のリンク・エディット要件については、252 ページの 『静的 CALL ステートメントを使用する Enterprise COBOL プログラムの追加』を参照してください。

AMODE および RMODE の考慮事項

すべての OS/VS COBOL プログラムは AMODE 24 および RMODE 24 です。Enterprise COBOL プログラムは常に AMODE ANY であり、RMODE 24 または RMODE ANY のいずれかです。WORKING-STORAGE データ項目は、DATA、RENT、および RMODE コンパイラー・オプションに基づいて、16MB 境界より上または下に置かれます。

OS/VS COBOL プログラムは、AMODE 問題なしで CALL ステートメントを使用して Enterprise COBOL プログラムを呼び出すことができます。これは、両方のプログラムが 16MB 境界より下のデータにアクセスできるためです。Enterprise COBOL プログラムが CALL ステートメントを使用して OS/VS COBOL プログラムを呼び出すときは、データまたはパラメーター・リストが 16MB 境界より上にあると、アドレッシング例外が起こる可能性があります。

アドレッシング例外を避けるためには、RENT プログラムの場合は DATA(24)、NORENT プログラムの場合は RMODE(24) または RMODE(AUTO) を指定してコンパイルすることによって、すべてのデータおよびパラメーター・リストを 16MB 境界より下に置くようにしてください。

注: バージョン 2 リリース 9 以前のリリースの言語環境プログラムでは、DATA(31) に対して要求を出すと、言語環境プログラムが 16MB 境界より下のストレージを獲得することにより、正常にコンパイルされなかったプログラムがアドレッシング例外を出さずにパラメーターを OS/VS COBOL プログラムに渡す可能性がありました。言語環境プログラム バージョン 2 リリース 9 以上では、このように DATA(31) の使用法を誤ると予期アドレッシング例外が発行されます。

259 ページの図 7 に、コンパイラー・オプションとコンパイラーの各種の組み合わせの結果を示します。すべての CALL ステートメントは動的であり、矢印によって表されています。実線は有効な CALL ステートメントを表し、点線は無効な呼び出しを表しています。

P1 および P5 は OS/VS COBOL プログラムであるため、WORKING-STORAGE データ項目はオブジェクト・モジュールに組み込まれ、16MB 境界より下になければなりません。RENT オプションを指定してコンパイルされた Enterprise COBOL プログラムの場合、WORKING-STORAGE データ項目はオブジェクト・モジュールから分離され、それらの位置は DATA コンパイラー・オプションによって制御されます (プログラム P2 および P3 と同様)。NORENT を指定してコンパイルされたプログラムの場合、WORKING-STORAGE データ項目はオブジェクト・モジュールに組み込まれるため、それらの位置は RMODE オプションによって異なります (プログラム P4 と同様)。

これらの呼び出しは、P5 を呼び出す P3 を除き、すべて正しく機能します。P3 は RENT および DATA(31) を指定してコンパイルされたため、P3 の WORKING-STORAGE およびパラメーター・リストは 16MB 境界より上に置かれます。これは、P3 が P2 から受け取ったパラメーターを渡す場合でも、P5 はパラ

メーター・リストにアドレッシングできないことを意味し、このため CALL は失敗します。P1 および P3 の WORKING-STORAGE SECTION 内のデータ項目のように、パラメーター自体が 16MB 境界より上にある場合も、CALL は失敗します。

注: AMODE 31 プログラムから OS/VS COBOL プログラムへの静的 CALL ステートメントは、必ず失敗します。

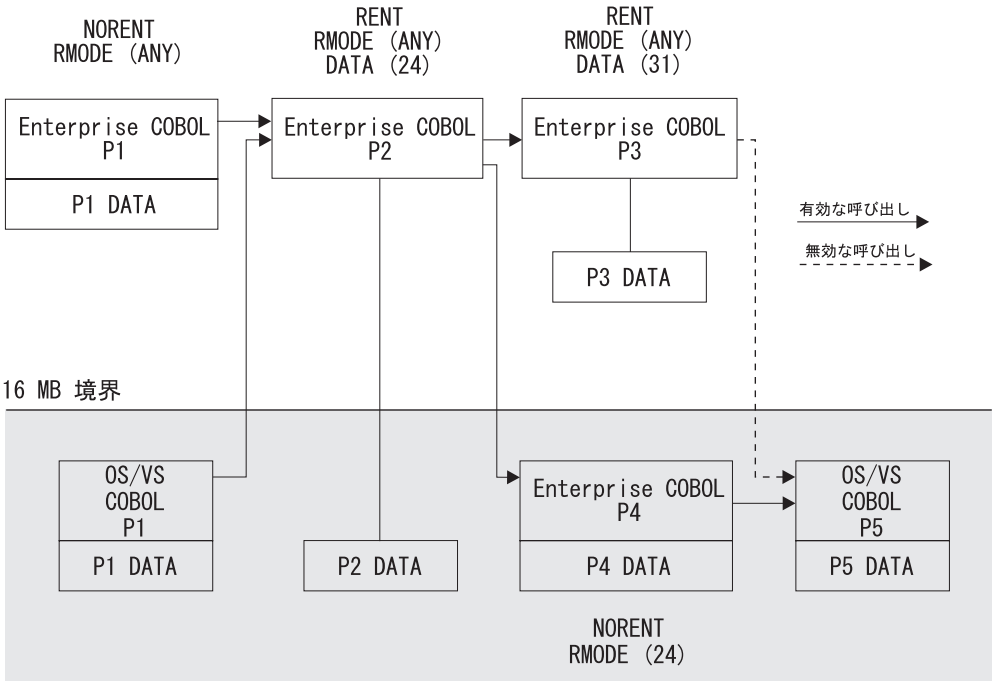


図 7. OS/VS COBOL プログラムと Enterprise COBOL プログラム間の有効な呼び出しと無効な呼び出し

ランタイムの考慮事項

Enterprise COBOL プログラムを既存のアプリケーションに追加する場合は、その他の考慮事項があります。

ILBOSRV

以下のいずれかの場合は、言語環境プログラム リリース 5 (またはそれ以上) の ILBOSRV コピーをロード・モジュールに組み込む必要があります。

- 既存のロード・モジュールが ILBOSTP0 を含んでおり、それを使用する場合
- 既存のロード・モジュールが OS/VS COBOL NORES プログラムを含んでいる場合

TGT (タスク・グローバル・テーブル) および RSA (レジスター保管域) の規則

Enterprise COBOL、COBOL (OS/390 および VM 版)、および COBOL (MVS および VM 版) における TGT と RSA の規則は、VS COBOL II および COBOL/370 の場合と異なります。既存の COBOL アプリケーションがレジスター 9 またはレ

ジスター 13 の値を前提とするようにコーディングされている場合は、それらのプログラムを、レジスター値についての組み込まれた前提事項を持たないように変更する必要があります。

以下のようにして、Enterprise COBOL、COBOL (OS/390 および VM 版)、または COBOL (MVS および VM 版) プログラムの TGT を検出できます。

- Enterprise COBOL、COBOL (OS/390 および VM 版)、または COBOL (MVS および VM 版) プログラムが実行されているときは、R13 に COBOL プログラム用の DSA (動的ストレージ域) のアドレスが入っています。COBOL プログラムの TGT のアドレスは、DSA 内の x『5C』にあります。
- R13 に呼び出し先アセンブラー・プログラムまたは COBOL ランタイム・ルーチンに関連した保管域のアドレスが入っている場合は、保管域逆チェーンを使用して COBOL プログラム用の DSA を見つけてください。その後、x『5C』を加算して、TGT のアドレスを見つけてください。

  直前の保管域の R15 スロットを使用すれば、DSA を所有するルーチンの入り口点およびシグニチャー情報を判別することができます。

第 6 部 付録

付録 A. よくある質問および回答

この付録では、Enterprise COBOL および言語環境プログラムへのアップグレードに関する最も一般的な質問への回答を示します。質問は、以下のカテゴリーに分類されています。

- 前提条件
- 互換性
- 言語環境プログラムとのリンク・エディット
- Enterprise COBOL でのコンパイル
- 言語環境プログラム・サービス
- 言語環境プログラム・ランタイム・オプション
- 言語間通信
- サブシステム
- OS/390
- z/OS
- パフォーマンス
- サービス

前提条件

マクロ・レベル COBOL プログラムはすべて、言語環境プログラムで実行するためには CICS バージョン 4 に移行しなければなりませんか ?

はい。言語環境プログラムは CICS バージョン 4 でのみ実行することができるため、言語環境プログラムのもとで実行するプログラムはすべて CICS バージョン 4 以上にアップグレードしなければなりません。

Enterprise COBOL でコンパイルされたプログラムを実行するには、言語環境プログラムが必要ですか ?

はい。言語環境プログラムは、Enterprise COBOL でコンパイルされたプログラムを実行するのに必要なライブラリー・ルーチンを含んでいます。Enterprise COBOL はコンパイラーであり、ランタイム・ライブラリー・ルーチンを含んでいません。

互換性

COBOL プログラムとアセンブラー・プログラムの混合があります。アセンブラー・プログラムを言語環境プログラム使用可能に変更する必要がありますか ?

いいえ。アセンブラー・プログラムを言語環境プログラム使用可能に変更する必要はありません。ただし、アセンブラー・プログラムは S/390 保管域規則に従っていなければなりません。つまり、次のとおりでなければなりません。

- アセンブラー保管域の最初のハーフワードは 16 進ゼロである。
- バック・チェーン・アドレスは呼び出し元の保管域に設定されている。
- バック・チェーン・アドレスは有効な 31 ビット・アドレスである。

言語環境プログラムのもとで DOS/VS COBOL プログラムを実行することはできますか ?

はい。DOS/VS COBOL でコンパイルされたプログラムは、IBM 言語環境プログラム (VSE/ESA 版) のもとで実行することができます。マイグレーション情報については、*IBM COBOL for VSE/ESA 移行の手引き*、GD88-6041 を参照してください。

DOS/VS COBOL でコンパイルされたプログラムは、プログラム番号 5688-198 (言語環境プログラム) のもとでは実行できません。

言語環境プログラムは、LANGLVL(1) を指定してコンパイルされた OS/VS COBOL プログラムと LANGLVL(2) を指定してコンパイルされた OS/VS COBOL プログラムを両方ともサポートしますか ?

はい。

OS/VS COBOL を言語環境プログラムとの互換モードで実行するとき、ランタイム制御ブロックは同じ方法でアクセスされますか?

はい。TGT 内および他の制御ブロック内のポインターを引き続き使用して、OS/VS COBOL 用の制御ブロックに到達することができます。

ただし、Enterprise COBOL の場合は、アセンブラー・プログラムは R13 から TGT のアドレスを取得できません。

VS COBOL II では、出力 DD につづりの誤りがあるとエラーが発生し、一時ディスク・ファイルが作成されました。これが 1 回のプログラム実行でラージ・ファイルについて起こると、問題が生じます。このことは、Enterprise COBOL の場合も問題になりますか ?

いいえ。QSAM の場合、言語環境プログラムの CBLQDA(OFF) ランタイム・オプションを使用して自動ファイル作成をオフにすることができます。

CMPR2 オプションを使用しなければならないのはいつですか ?

CMPR2/NOCMPR2 オプションは Enterprise COBOL では利用できません。Enterprise COBOL は NOCMPR2 が常に有効であるかのように動作します。以前のコンパイラーで CMPR2 を指定してコンパイルされたプログラムはいずれも、Enterprise COBOL で コンパイルするために COBOL 1985 標準にアップグレードする必要があります。

言語環境プログラムを LNKLST/LPALST 内に置くことはできますか ?

はい。ただし、その言語環境プログラムが LNKLST/LPALST 内にある唯一の COBOL 用のランタイム・ライブラリーになるようにしてください。言語環境プログラムは古い COBOL ランタイム・ライブラリーより先に LNKLST または LPALST 内に置くことができますが、古い方のランタイム・ライブラリーは到達不能になります。

言語環境プログラムを LNKLST/LPALST 内に置く前に、LNKLST/LPALST から言語環境プログラム・ライブラリー・ルーチンにアクセスする可能性があるすべてのアプリケーションをテストしてください。言語環境プログラムへの移行は、制御された方法で実現してください。

詳細については、 34 ページの『言語環境プログラムを段階的に実動モードに移す方法の決定』を参照してください。

言語環境プログラムは、システム上で OS/VS COBOL および VS COBOL II と共存することができますか ?

はい。しかし、各種のアプリケーションに必要とされ、使用されるライブラリーを理解しておかなければなりません。たとえば、連結内で VS COBOL II が言語環境プログラムより前にあり、VS COBOL II アプリケーションの 1 つのプログラムを Enterprise COBOL で再コンパイルした場合、そのアプリケーションは言語環境プログラムが連結内で VS COBOL II ランタイム・ライブラリーより前に来るまで実行されません。各種のプロダクト・ライブラリー間には重複した名前があるため、正しいライブラリーがアクセスされるようにすることが重要です。

詳細については、 65 ページの『既存のアプリケーションの呼び出し』を参照してください。

OS/VS COBOL から Enterprise COBOL への移行は、OS/VS COBOL から VS COBOL II への移行よりも容易ですか ?

ソースの移行手段はほとんど同じです。OS/VS COBOL と共に SVC LINK または条件処理を使用するアセンブラー・プログラムがある場合は、言語環境プログラム・ランタイムへの移行は若干困難です。時間の大部分はテストに費やされるため、2 つの移行経路はほとんど同じです。

ランタイムの考慮事項の要約については、以下の図を参照してください。

- 32 ページの表 12 (OS/VS COBOL ランタイムのもとで稼働しているプログラムの場合)
- 32 ページの表 13 (VS COBOL II ランタイムのもとで稼働しているプログラムの場合)

Enterprise COBOL プログラムのシグニチャー域は、OS/VS COBOL および VS COBOL II の場合と同じですか ?

いいえ。シグニチャー域のマップが *Enterprise COBOL プログラミング・ガイド* に記載されていますので、これを使用して、モジュールのコンパイルに使用されたコンパイラー・オプション、モジュールがいつコンパイルされたか、およびリリース・レベルなどを調べることができます。

言語環境プログラムとのリンク・エディット

アプリケーションを言語環境プログラムのもとで実行するために言語環境プログラムとリンク・エディットすることが必要なのはいつですか ?

正確なリンク・エディット要件については、以下のセクションを参照してください。

- 74 ページの『リンク・エディットが必要なプログラムの判別』 (OS/VS COBOL ランタイムのもとで稼働しているプログラムの場合)
- 88 ページの『リンク・エディットが必要なプログラムの判別』 (VS COBOL II ランタイムのもとで稼働しているプログラムの場合)

OS/VS COBOL のもとでは、OS/VS COBOL プログラムが NORES を指定してコンパイルされた場合でも、一部のライブラリー・ルーチンが必ず動的に呼び出されませんか ? 言語環境プログラムのもとでの実行時にこれらのライブラリー・ルーチンがサポートされるようにするために、言語環境プログラムとリンク・エディットすることが必要ですか ?

はい。OS/VS COBOL のもとでは、
ILBOD01、ILBODBE、ILBOPRM、ILBOSND、ILBOSTN、および ILBOTC2 が必ず動的に呼び出されます (リンク・エディットによって明示的に INCLUDE されない限り)。言語環境プログラムは、プログラムが言語環境プログラムとリンク・エディットされているかどうかに関係なく、これらのライブラリー・ルーチンをサポートします。

OS/VS COBOL および VS COBOL II プログラムで Enterprise COBOL プログラムを呼び出すことはできますか ?

非 CICS では、OS/VS COBOL、VS COBOL II、および Enterprise COBOL 間のすべての呼び出しがサポートされます。

CICS では、Enterprise COBOL プログラムと OS/VS COBOL プログラム間で呼び出しを行うことはできません。代わりに EXEC CICS LINK/XCTL を使用する必要があります。VS COBOL II プログラムおよび Enterprise COBOL プログラムとの間の呼び出しは許可されます。その他の詳細については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

COBOL とアセンブラー間の呼び出しの完全なリスト (言語環境プログラムのもとでの実行時にそれらがサポートされるかどうかを含む) については、307 ページの『CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』を参照してください。

OS/VS COBOL NORES ロード・モジュールと Enterprise COBOL プログラム間で呼び出しを行うことはできますか ?

Enterprise COBOL ロード・モジュールは OS/VS COBOL NORES ロード・モジュールを呼び出すことができます (NORES ロード・モジュールが言語環境プログラムとリンク・エディットされている場合)。OS/VS COBOL NORES ロード・モジュールは Enterprise COBOL ロード・モジュールに制御を戻さなければなりません。

Enterprise COBOL プログラムへの「動的」呼び出しを使用する (つまり、ロードおよび分岐を行うアセンブラー・プログラムを使用する) OS/VS COBOL NORES ロード・モジュールを持つことが可能です。この Enterprise COBOL プログラムは、その後、後続のプログラムへの COBOL 動的呼び出しを行うことができます。

プログラムを選択的に Enterprise COBOL に移行することはできますか ?

非 CICS アプリケーションの場合は、リンク・エディットの規則 (本書に記載されている) に従う限り、移行することができます。

CICS アプリケーションの場合は、同じ実行単位内で OS/VS COBOL プログラムと Enterprise COBOL プログラムを混在させることはできません。CICS で実行される OS/VS COBOL プログラムを含んでいるアプリケーションを移行するときは、実行単位内のすべての OS/VS COBOL プログラムを Enterprise COBOL に移行する必要があります。

Enterprise COBOL でのコンパイル

OS/VS COBOL 用に作成されたプログラムを、CMPR2 オプションを用いて Enterprise COBOL でコンパイルすることはできますか ?

いいえ。CMPR2 を Enterprise COBOL で使用することはできません。

VS COBOL II 用に作成されたプログラムを Enterprise COBOL でコンパイルすることはできますか ?

はい。詳細については、20 ページの『ソースを Enterprise COBOL にアップグレードする』を参照してください。

OS/VS COBOL および VS COBOL II プログラムは、言語環境プログラムのもとで実行するためには Enterprise COBOL でコンパイルしなければなりませんか ?

いいえ。ほとんどの OS/VS COBOL プログラムおよび VS COBOL II プログラム (および 2 つの混合) は、Enterprise COBOL にアップグレードしなくても、言語環境プログラムのもとで稼働します。

Enterprise COBOL にアップグレードしなければならないプログラムの詳細については、以下のセクションを参照してください。

- 75 ページの『アップグレードが必要なプログラムの判別』 (OS/VS COBOL ランタイムのもとで稼働しているプログラムの場合)
- 90 ページの『アップグレードが必要なプログラムの判別』 (VS COBOL II ランタイムのもとで稼働しているプログラムの場合)

OS/VS COBOL または VS COBOL II ソースを Enterprise COBOL ソースに移行するときに、どのユーティリティーまたはツールが役立ちますか ?

以下の移行ツール (IBM を介して購入可能) が、OS/VS COBOL および VS COBOL II ソースを Enterprise COBOL ソースに移行するときに役立ちます。

1. COBOL 移行援助プログラム (CCCA) 5648-B05 は、OS/VS COBOL および VS COBOL II ソースを Enterprise COBOL ソースに移行するときに役立ちます。
2. COBOL 報告書作成プログラム・プリコンパイラー 5798-DYR は、OS/VS COBOL 報告書作成プログラム・コードの変換に役立ちます。
3. CICS アプリケーション・マイグレーション・エイド 5695-061 は、OS/VS COBOL および VS COBOL II の CICS マクロ・レベル・コードを CICS コマンド・レベル・コードに変換するときに役立ちます。

4. Edge Portfolio Analyzer は、既存の OS/VS COBOL および VS COBOL II ロード・モジュール・ライブラリーの目録の作成に役立ちます。
5. WebSphere Studio Asset Analyzer は目録の作成、およびコード変更が企業の資産に及ぼす影響を分析する上で役立ちます。

Enterprise COBOL は COBOL 85 標準に適合していますか ?

はい。Enterprise COBOL は、COBOL 85 標準のすべての必要なモジュールを、標準によって定義されている最高のレベルでサポートします。

言語環境プログラム・サービス

VS COBOL II プログラムで 言語環境プログラム呼び出し可能サービスを呼び出すことはできますか ?

VS COBOL II プログラムから言語環境プログラムの日付および時刻呼び出し可能サービスへの動的呼び出しを使用することだけができます。VS COBOL II プログラムから他の言語環境プログラム呼び出し可能サービスへの動的呼び出しを使用することはできませんし、VS COBOL II プログラムから言語環境プログラム呼び出し可能サービスへの静的呼び出しを使用することもできません。

VS COBOL II プログラム (任意のリリース) から以下の言語環境プログラム・サービスを動的に呼び出すことができます。

CEECBLDY	CEEGMT	CEESCEN
CEEDATE	CEEGMTO	CEESECI
CEEDATM	CEEISEC	CEESECS
CEEDAYS	CEELOCT	CEE3CTY
CEEDYWK	CEEQCEN	

BAL (アセンブラー) プログラムが言語環境プログラム準拠のアセンブラー・プログラムである場合に、そのプログラムから言語環境プログラム呼び出し可能サービスを呼び出すことはできますか ?

はい。すべての言語環境プログラム準拠 BAL ルーチンは、言語環境プログラム呼び出し可能サービスを使用することができます。非言語環境プログラム準拠 BAL ルーチンは、言語環境プログラム呼び出し可能サービスを使用することができません。 *言語環境プログラム プログラミング・ガイド* に、IBM によって言語環境プログラム・プロダクトと共に提供されるマクロを使用して既存の BAL プログラムを言語環境プログラム準拠 BAL プログラムにする方法が記載されています。

OS/VS COBOL プログラムで 言語環境プログラム呼び出し可能サービスを使用することはできますか ?

いいえ。OS/VS COBOL プログラムでは、言語環境プログラム呼び出し可能サービスを直接に使用することはできません。しかし、サービスを呼び出す Enterprise COBOL プログラムを呼び出すことができます。

メインルーチンを Enterprise COBOL に移行することによって、純粋な OS/VS COBOL アプリケーションに条件処理を追加することはできますか ?

はい (リカバリーに関する制約事項があります)。詳細については、 308 ページの『条件処理に ESTAE/ESPIE を使用するプログラムの移行』を参照してください。

COBOL マルチスレッド化とは何ですか ? また、それは PL/I マルチタスキングとどのような関係がありますか ?

COBOL マルチスレッド化とは、同じアドレス・スペースで、同じプロセスで同時に稼動する複数のプログラムのサポートです。COBOL によって開始することはできませんが、「pthread create」を実行して C プログラムによって開始することができます。COBOL マルチスレッド化は、複数の PL/I タスクが THREAD コンパイラー・オプションを指定してコンパイルされた場合に COBOL プログラムを呼び出すことができるという点で PL/I マルチタスキングと互換性があります。

PL/I は、固有言語を用いてマルチタスキングを開始し、個々のタスク間の対話を管理することができます。

言語環境プログラム・ランタイム・オプション

COBOL パフォーマンスのための低い HEAP ストレージ値は、C または C++ プログラムのパフォーマンスに影響を与えますか ?

はい。HEAP ストレージ値が低い場合、C プログラムが多くの MALLOC を使用すると、C パフォーマンスは低下します。

COBOL パフォーマンスのための低い HEAP ストレージ値は、PL/I パフォーマンスに影響を与えますか ?

通常は、影響を与えません。ただし、ALLOCATE および FREE を頻繁に使用するアプリケーションの場合は、パフォーマンスが低下する可能性があります。この場合には、パフォーマンスを向上させるために HEAP 値を調整してください。さらに、アプリケーションに多くの自動変数がある場合は、パフォーマンスを向上させるために STACK 値も調整してください。

Enterprise COBOL は STACK ストレージを使用しますか ?

Enterprise COBOL プログラムは LOCAL-STORAGE データ項目用に STACK ストレージを使用します。その他の COBOL プログラムは STACK ストレージを使用しません。

COBOL ランタイム・ルーチンは STACK ストレージを使用します。

言語環境プログラムのもとで稼動する OS/VS COBOL は HEAP ストレージを使用しますか ?

いいえ。OS/VS COBOL の WORKING-STORAGE は HEAP ストレージを使用しません。

HEAP(KEEP) または LIBSTACK(KEEP) の役割は何ですか ? KEEP サブオプションは、HEAP または LIBSTACK ストレージのすべてを保持するのですか、それとも取得された追加の分のストレージだけを保持するのですか ?

KEEP サブオプションを指定すると、言語環境プログラムは、取得されたストレージの**すべて** (初期量および増分量を含む) を保持します。

ERRCOUNT は異常終了とどのような関係がありますか ? ERRCOUNT は処理された条件だけをカウントするのですか ?

ERRCOUNT は、言語環境プログラムがそれ自身の異常終了コードを出して異常終了するまでに認められたエラー、条件、異常終了、および例外のカウントです。エラーが処理されない場合は、アプリケーションが終了するため、ERRCOUNT は何の影響も与えません。

言語間通信

なぜ、言語環境プログラムの資料または IBM の発表の中の言語間通信ではアセンブラー言語プログラムが論じられていないのですか ?

アセンブラー言語は、言語環境プログラムでは独立した言語であると見なされません。アセンブラー・プログラムに関連したランタイム・ライブラリーはなく、このため、アセンブラーと他の高水準言語 (HLL) とのランタイム対立はありません。アセンブラー・プログラムは、言語環境プログラムによってサポートされる HLL との間で呼び出しを行うことができます。

OS/VS COBOL または VS COBOL II プログラムで、言語環境プログラム準拠のアセンブラー・プログラムを呼び出すことはできますか ?

以下の呼び出しがサポートされます。

- 非 CICS の場合
  - OS/VS COBOL RES プログラムまたは VS COBOL II RES プログラムから言語環境プログラム準拠のアセンブラーへの動的呼び出し。
  - OS/VS COBOL RES プログラムまたは VS COBOL II RES プログラムから言語環境プログラム準拠のアセンブラーへの静的呼び出し。CEEENTRY マクロで MAIN=NO および NAB=NO を指定しなければなりません。
- CICS の場合
  - VS COBOL II から言語環境プログラム準拠のアセンブラーへの動的呼び出し。

COBOL とアセンブラー間の呼び出しの完全なリスト (言語環境プログラムのもとでの実行時にそれらがサポートされるかどうかを含む) については、305 ページの『非 CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』および 307 ページの『CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート』を参照してください。

サブシステム

CICS 領域での実行時に、EXEC DLI は、CEETDLI または CBLTDLI へのアクセスに「変換」されますか ?

EXEC DLI は、CEETDLI または CBLIDLI へのアクセスに「変換」されません。CICS トランスレーターは、DFHELI への呼び出しを生成します。DFHELI への呼

び出しは静的呼び出しでなければなりません (CICS トランスレーターによって変換されたプログラムについては、NODYNAM コンパイラー・オプションが必要です)。

CALL 'CEETDLI' は CICS プログラム内でサポートされますか ? 言語環境プログラムのもとで稼働する CICS プログラム内の CALL 'CBLTDLI' の場合はどうですか ?

CEETDLI は CICS 環境でサポートされません (CICS は、DFHDLIAL で CEETDLI 入り口点を提供しません)。言語環境プログラムのもとでは、CBLTDLI は CICS 環境でサポートされます (CICS は、DFHDLIAL で CBLTDLI 入り口点を提供します)。

他の言語環境プログラム・サービスまたはユーザー作成の言語環境プログラム条件ハンドラーへの明示的な呼び出しを含んでいるバッチまたは IMS DC アプリケーションがある場合、すべての IMS インターフェースで CBLTDLI ではなく CEETDLI を使用しなければなりませんか ?

いいえ。プログラムまたは実行単位内のすべての呼び出しが CEETDLI である必要はありません。例外は、AIBTDLI インターフェースを使用している現行アプリケーションがある場合です。AIBTDLI は CEETDLI に変更してください。これは、CEETDLI が、ESTAE 処理の効率を高め、呼び出しを AIBTDLI から CEETDLI に変更すること以外に論理の変更を必要としないためです。

言語環境プログラム (および、COBOL プログラムと PL/I プログラムの混合に対する言語環境プログラムのサポート) は、COBOL プログラムが CBLTDLI を使用する場合に、PL/I と VS COBOL II (または Enterprise COBOL) を含んでいるアプリケーションを引き続きサポートしますか、それともそのようなプログラムは CEETDLI に移行しなければなりませんか ?

IMS の観点からは混合環境に問題はなく、プログラムを変更する必要はありません。移行の目的では、CBLTDLI と CEETDLI は同等であると見なしてください。

言語環境プログラムのもとでは、COBOL プログラムは引き続き CBLTDLI インターフェースを使用することができます。言語環境プログラムのもとでは、OS/VS COBOL と PL/I の混合は認められないため、プログラムは VS COBOL II または Enterprise COBOL でなければならないことに注意してください。CEETDLI が CICS 環境でサポートされない点を除き、CBLTDLI と CEETDLI のどちらかを使用することができます。

IMS のもとで CBLTDLI インターフェースを使用するときには、TRAP(OFF) ランタイム・オプションを指定する必要がありますか ?

いいえ。TRAP(OFF) は COBOL プログラムについては推奨されません。IMS のもとで CBLTDLI を使用するときには、言語環境プログラム条件処理を使用できない場合がいくつかあります。ただし、ABTERMENC(ABEND) を指定すると、重大エラー条件が発生した場合にデータベースのロールバックが自動的に実行されます。詳細は、*言語環境プログラム プログラミング・ガイド* を参照してください。

現在、CICS で OS/VS COBOL と VS COBOL II の両方のプログラムを実行しています。すべての VS COBOL II プログラムは AMODE 31 です。AMODE 24 の

OS/VS COBOL プログラムがあるため、言語環境プログラムのランタイム・オプション ALL31(OFF) を指定して実行しなければならないでしょうか ?

すべての VS COBOL II プログラムが AMODE 31 である場合は、ALL31(ON) を指定して実行することができます。CICS のもとでは、OS/VS COBOL プログラムは独自の特殊な互換環境で稼働し、言語環境プログラムのランタイム・オプションの影響を受けません。

IGZEDT4 は言語環境プログラムで提供されますか ?

はい。

OS/390

OS/390 では、言語環境プログラムを LNKLST に置く必要がありますか ?

いいえ。ただし、言語環境プログラムは OS/390 と同じゾーンにインストールしなければなりません。言語環境プログラムを LNKLST に置かない場合は、言語環境プログラムを必要とするそれぞれの OS/390 PROC で言語環境プログラムを STEPLIB で指定する必要があります。

どのエレメントが言語環境プログラムを必要とするかについては、以下の資料を参照してください。

- OS/390 リリース 10 または z/OS の場合は、OS/390 Program Directory
- OS/390 リリース 1、2、および 3 の場合は、情報 APAR II10425

OS/VS COBOL プログラムは OS/390 の言語環境プログラム・エレメントと共に稼働することができますか ?

はい。ただし、場合によっては、言語環境プログラムとリンク・エディットすることが必要です。その他の要因も適用される場合があります。詳細については、以下の章を参照してください。

- 59 ページの『第 5 章 言語環境プログラムのもとでの既存のアプリケーションの実行』
- 73 ページの『第 6 章 OS/VS COBOL ランタイムからの移行』

z/OS

OS/390 から z/OS に移行するために COBOL アプリケーションを再コンパイルまたは再リンクする必要がありますか ?

いいえ。OS/390 から z/OS への移行は、OS/390 のあるリリースから他のリリースに移行する場合と同様です。COBOL アプリケーションは、OS/390 リリース 10 で実行する場合と同様に、変更を加えることなく z/OS で実行できます。

COBOL は 64 ビットの z/OS で稼動しますか ?

はい。COBOL は COBOL プログラムで 64 ビットのアドレスをサポートしませんが、64 ビットの z/OS に移行することによってその利点の一部を得ることができます。64 ビットのアドレッシング可能な実メモリーで仮想メモリーをバックアップすると、ページングおよびスワッピングの回数が減るためシステム・パフォーマンス

が向上するうえ、プログラムを変更する必要がまったくありません。さらに、DB2 は COBOL プログラムを一切変更せずに COBOL プログラムの SQL ステートメントに 64 ビット・アドレッシングを利用できます。

z/OS システムを 64 ビット・モードで実行している場であっても、既存の AMODE 24 および AMODE 31 アプリケーションを再リンクまたは再コンパイルせずに実行できます。アプリケーションに変更を加えることなくシステム・パフォーマンスを改善することができます。

パフォーマンス

OS/VS COBOL から Enterprise COBOLに移行すると、CPU の負荷を減らすことができますか ?

OS/VS COBOL または VS COBOL II と比較したときの Enterprise COBOL のパフォーマンスは、アプリケーションの特性によって異なります。Enterprise COBOL のパフォーマンスについての情報は、以下の Web サイトの Library Section にあります。

www.ibm.com/software/ad/cobol

IBM COBOL と Enterprise COBOL のパフォーマンスはほとんど同じです。静的呼び出しと動的呼び出しについては COBOL (MVS および VM 版) および COBOL (OS/390 および VM 版) 以降、TRUNC(BIN) を指定してコンパイルされたプログラムについては COBOL for OS/390 以降でパフォーマンスが向上する可能性があります。

サービス

アプリケーションについて IBM サービス・サポートを得るためには、すべてのプログラムを再コンパイルする必要がありますか ?

プログラムが、サポートされるランタイムと共に稼働している限り、引き続き IBM サービス・サポートを得るためにプログラムを再コンパイルする必要はありません。詳細については、5 ページの『OS/VS COBOL および VS COBOL II プログラム用のサービス・サポート』を参照してください。

付録 B. COBOL 予約語の比較

この付録には、OS/VS COBOL、VS COBOL II、IBM COBOL、および Enterprise COBOL の間の予約語の違いを示す表が記載されています。ソース言語の比較に関する情報は、以下の場所に記載されています。

- 131 ページの『第 10 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 181 ページの『第 12 章 VS COBOL II ソース・プログラムのアップグレード』
- 191 ページの『第 14 章 IBM COBOL ソース・プログラムのアップグレード』

このリストでは、Enterprise COBOL、IBM COBOL、VS COBOL II、および OS/VS COBOL の予約語を示します。

注: Enterprise COBOL の予約語は、"Enterprise COBOL" と示された列に記載されています。IBM COBOL 以降に追加された新しい予約語 (将来の開発のために予約されている新しいワードを除く) は、**太文字**で強調表示されています。

キー:

X このワードはプロダクトで予約されています。

X* IBM COBOL 列では、そのワードは COBOL (OS/390 および VM 版) バージョン 2 リリース 2 でのみ予約されています。バージョン 2 リリース 1 以前では予約されていません。

- このワードはプロダクトで予約されていません (これには、フラグが立てられなくなった、廃止された予約語が含まれます)。

CDW このワードは、Enterprise COBOL コンパイラー指示ステートメントです。ユーザー定義語として使用された場合、重大メッセージでフラグが立てられます。

RFD このワードは、将来の開発のために予約されています。使用された場合、通知メッセージでフラグが立てられます。

SYS このワードは、オペレーティング・システムに対して特定の意味を持つワードです。プログラム内の特定のコンテキストでのみ使用できます。

UNS このワードは、このコンパイラーによってサポートされない機能のための COBOL 1985 標準予約語です。これらの予約語の中には、報告書作成プログラム・プリコンパイラーによってフィーチャーがサポートされるものもあります。プログラムで使用された場合、予約語として認識され、重大メッセージでフラグが立てられます。

表 48. 予約語の比較

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
ACCEPT	X	X	X	X
ACCESS	X	X	X	X
ACTIVE-CLASS	RFD	-	-	-
ACTUAL	-	-	-	X
ADD	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
ADDRESS	X	X	X	X
ADVANCING	X	X	X	X
AFTER	X	X	X	X
ALIGNED	RFD	-	-	-
ALL	X	X	X	X
ALLOCATE	RFD	-	-	-
ALPHABET	X	X	X	-
ALPHABETIC	X	X	X	X
ALPHABETIC-LOWER	X	X	X	-
ALPHABETIC-UPPER	X	X	X	-
ALPHANUMERIC	X	X	X	-
ALPHANUMERIC-EDITED	X	X	X	-
ALSO	X	X	X	X
ALTER	X	X	X	X
ALTERNATE	X	X	X	X
AND	X	X	X	X
ANY	X	X	X	-
ANYCASE	RFD	-	-	-
APPLY	X	X	X	X
ARE	X	X	X	X
AREA	X	X	X	X
AREAS	X	X	X	X
ARITHMETIC	-	RFD	RFD	-
AS	RFD	-	-	-
ASCENDING	X	X	X	X
ASSIGN	X	X	X	X
AT	X	X	X	X
AUTHOR	X	X	X	X
AUTOMATIC	RFD	-	-	-
B-AND	RFD	RFD	RFD	-
B-EXOR	-	RFD	RFD	-
B-LESS	-	RFD	RFD	-
B-NOT	RFD	RFD	RFD	-
B-OR	RFD	RFD	RFD	-
B-XOR	RFD	-	-	-
BASED	RFD	-	-	-
BASIS	CDW	CDW	CDW	X
BEFORE	X	X	X	X
BEGINNING	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
BINARY	X	X	X	-
BINARY-CHAR	RFD	-	-	-
BINARY-DOUBLE	RFD	-	-	-
BINARY-LONG	RFD	-	-	-
BINARY-SHORT	RFD	-	-	-
BIT	RFD	RFD	RFD	-
BITS	-	RFD	RFD	-
BLANK	X	X	X	X
BLOCK	X	X	X	X
BOOLEAN	RFD	RFD	RFD	-
BOTTOM	X	X	X	X
BY	X	X	X	X
CALL	X	X	X	X
CANCEL	X	X	X	X
CBL	CDW	CDW	CDW	X
CD	UNS	UNS	UNS	X
CF	UNS	UNS	UNS	X
CH	UNS	UNS	UNS	X
CHANGED	-	-	-	X
CHARACTER	X	X	X	X
CHARACTERS	X	X	X	X
CLASS	X	X	X	-
CLASS-ID	X	X	-	-
CLOCK-UNITS	UNS	UNS	UNS	-
CLOSE	X	X	X	X
COBOL	X	X	X	-
CODE	X	X	X	X
CODE-SET	X	X	X	X
COL	RFD	-	-	-
COLLATING	X	X	X	X
COLS	RFD	-	-	-
COLUMN	UNS	UNS	UNS	X
COLUMNS	RFD	-	-	-
COM-REG	X	X	X	-
COMMA	X	X	X	X
COMMIT	-	RFD	RFD	-
COMMON	X	X	X	-
COMMUNICATION	UNS	UNS	UNS	X
COMP	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
COMP-1	X	X	X	X
COMP-2	X	X	X	X
COMP-3	X	X	X	X
COMP-4	X	X	X	X
COMP-5	X	X*	RFD	-
COMP-6	-	RFD	RFD	-
COMP-7	-	RFD	RFD	-
COMP-8	-	RFD	RFD	-
COMP-9	-	RFD	RFD	-
COMPUTATIONAL	X	X	X	X
COMPUTATIONAL-1	X	X	X	X
COMPUTATIONAL-2	X	X	X	X
COMPUTATIONAL-3	X	X	X	X
COMPUTATIONAL-4	X	X	X	X
COMPUTATIONAL-5	X	X*	RFD	
COMPUTATIONAL-6	-	RFD	RFD	-
COMPUTATIONAL-7	-	RFD	RFD	-
COMPUTATIONAL-8	-	RFD	RFD	-
COMPUTATIONAL-9	-	RFD	RFD	-
COMPUTE	X	X	X	X
CONDITION	RFD	-	-	-
CONFIGURATION	X	X	X	X
CONNECT	-	RFD	RFD	-
CONSOLE	SYS	SYS	SYS	X
CONSTANT	RFD	-	-	-
CONTAINED	-	RFD	RFD	-
CONTAINS	X	X	X	X
CONTENT	X	X	X	-
CONTINUE	X	X	X	-
CONTROL	UNS	UNS	UNS	X
CONTROLS	UNS	UNS	UNS	X
CONVERTING	X	X	X	-
COPY	CDW	CDW	CDW	X
CORR	X	X	X	X
CORR-INDEX	-	-	-	X
CORRESPONDING	X	X	X	X
COUNT	X	X	X	X
CRT	RFD	-	-	-
CSP	SYS	SYS	SYS	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
CURRENCY	X	X	X	X
CURRENT	-	RFD	RFD	-
CURRENT-DATE	-	-	-	X
CURSOR	RFD	-	-	-
C01	SYS	SYS	SYS	X
C02	SYS	SYS	SYS	X
C03	SYS	SYS	SYS	X
C04	SYS	SYS	SYS	X
C05	SYS	SYS	SYS	X
C06	SYS	SYS	SYS	X
C07	SYS	SYS	SYS	X
C08	SYS	SYS	SYS	X
C09	SYS	SYS	SYS	X
C10	SYS	SYS	SYS	X
C11	SYS	SYS	SYS	X
C12	SYS	SYS	SYS	X
DATA	X	X	X	X
DATA-POINTER	RFD	-	-	-
DATE	X	X	X	X
DATE-COMPILED	X	X	X	X
DATE-WRITTEN	X	X	X	X
DAY	X	X	X	X
DAY-OF-WEEK	X	X	X	-
DB	-	RFD	RFD	-
DB-ACCESS-CONTROL-KEY	-	RFD	RFD	-
DB-DATA-NAME	-	RFD	RFD	-
DB-EXCEPTION	-	RFD	RFD	-
DB-RECORD-NAME	-	RFD	RFD	-
DB-SET-NAME	-	RFD	RFD	-
DB-STATUS	-	RFD	RFD	-
DBCS	X	X	X	-
DE	UNS	UNS	UNS	X
DEBUG	-	-	-	X
DEBUG-CONTENTS	X	X	X	X
DEBUG-ITEM	X	X	X	X
DEBUG-LINE	X	X	X	X
DEBUG-NAME	X	X	X	X
DEBUG-SUB-1	X	X	X	X
DEBUG-SUB-2	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
DEBUG-SUB-3	X	X	X	X
DEBUGGING	X	X	X	X
DECIMAL-POINT	X	X	X	X
DECLARATIVES	X	X	X	X
DEFAULT	RFD	RFD	RFD	-
DELETE	X	X	X	X
DELIMITED	X	X	X	X
DELIMITER	X	X	X	X
DEPENDING	X	X	X	X
DESCENDING	X	X	X	X
DESTINATION	UNS	UNS	UNS	X
DETAIL	UNS	UNS	UNS	X
DISABLE	UNS	UNS	UNS	X
DISCONNECT	-	RFD	RFD	-
DISP	-	-	-	X
DISPLAY	X	X	X	X
DISPLAY-ST	-	-	-	X
DISPLAY-1	X	X	X	-
DISPLAY-2	-	RFD	RFD	-
DISPLAY-3	-	RFD	RFD	-
DISPLAY-4	-	RFD	RFD	-
DISPLAY-5	-	RFD	RFD	-
DISPLAY-6	-	RFD	RFD	-
DISPLAY-7	-	RFD	RFD	-
DISPLAY-8	-	RFD	RFD	-
DISPLAY-9	-	RFD	RFD	-
DIVIDE	X	X	X	X
DIVISION	X	X	X	X
DOWN	X	X	X	X
DUPLICATE	-	RFD	RFD	-
DUPLICATES	X	X	X	X
DYNAMIC	X	X	X	X
EC	RFD	-	-	-
EGCS	X	X	X	-
EGI	UNS	UNS	UNS	X
EJECT	CDW	CDW	CDW	X
ELSE	X	X	X	X
EMI	UNS	UNS	UNS	X
EMPTY	-	RFD	RFD	-

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
ENABLE	UNS	UNS	UNS	X
END	X	X	X	X
END-ACCEPT	RFD	-	-	-
END-ADD	X	X	X	-
END-CALL	X	X	X	-
END-COMPUTE	X	X	X	-
END-DELETE	X	X	X	-
END-DISABLE	-	RFD	RFD	-
END-DISPLAY	RFD	-	-	-
END-DIVIDE	X	X	X	-
END-ENABLE	-	RFD	RFD	-
END-EVALUATE	X	X	X	-
END-EXEC	X	X*	-	-
END-IF	X	X	X	-
END-INVOKE	X	X	-	-
END-MULTIPLY	X	X	X	-
END-OF-PAGE	X	X	X	X
END-PERFORM	X	X	X	-
END-READ	X	X	X	-
END-RECEIVE	UNS	UNS	UNS	-
END-RETURN	X	X	X	-
END-REWRITE	X	X	X	-
END-SEARCH	X	X	X	-
END-SEND	-	RFD	RFD	-
END-START	X	X	X	-
END-STRING	X	X	X	-
END-SUBTRACT	X	X	X	-
END-TRANSCEIVE	-	RFD	RFD	-
END-UNSTRING	X	X	X	-
END-WRITE	X	X	X	-
END-XML	X	-	-	-
ENDING	X	X	X	X
ENTER	X	X	X	X
ENTRY	X	X	X	X
ENVIRONMENT	X	X	X	X
EO	RFD	-	-	-
EOP	X	X	X	X
EQUAL	X	X	X	X
EQUALS	-	RFD	RFD	-

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
ERASE	-	RFD	RFD	-
ERROR	X	X	X	X
ESI	UNS	UNS	UNS	X
EVALUATE	X	X	X	-
EVERY	X	X	X	X
EXACT	-	RFD	RFD	-
EXAMINE	-	-	-	X
EXCEEDS	-	RFD	RFD	-
EXCEPTION	X	X	X	X
EXCEPTION-OBJECT	RFD	-	-	-
EXCLUSIVE	-	RFD	RFD	-
EXEC	X	X*	-	-
EXECUTE	X	X*	-	-
EXHIBIT	-	-	-	X
EXIT	X	X	X	X
EXTEND	X	X	X	X
EXTERNAL	X	X	X	-
FACTORY	X	X*	-	-
FALSE	X	X	X	-
FD	X	X	X	X
FETCH	-	RFD	RFD	-
FILE	X	X	X	X
FILE-CONTROL	X	X	X	X
FILE-LIMIT	-	-	-	X
FILE-LIMITS	-	-	-	X
FILLER	X	X	X	X
FINAL	UNS	UNS	UNS	X
FIND	-	RFD	RFD	-
FINISH	-	RFD	RFD	-
FIRST	X	X	X	X
FLOAT-EXTENDED	RFD	-	-	-
FLOAT-LONG	RFD	-	-	-
FLOAT-SHORT	RFD	-	-	-
FOOTING	X	X	X	X
FOR	X	X	X	X
FORMAT	RFD	RFD	RFD	-
FREE	RFD	RFD	RFD	-
FROM	X	X	X	X
FUNCTION	X	X	-	-

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
FUNCTION-ID	RFD	-	-	-
FUNCTION-POINTER	X	-	-	-
GENERATE	UNS	UNS	UNS	X
GET	RFD	RFD	RFD	-
GIVING	X	X	X	X
GLOBAL	X	X	X	-
GO	X	X	X	X
GOBACK	X	X	X	X
GREATER	X	X	X	X
GROUP	UNS	UNS	UNS	X
GROUP-USAGE	RFD	-	-	-
HEADING	UNS	UNS	UNS	X
HIGH-VALUE	X	X	X	X
HIGH-VALUES	X	X	X	X
I-O	X	X	X	X
I-O-CONTROL	X	X	X	X
ID	X	X	X	X
IDENTIFICATION	X	X	X	X
IF	X	X	X	X
IN	X	X	X	X
INDEX	X	X	X	X
INDEX-1	-	RFD	RFD	-
INDEX-2	-	RFD	RFD	-
INDEX-3	-	RFD	RFD	-
INDEX-4	-	RFD	RFD	-
INDEX-5	-	RFD	RFD	-
INDEX-6	-	RFD	RFD	-
INDEX-7	-	RFD	RFD	-
INDEX-8	-	RFD	RFD	-
INDEX-9	-	RFD	RFD	-
INDEXED	X	X	X	X
INDICATE	UNS	UNS	UNS	X
INHERITS	X	X	-	-
INITIAL	X	X	X	X
INITIALIZE	X	X	X	X
INITIATE	UNS	UNS	UNS	X
INPUT	X	X	X	X
INPUT-OUTPUT	X	X	X	X
INSERT	CDW	CDW	CDW	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
INSPECT	X	X	X	X
INSTALLATION	X	X	X	X
INTERFACE	RFD	-	-	-
INTERFACE-ID	RFD	-	-	-
INTO	X	X	X	X
INVALID	X	X	X	X
INVOKE	X	X	-	-
IS	X	X	X	X
JNIENVPTR	X	-	-	-
JUST	X	X	X	X
JUSTIFIED	X	X	X	X
KANJI	X	X	X	-
KEEP	-	RFD	RFD	-
KEY	X	X	X	X
LABEL	X	X	X	X
LAST	UNS	UNS	UNS	X
LD	-	RFD	RFD	-
LEADING	X	X	X	X
LEAVE	-	-	-	X
LEFT	X	X	X	X
LENGTH	X	X	X	X
LESS	X	X	X	X
LIMIT	UNS	UNS	UNS	X
LIMITS	UNS	UNS	UNS	X
LINAGE	X	X	X	X
LINAGE-COUNTER	X	X	X	X
LINE	X	X	X	X
LINE-COUNTER	UNS	UNS	UNS	X
LINES	X	X	X	X
LINKAGE	X	X	X	X
LOCALLY	-	RFD	RFD	-
LOCAL-STORAGE	X	X	-	-
LOCALE	RFD	-	-	-
LOCK	X	X	X	X
LOW-VALUE	X	X	X	X
LOW-VALUES	X	X	X	X
MEMBER	-	RFD	RFD	-
MEMORY	X	X	X	X
MERGE	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
MESSAGE	UNS	UNS	UNS	X
METACLASS	-	X	-	-
METHOD	X	X	-	-
METHOD-ID	X	X	-	-
MINUS	RFD	-	-	-
MODE	X	X	X	X
MODIFY	-	RFD	RFD	-
MODULES	X	X	X	X
MORE-LABELS	X	X	X	X
MOVE	X	X	X	X
MULTIPLE	X	X	X	X
MULTIPLY	X	X	X	X
NAMED	-	-	-	X
NATIONAL	X	-	-	-
NATIONAL-EDITED	RFD	-	-	-
NATIVE	X	X	X	X
NEGATIVE	X	X	X	X
NESTED	RFD	-	-	-
NEXT	X	X	X	X
NO	X	X	X	X
NOMINAL	-	-	-	X
NONE	-	RFD	RFD	-
NOT	X	X	X	X
NOTE	-	-	-	X
NULL	X	X	X	-
NULLS	X	X	X	-
NUMBER	UNS	UNS	UNS	X
NUMERIC	X	X	X	X
NUMERIC-EDITED	X	X	X	-
OBJECT	X	X	-	-
OBJECT-COMPUTER	X	X	X	X
OBJECT-REFERENCE	RFD	-	-	-
OCCURS	X	X	X	X
OF	X	X	X	X
OFF	X	X	X	X
OMITTED	X	X	X	X
ON	X	X	X	X
ONLY	-	RFD	RFD	-
OPEN	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
OPTIONAL	X	X	X	X
OPTIONS	RFD	-	-	-
OR	X	X	X	X
ORDER	X	X	X	-
ORGANIZATION	X	X	X	X
OTHER	X	X	X	-
OTHERWISE	-	-	-	X
OUTPUT	X	X	X	X
OVERFLOW	X	X	X	X
OVERRIDE	X	X	-	-
OWNER	-	RFD	RFD	-
PACKED-DECIMAL	X	X	X	-
PADDING	X	X	X	-
PAGE	X	X	X	X
PAGE-COUNTER	UNS	UNS	UNS	X
PARAGRAPH	-	RFD	RFD	-
PASSWORD	X	X	X	X
PERFORM	X	X	X	X
PF	UNS	UNS	UNS	X
PH	UNS	UNS	UNS	X
PIC	X	X	X	X
PICTURE	X	X	X	X
PLUS	UNS	UNS	UNS	X
POINTER	X	X	X	X
POSITION	X	X	X	X
POSITIONING	-	-	-	X
POSITIVE	X	X	X	X
PRESENT	RFD	RFD	RFD	-
PREVIOUS	RFD	RFD	-	-
PRINT-SWITCH	-	-	-	X
PRINTING	UNS	UNS	UNS	-
PRIOR	-	RFD	RFD	-
PROCEDURE	X	X	X	X
PROCEDURE-POINTER	X	X	-	-
PROCEDURES	X	X	X	X
PROCEED	X	X	X	X
PROCESSING	X	X	X	X
PROGRAM	X	X	X	X
PROGRAM-ID	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
PROGRAM-POINTER	RFD	-	-	-
PROPERTY	RFD	-	-	-
PROTECTED	-	RFD	RFD	-
PROTOTYPE	RFD	-	-	-
PURGE	UNS	UNS	UNS	-
QUEUE	UNS	UNS	UNS	X
QUOTE	X	X	X	X
QUOTES	X	X	X	X
RAISE	RFD	-	-	-
RAISING	RFD	-	-	-
RANDOM	X	X	X	X
RD	UNS	UNS	UNS	X
READ	X	X	X	X
READY	X	X	X	X
REALM	-	RFD	RFD	-
RECEIVE	UNS	UNS	UNS	X
RECONNECT	-	RFD	RFD	-
RECORD	X	X	X	X
RECORD-NAME	-	RFD	RFD	-
RECORD-OVERFLOW	-	-	-	X
RECORDING	X	X	X	X
RECORDS	X	X	X	X
RECURSIVE	X	X	-	-
REDEFINES	X	X	X	X
REEL	X	X	X	X
REFERENCE	X	X	X	-
REFERENCES	X	X	X	X
RELATION	-	RFD	RFD	-
RELATIVE	X	X	X	X
RELEASE	X	X	X	X
RELOAD	X	X	X	X
REMAINDER	X	X	X	X
REMARKS	-	-	-	X
REMOVAL	X	X	X	X
RENAMES	X	X	X	X
REORG-CRITERIA	-	-	-	X
REPEATED	-	RFD	RFD	-
REPLACE	X	X	X	-
REPLACING	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
REPORT	UNS	UNS	UNS	X
REPORTING	UNS	UNS	UNS	X
REPORTS	UNS	UNS	UNS	X
REPOSITORY	X	X	-	-
REREAD	-	-	-	X
RERUN	X	X	X	X
RESERVE	X	X	X	X
RESET	X	X	X	X
RESUME	RFD	-	-	-
RETAINING	-	RFD	RFD	-
RETRIEVAL	-	RFD	RFD	-
RETRY	RFD	-	-	-
RETURN	X	X	X	X
RETURN-CODE	X	X	X	X
RETURNING	X	X	-	-
REVERSED	X	X	X	X
REWIND	X	X	X	X
REWRITE	X	X	X	X
RF	UNS	UNS	UNS	X
RH	UNS	UNS	UNS	X
RIGHT	X	X	X	X
ROLLBACK	-	RFD	RFD	-
ROUNDED	X	X	X	X
RUN	X	X	X	X
SAME	X	X	X	X
SCREEN	RFD	-	-	-
SD	X	X	X	X
SEARCH	X	X	X	X
SECTION	X	X	X	X
SECURITY	X	X	X	X
SEEK	-	-	-	X
SEGMENT	UNS	UNS	UNS	X
SEGMENT-LIMIT	X	X	X	X
SELECT	X	X	X	X
SELECTIVE	-	-	-	X
SELF	X	X	-	-
SEND	UNS	UNS	UNS	X
SENTENCE	X	X	X	X
SEPARATE	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
SEQUENCE	X	X	X	X
SEQUENTIAL	X	X	X	X
SERVICE	X	X	X	X
SESSION-ID	-	RFD	RFD	-
SET	X	X	X	X
SHARED	-	RFD	RFD	-
SHARING	RFD	-	-	-
SHIFT-IN	X	X	X	-
SHIFT-OUT	X	X	X	-
SIGN	X	X	X	X
SIZE	X	X	X	X
SKIP-1	-	-	-	X
SKIP-2	-	-	-	X
SKIP-3	-	-	-	X
SKIP1	CDW	CDW	CDW	-
SKIP2	CDW	CDW	CDW	-
SKIP3	CDW	CDW	CDW	-
SORT	X	X	X	X
SORT-CONTROL	X	X	X	-
SORT-CORE-SIZE	X	X	X	X
SORT-FILE-SIZE	X	X	X	X
SORT-MERGE	X	X	X	X
SORT-MESSAGE	X	X	X	X
SORT-MODE-SIZE	X	X	X	X
SORT-RETURN	X	X	X	X
SOURCE	UNS	UNS	UNS	X
SOURCE-COMPUTER	X	X	X	X
SOURCES	RFD	-	-	-
SPACE	X	X	X	X
SPACES	X	X	X	X
SPECIAL-NAMES	X	X	X	X
SQL	X	X*	-	-
STANDARD	X	X	X	X
STANDARD-1	X	X	X	X
STANDARD-2	X	X	X	-
STANDARD-3	-	RFD	RFD	-
STANDARD-4	-	RFD	RFD	-
START	X	X	X	X
STATUS	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
STOP	X	X	X	X
STORE	-	RFD	RFD	-
STRING	X	X	X	X
SUB-QUEUE-1	UNS	UNS	UNS	X
SUB-QUEUE-2	UNS	UNS	UNS	X
SUB-QUEUE-3	UNS	UNS	UNS	X
SUB-SCHEMA	RFD	RFD	RFD	-
SUBTRACT	X	X	X	X
SUM	UNS	UNS	UNS	X
SUPER	X	X	-	-
SUPPRESS	X	X	X	X
SYMBOLIC	X	X	X	X
SYNC	X	X	X	X
SYNCHRONIZED	X	X	X	X
SYSIN	SYS	SYS	SYS	X
SYSIPT	SYS	SYS	SYS	-
SYSLIST	SYS	SYS	SYS	X
SYSLST	SYS	SYS	SYS	-
SYSOUT	SYS	SYS	SYS	X
SYSPCH	SYS	SYS	SYS	-
SYSPUNCH	SYS	SYS	SYS	X
SYSTEM-DEFAULT	RFD	-	-	-
S01	SYS	SYS	SYS	X
S02	SYS	SYS	SYS	X
S03	SYS	SYS	SYS	-
S04	SYS	SYS	SYS	-
S05	SYS	SYS	SYS	-
TABLE	UNS	UNS	UNS	X
TALLY	X	X	X	X
TALLYING	X	X	X	X
TAPE	X	X	X	X
TENANT	-	RFD	RFD	-
TERMINAL	UNS	UNS	UNS	X
TERMINATE	UNS	UNS	UNS	X
TEST	X	X	X	-
TEXT	UNS	UNS	UNS	X
THAN	X	X	X	X
THEN	X	X	X	X
THROUGH	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
THRU	X	X	X	X
TIME	X	X	X	X
TIME-OF-DAY	-	-	-	X
TIMES	X	X	X	X
TITLE	CDW	CDW	CDW	-
TO	X	X	X	X
TOP	X	X	X	X
TOTALED	-	-	-	X
TOTALING	-	-	-	X
TRACE	X	X	X	X
TRACK-AREA	-	-	-	X
TRACK-LIMIT	-	-	-	X
TRACKS	-	-	-	X
TRAILING	X	X	X	X
TRANSCEIVE	-	RFD	RFD	-
TRANSFORM	-	-	-	X
TRUE	X	X	X	-
TYPE	X	X*	-	-
TYPEDEF	RFD	-	-	-
UNEQUAL	-	RFD	RFD	-
UNIT	X	X	X	X
UNIVERSAL	RFD	-	-	-
UNLOCK	RFD	-	-	-
UNSTRING	X	X	X	X
UNTIL	X	X	X	X
UP	X	X	X	X
UPDATE	RFD	RFD	RFD	-
UPON	X	X	X	X
UPSI-0	SYS	SYS	SYS	X
UPSI-1	SYS	SYS	SYS	X
UPSI-2	SYS	SYS	SYS	X
UPSI-3	SYS	SYS	SYS	X
UPSI-4	SYS	SYS	SYS	X
UPSI-5	SYS	SYS	SYS	X
UPSI-6	SYS	SYS	SYS	X
UPSI-7	SYS	SYS	SYS	X
USAGE	X	X	X	X
USAGE-MODE	-	RFD	RFD	-
USE	X	X	X	X

表 48. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
USER-DEFAULT	RFD	-	-	-
USING	X	X	X	X
VAL-STATUS	RFD	-	-	-
VALID	RFD	RFD	RFD	-
VALIDATE	RFD	RFD	RFD	-
VALIDATE-STATUS	RFD	-	-	-
VALUE	X	X	X	X
VALUES	X	X	X	X
VARYING	X	X	X	X
WAIT	-	RFD	RFD	-
WHEN	X	X	X	X
WHEN-COMPILED	X	X	X	X
WITH	X	X	X	X
WITHIN	-	RFD	RFD	-
WORDS	X	X	X	X
WORKING-STORAGE	X	X	X	X
WRITE	X	X	X	X
WRITE-ONLY	X	X	X	X
XML	X	-	-	-
XML-CODE	X	-	-	-
XML-EVENT	X	-	-	-
XML-NTEXT	X	-	-	-
XML-TEXT	X	-	-	-
ZERO	X	X	X	X
ZEROES	X	X	X	X
ZEROS	X	X	X	X
<	X	X	X	X
<=	X	X	X	-
+	X	X	X	X
*	X	X	X	X
**	X	X	X	X
-	X	X	X	X
/	X	X	X	X
>	X	X	X	X
>=	X	X	X	-
=	X	X	X	X

付録 C. ソース・プログラム用の移行ツール

この付録には、実際の移行作業時に役立てることができる移行ツールが記載されています。このようなツールには、以下のものがあります。

- MIGR コンパイラー・オプション (OS/VS COBOL)
- 移行を援助するその他のプログラム

この付録は、使用すべき移行ツール (ある場合) を判別し、それらを使用する方法を理解し、移行ツールの出力を分析して残りの移行処置の程度を評価する方法を理解するのに役立ちます。

注: これらすべての移行ツールでは、移行するプログラムが有効な OS/VS COBOL、VS COBOL II、または IBM COBOL プログラムである (つまり、*IBM VS COBOL for OS/VS*、*VS COBOL II Application Programming Language Reference*、および *COBOL for OS/390 & VM Language Reference* に記載されている規則に従って作成され、文書化されていない拡張を使用しないプログラムである) ことを前提としています。

MIGR コンパイラー・オプション

OS/VS COBOL プログラムを、Enterprise COBOL に移行することを計画しているときは、OS/VS COBOL の MIGR コンパイラー・オプションを使用することができます。このオプションは、移行処置の程度を理解するのに役立ちます。さらに、このオプションは、Enterprise COBOL によってサポートされない OS/VS COBOL ソース言語の使用を避けるのに役立つため、計画されている将来の移行を容易に行えるようにすることができます。MIGR を使用してプログラムをコンパイルすることによって、移行しなければならない言語エレメントを事前に判別することができます。

非互換性は以下の領域にあります。

- 追加された COBOL 機能のために取り入れられた新しい予約語 (以前は有効であったユーザー・ワードが現在は不当である場合があります)
- 異なる方法でサポートされる言語機能
- サポートされなくなった言語機能

MIGR コンパイラー・オプションは、インストール時にインストール先デフォルトとして設定することもできますし、OS/VS COBOL プログラムのコンパイル時に設定することもできます。MIGR をオンに設定すると、コンパイラーは、Enterprise COBOL では変更されているかまたはサポートされないステートメントのほとんどにフラグを立てます。

言語の相違

Enterprise COBOL と OS/VS COBOL の間には、以下の言語の違いがあります。

- ALPHABETIC クラスの変更
- PICTURE 文節内の B 記号

- CALL ステートメントの変更
- CBL コンパイラー指示ステートメントの変更
- 簡略複合比較条件の変更
- DIVIDE ID1 BY ID2 [GIVING ID3] ON SIZE ERROR . . .
- DIVIDE ID1 INTO ID2 [GIVING ID3] ON SIZE ERROR . . .
- プログラムの終わりで欠落している EXIT PROGRAM (または STOP RUN)
- FILE STATUS 文節
- ID1 IS [NOT] ALPHABETIC

  (IF、PERFORM、および SEARCH でのクラス・テスト)
- IF . . . OTHERWISE ステートメントの変更
- MOVE A TO B

  B が、それ自身の ODO オブジェクトを含んでいる可変長データ項目として定義されている場合。
- MULTIPLY ID1 BY ID2 [GIVING ID3] ON SIZE ERROR . . .
- PERFORM P1 [THRU P2] VARYING ID2 FROM ID3 BY ID4 UNTIL COND-1 AFTER ID5 FROM ID6 BY ID7 UNTIL COND-2 AFTER ID8 FROM ID9 BY ID10 UNTIL COND-3

  1. ID6 が (潜在的に) ID-2 に依存する場合。
  2. ID9 が (潜在的に) ID-5 に依存する場合。
  3. ID4 が (潜在的に) ID-5 に依存する場合。
  4. ID7 が (潜在的に) ID-8 に依存する場合。

  依存関係は、最初の ID または指標名 (IDx) が、2 番目の ID と同じであるか、それによって添え字付けされているか、またはそれによって修飾されている場合に発生します。また、2 番目の ID の部分的または完全な再定義によって依存関係が発生することもあります。
- OCCURS DEPENDING ON 文節の変更
- ON SIZE ERROR オプション — 中間結果の変更
- PROGRAM COLLATING SEQUENCE 文節の変更
- READ filename RECORD INTO B

  B が、ODO 句のオブジェクトを含んでいる可変長データとして定義されている場合。
- FD セクション内の RECORD CONTAINS integer-4 CHARACTERS
- RERUN 文節の変更
- RESERVE 文節の変更
- 予約語リストの変更
- SPECIAL-NAMES: alphabet-name IS xxxxx
- 範囲外の添え字 — 評価の変更
- UNSTRING A INTO B . . .

  B が、ODO 句のオブジェクトを含んでいる可変長データとして定義されている場合。

- UNSTRING ID1 DELIMITED BY ID2 INTO ID4 DELIMITER IN ID5 COUNT IN ID6 WITH POINTER ID7
- UPSI スイッチおよび UPSI 簡略名の参照
- VALUE 文節の条件名
- WHEN-COMPILED 特殊レジスター
- WRITE BEFORE/AFTER ADVANCING PAGE ステートメント
- WRITE AFTER POSITIONING

より高い正確度でサポートされるステートメント

以下の OS/VS COBOL ステートメントは、Enterprise COBOL では正確度がより高くサポートされます。これらは、Enterprise COBOL ではより正確な結果を提供する可能性があることを示すメッセージによってフラグが立てられます。

算術ステートメント

- 浮動小数点データ項目の定義
- 浮動小数点リテラルの USAGE
- 指数の USAGE

サポートされない LANGLVL(1) ステートメント

LANGLVL(1) コンパイラー・オプションにのみ適用される以下の OS/VS COBOL ステートメントは、Enterprise COBOL ではサポートされず、MIGR コンパイラー・オプションが指定されるとフラグが立てられます。

- COPY 言語 — 1968
- VALUE を指定した JUSTIFIED|JUST 文節
- MOVE ステートメントおよび比較 — 位取りの変更
- 簡略複合比較条件内の NOT
- 独立セグメント内の PERFORM ステートメント
- RESERVE integer AREAS
- SELECT OPTIONAL 文節 — 1968 標準の解釈
- SPECIAL-NAMES 段落: L、/、および = の使用
- DELIMITED BY ALL を指定した UNSTRING

サポートされない LANGLVL(1) および LANGLVL(2) ステートメント

LANGLVL(1) と LANGLVL(2) の両方のコンパイラー・オプションに適用される以下の OS/VS COBOL ステートメントは、Enterprise COBOL ではサポートされず、MIGR コンパイラー・オプションが指定されるとフラグが立てられます。

通信

- COMMUNICATION SECTION
- ACCEPT MESSAGE
- SEND、RECEIVE、ENABLE、および DISABLE 動詞 (RECEIVE ...MESSAGE は LANGLVL の影響を受けますが、通信のもとでのみフラグが立てられることに注意してください)。

報告書作成プログラム:

- INITIATE、GENERATE、および TERMINATE 動詞
- LINE-COUNTER、PAGE-COUNTER、および PRINT-SWITCH 特殊レジスター
- SPECIAL NAMES 内の、非数値リテラル IS 簡略名
- FD の REPORT 文節
- REPORT SECTION ヘッダー
- USE BEFORE REPORTING 宣言

注: 報告書作成プログラム・プリコンパイラーは、これらのステートメントを変換することができます。 300 ページの『COBOL 報告書作成プログラム・プリコンパイラー』を参照してください。

ISAM:

- APPLY REORG-CRITERIA (ISAM)
- APPLY CORE-INDEX (ISAM)
- 入出力動詞 — ISAM ファイルを参照するすべてのもの
- ISAM ファイル宣言
- NOMINAL KEY 文節
- 編成パラメーター『I』
- TRACK-AREA 文節
- START ステートメントの USING KEY 文節

BDAM:

- ACTUAL KEY 文節
- APPLY RECORD-OVERFLOW (BDAM)
- BDAM ファイル宣言
- 入出力動詞 — BDAM ファイルを参照するすべてのもの
- 編成パラメーター 『D』、『R』、および『W』
- SEEK ステートメント
- TRACK-LIMIT 文節

デバッグ用の使用:

- USE FOR DEBUGGING ON [ALL REFERENCES OF] identifiers, file-names, cd-names

その他のステートメント:

- APPLY RECORD-OVERFLOW
- ASCII を示す割り当て名編成パラメーター『C』
- ASSIGN . . . OR
- ASSIGN TO integer system-name
- ASSIGN . . . FOR MULTIPLE REEL/UNIT
- CLOSE . . . WITH POSITIONING/DISP
- CURRENT-DATE および TIME-OF-DAY 特殊レジスター
- デバッグ・パケット
- EXAMINE ステートメント
- EXHIBIT ステートメント
- FILE-LIMITS
- TOTALING/TOTALED AREA オプションを指定した LABEL RECORDS 文節

- NOTE ステートメント
- ON ステートメント
- OPEN . . . LEAVE/REREAD/DISP
- 修飾された指標名

  (このサポートされない形式を使用すると、重大 (RC = 12) レベルのメッセージが生成されます。)
- READY TRACE および RESET TRACE ステートメント
- REMARKS 段落
- RESERVE NO/ALTERNATE AREAS
- KEY 項目をサブジェクトではなくオブジェクトとして使用する SEARCH . . . WHEN 条件
- SERVICE RELOAD ステートメント
- START . . . USING key ステートメント
- ステートメント結合子としての THEN
- TIME-OF-DAY 特殊レジスター
- TRANSFORM ステートメント
- USE AFTER STANDARD ERROR . . . GIVING
- USE BEFORE STANDARD LABEL
- CALL ステートメントでの USING プロシージャー名またはファイル名

移行をサポートするその他のプログラム

以下のセクションでは、移行作業に役立ついくつかの移行ツールを記述します。これらのプログラムには、以下のものがあります。

- ワークステーションでは、
  - 報告書作成プログラム
- ホストでは、
  - COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA)
  - CICS アプリケーション・マイグレーション・エイド
  - COBOL 報告書作成プログラム・プリコンパイラー
  - 取引先製品

報告書作成プログラム (OS/2 用および Windows 用)

VisualAge for COBOL のオプショナル機能 (個別に購入可能) である報告書作成プログラム (OS/2 用および Windows 用) は、汎用 COBOL 印刷機能を提供し、よく知られているホスト報告書作成プログラム機能をワークステーションで提供します。

詳細については、 300 ページの『COBOL 報告書作成プログラム・プリコンパイラー』を参照してください。

WebSphere Studio Asset Analyzer

WebSphere Studio Asset Analyzer は、企業資産の目録を生成し、コード変更に必要な相対労力の指標を戻すツールを提供します。

COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA)

COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA) (プロダクト番号 5648-B05) は、CICS および非 CICS ソース・コードを、Enterprise COBOL でコンパイルできるソース・コードに変換します。

CCCA の目的は、非互換ソース・コードの識別およびその Enterprise COBOL ソースへの変換を自動化することです。CCCA を使用すると、移行処置がかなり軽減されるはずです。

CCCA は、CICS プログラムを変換するときには、Enterprise COBOL、IBM COBOL、VS COBOL II、または OS/VS COBOL コンパイラーが使用可能になっていることを必要とします。

CCCA の主要な機能は次のとおりです。

- OS/VS COBOL または VS COBOL II プログラムと Enterprise COBOL プログラム間の構文の違いの大部分の変換
- OS/VS COBOL、VS COBOL II、および IBM COBOL のユーザー定義名と Enterprise COBOL の予約語との対立の除去
- 直接変換できない言語エレメントのフラグ設定
- ステートメント単位の診断リスト
- COPY ブックおよびファイルの使用場所報告書を含む、変換管理情報
- EXEC CICS コマンドの変換
- BLL (リンク用ベース・ロケーター) セクションのメカニズムおよび参照の除去または変換 (あるいはその両方)

CCCA は、部門の要件に合わせて調整することができるように設計されています。実行される変換を判別する CCCA LCP (言語変換プログラム) は、COBOL と類似した言語で作成されています。システムに提供された LCP を変更したり、独自のものを追加したりすることができます。

詳細については、*COBOL and CICS/VS Command Level Conversion Aid* を参照してください。

注: プログラムの移行とコードの構造化の両方を行う必要がある場合、COBOL/SF が使用可能であれば、COBOL/SF 内で、CCCA を活動化するオプションを選択することができます。CCCA がプログラムを変換したあと、COBOL/SF がそれを構造化します。

CCCA を使用すべきとき

アプリケーションを OS/VS COBOL または VS COBOL II から Enterprise COBOL に移行することを計画している場合には、CCCA が移行プロジェクトに役立つかどうかを評価してください。個々のプログラムに対して必要とされる変更の数が少ない可能性があっても、CCCA はそれらの変更を識別し、ほとんどの場合、それらを標準的な方式で自動的に変換します。CCCA は、CICS と非 CICS の両方のプログラムを移行します。CCCA は、SERVICE RELOAD ステートメントと、BLL セルのアドレッシングの複雑な論理を、Enterprise COBOL にとって有効なステートメントに変換します。

CCCA は、非 CICS 構文も処理します。

CICS ステートメントの CCCA 処理

CICS オプションが ON である場合、BLL 定義および SERVICE RELOAD ステートメントは除去されます。BLL 構造全体が再定義されている場合には、再定義された構造が除去されます。BLL が 4 バイトの長さで定義されていないと、CICS 変換を実行することができません。

1 次 BLL を扱うステートメントの変換によって必要とされる場合には、以下のコードが POINTER 機能による使用に向けて WORKING-STORAGE SECTION で生成されます。

```
77  LCP-WS-ADDR-COMP PIC S9(8) COMP.
77  LCP-WS-ADDR-PNTR REDEFINES LCP-WS-ADDR-COMP USAGE POINTER.
```

EXEC CICS の処理: SET オプションと共に使用される 1 次 BLL は、対応する ADDRESS OF 特殊レジスターで置き換えられます。以下に例を示します。

```
EXEC CICS READ ... SET(BLL1) ...
```

これは、次のように置き換えられます。

```
EXEC CICS READ ... SET(ADDRESS OF REC1) ...
```

関係するステートメントは次のとおりです。

CONVERSE	GETMAIN	ISSUE RECEIVE
LOAD	POST	READ
READNEXT	READPREV	READQ
RECEIVE	RETRIEVE	SEND CONTROL
SEND PAGE	SEND TEXT	

CICS の ADDRESS ステートメントと共に使用される 1 次 BLL は、Enterprise COBOL の対応する ADDRESS OF 特殊レジスターで置き換えられます。

以下に例を示します。

```
EXEC CICS TWA(BLL).
```

これは、次のように置き換えられます。

```
EXEC CICS TWA(ADDRESS OF TWA).
```

関係するオプションは、CSA、CWA、EIB、TCTUA、および TWA です。

1 次 BLL を扱うステートメント

表 49 に、1 次 BLL を扱うステートメントを示します。

2 次 BLL を扱うステートメントは、CONTINUE によって置き換えられます。

表 49. 1 次 BLL を扱う COBOL ステートメント

元のソース	変換後のソース
MOVE BLL1 TO BLL2	SET ADDRESS OF REC2 TO ADDRESS OF REC1
MOVE ID TO BLL	MOVE ID TO LCP-WS-ADDR-COMP SET ADDRESS OF REC1 TO LCP-WS-ADDR-PNTR

表 49. 1 次 BLL を扱う COBOL ステートメント (続き)

元のソース	変換後のソース
MOVE BLL TO ID	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC MOVE LCP-WS-ADDR-COMP TO ID
ADD ID1, .. TO BLL	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC ADD ID1, TO LCP-WS-ADDR-COMP SET ADDRESS OF REC TO LCP-WS-ADDR-PNTR
ADD BLL TO ID1, ID2	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC ADD LCP-WS-ADDR-COMP TO ID1, ID2
ADD ID1, ID2 GIVING BLL	ADD ID1, ID2 GIVING LCP-WS-ADDR-COMP SET ADDRESS OF REC TO LCP-WS-ADDR-PNTR
ADD ID, BLL1 GIVING BLL2 BLL3	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC ADD ID, LCP-WS-ADDR-COMP GIVING LCP-WS-ADDR-COMP SET ADDRESS OF REC2 TO LCP-WS-ADDR-PNTR SET ADDRESS OF REC3 TO LCP-WS-ADDR-PNTR
ADD ID1, BLL1 GIVING ID2 ID3	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC ADD ID1, LCP-WS-ADDR-COMP GIVING ID2 ID3
SUBTRACT ステートメント	変換は、ADD と同じ方法で行われます。
COMPUTE BLL = 式 (BLL)	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC COMPUTE LCP-WS-ADDR-COMP = 式 (LCP-WS-ADDR-COMP)
COMPUTE ID = 式 (BLL)	SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC COMPUTE ID = 式 (LCP-WS-ADDR-COMP)
COMPUTE BLL = 式 ...	COMPUTE LCP-WS-ADDR-COMP = 式 ...

CICS アプリケーション・マイグレーション・エイド

CICS アプリケーション・マイグレーション・エイド (プロダクト番号 5695-061) は、COBOL およびアセンブラー言語アプリケーション・プログラムの移行 (マクロ・レベル・アプリケーション・プログラミング・インターフェース (API) からコマンド・レベル API への移行) を単純化します。単純なマクロは自動的に変換されて、同等のコマンド・レベル機能を持つ新しいソース・コードが提供されます。複雑なマクロは部分的に変換され、変換を完了するのに役立つガイダンス情報が提供されます。いずれの場合も、元のソース・コードはまだ使用可能です。

コマンド・レベルは、CICS/ESA バージョン 4 以上で実行されるアプリケーションについての要件です。

COBOL 報告書作成プログラム・プリコンパイラー

報告書作成プログラム・プリコンパイラー (プロダクト番号 5798-DYR) には、2 つの機能があります。これは、報告書作成プログラム・ステートメントを含んでいるアプリケーションを、コードが Enterprise COBOL コンパイラーで受け入れられるようにプリコンパイルするために使用するか、あるいは報告書作成プログラム・ステートメントを有効な Enterprise COBOL ステートメントに永続的に変換するために使用することができます。

報告書作成プログラム・プリコンパイラーは、以下の機能を提供します。

- 拡張された報告書作成プログラム言語機能。
- ターゲット COBOL コンパイラーの自動呼び出し (ソース・プログラム内の報告書作成プログラム・ステートメントが COBOL コンパイラー自体によって処理されているかのように呼び出される)。
- プリコンパイラー・リストと COBOL コンパイラー・リストからの情報をマージして 1 つにまとめられたソース・リスト。
- COPY ライブラリー・メンバーが、報告書作成プログラム・ステートメントを含むことができます。
- Enterprise COBOL のネストされた COPY 機能をサポートします。
- 入力の報告書作成プログラム・ソース・ステートメントの診断検査を実行します。
- 独立型方式で稼働して、COBOL プログラム内の報告書作成プログラム・ステートメントを、Enterprise COBOL コンパイラーで受け入れられる非報告書作成プログラム COBOL ソース・ステートメントに変換することができます。

詳細については、*COBOL Report Writer Precompiler Programmer's Manual* および *COBOL Report Writer Precompiler Installation and Operation* を参照してください。

Edge Portfolio Analyzer

Edge Portfolio Analyzer は、既存の OS/VS COBOL および VS COBOL II ロード・モジュールの目録の作成に役立ちます。Edge Portfolio Analyzer は、以下のことを行うことができます。

- ロード・モジュールを作成した OS/VS COBOL コンパイラーまたは VS COBOL II コンパイラーのバージョンおよびリリースの判別
- ロード・モジュールのコンパイル時に指定されたコンパイラー・オプションの判別
- 現行のシステム日付を必要とするロード・モジュールの判別
- 置き換える必要がある CSECT (ILBOSRV など) の判別

注: Edge Portfolio Analyzer は現在 IBM では販売しておりませんが、Edge から直接購入することができます。詳しくは、Web サイト www.edge-information.com を参照してください。

取引先製品

IBM 以外のいくつかの移行ツールが、ソース・プログラムを Enterprise COBOL プログラムにアップグレードするとき、および言語環境プログラムに移行するときに役立ちます。IBM では、言語環境プログラムおよび Enterprise COBOL と共に動作可能な取引先製品のリストを *Language Environment Enabled Vendor Tools and Application Packages* 文書に編集しています。この情報を入手するには、Web サイト `http://www.ibm.com/s390/le` の Library リンクに進んでください。

付録 D. COBOL とアセンブラーを含んでいるアプリケーション

この章には、COBOL プログラムとアセンブラー・プログラムの混合を含んでいるアプリケーションについての情報が記載されています。以下の情報が記載されています。

- 呼び出し元および呼び出し先アセンブラー・プログラムについての要件の判別
- 非 CICS でサポートされるアセンブラー /COBOL 呼び出しの判別
- CICS でサポートされるアセンブラー /COBOL 呼び出しの判別
- 条件処理に ESTAE/ESPIE を使用するプログラムの移行
- プログラム・マスクを変更するプログラムの移行
- アセンブラー・ドライバーを使用するアプリケーションのアップグレード
- MVS ATTACH による COBOL プログラムの呼び出し
- COBOL プログラムをロードし、呼び出すアセンブラー・プログラム
- サブプールのストレージの解放
- プログラムの呼び出し — AMODE 要件

アセンブラー・プログラムと COBOL プログラムの両方を含んでいるアプリケーションに関するいくつかの情報が、本書の他の章に記載されています。表 50 に、アセンブラー・プログラムに関する追加情報と、それが記載されているページをリストします。

表 50. アセンブラー・プログラムと COBOL プログラムに関する追加情報

プログラム属性	ページ
LINK SVC を使用するアセンブラー・ルーチン	75 ページの『アップグレードが必要なプログラムの判別』 または 90 ページの『アップグレードが必要なプログラムの判別』
再使用可能環境で LINK SVC を使用するアセンブラー・ルーチン	100 ページの『ILBOSTP0 の使用』
プロシージャー名を渡すアセンブラー・プログラム	157 ページの 『OS/VS COBOL から変更された言語エレメント』
MVS のもとでファイルをクローズしないアセンブラー・プログラム	97 ページの『非 COBOL および OS/VS COBOL プログラムでのファイルのクローズ』
COBOL プログラムがアセンブラー・プログラムに戻るときの AMODE 動作	123 ページの『第 9 章 言語環境プログラムのリリース・レベルのアップグレード』

呼び出し元および呼び出し先アセンブラー・プログラムについての要件の判別

アセンブラー・プログラムが COBOL プログラムによって呼び出されるか、または COBOL プログラムを呼び出すときには、S/390 リンケージ規則に従わなければなりません。そうしなければ、アプリケーションは 40XX 異常終了コードで終了します。

呼び出し元アセンブラー・プログラム

アセンブラー・プログラムが呼び出し元であるときには、R13 は呼び出し元のレジスター保管域 (18 ワード) を指さなければならず、保管域の最初の 2 バイトはゼロでなければなりません。保管域の逆チェーンが有効な 31 ビット・アドレスで設定されなければなりません (すなわち、AMODE 24 での実行時には高位バイトをクリアしなければなりません)。

レジスター保管域アドレスを確立するための推奨される方法は、次のとおりです。

```
         LA   13, SAVEAREA
         .
         .
         .
SAVEAREA DS   18F
```

上記のようにすれば、AMODE 24 プログラムの場合は高位バイトがクリアされます。

分岐リンク・タイプの命令を使用して保管域をジャンプし、保管域のアドレスを設定する必要がある場合は、次の方法をお勧めします。

```
         BAS  13,SKIP
SAVEAREA DS   18F
SKIP     DS   0H
```

BAL 命令を使用する場合は、ご自分で次のようにして高位バイトをクリアする必要があります。

```
         BAL  13,SKIP
SAVEAREA DS   18F
SKIP     LA   13,0(,13)
```

この BAL 命令は、命令長コード、条件コード (CC)、およびプログラム・マスクをレジスターの高位バイトに入れます。BAL 命令ではなく BAS 命令を使用することをお勧めします。BAS 命令はゼロを高位バイトに入れて、24 ビット・アドレッシング問題を防止するからです。

プログラムがパラメーターを渡す場合は、パラメーター・リストを準備して、このリストのアドレスを R1 にロードしなければなりません。パラメーター・リストを渡さない場合は、R1 をゼロに設定してください。R14 にはアセンブラー・プログラム内の戻りアドレスを入れなければならず、R15 には COBOL プログラムの入り口点のアドレスを入れなければなりません。

注: パラメーター・リストを渡す場合、それは 1 つ以上の隣接するフルワード (それぞれに、COBOL プログラムに渡されるデータ項目のアドレスが入っている) のグループでなければなりません。最後のフルワード・アドレスの高位ビットを 1 に設定して、リストの終わりをフラグで示すことをお勧めします。

呼び出し先アセンブラー・プログラム

呼び出し先アセンブラー・プログラムは、レジスターと、COBOL プログラムによって渡されたその他の情報を保管域に保管しなければなりません。特に、COBOL 保管域は、アセンブラー・プログラムの保管域から正しく逆チェーンされなければなりません。また、アセンブラー・プログラムには、以下のことを行う戻りルーチンが含まれていなければなりません。

- COBOL 保管域のアドレスを R13 にロードする
- その他のレジスターの内容を復元する
- オプションで、R15 に戻りコードを設定する
- R14 内のアドレスに分岐する

SVC LINK および COBOL 実行単位の境界

SVC LINK のターゲットが非言語環境プログラム準拠のアセンブラー・プログラムであり、アセンブラー・プログラムがあとで COBOL プログラムを呼び出す場合は、言語環境プログラム・エンクレーブおよび COBOL 実行単位の境界は、アセンブラー・プログラムではなく COBOL プログラムにあります。エンクレーブ (および実行単位) のメインプログラムは COBOL プログラムです。

SVC LINK のターゲットが言語環境プログラム準拠のアセンブラー・プログラムである場合は、言語環境プログラム・エンクレーブの境界はアセンブラー・プログラムにあります。アセンブラー・プログラムがエンクレーブのメインプログラムです (CEEENTRY マクロで MAIN=YES が指定されている場合)。アセンブラー・プログラムがあとで COBOL プログラムを呼び出す場合、COBOL プログラムはサブプログラムです。

非 CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート

表 51 に、COBOL プログラムとアセンブラー・プログラムに関係のある呼び出しの可能な組み合わせをリストし、非 CICS での言語環境プログラムのもとでの実行時に呼び出しがサポートされるかどうかを示します。サポートされない 呼び出しの場合は、多くのケースで戻される症状 (メッセージまたは異常終了コード) もリストします。一部のケースでは (アプリケーション環境によっては)、症状が発生しない場合もあります。その場合は、別の障害が発生することもありますし、アプリケーションが正常に稼働しているように見えることもあります。

注: *IBM COBOL* という用語は、COBOL/370、COBOL (MVS および VM 版) および COBOL (OS/390 および VM 版) を指しています。

表 51. 非 CICS で実行される COBOL プログラムとアセンブラー (Asm) プログラム間の呼び出しに対する言語環境プログラム (LE) のサポート

呼び出し元			発行先					
発行側プログラム	呼び出しタイプ	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL	LE[1] Asm メイン	LE[1] Asm サブルーチン	非 LE Asm
Enterprise COBOL	静的	はい	はい	はい	はい	いいえ[2]	はい	はい
IBM COBOL		はい	はい	はい	はい	いいえ[2]	はい	はい
VS COBOL II		はい	はい	はい	はい	いいえ[2]	はい[3]	はい
OS/VS COBOL		はい	はい	はい	はい	いいえ[2]	はい[3]	はい

COBOL とアセンブラーの混合

表 51. 非 CICS で実行される COBOL プログラムとアセンブラー (Asm) プログラム間の呼び出しに対する言語環境プログラム (LE) のサポート (続き)

呼び出し元			発行先					
発行側プログラム	呼び出しタイプ	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL	LE[1] Asm メイン	LE[1] Asm サブルーチン	非 LE Asm
Enterprise COBOL	動的	はい	はい	はい	はい	いいえ[2]	はい	はい
IBM COBOL		はい	はい	はい	はい	いいえ[2]	はい	はい
VS COBOL II		はい	はい	はい	はい	いいえ[2]	はい	はい
OS/VS COBOL		はい	はい	はい	はい	いいえ[2]	はい	はい
Asm (LE)	VCON	はい	はい	はい	はい	いいえ[2]	はい	はい
Asm (非 LE)		はい	はい	はい	はい	はい[4]	いいえ[5]	はい
Asm (LE)	LOAD	はい	はい	はい	はい	いいえ[2]	はい	はい
Asm (非 LE)	BALR	はい	はい	はい	はい	はい[4]	いいえ[5]	はい
Asm (LE)	LINK	はい	はい	はい	はい[6]	はい	いいえ[5]	はい
Asm (非 LE)		はい	はい	はい	はい[6]	はい	いいえ[5]	はい

注: 以下に記述されている障害症状は、言語環境プログラムの TRAP(ON) および ABTERMENC(ABEND) ランタイム・オプションが有効であるときのものです。

1. MAIN=YES を指定した CEEENTRY マクロは、言語環境プログラム・アセンブラー・メインを作成します。CEEENTRY マクロで MAIN=NO を指定すると、言語環境プログラム・アセンブラー・サブルーチンが作成されます。デフォルトは MAIN=YES です。
2. 確立された言語環境プログラム・エンクレーブ内から言語環境プログラム・アセンブラー・メインプログラムを呼び出すことはお勧めしません (SVC LINK の使用を介する場合を除く)。このため、この脚注に関連した表項目は「いいえ」になっています。ネストされたエンクレーブは作成されず、したがって、プログラムは呼び出し側エンクレーブ内のサブプログラムとして稼動します。この勧告に従う場合には、将来の再プログラミングの必要性を避けることができます。
3. CEEENTRY マクロで NAB=NO および MAIN=NO を指定しなければなりません。指定しなければ、障害症状 0C1、0C4、または 0C5 異常終了を受け取ります。
4. 非言語環境プログラム・アセンブラー呼び出し元が、確立された言語環境プログラム・エンクレーブ内で稼動している場合は、注 2 を参照してください。
5. 障害症状 0C1、0C4、または 0C5 異常終了。

6. OS/VS COBOL プログラムが、別の確立された言語環境プログラム・エンクレーブに存在する場合を除きます。詳細については、障害症状: メッセージ IGZ0005S を参照してください。

CICS でのアセンブラー COBOL 呼び出しのためのランタイム・サポート

表 52 に、COBOL プログラムとアセンブラー・プログラムに関係のある呼び出しの可能な組み合わせをリストし、CICS のもとでの言語環境プログラムでの実行時に呼び出しがサポートされるかどうかを示します。サポートされない 呼び出しの場合は、多くのケースで戻される症状 (メッセージまたは異常終了コード) もリストします。一部のケースでは (アプリケーション環境によっては)、症状が発生しない場合もあります。その場合は、別の障害が発生することもありますし、アプリケーションが正常に稼働しているように見えることもあります。

注: *IBM COBOL* という用語は、COBOL/370、COBOL (MVS および VM 版)、および COBOL (OS/390 および VM 版) を指しています。

表 52. CICS で実行される COBOL プログラムとアセンブラー (Asm) プログラム間の呼び出しに対する言語環境プログラム (LE) のサポート。

呼び出し元		発行先						
発行側プログラム	呼び出しタイプ	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL	LE[1] Asm メイン	LE[1] Asm サブルーチン	非 LE Asm
Enterprise COBOL	静的	はい	はい	はい	いいえ[2]	いいえ[3]	はい	はい
IBM COBOL		はい	はい	はい	いいえ[2]	いいえ[3]	いいえ[4]	はい
VS COBOL II		はい	はい	はい	いいえ[2]	いいえ[3]	いいえ[4]	はい
OS/VS COBOL		いいえ[4]	いいえ[4]	いいえ[4]	はい	いいえ[4]	いいえ[4]	はい
Enterprise COBOL	動的	はい	はい	はい	いいえ[3]	いいえ[3]	はい	はい
IBM COBOL		はい	はい	はい	いいえ[3]	いいえ[3]	はい	はい
VS COBOL II		はい	はい	はい	いいえ[3]	いいえ[3]	はい	はい
OS/VS COBOL		いいえ[5]	いいえ[5]	いいえ[5]	いいえ[4]	いいえ[5]	いいえ[4]	いいえ[5]
Enterprise COBOL	EXEC CICS LINK	はい	はい	はい	はい	いいえ[3]	いいえ[4]	はい
IBM COBOL		はい	はい	はい	はい	いいえ[3]	いいえ[4]	はい
VS COBOL I		はい	はい	はい	はい	いいえ[3]	いいえ[4]	はい
OS/VS COBOL		はい	はい	はい	はい	いいえ[3]	いいえ[4]	はい
Asm (LE)	VCON	はい	はい	いいえ[4]	いいえ[4]	いいえ[3]	はい	はい
Asm (非 LE)		いいえ[4]	いいえ[4]	いいえ[4]	いいえ[4]	いいえ[3]	いいえ[4]	はい
Asm (LE)	EXEC CICS LINK	はい	はい	はい	はい	いいえ[3]	いいえ[4]	はい
Asm (非 LE)		はい	はい	はい	はい	いいえ[3]	いいえ[4]	はい

注: 以下の注釈に記述されている障害症状は、LE の TRAP(ON) および ABTERMENC(ABEND) ランタイム・オプションが有効であるときのものです。

1. MAIN=YES を指定した CEEENTRY マクロは、言語環境プログラム・アセンブラー・メインを作成します。CEEENTRY マクロで MAIN=NO を指定すると、言語環境プログラム・アセンブラー・サブルーチンが作成されます。デフォルトは MAIN=YES です。
2. 障害症状: メッセージ IGZ0079S。

3. CICS のもとでは、言語環境プログラム準拠のアセンブラー・メインプログラムについてのサポートはありません。障害症状: 予測不能。アプリケーションは正常に稼働しているように見える場合もあります。
4. 障害症状: ASRA 異常終了 (タイプ 1 または 4 のプログラム・チェックによって発生する)。
5. 障害症状: U3504 異常終了。

条件処理に ESTAE/ESPIE を使用するプログラムの移行

OS/VS COBOL および VS COBOL II アプリケーションは、アセンブラーで書かれたユーザー作成条件処理ルーチンを含むことができます。通常は、ESTAE を設定して ESTAE 出口を登録するためのアセンブラー・ルーチンをコーディングします (これがユーザーの ESTAE ルーチンです)。異常終了が発生すると、ESTAE 出口が制御を受け取ります (この章では ESTAE を参照しますが、この情報は、ESPIE を設定して ESPIE 出口を登録するアセンブラー・ルーチンにも適用されます)。

言語環境プログラムのもとでプログラムを実行しているときは、エラー、プログラム割り込み、および異常終了が発生すると、言語環境プログラムの条件マネージャーが制御を受け取ります。既存のユーザー作成条件処理ルーチンは、言語環境プログラムのもとでは作動しません。

既存のプログラム内のエラー処理ルーチン

OS/VS COBOL アプリケーションおよび VS COBOL II アプリケーションの場合、全体的なソースの移行は必要ありません。条件処理を確立または実行するプログラムだけを移行する必要があります。

ただし、アプリケーション・コードが OS/VS COBOL または VS COBOL II で保守されている場合は、自動スタック・フレーム縮小は行われません。条件処理ルーチンが制御を取得したあとでアプリケーションが再開する場合は、再入されるすべてのプログラムをそれらが再入される前に明示的に取り消す必要があります。

条件処理を確立または実行するプログラムを移行するには、次のようにします。

1. ESTAE ルーチンへの CALL を CEEHDLR (言語環境プログラム呼び出し可能サービス) への CALL で置き換えます。Enterprise COBOL プログラムだけが CEEHDLR を使用することができます。

   再開する必要があり、再開場所を制御したい場合は、SET RESUME POINT サービス (CEESRP) を使用してください。詳細については、*言語環境プログラム プログラミング・リファレンス* を参照してください。
2. ESTAE 出口を、独立した言語環境プログラム準拠ルーチン (Enterprise COBOL、言語環境プログラム準拠のアセンブラー、または言語環境プログラム準拠の C) にします。
3. 条件処理ルーチンの論理を変更します。言語環境プログラム・サービスを使用して、RESUME、PERCOLATE (別の処理ルーチンに制御を取らせる)、または PROMOTE (条件を別の条件に変更する) を行いたいかどうかを指示しなければなりません。さらに、言語環境プログラム・サービスを使用して、条件をもたらしたプログラムの名前を調べたり、条件に関連したエラー・メッセージを検索したりすることができます。

ユーザー作成条件処理ルーチンの確立

Enterprise COBOL の PROCEDURE-POINTER データ項目 (COBOL 85 標準への IBM 拡張) を SET ステートメントと共に使用することによって、言語環境プログラム提供の呼び出し可能サービスを使用する独自の条件処理ルーチンを確立することができます。ユーザー作成条件処理ルーチンは、言語環境プログラムのデフォルト条件処理ルーチンの前に制御を受け取ります。ユーザー作成条件処理ルーチンは、Enterprise COBOL、言語環境プログラム準拠の C、または言語環境プログラム準拠のアセンブラーで書くことができます。

ユーザー作成条件処理ルーチンの利点

言語環境プログラムでは、エラー処理後に実行を再開するポイントの管理が OS/VS COBOL または VS COBOL II の場合よりも非常に簡単です。OS/VS COBOL、VS COBOL II、および言語環境プログラムのもとでは、再帰呼び出しが許可されません。たとえば、OS/VS COBOL および VS COBOL II のもとでは、プログラム A がプログラム B を呼び出し、プログラム B 内でエラーが発生したために制御がプログラム A に戻されると、その後、プログラム A はプログラム B を呼び出すことができません (呼び出すと再帰エラーが発生するため)。したがって、条件処理ルーチンは、エラーを代行受信した後、エラーをもたらした命令の後の次に続く命令 (NSI) で再開しなければなりません。

OS/VS COBOL および VS COBOL II では、エラーをもたらしたプログラム以外のプログラムで再開するには、エラー発生時にアクティブであったプログラムと、ESTAE 出口が制御を受け取った後で再入される可能性のあるプログラムを取り消すことが必要です。

言語環境プログラムのもとで Enterprise COBOL または言語環境プログラム準拠のアセンブラー・プログラムを実行しているときには、再開ポイントを変更すると、言語環境プログラムが自動的にプログラムを非活動化します。言語環境プログラムのもとで稼動しているプログラムに上記の例を当てはめると、プログラム A 内で再開したあと、プログラム B を呼び出すことができます。言語環境プログラムはプログラム B を非活動化しているため、再帰は発生しません。

VS COBOL II プログラムの場合、再開カーソル移動機能 (言語環境プログラム呼び出し可能サービス CEEMRCR) によってスタック・フレームの縮小が起こります。ここで、アクティブであったプログラムが非アクティブにされ、再入できるようになります (ネストされたプログラムを使用する、NOCMPR2 を指定してコンパイルされた VS COBOL II プログラムを除く)。

再入できる VS COBOL II プログラムの場合、以下のことを行うことができます。

- 障害が発生したプログラムの NSI で再開する。
- 再開カーソルを、障害が発生したプログラムへの CALL ステートメントの後の命令に移動する。
- 再開カーソルを、障害が発生したプログラムの呼び出し元の呼び出し元の NSI に移動する。

プログラム・マスクを変更するプログラムの移行

VS COBOL II プログラムが、プログラム・マスクを変更する (たとえば、SPM 命令を使用する) アセンブラー・プログラムを呼び出す場合、アセンブラー・プログラムへの呼び出しのあとでプログラム・マスクは復元されます。

Enterprise COBOL では、プログラム・マスクは復元されません。したがって、アセンブラー・プログラムでプログラム・マスクを変更する場合は、COBOL プログラムに戻る前にプログラム・マスクを復元する必要があります。プログラム・マスクを復元しなければ、検出されないデータ・エラー (固定小数点オーバーフロー、10 進オーバーフロー、指数アンダーフロー、および有効数字例外など) が発生する可能性があります。

特定のプログラム・マスクを予期するアセンブラー・プログラムの呼び出し

VS COBOL II では、アセンブラー・プログラムが呼び出されたとき、プログラム・マスクは常に同じ設定値を持っていました。言語環境プログラムでは、COBOL から呼び出されたとき、プログラム・マスクは予測不能です。特定のプログラム・マスク要件を持つアセンブラー・プログラムは、プログラム・マスクを必要な値に設定し、戻る前にプログラム・マスクを復元する必要があります。

アセンブラー・ドライバーを使用するアプリケーションのアップグレード

アセンブラー・ドライバーを使用して COBOL サブルーチンを呼び出すアプリケーションをアップグレードするために使用できる方法は 3 つあります。

- アセンブラー・ドライバーを言語環境プログラム準拠のアセンブラーに移行する
- アセンブラー・ドライバーを変更して言語環境プログラム環境をセットアップする
- RTEREUS ランタイム・オプションを使用する (アセンブラー・ドライバーを変更できない場合)

以下のセクションで、これらの方法を説明します。いずれのケースでも、COBOL サブルーチンは、他の COBOL 移行シナリオで記述されているのと同じ方法でアップグレードすることができます。

アセンブラー・ドライバーの移行

アセンブラー・ドライバーを持つアプリケーションをアップグレードするには、アセンブラー・ドライバーを言語環境プログラム準拠のアセンブラー・メインプログラムに変更することができます。既存のアセンブラー・プログラムを言語環境プログラム準拠にする方法については、*言語環境プログラム プログラミング・ガイド* を参照してください。

アセンブラー・ドライバーの変更

アセンブラー・ドライバー・ルーチンを変更したい場合は、OS/VS COBOL ILBOSTP0 ルーチンを言語環境プログラムの CEEPIPI INIT_SUB、CEEPIPI INIT_MAIN、および CEEPIPI TERM 機能で置き換えることができます。これらの言語環境プログラム・ルーチンには、OS/VS COBOL では利用できない便利な補足終了機能があります。

IGZERRE または ILBOSTP0 がアセンブラー・ドライバーと静的にリンクされる場合は、アセンブラー・ドライバーを言語環境プログラムとリンク・エディットしなければなりません。ILBOSTP0 を使用する場合は、言語環境プログラム・オプション ALL31(OFF) および STACK(,,BELOW) を指定する必要があります。

事前初期設定を使用するための代替方式については、113 ページの『ランタイム環境の初期設定』を参照してください。

変更しないアセンブラー・ドライバーの使用

非 COBOL ドライバーを変更できない (または変更したくない) 場合は、言語環境プログラムの RTEREUS ランタイム・オプションを指定すれば、変更しないドライバーを使用することができます (RTEREUS は、最初の COBOL プログラムが呼び出されるときに、ランタイム環境を再使用できるように初期設定します)。

重要: RTEREUS は、すべてのアプリケーションについて推奨されるわけではありません。場合によっては、望ましくない動作を示します。RTEREUS を使用する前に、可能性のある副次作用を徹底的に調べ、アプリケーションに与える影響を理解しておいてください。詳細については、59 ページの『非 CICS アプリケーションについて推奨されるランタイム・オプション』を参照してください。

MVS ATTACH による COBOL プログラムの呼び出し

MVS ATTACH を使用して COBOL メインプログラムを呼び出す場合、言語環境プログラムは VS COBOL II とは異なる方法でパラメーター・リストを処理します。

言語環境プログラムのもとでは、COBOL プログラムが ATTACH SVC によって直接呼び出された (オペレーティング・システムによるバッチ・プログラムの呼び出しおよび TSO CALL/ATTACH からの呼び出しを含む) ときには、パラメーター・リストは常に ″PARM=″ スタイルとして処理されます。

VS COBOL II のもとでは、COBOL プログラムが ATTACH SVC によって直接呼び出された (オペレーティング・システムによるバッチ・プログラムの呼び出しおよび TSO CALL/ATTACH からの呼び出しを含む) ときには、以下の場合にのみ、パラメーター・リストは ″PARM=″ スタイルとして処理されます。

- レジスター 1 がゼロ以外である。
- レジスター 1 によってアドレッシングされるワード (最初のパラメーター・ポインター・ワード) で、リストの終わり (EOL) ビットがオンである。
- EOL によってアドレッシングされるパラメーターがハーフワード以上の境界に位置合わせされている。

上記以外の場合は、レジスター 1 およびパラメーター・リストは変更されずに渡されます。

互換性のある動作を得るための方法が 2 つあります。

1. メインプログラムを言語環境プログラム準拠のアセンブラーに変更し、CEEENTRY マクロで PLIST=OS キーワードを使用します。その後、アセンブラー・プログラムが COBOL プログラムを呼び出すようにします。これを行う方法については、以下のサンプル・コードを参照してください。

```
ASMLE3   CEEENTRY PPA=MAINPPA,AUTO=WORKSIZE,MAIN=YES,PLIST=OS
         USING WORKAREA,13
         L     15,A1C401P         Get the addr of the COBOL pgm
         BALR  14,15              Call it with parm list unchanged
*=================================================================
*   Terminate Language Environment.
*=================================================================
         CEETERM  RC=0
MAINPPA  CEEPPA                   Constants describing the code block
*=================================================================
*        The Workarea and DSA
*=================================================================
A1C401P  DC    V(A1C401P)         VCON FOR COBOL pgm
WORKAREA DSECT
         ORG    *+CEEDSASZ        Leave space for the DSA fixed part
         DS     0D
WORKSIZE EQU    *-WORKAREA
         CEEDSA                   Mapping of the Dynamic Save Area
         CEECAA                   Mapping of the Common Anchor Area
         CEEEDB                   Mapping of the Enclave Data Block
         END   ASMLE3
```

2. 言語環境プログラムを変更することにより、ATTACH によって呼び出された COBOL メインプログラムのためのパラメーター・リスト処理が、COBOL プログラムが VS COBOL II ランタイムと共に実行されるときと同様に動作するようにします。言語環境プログラムのもとでのパラメーター・リスト処理を変更するには、サンプル・カスタマイズ・ジョブ IGZWAPSX を IGZEPSX (COBOL パラメーター・リスト出口ルーチン) の変更済みコピーと共に実行してください。

   IGZEPSX を変更する方法の指示については、以下の資料を参照してください。
   - OS/390 および z/OS については、*言語環境プログラム (OS/390 版) カスタマイズ* を参照してください。

COBOL プログラムをロードし、呼び出すアセンブラー

VS COBOL II ランタイムでは VS COBOL II プログラムは各呼び出しごとに WORKING-STORAGE の異なるコピーを使用しますが、言語環境プログラムでは各呼び出しごとに WORKING-STORAGE の同じコピーが使用されます。このような状況の違いは、以下の条件が該当する場合に発生します。

- RENT オプションを指定してコンパイルされる
- OS/VS COBOL、VS COBOL II、IBM COBOL、または Enterprise COBOL プログラムから動的に呼び出される
- PL/I によって取り出され、呼び出される

さらに、プログラムは VS COBOL II ランタイム・ライブラリーのもとで実行されると初期状態になります。ただし、言語環境プログラムのもとで実行されると、CANCEL による介入がない場合に限り、プログラムは最後に使われた状態になります。

COBOL プログラムをロードし、削除するアセンブラー・プログラム

言語環境プログラムでは、アセンブラー・プログラムは、以下のいずれかを含んでいるロード・モジュールを SVC ロードし、SVC 削除することができます。

- OS/VS COBOL プログラム
- NORENT オプションを指定してコンパイルされた VS COBOL II プログラム

- NORENT オプションを指定してコンパイルされた IBM COBOL プログラム
- NORENT オプションを指定してコンパイルされた Enterprise COBOL プログラム

注: デバッグ・ツールは、アセンブラーが SVC 削除を使用して削除したロード・モジュールに含まれている COBOL プログラムをサポートしません。

言語環境プログラムでは、アセンブラー・プログラムは、以下のいずれかを含んでいるロード・モジュールを SVC ロードすることができますが、SVC 削除することはできません。

- RENT オプションを指定してコンパイルされた VS COBOL II プログラム
- RENT オプションを指定してコンパイルされた IBM COBOL プログラム
- RENT オプションを指定してコンパイルされた Enterprise COBOL プログラム

アセンブラー・プログラムが、3 種類のプログラムを含んでいるロード・モジュールを SVC 削除すると、予測できない結果になる可能性があります。

COBOL RENT プログラムを含んでいるロード・モジュールをロードして削除する必要があるアセンブラー・プログラムの場合、以下のいずれかを行ってください。

- アセンブラー・プログラムが COBOL プログラムを静的に呼び出し、その COBOL プログラムで動的呼び出しを実行し、CANCEL を実行するようにする。
- 言語環境プログラム リリース 7 (OS/390 リリース 3) 以上のもとで、CEEFETCH および CEERELES マクロを使用する。

注: CEEFETCH は、言語環境プログラム準拠のプログラムをロードするためにのみ使用することができます。CEEFETCH は、IBM COBOL および Enterprise COBOL プログラムをロードするために使用することができます。CEEFETCH を使用して VS COBOL II または OS/VS COBOL プログラムをロードすることはできません。

サブプールのストレージの解放 (z/OS および OS/390 のみ)

z/OS および OS/390 では、言語環境プログラムはサブプール 1 およびサブプール 2 からのストレージを割り振ります。VS COBOL II ではサブプール 0 のみが使用されました。サブプール 1 またはサブプール 2 のストレージを解放すると、言語環境プログラムによってこれらのサブプールに置かれたデータが失われる可能性があります。

プログラムの呼び出し — AMODE 要件

VS COBOL II の場合、アセンブラー・プログラムは、AMODE 指定に関係なく COBOL プログラムを呼び出すことができました。以下に例を示します。

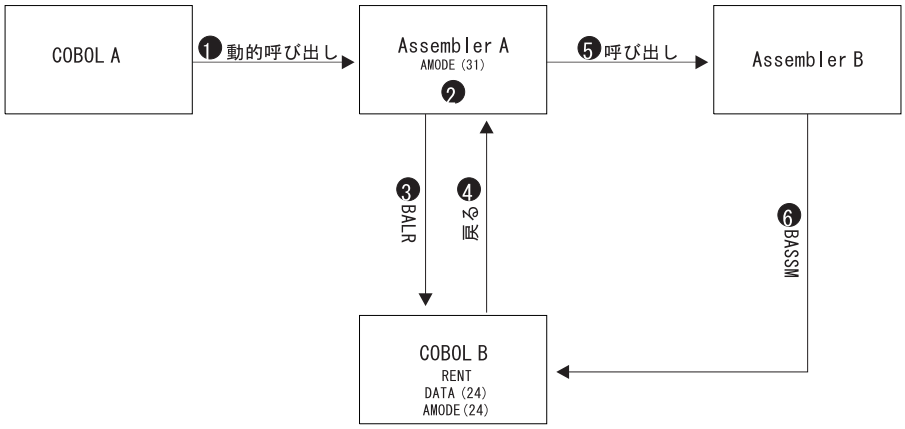


図 8. プログラムを呼び出す場合の AMODE の影響

上の図では、以下のことが行われます。

1. プログラム COBOLA が AssemblerA を動的に呼び出します (アセンブラーは AMODE 31 です)。
2. AssemblerAが COBOLB をロードします。COBOLB は RENT、DATA(24)、および AMODE(24) です。
3. AssemblerA が COBOLB への BALR を行います (COBOLB は AMODE 31 で入られました)。
4. COBOLB が AssemblerA に戻ります。
5. AssemblerA が AssemblerB を呼び出します。
6. AssemblerB が COBOLB への BASSM を行います (COBOLB は AMODE 24 で入られました)。
7. COBOLB は、AMODE 31 で入られることを予期しているため、異常終了します。

言語環境プログラムの場合、VS COBOL II、IBM COBOL、または Enterprise COBOL でコンパイルされ、アセンブラー・プログラムによって呼び出される COBOL プログラムは、それが呼び出されるたびに同じ AMODE で入れられる必要があります。

上の例で異常終了を避けるためには、

- ステップ 2 で、AssemblerA は AMODE を保管する必要があります。
- ステップ 3 で、AssemblerA は COBOLB への BASSM (BALR ではなく) を行う必要があります。
- ステップ 4 で、AssemblerA はステップ 2 で保管した AMODE を復元する必要があります。

AMODE 31 であるアセンブラー・プログラムが AMODE 24 である COBOL プログラムを呼び出す場合、COBOL がアセンブラー・プログラムに戻るためには、アセンブラー・プログラムは RMODE 24 であることも必要です。この場合、アセンブラー・プログラムが AMODE ANY であれば、COBOL プログラムからの戻り時に、無効アドレスへの分岐の結果として異常終了が発生する可能性があります。そ

の理由は、R14 にはアセンブラー・プログラムの保管域からの 31 ビット・アドレスが入りますが、COBOL は AMODE 24 でアセンブラー・プログラムに戻るためです。

付録 E. デバッグ・ツールの比較

デバッグ・ツールは、言語環境プログラム内で実行されるプログラム分析ルーチンであり、いくつかの高水準言語 (Enterprise COBOL を含む) をサポートします。

Enterprise COBOL の場合、デバッグ・ツール はコンパイラーのフィーチャーとして注文可能です。

既存のアプリケーションのデバッグ

デバッグ・ツールは、OS/VS COBOL プログラムのデバッグをサポートしません。

デバッグ・ツールは、VS COBOL II リリース 3.0 以上をサポートします (VS COBOL II プログラムが TEST コンパイラー・オプションを指定してコンパイルされている場合)。VS COBOL II プログラムのデバッグに関する考慮事項には、以下のものがあります。

COBTEST
COBTEST は、VS COBOL II ライブラリーを使用する VS COBOL II プログラムのデバッグのために引き続き使用できます。COBTEST はデバッグ・ツールと互換性がなく、言語環境プログラムと共に使用することはできません。

テスト・スクリプト
COBTEST を使用してプログラム・テストを実行し、COBTEST テスト・スクリプトのライブラリーがある場合に備えて、デバッグ・ツールには移行処置を援助するコマンド・トランスレーターが入っています。

マイグレーション済みアプリケーションのデバッグ

OS/VS COBOL、VS COBOL II、または IBM COBOL プログラムを Enterprise COBOL でコンパイルするときは、どのコンパイラー・オプションを使用してデバッグを使用可能にするかを理解していることが必要です。

OS/VS COBOL プログラムを含んでいるアプリケーション

デバッグ・ツールは、Enterprise COBOL プログラムと OS/VS COBOL プログラムの組み合わせを含んでいる実行単位のデバッグに使用できますが、OS/VS COBOL プログラムについての情報はデバッグ・ツールで使用可能ではありません。

実行単位が FLOW または COUNT コンパイラー・オプションを指定する OS/VS COBOL プログラムを含んでいる場合は、実行単位内のすべての プログラム (VS COBOL II、IBM COBOL、および Enterprise COBOL プログラムを含む) がデバッグ・ツールによってデバッグできません。

VS COBOL II プログラムを含んでいるアプリケーション

FDUMP コンパイラー・オプション

FDUMP コンパイラー・オプションは、Enterprise COBOL でサポートされませんが、その機能はサポートされます。FDUMP は TEST(SYM) コンパイラー・オプションにマップされています。

TEST コンパイラー・オプション

TEST コンパイラー・オプションの構文が変更されたことにより、ユーザーは、コンパイル・フックが置かれる必要があるかどうか (および、それが置かれる位置) と、ディクショナリーおよび CALC テーブルが生成される必要があるかどうかを指定することができます。5 つのフック位置サブオプション (ALL、NONE、STMT、PATH、および BLOCK) が使用可能です。また、2 つの記号テーブル・サブオプション (SYM および NOSYM) が使用可能です。

VS COBOL II プログラムと Enterprise COBOL プログラムが混在するアプリケーションでは、デバッグ・ツールは、TEST を指定してコンパイルされた VS COBOL II プログラムのみをデバッグすることができます。

バッチ・デバッグの考慮事項

デバッグ・ツールは、COBTEST のバッチ・モードと同等の機能を、いくつかの追加機能と共に提供します。

バッチ・モードでは、デバッグ・ツールは入力を入力ファイルから取得し、デバッグ・ツール出力は COBTEST バッチ・モードの場合と類似した出力ファイルに書き込まれます。デバッグ・ツールのバッチ・モードと COBTEST のバッチ・モード間の違いは、COBTEST のコマンド (RECORD、 QUALIFY、RESTART、 GO、および RUN) がバッチ・モードでは対話モードの場合と異なる動作を示すという点です。デバッグ・ツールでは、すべてのコマンドがいずれのモードでも同じ動作を示します。

デバッグ・ツールのいくつかのコマンド (フルスクリーン・コマンドなど) は、バッチ・モードでは使用できません。

デバッグ・ツールの開始

Enterprise COBOL プログラムの場合、デバッグ・ツールを開始する方法が異なります。デバッグ・ツールの使用時には、まず、アプリケーション・プログラムが開始され、言語環境プログラムの TEST ランタイム・オプションがデバッグ・ツールの呼び出しを制御します。(VS COBOL II では、最初に COBTEST を呼び出すことが必要です。COBTEST は、その後、デバッグされるアプリケーションを開始します。)

言語環境プログラムの呼び出し可能サービス CEETEST を使用することによって、デバッグ・ツールをアプリケーションから直接呼び出すこともできます。これらの 2 つの方法について、以下で簡単に説明します。

TEST ランタイム・オプション

言語環境プログラムの TEST ランタイム・オプションは、アプリケーション・プログラムが言語環境プログラムと共に実行されているときに、デバッグ・ツールが呼び出されるかどうかを決定するために使用します。オプションのサブパラメーターに応じて、呼び出しを即時実行することもできますし、据え置くこともできます。

IBM 提供のデフォルトは NOTEST です。これは、デバッグ・ツールが、初期コマンド・ストリングを処理するためにも、プログラムの実行時に発生する可能性があるプログラム条件のためにも初期設定されないことを指定します。ただし、デバッグ・サービスが必要である場合には、ライブラリー・サービス CEETEST を使用してデバッグ・ツールを呼び出すことができます。

言語環境プログラムの TEST オプションのサブパラメーターおよびサブオプションの詳細については、*言語環境プログラム プログラミング・リファレンス* を参照してください。

CEETEST

言語環境プログラムには、デバッグ・ツールが制御を獲得することを可能にし、デバッグ・ツールに渡されるコマンド・ストリングを指定するための呼び出し可能サービス CEETEST が用意されています。このサービスを呼び出すと、デバッグ・ツールが初期設定され、呼び出されます (デバッグ・ツールがすでに初期設定されている場合は、この再入は停止点と同様になります)。

CEETEST を使用してデバッグ・ツールを呼び出すときは、コマンド・リストを含んでいるストリング・パラメーターはオプションです。コマンド・リストを使用すると、コマンドがデバッグ・ツールに渡され、実行されます。コマンド・リストに GO、GOTO、STEP、または QUIT コマンドのどれも含まれていない場合は、端末または基本コマンド・ファイルからのコマンドが要求されます。いずれかのポイント (コマンド・リスト、端末、またはコマンド・ファイル) で GO コマンドが検出されると、デバッグ・ツールはアプリケーション・プログラム内のサービス呼び出しに続くポイントに戻り、プログラムが実行を継続します。

言語環境プログラムの呼び出し可能サービス CEETEST の詳細および例については、*言語環境プログラム プログラミング・リファレンス* を参照してください。

コマンド言語の比較

表 53 では、TESTCOB、COBTEST、およびデバッグ・ツールのコマンド言語を比較します。この表から分かるように、デバッグ・ツールの多くのコマンドが、COBTEST および TESTCOB で使用されるコマンドと異なります。ほとんどの場合、機能は同じです。

特定のデバッガーによってサポートされないコマンドは、ダッシュで示されています。

表 53. デバッグ・ツールのコマンド言語の比較

TESTCOB のコマンド	COBTEST のコマンド	デバッグ・ツールのコマンド	機能
AT	AT	AT	停止点を設定する
-	AUTO[1]	MONITOR LIST	変数を自動的にモニターする
-	COLOR[1]	PANEL COLORS	カラー属性を設定するパネルを表示する
-	DOWN[2]	WINDOW DOWN	ウィンドウを下方に移動する
DROP	DROP	CLEAR	定義済み記号を削除する
DUMP	DUMP	CALL %DUMP	メモリー・ダンプを作成する
END	QUIT	QUIT	デバッグ・セッションを終了する

表 53. デバッグ・ツールのコマンド言語の比較 (続き)

TESTCOB のコマンド	COBTEST のコマンド	デバッグ・ツールのコマンド	機能
EQUATE	EQUATE	SET EQUATE または CLEAR EQUATE	記号を定義する
-	FLOW	LIST LAST	制御フロー情報を収集する
-	FREQ	SET FREQUENCY	動詞の実行カウントを数える
GO	GO	GO	COBOL プログラムの実行を開始する
HELP (TSO のみ)	HELP[3]	HELP	オンライン・ヘルプ情報を提供する
IF	IF	IF	条件を評価する
-	LEFT[2]	WINDOW LEFT	ウィンドウを左方に移動する
-	LINK	LOAD または CALL[5]	LINKAGE SECTION をセットアップする
LIST	LIST	LIST、DESCRIBE[6]	変数の内容をリストする
LISTBRKS	LISTBRKS	LIST AT	有効な停止点をリストする
-	LISTEQ	QUERY EQUATES	定義済み記号をリストする
LISTFILE	LIST	DESCRIBE ATTRIBUTES	ファイルの属性をリストする
-	LISTFREQ	LIST FREQUENCY	動詞の頻度カウントをリストする
-	LISTINGS[1]	PANEL LISTINGS	プログラムをリストと関連付けるパネルを表示する
-	MOVECURS[1]	CURSOR	カーソルをコマンド行からウィンドウへ、またはその逆に移動する
NEXT	NEXT	STEP	次の動詞に一時的停止点を設定する
-	NORECORD	SET LOG OFF	デバッグ・セッションのロギングをオフにする
OFF	OFF	CLEAR AT	有効な停止点をオフにする
OFFWN	OFFWN	CLEAR AT	条件付き停止点をオフにする
-	ONABEND	AT OCCURRENCE CEExxxx	言語環境プログラムの条件の発生時にアクションを実行する (xxxx は言語環境プログラムから戻される条件)
-	PEEK	QUERY prefix	行内の停止点を示す
-	POSITION[1]	SCROLL TO	表示されたオブジェクトの特定の位置へ移動する
-	PREVDISP[1]	-[6]	最後に表示されたユーザー ISPF 画面を示す
-	PRINTDD	-[7]	出力をデータ・セットへ送る
-	PROC	AT CALL	特定のサブプログラムへの呼び出しをトラップする
-	PROFILE[1]	PANEL PROFILE	ユーザー・プロファイル属性を設定するパネルを表示する
QUALIFY	QUALIFY	SET QUALIFY	現行プログラム修飾を変更する
-	RECORD	SET LOG ON	デバッグ・セッションのロギングをオンにする
-	RESTART	RESTART[8]	デバッガーを終了せずにプログラムを再初期設定する
-	RESTORE[1]	SET または QUALIFY RESET	実行の現在位置に戻る
-	RIGHT[2]	WINDOW RIGHT	ウィンドウを右方に移動する

表 53. デバッグ・ツールのコマンド言語の比較 (続き)

TESTCOB のコマンド	COBTEST のコマンド	デバッグ・ツールのコマンド	機能
RUN	RUN	CLEAR AT の後に GO	停止点を除去し、完了まで実行する
-	SEARCH[1]	FIND	表示されたオブジェクト内でストリングを検索する
-	SELECT	SHOW	特定の頻度カウントを表示する
SET	SET	MOVE または SET または COMPUTE	変数の内容を変更する
SOURCE	SOURCE[9]	WINDOW OPEN	ソース・プログラム・ステートメントを表示する
-	STEP	STEP	指定された数の動詞を実行する
-	SUFFIX[1]	SET SUFFIX	頻度の視覚表示をオンにする
TRACE	TRACE	AT GLOBAL STATEMENT または AT GLOBAL PATH	実行のフローをトレースする
-	UP[2]	WINDOW UP	ウィンドウを上方に移動する
-	VTRACE[1]	STEP	プログラムを動的に視覚トレースする
WHEN	WHEN	AT * IF...	条件付き停止点をセットアップする
-	WHERE	QUERY LOCATION	実行の現在位置をリストする

注:

1. これらのコマンドは、COBTEST FULL SCREEN デバッグでのみサポートされます。
2. DOWN、LEFT、RIGHT、および UP コマンドは、ISPF によって実行されます。これらは、実際には COBTEST のコマンドではありません。
3. HELP は、TSO のもとでの COBTEST LINE MODE デバッグでのみサポートされます。
4. COBTEST の LINK は、LINKAGE SECTION 内のデータ項目に実記憶装置を提供します。デバッグ・ツールは、それらのデータ名を宣言することによってこれを行います。その後、COBOL プログラムを呼び出し、必要なパラメーターを渡します。
5. デバッグ・ツールには、ある範囲の名前をリストする機能がありません。ただし、あるブロックのすべての名前、または特定のパターンに適合するすべての名前をリストすることは可能です。
6. PREVDISP コマンドは必要なくなりました。デバッグ・ツールは、表示のために ISPF を使用しません。
7. PRINTDD コマンドは、デバッグ・ツールの場合は不要です。印刷ファイルには固定 DD 名があります (ただし、この DD 名は呼び出し時に指定できます)。
8. RESTART コマンドは、プログラマブル・ワークステーションでのみ使用することができます。RESTART コマンドは、COBTEST の場合とは異なり、停止点の設定を保存しないことに注意してください。
9. COBTEST SOURCE コマンドは、FULL SCREEN デバッグでのみサポートされます。また、TESTCOB SOURCE コマンドとは意味が異なります。

付録 F. コンパイラー・オプションの比較

この付録では、Enterprise COBOL のコンパイラー・オプションを記述します。 また、Enterprise COBOL のコンパイラー・オプションと VS COBOL II、OS/VS COBOL、および IBM COBOL のコンパイラー・オプションとの比較を示します。Enterprise COBOL オプションの完全な説明については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

表 54. コンパイラー・オプションの比較

	適用対象				
オプション	OSVS	VSII	IBM COBOL	Enterprise COBOL	使用上の注意
ADATA			✓	✓	コンパイル時に関連データ・ファイルを生成します。NOADATA がデフォルトです。COBOL/370 では、Enterprise COBOLの ADATA オプションの代わりに EVENTS オプションが使用されます。
ANALYZE			✓**		コンパイラーに、固有 COBOL ステートメントに加えて、組み込み SQL および CICS ステートメントの構文を検査するように指示します。
ADV	✓	✓	✓	✓	レコードの先頭に印刷制御バイトを追加します。ADV がデフォルトです。
ALOWCBL		✓	✓	✓	ソース・プログラム内で PROCESS または CBL ステートメントを使用できるようにします。このオプションは、インストール時にのみ指定することができます。ALOWCBL がデフォルトです。
APOST	✓	✓	✓	✓	リテラルの区切り文字としてアポストロフィ (') を指定します。QUOTE がデフォルトです。 Enterprise COBOL では、APOST と QUOTES のどちらが有効であるかには関係なく、リテラルを引用符とアポストロフィのどちらで区切っても構いません。APOST を使用すると、形象定数 QUOTE/QUOTES は 1 つ以上のアポストロフィ (') 文字を表します。

コンパイラー・オプションの比較

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
ARITH			✓*	✓	10 進データに指定できる桁の最大数を設定し、中間結果の精度に影響を与えます。 ARITH(COMPAT) の場合、PICTURE 文節、固定小数点数値リテラル、NUMVAL および NUMVAL-C への引き数に 18 桁を、FACTORIAL への引き数に 28 桁を指定することができます。 ARITH(EXTEND) の場合、PICTURE 文節、固定小数点数値リテラル、NUMVAL および NUMVAL-C への引き数に 31 桁を、FACTORIAL への引き数に 29 桁を指定することができます。
AWO		✓	✓	✓	VB フォーマットの物理順次ファイルについて APPLY WRITE-ONLY 処理を活動化します。NOAWO がデフォルトです。
BUF	✓				コンパイラー作業データ・セット用のバッファー・ストレージを割り振ります。Enterprise COBOL では、OS/VS COBOL の BUF オプションの代わりに BUFSIZE オプションが使用されます。
BUFSIZE		✓	✓	✓	コンパイラー作業データ・セット用のバッファー・ストレージを割り振ります。次の 3 つのサブオプションが使用可能です。BUFSIZE(nnnnn)、BUFSIZE(nnnK)、および BUFSIZE(4096)。BUFSIZE(4096) がデフォルトです。OS/VS COBOL の BUF オプションに代わって、BUFSIZE が使用されるようになりました。
CICS			✓**	✓	組み込みの CICS (顧客情報管理システム) トランスレーター機能を使用可能にし、CICS オプションを指定します。NOCICS がデフォルトです。
CLIST	✓				圧縮された PROCEDURE DIVISION リストと、テーブルおよびプログラム統計を生成します。NOCLIST がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の CLIST オプションの代わりに OFFSET オプションが使用されます。

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
CMPR2		✓	✓		VS COBOL II リリース 2 またはその他の VS COBOL II の動作と互換性のある IBM COBOL ソース・コードの生成を指定します。 NOCMPR2 がデフォルトです。NOCMPR2 は、すべての IBM COBOL 言語機能 (オブジェクト指向 COBOL のための言語拡張、および C プログラムとの向上したインターオペラビリティーのための言語拡張を含む) の完全な使用を指定します。
CODEPAGE				✓	実行時に、COBOL ソース・プログラムの英数字、国別、および DBCS (2 バイト文字セット) リテラルと同様に英数字および DBCS データ項目の内容をエンコードするために使用するコード・ページを指定します。
COMPILE		✓	✓	✓	無条件の完全コンパイルを要求します。その他のオプションは NOCOMPILE および NOCOMPILE(W|E|S) です。デフォルトは NOCOMPILE(S) です。 NOCOMPILE は、無条件の構文検査を指定します。NOCOMPILE(W|E|S) は、エラーの重大度に基づく条件付き構文検査を指定します。 COMPILE は、OS/VS COBOL の NOSYNTAX および NOCSYNTAX オプションと同等です。NOCOMPILE は、OS/VS COBOL の SYNTAX オプションと同等です。NOCOMPILE(W|E|S) は、OS/VS COBOL の CSYNTAX および SUPMAP オプションと同等です。
CURRENCY			✓**	✓	デフォルト通貨記号を定義します。プログラムで CURRENCY オプションと CURRENCY SIGN 文節の両方が使用されると、CURRENCY SIGN 文節に指定された記号が、PICTURE 文節内で通貨記号であると見なされます。 NOCURRENCY がデフォルトであり、CURRENCY オプションによって代替通貨記号が提供されないことを示します。
DATA(24) DATA(31)		✓	✓	✓	再入可能プログラムのデータ域が 16MB 境界の上と下のどちらに存在するかを指定します。DATA(24) の場合は、再入可能プログラムは 16MB 境界より下に置かなければなりません。DATA(31) の場合は、再入可能プログラムを 16MB 境界より上に置くことができます。DATA(31) がデフォルトです。

コンパイラー・オプションの比較

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
DATEPROC			✓	✓	COBOL コンパイラーの 2000 年言語拡張 (MLE) を使用可能にします。オプションは、DATEPROC(FLAG)、DATEPROC(NOFLAG) 、DATEPROC(TRIG)、DATEPROC(NOTRIG)、および NODATEPROC です。
DBCS		✓	✓	✓	コンパイラーに、DBCS シフトインおよびシフトアウト・コードを認識するように指示します。 DBCS がデフォルトです。
DBCSXREF=code		✓	✓	✓	DBCS 文字への相互参照のために順序付けプログラムが使用されることを指定します。ここで、code は、DBCS 順序付けサポート・プログラムについての情報を与えるパラメーターを設定します。DBCSXREF は、インストール時にのみ指定することができます。 DBCSXREF=NO がデフォルトです。
DECK	✓	✓	✓	✓	オブジェクト・コードを 80 文字のカード・イメージとして生成し、それを SYSPUNCH ファイルに入れます。NODECK がデフォルトです。
DIAGTRUNC			✓*	✓	受け取り側が数値である MOVE ステートメントの場合に、受け取りデータの整数位置の数が送り出しデータ項目またはリテラルよりも少ないときは重大度 4 (警告) の診断を出すようにコンパイラーに指示します。
DLL			✓**	✓	コンパイラーは、DLL (ダイナミック・リンク・ライブラリー) サポートに使用可能であるオブジェクト・モジュールを生成することができます。NODLL がデフォルトです。
DMAP	✓				データ部および暗黙に宣言された項目のリストを生成します。NODMAP がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の DMAP オプションの代わりに MAP オプションが使用されます。
DUMP	✓	✓	✓	✓	コンパイルの終了時にシステム・ダンプが生成されることを指定します。NODUMP がデフォルトです。

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
DYNAM	✓	✓	✓	✓	CALL リテラル・ステートメントの動作を変更して、実行時にサブプログラムを動的にロードします。NODYNAM がデフォルトです。NODYNAM の場合、CALL リテラル・ステートメントにより、サブプログラムはロード・モジュール内で静的にリンク・エディットされます。
EXIT(IN-id) EXIT(LIB-id) EXIT(PRT-id) EXIT(ADT-id)		✓	✓	✓	これにより、コンパイラーはユーザー提供モジュールを受け入れることができます (それぞれの *string* は出口モジュールへのオプションのユーザー提供入力ストリングであり、それぞれの *mod* はユーザー提供出口モジュールの名前です)。 ADT-id サブオプションは、COBOL (MVS および VM 版) および COBOL (OS/390 および VM 版) でのみ使用可能です。 NOEXIT がデフォルトです。
EXPORTALL			✓**	✓	オブジェクト・デックをリンク・エディットして DLL を作成するときに特定の記号を自動的にエクスポートするようにコンパイラーに指示します。NOEXPORTALL がデフォルトです。
FASTSRT		✓	✓	✓	IBM DFSORT ライセンス・プログラムによる高速ソートを指定します。NOFASTSRT がデフォルトであり、Enterprise COBOL が SORT または MERGE 入出力 (I/O) を行うことを指定します。
FLAG	✓	✓	✓	✓	指示されたレベルで構文メッセージが生成されることを示します。OS/VS COBOL の場合、FLAG オプションは FLAGW および FLAGE です。Enterprise COBOL の場合、FLAG オプションは以下のとおりです。 FLAG(I) FLAG(W) FLAG(E) FLAG(S) FLAG(U) FLAG(I|W|E|S|U,I|W|E|S|U) VS COBOL II および IBM COBOL の場合、FLAG(I) がデフォルトです。Enterprise COBOL の場合、FLAG(I,I) がデフォルトです。
FLAGMIG		✓	✓		VS COBOL II リリース 2 または CMPR2 を用いるその他のプログラムの動作から変更された可能性があるセマンティックに対する、NOCMPR2 のフラグ設定を指定します。 NOFLAGMIG がデフォルトです。

コンパイラー・オプションの比較

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
FLAGSTD		✓	✓	✓	COBOL 85 標準のフラグ設定を指定します。COBOL (OS/390 および VM 版) および COBOL (MVS および VM 版) の場合、FLAGSTD はオブジェクト指向 COBOL 用の言語構文、向上した C インターオペラビリティー用の言語構文、および PGMNAME(LONGMIXED) コンパイラー・オプションの使用にもフラグを立てます。 NOFLAGSTD がデフォルトです。
FDUMP		✓			アプリケーションが異常終了で終了するときに、デバッグ情報を含んでいるダンプを生成するために使用します。NOFDUMP がデフォルトです。 Enterprise COBOL では、VS COBOL II の FDUMP オプションの代わりに TEST(SYM) オプションが使用されます。
IDLGEN			✓		IDLGEN は、COBOL ソース・ファイルの通常コンパイルに加えて、定義されたクラスについての IDL 定義を生成します。NOIDLGEN がデフォルトです。
INTDATE(ANSI) INTDATE(LILIAN)			✓	✓	整数形式の日付が日付組み込み関数で使用されるときの開始日付を決定します。ANSI では、COBOL 85 標準の開始日付 (Day 1 = January 1, 1601) が使用されます。LILIAN では、言語環境プログラムのリリアン開始日付 (Day 1 = October 15, 1582) が使用されます。 INTDATE(ANSI) がデフォルトです。
LANGUAGE		✓	✓	✓	LANGUAGE(AAa...a) は、コンパイラー・メッセージが出されるときの言語を指定します。ここで、`AAa...a` は、以下のとおりです。 **UE または UENGLISH** 英語 (大文字) **EN または ENGLISH** 英語 (大 / 小文字混合) **JA、JP、または JAPANESE** 日本語 (漢字文字セットを使用) LANGUAGE=(EN) がデフォルトです。
LIB	✓	✓	✓	✓	プログラムが COPY ライブラリーを使用することを指定します。NOLIB がデフォルトです。

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
LINECNT=nn	✓				出力リストのページ当たり行数を指定します。VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の LINECNT オプションの代わりに LINECOUNT オプションが使用されます。
LINECOUNT		✓	✓	✓	出力リストのページ当たり行数を指定します。LINECOUNT の 2 つの形式は、LINECOUNT(60) および LINECOUNT(nn) です。LINECOUNT(60) がデフォルトです。 OS/VS COBOL の LINECNT オプションに代わって、LINECOUNT が使用されるようになりました。
LIST		✓	✓	✓	ソース・コードのアセンブラー言語展開のリストを生成します。NOLIST がデフォルトです。 OS/VS COBOL の PMAP オプションに代わって、LIST が使用されるようになりました。
LOAD	✓				オブジェクト・コードをリンケージ・エディターへの入力用にディスクまたはテープに保管します。NOLOAD がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の LOAD オプションの代わりに OBJECT オプションが使用されます。
MAP		✓	✓	✓	データ部および暗黙に宣言された項目のリストを生成します。NOMAP がデフォルトです。 OS/VS COBOL の DMAP オプションに代わって、MAP が使用されるようになりました。
NAME	✓	✓	✓	✓	作成されたそれぞれのオブジェクト・モジュールにリンケージ・エディターの NAME ステートメントが付加されることを指示します。VS COBOL II、IBM COBOL、および Enterprise COBOL の場合、NAME にはサブオプション (ALIAS|NOALIAS) があります。ALIAS を指定すると、各 ENTRY ステートメントごとに ALIAS ステートメントも生成されます。 NONAME がデフォルトです。
NSYMBOL				✓	リテラルおよびピクチャー文節で使用される「N」記号の解釈を制御し、国別処理または DBCS 処理のどちらを前提とするかを指示します。 NSYMBOL(NATIONAL) がデフォルトです。

コンパイラー・オプションの比較

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
NUM	✓				エラー・メッセージおよびリスト内に行番号を印刷します。NONUM がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の NUM オプションの代わりに NUMBER オプションが使用されます。
NUMBER		✓	✓	✓	エラー・メッセージおよびリスト内に行番号を印刷します。NONUMBER がデフォルトです。 OS/VS COBOL の NUM オプションに代わって、NUMBER オプションが使用されるようになりました。
NUMCLS		✓	✓	✓	NUMPROC オプションと共に、NUMERIC クラス・テスト内の数値項目についての有効な符号構成を決定します。NUMCLS には 2 つのサブオプション (PRIM/ALT) があります。NUMCLS(PRIM) がデフォルトです。 NUMCLS は、インストール時にのみ指定することができます。詳細については、以下の資料を参照してください。 • *Enterprise COBOL カスタマイズ・ガイド*
NUMPROC		✓	✓	✓	パック / ゾーン 10 進数の符号を以下のように処理します。 **NUMPROC(PFD)** 10 進数フィールドは、システム /370 の標準の符号を持つものと見なされます。 **NUMPROC(NOPFD)** コンパイラーがすべての必要な符号変換および修理を行います。 **NUMPROC(MIG)** Enterprise COBOL が OS/VS COBOL と非常に類似した方法で符号変換を処理します。 NUMPROC(NOPFD) がデフォルトです。
OBJECT		✓	✓	✓	オブジェクト・コードをリンケージ・エディターへの入力用にディスクまたはテープに保管します。NOOBJECT がデフォルトです。 OS/VS COBOL の LOAD オプションに代わって、OBJECT が使用されるようになりました。

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
OFFSET		✓	✓	✓	圧縮された PROCEDURE DIVISION リストと、テーブルおよびプログラム統計を生成します。NOOFFSET がデフォルトです。 OS/VS COBOL の CLIST オプションに代わって、OFFSET が使用されるようになりました。
OPTIMIZE	✓	✓	✓	✓	オブジェクト・プログラムを最適化します。IBM COBOL および Enterprise COBOL の場合、OPTIMIZE にはサブオプション (STD/FULL) があります。OPTIMIZE(FULL) は、OS/VS COBOL と VS COBOL II の両方の OPTIMIZE オプションと比較して向上したランタイム・パフォーマンスを提供します。その理由は、コンパイラーが未使用データ項目を廃棄し、これらのデータ項目について VALUE 文節のコードを生成しないためです。 NOOPTIMIZE がデフォルトです。
OUTDD(SYSOUT) OUTDD(ddname)		✓	✓	✓	DISPLAY 出力を SYSOUT または指定されたデータ・セットに送ります。OUTDD(SYSOUT) がデフォルトです。 OS/VS COBOL の SYSx オプションに代わって、OUTDD が使用されるようになりました。
PGMNAME			✓	✓	長さおよび大文字小文字に関してプログラム名の処理を制御します。 **PGMNAME(LONGMIXED)** プログラム名は、全部の長さで (切り捨てなしで) 使用され、コンパイラーによる変換および大文字への変換は行われません。 **PGMNAME(LONGUPPER)** プログラム名は、全部の長さで (切り捨てなしで) 使用されます。 **PGMNAME(COMPAT)** プログラム名は、リリース 1 と同様に処理されます。 PGMNAME(COMPAT) がデフォルトです。
PMAP	✓				ソース・コードのアセンブラー言語展開のリストを生成します。NOPMAP がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の PMAP オプションの代わりに LIST コンパイラー・オプションが使用されます。

コンパイラー・オプションの比較

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
QUOTE	✓	✓	✓	✓	リテラルの区切り文字として引用符 (") を指定します。QUOTE がデフォルトです。 Enterprise COBOL では、APOST と QUOTES のどちらが有効であるかには関係なく、リテラルを引用符とアポストロフィのどちらで区切っても構いません。QUOTE を使用すると、形象定数 QUOTE/QUOTES は 1 つ以上のアポストロフィ (’) 文字を表します。
RES	✓	✓			ほとんどのライブラリー・ルーチンを、COBOL プログラムとリンク・エディットせずに、動的にロードされるようにします。
RENT		✓	✓	✓	オブジェクト・プログラムの再入可能コードを指定します。RENT がデフォルトです。
RMODE(AUTO) RMODE(24) RMODE(ANY)			✓	✓	NORENT プログラムが RMODE(ANY) を持つことを可能にします。RMODE(AUTO) がデフォルトです。
SEQ	✓				ソース・ステートメントの行番号の昇順を検査します。SEQ がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の SEQ オプションの代わりに SEQUENCE オプションが使用されます。
SEQUENCE		✓	✓	✓	ソース・ステートメントの行番号の昇順を検査します。SEQUENCE がデフォルトです。 OS/VS COBOL の SEQ オプションに代わって、SEQUENCE が使用されるようになりました。
SIZE(MAX) SIZE(nnnnn) SIZE(nnnK)		✓	✓	✓	コンパイルに使用される仮想記憶域を指定します。SIZE(MAX) が デフォルトです。
SOURCE	✓	✓	✓	✓	ソース・プログラムおよび組み込まれたメッセージのリストを生成します。SOURCE がデフォルトです。
SPACE	✓	✓	✓	✓	リストを 1 行、2 行、または 3 行送りで生成します。OS/VS COBOL での SPACE オプションの構文は、SPACE1、SPACE2、SPACE3 です。VS COBOL II および Enterprise COBOL での SPACE オプションの構文は、SPACE(1)、SPACE(2)、SPACE(3) です。 SPACE(1) がデフォルトです。
SQL			✓*	✓	DB2 coprocessor 能力を使用可能にし、DB2 サブオプションを指定します。

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
SSRANGE		✓	✓	✓	実行時に、添え字、指標、および参照変更の参照についての妥当性を検査します。 NOSSRANGE がデフォルトです。
SYSx	✓				DISPLAY 出力を SYSOUT または指定されたデータ・セットに送ります。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の SYSx オプションの代わりに OUTDD オプションが使用されます。
STATE	✓				アプリケーションが異常終了で終了するときに、デバッグ情報を含んでいるダンプを生成するために使用します。NOSTATE がデフォルトです。 IBM COBOL および Enterprise COBOL では、OS/VS COBOL の STATE オプションの代わりに TEST(NONE,SYM) オプションが使用されます。
SUPMAP SYNTAX CSYNTAX	✓				コンパイルの範囲を指定します。SYNTAX は、無条件の構文検査を指定します。CSYNTAX および CSUPMAP は、条件付きの構文検査を指定します。NOSYNTAX および NOCSYNTAX は、無条件の完全コンパイルを指定します。 VS COBOL II、 IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の SYNTAX、 CSYNTAX、および CSUPMAP オプションの代わりに COMPILE オプションが使用されます。
SXREF	✓				プログラム内で使用されているデータ名およびプロシージャー名のソート済み相互参照リストを生成します。デフォルトは NOSXREF です。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の SXREF オプションの代わりに XREF オプションが使用されます。
TERM	✓				SYSTERM データ・セットに進行メッセージを送ります。NOTERM がデフォルトです。 VS COBOL II、IBM COBOL、および Enterprise COBOL では、OS/VS COBOL の TERM オプションの代わりに TERMINAL オプションが使用されます。

コンパイラー・オプションの比較

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
TERMINAL		✓	✓	✓	SYSTERM データ・セットに進行メッセージを送ります。NOTERMINAL がデフォルトです。 OS/VS COBOL の TERM オプションに代わって、TERMINAL が使用されるようになりました。
TEST	✓	✓	✓	✓	プロダクト用のデバッグ・ツールで使用できるオブジェクト・コードを生成します。NOTEST がデフォルトです。 Enterprise COBOL の TEST オプションのサブオプションの詳細については、*Enterprise COBOL プログラミング・ガイド* を参照してください。
THREAD				✓	PL/I タスクにおいて複数の POSIX スレッドを持つ 1 つの実行単位で COBOL プログラムを実行できるようにします。NOTHREAD がデフォルトです。
TRUNC	✓	✓	✓	✓	最終の中間結果を切り捨てます。OS/VS COBOL には、TRUNC および NOTRUNC オプションがあります (NOTRUNC がデフォルトです)。VS COBOL II、IBM COBOL、および Enterprise COBOL には、TRUNC(STD|OPT|BIN) オプションがあります。 **TRUNC(STD)** 数値フィールドを 2 進数受け取りフィールドの PICTURE 指定に従って切り捨てます。 **TRUNC(OPT)** 数値フィールドを最適な方法で切り捨てます。 **TRUNC(BIN)** 2 進数フィールドを、それが占有するストレージに基づいて切り捨てます。 TRUNC(STD) がデフォルトです。 完全な説明については、Enterprise COBOL の使用しているバージョンの *Enterprise COBOL プログラミング・ガイド* を参照してください。
TYPECHK			✓		OO タイプの適合性に関する規則を強制し、違反についての診断を発行します。 デフォルトは NOTYPECHK です。

表 54. コンパイラー・オプションの比較 (続き)

オプション	適用対象				使用上の注意
	OSVS	VSII	IBM COBOL	Enterprise COBOL	
VBREF VBSUM	✓	✓	✓	✓	プログラム内のすべての動詞タイプの相互参照リストを生成します。OS/VS COBOL のみが VBSUM をサポートします。 デフォルトは NOVBREF です (OS/VS COBOL の場合は NOVBSUM)。
WORD		✓	✓	✓	コンパイラーに、使用すべき予約語テーブルを指示します。インストール先固有の予約語テーブルを使用するには、WORD(table-name) を指定してください。デフォルト予約語テーブルを使用するには、NOWORD を指定してください。 NOWORD がデフォルトです。
XREF		✓	✓	✓	プログラム内で使用されているデータ名およびプロシージャー名のソート済み相互参照リストを生成します。デフォルトは XREF です。 OS/VS COBOL の SXREF オプションに代わって、XREF が使用されるようになりました。
YEARWINDOW			✓	✓	COBOL コンパイラーによるウィンドウ化日付フィールド処理に適用される世紀ウィンドウの最初の年を指定します。
ZWB	✓	✓	✓	✓	英数字フィールドと比較するときに、符号付き数値 DISPLAY フィールドから符号を除去します。ZWB がデフォルトです。

注:

- ✓* COBOL (OS/390 および VM 版) バージョン 2 リリース 2 でのみ使用可能
- ✓** COBOL (OS/390 および VM 版) バージョン 2 リリース 1 および 2 でのみ使用可能

付録 G. コンパイラー限界値の比較

以下の表には、Enterprise COBOL、IBM COBOL、VS COBOL II、および OS/VS COBOL プログラムのコンパイラー限界値がリストされています。

表内の限界値のガイドラインを示します。

- メガバイト (MB) 単位で記述されている限界値は、x メガバイト − 1-B (バイト) として解釈してください。
- キロバイト (KB) 単位で記述されている限界値は、x キロバイト − 1-B (バイト) として解釈してください。
- ギガバイト (GB) 単位で記述されている限界値は、x ギガバイト − 1-B (バイト) として解釈してください。
- B はバイトを表します。
- N/L は制限がないことを表します。
- 脚注は表の最後にあります。

言語エレメント	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
プログラムのサイズ	999,999 行	999,999 行	999,999 行	1-MB 行
リテラルの数	4,194,303-B[1]	4,194,303-B[1]	4,194,303-B[1]	16-KB
リテラルの全長	4,194,303-B[1]	4,194,303-B[1]	4,194,303-B[1]	32-KB (OPT の後)
予約語テーブルの項目の数	1536	1536	1536	N/L
COPY REPLACING . . . BY (COPY ステートメント当たりの項目数)	N/L	N/L	N/L	150
COPY ライブラリーの数	N/L	N/L	N/L	N/L
COPY ライブラリーのブロック・サイズ	32,767-B	32,767-B	32,767-B	16-KB
見出し部				
環境部				
構成セクション				
SPECIAL-NAMES 段落				
関数名 IS	18	18	18	18
UPSI-n ... (スイッチの数)	0 〜 7	0 〜 7	0 〜 7	0 〜 7
英字名 IS ...	N/L	N/L	N/L	N/L
リテラル THRU/ALSO ...	256	256	256	256
入出力セクション				
FILE-CONTROL 段落				

コンパイラー限界値の比較

言語エレメント	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
SELECT ファイル名 ...	最大 65,535 の ファイル名に 外部名を 割り当てる ことが できる	最大 65,535 の ファイル名に 外部名を 割り当てる ことが できる	最大 65,535 の ファイル名に 外部名を 割り当てる ことが できる	最大 64-KB ファイル名に 外部名を 割り当てる ことが できる
ASSIGN システム名 ...	N/L	N/L	N/L	N/L
ALTERNATE RECORD KEY データ名 ...	253	253	253	253
RECORD KEY の長さ	N/L[2]	N/L[2]	N/L[2]	255
RESERVE 整数 (バッファー)	255[3]	255[3]	255[3]	255[3]
I-O-CONTROL 段落				
RERUN ON システム名 ...	32,767	32,767	32,767	32-KB
整数 RECORDS	16,777,215	16,777,215	16,777,215	16-MB
SAME RECORD AREA	255	255	255	255
FOR ファイル名 ...	255	255	255	255
SAME SORT/MERGE AREA	N/L[4]	N/L[4]	N/L[4]	N/L[4]
MULTIPLE FILE ... ファイル名	N/L[4]	N/L[4]	N/L[4]	N/L[4]
データ部				
FILE SECTION				
FD ファイル名 ...	65,535	65,535	65,535	64-KB
LABEL データ名 ... (オプショナル文節が ない場合)	255	255	255	185
ラベル・レコードの長さ	80-B	80-B	80-B	80-B
DATA RECORD dnm ...	N/L[4]	N/L[4]	N/L[4]	N/L[4]
BLOCK CONTAINS 整数	2,147,483,647[5]	2,147,483,647[5]	1,048,575[6]	32760
RECORD CONTAINS 整数	1,048,575[6]	1,048,575[6]	1,048,575[6]	32-KB
項目の長さ	1,048,575[6]	1,048,575[6]	1,048,575[6]	32-KB
SD ファイル名 ...	65,535	65,535	65,535	64-KB
DATA RECORD dnm ...	N/L[4]	N/L[4]	N/L[4]	N/L[4]
ソート・レコードの長さ	32,751-B	32,751-B	32,751-B	32K-16-B
WORKING-STORAGE セクション (EXTERNAL 属性を持たない項目)	134,217,727-B	134,217,727-B	134,217,727-B	1-MB
WORKING-STORAGE セクション (EXTERNAL 属性を持つ項目)	134,217,727-B	134,217,727-B	134,217,727-B	1MB

言語エレメント	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
77 データ名	16,777,215	16,777,215	16,777,215	1-MB
01 ～ 49 データ名	16,777,215	16,777,215	16,777,215	1-MB
88 条件名 ...	N/L	N/L	N/L	N/L
VALUE リテラル ...	N/L	N/L	N/L	N/L
66 RENAMES ...	N/L	N/L	N/L	N/L
PICTURE 文字ストリング	50	30	30	30
数値項目の桁位置	18 (または 31)[7]	18 (または 31)[7]	18	18
数字編集項目の文字位置				
PICTURE の複製	249	249	249	127
PICTURE の複製 (編集)	16,777,215	16,777,215	16,777,215	99999
DBCS ピクチャーの再活動化	32,767	32,767	32,767	99999
グループ項目のサイズ:	8,388,607	8,388,607		
FILE SECTION				
基本項目のサイズ	1,048,575	1,048,575		
VALUE 初期設定	16,777,215	16,777,215	16,777,215	32-KB
(VALUE リテラルの全長)	16,777,215	16,777,215	16,777,215	64-KB
OCCURS 整数				
ODO の合計数	16,777,215	16,777,215	16,777,215	32-KB
ODO のレベル	4,194,303[1]	4,194,303[1]	4,194,303[1]	64-KB[1]
テーブルのサイズ	N/L	N/L	N/L	3
テーブル・エレメント	16,777,215	16,777,215	16,777,215	32-KB
ASCENDING /DESCENDING KEY ...	8,388,607	8,388,607	8,388,607	32-KB
	12	12	12	12
(OCCURS 文節当たり)	256B	256B	256B	256-B
全長	12	12	12	12
INDEXED BY ... (指標名)	65,535	65,535	65,535	64-KB
指標 (指標名) の合計数	32,765	32,765	32,765	32-KB
相対指標のサイズ				
リンケージ・セクション	134,217,727	134,217,727	134,217,727	1-MB
01 + 77 の合計 (データ項目)	N/L	N/L	N/L	255
手続き部				

コンパイラー限界値の比較

言語エレメント	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
プロシージャー + 定数域	4,194,303[1]	4,194,303[1]	4,194,303[1]	1M+32-KB
USING ID ...	32,767	32,767	32,767	N/L
プロシージャー名	1,048,575[1]	1,048,575[1]	1,048,575[1]	64-KB[1]
行当たりの動詞の数	7	7	7	7
(FDUMP/TEST)	32,767	32,767	32,767	511
動詞当たりの添え字	N/L	N/L	N/L	N/L
付けされたデータ名の数	4,194,303[1]	4,194,303[1]	4,194,303[1]	64-KB
ADD ID ...	2,147,483,647	2,147,483,647	2,147,483,647	N/L
ALTER pn1 TO pn2 ...	4,194,303[1]	4,194,303[1]	4,194,303[1]	N/L
CALL ... BY CONTENT ID	16,380	16,380	16,380	N/L
CALL リテラル ...	32,767	32,767	32,767	32-KB
USING ID/ リテラル ...	N/L	N/L	N/L	64-KB
実行単位内のアクティブ・プログラム	N/L	N/L	N/L	N/L
呼び出される名前の	N/L	N/L	N/L	N/L
RES/DYN 数	N/L	N/L	N/L	N/L
CANCEL ID/ リテラル ...	N/L[8]	N/L[8]	N/L[8]	N/L[8]
CLOSE ファイル名 ...	N/L	N/L	N/L	N/L
COMPUTE ID ...	N/L	N/L	N/L	N/L
DISPLAY ID/ リテラル ...	64	64	64	N/L
DIVIDE ID ...	256	256	256	N/L
ENTRY USING ID/ リテラル ...	255	255	255	2031
EVALUATE ... サブジェクト	N/L	N/L	N/L	15
EVALUATE ... WHEN 文節				
GO pn ... DEPENDING	N/L	N/L	N/L	12
INSPECT TALLYING	4092-B[9]	4092-B[9]	4092-B[9]	256
/REPLACING	16[10]	16[10]	16[10]	16[10]
MERGE ファイル名	N/L	N/L	N/L	N/L
ASCENDING	N/L	N/L	N/L	N/L
/DESCENDING KEY ...	N/L	N/L	N/L	N/L
キーの全長	4,194,303	4,194,303	4,194,303	64-KB
USING ファイル名 ...	N/L	N/L	N/L	N/L
MOVE ID/ リテラル TO ID ...	12	12	12	12
MULTIPLY ID ...	N/L	N/L	N/L	N/L
OPEN ファイル名	N/L	N/L	N/L	N/L
PERFORM				
SEARCH ... WHEN ...	N/L	N/L	N/L	12
SEARCH ALL ... WHEN ...	4092-B[9]	4092-B[9]	4092-B[9]	256
SET 指標 /ID ... TO	16[10]	16[10]	16[10]	16[10]
SET 指標 ... UP/DOWN	N/L	N/L	N/L	N/L
SORT ファイル名	N/L	N/L	N/L	N/L
ASCENDING				
/DESCENDING KEY	255	255	255	15
キーの全長	N/L	N/L	N/L	N/L
USING ファイル名 ...	N/L	N/L	N/L	N/L
STRING ID ...				
DELIMITED ID/ リテラル ...				

言語エレメント	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
STRING ID ...				
DELIMITED ID/ リテラル ...				
UNSTRING ID				
DELIMITED ID/ リテラル				
OR ID/ リテラル ...				
OR id/lit ...				
INTO ID/ リテラル ...				
USE ... ON ファイル名 ..				

注:

1. プロシージャー + 定数域についての 4,194,303 バイトの限界値に含まれる項目。
2. コンパイラー限界値はありませんが、VSAM によって 255 バイトに制限されます。
3. QSAM の制限。
4. コメントとして扱われます。制限はありません。
5. OS/390 DFSMS バージョン 2 リリース 10.0 または z/OS によって提供される LBI (ラージ・ブロック・インターフェース) サポートが必要です。以前のリリースの DFSMS を備えた OS/390 システムでは、限界値は 32,767 バイトです。
6. コンパイラー限界値が示されていますが、QSAM によって 32,767 バイトに制限されます。
7. COBOL (OS/390 および VM 版) バージョン 2 リリース 2 以上のバージョンの場合は、ARITH(COMPAT) が有効であれば 18、ARITH(EXTEND) が有効であれば 31 です。
8. コンパイラー限界値が示されていますが、DISPLAY UPON SYSOUT を使用して表示できるデータ項目の最大長は、言語環境プログラムによって 16,384 に制限されます。
9. OPTION 制御ステートメントに EQUALS がコード化されている場合、限界値は 4088 バイトです。
10. SORT の制限。

DISPLAY を OS/VS COBOL プログラムと共に使用する方法の追加情報については、85 ページの『SYSOUT 出力の変更点の理解』を参照してください。

DISPLAY を VS COBOL II プログラムと共に使用する方法の追加情報については、107 ページの『SYSOUT 出力の変更点の理解』を参照してください。

付録 H. QSAM ファイルでのファイル状況 39 の防止

この付録では、QSAM ファイルでよくあるファイル状況 39 問題の防止に役立つ指針を示します。この問題は、ファイル編成、レコード・フォーマット (固定または可変)、レコード長、またはコード・セットの属性の不一致が原因で発生します。

既存ファイルの処理

プログラムで既存ファイルを処理する場合、COBOL プログラムでのそのファイルの記述を、データ・セットのファイル属性と一貫性を持たせてコーディングしてください。たとえば、次のように指定します。

フォーマット V ファイルまたはフォーマット S ファイル	プログラムで指定する最大レコード長は、データ・セットの長さ属性よりも正確に 4 バイト小さいことが必要です。
フォーマット F ファイル	プログラムで指定するレコード長は、データ・セットの長さ属性と正確に一致することが必要です。
フォーマット U ファイル	プログラムで指定する最大レコード長は、データ・セットの長さ属性と正確に一致することが必要です。

注: JCL の情報がデータ・セット・ラベル内の情報をオーバーライドすることを忘れないでください。

レコード長がプログラム内の FD 記入項目とレコード記述から決定される方法についての詳細は、*Enterprise プログラミング・ガイド* を参照してください。

可変長レコードの定義

プログラム内で可変長レコードを定義する最も簡単な方法は、FD 記入項目で RECORD IS VARYING FROM integer-1 TO integer-2 を使用し、integer-2 に適切な値を指定することです。たとえば、データ・セットの長さ属性を 104 (LRECL=104) に決めたと仮定します。最大レコード長が、レベル-01 レコード記述からではなく、(値が指定されている) RECORD IS VARYING 文節から決定されることに注意しながら、以下のコードを使用してプログラム内でフォーマット V ファイルを定義することができます。

```
FILE SECTION.
FD  COMMUTER-FILE-MST
    RECORDING MODE IS V
    RECORD IS VARYING FROM 4 TO 100 CHARACTERS.
01  COMMUTER-RECORD-A           PIC X(4).
01  COMMUTER-RECORD-B           PIC X(75).
```

上記の例で、既存のファイルがフォーマット V ではなくフォーマット U であると仮定します。104 バイトがすべてユーザー・データである場合、以下のコードを使用してプログラム内でファイルを定義することができます。

```
FILE SECTION.
FD  COMMUTER-FILE-MST
    RECORDING MODE IS U
    RECORD IS VARYING FROM 4 TO 104 CHARACTERS.
01  COMMUTER-RECORD-A          PIC X(4).
01  COMMUTER-RECORD-B          PIC X(75).
```

固定長レコードの定義

プログラム内で固定長レコードを定義するには、RECORD CONTAINS integer 文節を使用するか、またはこの文節を省略してすべてのレベル-01 レコード記述が同じ固定サイズになるように指定します。どちらの場合も、データ・セットの長さ属性の値と等しい値を使用してください。実行時に異なるファイルを同じプログラムで処理しようとする場合、それらのファイルの固定長レコードの長さが異なる際に、レコード長の矛盾を避けるために推奨される方法は、RECORD CONTAINS 0 とコーディングすることです。

既存ファイルが ASCII データ・セット (DCB=(OPTCD=Q)) である場合は、プログラムで、そのファイル用の FD 記入項目で CODE-SET 文節を指定する必要があります。

COBOL レコードと一致しない既存ファイルの変換

一致する LRECL を持つ新規ファイルを再割り当てして、既存ファイルから新規ファイルにデータをコピーし、その後、この新規ファイルを入力ファイルとして使用することができます。

新規ファイルの処理

COBOL プログラムが、プログラムの実行前に使用可能にされた新規ファイルにレコードを書き込む場合は、DD ステートメントまたは割り振りで指定するファイル属性が、プログラムで指定した属性と矛盾しないようにしてください。ほとんどの場合、以下の例 (この例は、DD ステートメントとプログラム内の FILE-CONTROL および FD 記入項目との関係を示しています) に示されているように、ファイルを事前定義するときに指定する必要があるのは最小限のパラメーターだけです。

```
JCL DD Statement:

 1
//OUTFILE  DD  DSNAME=OUT171,UNIT=SYSDA,SPACE=(TRK,(50,5)),
//             DCB=(BLKSIZE=400)


/*

Enterprise COBOL  Program Code:

ENVIRONMENT DIVISION.
  INPUT-OUTPUT SECTION.
    FILE-CONTROL.
      SELECT CARPOOL  2
          ASSIGN TO OUTFILE  1
          ORGANIZATION IS SEQUENTIAL
          ACCESS IS SEQUENTIAL.
    .
    .
    .
DATA DIVISION.
  FILE SECTION.
    FD CARPOOL  2
      LABEL RECORD STANDARD
      BLOCK CONTAINS 0 CHARACTERS
      RECORD CONTAINS 80 CHARACTERS
```

図 9. JCL、FILE-CONTROL 記入項目、および FD 記入項目の例

ここで、

1 DD ステートメントの *ddname* は、ASSIGN 文節の *assignment-name* に対応します。

```
//OUTFILE  DD  DSNAME=OUT171 ...
```

以下の *assignment-name* は、DD ステートメントの OUTFILE の *ddname* を指します。

```
ASSIGN TO OUTFILE
```

2 COBOL FILE-CONTROL 記入項目でファイルを指定する場合、そのファイルを *file-name* 用の FD 記入項目で記述する必要があります。

```
SELECT CARPOOL

FD CARPOOL
```

データ・セットの長さ属性を明示的に指定する必要がある場合 (たとえば、ISPF 割り振りパネルを使用する場合、または DD ステートメントが、プログラムで RECORD CONTAINS 0 を使用するバッチ・ジョブのためのものである場合)、次の規則に従ってください。

- フォーマット V およびフォーマット S ファイルでは、プログラムで定義されている長さよりも 4 バイト大きい長さ属性を指定してください。
- フォーマット F およびフォーマット U ファイルでは、プログラムで定義されている長さと同じ長さ属性を指定してください。
- ファイルを OUTPUT としてオープンし、それをプリンターに書き込む場合は、ADV コンパイラー・オプションおよびプログラムで使用された COBOL 言語に

よっては、コンパイラーが紙送り制御文字のための 1 バイトをレコード長に追加することがあります。その場合は、LRECL を指定するときに、追加される 1 バイトを考慮に入れてください。

たとえば、可変長レコードを持つファイルについての以下のコードがプログラムに含まれているとします。

```
FILE SECTION.
  FD  COMMUTER-FILE-MST
      RECORDING MODE IS V
      RECORD CONTAINS 10 TO 50 CHARACTERS.
  01  COMMUTER-RECORD-A          PIC X(10).
  01  COMMUTER-RECORD-B          PIC X(50).
```

DD ステートメントまたは割り振りで指定する LRECL は 54 にする必要があります。

COBOL によって動的に作成されたファイルの処理

Enterprise COBOL は、以下のすべての状況が存在する場合にファイルを動的に割り振ります。

- CBLQDA(ON) ランタイム・オプションが有効である。
- ファイルの DD 名が明示的に割り振られていない。
- 同じ名前の環境変数が設定されていない。
- COBOL プログラムが書き込みを行うためのファイルをオープンする。

ファイルがオープンされるとき、プログラムで指定された属性が使用されます。

CBLQDA(OFF) が有効な場合は、エラーが生成されます。

付録 I. リンケージ・エディターのデフォルトのオーバーライド

この付録では、RENT および RMODE コンパイラー・オプションの使用に基づいて Enterprise COBOL コンパイラーによって割り当てられたデフォルトの AMODE および RMODE 設定値をオーバーライドするために必要な指示を記述します。

デフォルトの設定値をオーバーライドしてはならないとき

以下の場合は、デフォルト AMODE/RMODE 設定値をオーバーライドしないでください。

AMODE

NORES を指定してコンパイルされた VS COBOL II プログラムまたは任意の OS/VS COBOL プログラムを含んでいるロード・モジュールの場合、リンケージ・エディターのオーバーライドとして AMODE ANY または AMODE 31 を指定しないでください。唯一の例外は、プログラムが、システムによってまたはシステム・サービスを介して呼び出される外部入り口点であり、アプリケーションの論理が、適切な AMODE 切り替えによって、そのプログラムが AMODE 24 で入られることを保証できる場合です。このようなプログラムは、他のプログラムを静的に呼び出すときは、AMODE を切り替えません。

RMODE

NORES および NORENT を指定してコンパイルされた VS COBOL II プログラムまたは任意の OS/VS COBOL プログラムを含んでいるロード・モジュールの場合、リンケージ・エディターのオーバーライドとして RMODE ANY を指定しないでください。これは、コンパイラーによって生成されたオブジェクト・モジュールに含まれている特定の制御ブロックが 16MB 境界より下に置かれる必要があるためです。

デフォルトの設定値をオーバーライドするとき

Enterprise COBOL と OS/VS COBOL の両方のプログラムを含んでいるロード・モジュールの場合、ロード・モジュールが、NORENT を指定してコンパイルされた Enterprise COBOL プログラムを含んでいるときは、デフォルト AMODE 設定値をオーバーライドして AMODE(24) にする必要があります。(RENT を指定してコンパイルされたプログラムの場合は、処置は必要ありません。リンケージ・エディターが正しい AMODE 設定を自動的に割り当てます。)

デフォルトをオーバーライドする方法

デフォルトをオーバーライドするには、以下のいずれかを使用して AMODE または RMODE を指定してください。

- リンク・エディット・ジョブ・ステップの EXEC ステートメント

  //LKEDEXECPGM=*programname*,
  //PARM='AMODE=*xx*,RMODE=*yy*'

- リンケージ・エディター・モード制御ステートメント

```
MODE AMODE(xx),RMODE(yy)
```

- 以下の TSO コマンド LINK または LOADGO のいずれか

  LINK(*dsn-list*) AMODE(*xx*) RMODE(*yy*)

  LOADGO(*dsn-list*) AMODE(*xx*) RMODE(*yy*)

指定できる *xx* と *yy* およびリンケージ・エディター・モード制御ステートメントについては、*リンケージ・エディターおよびローダー 使用者の手引き* を参照してください。

リンケージ・エディターは、プログラムの AMODE 属性を使用して、ATTACH、LINK、XCTL、または LOAD/BASSM によって呼び出されたプログラムが 24 ビットまたは 31 ビットのどちらのアドレッシング・モードで制御を受け取るかを判別します。ローダーは、RMODE 属性を使用して、プログラムのロード先が 16MB より下の仮想記憶域でなければならないか、それとも仮想記憶域内の任意の場所 (16 MBより上でも下でも) でよいかを判別します。

付録 J. リンク・エディットの例

この付録では、ロード・モジュール内の現行ライブラリー・ルーチンを言語環境プログラム・ライブラリー・ルーチンで置き換える方法を示す JCL の例を示します。SCEESAMP データ・セットに、OS/VS COBOL または VS COBOL II ロード・モジュールを再リンク・エディットするときに役立つ 3 つのサンプル・ジョブ (IGZWRLKA、IGZWRLKB、および IGZRLKC) が入っています。

リンク・エディットの例

```
//*********************************************************************
//*                                                                   *
//*  RELINK A LOAD MODULE THAT HAS BOTH OS/VS COBOL PROGRAMS          *
//*  AND VS COBOL II PROGRAMS WITH Language Environment.              *
//*                                                                   *
//*********************************************************************
//*
//LINK   EXEC  PGM=IEWL,PARM='LIST,MAP,XREF'
//SYSPRINT DD SYSOUT=*
//*********************************************************************
//*   CHANGE 'ZZZZZZ.SCEELKED' IN THE FOLLOWING DD STATEMENT TO       *
//*   THE Language Environment SCEELKED DATA SET NAME.                *
//*                                                                   *
//*   CHANGE 'XXXXXX' IN THE FOLLOWING DD STATEMENT TO                *
//*   THE DATA SET NAME WHICH CONTAINS THE LOAD MODULE.               *
//*                                                                   *
//*   CHANGE 'YYYYYY' IN THE FOLLOWING DD STATEMENT TO                *
//*   THE DATA SET NAME WHICH THE RELINK-EDITED LOAD                  *
//*   MODULE SHOULD BE SAVED INTO.                                    *
//*                                                                   *
//*********************************************************************
//SYSLIB   DD  DSN=ZZZZZZ.SCEELKED,DISP=SHR
//LOADLIB  DD  DSN=XXXXXX,DISP=SHR
//SYSLMOD  DD  DSN=YYYYYY,DISP=SHR
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(2,2)),DISP=NEW
//*********************************************************************
//*   CHANGE 'UUUUUU' IN THE FOLLOWING INCLUDE STATEMENT              *
//*   TO THE LOAD MODULE NAME.                                        *
//*                                                                   *
//*   CHANGE 'VVVVVV' IN THE FOLLOWING NAME STATEMENT TO              *
//*   THE RELINK-EDITED LOAD MODULE NAME.                             *
//*                                                                   *
//*   CHANGE 'EEEEEE' IN THE FOLLOWING ENTRY STATEMENT TO             *
//*   THE RELINK-EDITED LOAD MODULE ENTRY POINT NAME OR               *
//*   OMIT THE ENTRY STATEMENT IF IT IS NOT REQUIRED.                 *
//*                                                                   *
//*********************************************************************
//SYSLIN   DD  *
  REPLACE ILBOxxxx   1
    .
    .
    .
  REPLACE IGZxxxx   2
    .
    .
    .
  INCLUDE LOADLIB(UUUUUU)
  ENTRY   EEEEEE
  NAME    VVVVVV(R)
/*
```

ここで、 **1** および **2** は以下のものを表します。

1

```
REPLACE ILBOABN
REPLACE ILBOACP
REPLACE ILBOACS
REPLACE ILBOANE
REPLACE ILBOANF
REPLACE ILBOATB
REPLACE ILBOBEG
REPLACE ILBOBID
REPLACE ILBOBIE
REPLACE ILBOBII
REPLACE ILBOBUG
REPLACE ILBOCHN
REPLACE ILBOCJS
REPLACE ILBOCKP
REPLACE ILBOCLS
REPLACE ILBOCMM
REPLACE ILBOCOM0
REPLACE ILBOCT1
REPLACE ILBOCVB
REPLACE ILBODBE
REPLACE ILBODBG
REPLACE ILBODCI
REPLACE ILBODSP
REPLACE ILBODSS
REPLACE ILBODTE
REPLACE ILBOD01
REPLACE ILBOD10
REPLACE ILBOD11
REPLACE ILBOD12
REPLACE ILBOD13
REPLACE ILBOD14
REPLACE ILBOD20
REPLACE ILBOD21
REPLACE ILBOD22
REPLACE ILBOD23
REPLACE ILBOD24
REPLACE ILBOD25
REPLACE ILBOD26
REPLACE ILBOEFL
REPLACE ILBOERR
REPLACE ILBOETB
REPLACE ILBOEXT
REPLACE ILBOFLW
REPLACE ILBOFPW
REPLACE ILBOGDO
REPLACE ILBOGPW
REPLACE ILBOIDB
REPLACE ILBOIDR
REPLACE ILBOIDT
REPLACE ILBOIFB
REPLACE ILBOIFD
REPLACE ILBOINS
REPLACE ILBOINT
REPLACE ILBOITB
REPLACE ILBOIVL
REPLACE ILBOLBL
REPLACE ILBOMRG
REPLACE ILBOMSG
REPLACE ILBOMSC
REPLACE ILBONBL
REPLACE ILBONED
REPLACE ILBONTR
REPLACE ILBOOCR
REPLACE ILBOPRM
REPLACE ILBOPTV
REPLACE ILBOQIO
REPLACE ILBOQSS
REPLACE ILBOQSU
REPLACE ILBOREC
REPLACE ILBORNT
REPLACE ILBOSAM
REPLACE ILBOSCD
REPLACE ILBOSCH
REPLACE ILBOSDB
REPLACE ILBOSGM
REPLACE ILBOSMG
REPLACE ILBOSMV
REPLACE ILBOSND
REPLACE ILBOSNT
REPLACE ILBOSPI
REPLACE ILBOSPN
REPLACE ILBOSPA
REPLACE ILBOSQA
REPLACE ILBOSRT
REPLACE ILBOSRV
REPLACE ILBOSSN
REPLACE ILBOSTG
REPLACE ILBOSTI
REPLACE ILBOSTN
REPLACE ILBOSTR
REPLACE ILBOSTT
REPLACE ILBOSYN
REPLACE ILBOTC0
REPLACE ILBOTC2
REPLACE ILBOTC3
REPLACE ILBOTEF
REPLACE ILBOTRN
REPLACE ILBOUST
REPLACE ILBOUTB
REPLACE ILBOVCO
REPLACE ILBOVIO
REPLACE ILBOVMO
REPLACE ILBOVOC
REPLACE ILBOVTR
REPLACE ILBOWAT
REPLACE ILBOWTB
REPLACE ILBOXDI
REPLACE ILBOXMU
REPLACE ILBOXPR
```

2

```
REPLACE IGZCA2D
REPLACE IGZCACP
REPLACE IGZCACS
REPLACE IGZCANE
REPLACE IGZCANF
REPLACE IGZCBID
REPLACE IGZCBUG
REPLACE IGZCCCO
REPLACE IGZCCLS
REPLACE IGZCCTL
REPLACE IGZCCVB
REPLACE IGZCD2A
REPLACE IGZCDIF
REPLACE IGZCDSP
REPLACE IGZCFDP
REPLACE IGZCFDW
REPLACE IGZCFPW
REPLACE IGZCGDR
REPLACE IGZCIDB
REPLACE IGZCINS
REPLACE IGZCIN1
REPLACE IGZCIN2
REPLACE IGZCIPS
REPLACE IGZCIVL
REPLACE IGZCKCL
REPLACE IGZCLNK
REPLACE IGZCMSF
REPLACE IGZCMST
REPLACE IGZCONV
REPLACE IGZCONVX
REPLACE IGZCSCH
REPLACE IGZCSMV
REPLACE IGZCSPA
REPLACE IGZCSPC
REPLACE IGZCSSN
REPLACE IGZCSTG
REPLACE IGZCTCO
REPLACE IGZCULE
REPLACE IGZCUST
REPLACE IGZCVIN
REPLACE IGZCVMO
REPLACE IGZCXDI
REPLACE IGZCXFR
REPLACE IGZCXMU
REPLACE IGZCXPR
REPLACE IGZEABX
REPLACE IGZEABN
REPLACE IGZEBRG
REPLACE IGZEBST
REPLACE IGZECKP
REPLACE IGZECMS
REPLACE IGZEDBR
REPLACE IGZEDBW
REPLACE IGZEDTE
REPLACE IGZEINP
REPLACE IGZEMSG
REPLACE IGZENRI
REPLACE IGZENRT
REPLACE IGZEOPD
REPLACE IGZEOPT
REPLACE IGZEOUT
REPLACE IGZEPRM
REPLACE IGZEPTV
REPLACE IGZEQBL
REPLACE IGZEQOC
REPLACE IGZERRE
REPLACE IGZESAT
REPLACE IGZESMG
REPLACE IGZESNP
REPLACE IGZESPM
REPLACE IGZESTA
REPLACE IGZETRM
REPLACE IGZETUN
REPLACE IGZEVAM
REPLACE IGZEVEX
REPLACE IGZEVIO
REPLACE IGZEVOC
REPLACE IGZEVOP
REPLACE IGZEVSV
REPLACE IGZTCAM2
REPLACE IGZTCAM4
REPLACE IGZTCM21
REPLACE IGZTCM41
REPLACE IGZTCM42
```

付録 K. DB2 coprocessor の組み込み

coprocessor アプローチによって、SQL ステートメントを含む COBOL プログラムで DB2 プリコンパイラーを使用してプリコンパイルする必要がなくなります。

coprocessor アプローチでは、COBOL コンパイラーを使用してネイティブの COBOL とソース・プログラムに組み込まれた SQL ステートメントの両方を処理します。SQL ステートメントが検出された場合、コンパイラーは DB2 coprocessor とインターフェースをとります。DB2 coprocessor は適切な処置を行ってから、通常は、生成するネイティブ言語ステートメントを指示してコンパイラーに制御を戻します。

個別プリコンパイラー・アプローチは現在でも Enterprise COBOL でサポートされますが、coprocessor アプローチの方が望ましく、推奨されるソリューションです。coprocessor アプローチは使用可能度が高く、機能面でも優れています。特に、coprocessor ソリューションを使用すると、デバッグ・ツールを使用した COBOL アプリケーションの対話式デバッグ機能が拡張されます。これは、DB2 プリコンパイラーによって生成された拡張ソースではなく、元のソース・レベルでアプリケーションをデバッグすることができるためです。

coprocessor アプローチには DB2 バージョン 7 以上が必要です。coprocessor アプローチには以下のような利点があります。

- デバッグ・ツールを使用した COBOL アプリケーションの対話式デバッグ機能が強化される。CICS トランスレーターによって生成された拡張ソースのレベルではなく、元のソース・レベルでアプリケーションをデバッグすることができます。
- ソース・プログラムの変換済みでコンパイルされていないバージョンを格納する中間データ・セットが不要である。
- 出力リストは 2 つではなく 1 つだけである。
- EXEC SQL ステートメントの入っているネストされたプログラムを個別のファイルに保管し、COPY ステートメントを使ってこれをインクルードすることができる。
- REPLACE ステートメントを EXEC SQL ステートメントに影響させることができる。

DB2 プリコンパイラーの使用例を以下に示します。

```
//DB2INT JOB  . . .,
//    NOTIFY=JDOE,MSGCLASS=A,CLASS=A,TIME=(0,5),          MIN00020
//    REGION=10M,MSGLEVEL=(1,1)                           MIN00030
//COB      EXEC PGM=IGYCRCTL,
//   PARM=(QUOTE,NODYNAM,ADV,'BUF(12288)',SOURCE,NOXREF)
//STEPLIB  DD DSN=IGY.V3R1M0.SIGYCOMP,DISP=SHR
//         DD DSN=DSN710.SDSNLOAD,DISP=SHR
//DBRMLIB  DD DSN=JDOE.DBRMLIB.DATA(COBTEST),DISP=SHR
//SYSIN    DD *
           CBL SQL('HOST(COB2),QUOTE,APOSTSQL,SOURCE,XREF'),LIB
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  COBTEST.
```

```
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  RES                PIC X(10).
           EXEC SQL
             INCLUDE SQLCA
           END-EXEC.
       PROCEDURE DIVISION.
           EXEC SQL
             SELECT COL1           INTO :RES     FROM TABLE1
           END-EXEC.
//SYSLIN   DD DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//            SPACE=(800,(500,500))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT2   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT3   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT4   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT5   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT6   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT7   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//LKED.SYSIN DD  *
 INCLUDE SYSLIB(DSNELI)
 INCLUDE SYSLIB(DSNTIAR)
 NAME COBTEST(R)
//*INDRUN  EXEC TSOBATCH,DB2LEV=DB2A,COND=(4,LT)
//*YSTSIN  DD *
 DSN SYSTEM(V71A)
 FREE PLAN(COBPLAN)
 BIND PLAN(COBPLAN) MEMBER(COBTEST)
 RUN PROGRAM(COBTEST) PLAN(COBPLAN)
```

組み込みの SQL coprocessor の例を以下に示します。

```
//DB2INT JOB  (JDOE,E264,090,D48),'John Doe',
//    NOTIFY=JDOE,MSGCLASS=A,CLASS=A,TIME=(0,5),          MIN00020
//    REGION=10M,MSGLEVEL=(1,1)                           MIN00030
//COB      EXEC PGM=IGYCRCTL,
//   PARM=(QUOTE,NODYNAM,ADV,'BUF(12288)',SOURCE,NOXREF)
//STEPLIB  DD DSN=IGY.V3R1M0.SIGYCOMP,DISP=SHR
//         DD DSN=DSN710.SDSNLOAD,DISP=SHR
//DBRMLIB  DD DSN=JDOE.DBRMLIB.DATA(COBTEST),DISP=SHR
//SYSIN    DD *
           CBL SQL('HOST(COB2),QUOTE,APOSTSQL,SOURCE,XREF'),LIB
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  COBTEST.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  RES                PIC X(10).
           EXEC SQL
             INCLUDE SQLCA
           END-EXEC.
       PROCEDURE DIVISION.
           EXEC SQL
             SELECT COL1           INTO :RES     FROM TABLE1
           END-EXEC.
//SYSLIN   DD DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//            SPACE=(800,(500,500))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT2   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT3   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT4   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT5   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
```

```
//SYSUT6   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT7   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//LKED.SYSIN DD  *
 INCLUDE SYSLIB(DSNELI)
 INCLUDE SYSLIB(DSNTIAR)
 NAME COBTEST(R)
//*INDRUN  EXEC TSOBATCH,DB2LEV=DB2A,COND=(4,LT)
//*YSTSIN  DD *
 DSN SYSTEM(V71A)
 FREE PLAN(COBPLAN)
 BIND PLAN(COBPLAN) MEMBER(COBTEST)
 RUN PROGRAM(COBTEST) PLAN(COBPLAN)
```

付録 L. IMS の考慮事項

OS/VS COBOL プログラム、VS COBOL II プログラム、IBM COBOL プログラム、および Enterprise COBOL プログラムの任意の組み合わせを言語環境プログラムのもとで実行する場合、以下の事柄を知っている必要があります。

- サポートされない VS COBOL II 機能
- IMS で実行されるプログラムに関係のあるコンパイラー・オプション
- IMS のもとで実行するための COBOL プログラムのコンパイルおよびリンク
- ENDJOB/NOENDJOB コンパイラー・オプション要件
- プリロードをお勧めするモジュール
- IMS で CBLTDI を使用する条件処理
- OS/VS COBOL プログラムを実行するときのパフォーマンスの考慮
- ロードされたモジュールを判別するための GTF トレースの使用
- サポートされない DFSPCC20 変更

サポートされない VS COBOL II 機能

サポートされない BLDL ユーザー出口

VS COBOL II の BLDL ユーザー出口は、言語環境プログラムではサポートされません。これに代わるものはありません。ただし、言語環境プログラムにはロード通知出口があり、これがユーザーの要件を満たすことがあります。ロード通知出口については、*言語環境プログラム (OS/390 版) カスタマイズ* を参照してください。

サポートされない LIBKEEP

VS COBOL II の LIBKEEP ランタイム・オプションは、言語環境プログラムではサポートされません。LIBKEEP の代わりに使用できるものについては、113 ページの『LIBKEEP を使用する既存のアプリケーション』を参照してください。

IMS で実行されるプログラムに関係のあるコンパイラー・オプション

表 55 に、COBOL IMS アプリケーションに関係のある Enterprise COBOL コンパイラー・オプションをリストします。

表 55. IMS で実行される Enterprise COBOL プログラムに関係のあるコンパイラー・オプション

コンパイラー・オプション	説明
RENT	RENT コンパイラー・オプションを使用すると、Enterprise COBOL は再入可能コードを生成するため、COBOL モジュールを LPA (リンク・パック域) または ELPA (拡張リンク・パック域) に入れることができ、その結果、IMS のもとの複数の従属領域で共用することができます。また、LPA/ELPA にはストレージ保護キーがあるため、モジュールを上書きすることはできません。 RENT を使用すると、異なるオプションでコンパイルしなくても、IMS プログラムをプリロードまたは非プリロードのどちらのモードででも実行でます。

IMS のもとで実行するための COBOL プログラムのコンパイルおよびリンク

IMS 環境で最高のパフォーマンスを得るため、RENT コンパイラー・オプションを使用します。このオプションを使用すると、COBOL で再入可能コードが生成されます。その後、アプリケーション・プログラムをプリロード・モード (プログラムが常にストレージにある) または非プリロード・モードのいずれでも実行することができます。別のオプションで再コンパイルする必要はありません。

IMS を使用すると、COBOL プログラムをプリロードできます。このプリロードによってパフォーマンスが向上します。これは、プログラムがすでにストレージにあると (必要が生じるたびにライブラリーから取り出すよりも) プログラムに対する後続の要求をより速く処理できるためです。

RENT コンパイラー・オプションを使用して、プリロードして実行するプログラム、またはプリロードおよび非プリロードの両方で実行するプログラムをコンパイルする必要があります。COBOL プログラムを含むロード・モジュールをプリロードするときは、そのロード・モジュール内のすべての COBOL プログラムを RENT オプションでコンパイルする必要があります。

Enterprise COBOL プログラム、IBM COBOL プログラム、VS COBOL II プログラム、および OS/VS COBOL プログラムの任意の組み合わせを使用するアプリケーションでは、以下のコンパイラー・オプションが推奨されます。

表 56. COBOL プログラムの任意の組み合わせを使用するアプリケーションで推奨されるコンパイラー・オプション

Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
RENT	RENT	RENT および RES	プリロード・プログラムの場合は RES および NOENDJOB 非プリロード・プログラムの場合は RES および ENDJOB

RENT オプションを指定してコンパイルされたプログラムを LPA に入れることができます。LPA では、プログラムを IMS 従属領域間で共用できます。

16-MB の境界を超えて実行するために、アプリケーション・プログラムを IMS 環境に応じて RENT または NORENT RMODE(ANY) を指定してコンパイルする必要があります。

IMS の場合、IMS アプリケーション・プログラム用のデータは 16-MB 境界を超えて常駐できます。また、IMS サービスを使用するプログラムに DATA(31) RENT、または RMODE(ANY) NORENT を使用できます。

IMS のもとで COBOL プログラムを正常に実行するために推奨されるリンク・エディット属性は以下のとおりです。

- RENT コンパイラー・オプションを指定してコンパイルされた COBOL プログラムのみを含む RENT ロード・モジュールとしてリンクします。
- COBOL RENT プログラムとその他のプログラムが混在するロード・モジュールをリンクするには、他のプログラムについて推奨されるリンク・エディット属性を使用します。

ENDJOB/NOENDJOB コンパイラー・オプション要件

OS/VS COBOL プログラムと Enterprise COBOL プログラムの混合を実行する場合、少なくとも 1 つの OS/VS COBOL プログラムが ENDJOB オプションでコンパイルされていなければなりません (それらのプログラムがプリロードされていない場合)。

ENDJOB オプションを指定すると、COBOL アプリケーションおよび言語環境プログラム・ランタイム環境がプログラム終了時に完全に終結処理されるようになります。

注: OS/VS COBOL および Enterprise COBOL と同時に LRR を使用する場合は、ランタイム環境は完全には終結処理されません。LRR は NOENDJOB 動作を強制するからです。

プリロードの要件

ILBOCOM をプリロードする場合、以下のライブラリー・ルーチンもプリロードする必要があります。

- ILBOCMM、ILBONTR、および ILBOSRV

- ILBOACS、ILBOCVB、および ILBOINS (OS/VS COBOL RES プログラムが INSPECT 動詞または UNSTRING 動詞 (あるいはその両方) を使用する場合)

言語環境プログラムのもとで最後に使用された状態の動作

OS/VS COBOL の NOENDJOB/ENDJOB コンパイラー・オプションは、言語環境プログラム・ライブラリーを使用して実行するときは、OS/VS COBOL ライブラリーを使用して実行するときとは別の結果を生成します。NOENDJOB を指定して OS/VS COBOL のもとで実行するときは、サブプログラムには常に最後に使われた状態で入ります。

言語環境プログラム・ランタイム環境では、RES および NOENDJOB を指定してコンパイルされたプリロードされていない OS/VS COBOL サブプログラムが終了すると、プログラムおよび内部作業域は削除されます。外部作業域 (GETMAIN を使用して獲得されたストレージ) およびライブラリー・ルーチンは動的ストレージに残ります。NOENDJOB ではなく ENDJOB を使用した場合、外部作業域およびライブラリー・ルーチンも削除されることを除けば、本質的な違いはありません。

Enterprise COBOL プログラムおよびそのリソース (すべての作業域およびライブラリー・ルーチン) は常に終了時に削除されるので、ENDJOB および NOENDJOB が言語環境プログラム・ランタイム環境で機能する方法によって、Enterprise COBOL プログラムと OS/VS COBOL プログラムの両方が、最後に使われた状態に関して同様に扱われます。

OS/VS COBOL ランタイム環境では、プリロードされていないプログラムを起動する IMS トランザクションを実行するときに強く推奨されるコンパイラー・オプションは ENDJOB でした。

VS COBOL II ランタイム環境で LIBKEEP ランタイム・オプションを使用する場合、ENDJOB を指定してコンパイルされた OS/VS COBOL プログラムは NOENDJOB と同様に扱われました。プリロードされた OS/VS COBOL プログラムの場合は NOENDJOB が必須でした。

注：LRR を使用するときは、NOENDJOB 動作が常に有効です。

プログラムが最後に使われた状態で残っている場合

プログラムが COBOL トランザクション間で最後に使われた状態で残るのは、以下のすべての条件が該当する場合だけです。

- それは NOENDJOB オプションを指定してコンパイルされた OS/VS COBOL プログラムであるか、または LRR が使用されている。
- それは REUS オプションを指定してリンク・エディットされている。
- それはプリロードされている。
- それは動的に呼び出されるサブプログラムであるか、または動的に呼び出されたサブプログラムによって呼び出された、静的に呼び出されるサブプログラムである。

プリロードをお勧めするモジュール

IMS では、アプリケーション・プログラムをプリロード (つまり、使用後もストレージに常駐) させておくことができます。プリロードによってパフォーマンスが向上することがあります。プログラムがすでにストレージに常駐していれば、そのプログラムに対する要求をより速く処理できます。プログラムを外部メディアからストレージへ移すために時間が浪費されることがありません。

Enterprise COBOL プログラム

プリロードをお勧めするモジュールのリストについては、Web サイト `http://www.ibm.com/software/ad/cobol` の Library Section にある *IBM Enterprise COBOL Version 3 Release 1 Performance Tuning Paper* を参照してください。Library セクションに行くには `Library` リンクをクリックします。

OS/VS COBOL プログラム

表 57 は、OS/VS COBOL RES コンパイラー・オプションを使用する場合にプリロードする OS/VS COBOL 互換サブルーチンのサンプル・リストを示しています。OS/VS COBOL プログラムを言語環境プログラムで実行する場合は、*Performance Paper* にリストされているモジュールもプリロードすることができます。

表 57. プリロードする ILBO モジュールのサンプル・リスト

ILBOCHN0	ILBOCMM0	ILBOCOM0
ILBOCVB0	ILBODTE0	ILBOETB0
ILBOGDO0	ILBOINS0	ILBOITB0
ILBONTR0	ILBOSCH0	ILBOUST0
ILBOWTB0	ILBOSRV0	ILBOSTG0
ILBOSTT2	ILBOTRN0	

言語環境プログラムのもとで実行するときは、

- OS/VS COBOL ライブラリーの使用時にプリロードしたのと同じ ILBO ライブラリー・ルーチンを引き続きプリロードしてください (これらの ILBO ルーチンは言語環境プログラムに組み込まれています)。
- OS/VS COBOL ライブラリーを使用して実行される OS/VS COBOL プログラム用に ILBOSTT をプリロードしていた場合は、ILBOSTT2 をプリロードして (それをプリロード・リストに 2 回入れて) ください。

IMS で CBLTDLI を使用する条件処理

1. その条件を条件処理で処理しないでください。エラーを条件処理で処理しようとすると、IMS データベースの保全性が危険にさらされます。
2. ABTERMENC(ABEND) および TRAP(ON) ランタイム・オプションを使用して、アプリケーションが異常終了するようにし、すべての異常終了をオペレーティング・システムの異常終了に変換して IMS ロールバックが発生するようにしてください。

IMS バージョン 2 およびバージョン 3 での違い

言語環境プログラムのもとで IMS/ESA バージョン 2 リリース 2 (または PTF UN49280 が適用されていないバージョン 3 リリース 1) と共に実行される

IMS の考慮事項

CBLTDLI または ASMTDLI (PL/I 以外のルーチンから出された) は、IMS への呼び出しおよび IMS からの戻りを記録しません。

IMS と言語環境プログラムとの間の条件処理を調整しなければ、プログラム割り込みまたは異常終了が発生した場合、言語環境プログラム条件マネージャーは問題がアプリケーションで発生したのか IMS で発生したのかを判別できません。

条件が発生した場合にデータベースが汚染されないようにするために、次のようにしてください。

1. その条件を条件処理で処理しないでください。エラーを条件処理で処理しようとすると、IMS データベースの保全性が危険にさらされます。
2. ABTERMENC(ABEND) および TRAP(ON) ランタイム・オプションを使用して、アプリケーションが異常終了するようにし、すべての異常終了をオペレーティング・システムの異常終了に変換して IMS ロールバックが発生するようにしてください。

表 58は、各種の IMS リリースで条件処理を使用できる場合を示しています。

表 58. IMS (リリースごと) での条件処理の制限

インターフェース	IMS V2	IMS V3 (PTF UN49280 なし)	IMS V3 (PTF UN49280 あり)	IMS V4 以上
CEETDLI	N/A	N/A	N/A	はい
CBLTDLI	いいえ	いいえ	はい	はい
PLITDLI	はい	はい	はい	はい
CTDLI	はい	はい	はい	はい
ASMTDLI (非 PL/I)	いいえ	いいえ	はい	はい
ASMTDLI (PL/I)	はい	はい	はい	はい

OS/VS COBOL プログラムを実行するときのパフォーマンスの考慮

IMS 領域でスケジュールされたほとんどのプログラムが OS/VS COBOL プログラムである場合は、言語環境プログラムのストレージ関連のランタイム・オプションに低い値を指定すると、パフォーマンスが向上することがあります。たとえば、以下のストレージ値を使用することができます。

```
STACK(16K)
HEAP(4K,4K,ANY,4K,4K)
BELOWHEAP(4K)
ANYHEAP(4K)
```

ロードされたモジュールを判別するための GTF トレースの使用

一部のお客様は、GTF トレース情報を収集し、SVC 8 (LOAD) トレース情報を調べて、どのアプリケーション・プログラムがロードされているかを判別しています。言語環境プログラムは SVC 8 と SVC 122 の両方を使用してプログラムをロードします。

サポートされない DFSPCC20 変更

OS/VS COBOL では、IMS プログラム・コントローラー (DFSPCC20) へのユーザー変更のあるプリロードされたプログラムおよびプリロードされていないプログラムのどちらの場合も、ENDJOB オプションを使用することができました。

このユーザー変更には、特定のプリロード済みサブルーチンのアドレスを保管するコードが含まれていますが、そのコードは実行時に非再現性問題の原因となる可能性がありました。言語環境プログラムのもとで実行する前に、このユーザー変更 (または、同じタスクを行うユーザー提供コード) を除去する必要があります。

付録 M. TSO の考慮事項

この付録では、TSO で実行されるプログラムの移行に関する考慮事項を説明します。REXX EXEC の使用についての情報を示します。

REXX exec の使用

REXX EXEC から COBOL プログラムを実行するときは、異なる ″address″ オプションの使用に関してパラメーター・リスト・フォーマットの違いを知っている必要があります。’Address TSO’ (デフォルト) または ’Address ATTCHMVS’ を使用するときは、プログラム・パラメーターと言語環境プログラム・ランタイム・オプションの両方が処理されます。’Address LINKMVS’ を使用するときは、ランタイム・オプションは処理されませんが、それらはプログラム・パラメーターとして COBOL プログラムに渡されます。

パラメーター・リスト・フォーマットおよび保管域規則の違いのため、’Address LINK’、’Address ATTACH’、’Address LINKPGM’、および ’Address ATTCHPGM’ はサポートされません。

付録 N. 特記事項

本書に記載の製品、プログラム、またはサービス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、プログラム、またはサービスについては、日本アイ・ビー・エム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。IBM の有効な知的所有権、またはその他の法的に保護された権利に従い、IBM 製品、プログラム、またはサービスに代えて、機能的に同等な製品、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 によって明示的に指定されたものを除き、他社の製品と組み合わせた場合の操作の評価と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。使用許諾については、下記の宛先に書面にてご照会ください。

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

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, Department HHX/H3
555 Bailey Avenue
San Jose, CA 95141-1099
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。

プログラミング・インターフェース情報

本書は、IBM Enterprise COBOL for z/OS and OS/390 を使用してプログラムを作成する際に役立ちます。本書は、IBM Enterprise COBOL for z/OS and OS/390 が提供している汎用プログラミング・インターフェースとそれに関連する情報を記述しています。汎用プログラミング・インターフェースにより、お客様は IBM Enterprise COBOL for z/OS and OS/390 の機能を使用するプログラムを書くことができます。

商標

以下は、IBM Corporation の商標です。

AIX
C/370
CICS
CICS/ESA
COBOL/370
DB2
DFSMS
DFSORT
IBM
IMS
IMS/ESA
Language Environment
MVS
MVS/ESA
OpenEdition
Operating System/2
OS/2
OS/390
S/370
SOMobjects
System Object Model
System/370
VisualAge
VSE/ESA
WebSphere
z/OS

他の会社名、製品名、およびサービス名などは、それぞれ各社の商標または登録商標です。

Microsoft、Windows、Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

参考文献

IBM Enterprise COBOL for z/OS and OS/390

コンパイラーおよび実行時プログラム 移行ガイド、GC88-9118
カスタマイズ・ガイド、GC88-9119
Debug Tool User's Guide、SC27-1573
Debug Tool Reference Manual and Messages、SC27-1575
Fact Sheet、GC27-1407
言語解説書、SC88-9117
Licensed Program Specifications、GC27-1411
プログラミング・ガイド、SC88-9121

z/OS 言語環境プログラム

概念、SA88-8555
デバッグのガイド、GA88-8548
ランタイム・メッセージ、SA88-8554
カスタマイズ、SA88-8552
プログラミング・ガイド、SA88-8549
プログラミング・リファレンス、SA88-8550
ランタイム マイグレーション・ガイド、GA88-8553
ILC (言語間通信) アプリケーションの作成、SA88-8551

言語環境プログラム (OS/390 版)

概念、GC88-7592
デバッグのガイドおよびランタイム・メッセージ、SC88-7596
カスタマイズ、SC88-7595
プログラミング・ガイド、SC88-7593
プログラミング・リファレンス、SC88-7594
ランタイム 移行ガイド、SC88-7598
ILC (言語間通信) アプリケーションの作成、SC88-7597

関連資料

- **IBM C/370**

  移行の手引き、SC88-7046
  プログラミングの手引き、N:SC09-1356
  概説書、GC88-7005

Diagnosis Guide、LY09-1806

Licensed Program Specifications、GC09-1357

参照要約、SX88-7014

- **IBM PL/I for MVS & VM**

  Fact Sheet、GC26-3112

  Licensed Program Specification、GC26-3116

  コンパイラーと実行時の移行の手引き、SC88-7220

  導入およびカストマイズ (MVS)、SC88-7221

  導入およびカストマイズ (CMS)、SC88-7222

  プログラミングの手引き、SC88-7218

  言語解説書、SC88-7219

  参照要約、SX88-7011

  診断の手引き、SC88-7223

  コンパイル時メッセージおよびコード、SC88-7224

- **DB2**

  アプリケーション・プログラミングおよび SQL ガイド、SC88-8763

- **CCCA**

  COBOL and CICS/VS Command Level Conversion Aid、SB11-6432

- **CICS/ESA**

  アプリケーション・プログラミング・ガイド、SC88-7689

  Application Programming Reference、SC33-1170

  カスタマイズ・ガイド、SC88-7686

  External CICS Interface、SC33-1390

  Sample Applications Guide、SC33-1173

- **COBOL 報告書作成プログラム**

  COBOL Report Writer Precompiler Programmer's Manual、SC26-4301

  COBOL Report Writer Precompiler Installation and Operation for MVS and CMS、SC26-4302

- **IMS**

  アプリケーション・プログラミング: EXEC DLI コマンド (CICS および IMS)、SC88-7554

用語集

この用語集に記載されている用語は、COBOL における意味に従って定義されています。これらの用語は、他の言語での意味と同じ場合もあれば、異なる場合もあります。

米国規格協会 (ANSI) の許可のもとに、次の資料から定義を転載しています。

- *American National Standard Programming Language COBOL, ANSI X3.23-1985* (Copyright 1985 American National Standards Institute, Inc.)。これは、技術委員会 X3J4 (米国標準規格 COBOL の改訂を行う委員会) によって準備されたものです。
- *American National Dictionary for Information Processing Systems* (Copyright 1982 by the Computer and Business Equipment Manufacturers Association)。

米国標準規格 (ANS) の定義の前にはアスタリスク (*) を付けています。

［ア行］

アクセス・モード (* access mode). ファイル内でレコードが操作されるときの方式。

遊びバイト (slack bytes). 一部の数値項目の位置合わせが正しく行われるように、データ項目相互間またはレコード相互間に挿入されるバイト。遊びバイトには意味のあるデータは含まれない。場合によっては、コンパイラーによって遊びバイトが挿入されるが、それ以外の場合、遊びバイトの挿入はプログラマーが行う。正しい位置合わせを行うために遊びバイトが必要なときは、SYNCHRONIZED 文節によって、コンパイラーに遊びバイトを挿入させる。レコード相互間の遊びバイトは、プログラマーが挿入する。

アプリケーション (application). 特定の目的を達成するために連携する 1 つ以上のルーチンの集合。

アンパック 10 進数フォーマット (unpacked decimal format). ビット 4 ～ 7 に数字が含まれ、右端のバイトのビット 0 ～ 3 に符号が含まれる、数を表現する形式。他のすべてのバイトのビット 0 ～ 3 には 1 (16 進数の F) が含まれる。たとえば、10 進値 +123 は 1111 0001 1111 0010 1111 0011 として表される。「ゾーン 10 進フォーマット (zoned decimal format)」と同義。

暗黙範囲終了符号 (* implicit scope terminator). 先行する終了していないステートメントの有効範囲を終了させる区切りピリオド、または先行する句に含まれているステートメントの有効範囲の終わりをその出現によって示すステートメントの句。

異常終了 (abend). プログラムの異常終了。

移植性 (portability). アプリケーション・プログラムを、あるアプリケーション・プラットフォームから別のアプリケーション・プラットフォームへ、ソース・プログラムをあまり変更せずに転送する能力。

インスタンス・データ (instance data). オブジェクトの状態を定義するデータ。クラスによって導入されるインスタンス・データは、クラス定義のデータ部の WORKING-STORAGE セクション内に定義される。オブジェクトの状態には、現行クラスによって継承されている基礎クラスが導入したインスタンス変数の状態も含まれる。オブジェクト・インスタンスごとにインスタンス・データの個々のコピーが作成される。

インターフェース (interface). クライアント がクラス を使用するために知っていなければならない情報 (クラスの属性 の名前およびクラスのメソッド のシグニチャー)。COBOL のような direct-to-SOM コンパイラーの場合、クラスへのインターフェースは、クラス定義用の固有言語構文によって定義することができる。他の言語でインプリメントさ

れたクラスの場合、クラスへのインターフェースが SOM Interface Definition Language (IDL) で直接定義されている場合がある。COBOL コンパイラーには、COBOL クラス用の IDL を自動的に生成するのためのコンパイラー・オプション IDLGEN がある。

インターフェース定義言語 (IDL) (Interface Definition Language (IDL)). オブジェクト のクラスへのインターフェース を IDL ファイル内に定義するための正式な言語 (プログラミング言語とは独立している)。SOM コンパイラーはこの IDL ファイルを解釈して、インプリメンテーション・テンプレート・ファイルおよびバインディング・ファイルを作成する。SOM の Interface Definition Language は、オブジェクト管理グループの共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (*CORBA*) によって確立された標準に完全に従っている。

インライン (inline). プログラムにおいて、ルーチン、サブルーチン、または他のプログラムに分岐することなく、順次に実行される命令。

埋め込み文字 (* padding character). 物理レコードの未使用文字位置を充てんするために使用される英数字。

英字 (* alphabetic character). 文字またはスペース文字。

英字名 (* alphabet-name). 環境部の SPECIAL-NAMES 段落の中で、特定の文字セットまたは照合シーケンス (あるいはその両方) に名前を割り当てるユーザー定義語。

英数字 (* alphanumeric character). コンピューターの文字セット内の任意の文字。

英数字関数 (* alphanumeric function). コンピューターの文字セットからの 1 つ以上の文字のストリングで構成される値を持つ関数。

英数字編集文字 (alphanumeric-edited character). 少なくとも 1 つの B、0 (ゼロ)、または / (スラッシュ) を含んでいる英数字ストリング内の文字。

エレメント (テキスト・エレメント) (element (text element)). テキスト・ストリングの 1 つの論理単位であり、単一のデータ項目または動詞の記述の前に、エレメント・タイプを識別する固有のコードが付いているもの。

エンクレーブ (enclave). 言語環境プログラムでは独立したルーチンの集合を指し、そのうちの 1 つがメインルーチンとして指定され、最初に呼び出される。エンクレーブは、プログラムまたは実行単位とほぼ同様である。実行可能プログラムである。

演算式 (* arithmetic expression). 数値基本項目の ID、数値リテラル、算術演算子によって区切られたそのような ID およびリテラル、算術演算子によって区切られた 2 つの演算式、または括弧で囲まれた演算式。

演算符号 (* operational sign). 数値データ項目または数値リテラルに関連した代数符号であり、その値が正であるか負であるかを示す。

オーバーフロー条件 (overflow condition). 演算の結果の一部が意図したストレージの容量を超える場合に発生する条件。

オープン・モード (* open mode). OPEN ステートメントを実行してから、REEL または UNIT 句を指定せずに CLOSE ステートメントを実行するまでの、ファイルの状態。特定のオープン・モードは、OPEN ステートメント内に、INPUT、OUTPUT、I-O、または EXTEND のいずれかとして指定される。

オブジェクト (object). 状態 (そのデータ値) および演算 (そのメソッド) を持つエンティティー。オブジェクトは、状態と動作をカプセル化する方法。

オブジェクト時 (* object time). オブジェクト・プログラムが実行されるとき。この用語は「実行時 (execution time)」と同義。

オブジェクト・コード (object code). コンパイラーまたはアセンブラーからの出力であり、それ自体が実行可能なマシン・コードであるか、または実行可能なマシン・コードを作成する処理に適している。

オブジェクト・コンピューター記入項目 (* object computer entry). 環境部の OBJECT-COMPUTER 段落の中の記入項目であり、オブジェクト・プログラムが実行されるコンピューター環境を記述する文節を含んでいる。

オブジェクト・デック (object deck). リンケージ・エディターへの入力として適切なオブジェクト・プログラムの部分。「オブジェクト・モジュール *(object module)*」および「テキスト・デック *(text deck)*」と同義。

オブジェクト・プログラム (* object program). データと相互作用して問題解決を行うように設計された、実行可能なマシン言語命令および他のデータの集合あるいはグループ。このコンテキストでは、オブジェクト・プログラムは一般に、COBOL コンパイラーがソース・プログラムに操作した結果のマシン言語である。あいまいになる危険性がない場合、「オブジェクト・プログラム」という句の代わりに「プログラム」というワードだけを使用できる。

オブジェクト・モジュール (object module). アセンブラーまたはコンパイラーによって作成され、リンケージ・エディターやバインダーへの入力として使用される 1 つ以上の制御セクション (CSECT) の集合。「テキスト・デック (text deck)」または「オブジェクト・デック (object deck)」と同義。

オプショナル・ファイル (* optional file). オブジェクト・プログラムの実行ごとに、必ずしも存在する必要がないものとして宣言されるファイル。オブジェクト・プログラムでは、ファイルが存在しているかどうかについて疑問が生じる。

オプショナル・ワード (* optional word). 言語を読みやすくすることだけを目的として特定の形式に含まれている予約語であり、その予約語が現れる形式がソース・プログラムで使用されるときに、ユーザーがそれを指定するか否かは自由である。

オペランド (* operand). 本書の目的では、オペランドの一般的な定義は「操作を行う対象のコンポーネント」であるが、ステートメントまたは記入項目に現れる小文字のワード (1 つ以上) は、オペランドが示すデータへの暗黙参照のように、オペランドであると見なすことができる。

[カ行]

カーネル (kernel). 入出力、管理、および通信などのタスク用のプログラムを含むコンポーネントの一部。

外部 10 進数項目 (external decimal item). ビット 4 〜 7 に数字が含まれ、右端のバイトのビット 0 〜 3 に符号が含まれる、数を表現する形式。他のすべてのバイトのビット 0 〜 3 には 1 (16 進数の F) が含まれる。たとえば、10 進値 +123 は 1111 0001 1111 0010 1111 0011 として表される。(「ゾーン 10 進数項目 (zoned decimal item)」としても知られる。)

外部スイッチ (* external switch). インプリメントする人によって定義され、名前を付けられたハードウェアまたはソフトウェア装置であり、2 つの代替状態のいずれかが存在していることを示す。

外部データ (* external data). エンクレーブの存続時間中存続し、エンクレーブ内のルーチンが再入されるたびに最後に使用された値を保守するデータ。単一のロード・モジュールから成るエンクレーブ内では、静的ストレージ期間、A FORTRAN 共通ブロック、および COBOL EXTERNAL データを持つ任意の C データ・オブジェクトと同義である。

外部データ項目 (* external data item). 実行単位の 1 つ以上のプログラムにおいて外部レコードの一部として記述されるデータ項目であり、その項目が記述されているプログラムからその項目自体を参照できるもの。

外部データ・レコード (* external data record). 実行単位の 1 つ以上のプログラムにおいて記述される論理レコードであり、そのデータ項目は、それらが記述されているプログラムから参照できる。

外部ファイル結合子 (* external file connector). 実行単位において 1 つ以上のオブジェクト・プログラムにアクセスできるファイル結合子。

外部浮動小数点データ項目 (external floating-point item). 実数を一対の数表示で表す、数を表記するための形式。浮動小数点表記では、実数は、固定小数点部分 (最初の数表示) と、暗黙的な浮動小数点の底を指数 (2 番目の数表示) で累乗することで得られる値との積である。

たとえば、0.0001234 の浮動小数点表記は 0.1234 -3 である。この場合、0.1234 が小数部であり、-3 が指数である。

外部プログラム (external program). 最外部プログラム。ネストされていないプログラム。

カウンター (* counter). 数値または数表現を保管するために使用されるデータ項目であり、その保管は、これらの数値が別の数値によって増加または減少させられたり、ゼロまたは任意の正の値または負の値に変更またはリセットされたりする方法で行われる。

拡張 (extensions). ANSI 規格で記述されるもの以外で、IBM コンパイラーがサポートする特定の COBOL 構文とセマンティクス。

拡張モード (* extend mode). ファイルに関して EXTEND 句を指定し、OPEN ステートメントを実行したあと、そのファイルに関して REEL 句または UNIT 句を指定せずに CLOSE ステートメントを実行するまでの、ファイルの状態。

カタログ式プロシージャー (cataloged procedure). プロシージャー・ライブラリー (SYS1.PROCLIB) と呼ばれる区分データ・セットに入っている 1 組のジョブ制御ステートメント。カタログ式プロシージャーを使用すると、JCL のコーディングに要する時間を節減でき、エラーを削減できる。

可変位置グループ (* variably located group). 同じレベル 01 レコード内の可変長テーブルに続く (そのテーブルに従属しない) グループ項目。

可変位置項目 (* variably located item). 同じレベル 01 レコードの可変長テーブルに続く (そのテーブルに従属しない) データ項目。

可変出現データ項目 (* variable occurrence data item). 可変出現データ項目とは、繰り返される回数が可変であるテーブル・エレメント。そのような項目は、その項目のデータ記述記入項目に OCCURS DEPENDING ON 文節を含んでいなければならないか、またはその種の別の項目に従属しなければならない。

可変長レコード (* variable length record). レコードが可変数の文字位置を含むことを許可するファイル記述記入項目またはソート・マージ記述記入項目を持つファイルに関連したレコード。

環境部 (ENVIRONMENT DIVISION). COBOL プログラム、クラス定義、またはメソッド定義の 4 つの主コンポーネントの 1 つ。環境部では、ソース・プログラムがコンパイルされるコンピューター、およびオブジェクト・プログラムが実行されるコンピューターを記述し、ファイルおよびそれらのレコードの論理概念と、ファイルが保管される装置の物理的局面とのリンケージを提供する。

環境文節 (* environment clause). 環境部の記入項目の一部として現れる文節。

環境変数 (environment variable). オペレーティング・システムを実行に移す方法、およびオペレーティング・システムに装置を認識させる方法を記述した、一連の変数。

環境名 (environment-name). IBM が指定する名前であり、システム論理装置、プリンターおよびカード穿孔装置制御文字、報告書コード、またはプログラム・スイッチ、あるいはそれらの組み合わせを識別する。環境部で環境名が簡略名と関連がある場合、置換が有効な任意の形式においてその簡略名で置き換えることができる。

関係 (* relation). 「関係演算子 (relational operator)」または「比較条件 (relation condition)」を参照。

関係演算子 (* relational operator). 比較条件の構造で使用される、予約語、比較文字、連続する予約語のグループ、または連続する予約語と比較文字のグループ。使用できる演算子とそれらの意味は次のとおり。

演算子　意味
IS GREATER THAN
　　より大きい
IS >　より大きい
IS NOT GREATER THAN
　　より大きくない

IS NOT >
より大きくない

IS LESS THAN
より小さい
IS < より小さい
IS NOT LESS THAN
より小さくない
IS NOT <
より小さくない

IS EQUAL TO
等しい
IS = 等しい
IS NOT EQUAL TO
等しくない
IS NOT =
等しくない

IS GREATER THAN OR EQUAL TO
より大きいか等しい
IS >= より大きいか等しい

IS LESS THAN OR EQUAL TO
より小さいか等しい
IS <= より小さいか等しい

関数 (* function). 式において名前をコーディングすることにより呼び出されるルーチン。ルーチンはルーチン名を使用して呼び出し側に結果を戻す。

関数 ID (* function-identifier). 関数を参照する、文字ストリングと区切り文字の構文的に正しい組み合わせ。関数で表現されるデータ項目は、関数名と引き数 (ある場合) によって一意的に識別される。関数 ID には参照変更子を含めることができる。英数字関数を参照する関数 ID は、ID を指定することのできる一般形式で、任意の場所に指定できる (ただし、特定の制約がある)。整数関数または数字関数を参照する関数 ID は、演算式を指定することのできる一般形式で、任意の場所で参照できる。

関数名 (function-name). 必要な引き数を指定した呼び出しによって関数の値が決定されるメカニズムに名前を割り当てるワード。

簡略複合比較条件 (* abbreviated combined relation condition). 連続する一連の比較条件において、共通サブジェクトの明示的な省略、または共通サブジェクトと共通関係演算子の明示的な省略によって生じる複合条件。

簡略名 (* mnemonic-name). 環境部において、指定されたインプリメントする人の名前に関連したユーザー定義語。

キー (* key). レコードの位置を識別するデータ項目、またはデータの順序付けを識別するための一連のデータ項目。

キーワード (* key word). 予約語または関数名であり、そのキーワードが現れる形式がソース・プログラムで使用されるときに必要である。

記号文字 (* symbolic-character). ユーザー定義の形象定数を指定するユーザー定義語。

疑似テキスト (* pseudo-text). ソース・プログラムまたは COBOL ライブラリーにおいて、疑似テキスト区切り文字によって区切られた一連のテキスト・ワード、コメント行、または区切り文字スペース (疑似テキスト区切り文字を含まない)。

疑似テキスト区切り文字 (* pseudo-text delimiter). 疑似テキストを区切るために使用される 2 つの連続する等号文字 (==)。

記入項目 (* entry). 一連の連続する記述の文節であり、区切りピリオドで終了し、COBOL プログラムの見出し部、環境部、またはデータ部に書かれる。

記入項目のオブジェクト (* object of entry). COBOL プログラムのデータ部の記入項目の中の一連のオペランドと予約語であり、その記入項目のサブジェクトの直後に続く。

記入項目のサブジェクト (* subject of entry). データ部の記入項目のレベル標識またはレベル番号の直後に現れるオペランドまたは予約語。

基本項目 (* elementary item). それ以上論理的に分割されないものとして記述されるデータ項目。

基本レコード・キー (* prime record key). その内容が索引付きファイル内のレコードを一意的に識別するキー。

共通プログラム (* common program). 別のプログラムに直接含まれているが、任意のプログラムから直接呼び出すことができたり、別のプログラムに間接的に含めることができるプログラム。

切り替え状況条件 (switch-status condition). 「オン」または「オフ」状況に設定できる UPSI スイッチが特定の状況に設定されているかという命題。この命題に対して真理値を判別できる。

記録モード (recording mode). ファイル内の論理レコードの形式。記録モードには、F (固定長)、V (可変長)、S (スパン)、および U (未定義) がある。

句 (* phrase). 句とは、COBOL プロシージャー・ステートメントの一部または COBOL 文節の一部を形成する、1 つ以上の連続する COBOL 文字ストリングの順序付けられたセット。

区切りコンマ (* separator comma). 文字ストリングを区切るために使用される 1 つのコンマ (,) とその後の 1 つのスペース。

区切りセミコロン (* separator semicolon). 文字ストリングを区切るために使用される 1 つのセミコロン (;) とその後の 1 つのスペース。

区切りピリオド (* separator period). 文字ストリングを区切るために使用される 1 つのピリオド (.) とその後の 1 つのスペース。

区切り文字 (* delimiter). 1 つの文字、または一連の連続する文字であり、文字ストリングの終わりを識別し、その文字ストリングを後続の文字ストリングから区切る。区切り文字は、それが区切る文字ストリングの一部ではない。

区切り文字 (* separator). 文字ストリングを区切るために使用される 1 つの文字または 2 つの連続する文字。

句読文字 (* punctuation character). 以下のセットに属する文字。

文字	意味
,	コンマ
;	セミコロン
:	コロン
.	ピリオド (終止符)
"	引用符
(	左括弧
)	右括弧
b	スペース
=	等号

組み込み関数 (built-in function). 「組み込み関数 (intrinsic function)」を参照。

組み込み関数 (intrinsic function). 組み込み関数参照によって呼び出される、事前定義された関数 (共通に使用される算術関数など)。

クラス (* class). ゼロ、1 つ、または複数のオブジェクトの共通の動作およびインプリメンテーションを定義するエンティティー。同じインプリメンテーションを共用する複数のオブジェクトは、同じクラスのオブジェクトと見なされる。

クラス識別記入項目 (* class identification entry). 見出し部の CLASS-ID 段落の中の記入項目であり、この記入項目には、クラス名を指定し、選択された属性をクラス定義に割り当てる文節が含まれる。

クラス終了ヘッダー (* end class header). ワードの組み合わせに区切りピリオドが続いたもので、COBOL クラス定義の終わりを示す。クラス終了ヘッダーは次のとおり。

```
END CLASS class-name.
```

クラス条件 (* class condition). 項目の内容がすべて英字であるか、すべて数字であるか、あるいはクラス名の定義にリストされている文字だけで構成されているかという命題。この命題に対して真理値を判別できる。

クラス定義 (* Class Definition). クラスを定義する COBOL ソース単位。

クラス名 (* class-name). 環境部の SPECIAL-NAMES 段落で定義されたユーザー定義語であり、データ項目の内容がクラス名の定義にリストされている文字だけで構成されるかどうかという命題 (この命題に対して真理値を定義できる) に名前を割り当てる。

クラス・オブジェクト (class object). SOM クラスを表すランタイム・オブジェクト。

グループ項目 (* group item). 従属データ項目で構成されるデータ項目。

グローバル名 (* global name). 1 つのプログラムにおいてのみ宣言されるが、そのプログラムおよびそのプログラムに含まれている任意のプログラムから参照できる名前。条件名、データ名、ファイル名、レコード名、報告書名、および一部の特殊レジスターをグローバル名にすることができる。

ケース構造 (case structure). いくつかのアクションの中から選択を行うために一連の条件をテストするプログラム処理ロジック。

継承 (クラスの場合) (* inheritance (for classes)). 1 つ以上のクラス のインプリメンテーションを別のクラスの基本として使用するメカニズム。サブクラス は、1 つ以上のスーパークラス から継承する。定義により、継承するクラスは、継承されるクラスに準拠する。

形象定数 (* figurative constant). 特定の予約語を使用して参照されるコンパイラー生成の値。

桁 (* column). 印刷行における文字位置。桁は、印刷行の左端の文字位置から印刷行の右端の位置まで、1 から番号が付けられ、1 ずつ増加する。

桁位置 (* digit position). 1 つの桁を保管するために必要な物理ストレージの大きさ。この大きさは、データ項目を定義するデータ記述記入項目に指定された用途によって異なる。

結果 ID (* resultant identifier). ユーザー定義のデータ項目であり、算術演算の結果が入る。

現行ボリューム・ポインター (* current volume pointer). 順次ファイルの現行のボリュームを指す概念的エンティティー。

現行レコード (* current record). ファイル処理では、ファイルに関連したレコード域で使用できるレコード。

言語環境プログラム (Language Environment). z/OS 言語環境プログラムの省略名。言語環境プログラム準拠のコンパイラーによってコンパイルされた C、C++、COBOL、FORTRAN、PL/I、VisualAge PL/I および Java アプリケーションに共通のランタイム環境およびランタイム・サービスを提供する一連の構造体およびインターフェース。

言語環境プログラム準拠 (Language Environment-conforming). 言語環境プログラムの共通インターフェース規則に準拠していること。

言語間通信 (ILC) (interlanguage communication (ILC)). 異なるプログラム言語で書かれた複数のルーチンが通信できること。ILC サポートにより、アプリケーション作成者は、各種の言語で書かれたコンポーネント・ルーチンからアプリケーションを容易に構築することができる。

言語名 (* language-name). 特定のプログラム言語を指定するシステム名。

コード・ページ (code page). 図形文字および制御機能の意味をすべてのコード・ポイントに割り当てたもの。たとえば、8 ビット・コードの場合は 256 のコード・ポイントに文字と意味を割り当てたもので、7 ビット・コードの場合は 128 のコード・ポイントに文字と意味を割り当てたもの。

高位桁 (* high order end). 文字ストリングの左端の文字。

降順キー (* descending key). データ項目を比較するための規則に従って、キーの最も高い値から最も低い値にデータを順に並べるために使用するキー。

構成セクション (* CONFIGURATION SECTION). 環境部のセクションであり、ソース・プログラムとオブジェクト・プログラムおよびクラス定義の全体的な仕様を記述する。

構造化プログラミング (structured programming). コンピューター・プログラムを編成してコーディングするための技法であり、この技法では、プログラムはセグメントの階層で構成され、それぞれのセグメントには 1 つの入り口点と 1 つの出口点がある。制御は構造の下方に渡され、階層のより上位のレベルに無条件ブランチしない。

構文 (syntax). プログラミング言語の構造、およびプログラミング言語におけるステートメントの構築を支配する規則。

固定小数点数 (fixed-point number). オプショナルの符号の位置、含んでいる桁の数、およびオプショナルの小数点の位置を指定する PICTURE 文節で定義された数値データ項目そのフォーマットは 2 進数、パック 10 進数、または外部 10 進数。

固定長レコード (* fixed length record). すべてのレコードが同じ数の文字位置を含んでいる必要があることがファイル記述記入項目またはソート・マージ記述記入項目で記述されたファイルに関連したレコード。

固定ファイル属性 (* fixed file attributes). ファイルに関する情報であり、ファイルの作成時に確立され、それ以降はファイルが存在する限り変更できない。これらの属性には、ファイルの編成 (順次、相対、または索引付き)、基本レコード・キー、代替レコード・キー、コード・セット、最小および最大レコード・サイズ、レコード・タイプ (固定または可変)、索引ファイルのキーの照合シーケンス、ブロック化因数、埋め込み文字、およびレコード区切り文字がある。

コピーブック (copybook). コンパイル時に COPY ステートメントによってソース・プログラムに組み込まれる一連のコードを含んでいるファイルまたはライブラリー・メンバー。このファイルはユーザーが作成するか、COBOL が提供するか、または別のプロダクトが提供することができる。

コメント記入項目 (* comment-entry). 見出し部の記入項目であり、コンピューターの文字セットからの文字の任意の組み合わせ。

コメント行 (* comment line). 行の標識区域ではアスタリスク(*)、およびその行の区域 A および B ではコンピューターの文字セットからの任意の文字で表されるソース・プログラム行。コメント行は、プログラムの文書化にのみ役立つ。行の標識区域では斜線 (/)、およびその行の区域 A および B ではコンピューターの文字セットからの任意の文字で表される特殊形式のコメント行があると、コメントの印刷前にページ替えが行われる。

固有照合シーケンス (* native collating sequence). OBJECT-COMPUTER 段落で指定されたコンピューターに関連した、インプリメントする人が定義した照合シーケンス。

固有文字セット (* native character set). OBJECT-COMPUTER 段落で指定されたコンピューターに関連した、インプリメントする人が定義した文字セット。

コンパイラー (compiler). 高水準言語で書かれたプログラムをマシン言語オブジェクト・プログラムに変換するプログラム。

コンパイラー指示ステートメント (compiler directing statement). コンパイラー指示動詞で始まるステートメント (または指示) で、これによって、コンパイラーはコンパイル時に特定のアクションを取る。コンパイラー指示は COBOL ソース・プログラムに含まれている。コンパイラー指示は COBOL ソース・プログラムに含まれている。したがって、複数のコンパイラー指示ステートメントを使用することにより、ソース・プログラム内において異なる指示のサブオプションを指定できる。

コンパイル (* compile). (1) 高水準言語で表現されたプログラムを、中間言語、アセンブリー言語、またはコンピューター言語で表現されたプログラムに変換すること。(2) 別のプログラミング言語で書かれたコンピューター・プログラムからマシン言語プログラムを作成すること。これを行うためには、アセンブラーの機能を実行するほかに、プログラムの全体的な論理構造を利用するか、記号ステートメントごとに複数のコンピューター命令を生成するか、あるいはその両方を行う。

コンパイル時 (* compile time). COBOL ソース・プログラムが COBOL コンパイラーによって COBOL オブジェクト・プログラムに変換されるとき。

コンパイル時間オプション (compiler options). コンパイルのある局面を制御するために指定できるキーワード。コンパイラー・オプションによってコンパイラーが生成するロード・モジュールの性質、作成される印刷出力のタイプ、コンパイラーの効果的使用、およびエラー・メッセージの宛先を制御できる。「コンパイル時間オプション (compiler-time options)」も参照。

コンパイル時間オプション (compiler-time options). コンパイルのある局面を制御するために指定できるキーワード。コンパイラー・オプションによってコンパイラーが生成するロード・モジュールの性質、作成される印刷出力のタイプ、コンパイラーの効果的使用、およびエラー・メッセージの宛先を制御できる。

コンパイル用コンピューター記入項目 (* source computer entry). 環境部の SOURCE-COMPUTER 段落の記入項目であり、ソース・プログラムがコンパイルされるコンピューター環境を記述する文節が入る。

コンピューター名 (* computer-name). プログラムがコンパイルまたは実行されるコンピューターを識別するシステム名。

[サ行]

再帰 (recursion). それ自身を呼び出すプログラム、またはその呼び出し先プログラムのいずれかによって直接または間接に呼び出されるプログラム。

再帰可能 (recursively capable). RECURSIVE 属性を PROGRAM-ID ステートメントに指定してあれば、プログラムは再帰可能である。

最後に使われた状態 (last-used state). プログラムが、最後に使われた状態にあるのは、プログラムの内部値がプログラム終了時と同じままである (初期値にリセットされていない) 場合である。

再使用可能環境 (reusable environment). 再使用可能環境とは、ILBOSTP0 プログラム、IGZERRE プログラム、または RTEREUS ランタイム・オプションのいずれかを使用してアセンブラー・プログラムをメインプログラムとして確立する場合のこと。

再入可能 (reentrant). 複数のユーザーがロード・モジュールの単一コピーを共用することを可能にする、プログラムまたはルーチンの属性。

索引 (* index). その内容がテーブルの特定エレメントの識別を表す、コンピューターのストレージ域またはレジスター。

索引付きデータ名 (indexed data-name). データ名とそれに続く (括弧で囲まれた) 1 つ以上の索引名で構成される ID。

索引付きファイル (* indexed file). 索引編成のファイル。

索引付け (indexing). 索引名を使用しての添え字付けと同義。

索引データ項目 (* index data item). 索引名と関連した値を、インプリメントする人によって指定された形式で保管できるデータ項目。

索引編成 (* indexed organization). 各レコードが、そのレコード内の 1 つ以上のキーの値で識別される、永続論理ファイル構造。

索引名 (* index-name). 特定のテーブルに関連した索引に名前を割り当てるユーザー定義語。

サブクラス (* subclass). 別のクラスから継承するクラス。継承関係において 2 つのクラスが一緒に認識されるとき、サブクラスは継承側クラスまたは継承するクラスであり、スーパークラス は被継承クラスまたは継承されるクラスである。

サブプログラム (* subprogram). 「呼び出し先プログラム (called program)」を参照。

算術演算 (* arithmetic operation). 算術ステートメントの実行によって生じるプロセス、または演算式の計算であり、与えられた引き数に対する数学的に正しい解を結果として出す。

算術演算子 (* arithmetic operator). 以下のセットに属する 1 つの文字、または固定の 2 文字の組み合わせ。

文字	意味
+	加算
-	減算
*	乗算
/	除算
**	指数

算術ステートメント (* arithmetic statement). 算術演算を実行させるステートメント。算術ステートメントには、ADD、COMPUTE、DIVIDE、MULTIPLY、および SUBTRACT ステートメントがある。

参照キー (* key of reference). 索引付きファイル内のレコードにアクセスするために現在使用されているキー (基本キーまたは代替キー)。

参照形式 (* reference format). COBOL ソース・プログラムを記述するための標準方式を提供する形式。

参照変更 (reference modification). 別の英数字データ項目の左端の文字および左端の文字からの相対的な長さを指定することによって、新しい英数字データ項目を定義する方法。

参照変更子 (* reference-modifier). 固有のデータ項目を定義する、文字ストリングと区切り文字の構文的に正しい組み合わせ。これには、区切り用の左括弧区切り文字、左端文字位置、コロン区切り文字、長さ (オプション)、および区切り用の右括弧区切り文字が含まれる。

シーケンス構造 (sequence structure). 一連のステートメントが順次に実行されるプログラム処理ロジック。

式 (* expression). 演算式または条件式。

指数 (exponent). 別の数 (底) を乗ずる指数を示す数。正の指数は乗算を示し、負の指数は除算を示し、小数の指数は数量の根を示す。COBOL では、指数式は記号 ‘**’ の後に指数を付けて表す。

システム名 (* system-name). 操作環境と通信するために使用される COBOL ワード。

システム・オブジェクト・モデル (SOM) (System Object Model (SOM)). クラス・ライブラリーを作成し、パッケージし、取り扱うための、IBM のオブジェクト指向プログラミング・テクノロジー。SOM は、オブジェクト管理グループ (OMG) の共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA) 標準に準拠する。

実行時 (execution time). ランタイムの同義語。

実行時環境 (execution-time environment). 「ランタイム環境 (run-time environment)」を参照。

実行単位 (* run unit). ともに実行される 1 つ以上のオブジェクト・プログラム。言語環境プログラムでは、実行単位はエンクレーブと同義である。

実際の小数点 (* actual decimal point). 10 進小数点文字のピリオド (.) またはコンマ (,) を使用して、データ項目内の小数点の位置を物理的に表現したもの。

修飾子 (* qualifier).

1. 修飾子に従属する項目の名前である別のデータ名と共に、あるいは条件名と共に参照で使用される、レベル標識に関連したデータ名または名前。
2. そのセクションで指定された段落名と共に参照で使用されるセクション名。
3. そのライブラリーに関関連したテキスト名と共に参照で使用されるライブラリー名。

修飾データ名 (* qualified data-name). データ名と、1 組または複数組の連結語 OF または IN とデータ名修飾子から構成される ID。

出力ファイル (* output file). OUTPUT モードまたは EXTEND モードのいずれかでオープンされるファイル。

出力プロシージャー (* output procedure). SORT ステートメントの実行中に、ソート機能の完了後に制御を与えられる一連のステートメント。または、MERGE ステートメントの実行中に、マージ機能が、要求された時点でマージされた順序の次のレコードを選択できるポイントに到達した後に、制御を与えられる一連のステートメント。

出力モード (* output mode). OUTPUT 句または EXTEND 句を指定して OPEN ステートメントを実行してから、REEL 句または UNIT 句を指定せずに CLOSE ステートメントを実行するまでの、ファイルの状態。

順次アクセス (* sequential access). ファイル内のレコードの順序によって決定される、連続する前後関係の論理レコード順序付けに従って、論理レコードがファイルから取得されたり、論理レコードをファイルに入れたりするアクセス・モード。

順次ファイル (* sequential file). 順次編成のファイル。

順次編成 (* sequential organization). レコードがファイルに入れられたときに確立されたレコードの前後関係によってレコードが識別される、永続論理ファイル構造。

条件 (condition). 言語環境プログラムによって使用可能にされたか、または認識された例外であり、したがって、ユーザーおよび言語の条件ハンドラーを活動化するのに適格である例外。アプリケーションの通常のプログラミングされたフローを変えるもの。条件は、ハードウェアまたはオペレーティング・システムによって検出され、その結果、割り込みが起こる。このほかにも、条件は言語固有の生成コードまたは言語ライブラリー・コードによっても検出される。

条件 (* condition). 真理値を判別できる、実行時のプログラムの状況。「条件」(condition-1、condition-2、...) という用語が、一般形式の「条件」(condition-1、condition-2、...) に関連して、またはその言語仕様で使われている場合は、それは単純条件 (括弧で囲まれている場合もある) か、または構文的に正しい単純条件、論理演算子、括弧の組み合わせから成る複合条件で構成される条件式で、この式に関する真理値を判別できる。

条件式 (* conditional expression). EVALUATE、IF、PERFORM、または SEARCH のステートメントで指定される、単純条件または複合条件。(「単純条件 (simple condition)」および「複合条件 (complex condition)」も参照。)

条件ステートメント (* conditional statement). 条件の真理値が判別されること、およびオブジェクト・プログラムの以降のアクションがその真理値に基づいて行われることを指定するステートメント。

条件付き句 (* conditional phrase). 条件付き句は、条件ステートメントの実行から生じる条件の真理値が判別されたときに取られるアクションを指定する。

条件変数 (* conditional variable). 1 つ以上の値に 1 つの条件名が割り当てられているデータ項目。

条件名 (* condition-name). 条件変数がとることのできる値のサブセットに名前を割り当てるユーザー定義語。またはインプリメントする人が定義したスイッチまたは装置の状況に割り当てられたユーザー定義語。「条件名」が一般形式で使用されるとき、それは、「条件名」を修飾子と添え字と一緒に (参照の固有性の要件に応じて) 構文的に正しく組み合わせたものから構成される、固有のデータ項目参照を表す。

条件名条件 (* condition-name condition). 条件変数の値が、その条件変数と関連した条件名に属する一連の値のメンバーであるかという命題。この命題に対して真理値を判別できる。

照合シーケンス (* collating sequence). ソート、マージ、比較を行うために、また索引ファイルを順次処理するために、コンピューターに受け入れられる文字が順序付けられているシーケンス。

昇順キー (* ascending key). データ項目を比較するための規則に従って、キーの最も低い値から最も高い値にデータを順に並べるために使用するキー。

初期状態 (* initial state). 実行単位内で最初に呼び出されたときのプログラムの状態。

初期プログラム (* initial program). 実行単位内で呼び出されるたびに初期状態に置かれるプログラム。

シリアル検索 (serial search). ある集合のメンバーが、最初のメンバーから最後のメンバーまで連続して調べられる検索。

真理値 (* truth value). 2 つの値 (真または偽) のいずれかによる、条件の評価の結果の表現。

スーパークラス (* superclass). 別のクラスによって継承されるクラス。「サブクラス (*subclass*)」も参照。

数字 (digit). 0 ～ 9 までの任意の数字。COBOL では、この用語は別の記号に関しては使用されない。

数字 (* numeric character). 次の一連の数字に属する文字。0, 1, 2, 3, 4, 5, 6, 7, 8, 9。

数字関数 (* numeric function). クラスとカテゴリーは数字であるが、ある種の計算においては、整数関数の要件が満たされない関数。

数字編集項目 (numeric-edited item). 印刷出力で使用できる形式の数字項目。外部 10 進数字の 0 ～ 9 までの数字、小数点、コンマ、ドル記号、編集記号制御文字、その他の編集記号から構成される。

数値項目 (* numeric item). 記述により、その内容が、‘0’ ～ ‘9’ までの数字から選択された文字で表される値に制限されるデータ項目。符号付きの場合は、その項目には ‘+’、‘-’、または他の演算符号の表記を入れることもできる。

数値リテラル (* numeric literal). 小数点または代数符号 (あるいはその両方) を含むことができる、1 つ以上の数字で構成されるリテラル。小数点は右端の文字であってはならない。代数符号 (存在する場合) は左端の文字でなければならない。

ステートメント (* statement). COBOL ソース・プログラムに書かれる、動詞で始まる、ワード、リテラル、および区切り文字の構文的に有効な組み合わせ。

世紀ウィンドウ (century window). 100 年の間隔であり、言語環境プログラムはこの間隔の中にすべての 2 桁の年が存在すると想定する。言語環境プログラムのデフォルトの世紀ウィンドウは、システム日付より 80 年前から始まる。

整数 (* integer). (1) 小数点の右側に桁位置がない数値リテラル。

(2) データ部に定義される数値データ項目であり、小数点の右側に桁位置がないもの。

(3) 関数の計算で戻される値の、小数点の右側の桁がすべてゼロであることが定義されている数字関数。

整数関数 (integer function). カテゴリーが数値であり、小数点の右側の桁位置が定義に入っていない関数。

セクション (* section). ゼロ、1 つ、または複数の段落またはエンティティー (セクション本体と呼ばれる) と、その最初のものの前にセクション・ヘッダーが付いているもの。各セクションは、セクション・ヘッダーと関連セクション本体で構成される。

セクション名 (* section-name). 手続き部のセクションに名前を割り当てるユーザー定義語。

セクション・ヘッダー (* section header). 環境部、データ部、および手続き部のセクションの先頭を示す、ワードの組み合わせとその後の区切りピリオド。環境部およびデータ部では、セクション・ヘッダーは予約語とその後の区切りピリオドで構成される。環境部で使用できるセクション・ヘッダーは次のとおり。

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

データ部で使用できるセクション・ヘッダーは次のとおり。

```
FILE SECTION.
WORKING-STORAGE SECTION.
LOCAL-STORAGE SECTION.
LINKAGE SECTION.
```

手続き部では、セクション・ヘッダーは、セクション名と予約語 SECTION と区切りピリオドで構成される。

宣言部分 (* declaratives). 手続き部の先頭に書かれた一連の、1 つ以上の特殊目的セクションであり、その先頭にはキーワード DECLARATIVES が付き、その最後にはキーワード END DECLARATIVES が続く。宣言部分は次の順序で構成される。セクション・ヘッダー、USE コンパイラー指示文、ゼロ、1 つ、または複数の関連する段落。

宣言文 (* declarative sentence). 1 つの USE ステートメントから構成され、区切りピリオドで終了するコンパイラー指示ステートメント。

選択構造 (selection structure). 条件が真であるか偽であるかによって、一連のステートメントまたは別の一連のステートメントが実行されるプログラム処理ロジック。

ソース項目 (* source item). SOURCE 文節によって指定される ID で、印刷可能項目の値を指定する。

ソース・プログラム (source program). ソース・プログラムを他の形式と記号で表現できることが認められているが、本書では必ず、構文的に正しい一連の COBOL ステートメントを指す。COBOL ソース・プログラムは、見出し部または COPY ステートメントから始まる。COBOL ソース・プログラムは、プログラム終了ヘッダーで終わる (指定されている場合) か、またはソース・プログラム行がそれ以上ないことで終わる。ソース・プログラムにはプログラミング言語で書かれた一連の命令が含まれており、プログラムを実行する前にこれをマシン言語に変換する必要がある。

ソート・ファイル (* sort file). SORT ステートメントによってソートされるレコードの集合。ソート・ファイルは、ソート機能によって作成され、ソート機能によってのみ使用できる。

ソート・マージ・ファイル記述記入項目 (* sort-merge file description entry) . データ部のファイル・セクションの記入項目であり、レベル標識 SD、ファイル名、および (必要に応じて) 1 組のファイル文節で構成される。

ゾーン 10 進数項目 (zoned decimal item). 「外部 10 進数項目 (external decimal item)」を参照。

ゾーン 10 進フォーマット (zoned decimal format). 「アンパック 10 進数フォーマット (unpacked decimal format)」と同義。

相互参照リスト (cross-reference listing). コンパイラー・リストの部分であり、プログラム内でファイル、フィールド、および標識を定義、参照、および変更している場所についての情報が入っている。

相対キー (* relative key). その内容が相対ファイル内の論理レコードを識別するキー。

相対ファイル (* relative file). 相対編成のファイル。

相対編成 (* relative organization). 各レコードが、ファイル内でのレコードの論理的な順序位置を指定する整数値 (ゼロより大きい) によって一意的に識別される永続論理ファイル構造。

相対レコード番号 (* relative record number). 相対編成ファイル内でのレコードの序数。この数は、整数である数値リテラルとして扱われる。

想定小数点 (* assumed decimal point). データ項目の中に実際には小数点のための文字が入っていない小数点位置。想定小数点は、論理的な意味を持ち、物理的には表現されない。

添え字 (* subscript). 整数、データ名 (およびその後に任意指定として続く演算子 + または - のある整数)、または索引名 (およびその後に任意指定として続く演算子 + または - のある整数) のいずれかで表現される出現番号であり、テーブルの特定エレメントを識別する。添え字付き ID を、可変数の引き数が許可される関数の関数引き数として使用する場合は、添え字をワード ALL にすることができる。

添え字付きデータ名 (* subscripted data-name). データ名とその後の括弧で囲まれた 1 つ以上の添え字で構成される ID 。

[タ行]

代替レコード・キー (* alternate record key). 基本レコード・キー以外のキーであり、その内容が索引付きファイル内のレコードを識別する。

ダイナミック・リンク・ライブラリー (dynamic link library). リンク時ではなく、ロード時または実行時にプログラムに結合される実行コードおよびデータが入っているファイル。ダイナミック・リンク・ライブラリー内のコードおよびデータは、複数のアプリケーションが同時に共用することができる。

大容量記憶 (* mass storage). データが順次または非順次に編成されて保守されるストレージ・メディア。

大容量記憶装置 (* mass storage device). 大規模の記憶容量を備えた装置。たとえば、磁気ディスク、磁気ドラムなど。

大容量記憶ファイル (* mass storage file). 大容量ストレージ・メディアに割り当てられたレコードの集合。

単項演算子 (* unary operator). 正 (+) または負 (-) の符号であり、演算式の変数または左括弧の前に付けられ、式にそれぞれ +1 または -1 を乗算する働きをする。

単純条件 (* simple condition). 以下の集合から選択された単一の条件。

比較条件
クラス条件
条件名条件
切り替え状況条件
符号条件

単純否定条件 (* negated simple condition). ‘NOT’ 論理演算子と、その直後に続く単純条件。

段落 (* paragraph). 手続き部では、段落名とその後の区切りピリオド、およびゼロ、1 つ、または複数の文。見出し部および環境部では、段落ヘッダーとその後のゼロ、1 つ、または複数の記入項目。

段落ヘッダー (* paragraph header). 予約語とその後の区切りピリオドであり、見出し部および環境部の段落の先頭を示す。見出し部で許される段落ヘッダーは次のとおり。

```
PROGRAM-ID. (Program IDENTIFICATION DIVISION)
CLASS-ID. (Class IDENTIFICATION DIVISION)
METHOD-ID. (Method IDENTIFICATION DIVISION)
AUTHOR.
```

```
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
```

環境部で許される段落ヘッダーは次のとおり。

```
SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
REPOSITORY. (Program or Class CONFIGURATION SECTION)
FILE-CONTROL.
I-O-CONTROL.
```

段落名 (* paragraph-name). 手続き部の段落を識別し、開始するユーザー定義語。

チェックポイント (checkpoint). ジョブ・ステップをあとで再始動できるように、ジョブおよびシステムの状況に関する情報が記録されるポイント。

中間結果 (intermediate result). 一連の算術演算の結果が入っている中間フィールド。

直接アクセス (* direct access). プロセスが、以前にアクセスされたデータへの参照ではなく、そのデータの位置にのみ依存する方法で、ストレージ・デバイスからデータを入手したり、ストレージ・デバイスにデータを入力したりする機能。

通貨記号 (currency sign). COBOL 文字セットの文字 '$'、または CURRENCY コンパイラー・オプションで定義されるその文字。NOCURRENCY コンパイラー・オプションが有効な場合、通貨記号は文字 '$' として定義される。

通貨記号 (currency symbol). CURRENCY コンパイラー・オプションによって、または SPECIAL-NAMES 段落の CURRENCY SIGN 文節によって定義される文字。NOCURRENCY コンパイラー・オプションが COBOL ソース・プログラムに対して有効で、CURRENCY SIGN 文節がソース・プログラムに存在**しない**場合、通貨記号 (currency symbol) と通貨記号 (currency sign) は同じである。

次の実行可能ステートメント (* next executable statement). 現行ステートメントの実行が完了した後、次に制御が移されるステートメント。

次の実行可能文 (* next executable sentence). 現行ステートメントの実行が完了した後、次に制御が移される文。

次のレコード (* next record). ファイルの現行レコードに論理的に続くレコード。

データ記述記入項目 (* data description entry). COBOL プログラムのデータ部の記入項目であり、レベル番号の後に (必要な場合) データ名が続き、必要に応じてその後に 1 組のデータ文節が続いている。

データ項目 (* data item). COBOL プログラムによって、または関数評価の規則によって定義されるデータ単位 (リテラルを除く)。

データ部 (DATA DIVISION). COBOL では、プログラムで使用されるファイル、およびそのファイルに含まれるレコードを記述するプログラムの部分。また、必要な WORKING-STORAGE データ項目、LINKAGE SECTION データ項目、および LOCAL-STORAGE データ項目も記述する。

データ文節 (* data clause). COBOL プログラムのデータ部のデータ記述記入項目に現れる文節であり、データ項目の特定の属性を記述する。

データ名 (* data-name). データ記述記入項目で記述されたデータ項目に名前を割り当てるユーザー定義語。一般形式で使用されるときは、「データ名」は、形式の規則で特に許される場合を除き、参照変更、添え字付け、または修飾を行ってはならないワードを表す。

テーブル (* table). データ部で OCCURS 文節によって定義される 1 組の論理的に連続するデータ項目。

テーブル・エレメント (* table element). テーブルを構成する 1 組の反復項目に属するデータ項目。

低位桁 (* low order end). 文字ストリングの右端の文字。

停止点 (breakpoint). 通常はコマンドまたは条件によって指定されるプログラム内の場所。この場所で実行が中断され、制御がワークステーション・ユーザーまたは指定されたデバッグ・プログラムに渡される場合がある。

テキスト名 (* text-name). ライブラリー・テキストを識別するユーザー定義語。

テキスト・デック (text deck). 「*オブジェクト・デック (object deck)*」または「*オブジェクト・モジュール (object module)*」と同義。

テキスト・ワード (* text word). COBOL ライブラリー、ソース・プログラムのマージン A とマージン R の間、または疑似テキストの中の、1 つの文字または一連の連続する文字。

- 区切り文字。ただし、次のものは除く。スペース、疑似テキスト区切り文字、および非数値リテラルの左区切り文字と右区切り文字。ライブラリー、ソース・プログラム、または疑似テキスト内のコンテキストに関係なく、右括弧文字と左括弧文字は常にテキスト・ワードと見なされる。
- リテラル (非数値リテラルの場合は、そのリテラルを囲む左引用符と右引用符を含む)。
- 他の任意の一連の連続する COBOL 文字 (コメント行、および区切り文字で囲まれたワード 'COPY' を除く) で、区切り文字でもリテラルでもないもの。

手続き部 (Procedure Division). COBOL プログラム、クラス定義、またはメソッド定義の 4 つの主コンポーネントの 1 つ。手続き部には、問題を解決するための命令を入れる。プログラムおよびメソッドの手続き部には、命令ステートメント、条件ステートメント、コンパイラー指示ステートメント、段落、プロシージャー、およびセクションを入れることができる。クラスの手続き部には、メソッド定義だけを入れる。

手続き部の終わり (* end of Procedure Division). COBOL ソース・プログラムの物理的な位置であり、その後にプロシージャーは現れない。

デバッグ行 (* debugging line). デバッグ行とは、行の標識区域に 'D' がある行のこと。

デバッグ・セクション (* debugging section). USE FOR DEBUGGING ステートメントが入っているセクション。

動詞 (* verb). COBOL コンパイラーまたはオブジェクト・プログラムによって行われるアクションを表すワード。

動的アクセス (* dynamic access). 特定のレコードを大容量記憶ファイルとの間で非順次方式でやり取りでき、特定のレコードを同じ OPEN ステートメントの範囲内で順次方式でファイルから入手できる、アクセス・モード。

動的ストレージ域 (DSA) (Dynamic Storage Area (DSA)). 動的に獲得されるストレージであり、レジスター保管域、および動的ストレージ割り振りに使用可能な区域 (プログラム変数など) から構成される。DSA は一般に、言語環境プログラムが管理する STACK セグメント内に割り振られる。

特殊名記入項目 (* special names entry). 環境部の SPECIAL-NAMES 段落内の記入項目であり、次のことを行う手段となる。通貨記号を指定する、小数点を選択する、記号文字を指定する、インプリメントする人の名前をユーザー指定の簡略名に関係付ける、英字名を文字セットまたは照合シーケンスに関係付ける、およびクラス名を一連の文字に関係付ける。

特殊文字 (* special character). 以下のセットに属する文字。

文字	意味
+	正符号
-	負符号 (ハイフン)
*****	アスタリスク
/	斜線 (スラッシュ)
=	等号
$	通貨記号
,	コンマ (小数点)

; セミコロン
. ピリオド (小数点、終止符)
" 引用符
(左括弧
) 右括弧
> より大記号
< より小記号
: コロン

特殊文字ワード (* special-character word). 算術演算子または比較文字である予約語。

特殊レジスター (* special registers). 特定のコンパイラー生成ストレージ域であり、その基本的な用途は、特定の COBOL 機能を併用して作成した情報を保管すること。

独立項目 (* noncontiguous items). 別のデータ項目への階層関係がない、WORKING-STORAGE および LINKAGE SECTION の基本データ項目。

トップダウン開発 (top-down development). 「構造化プログラミング (structured programming)」を参照。

トップダウン設計 (top-down design). 階層構造を使用するコンピューター・プログラムの設計。この設計では、階層構造の各レベルで関連機能が実行される。

トレーラー・ラベル (trailer-label). (1) 記録メディアの装置上のデータ・レコードに続くファイル・ラベルまたはデータ・セット・ラベル。(2)「ファイル終わりラベル (end-of-file label)」と同義。

［ナ行］

内部 10 進項目 (internal decimal item). フィールドの各バイト (右端バイトを除く) が 2 桁の数字を表すフォーマット。右端バイトには 1 桁の数字と符号が入る。たとえば、10 進値 +123 は 0001 0010 0011 1111 として表される。(「パック 10 進数 (packed decimal)」としても知られている。)

内部データ (* internal data). プログラムに記述されるデータであり、外部データ項目および外部ファイル結合子はすべて除く。プログラムの LINKAGE SECTION に記述された項目は、内部データとして扱われる。

内部データ項目 (* internal data item). 実行単位内の 1 つのプログラムに記述されているデータ項目。内部データ項目にグローバル名を付けることができる。

内部ファイル結合子 (* internal file connector). 実行単位内の 1 つのオブジェクト・プログラムでのみアクセスできるファイル結合子。

名前 (name). 最大 30 文字で構成されるワードであり、COBOL オペランドを定義する。

二分探索 (binary search). 探索の各ステップにおいて、1 組のデータ・エレメントを 2 つに分ける二分探索。奇数の場合は、なんらかの適切なアクションが取られる。

入出力状況 (* I-O-status). 入出力操作の結果生じる状況を示す 2 文字の値を含んでいる概念上のエンティティー。この値は、ファイル用のファイル制御記入項目で FILE STATUS 文節を使用することによって、プログラムで使用できるようになる。

入出力ステートメント (* Input-Output statement). 個々のレコードに、または 1 単位としてのファイルに操作を実行することにより、ファイルが処理されるようにするステートメント。入出力ステートメントには、ACCEPT (ID 句を指定する)、CLOSE、DELETE、DISPLAY、OPEN、READ、REWRITE、SET (TO ON または TO OFF 句を指定する)、START 、および WRITE がある。

入出力セクション (* INPUT-OUTPUT SECTION). 環境部のセクションであり、オブジェクト・プログラムまたはメソッドに必要なファイルおよび外部メディアに名前を割り当て、オブジェクト・プログラムまたはメソッド定義の実行中にデータを伝送および処理するのに必要な情報を提供する。

入出力ファイル (* input-output file). 入出力モードでオープンされるファイル。

入出力モード (* I-O-Mode). I-O 句を指定して OPEN ステートメントを実行してから、REEL または UNIT 句を指定せずに CLOSE ステートメントを実行するまでの、ファイルの状態。

入力ファイル (* input file). INPUT モードでオープンされるファイル。

入力プロシージャー (* input procedure). SORT ステートメントの実行中に、ソート対象に指定されたレコードの RELEASE を制御するために制御を与えられる一連のステートメント。

入力モード (* input mode). INPUT 句を指定して OPEN ステートメントを実行してから、REEL または UNIT 句を指定せずに CLOSE ステートメントを実行するまでの、ファイルの状態。

ヌル (null). 空。意味を持たないこと。

ネストされたプログラム (nested program). COBOL では、別のプログラム内に直接含まれているプログラム。

[ハ行]

バイト (byte). ストレージのアドレス可能度の基本単位。8 ビットの長さを持つ。

バイナリー項目 (binary item). 2 進表記 (基数 2 の表記体系) で表される数値データ項目。バイナリー項目には、0 ～ 9 までの 10 進数字で構成される 10 進数と同等の数と演算符号がある。項目の左端ビットは演算符号。

配列 (array). 言語環境プログラムでは、データ・オブジェクトで構成される集合であり、それぞれのデータ・オブジェクトは添え字付けによって一意的に参照できる。COBOL テーブルにほぼ類似している。

パスワード (password). プログラム、コンピューター・オペレーター、またはユーザーがデータにアクセスする前にセキュリティー要件を満たすために指定しなければならない固有の文字ストリング。

パック 10 進数項目 (packed decimal item). 「内部 10 進項目 (internal decimal item)」を参照。

バッファー (buffer). データが読み取られたり、書き込まれたりするストレージ域。通常、バッファーは一時記憶域の場合のみ使用される。

パラメーター (parameter). ルーチンによって受け取られるデータ項目。FORTRAN で使用される仮引き数という用語に対して他の言語で使用される用語。

範囲区切りステートメント (* delimited scope statement). 明示範囲終了符号を含むステートメント。

範囲終了符号 (scope terminator). 手続き部の特定のステートメントの終わりのマークを付ける COBOL 予約語。明示的なもの (たとえば、END-ADD) も暗黙のもの (区切りピリオド) もある。ステートメントの終わりの変数。

反復構造 (iteration structure). 条件が真である間、または条件が真になるまで、一連のステートメントが繰り返されるプログラム処理ロジック。

汎用オブジェクト参照子 (universal object reference). 任意のクラスのオブジェクトを参照できるデータ名。

比較条件 (* relation condition). 演算式、データ項目、非数値リテラル、または指標名の値が、別の演算式、データ項目、非数値リテラル、または指標名の値と特定の関係を持つかどうかという命題。この命題に対して真理値を判別できる。(「関係演算子 (relational operator)」も参照。)

比較文字 (* relation character). 以下のセットに属する文字。

文字	意味
>	より大きい
<	より小さい
=	等しい

引き数 (* argument). (1) 呼び出し点で、呼び出し先ルーチンに渡されるデータ項目または集合体を指定するために使用される式。(2) 呼び出し点で呼び出し先ルーチンに渡されるデータ、または呼び出し先ルーチンによって受け取られるデータ。

非数字項目 (* nonnumeric item). 記述によって、その内容を、コンピューターの文字セットからの文字の任意の組み合わせで構成することができるデータ項目。特定のカテゴリーの非数字項目は、さらに制限された文字セットから形成することができる。

非数値リテラル (* nonnumeric literal). 引用符で囲まれたリテラル。文字のストリングには、コンピューターの文字セットからの任意の文字を入れることができる。

ビッグ・エンディアン (big-endian). メインフレームおよび AIX ワークステーションがバイナリー・データを保管するときに使用するデフォルト形式。この形式では、最小重み数字が最高位アドレスにある。「リトル・エンディアン (little-endian)」と対比。

標準データ・フォーマット (* standard data format). COBOL データ部でデータの特性を記述するために使用される概念。この概念のもとでは、データの特性は、データが内部的にコンピューターに、または特定の外部メディアに保管される方法に適した形式ではなく、印刷ページ上での無限の長さと幅を持つデータ外観に適した形式で表現される。

部 (* division). 部の本体と呼ばれる、ゼロ、1 つ、または複数のセクションまたは段落の集合であり、これらのセクションまたは段落は、特定の規則に従って形成および結合されたものである。それぞれの部は、部のヘッダーおよび関連した部の本体で構成される。COBOL プログラムには、見出し部、環境部、データ部、および手続き部の 4 つの部がある。

ファイル (* file). 割り当てられた名前によって保管および検索される、関連データ・レコードの集合に名前を付けたもの。MVS データ・セットと同義。

ファイル位置標識 (* file position indicator). 概念上のエンティティーであり、索引付きファイルの参照キー内の現行キーの値、または順次ファイルの現行レコードのレコード番号、または相対ファイルの現行レコードの相対レコード番号が入っているか、あるいは次の論理レコードが存在しないことを示すか、オプションの入力ファイルが存在しないことを示すか、AT END 条件がすでに存在していることを示すか、もしくは有効な次のレコードが設定されていないことを示す。

ファイル記述記入項目 (* file description entry). レベル標識 FD、ファイル名、(必要に応じて) 1 組のファイル文節から構成される、データ部のファイル・セクション内の記入項目。

ファイル結合子 (* file connector). ファイルに関する情報が入っているストレージ域であり、ファイル名と物理ファイルの間のリンケージとして、およびファイル名とその関連レコード域の間のリンケージとして使用される。

ファイル制御 (File-Control). ソース・プログラムで用いられるデータ・ファイルが宣言されている環境部の段落の名前。

ファイル制御記入項目 (* file control entry). ファイルの関連物理属性を宣言する、SELECT 文節およびそのすべての従属文節。

ファイル制御ブロック (FCB) (file control block). I/O ルーチンのアドレス、それらがどのようにオープンおよびクローズされたかに関する情報、およびファイル情報ブロック (FIB) へのポインターを含むブロック。

ファイル属性対立条件 (* file attribute conflict condition). ファイルに入出力操作を実行しようとして失敗した。プログラム内でそのファイルに関して指定されたファイル属性が、そのファイルの固定属性と一致しない。

ファイル文節 (* file clause). データ部の記入項目であるファイル記述記入項目 (FD 記入項目) およびソート・マージ・ファイル記述記入項目 (SD 記入項目) のいずれかの一部として現れる文節。

ファイル編成 (* file organization). ファイルの作成時に確立される永続論理ファイル構造。

ファイル名 (* file-name). データ部のファイル・セクション内のファイル記述記入項目またはソート・マージ・ファイル記述記入項目で記述されたファイル結合子に名前を割り当てるユーザー定義語。

ファイル・システム (file system). ファイルとその属性の集合。ファイル・システムは、それらのファイルを参照するファイル・シリアル番号用のネーム・スペースを提供する。

ファイル・セクション (* File Section). データ部のセクションであり、ファイル記述記入項目およびソート・マージ・ファイル記述記入項目がそれらの関連レコード記述と一緒に入っているもの。

フォーマット (* format). 一連のデータの特定の配置。

複合条件 (* combined condition). 2 つ以上の条件を AND または OR 論理演算子で結合した結果生じる条件。

複合条件 (* complex condition). 1 つ以上の論理演算子が 1 つ以上の条件に基づいて作動する条件。(「単純否定条件 (negated simple condition)」、「複合条件 (combined condition)」、および「複合否定条件 (negated combined condition)」も参照。)

複合否定条件 (* negated combined condition). ‘NOT’ 論理演算子と、その直後に続く括弧で囲まれた複合条件。

符号条件 (* sign condition). データ項目または演算式の代数値がゼロより小さいか、大きいか、または等しいかという命題。この命題に対して真理値を判別できる。

不成功の実行 (* unsuccessful execution). ステートメントの実行が試みられたが、そのステートメントに指定された操作すべてを実行できなかったこと。ステートメントの不成功の実行は、そのステートメントで参照されたデータには影響を与えないが、状況表示に影響を与える場合がある。

物理レコード (* physical record). 「ブロック (block)」を参照。

浮動小数点数 (floating-point number). 小数部と指数を含んでいる数値データ項目。その値は、小数部と、数値データ項目の底を指数で累乗することで得られる値との積である。

部のヘッダー (* division header). ワードとその後に続く、部の先頭を示す区切りピリオドの組み合わせ。部のヘッダーは次のとおり。

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
```

プリロード済み (preloaded). COBOL ではこの用語は、COBOL プログラムが呼び出されるたびにロードされるのではなく、IMS のもとでストレージに常駐していることを言う。

古くなったエレメント (* obsolete element). 標準 COBOL の次の改訂からは削除される、標準 COBOL 内の COBOL 言語エレメント。

プログラム識別記入項目 (* program identification entry). 見出し部の PROGRAM-ID 段落の記入項目であり、プログラム名を指定し、選択されたプログラム属性をプログラムに割り当てる文節が入る。

プログラム終了ヘッダー (* end program header). ワードの組み合わせに区切りピリオドが続いたもので、COBOL ソース・プログラムの終わりを示す。プログラム終了ヘッダーは次のとおり。

```
END PROGRAM program-name.
```

プログラム名 (* program-name). 見出し部およびプログラム終了ヘッダーにおいて、COBOL ソース・プログラムを識別するユーザー定義語。

プロシージャー (* procedure). COBOL では、プロシージャーとはプログラム内部からのみ実行可能な段落またはセクションを指す。PL/I では、通常は呼び出しによって外部から呼び出すことができる名前付きコード・ブロック。

プロシージャー統合 (procedure integration). COBOL 最適化プログラムの機能の 1 つであり、実行されるプロシージャーまたは含まれているプログラムへの呼び出しを単純化する。

PERFORM のプロシージャー統合とは、PERFORM ステートメントが、実行されるプロシージャーで置き換えられるプロセスのこと。含まれているプログラムのプロシージャー統合とは、含まれているプログラムへの CALL がプログラム・コードで置き換えられるプロセスのこと。

プロシージャー名 (* procedure-name). 手続き部の段落またはセクションを指名するために使用されるユーザー定義語。段落名 (修飾される場合もある) またはセクション名で構成される。

プロシージャー・ブランチ・ステートメント (* procedure branching statement). ステートメントがソース・プログラムに書かれている順序で次の実行可能なステートメントではないステートメントに制御を明示的に移させるステートメント。プロシージャー・ブランチ・ステートメントには、ALTER、CALL、EXIT、EXIT PROGRAM、GO TO、MERGE (OUTPUT PROCEDURE 句を指定する)、PERFORM、および SORT (INPUT PROCEDURE 句または OUTPUT PROCEDURE 句を指定する) がある。

プロシージャー・ポインター・データ項目 (procedure-pointer data item). 入り口点を指すポインターを保管できるデータ項目。USAGE IS PROCEDURE-POINTER 文節で定義されたデータ項目に、プロシージャー入り口点のアドレスが入る。

ブロック (* block). 通常は 1 つ以上の論理レコードで構成される物理的データ単位。大容量記憶ファイルの場合、ブロックには論理レコードの一部が含まれることもある。ブロックのサイズは、そのブロックが含まれているファイルのサイズと直接関係はなく、そのブロックに含まれているか、そのブロックにオーバーラップしている論理レコードのサイズとも直接関係はない。この用語は物理レコードと同義。

文 (* sentence). 1 つ以上のステートメントのシーケンスであり、最後のステートメントは区切りピリオドで終了する。

文節 (* clause). 記入項目の属性を指定することを目的とする、1 組の連続する COBOL 文字ストリング。

ページ (page). 出力データの物理的な分割を示す出力データの垂直分割であり、分割は内部論理要件または出力メディアの外部特性 (あるいはその両方) に基づく。

ページ本体 (* page body). 行の書き込みまたはスペース埋込み (あるいはその両方) を行うことができる論理ページの部分。

ヘッダー・ラベル (header label). (1) 記録メディア装置上のデータ・レコードの前に付いているファイル・ラベルまたはデータ・セット・ラベル。(2) ファイル開始ラベルの同義語。

別々にコンパイルされるプログラム (* separately compiled program). 含まれているプログラムとは一緒だが、その他のすべてのプログラムとは別々にコンパイルされるプログラム。

編集解除 (* de-edit). 項目の編集解除された数値を判別するために、数字編集されたデータ項目からすべての編集文字を論理的に除去すること。

編集済みデータ項目 (edited data item). ゼロの抑止または編集文字の挿入 (あるいはその両方) を行うことによって変更されたデータ項目。

編集文字 (* editing character). 次のセットに属する 1 つの文字または固定された 2 文字の組み合わせ。

文字	意味
b	スペース

0 ゼロ
+ 正符号
- 負符号
CR 貸方
DB 借方
Z ゼロ抑止
***** 不渡り手形
$ 通貨記号
, コンマ (小数点)
. ピリオド (小数点)
/ 斜線 (スラッシュ)

変数 (* variable). オブジェクト・プログラムを実行することで値を変更できるデータ項目。演算式で使用される変数は、数値基本項目でなければならない。

ポインター・データ項目 (pointer data item). アドレス値を保管できるデータ項目。USAGE IS POINTER 文節を使用すれば、データ項目はポインターとして明示的に定義される。ADDRESS OF 特殊レジスターは、ポインター・データ項目として暗黙的に定義される。ポインター・データ項目は等しいかどうか比較でき、別のポインター・データ項目に移動することもできる。

ボリューム (volume). データのある一定の部分で、データ・キャリアとともにユニットとして簡便に処理できる。ユニットとして取り付けおよび取り外しできるデータ・キャリアには、磁気テープのリール、ディスク・パックなどがある。

ボリューム切り替え処理手順 (volume switch procedures). ファイル終わりに達する前にユニットまたはリールの終わりに達したときに自動的に実行されるシステム固有のプロシージャー。

［マ行］

マージ・ファイル (* merge file). MERGE ステートメントでマージされるレコードの集合。マージ・ファイルは、マージ機能によって作成され、マージ機能によってのみ使用できる。

マルチタスキング (multitasking). 2 つ以上のタスクの同時実行またはインターリーブド実行を可能にする動作モード。言語環境プログラム・プロダクトのもとで実行する場合、マルチタスキングは「*マルチスレッド化 (multithreading)*」と同義。

見出し部 (IDENTIFICATION DIVISION). COBOL プログラム、クラス定義、またはメソッド定義の 4 つの主コンポーネントの 1 つ。見出し部は、プログラム名、クラス名、またはメソッド名を識別する。見出し部には、作成者の名前、インストール、または日付の文書を入れることができる。

無効キー条件 (* invalid key condition). オブジェクト時に、索引付きファイルまたは相対ファイルに関連したキーの特定の値が無効であると判別されたときに発生する条件。

明示範囲終了符号 (* explicit scope terminator). 手続き部の特定のステートメントの有効範囲を終了させる予約語。

命令ステートメント (* imperative statement). 命令動詞で始まり無条件処置が取られるよう指定するステートメント、または明示範囲終了符号 (範囲区切りステートメント) で区切られた条件ステートメント。命令ステートメントは、一連の命令ステートメントで構成される。

メインプログラム (main program). エンクレーブにおいて呼び出し側から最初に制御を受け取るルーチン。FORTRAN では、メインプログラムには最初のステートメントとして FUNCTION、SUBROUTINE、または BLOCK DATA ステートメントが含まれない。最初のステートメントとしては PROGRAM ステートメントが含まれる可能性がある。サブプログラムと対比。

メソッド (method). オブジェクトによってサポートされる操作の 1 つを定義するプロシージャー・コードであり、そのオブジェクトへの INVOKE ステートメントによって実行される。

メソッド識別記入項目 (* method identification entry). 見出し部の METHOD-ID 段落の中の記入項目であり、メソッド名を指定する文節および選択された属性をメソッド定義に割り当てる文節を含んでいる。

メソッド終了ヘッダー (* end method header). ワードの組み合わせに区切りピリオドが続いたもので、COBOL メソッド定義の終わりを示す。メソッド終了ヘッダーは次のとおり。

```
END METHOD method-name.
```

メソッド定義 (* Method Definition). メソッドを定義する COBOL ソース単位。

メソッド名 (* method-name). メソッドを識別するユーザー定義語。

メタクラス (metaclass). そのインスタンスが SOM クラス・オブジェクトである SOM クラス。メタクラス内に定義されたメソッドは、そのクラスのオブジェクト・インスタンスが存在しなくても実行され、そのクラスのインスタンスを作成するために頻繁に使用される。

文字 (* character). データの編成、制御、または表現の一部として使用される文字、数字、またはその他の記号。文字はしばしば隣接または連続したストロークを空間的に配置した形式になる。

文字 (* letter). 以下の 2 つのセットのいずれかに属する文字。

1. 大文字: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
2. 小文字: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

文字位置 (character position). USAGE IS DISPLAY として記述される単一の標準データ・フォーマット文字を保管するために必要な物理ストレージの量。

文字ストリング (* character-string). COBOL ワード、リテラル、PICTURE 文字ストリング、またはコメント記入項目を形成する一連の隣接する文字。区切り文字で区切らなければならない。

文字セット (character set). プログラミング言語またはコンピューター・システム用のすべての有効な文字。

［ヤ行］

ユーザー定義語 (* user-defined word). 文節またはステートメントの形式を満たすためにユーザーが指定しなければならない COBOL ワード。

優先順位番号 (* priority-number). セグメント化の目的で、手続き部のセクションを分類するユーザー定義語。セグメント番号には、’0’,’1’, ... , ’9’ の文字しか使用できない。セグメント番号は 1 桁または 2 桁の数として表現される。

ユニット (unit). 直接アクセスのモジュールであり、その大きさは IBM によって決められている。

呼び出し可能サービス (callable services). 言語環境プログラムから定義済み呼び出しインターフェースを使用して呼び出すことができるサービスの集合で、言語環境プログラムの規則を順守するすべてのプログラムから使用できる。

呼び出し先プログラム (called program). CALL ステートメントのオブジェクトであるプログラム。

呼び出しプログラム (* calling program). 別のプログラムへの CALL を実行するプログラム。

予約語 (* reserved word). COBOL ソース・プログラムで使用できるが、プログラムにユーザー定義語またはシステム名として現れてはならないワードのリストに指定されている COBOL ワード。

［ラ行］

ライブラリー名 (* library-name). COBOL ライブラリーに名前を割り当てるユーザー定義語であり、所定のソース・プログラムをコンパイルするためにコンパイラーが使用する。

ライブラリー・テキスト (* library text). COBOL ライブラリー内の一連のテキスト・ワード、コメント行、区切りスペース、または区切りの疑似テキスト区切り文字。

ランタイム (* run time). オブジェクト・プログラムが実行されるとき。この用語は「オブジェクト時 (object time)」と同義。

ランタイム環境 (run-time environment). COBOL プログラムが実行される環境。

ランダム・アクセス (* random access). キー・データのプログラム指定の値が、相対ファイルまたは索引付きファイルから取得され、削除され、またはそれに入れられる論理レコードを識別するアクセス・モード。

リール (reel). ストレージ・メディアの離散的部分であり、その大きさはそれぞれのインプリメントする人によって決められており、1 つのファイルの一部、1 つのファイル全体、または任意の数のファイルを含む。この用語は「ユニット (unit)」または「ボリューム (volume)」と同義。

リソース (* resource). オペレーティング・システムによって制御され、実行中のプログラムで使用できる機能またはサービス。

リテラル (literal). 文字ストリングであり、その値は、そのストリングを構成する順序付けられた一連の文字により、または形象定数の使用により指定される。

リリアン日 (LILIAN DATE). グレゴリオ暦の開始以降の日数。第 1 日は 1582 年 10 月 15 日、金曜日。リリアン日形式は、グレゴリオ暦の作成者 Luigi Lilio を記念して命名された。

リンク・エディット (link-edit). リンケージ・エディターまたはバインダーを使用してロード可能なコンピューター・プログラムを作成すること。

リンケージ・セクション (LINKAGE SECTION). 呼び出し先プログラムのデータ部のセクションであり、呼び出しプログラムから使用できるデータ項目を記述する。これらのデータ項目は、呼び出しプログラムと呼び出し先プログラムの両方によって参照できる。

ルーチン (routine). コンピューターに操作または一連の関連操作を実行させる、COBOL プログラム内の一連のステートメント。言語環境プログラムでは、プロシージャー、機能、またはサブルーチンを指す。

ルーチン名 (* routine-name). COBOL 以外の言語で書かれたプロシージャーを識別するユーザー定義語。

レコード (* record). 「論理レコード(logical record)」を参照。

レコード域 (* record area). データ部のファイル・セクションのレコード記述記入項目に記述されたレコードを処理するために割り振られたストレージ域。ファイル・セクションでは、レコード域の現行の文字位置数は、明示的または暗黙の RECORD 文節で決定される。

レコード記述 (* record description). 「レコード記述記入項目 (record description entry)」を参照。

レコード記述記入項目 (* record description entry). 特定のレコードに関連したデータ記述記入項目全体。この用語は「レコード記述 (record description)」と同義。

レコード内データ構造 (* intra-record data structure). レコードを記述するデータ記述記入項目が複数個連続した 1 つのサブセットによって定義される、論理レコードのグループおよび基本データ項目の集合全体。これらのデータ記述記入項目には、レベル番号が、内部レコード・データ構造を記述する最初のデータ記述記入項目のレベル番号より大きいすべての記入項目が含まれる。

レコード番号 (* record number). 順次編成ファイル内でのレコードの序数。

レコード名 (* record-name). COBOL プログラムのデータ部のレコード記述項目で記述されたレコードに名前を割り当てるユーザー定義語。

レコード・キー (record key). その内容が索引付きファイル内のレコードを識別するキー。

レベル番号 (* level-number). 2 桁の数字として表されるユーザー定義語であり、データ項目の階層位置またはデータ記述記入項目の特殊な特性を示す。1 ～ 49 までの範囲のレベル番号は、論理レコードの階層構造におけるデータ項目の位置を示す。1 ～ 9 までの範囲のレベル番号は、1 桁の数字として書くことも、ゼロの後に有効数字を書くこともできる。レベル番号 66、77、および 88 は、データ記述記入項目の特殊な特性を識別する。

レベル標識 (* level indicator). 特定のタイプのファイルを識別するか、または階層での位置を識別する 2 つの英字。データ部でのレベル標識は、CD、FD、および SD。

連続項目 (* contiguous items). データ部の連続項目により記述され、相互の明確な階層的な関係を示す項目。

ローカル (local). プログラム実行環境の一連の属性であり、文化的に重要な考慮事項を示す。たとえば、文字コード・ページ、照合シーケンス、日時形式、通貨値表記、数値表記、または言語など。

ローカル・ストレージ・セクション (* LOCAL-STORAGE SECTION). データ部のセクションであり、VALUE 文節に割り当てられた値に応じて、呼び出しごとに割り振られ、解放されるストレージを定義する。

論理演算子 (* logical operator). 予約語 AND、OR、または NOT のいずれか。条件の形成において、AND または OR、あるいはその両方を論理連結語として使用できる。NOT は論理否定に使用できる。

論理レコード (* logical record). 最も包括的なデータ項目。レコードのレベル番号は 01。レコードは基本項目または項目グループのいずれかで構成される。この用語は「レコード (record)」と同義。

［ワ行］

ワード (* word). 30 文字までの文字ストリングであり、ユーザー定義語、システム名、予約語、または機能名を形成する。

割り当て名 (assignment-name). COBOL ファイルの編成を識別する名前であり、ファイルはこの名前によってシステムに認識される。

［数字］

1 バイト文字セット (SBCS) (Single Byte Character Set (SBCS)). 各文字が 1 バイトで表現される文字のセット。「EBCDIC (拡張 2 進化 10 進コード) (EBCDIC (Extended Binary-Coded Decimal Interchange Code))」も参照。

2 バイト文字セット (DBCS) (Double-Byte Character Set (DBCS)). 各文字が 2 バイトで表現される文字のセット。256 個のコード・ポイントで表現される記号より多くの記号を含んでいる言語 (日本語、中国語、および韓国語など) は、2 バイト文字セットを必要とする。各文字に 2 バイトが必要なため、DBCS 文字を入力、表示、および印刷するには、DBCS をサポートするハードウェアおよびソフトウェアが必要になる。

2000 年問題 (year 2000 problem). 2000 年問題とは、1960 年代と 1970 年代にストレージの節約のために使用された 2 桁の年の日付フィールドの制限を指す。たとえば、2 桁の年の日付フィールドを使用して、100 歳以上の人間の年齢を計算することはできない。さらに、1/1/2000 になっても、現在日付は前日の日付より進まない。非常に多くのアプリケーションおよびデータには、2 桁の年の日付しか備わっていないため、2000 年になる前にそれらのアプリケーションとデータをすべて変更して、障害を回避しなければならない。

77 レベル記述記入項目 (* 77-level-description-entry). レベル番号 77 の不連続データ項目を記述するデータ記述記入項目。

A

AMODE. リンケージ・エディターによって提供され、ロード・モジュールが入る必要のあるアドレッシング・モードを示すロード・モジュールの属性。

ANSI (米国規格協会) (ANSI (American National Standards Institute)). 米国で認定された組織が自発的業界標準を作成して維持するときの手順を確立する組織であり、製造業者、消費者、および一般の利害関係者で構成される。

ASCII. 情報交換用米国標準コード。7 ビット・コード化文字 (パリティー・チェックを含めて 8 ビット) から構成されるコード化文字セットを使用した、データ処理システム、データ通信システム、および関連装置の間での情報交換のために使用する標準コード。ASCII セットは、制御文字と図形文字から構成されている。

拡張: IBM では、ASCII コードへの拡張部分を定義している (文字 128 〜 255)。

AT END 条件 (* AT END condition). 次のような条件。

1. 順次アクセスされるファイルに READ ステートメントを実行しているときに、ファイルに次の論理レコードが存在しないか、または相対レコード番号の有効数字の桁数が相対キー・データ項目のサイズより大きいか、またはオプショナル入力ファイルが存在しない。
2. RETURN ステートメントの実行時に、関連するソート・ファイルまたはマージ・ファイルについて、次の論理レコードが存在しない。
3. SEARCH ステートメントの実行時に、関連する WHEN 段落のいずれかで指定された条件を満たさずに検索操作が終了した。

B

Btrieve. キー索引付きレコード管理システム。これにより、アプリケーションは、キー値、順次アクセス方式、またはランダム・アクセス方式によってレコードを管理することができる。Enterprise COBOL は、Btrieve によって、COBOL 順次および索引付きファイルの入出力言語をサポートする。

C

C 言語 (C language). ソフトウェア・アプリケーションを開発するために使用される高水準言語 (HLL)。大きな変更を加えなくてもさまざまな機種のコンピューター上で実行できる簡潔で効率的なコードでアプリケーションを開発することができる。

CEEDUMP. 言語環境プログラムのランタイム環境のダンプ、およびメンバー言語のライブラリー。ダンプのセクションは、ダンプ起動時に指定されたオプションに応じて指定選択的にインクルードされる。これはフル・アドレス・スペースのダンプではなく、言語環境プログラムとそのメンバーが制御するストレージおよび制御ブロックのダンプである。

CICS. 顧客情報管理システム (CICS)。

CICS トランスレーター (CICS translator). EXEC CICS コマンドを含むアプリケーションを入力として受け入れ、同等のアプリケーションを出力として生成するルーチンであり、各 CICS コマンドはソース言語に変換されている。

CMS (会話型モニター・システム) (CMS (Conversational Monitor System)). 汎用対話式機能、タイム・シェアリング機能、問題解決機能、およびプログラム開発機能を提供する仮想計算機オペレーティング・システム。VM/SP 制御プログラムの制御下でのみ稼働する。

COBOL 文字セット (* COBOL character set). 完全な COBOL 文字セットは、以下にリストする文字で構成される。

文字	意味
0,1...,9	数字

A,B,...,Z
英大文字
a,b,...,z 英小文字
b スペース
+ 正符号
- 負符号 (ハイフン)
***** アスタリスク
/ 斜線 (スラッシュ)
= 等号
$ 通貨記号
, コンマ (小数点)
; セミコロン
. ピリオド (小数点、終止符)
" 引用符
(左括弧
) 右括弧
> より大記号
< より小記号
: コロン

COBOL ワード (* COBOL word). 「ワード (word)」を参照。

CONSOLE. オペレーター・コンソールと関連した COBOL 環境名。

CORBA. オブジェクト管理グループによって確立された共通オブジェクト・リクエスト・ブローカー・アーキテクチャー。SOM クラスのインターフェース を記述するのに使用される IBM の *Interface Definition Language (IDL)* は CORBA 標準に完全に従っている。

C++ 言語 (C++ language). C 言語から発展したオブジェクト指向の高水準言語 (HLL)。C++ は、コードのモジュール性、移植性、再利用性などのオブジェクト指向テクノロジーの利点を活用している。

D

DBCS (2 バイト文字セット) (DBCS (Double-Byte Character Set)). 「2 バイト文字セット (DBCS) (Double-Byte Character Set (DBCS))」を参照。

DLL. 「ダイナミック・リンク・ライブラリー (dynamic link library)」を参照。

do 構造 (do construction). 構造化プログラミングにおいて、DO ステートメントを使用してプロシージャー内の一連のステートメントをグループ化すること。COBOL では、インライン PERFORM ステートメントが同じように機能する。

do-until. 構造化プログラミングにおいて、do-until ループは、少なくとも 1 回は実行され、所定の条件が真になるまで実行される。COBOL では、PERFORM ステートメントで使用される TEST AFTER 句が同じように機能する。

do-while. 構造化プログラミングにおいて、do-while ループは、所定の条件が真である場合、および真である間に実行される。COBOL では、PERFORM ステートメントで使用される TEST BEFORE 句が同じように機能する。

E

EBCDIC (拡張 2 進化 10 進コード) (* EBCDIC (ExtendedBinary-Coded Decimal Interchange Code)). 8 ビットのコード化文字で構成されるコード化文字セット。

EBCDIC 文字 (EBCDIC character). 8 ビット EBCDIC (拡張 2 進化 10 進コード) セットに入っているいずれかの記号。

H

HLL. 高水準言語 (High level language)。

I

IBM COBOL 拡張 (IBM COBOL extension). ANSI 規格で記述されるもの以外で、IBM コンパイラーがサポートする特定の COBOL 構文とセマンティクス。

ID (* identifier). データ項目に名前を割り当てる、文字ストリングと区切り文字の構文的に正しい組み合わせ。関数ではないデータ項目を参照するときは、ID は、データ名と、修飾子、添え字、または参照変更子 (一意的に参照するために必要な場合) から構成される。関数であるデータ項目を参照するときは、関数 ID が使用される。

IGZCBSN. COBOL/370 リリース 1 のブートストラップ・ルーチン。これは、COBOL/370 リリース 1 プログラムを含んでいるモジュールとリンク・エディットしなければならない。

IGZCBSO. COBOL (MVS および VM 版) リリース 2 および COBOL (OS/390 および VM 版) のブートストラップ・ルーチン。これは、COBOL (MVS および VM 版) リリース 2 または COBOL (OS/390 および VM 版) プログラムを含んでいるモジュールとリンク・エディットしなければならない。

IMS. 情報管理システム (Information Management System)。IBM のライセンス製品。IMS は、階層データベース、データ通信 (DC)、変換処理、およびデータベースのバックアウトとリカバリーをサポートする。

*** I-O-CONTROL.** 環境部の段落の名前であり、この段落では、再実行開始点に対するオブジェクト・プログラム要件、複数のデータ・ファイルによる同じ区域の共用、および単一の入出力装置上の複数ファイル・ストレージが指定される。

I-O-CONTROL 記入項目 (* I-O-CONTROL entry). 環境部の I-O-CONTROL 段落の記入項目であり、プログラムの実行時に指定されたファイルのデータを伝送および処理するのに必要な情報を提供する文節を含んでいる。

K

K. 記憶容量を表すときの 2 の 10 乗。10 進表記では 1024。

K バイト(KB) (kilobyte (KB)). 1K バイトは 1024 バイト。

L

*** LINAGE-COUNTER.** 特殊レジスターであり、その値はページ本体内での現在位置を指す。

M

M バイト (M) (* megabyte (M)). 1M バイトは 1,048,576 バイト。

MVS. 多重仮想記憶 (Multiple Virtual Storage) オペレーティング・システム。

O

*** OBJECT-COMPUTER.** 環境部の段落の名前であり、この段落では、オブジェクト・プログラムが実行されるコンピューター環境が記述される。

ODBC. ユーザーが各種のデータベースおよびファイル・システムからのデータにアクセスできるようにする Open Database Connectivity。

ODO オブジェクト (ODO object). 以下に例を示す。

```
WORKING-STORAGE SECTION
01  TABLE-1.
    05  X                   PICS9.
    05  Y OCCURS 3 TIMES
          DEPENDING ON X    PIC X.
```

この場合、X が OCCURS DEPENDING ON 文節のオブジェクト (ODO オブジェクト) である。ODO オブジェクトの値によって、テーブル内に現れる ODO サブジェクトの数が決まる。

ODO サブジェクト (ODO subject). 上記の例では、Y が OCCURS DEPENDING ON 文節のサブジェクト (ODO サブジェクト) である。テーブル内に現れる Y ODO サブジェクトの数は、X の値によって異なる。

OS/2 (オペレーティング・システム /2) (OS/2 (Operating System/2*)). IBM パーソナル・コンピューター・ファミリーのマルチタスキング・オペレーティング・システムであり、これを使用すれば、DOS モードのプログラムと OS/2 モードのプログラムの両方を実行できる。

Q

QSAM (待機順次アクセス方式) (QSAM (Queued Sequential Access Method)). 基本順次アクセス方式 (BSAM) の拡張版。この方式を使用する場合、キューは、処理を待っている入力データ・ブロック、または処理済みで補助記憶装置または出力装置への転送を待っている出力データ・ブロックで形成される。

S

SBCS (1 バイト文字セット) (SBCS (Single Byte Character Set)). 「1 バイト文字セット (SBCS)」を参照。

SOM. 「システム・オブジェクト・モデル (System Object Model)」を参照。

*** SOURCE-COMPUTER.** 環境部の段落の名前であり、この段落で、ソース・プログラムがコンパイルされるコンピューター環境が記述される。

SPECIAL-NAMES. 環境部の段落の名前であり、この段落で、環境名がユーザー指定の簡略名と関係付けられる。

STL. STL ファイル・システム: COBOL および PL/I 用の固有ワークステーションおよび PC ファイル・システム。順次ファイル、相対ファイル、および索引付きファイルをサポートする。サポート対象には、完全版 ANSI 85 COBOL 標準入出力言語、および *COBOL 言語解説書* に記述されている拡張機能のすべてが含まれる。ただし、例外が明示的に言及されている場合を除く。

U

UPSI スイッチ (UPSI switch). ハードウェア・スイッチの機能を実行するプログラム・スイッチ。UPSI-0 ～ UPSI-7 まで、8 つが提供される。

V

VM/SP (仮想計算機 / システム・プロダクト) (VM/SP (Virtual Machine/SystemProduct)). IBM ライセンス・プログラムであり、複数のコンピューター・システムが存在するかのように、単一のコンピューターのリソースを管理する。それぞれの仮想計算機は、「実」計算機と機能的に同等である。

W

*** WORKING-STORAGE SECTION.** データ部のセクションであり、独立項目または作業用ストレージ・レコード (あるいはその両方) で構成される作業用ストレージ・データ項目を記述する。

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］

［数字］

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

［特殊文字］

Printed in Japan

GC88-9118-00

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