Enterprise COBOL for z/OS

移行ガイド

バージョン 5.2

GC43-0801-03

Enterprise COBOL for z/OS

移行ガイド

バージョン 5.2

GC43-0801-03

お願い

本書および本書で紹介する製品をご使用になる前に、 335 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM Enterprise COBOL for z/OS バージョン 5 リリース 2 (プログラム番号 5655-W32) および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。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。

ソフトコピー資料は www.ibm.com/shop/publications/order/ で、無料で表示またはダウンロードできます。

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

原典： GC14-7383-03
Enterprise COBOL for z/OS
Migration Guide
Version 5.2

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

表

前書き

本書について

本書には、IBM® Enterprise COBOL バージョン 5.1 および 5.2 に移行するときに役立つ情報が記載されています。

本書では、Language Environment® へのランタイムの移行が完了していることを前提としています。

用語の説明

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

- IBM Enterprise COBOL for z/OS® および OS/390® 版バージョン 3 リリース 1
- IBM Enterprise COBOL for z/OS および OS/390 版バージョン 3 リリース 2
- IBM Enterprise COBOL for z/OS バージョン 3 リリース 3
- IBM Enterprise COBOL for z/OS バージョン 3 リリース 4
- IBM Enterprise COBOL for z/OS バージョン 4 リリース 1
- IBM Enterprise COBOL for z/OS バージョン 4 リリース 2
- IBM Enterprise COBOL for z/OS バージョン 5 リリース 1
- IBM Enterprise COBOL for z/OS バージョン 5 リリース 2

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

- 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

詳しくは、 xviii ページの『COBOL コンパイラーに対する変更の要約』を参照してください。

IBM COBOL コンパイラー、名前およびバージョン別

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	製品番号
OS/VS COBOL	バージョン 1 リリース 2 モディフィケーション 4	5740-CB1
VS COBOL II	バージョン 1 リリース 3 およびバージョン 1 リリース 4	5668-958
COBOL/370	バージョン 1 リリース 1	5688-197

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 (続き)

コンパイラー	リリース・レベル	製品番号
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
Enterprise COBOL for z/OS	バージョン 3 リリース 1	5655-G53
Enterprise COBOL for z/OS	バージョン 3 リリース 2	5655-G53
Enterprise COBOL for z/OS	バージョン 3 リリース 3	5655-G53
Enterprise COBOL for z/OS	バージョン 3 リリース 4	5655-G53
Enterprise COBOL for z/OS	バージョン 4 リリース 1	5655-S71
Enterprise COBOL for z/OS	バージョン 4 リリース 2	5655-S71
Enterprise COBOL for z/OS	バージョン 5 リリース 1	5655-W32
Enterprise COBOL for z/OS	バージョン 5 リリース 2	5655-W32

ランタイム・ライブラリーを Language Environment に移行するときに役立つように、既存の VS COBOL II ロード・モジュールおよび OS/VS COBOL ロード・モジュールを Language Environment の下で実行する方法に関する情報 (サポートのリンク・エディット要件、および互換動作用の推奨ランタイム・オプションなど) が「*Enterprise COBOL for z/OS Compiler and Runtime Migration Guide Version 4 Release 2*」(http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf) に記載されています。

本書では、古い COBOL コンパイラーから Enterprise COBOL への移行に役立つように、古い COBOL コンパイラーと Enterprise COBOL の言語の相違を説明するとともに、ソース・プログラムを Enterprise COBOL プログラムに変換するときに役立てることができる IBM 変換ツールについて説明します。また、Enterprise COBOL を使用するためにアプリケーション開発プロセスにおいて変更が必要となる可能性のある他の相違についても説明します。

謝辞

IBM は、OS/VS COBOL から VS COBOL II への移行ガイドの準備における 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 IBM のお客様から得た情報は、この「移行ガイド」の作成に役立ちました。これらのご支援がなければ、本書の作成はさらに困難であったと思われます。

資料の使い方

Enterprise COBOL および Language Environment と共に提供される資料は、z/OS 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 バージョン 5

表 2. Enterprise COBOL for z/OS の資料

作業	資料
保証情報を理解する	*Licensed Program Specifications*
z/OS のもとでコンパイラーをインストールする	*Program Directory for Enterprise COBOL*
プロダクトの変更内容を理解し、ソースを最新バージョンの Enterprise COBOL for z/OS にアップグレードする	*移行ガイド.*
ランタイム環境を Language Environment にアップグレードする	**注:** ランタイム・ライブラリーを Language Environment にまだ移行していない場合は、「*Enterprise COBOL for z/OS Compiler and Runtime Migration Guide Version 4 Release 2*」(http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf) で、移行についての説明を参照してください。
Enterprise COBOL for z/OS をカスタマイズする	*Enterprise COBOL カスタマイズ・ガイド*
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	*Enterprise COBOL プログラミング・ガイド*
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	*Enterprise COBOL 言語解説書*

z/OS のLanguage Environment エレメント

表 3. z/OS の Language Environment エレメントの資料

作業	資料
プロダクトを評価する	*Language Environment 概念*
Language Environment のインストール	*z/OS Program Directory*
Language Environment のプログラム・モデルおよび概念を理解する	*Language Environment プログラミング・ガイド*
Language Environment 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	*Language Environment プログラミング・リファレンス*
Language Environment 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・メッセージに関する詳細を入手し、Language Environment に関する問題を診断する	*Language Environment デバッグ・ガイドおよびランタイム・メッセージ*
Language Environment のリリース間でアプリケーションを移行する	*Language Environment ランタイム マイグレーション・ガイド*

表 3. z/OS の Language Environment エレメントの資料 (続き)

作業	資料
言語間通信 (ILC) アプリケーションを開発する	*Language Environment ILC (言語間通信) アプリケーションの作成*
共通デバッグ・アーキテクチャー (CDA) の概念および使用法について学習する	*共通デバッグ・アーキテクチャー ユーザーズ・ガイド*
デバッグ・データ・プログラム情報ライブラリー (llibddpi) の API に関する詳細を入手する	*共通デバッグ・アーキテクチャー ライブラリー・リファレンス*
DWARF 4 標準の DWARF API および ELF API への IBM 拡張に関する詳細を入手する	*DWARF/ELF エクステンション ライブラリー・リファレンス*

本書の変更の要約

このセクションでは、IBM COBOL for OS/390 & VM バージョン 2 リリース 1 以降に、この移行ガイドの各版に加えられた主な変更をリストします。

2015 年 2 月 - GC43-0801-03 (英語原典 GC14-7383-03) における変更

- Enterprise COBOL V5.2 の変更に関する情報が『Enterprise COBOL バージョン 5 での新機能および相違点』の章に追加されました。変更は主に、以下のトピックに対するものです。
  - ソース・コードの相違
  - コンパイラー・オプションの変更
  - ユーザー作成条件ハンドラーに関するコンパイルの制約事項
  - 可変長レコード - 不正な長さの READ
  - オブジェクト指向 COBOL の使用または C プログラムとの相互運用
- XML PARSE ステートメントを含む Enterprise COBOL バージョン 3 またはバージョン 4 のプログラムのアップグレードに関する情報が追加されました。
- Enterprise COBOL for z/OS V5.2 より前のバージョンでコンパイルされた既存の COBOL プログラムを含め、拡張アドレッシング機能属性を使用した VSAM データ・セットへのアクセスに関する情報が追加されました。
- Enterprise COBOL V5 を呼び出す、または Enterprise COBOL V5 から呼び出されるアセンブラー・プログラムで汎用レジスター (GPR) の高位半分を保存および復元する方法に関する情報が記載されています。

2014 年 3 月 - GC43-0801-01 (英語原典 GC14-7383-01) における変更

いくつかの例外を除いて、COBOL プログラムの AMODE 24 実行のサポートが再び追加されました。 Enterprise COBOL 5.1.1 によってコンパイルされる多くのプログラムは、AMODE 31 または AMODE 24 のいずれかで実行されます。

GA88-5312-00 における変更 - 2013 年 6 月

この移行ガイドは、Enterprise COBOL バージョン 5.1 向けに再編成されています。Language Environment へのランタイム移行がまだ完了していない場合は、このガイドの前のバージョンを参照してください。ランタイム移行の実行に役立つ説明として、http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf にある「*Enterprise COBOL for z/OS コンパイラーおよびランタイム 移行ガイド バージョン 4 リリース 2*」を使用することができます。

この移行ガイドに対して行われた主な変更は以下のとおりです。

- Language Environment に関する情報の削除
- Enterprise COBOL バージョン 3 および Enterprise COBOL バージョン 4 からの移行に関する固有の章の追加
- Enterprise COBOL バージョン 5 に関するセクションの追加
- 他の IBM 製品とともに COBOL コンパイラーをアップグレードすることに関するセクションの追加。このセクションには、Debug Tool、CICS®、および DB2® に関する情報が含まれています。詳しくは、 213 ページの『第 5 部 Enterprise COBOL の移行およびその他の IBM 製品』を参照してください。

本書には多くの情報が記載されていますが、その多くは、ほとんどのお客様にとって不要なものです。例えば、Enterprise COBOL バージョン 4 から移行する場合、すべてのアプリケーションのランタイム移行を完了してあるのであれば、いくつかのセクションを参照するだけで済みます。詳細については、 177 ページの『第 13 章 Enterprise COBOL バージョン 4 からのアップグレード』、 185 ページの『第 14 章 Enterprise COBOL バージョン 4 プログラムのコンパイル』、および 189 ページの『第 15 章 IBM Enterprise COBOL for z/OS バージョン 5 における変更』を参照してください。

2009 年 8 月 - GC88-4746-01 (英文原典: GC23-8527-01) における変更

コンパイラー

- 組み込み DB2 コプロセッサーに関する説明が追加されました。
- XMLPARSE(COMPAT) から XMLPARSE(XMLSS) へのマイグレーションに関する説明が更新されました (例: いくつかの XML イベントの処理が変更されました)。
- XMLPARSE(XMLSS) を使用してコンパイルした場合の構文解析動作の違いに関する説明が更新されました。
- 新しい予約語が追加されました。
- 新しいコンパイラー・オプションが追加されました。
- 付録のよくある質問および回答に、以下の説明が追加されました。
  - COBOL プログラム呼び出しについて
  - 既存の COBOL アプリケーションを Java™ 5 または Java 6 を使用して実行する方法について

ランタイム

- 領域全体のデフォルトについての説明が更新されました。
- TEST オプションについての説明が更新されました。
- Language Environment STORAGE(00) オプションについての説明が更新されました。

CICS に関する説明が修正されました。

各種の保守上および編集上の変更が加えられ、例えば、 251 ページの『付録 B. COBOL 予約語の比較』 および 307 ページの『付録 F. コンパイラー限界値の比較』 が更新されました。

2007 年 12 月 - SC88-4746-00 (英文原典: GC23-8527-00) における変更

コンパイラー

- XML PARSE のEnterprise COBOL バージョン 3 から Enterprise COBOL バージョン 4 へのマイグレーションに関してセクション (XMLPARSE(COMPAT) から XMLPARSE(XMLSS) へのマイグレーション) が追加されました。
- Enterprise COBOL バージョン 4 の新しい TEST サブオプションに関する情報が追加されました。
- 新しい予約語が追加されました。
- CMPR2 から NOCMPR2 へのマイグレーションに関するセクションに、情報が追加されました。
  - JCL の固定ファイル属性および DCB= パラメーター
  - 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)
  - 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)
- DB2 コプロセッサーの統合に関する付録に情報が追加されました。
  - 分離プリコンパイラーからの違いが追加されました。

ランタイム

- Enterprise COBOL バージョン 4 リリース 1 プログラムの、SIMVRD ランタイム・オプションのサポートの削除に関する情報が追加されました。

2006 年 11 月 - SK88-8015-01 (英文原典： GC27-1409-05) における変更

- DB2 プリコンパイラーと DB2 コプロセッサー間の違いの説明が更新されました。
- コンパイラー・オプション SQLCCSID が追加されました。

2006 年 3 月 - GC88-9118-04 (英文原典： GC27-1409-04) における変更

- DB2 プリコンパイラーと DB2 コプロセッサー間の違いに関する説明が追加されました。

- SEARCH ALL ステートメントの V3R4 への移行に関するセクションが追加されました。

2005 年 7 月 - GC88-9118-03 (英文原典： GC27-1409-03) における変更

- コンパイラー・オプション MDECK が追加されました。
- 新しい予約語が追加されました。
- DB2 プリコンパイラーと DB2 コプロセッサー間の SQL コードの違いが追加されました。
- データ項目サイズの変更

2003 年 12 月 - GC88-9118-02 (英文原典: GC27-1409-02) における変更

- 本書に適用されたサービス更新

2002 年 9 月 - GC88-9118-01 (英文原典: GC27-1409-01) における変更

コンパイラー

- TEST(. . .,SYM,. . .) コンパイラー・オプションでの SEPARATE サブオプションの使用に関する情報が追加されました。

ランタイム

- 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に関する情報がより明確になりました。
- OS/390 バージョン 2 リリース 10 において RECFM=U として定義された出力ファイル用に使用するスペースの量の変更に関する情報が追加されました。
- z/OSバージョン 1 リリース 2 以降の Language Environment におけるアセンブラー・プログラムの動的呼び出しに関する情報が追加されました。

2001 年 11 月 - GC88-9118-00 (英文原典: GC27-1409-00) における変更

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 変換プログラム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

- 47 ページの『第 5 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 以降の Language Environment の場合は ABEND) の新しいデフォルトおよび OS/390 V2R7 以降の Language Environment 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。
- Language Environment 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
- Language Environment 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。

IBM Enterprise COBOL for z/OS バージョン 5 リリース 2 における変更

新しいオプションおよび変更されたオプション

- 使用可能なコンパイラー・オプションは次のとおりです。
  - 著作権
  - QUALIFY(COMPAT|EXTEND)
  - SERVICE
  - SQLIMS
  - VLR(COMPAT|STANDARD)
  - XMLPARSE(XMLSS|COMPAT)
  - ZONEDATA(PFD|MIG)
- 以下のコンパイラー・オプションは変更されました。
  - ARCH: ARCH(6) は受け入れられなくなりました。新しい上位レベルの ARCH(11) が受け入れられ、ARCH(7) がデフォルトです。
  - MAP: 新しいサブオプション HEX および DEC が MAP コンパイラー・オプションに追加されました。これらのサブオプションは、コンパイラー・リストの MAP 出力に、16 進オフセットと 10 進オフセットのどちらを表示するかを制御します。これにより、ご使用のプログラムが Enterprise COBOL V4 以前のバージョンでコンパイルされている場合に、Enterprise COBOL V5.2 への移行が容易になります。
- 次のコンパイラー・オプションは削除されました。
  - SIZE

新しい機能および変更された機能

- COBOL ライブラリーの互換モード COBOL XML パーサーがサポートされています。 XMLPARSE(XMLSS|COMPAT) コンパイラー・オプションを指定して、構文解析に z/OS XML System Services パーサーを使用するか、COBOL ライブラリーの互換モード COBOL XML パーサーを使用するかを選択できます。この機能により、Enterprise COBOL V3 で、または V4 で XMLPARSE(COMPAT) コンパイラー・オプションを指定してコンパイルされた、XML PARSE ステートメントのあるプログラムの Enterprise COBOL V5 コンパイラーへの移行が容易になります。
- Java インターオペラビリティーのためにオブジェクト指向構文を使用する Enterprise COBOL アプリケーションが、Java 6、Java 7、および Java 8 でサポートされるようになりました。 Java SDK 1.4.2 および Java 5 はサポートされなくなりました。

新しいステートメントおよび変更されたステートメント

- 新しい CALLINTERFACE ディレクティブは、CALL ステートメントおよび SET ステートメントのインターフェース規約を指定します。指定した規約は、ソースで別の CALLINTERFACE ディレクティブが検出されるまで有効のままになります。 CALLINTERFACE ディレクティブには、DLL、DYNAMIC、および STATIC の 3 つのサブオプションがあります。

- EXIT ステートメントには、以下の新しい形式が組み込まれました。これにより、GO TO ステートメントを使用せずに終了するための、構造化された方法を使用できます。新しい形式は、2002 COBOL 標準の一部です。
  - EXIT PERFORM - 行内 PERFORM ステートメントの終了
  - EXIT PARAGRAPH - パラグラフの途中からの終了
  - EXIT SECTION - セクションの終了
- SORT ステートメントの新しい形式であるテーブル SORT ステートメントは、ユーザー指定の順序でテーブル・エレメントを整列します。これは 2002 COBOL 標準の一部です。
- 新しいキーワード LEADING および TRAILING が、COPY ステートメントの REPLACING 句、および REPLACE ステートメントに追加されました。これらは部分語の置換操作を改善します。新しいキーワードは、2002 COBOL 標準の一部です。
- 新しいキーワード VOLATILE が形式 1 データ記述項目に追加されました。VOLATILE 節は、Language Environment (LE) 条件ハンドラー・ルーチンやその他の非同期のプロセスまたはスレッドなどの、コンパイラーが検出できない方法でデータ項目の値を変更または参照できることを示します。このため、データ項目に対する最適化は制限されます。
- 新しい構文が XML GENERATE ステートメントに導入されました。明示的形式の SUPPRESS 句の WHEN 句を省略して、XML GENERATE ステートメントの出力で無条件に *identifier-8* を抑止できます。 WHEN 句を省略すると、*identifier-8* をグループ・データ項目にすることができます。さらに、XML GENERATE ステートメントの *generic-suppression-phrase* は、生成された XML 出力から、データ項目のクラスおよびカテゴリー全体を、抑止基準に基づいて除外するための便利な方法を提供します。抑止指定の適用対象であり、実行時に基準に一致するデータ項目が除外されます。 CONTENT は抑止の特殊タイプとして扱われます。

IBM Enterprise COBOL for z/OS バージョン 5 リリース 1 モディフィケーション 1 における変更

- いくつかの例外を除いて、COBOL プログラムの AMODE 24 実行がサポートされています。 IBM Enterprise COBOL for z/OS V5.1.1 によってコンパイルされた多くのプログラムは、AMODE 31 または AMODE 24 で動作します。
- 新しいコンパイラー・オプション SQLIMS によって、(IMS™ では SQL ステートメント・コプロセッサーと呼ばれる) 新しい IMS SQL コプロセッサーが有効になりました。この新しいコプロセッサーは、組み込み SQLIMS ステートメントが入っているソース・プログラムを処理します。
- 新しい重大例外および警告例外コードが、XML PARSE 例外用に追加されました。
- コンパイラー・リストにおける LIST オプション出力に、すべての COBOL 特殊レジスター変数ロケーション情報を提供する、新しい特殊変数テーブルが入ります。

最新サービスが適用されている Enterprise COBOL V5.1.0 は、V5.1.1 のように動作し、以下の新しいコンパイラー・オプションを備えています。

- SQLIMS
- VLR(COMPAT|STANDARD)
- XMLPARSE(XMLSS|COMPAT)
- 新しいサブオプション HEX および DEC が MAP コンパイラー・オプションに追加されました。このサブオプションは、コンパイラー・リストの MAP 出力に、16 進または 10 進のどちらのオフセットを表示するかを制御します。

IBM Enterprise COBOL for z/OS バージョン 5 リリース 1 における変更

新規および変更された COBOL 関数

IBM Enterprise COBOL for z/OS でサポートされる XML 関数が拡張されました。

- XML GENERATE ステートメントは、生成される XML 文書の書式に関して、プログラマーにより高い柔軟性と制御性を提供する、新しい構文で拡張されました。
  - NAME 句が追加され、ユーザーによるエレメント名および属性名の指定が可能になりました。
  - TYPE 句が追加され、属性およびエレメントの生成をユーザーが制御できるようになりました。
  - SUPPRESS 句が追加され、空の属性およびエレメントの抑制が可能になりました。
- XML 構文解析サポートが特殊レジスター XML-INFORMATION で拡張されました。これにより、XML イベントに対応して送信される XML の内容が完全であるのか、または次のイベントに継続するものであるのかを容易に判別できます。
- COBOL ライブラリーからの互換モード COBOL XML パーサーは、Enterprise COBOL V5 プログラムでの使用がサポートされなくなりました。V5 プログラム内の XML PARSE ステートメントでは、z/OS XML システム・サービス内の XML パーサーが必ず使用されます。

無制限表および無制限グループが新たにサポートされたことにより、XML アプリケーションと COBOL アプリケーションの間でデータ構造のトップダウン・マッピングが可能になりました。

このリリースでは、新たに以下の 6 つの組み込み関数が追加されたことにより、Unicode サポートが拡張されました。

- ULENGTH
- UPOS
- USUBSTR
- USUPPLEMENTERY
- UVALID
- UWIDTH

新しいインライン・コメント標識 (文字ストリング「*>」) は、当該行における後続のテキストがコメントであることを示すためにコーディングできます。

Enterprise COBOL バージョン 5.1 では、不正な長さのレコードに対する READ ステートメントの処理が修正されました。

2000 年言語拡張はサポートされなくなりました。削除されたエレメントは次のとおりです。

- DATEVAL 組み込み関数
- UNDATE 組み込み関数
- YEARWINDOW 組み込み関数
- DATEPROC コンパイラー・オプション
- YEARWINDOW コンパイラー・オプション

C および C++ で使用されていた規則との互換性をとるため、RETURNING 句の PROCEDURE DIVISION ヘッダーで指定されたダブルワード・バイナリー項目や、CALL ステートメントで指定されたダブルワード・バイナリー項目を返すためのリンケージ規則が変更されました。 COBOL プログラムが、CALL ... RETURNING ステートメントが指定された呼び出し側 COBOL プログラムに、PROCEDURE DIVISION RETURNING ヘッダーでダブルワード・バイナリー項目を返す場合、プログラムの 1 つのみが Enterprise COBOL V5 で再コンパイルされていると、問題が発生します。 RETURNING 項目のリンケージ規約が整合するように、呼び出されるプログラムおよび呼び出し側プログラムの両方を一緒に Enterprise COBOL V5 で再コンパイルする必要があります。

フォーマット-2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS はサポートされなくなりました。

オプションの変更

- 使用可能なコンパイラー・オプションは次のとおりです。
  - AFP(*VOLATILE* | *NOVOLATILE*)
  - ARCH(*n*)
  - DISPSIGN(*SEP* | *COMPAT*)
  - HGPR(*PRESERVE* | *NOPRESERVE*)
  - MAXPCF(*nnn*)
  - STGOPT | NOSTGOPT
- 以下のコンパイラー・オプションは変更されました。
  - コンパイラーは LIB オプションが常に有効であるかのように動作するため、MDECK オプションは LIB オプションに依存しなくなりました。
  - NUMPROC コンパイラー・オプションの MIG サブオプションは、サポートされていません。
  - ランタイム・オプション CHECK(OFF) を使用しても、コンパイルされた範囲検査を実行時に無効にすることはできません。
  - 16 MB 境界より上での NORENT プログラムの実行はサポートされません。
  - TEST コンパイラー・オプションの HOOK | NOHOOK サブオプションおよび SEPARATE | NOSEPARATE サブオプションは、サポートされなくなりました。こ

れらのサブオプションは、移行を容易にするために引き続き許容されます。新しいサブオプション SOURCE および NOSOURCE が TEST コンパイラー・オプションに追加されました。

  - NOTEST オプションが拡張され、DWARF および NODWARF サブオプションが組み込まれました。
  - EXIT コンパイラー・オプションは、DUMP コンパイラー・オプションと相互に排他的ではなくなり、コンパイラー出口規則が更新されました。
  - OPTIMIZE オプションは、複数レベルの最適化を設定できるように変更されました。以前の OPTIMIZE オプション・フォーマットは非推奨ですが、互換性のための使用は許容されます。
  - LIST オプションから生成されるリストのフォーマットおよび内容が新しくなりました。
  - MAP オプションから生成されるリスト出力のフォーマットおよび内容が変更されました。
- 以下のコンパイラー・オプションに対するサポートが廃止されました。
  - DATEPROC
  - LIB
  - SIZE(MAX)
  - YEARWINDOW
  - XMLPARSE

コンパイラー動作の変更

Enterprise COBOL V5.1 では、異なる動作となる変更がいくつか行われています。

- Enterprise COBOL V5.1.0 でコンパイルされたプログラムの AMODE 24 の実行はサポートされなくなりました。 Enterprise COBOL V5.1.0 実行可能モジュールは AMODE 31 でなければなりません。
- 再使用可能な COBOL 環境を管理するための IGZERRE および ILBOSTP0 インターフェースは、Enterprise COBOL V5 でコンパイルされたプログラムを含むアプリケーションではサポートされません。
- 静的呼び出しを動的呼び出しに変換するための IGZBRDGE マクロは、Enterprise COBOL V5 でコンパイルされたプログラムではサポートされません。
- COBOL ライブラリーからの互換モード COBOL XML パーサー (Enterprise COBOL V3 からの古いパーサー) は、Enterprise COBOL V5 プログラムで使用できなくなりました。V5 プログラム内の XML PARSE ステートメントには、常に z/OS システム・サービス XML パーサー (XMLSS) が使用されます。
- Enterprise COBOL バージョン 5 では現在、コンパイル時に言語環境プログラムが必要です。MVS LNKLST または LPALST に言語環境プログラム・データ・セット SCEERUN と SCEERUN2 がインストールされていない場合は、コンパイル用にこれらのデータ・セットを STEPLIB または JOBLIB 連結に組み込む必要があります。
- Enterprise COBOL バージョン 5.1 の新規言語環境プログラム・メンバー ID は 4 です。以前のバージョンの COBOL では ID 5 が使用されていました。

- Enterprise COBOL バージョン 5 プログラムには、旧バージョンの COBOL とのインターオペラビリティーにいくつかの制約事項があります。詳細については、24 ページの『古いレベルの IBM COBOL プログラムとのインターオペラビリティー』を参照してください。
- 以下の特性を持つ COBOL プログラムの動作は、Enterprise COBOL V5 と、それより前のバージョンとでは異なる場合があります。
  - サポートされない COBOL 言語構文を使用するプログラム。
  - 実行時に、データ記述項目の PICTURE 文節に準拠しない値を含むデータ項目を参照するプログラム。以下に例を示します。
    - サイズ超過値の +123456789 を含む、ピクチャー S9(6) USAGE BINARY が指定されたフルワード・バイナリー項目 (TRUNC(BIN) オプションが指定されている場合は除く)
    - サイズ超過値 123 (16 進数の 123C など) を含む、ピクチャー S99 が指定された 2 バイトの PACKED-DECIMAL 項目。
    - 無効または非優先の符号を含み、データ記述項目の符号要件および有効な NUMPROC(PFD) コンパイラー・オプション設定に準拠しないパック 10 進数またはゾーン 10 進数項目。
  - 未診断の添え字範囲エラーが発生していて (SSRANGE コンパイラー・オプションが指定されていない場合)、基本データ項目に対するストレージ割り振り以外のストレージを参照するプログラム。
  - 生成された特定のコード・シーケンス、レジスター規則、内部 IBM 制御ブロックに対する低レベルの依存関係を持つアプリケーションの Enterprise COBOL V5 での動作は、以前のバージョンにおける動作と異なる場合があります。
  - 整数-2 より大きい値を OCCURS DEPENDING ON 節のオブジェクトに指定することは不正であるため、その動作は確定されません。ただし、Enterprise COBOL V5.1 は、これが発生したときに旧バージョンとは異なる動作をします。
- DATA(31) が有効な場合、再入可能 COBOL プログラム用の VSAM レコード域は 16 MB より上に割り振られます。 VSAM ファイル・レコード内のデータを CALL ... USING BY REFERENCE パラメーターとして AMODE 24 サブプログラムに渡すプログラムが、影響を受けることがあります。このようなプログラムは、DATA(24) コンパイラー・オプションを指定して再コンパイルするか、言語環境プログラムの HEAP(BELOW) オプションを使用すると、レコードが AMODE 24 プログラムによってアドレッシング可能になります。
- コンパイル時のストレージ要件は、以前のバージョンの Enterprise COBOL と比べると、大幅に増加しています。SIZE オプションの説明を参照してください。これは、より高い最適化レベル、すなわち、OPT(1) または OPT(2) コンパイラー・オプションでコンパイルされたプログラムの場合に特に顕著です。
- コンパイル時の CPU 時間要件は、以前のバージョンの Enterprise COBOL と比べると、大幅に増加しています。
- コンパイル時と実行時の診断メッセージは異なる場合があり、異なるタイミングまたは場所で生成されることがあります。
  - 情報レベルおよび警告レベルの診断の有無が異なる場合があります。

– サポートされていない過度の量のストレージを定義するプログラムは、コンパイル時にコンパイラーで診断されずに、バインド時にバインダーで診断されるか、実行時に言語環境プログラムで診断されることがあります。

- コンパイラー・リストのフォーマットと内容は、以前のバージョンの Enterprise COBOL のものとは異なります。

アプリケーション・パフォーマンスの変更

いくつかのレベルのアプリケーション・パフォーマンス最適化をサポートするように OPTIMIZE オプションが変更されました。サブオプションも変更されました。以前の OPTIMIZE オプション・フォーマットは非推奨ですが、互換性のための使用は許容されます。

注: OPT(0) は多くの点で以前の NOOPTIMIZE オプションと同等ですが、OPT(0) は、以前の NOOPTIMIZE では削除されなかった一部の到達不能コードを削除します。

デバッグの変更

TEST オプションを指定すると、DWARF デバッグ情報がアプリケーション・モジュールに含められます。

プログラム・オブジェクトに NOLOAD デバッグ・セグメントがある場合、Enterprise COBOL V5 デバッグ・データは常に実行可能ファイルに適合し、常に使用可能です。この場合、検索先となるデータ・セットのリストは提供されず、ロードされるプログラムのサイズが増えることはありません。

TEST(SOURCE) オプションが指定された場合は、DWARF デバッグ情報に拡張ソース・コードが含まれるため、IBM Debug Tool でコンパイラー・リストは不要になります。TEST(NOSOURCE) が指定されていると、生成された DWARF デバッグ情報に拡張ソース・コードは含まれません。

NOTEST(DWARF) オプションを使用して、基本 DWARF 診断情報をアプリケーション・モジュールに含めることができます。これにより、CEEDUMP や IBM Fault Analyzer などのアプリケーション障害分析ツールが実現されます。

パッケージ化および JCL (ジョブ制御言語) の変更

Enterprise COBOL V5.1 では、パッケージ化、インストール、および JCL にいくつかの変更が行われています。

SIGYCOMP データ・セットは、以前のバージョンでは PDS データ・セットでしたが、現在は PDSE データ・セットになりました。

Enterprise COBOL バージョン 5.1 では追加のデータ・セットが必要になります。

- z/OS TSO またはバッチでコンパイルする場合、COBOL コンパイラーには現在、SYSUT1 から SYSUT15 までの 15 個のユーティリティー・データ・セットが必要です。
- SYSMDECK データ・セットは、すべてのコンパイルに必要となりました。 NOMDECK オプションが指定されている場合、SYSMDECK をユーティリティー (一時) デー

タ・セットとして指定できます。MDECK(...) を指定するときは、SYSMDECK DD 割り振りで永続データ・セットを指定する必要があります。

- 代替 DDNAME リスト・パラメーターは、COBOL コンパイラーがアセンブリー言語プログラムから呼び出されるときに使用され、追加の作業データ・セット用の項目により拡張されます。

Enterprise COBOL バージョン 5.1 に同梱されているカタログ式プロシージャーは変更されています。

- IGYWC
- IGYWCL
- IGYWCLG

以下の JCL カタログ式プロシージャーはサポートされなくなりました。これらのプロシージャーはすべて、言語環境プログラム・プリリンカーまたは DFSMS ローダーを使用しますが、言語環境プログラム・プリリンカーも DFSMS ローダーもサポートされなくなったことが、その理由です。

- IGYWCG
- IGYWCPG
- IGYWCPL
- IGYWCPLG
- IGYWPL

制限

IMS 出口ルーチンに COBOL が使用されていれば、その出口が、PDSE データ・セット内の COBOL V5.1 プログラムをロードして呼び出す、PDS データ・セット内のアセンブラー・プログラムである場合のみ、Enterprise COBOL V5.1 はプログラムをコンパイルできます。この制限を回避する方法については、 237 ページの『第 20 章 IMS プログラムを Enterprise COBOL V5 に移動する』を参照してください。

IBM Enterprise COBOL for z/OS バージョン 4 リリース 2 における変更

- z/OS システム・サービス XML パーサーを使用するときに、新規および拡張 XML PARSE 機能が使用可能です。
  - XML PARSE ステートメントの VALIDATING 句を使用する場合、XML スキーマに対する妥当性検査を伴う文書解析を行うことができます。
  - XMLPARSE(XMLSS) コンパイラー・オプションを指定した、検証を行わない解析のパフォーマンスが、Enterprise COBOL バージョン 4 リリース 1 での XMLPARSE(XMLSS) コンパイラー・オプションを指定した検証を行わない解析のパフォーマンスに比べて向上しました。
  - 文書の 1 バイト EBCDIC コード・ページに含まれていない文字への参照が含まれている XML 文書に対する文字処理が改良されました。
- コンパイラー・メッセージおよび FIPS (FLAGSTD) メッセージをカスタマイズ (重大度を変更またはメッセージを抑制) する機能が、EXIT コンパイラー・オプションの新しいサブオプション MSGEXIT によって可能になりました。

- 新しいコンパイラー・オプション BLOCK0 は、プログラム内のすべての適格な QSAM ファイルに対する暗黙的な BLOCK CONTAINS 0 節をアクティブにします。
- データ名やプログラム名などのユーザー定義語で下線文字 (_) がサポートされるようになりました。下線はリテラル形式のプログラム名でもサポートされます。
- 組み込み CICS 変換プログラムを使用する場合、コンパイラーのリストに、有効な CICS オプションが示されるようになりました。
- Java SDK 1.4.2 に加えて、Java 5 および Java 6 で、Java インターオペラビリティーのためのオブジェクト指向構文を使用する Enterprise COBOL アプリケーションがサポートされるようになりました。

IBM Enterprise COBOL for z/OS バージョン 4 リリース 1 における変更

- XML GENERATE ステートメントは、生成される XML 文書の書式に関して、プログラマーにより高い柔軟性と制御性を提供する、新しい構文で拡張されました。
  - WITH ATTRIBUTES 句を使うと、XML 文書内の適格な項目は、エレメントではなく XML 属性として生成されます。
  - WITH ENCODING 句を使うと、ユーザーは生成される文書のエンコード方式を指定できます。
  - WITH XML-DECLARATION 句を使うと、文書内にバージョンとエンコード方式の情報が生成されます。
  - NAMESPACE および NAMESPACE-PREFIX 句を使うと、XML 名前空間を使用する XML 文書を生成できます。
  - XML GENERATE ステートメントは、UTF-8 Unicode でエンコードされた XML 文書の生成をサポートするようになりました。
- XML PARSE のサポートが拡張されました。
  - COBOL ライブラリーの一部である既存の XML パーサーの代わりとして、z/OS System Services XML パーサーがサポートされました。
  - z/OS System Services XML パーサーには以下の利点があります。
    - COBOL ユーザーのための最新の IBM 構文解析テクノロジーが使用可能です。
    - COBOL XML 構文解析を zAAP 専門プロセッサーにオフロードできます。
    - XML 名前空間を使用する XML 文書の構文解析のサポートが改良されました。
    - UTF-8 Unicode でエンコードされた XML 文書の構文解析が直接サポートされます。
    - 非常に大規模な XML 文書を、一度に 1 つのバッファーで構文解析のサポートをします。
  - XML PARSE ステートメントの実行中に名前空間を処理するために、新規に 4 つの特殊レジスターが導入されました。

  - XML PARSE ステートメントが、新しい構文で拡張されました。新しい WITH ENCODING および RETURNING NATIONAL 句により、プログラマーは入力 XML 文書の想定されるエンコード方式を制御することができ、Unicode での構文解析が容易になります。
  - 新しいコンパイラー・オプション XMLPARSE が作成され、XML PARSE ステートメントで z/OS System Services パーサーまたは既存の COBOL パーサーのどちらを使用するかを制御できます。XMLPARSE(COMPAT) オプションを指定した場合、XML 構文解析は Enterprise COBOL バージョン 3 と完全に互換です。デフォルトの XMLPARSE(XMLSS) オプションを指定した場合、z/OS System Services パーサーが使用され、新しい XML 構文解析機能が使用可能になります。
- 新しいz/Architecture® 命令を利用することにより、COBOL アプリケーション・プログラムのパフォーマンスが拡張されました。COBOL Unicode のサポート (USAGE NATIONAL データ) のパフォーマンスが著しく改良されました。
- DB2 バージョン 9 の利用および、コプロセッサーの統合および使用可能度の向上をはじめとして、このリリースで DB2 のサポートが拡張されました。
  - DB2 V9 が提供する新しい SQL データ型と新しい SQL 構文のサポート
  - コンパイラー・リストに、DB2 プリコンパイラー・オプションが表示されます。(DB2 V9 のみ)
  - コンパイラー・リスト中で、SQLCA および SQLDA 制御ブロックが展開されます (すべての DB2 リリース)。
  - 新しいコンパイラー・オプション SQLCCSID が提供され、コード化文字セット ID (CCSID) を COBOL と DB2 との間で調整できます。
- DFSMS 大規模フォーマット・データ・セットのサポート
- デバッグ機能の強化
  - Debug Tool V8 の使用可能化と新しいデバッグ・コマンド
  - 最適化されたコード内での GOTO/JUMPTO と、新しい TEST サブオプション EJPD
- コンパイラー・オプションをデータ・セット内で指定可能 (OPTFILE オプション)
- コンパイラー・リスト内での COPY ステートメント、ライブラリー、およびデータ・セットの相互参照

IBM Enterprise COBOL for z/OS バージョン 3 リリース 4 における変更: サービス・アップデート、2006 年 11 月

- PK31411: 新しいコンパイラー・オプション SQLCCSID は、DB2 コプロセッサーとともに機能し、CODEPAGE コンパイラー・オプションが COBOL プログラム内の SQL ステートメントの処理に影響するかどうかを決定します。SQLCCSID は、 APAR PK31411 から追加されました。
- PK16765: SEARCH ALL ステートメントの動作に対しては多くの修正が加えられています。

  現行のサービス (特に APAR PK16765 の API) が適用されていると、新しいコンパイラー診断メッセージおよびランタイム診断メッセージが追加されているので、こうした修正の対象となる可能性のあるプログラムおよび SEARCH ALL ステートメントを特定し、V3R4 に移行するために修正を加える際にご利用いただ

けます。コンパイラーにこの PTF がある場合、リスト・ヘッダーとオブジェクト・プログラムは、バージョン 3 リリース 4 モディフィケーション 1 と表示します。

IBM Enterprise COBOL for z/OS バージョン 3 リリース 4

- COBOL データ項目サイズのいくつかの限界値が大幅に引き上げられました。例えば、
  - データ項目の最大サイズが 16 MB から 128 MB に引き上げられました。
  - 最大の PICTURE 記号複製が 134,217,727 に引き上げられました。
  - 最大の OCCURS 整数が 134,217,727 に引き上げられました。

  (変更されたコンパイラー限界値の詳細については、「COBOL 言語解説書」を参照してください。) このサポートにより、大量のデータを使うプログラミングが可能になります。例えば、
  - DB2 BLOB および CLOB データ型を使用する DB2/COBOL アプリケーション
  - 大規模 XML 文書を解析または生成する COBOL XML アプリケーション
- 国別 (Unicode UTF-16) データのサポートが拡張されました。追加された数種類のデータ項目は、USAGE NATIONAL として暗黙的にまたは明示的に記述することができます。例えば、次のようなデータ項目です。
  - 外部 10 進数 (*国別 10 進数*) 項目
  - 外部浮動小数点 (*国別浮動小数点*) 項目
  - 数字編集項目
  - 国別編集項目
  - GROUP-USAGE NATIONAL 文節によってサポートされる、グループ (*国別グループ*) 項目
- 多くの COBOL 言語要素が新しい種類の UTF-16 データをサポートします。つまり、国別データの処理を新しくサポートするようになりました。すなわち、次のようなデータがサポートされます。
  - USAGE NATIONAL の数値データ (国別 10 進数および国別浮動小数点数) は、算術演算、および数値オペランドをサポートするすべての言語構成要素の中で使用できます。
  - USAGE NATIONAL の編集済みデータは、既存の編集済みの型と同じ言語構成要素の中でサポートされます。移行に関連した編集操作および編集解除操作が含まれます。
  - すべての国別データを含むグループ項目は、GROUP-USAGE NATIONAL 文節で定義することができます。その結果、そのグループは、ほとんどすべての言語構成要素の中で基本項目として振る舞えるようになります。このサポートにより、STRING、UNSTRING、および INSPECT などのステートメントで国別グループが使用できるようになりました。
  - XML GENERATE ステートメントは、受信データ項目として国別グループをサポートし、送信データ項目として国別編集済み項目、USAGE NATIONAL の数値編集済み項目、国別 10 進数、国別浮動小数点数、および国別グループ項目をサポートします。

– NUMVAL および NUMVAL-C 組み込み関数は、引数として国別リテラルまたは国別データ項目を取ることができます。

このような新規国別データ機能を使用することにより、すべてのアプリケーション・データに Unicode を排他的に使用する COBOL プログラムを開発できるようになりました。

- REDEFINES 文節は、レベル 01 でないデータ項目の場合、その項目のサブジェクトを、再定義されるデータ項目より大きくできるように拡張されました。
- 新規コンパイラー・オプション MDECK は、ライブラリー処理ステートメントからの出力がファイルに書き出されるようにします。
- DB2 コプロセッサー・サポートが拡張されました。XREF が改良されています。
- クラス「NATIONAL」のデータ項目の VALUE 文節のリテラルに英数字を使用できます。

このリリースでは、次のような用語の変更も行われています。

- 用語 *英数字グループ* は、国別グループ以外のグループを明示的に指すために導入されました。
- 用語 *グループ* は、明らかに英数字グループのみ、または国別グループのみを指すコンテキストに使用された場合を除いて、英数字グループと国別グループの両方を意味します。
- 用語 *外部 10 進数* は、ゾーン 10 進数項目と国別 10 進数項目の両方を指します。
- 用語 *英数字浮動小数点数* は、USAGE DISPLAY が指定された外部浮動小数点項目を指すために導入されました。
- 用語 *外部浮動小数点数* とは、英数字浮動小数点項目と国別浮動小数点項目の両方を指します。

IBM Enterprise COBOL for z/OS バージョン 3 リリース 3 における変更

- XML サポートが拡張されました。新しいステートメント XML GENERATE を使用して、COBOL データ・レコードの内容を XML 形式に変換できます。XML GENERATE では、Unicode UTF-16、または 1 バイト文字の EBCDIC コード・ページのいずれかでエンコードされる XML 文書を作成します。
- Debug Tool に次の新機能の追加および機能強化が行われました。
  - COBOL SYSDEBUG ファイルを使用する際のパフォーマンスが向上します。
  - 各国語データを使用するプログラムのデバッグがさらに容易になります。各国語データを定様式ダンプで、または Debug Tool の LIST コマンドを使用して表示する場合、データは、CODEPAGE コンパイラー・オプションで指定されたコード・ページを使用して、EBCDIC 表記に自動的に変換されます。Debug Tool の MOVE コマンドを使用して、各国語データ項目に値を割り当てでき、またグループ・データ項目との間で各国語データ項目を移動できます。各国語データを Debug Tool の条件コマンド (IF や EVALUATE など) での被比較値として使用できます。
  - COBOL と Java の混合アプリケーション、COBOL クラス定義、およびオブジェクト指向構文を含む COBOL プログラムをデバッグできます。

デバッグ・サポートに対するこれらの機能拡張の詳細については、「*Debug Tool ユーザーズ・ガイド*」を参照してください。

- 組み込み DB2 コプロセッサーを使用する場合に、DB2 バージョン 8 の SQL 機能がサポートされます。
- COBJVMINITOPTIONS 環境変数に指定するオプションの構文が変更されました。

IBM Enterprise COBOL for z/OS および OS/390 版バージョン 3 リリース 2 における変更

- コンパイラーが拡張され、以下の Debug Tool の機能をサポートします。
  - プレーバック・サポートによりアプリケーション実行パスおよびデータ値の記録および再生が可能です。
  - 自動モニター・サポートにより、デバッグ中のステートメント内で参照される変数の値が表示されます。
  - OPTIMIZE および TEST(NONE,SYM,.. .) オプションでコンパイルされたプログラムはデバッグに対応します。
  - Debug Tool GOTO コマンドは、サブオプション付きの NOOPTIMIZE オプションおよび TEST オプションでコンパイルされたプログラムに使用することができます。 (以前のリリースでは、GOTO コマンドは TEST(NONE, . . .) でコンパイルされたプログラムをサポートしていませんでした。)

  デバッグ・サポートに対するこれらの機能拡張の詳細については、「*Debug Tool ユーザーズ・ガイド*」を参照してください。
- IMS (情報管理システム) への Java とのインターオペラビリティーの拡張: オブジェクト指向の COBOL プログラムが IMS の Java 従属リージョンで稼働します。オブジェクト指向の COBOL と Java 言語は、単一のアプリケーションで組み合わせて使用することができます。
- Java とのインターオペラビリティーの拡張サポート:
  - OPTIMIZE コンパイラー・オプションは、Java とのインターオペラビリティー用の OO 構文規則を含むプログラムを完全にサポートします。
  - jobjectArray タイプのオブジェクト参照子は、COBOL および Java 間の相互協調処理をサポートします。
  - COBOL のメイン・ファクトリー・メソッドで始まる OO アプリケーションは、 java コマンドで起動させることができます。
  - 新規環境変数 COBJVMINITOPTIONS は、COBOL プログラムで始まる OO アプリケーション用の Java 仮装計算機 (VM) を初期化します。
  - COBOL プログラムで始まる OO アプリケーションは、制限をいくつか伴いますが、PDSE (拡張区分データ・セット) 内のモジュールとして結合させることができ、JCL (ジョブ制御言語) バッチを使用して稼働します。
- DB2 で作動する Unicode の機能強化: ホスト変数用のコード・ページは DB2 統合コプロセッサーを使用する時に暗黙的に取り扱われます。 SQL DECLARE ステートメントは COBOL および DB2 のコード・ページが一致しない場合、USAGE DISPLAY または USAGE DISPLAY-1 で記述される変数のためにのみ必要です。

IBM Enterprise COBOL for z/OS および OS/390 版バージョン 3 リリース 1 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 変換プログラムの使用と 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
- コンパイラー・オプション DBCS、FLAG(I,I)、RENT、および XREF(FULL) のデフォルト値の変更

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あての 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 コプロセッサー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・オプションはインストール・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。

ご意見の送付方法

本書または Enterprise COBOL の他のマニュアルについてご意見がありましたら、IBM 発行のマニュアルに関する情報の Web ページ (http://www.ibm.com/jp/manuals/) よりお送りください。今後の参考にさせていただきます。(URL は、変更になる場合があります)

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

アクセシビリティー

アクセシビリティー機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。 z/OS のアクセシビリティー機能は、Enterprise COBOL のアクセスを支援します。

z/OS のアクセシビリティーの主要機能は、次のとおりです。

- スクリーン・リーダーおよびスクリーン拡大ソフトウェアで一般的に使用されるインターフェース
- キーボードのみによるナビゲーション
- 色、コントラスト、フォント・サイズなどの表示属性をカスタマイズできる機能

インターフェース情報

支援テクノロジー製品は、z/OS で提供されるユーザー・インターフェースで作動します。具体的な手引きとなる情報については、z/OS インターフェースのアクセスに使用する支援テクノロジー製品の資料を参照してください。

キーボード・ナビゲーション

ユーザーは、TSO/E または ISPF を使用して z/OS ユーザー・インターフェースにアクセスできます。

また、ユーザーは、IBM Rational Developer for System z® を使用して z/OS サービスにアクセスすることもできます。

これらのインターフェースのアクセスに関する説明は、以下の資料を参照してください。

- 「*z/OS TSO/E Primer*」
- 「*z/OS TSO/E User's Guide*」
- 「*z/OS ISPF User's Guide Volume I*」
- IBM Rational® Developer for System z インフォメーション・センター

上記の資料には、キーボード・ショートカットまたはファンクション・キー (PF キー) の使用方法を含む TSO/E および ISPF の使用方法が記載されています。それぞれの資料では、PF キーのデフォルトの設定値とそれらの機能の変更方法についても説明しています。

本書のアクセシビリティー

本書の英語版は XHTML 形式で IBM System z Enterprise Development Tools & Compilers Information Center (publib.boulder.ibm.com/infocenter/pdthelp/index.jsp) から入手でき、スクリーン・リーダーを使用する視覚障害者の方がご利用になれます。

スクリーン・リーダーが構文図、ソース・コード例、およびピリオドやコンマといった PICTURE 記号を含むテキストを正確に読み取ることができるようにするには、句読点をすべて読み上げるようにスクリーン・リーダーを設定する必要があります。

IBM とアクセシビリティー

アクセシビリティーに関する IBM の方針について詳しくは、www.ibm.com/jp/accessibility/ にある 「アクセシビリティ・センター」を参照してください。

第 1 部 概要

第 1 章 新しいコンパイラーとランタイムの紹介

このセクションでは、Enterprise COBOL コンパイラー (IBM Enterprise COBOL for z/OS) および共通のランタイム (Language Environment) の概要を示し、本書全体で使用する用語を紹介します。

Enterprise COBOL バージョン 5 実行可能ファイルはプログラム・オブジェクトで、PDSE データ・セットにのみ常駐できます。ご使用の COBOL ロード・ライブラリーが PDS データ・セットにあれば、それらを PDSE データ・セットに移行してください。

本書では、Language Environment へのランタイムの移行が完了していることを前提としています。これはどのような意味なのかを説明します。簡単に言うと、COBOL ランタイムの移行を完了するために、以下の条件を事前に満たす必要があります。

- 言語環境プログラムのデータ・セット SCEERUN が、LNKLST または LPALST にインストールされている。
- LNKLST または LPALST 内に COBLIB、VSCLLIB、および COB2LIB のインスタンスがない。
- バッチ・ジョブ内の JCL STEPLIB/JOBLIB ステートメントや、CICS 始動 JCL に、COBLIB、VSCLLIB、COB2LIB のインスタンスがない。
- NORES でコンパイルされたプログラム用の、静的にバインドされたランタイム・ライブラリー・ルーチンがすべて、言語環境プログラムにあるルーチンで置き換えられている。
- RES でコンパイルされた VS COBOL II プログラム用の IGZEBST ブートストラップ・モジュールが、APAR PN74000 が適用された VS COBOL II ランタイム・バージョンの IGZEBST か、または言語環境プログラムにある IGZEBST で置き換えられた IGZEBST のいずれかにリンクしていた。CICS プログラムでは、COBOL V5 プログラムをアプリケーションに導入する場合、APAR PI33330 の PTF がインストールされた LE から IGZEBST を使用して再リンクすることが必要になります。

これらの 4 つの条件を理解していて、すべてを満たしている場合、29 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』にスキップできます。

これらの 4 つの条件を理解しているが、インストール先がランタイム・ライブラリーの移行を完了していない場合、本書を使用する前にその移行を完了する必要があります。Language Environment への移行を行うときは、「*Enterprise COBOL for z/OS Compiler and Runtime Migration Guide Version 4 Release 2*」(http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf) を参考にしてください。

これらの条件を理解していない場合は、これらの概要の章を引き続きお読みください。インストール先がランタイム・ライブラリーの移行を完了していないことが判明した場合、ランタイム・ライブラリー移行の実行に役立つ説明として「*Enterprise COBOL for z/OS コンパイラーおよびランタイム 移行ガイド バージョン 4 リリース 2*」を使用してください。

このセクションでは、Enterprise COBOL コンパイラー (IBM Enterprise COBOL for z/OS) および共通のランタイム (Language Environment) の概要を示し、本書全体で使用する用語を紹介します。このセクションには、以下の情報が含まれます。

- 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
- 各コンパイラーに対する Language Environment 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
- 段階的移行の提案
- 新しいコンパイラーとランタイムに関する変更
- 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総称的に指しています。

- IBM Enterprise COBOL for z/OS および OS/390、バージョン 3 リリース 1
- IBM Enterprise COBOL for z/OS および OS/390、バージョン 3 リリース 2
- IBM Enterprise COBOL for z/OS バージョン 3 リリース 3
- IBM Enterprise COBOL for z/OS バージョン 3 リリース 4
- IBM Enterprise COBOL for z/OS バージョン 4 リリース 1
- IBM Enterprise COBOL for z/OS バージョン 4 リリース 2
- IBM Enterprise COBOL for z/OS バージョン 5 リリース 1

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

- 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

詳しくは、 xviii ページの『COBOL コンパイラーに対する変更の要約』を参照してください。

プロダクトの関係: コンパイラー、ランタイム・ライブラリー、デバッグ

IBM Enterprise COBOL for z/OS は、zSeries プラットフォーム用の IBM の戦略的 COBOL コンパイラーです。 Enterprise COBOL は、IBM COBOL、VS COBOL II、OS/VS COBOL の機能の他に、マルチスレッド使用可能化、Unicode、XML 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 CICS 変換プログラム、および組み込み DB2 コプロセッサー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 85 COBOL 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 では使用できません。

Language Environment は、COBOL、PL/I、C/C++、および FORTRAN 用に単一の言語ランタイム・ライブラリーを提供します。既存のアプリケーションのサポートに加えて、Language Environment 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、およびセットで提供される共用呼び出し可能サービスを使用することにより、単純化されます。Enterprise COBOL プログラムを実行するには、Language Environment が必要です。

デバッグ能力は Debug Tool によって提供されます。Debug Tool 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、Language Environment 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、およびその他のプログラム (アセンブラー 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。

OS/VS COBOL および VS COBOL II では、ランタイム・ライブラリーはコンパイラーとともに組み込まれていました。さらに、デバッグ・コンポーネントも単一の COBOL 製品のオプションの一部でした。 Enterprise COBOL バージョン 3 では、Debug Tool 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した。

Enterprise COBOL バージョン 5 では、コンパイラー、デバッグ・コンポーネント、およびランタイム・ライブラリーはすべて分離していますが、ランタイム・ライブラリー (Language Environment) は、z/OS オペレーティング・システムと一緒に組み込まれており、別々に購入する必要はありません。

各種の COBOL コンパイラーの比較

表 4 に、OS/VS COBOL、VS COBOL II、COBOL (MVS および VM 版)、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 4. 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
				以下のサポート: IMS 下での Java インターオペラビリティー、 Java インターオペラビリティーのための OO サポート、XML、統合 CICS 変換プログラム、マルチスレッド、 Unicode
			以下のサポート: DLL 31 桁 DB2 コプロセッサー OS/390 UNIX Debug Tool の拡張サポート	以下のサポート: DLL 31 桁 DB2 コプロセッサー OS/390 UNIX Debug Tool の拡張サポート

表 4. 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
		以下の拡張機能: オブジェクト指向の COBOL、 C インターオペラビリティー、 組み込み関数、 85 Std 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	以下の拡張機能: オブジェクト指向の COBOL、 C インターオペラビリティー、 組み込み関数、 85 Std 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	以下の拡張機能: C インターオペラビリティー、 組み込み関数、 85 Std の改訂、 以下のサポート: Language Environment Debug Tool
	85 COBOL 標準、組み込み関数なし、構造化プログラミング、DBCS 各国語、CICS インターフェース改良、31 ビットのアドレッシング、再入可能性、ファースト・ソート・オプティマイザー、対話式デバッグ (フルスクリーン・モード)	85 COBOL 標準、構造化プログラミング、DBCS 各国語、CICS インターフェース改良、31 ビットのアドレッシング、再入可能性、ファースト・ソート・オプティマイザー、対話式デバッグ (フルスクリーン・モード)	85 COBOL 標準、構造化プログラミング、DBCS 各国語、CICS インターフェース改良、31 ビットのアドレッシング、再入可能性、ファースト・ソート・オプティマイザー、対話式デバッグ (フルスクリーン・モード)	85 COBOL 標準、構造化プログラミング、DBCS 各国語、CICS インターフェース改良、31 ビットのアドレッシング、再入可能性、ファースト・ソート・オプティマイザー、対話式デバッグ (フルスクリーン・モード)
74 COBOL 標準、74 STD 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 (行モード)	COBOL 74 互換、85 STD 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	COBOL 74 互換、85 STD 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	COBOL 74 互換、85 STD 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	85 STD 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

ホストのバージョンとリリースの全リストについては、Language Environment 用および使用中のコンパイラー用の「*Licensed Program Specifications*」を参照してください。

各コンパイラーに対する Language Environment のランタイム・サポート

OS/VS COBOL ランタイム・ライブラリーでは、 OS/VS COBOL プログラムのみサポートしました。アセンブラー・プログラムは組み込むことができましたが、 VS COBOL II プログラムは組み込むことができませんでした。

VS COBOL II ランタイム・ライブラリーは、OS/VS COBOL および VS COBOL II 両方のプログラムをサポートしていました。アセンブラー・プログラムも組み込むことができました。

Language Environment は、IBM COBOL プログラムと Enterprise COBOL プログラムのみならず、 OS/VS COBOL プログラムと VS COBOL II プログラムをサポートします。さらに、Language Environment では、PL/I、C/C++ および Fortran などのその他の高水準言語をサポートします。以前のランタイム・ライブラリーと同様に、アセンブラー・プログラムを Language Environment のもとで実行されるアプリケーションに組み込むことができます。

Enterprise COBOL のバージョンが異なれば、 Language Environment の最小リリース・レベルの要件が異なります。例えば、Enterprise COBOL for z/OS バージョン 4.2 では、最小レベルとして z/OS バージョン 1 リリース 9 が必要でしたが、Enterprise COBOL for z/OS バージョン 5.1 では、最小レベルとして z/OS バージョン 1 リリース 13 が必要です。

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 Language Environment 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 5 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 5. Enterprise COBOL および Language Environment の利点

		利点を備えていない製品		
利点	**注**	**OS/VS COBOL**	**VS COBOL II**	**IBM COBOL**
XML サポート	Enterprise COBOL は、XML 文書の構文解析および生成のための新規のステートメントを提供します。このステートメントによって、プログラムで XML コンテンツを COBOL データ構造に変換し、COBOL データ構造を XML 文書に変換することができます。	X	X	X
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。 Java との相互協調処理は IMS でもサポートされます。	X	X	X
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。	X	X	X
Unicode サポート	COBOL の Unicode サポートは、プロダクト *z/OS Support for Unicode* を使用します。	X	X	X
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。	X	X	
	DB2 コプロセッサーのサポート	X	X	*

表 5. Enterprise COBOL および Language Environment の利点 (続き)

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
CICS 機能の改善	Enterprise COBOL には、CALL ステートメント・サポート (EXEC CICS LINK を使用する場合よりも高速の CICS パフォーマンスを得られる) が組み込まれていて、ユーザー作成の BLL セルは不要です。を参照してください。	X		
	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 2GB 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。	X	X	X
	組み込みの CICS 変換プログラムのサポート	X	X	*
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 を指定した引数	X	X	X
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・ロジックを単純化することができます。	X	X	
境界より上のサポート	仮想記憶制約解放 (VSCR) により、プログラムを置いたり、コンパイルしたり、プログラムにアクセスしたりする場所が、 16MB 境界より下または上のどちらにあっても構いません。	X		
	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。	X	X	
	COBOL 外部データは、境界より上に置くことができます。	N/A	X	
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。	X	X	*
z/OS UNIX システム・サービスのサポート	cob2 コマンドにより、z/OS UNIX シェルで COBOL プログラムをコンパイルして、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。	X	X	

表 5. Enterprise COBOL および Language Environment の利点 (続き)

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
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 Language Environment 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。Language Environment は、すべての条件管理を処理します。	X	X	
高精度の数学ルーチン	Language Environment 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。	X	X	
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。(例えば、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。)	X	X	
パフォーマンス	より高速の算術計算。	X		
	より高速の動的および静的 CALL ステートメント。		X	
	可変長 MOVE の向上したパフォーマンス。		X	
	Language Environment CBLPSHPOP ランタイム・オプションを使って CALL ステートメント用の PUSH HANDLE および POP HANDLE を回避すれば、CICS のパフォーマンスが速くなります。	X		
	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。	N/A	X	
向上した ILC	共通の Language Environment ライブラリーにより、COBOL と他の Language Environment 準拠言語との ILC が向上します。例えば、Language Environment 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 のもとでは、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。	X	X	
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。	X		
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。	X		

表 5. Enterprise COBOL および Language Environment の利点 (続き)

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
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 (例えば、 END-IF、 END-PERFORM、 END-READ)	X		
85 COBOL 標準の適合性	85 COBOL 標準のサポート	X		
	85 COBOL 標準の改訂 1 (組み込み関数モジュール) のサポート。	X	X	
サブシステムのサポート	IMS、ISPF、DFSORT、DB2、WAS の向上したサポート。	X		
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。	X		

表 5. *Enterprise COBOL および Language Environment の利点 (続き)*

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
Debug Tool のサポート	Debug Tool 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機能 • Rational Developer for System z 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機能	X	X	
	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 プログラムをコンパイルするための動的デバッグ機能。	X	X	
	Enterprise COBOL バージョン 4 以降のプログラムの場合: • コンパイラー TEST サブオプション EJPD を利用すれば、ゼロ以外の OPTIMIZE レベルを指定してコンパイルされたプログラム内でも予測可能な GOTO/JUMPTO を使用できます。 **注:** ゼロ以外の OPTIMIZE レベルおよび TEST(NOJEPD) を指定してコンパイルされたプログラム内で予測不能な GOTO/JUMPTO を使用するには、Debug Tool の SET WARNING OFF コマンドを使用します。	X	X	X
ランタイム・オプション	ABTERMENC および TERMTHDACT - エラー動作を制御できます。	X	X	
	CBLQDA - QSAM ファイルの動的割り振りを制御できます。		X	
	LANGUAGE - ランタイム・エラー・メッセージの言語を変更できます。	X		
	RPTSTG - ストレージ使用報告書を入手できます。	X		
	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できます。	X	X	

表 5. *Enterprise COBOL および Language Environment の利点 (続き)*

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
Enterprise COBOL バージョン 5 のコンパイラー・オプション	Enterprise COBOL バージョン 5 のコンパイラー・オプションおよびサブオプションには多数の変更が行われています。これらの変更の詳細については、 191 ページの『Enterprise COBOL バージョン 5 におけるコンパイラー・オプションの変更』を参照してください。以下のコンパイラー・オプションは、Enterprise COBOL バージョン 5 のプログラムでのみ使用可能です。 • AFP(VOLATILE|NOVOLATILE) • ARCH(*n*) • COPYRIGHT | NOCOPYRIGHT • DISPSIGN(SEP|COMPAT) • HGPR(PRESERVE|NOPRESERVE) • MAXPCF(*n*) • QUALIFY(COMPAT|EXTEND) • SERVICE | NOSERVICE • STGOPT | NOSTGOPT • VLR(COMPAT|STANDARD) • XMLPARSE(XMLSS|COMPAT) • ZONEDATA(PFD|MIG) **注:** – COPYRIGHT、QUALIFY、および SERVICE の各オプションは、Enterprise COBOL V5.2 でのみ使用できます。 – VLR オプションは、Enterprise COBOL V5.1 (サービス PTF 適用済み) および V5.2 で使用できます。 – XMLPARSE オプションは以前に Enterprise COBOL V5.1 で削除されましたが、サービスによって V5.1 に復元され、V5.2 に組み込まれています。	X	X	X

表 5. *Enterprise COBOL および Language Environment の利点 (続き)*

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
Enterprise COBOL バージョン 4 のコンパイラー・オプション	以下のコンパイラー・オプションは、Enterprise COBOL バージョン 4 以降のプログラムでのみ使用可能です。 • XMLPARSE - XML PARSE ステートメントで z/OS XML システム・サービス・パーサーを使用するのか、または既存の COBOL パーサーを使用するのかを制御します。XMLPARSE(COMPAT) オプションでは、XML 構文解析は Enterprise COBOL バージョン 3 と互換性があります。XMLPARSE(XMLSS) オプションでは、z/OS システム・サービス・パーサーが使用され、新規の XML 構文解析機能が使用可能になります。 **注:** XMLPARSE オプションは以前に Enterprise COBOL V5.1 で削除されましたが、サービスによって V5.1 に復元され、V5.2 に組み込まれています。 • OPTFILE - コンパイラー・オプションが SYSOPTF DD ステートメントで指定されたデータ・セットから読み取られるかどうかを制御します。 • SQLCCSID - COBOL と DB2 間のコード化文字セット ID (CCSID) の調整を制御します。 • BLOCK0 - プログラム内のすべての適格な QSAM ファイルに対して暗黙の BLOCK CONTAINS 0 節をアクティブにします。 • MSGEXIT - EXIT コンパイラー・オプションの MSGEXIT サブオプションは、 FIPS (FLAGSTD) メッセージも含めて、コンパイラー・メッセージのカスタマイズ (重大度の変更またはメッセージの抑制) を行う機能を提供します。	X	X	X

表 5. Enterprise COBOL および Language Environment の利点 (続き)

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
Enterprise COBOL バージョン 3 のコンパイラー・オプション	以下のコンパイラー・オプションは、Enterprise COBOL バージョン 3 以降のプログラムでのみ使用可能です。 • CICS - 組み込みの CICS 変換プログラム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、COBOL ソース・プログラムの英数字、国別、および DBCS 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。 • MDECK(COMPILE, NOCOMPILE) - ライブラリー処理の出力をファイルに書き出すどうか、およびライブラリー処理および出力ファイルの生成後にコンパイルを通常に続行させるかどうかを制御します。 • 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 Language Environment 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。	X	X	X
COBOL (OS/390 および VM 版) のコンパイラー・オプション	以下のコンパイラー・オプションは、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。	X	X	

表 5. *Enterprise COBOL および Language Environment の利点 (続き)*

利点	注	利点を備えていない製品		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
COBOL (MVS および VM 版) のコンパイラー・オプション	以下のコンパイラー・オプションは、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) - 新規サブオプション FULL を指定した OPTIMIZE はオブジェクト・プログラムを最適化し、そのランタイム・パフォーマンスは、OS/VS COBOL および VS COBOL II OPTIMIZE 両オプションの場合より改善されます。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/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) - NORENT プログラムを 16MB 境界より上に置くことができます。	X	X	
* 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 DB2 コプロセッサー、組み込みの CICS 変換プログラム、および 31 桁のサポートが追加されました。				

新しいコンパイラーとランタイムに関する変更

Enterprise COBOL を使用すれば、コンパイラー・オプションが廃止されたこと、デフォルト・コンパイラー・オプションが異なること、SOM ベースの OO COBOL がサポートされないこと、DB2 コプロセッサーや CICS 変換プログラムが組み込まれたことなど、いくつかの点で、既存 COBOL アプリケーションの再コンパイルに影響が及びます。以下のトピックで、削除または改善されたエレメント、および互換性を保つために必要な処置について簡単に説明します。

CMPR2 コンパイラー・オプションは利用不能

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 (85 COBOL 標準) に変換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。

- 47 ページの『第 5 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 99 ページの『第 7 章 VS COBOL II ソース・プログラムのアップデート』
- 111 ページの『第 9 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

Enterprise COBOL V5 は FLAGMIG コンパイラー・オプションを提供しません。

Enterprise COBOL V5 への移行を支援するため、Enterprise COBOL V5 への移行に必要となるソース・コード構文関連の変更にフラグを立てる Enterprise COBOL V4.2 における新規オプション FLAGMIG4 が提供されています。

FLAGMIG オプションについて詳しくは、以下を参照してください。

- 47 ページの『第 5 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 99 ページの『第 7 章 VS COBOL II ソース・プログラムのアップデート』
- 111 ページの『第 9 章 IBM COBOL ソース・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 は利用不能

Enterprise COBOL は SOM ベースのオブジェクト指向 COBOL をサポートしませんが、Enterprise COBOL は COBOL プログラムと Java プログラム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 COBOL が削除されたことにより、コンパイラー・オプション TYPECHK および IDLGEN も削除されました。その理由は、これらのオプションでは、SOM を実行する必要があるためです。 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、155 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』を参照してください。

組み込みの DB2 コプロセッサーが利用可能

Enterprise COBOL は組み込みの DB2 コプロセッサ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 SQL 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。分離型の DB2 プリコンパイラーから組み込みの DB2 コプロセッサーに移行することを選択するか、または、分離型の DB2 プリコンパイラーを使用し続けることを選択できます。

SQL ステートメントを含む COBOL ソース・プログラムを DB2 コプロセッサーで処理できるようにするには、SQL 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。

- 229 ページの『第 19 章 DB2 コプロセッサー移行における考慮事項』

組み込みの CICS 変換プログラムが利用可能

Enterprise COBOL は組み込みの CICS 変換プログラム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。分離型の CICS 変換プログラムから組み込みの CICS 変換プログラムに移行することを選択するか、または、分離型の CICS 変換プログラムを使用し続けることを選択できます。

CICS ステートメントを含む COBOL ソース・プログラムを CICS 変換プログラム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。

- 223 ページの『第 18 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

インストール先のプログラミング環境によっては、新しいコンパイラーおよびランタイムに移行するために、1 つ以上の移行作業を実行しなければならない可能性があります。

そのような作業には以下が含まれます。

- 戦略を計画する
- ソースを Enterprise COBOL にアップグレードする
- Enterprise COBOL プログラムを既存アプリケーションに追加する

戦略を計画する

ソース・プログラムを Enterprise COBOL にアップグレードする前に、変換の戦略を作成してください。 Language Environment へのランタイム・ライブラリー移行を行うときの参考として、「*Enterprise COBOL for z/OS Compiler and Runtime Migration Guide Version 4 Release 2*」(http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf) を参照してください。

移行戦略としては、必要に応じて段階的に既存のアプリケーション全体を Enterprise COBOL で再コンパイルします。個々のプログラムを、移行時に個別に再コンパイルすることもできます。

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、これらのプログラムに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) のいずれかでコンパイルされた OS/VS COBOL プログラムには、68 COBOL 標準または 74 COBOL 標準のいずれかのエレメントを組み込むことができます。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、55 ページの『85 COBOL 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、VS COBOL II と Enterprise COBOL バージョン 5 には、以下の言語上の相違点があります。

- CMPR2 サポートの除去
- 一部の SEARCH ALL ステートメントの動作
- 新しい予約語

- 単純化された TEST コンパイラー・オプション
- SIMVRD に対するランタイム・サポートの除去
- 形式 2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS のサポートの廃止。

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、251 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、99 ページの『第 7 章 VS COBOL II ソース・プログラムのアップデート』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 VS COBOL II プログラムと同様に、74 COBOL 標準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 ではサポートされません。このため、VS COBOL II リリース 1 または 2 のすべてのプログラム、および CMPR2 を使用してコンパイルされた VS COBOL II リリース 3 または 4 のすべてのプログラムでは、ソースの移行を行う必要があります。ソース・プログラムを 85 COBOL 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、118 ページの『CMPR2 コンパイラー・オプションから 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、267 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

多くの IBM COBOL プログラムは、変更を行わずに Enterprise COBOL でコンパイルされます。

ただし、以下のプログラムは、Enterprise COBOL でコンパイルを行う前にアップグレードする必要があります。

- 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
- 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 IBM COBOL 拡張機能を持つプログラム
- 形式 2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS (オプション) が含まれるプログラム。

詳細については、111 ページの『第 9 章 IBM COBOL 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。

Enterprise COBOL バージョン 3

多くの Enterprise COBOL バージョン 3 プログラムは、Enterprise COBOL バージョン 5 の下で変更なしでコンパイルされます。

ただし、以下のプログラムは、アップグレードする必要があります。

- 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
- フォーマット-2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS が含まれるプログラム。
- XML PARSE ステートメントを含むプログラム。

詳細については、 163 ページの『第 11 章 Enterprise COBOL バージョン 3 からプログラムのアップグレード』を参照してください。

Enterprise COBOL バージョン 4

多くの Enterprise COBOL バージョン 4 プログラムは、Enterprise COBOL バージョン 5 の下で変更なしでコンパイルされます。

ただし、以下のプログラムは、アップグレードする必要があります。

- 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
- フォーマット-2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS が含まれるプログラム。
- XML PARSE ステートメントを含み、XMLPARSE(COMPAT) コンパイラー・オプションでコンパイルされたプログラム。

詳細については、 177 ページの『第 13 章 Enterprise COBOL バージョン 4 からのアップグレード』を参照してください。

Enterprise COBOL プログラムの既存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 Language Environment のもとで実行することができます。

注: この移行ガイドは、LE より前のランタイム・ライブラリーから Language Environment へのランタイム移行を完了してある場合にのみ使用してください。COBOL ランタイム移行を完了するためには、以下の条件が満たされている必要があります。

- 言語環境プログラムのデータ・セット SCEERUN が、LNKLST または LPALST にインストールされている。
- LNKLST または LPALST 内に COBLIB、VSCLLIB、および COB2LIB のインスタンスがない。
- バッチ・ジョブ内の JCL STEPLIB/JOBLIB ステートメントや、CICS 始動 JCL に、COBLIB、VSCLLIB、COB2LIB のインスタンスがない。
- NORES でコンパイルされたプログラム用の、静的にバインドされたランタイム・ライブラリー・ルーチンがすべて、言語環境プログラムにあるルーチンで置き換えられている。
- RES でコンパイルされた VS COBOL II プログラム用の IGZEBST ブートストラップ・モジュールが、APAR PN74000 が適用された VS COBOL II ランタイム・バージョンの IGZEBST か、または言語環境プログラムにある IGZEBST で置き換えられた IGZEBST のいずれかにリンクしていた。CICS プログラムで

は、COBOL V5 プログラムをアプリケーションに導入する場合、APAR PI33330 の PTF がインストールされた LE から IGZEBST を使用して再リンクすることが必要になります。

•

これらの手順が完了していない場合は、ここでの手順に従う前に、最初に「*Enterprise COBOL for z/OS コンパイラーおよびランタイム移行ガイド バージョン 4 リリース 2*」に記載されたランタイム移行作業をすべて完了しておいてください。

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、以下の項目について知っておく必要があります。

- プログラムを特定の古い COBOL プログラムと一緒に実行するときの制限
- 16-MB 境界の上下での WORKING-STORAGE の獲得
- コンパイラー・オプション変更の効果
- 予約語の変更
- 他の動作における Enterprise COBOL V5 との違い

詳細については、 209 ページの『第 16 章 Enterprise COBOL バージョン 5 プログラムを既存 COBOL アプリケーションに追加する』を参照してください。

制約事項: Enterprise COBOL バージョン 5 プログラムは、以下のプログラムと混用できません。

- OS/VS COBOL プログラム。このプログラムは、Enterprise COBOL に移行する必要があります。
- VS COBOL II NORES プログラム。このプログラムは、Enterprise COBOL に移行する必要があります。

第 2 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 (現在、 IBM Enterprise COBOL for z/OS のみがサポートされています) 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 (Language Environment) を使用して実行することが理想的です。以下の 2 段階でプログラムを徐々に移行します。

- 段階 1: ランタイム・マイグレーション。ランタイム・ライブラリー移行の実行に役立つ説明として、http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf にある「*Enterprise COBOL for z/OS コンパイラーおよびランタイム 移行ガイド バージョン 4 リリース 2*」を使用することができます。
- 段階 2: コンパイラーの移行 (既存アプリケーションのうちのプログラムを 1 つのみ、または複数コンパイルできます)。

このセクションの残りの部分では、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

移行プロセスには、コンパイラー移行 (ソース・プログラムを新しいコンパイラーで再コンパイル) が含まれ、またランタイム移行 (アプリケーションを新しいランタイム・ライブラリーに移動) も含まれる場合があります。マイグレーション作業の一部として、目録の評価およびテストを行うことも必要になります。前述のとおり、再コンパイルとランタイム移行を同時に行う必要はありません。

マイグレーション作業の詳細については、 17 ページの『一般的な移行作業』を参照してください。

ランタイムのマイグレーション

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・ルーチンを必要とします。旧コンパイラー OS/VS COBOL および VS COBOL II では、ランタイム・ルーチンを静的にロード・モジュールにリンクするオプション (NORES コンパイラー・オプション) または実行時に動的にアクセスするオプション (RES コンパイラー・オプション) がありました。 1991 年の COBOL/370 V1 以降、すべての COBOL コンパイラーで、デフォルトは RES 動作に設定されています。

Language Environment 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 Language Environment 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、Language Environment を使用してリンク・エディットするだけです。

注: VS COBOL II による IGZEBST ブートストラップ・ルーチンに PN74000 がインストールされている場合、その IGZEBST を言語環境プログラム・バージョンの IGZEBST で置き換える (REPLACE) 必要はありません。

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、LNKLST、LPALST、JOBLIB、または STEPLIB を使用して、ランタイムに OS/VS COBOL または VS COBOL II ライブラリー・ルーチンの代わりに Language Environment ライブラリー・ルーチンを使用可能にしてください。

ランタイム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ないでください。例えば、LNKLST には、COBOL ランタイム・ライブラリーが 1 つ (Language Environment 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・ライブラリーの移行がまだ完了していない場合、本書を使用する前にその移行を完了する必要があります。ランタイム・ライブラリー移行の実行に役立つ説明として、http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf にある「*Enterprise COBOL for z/OS コンパイラーおよびランタイム 移行ガイド バージョン 4 リリース 2*」を使用することができます。

コンパイラー移行

コンパイラー・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 Language Environment 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。

ほとんどのプログラムの場合、Enterprise COBOL バージョン 5 での再コンパイル時にソース・コード変更は不要です。 Enterprise COBOL バージョン 5 への移行時に、各アプリケーション内のすべてのプログラムを再コンパイルすることが推奨されますが、これは必須ではありません。 OS/VS COBOL でコンパイルされたプログラム、またはそれ以降のコンパイラーで旧 CMPR2 コンパイラー・オプションを使用してコンパイルされたプログラムには、ソース・コード変更が必要になります。

Enterprise COBOL バージョン 5 プログラムから呼び出される (またはそのプログラムを呼び出す必要がある) 場合、OS/VS COBOL プログラムおよび VS COBOL II NORES プログラムにはコンパイラー移行および再コンパイルが必要です。Enterprise COBOL V5 プログラムは、VS COBOL II RES プログラムを動的に呼び出すことができます (また、VS COBOL II RES プログラムから動的に呼び出されることもあります)。

コンパイラー移行は通常、使用するソース言語レベルのアップグレード (OS/VS COBOL でサポートされる 74 Standard COBOL から Enterprise COBOL でサポートされる 85 Standard COBOL など) から構成されます。また、アプリケーションを Language Environment のもとで実行できるようにするためにコンパイラー・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、 267 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

IBM は、Language Environment のもとで実行される OS/VS COBOL および VS COBOL II プログラムを、一部の場合に引き続きサポートします。

OS/VS COBOL リリース 2 および VS COBOL II リリース 3 以上のコンパイラーでコンパイルされたプログラムが Language Environment ランタイム・ライブラリー・バージョンの COBOL ライブラリー・ルーチンを使用する場合、IBM は、引き続きプログラム実行のためのサービス・サポートを提供します。ただし、以下の例外があります。

- CICS Transaction Server の下で実行される OS/VS COBOL プログラム
- Enterprise COBOL V5 プログラムと相互運用される OS/VS COBOL プログラム
- Enterprise COBOL V5 プログラムと相互運用される VS COBOL II プログラムのうち、NORES オプションでコンパイルされたもの

例えば、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 Language Environment ランタイム・ライブラリーに存在します。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。Language Environment ランタイム・ライブラリーのサポート対象リリースを使用して実行されている OS/VS COBOL プログラムは、IBM サービスではサポートされますが、Enterprise COBOL V5 プログラムとの相互運用はできません。

CICS TS (Transaction Server) では、 OS/VS COBOL プログラムを実行できなくなりました。

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、このコンパイラーで生成されたプログラムは、Language Environment の下で実行されていて Enterprise COBOL V5 と相互運用されていなければ、サポートされます。ランタイム・ライブラリーを Language Environment に移行した後で、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・コードを実行してから、Enterprise COBOL コンパイラーを使用してコンパイルを行うことができます。

CCCA の詳細については、 267 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。

古いレベルの IBM COBOL プログラムとのインターオペラビリティー

Enterprise COBOL V5 のプログラムが、以前のバージョンの COBOL でコンパイルされたプログラムを呼び出したり、以前のバージョンの COBOL でコンパイルされたプログラムから呼び出されたりする (すなわち、V5 のプログラムが、以前のバージョンの COBOL でコンパイルされたプログラムと相互運用される) 場合に、いくつかの制約事項があります。

Enterprise COBOL V5 プログラムは、単一のアプリケーションにおいて OS/VS COBOL プログラムや VS COBOL II NORES プログラムと相互運用できません。Enterprise COBOL V5 でコンパイルされたプログラムを含む COBOL 実行単位 (言語環境プログラム・エンクレーブ) に OS/VS COBOL プログラムや VS COBOL II NORES プログラムを含めることはできません。

注: Enterprise COBOL V4 以前のバージョンでコンパイルされた COBOL プログラムのみを含む実行単位は、OS/VS COBOL プログラムや VS COBOL II NORES プログラムと相互運用できます。

Enterprise COBOL V5 でコンパイルされたプログラムは、以下の条件および呼び出しタイプに基づいて、VS COBOL II 以降でコンパイルされたプログラムと相互運用できます。

- 静的呼び出し。 Enterprise COBOL V5 でコンパイルされたプログラムは、単一のプログラム・オブジェクトを形成するために、以下のオブジェクト・モジュールまたはプログラムとバインド (リンク・エディット) したりできます。プログラム・オブジェクト内のプログラムは、相互に静的呼び出しを指定することができます。
  - RES コンパイラー・オプションを指定して VS COBOL II でコンパイルされたプログラム
  - VS COBOL II より後の任意の IBM COBOL コンパイラー・バージョンでコンパイルされたプログラム
  - Enterprise COBOL V3 または V4 でコンパイルされたプログラム

  注: NORES コンパイラー・オプションを指定して VS COBOL II でコンパイルされたプログラムは、Enterprise COBOL V5 でコンパイルされたプログラムと相互運用できません。
- 動的呼び出し。 RES オプションを指定して VS COBOL II でコンパイルされたプログラム、または VS COBOL II 以降のバージョンの COBOL でコンパイルされたプログラムを含むプログラム・モジュールも、動的 CALL ステートメントを使用して Enterprise COBOL V5 プログラム・オブジェクトと相互運用できます。
- DLL 呼び出し。 DLL リンケージをサポートしていた前のバージョンの COBOL でコンパイルされたプログラム・モジュールは、DLL リンケージを使用して Enterprise COBOL V5 プログラム・オブジェクトと相互運用できます。

第 2 部 移行戦略

第 3 章 コンパイラーのアップグレード・チェックリスト

プログラムを Enterprise COBOL にアップグレードするには、次のチェックリストを使用します。

以下の作業を実行してください。

1. ご使用の COBOL ロード・ライブラリーが PDS データ・セットにあれば、それらを PDSE データ・セットに移行してください。
2. ランタイム移行を完了します。これは、以下の状態を意味します。
   - 言語環境プログラムのデータ・セット SCEERUN が、LNKLST または LPALST にインストールされている。
   - LNKLST または LPALST 内に COBLIB、VSCLLIB、および COB2LIB のインスタンスがない。
   - バッチ・ジョブ内の JCL STEPLIB/JOBLIB ステートメントや、CICS 始動 JCL に、COBLIB、VSCLLIB、COB2LIB のインスタンスがない。
   - NORES でコンパイルされたプログラム用の、静的にバインドされたランタイム・ライブラリー・ルーチンがすべて、言語環境プログラムにあるルーチンで置き換えられている。
   - RES でコンパイルされた VS COBOL II プログラム用の IGZEBST ブートストラップ・モジュールが、APAR PN74000 が適用された VS COBOL II ランタイム・バージョンの IGZEBST か、または言語環境プログラムにある IGZEBST で置き換えられた IGZEBST のいずれかにリンクしていた。CICS プログラムでは、COBOL V5 プログラムをアプリケーションに導入する場合、APAR PI33330 の PTF がインストールされた LE から IGZEBST を使用して再リンクすることが必要になります。
3. Enterprise COBOL の「*Licensed Program Specifications*」で定義されているソフトウェア前提条件およびハードウェア前提条件がすべて満たされていることを確認します。(「*Licensed Program Specifications*」は、「Enterprise COBOL for z/OS library」(http://www.ibm.com/support/docview.wss?uid=swg27036733) から入手してください。)
4. すべての実動システムを含め、COBOL プログラムがコンパイルまたは実行される可能性があるすべてのシステムに、Language Environment ランタイム・ライブラリー用の前提条件 PTF をインストールします。
5. COBOL が実行されるすべてのシステム、および COBOL と連動する必要のあるすべてのソフトウェア (z/OS、Debug Tool、Fault Analyzer、DB2 など) で、新しい COBOL コンパイラーでコンパイルされたプログラムを使用する準備ができていることを確認します。APAR のリストについては、 189 ページの『Enterprise COBOL バージョン 5 の前提ソフトウェアおよび前提サービス』を参照してください。
6. 古い COBOL コンパイラーを緊急用に保存します。
7. 新しい Enterprise COBOL コンパイラーを購入し、インストールします。
8. 新しいコンパイラーと古いコンパイラーの互換性がとられるように、デフォルトのコンパイラー・オプションおよびライブラリー制御システム・オプション

をセットアップします。将来再利用できるように、行ったカスタマイズおよびセットアップの内容をすべて文書に保存します。

9. 移行元の COBOL コンパイラーによっては、COBOL ソース・コードの変更が必要になることがあります。詳しくは、現在ご使用のコンパイラーに該当する本書の『*プログラムのアップグレード*』 セクションのトピックを参照してください。
10. Enterprise COBOL V5 に推奨される移行戦略は、各アプリケーション (プログラムのグループ) を COBOL V5 でコンパイルし、そのアプリケーションと、現行形式の同じアプリケーション (Enterprise COBOL V4 以前のコンパイラーでコンパイルされたアプリケーション) に対してリグレッション・テストを行います。旧コンパイラーと同じ結果が新しいコンパイラーで得られることが分かったら、アプリケーションを実動に移行します。
11. すべてのプログラムを新しいコンパイラーでコンパイルした後で、古いコンパイラーをアンインストールします。この方法により、ライセンス料を節約します。

第 4 章 ソース・プログラムのアップグレードの計画

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に従うことができます。

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. アプリケーションの優先順位付け
4. 移行手順の設定
5. 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

コンパイラーがまだインストールされていない場合、コンパイラーをインストールしてください。「*Program Directory for Enterprise COBOL*」を参照してください。必ず、Enterprise COBOL バージョン 5 でのデフォルト・コンパイラー・オプションを、旧コンパイラーで使用していたものと同じインストール・デフォルトに設定してください。

ストレージ要件の評価

Enterprise COBOL コンパイラーの大部分は、16 MB 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、16 MB 境界より上に常駐することができます。これにより、16 MB 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 V5 でのコンパイル時にソース・コードから生成されるプログラム・オブジェクトは、すべての旧バージョンの COBOL でのコンパイル時に生成されたロード・モジュールよりも大きくなります。

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、CMPR2 を使用する古い COBOL プログラム (OS/VS COBOL、VS COBOL II、または IBM COBOL のいずれか) を、Enterprise COBOL でコンパイルできる85 COBOL 標準コードに自動的に変換します。また、変更されたステートメントのレポートも提供されます。 CCCA は、IBM Debug Tool 製品に組み込まれています。

CCCA の詳細については、 267 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 MIGR コンパイラー・オプション
MIGR オプションは、Enterprise COBOL でコンパイルするためには変換が必要なソース・ステートメントを識別します。

CMPR2、FLAGMIG、および NOCOMPILE コンパイラー・オプション

COBOL CMPR2、FLAGMIG、および NOCOMPILE オプションは、Enterprise COBOL でコンパイルするためには変換が必要なソース・ステートメントを識別します。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。

Enterprise COBOL V4.2 FLAGMIG4 コンパイラー・オプション
Enterprise COBOL V5 への移行に役立つよう、新しいコンパイラー・オプション FLAGMIG4 が、Enterprise COBOL V4.2 用の APAR PM93450 に用意されています。 Enterprise COBOL V4 プログラムに含まれている言語エレメントがサポートされなくなったものであったり、Enterprise COBOL V5 では異なる意味でサポートされるものであったりする場合に、FLAGMIG4 オプションは、そのような言語エレメントを特定します。コンパイラーは、そのようなすべての言語エレメントに対して警告診断メッセージを生成します。

使用をお勧めする移行ツールはほかに、COBOL 報告書作成プログラム・プリコンパイラーがあります。これによって、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。

これらの移行ツールについては、 267 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 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、Language Environment、および Debug Tool 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。

Standard COBOL 68、Standard COBOL 74、および Standard COBOL 85 間のソース言語の相違点に加え、プログラマーは、Language Environment 条件処理および Language Environment 呼び出し可能サービスについて習熟することが必要になります。

IBM を通じて利用できる Enterprise COBOL および Language Environment の教育については、営業担当員にお尋ねください。Language Environment の資料またはテクニカル・コンファレンス (SHARE、www.share.org 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。

Debug Tool ロード・モジュール・アナライザーは、プログラム・オブジェクト内の各オブジェクトに使用された言語変換プログラムを判別できます。詳しくは、 275 ページの『Debug Tool ロード・モジュール・アナライザー』を参照してください。

Edge Portfolio Analyzer は、使用されたコンパイラー、コンパイラーのリリース、およびコンパイラー・オプションを報告することによって、既存プログラム・オブジェクトの目録の作成を支援します。詳しくは、 275 ページの『Edge Portfolio Analyzer』を参照してください。

言語環境プログラムは、ご使用のインベントリーにある OS/VS COBOL プログラムを常に実行しているかどうかを調べるために役立ちます。ご使用の言語環境プログラムに APAR PM86742 用の修正をインストールし、実行時に検出された OS/VS COBOL プログラムに関する警告メッセージがないかを調べてください。

IGZ0268W

OS/VS COBOL プログラム「program-name」の呼び出しが行われました (An invocation was made of OS/VS COBOL program "program-name")

IGZ0269W

「program-lang」バージョン「program-version」プログラム「program-name」が OS/VS COBOL プログラム「program-name」への呼び出しを行いました ("program-lang" version "program-version" program "program-name" made a call to OS/VS COBOL program "program-name")

Rational Asset Analyzer for z/OS を使用すれば、アプリケーションのコード変更による影響を分析できます。詳しくは、272 ページの『Rational 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知っておく必要があります。以下の項目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 C85 COBOL 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 85 COBOL 標準言語で書かれていること。
- サード・パーティー・ツール (デバッガーやデータベースなど) が Enterprise COBOL をサポートしていること。

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。

すべての旧バージョンの 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、Enterprise COBOL V3、Enterprise COBOL V4)
- 使用したコンパイラー・オプション (特に CMPR2、NORES、XMLPARSE)
- 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
- 呼び出し側プログラム
- 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
- SIMVRDS、SOM ベース OO、2000 年言語拡張、または LABEL 宣言の使用

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、以下のように移行処置を優先順位付けできるようになりました。

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。 34 ページの表 6 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。

複雑度	要件
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	大部分のコードが CCCA によってエラーなしで移行される 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。

34 ページの表 6 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 6. 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 から 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
74 COBOL 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+		
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		
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[3]	10	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

表 6. 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
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
SIMVRD		3
SOM ベースの OO		8
LABEL 宣言		3
XMLPARSE(COMPAT)		7

1. 詳細については、 313 ページの『付録 G. 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. z/OS 1.7 では ISAM によるサポートはありません。
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 7 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 7. 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。

表 7. プログラム移行優先順位の割り当て (続き)

移行優先順位	複雑度	その他の考慮事項
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、以下の状態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 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。

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。

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、Language Environment 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、Language Environment 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。 209 ページの『第 16 章 Enterprise COBOL バージョン 5 プログラムを既存 COBOL アプリケーションに追加する』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、次のフローチャートに示したステップに従ってください。

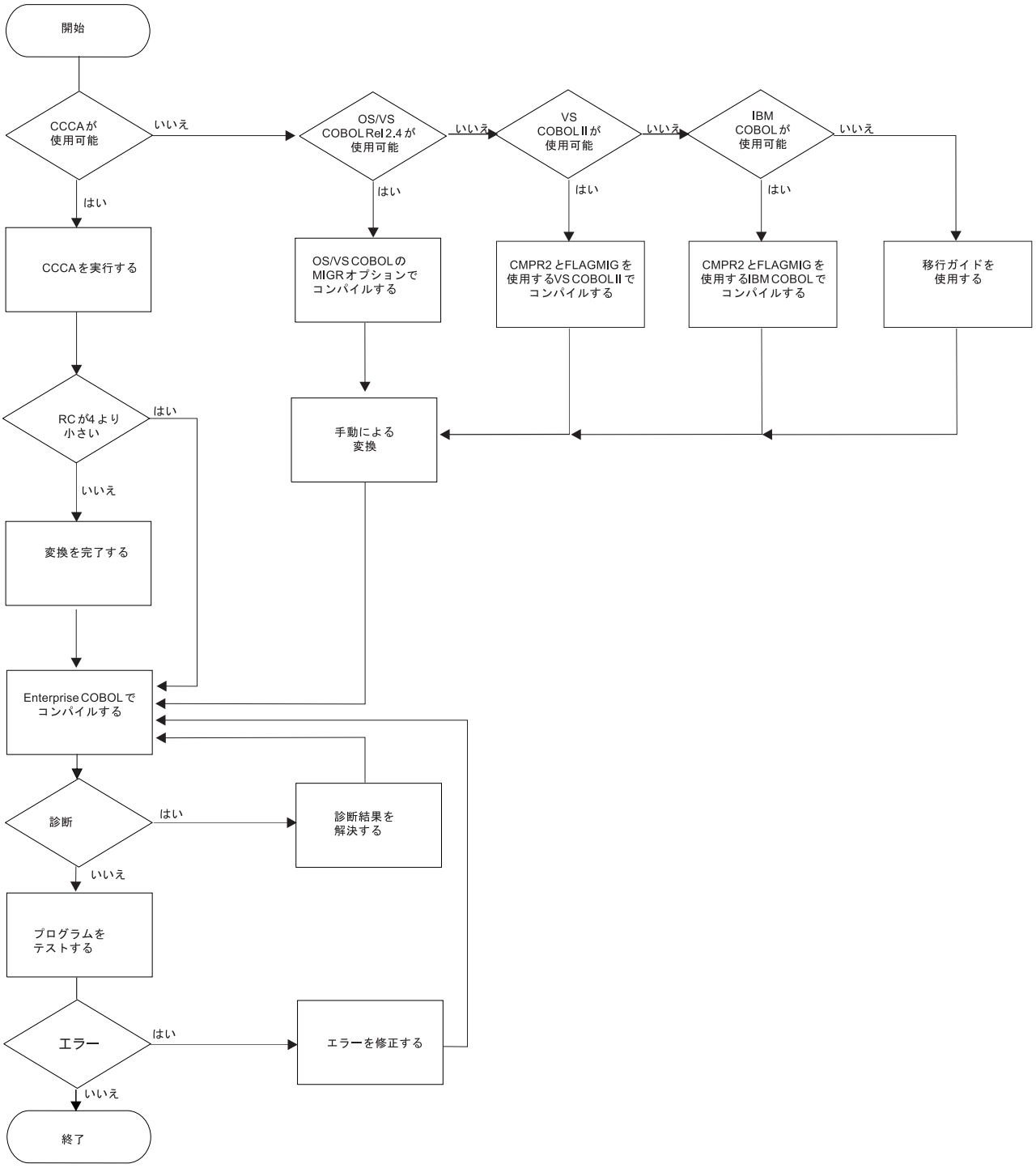

図 1. OS/VS COBOL プログラムを Enterprise COBOL プログラムに変換する手順

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、以下のフローチャートに示したステップに従ってください。

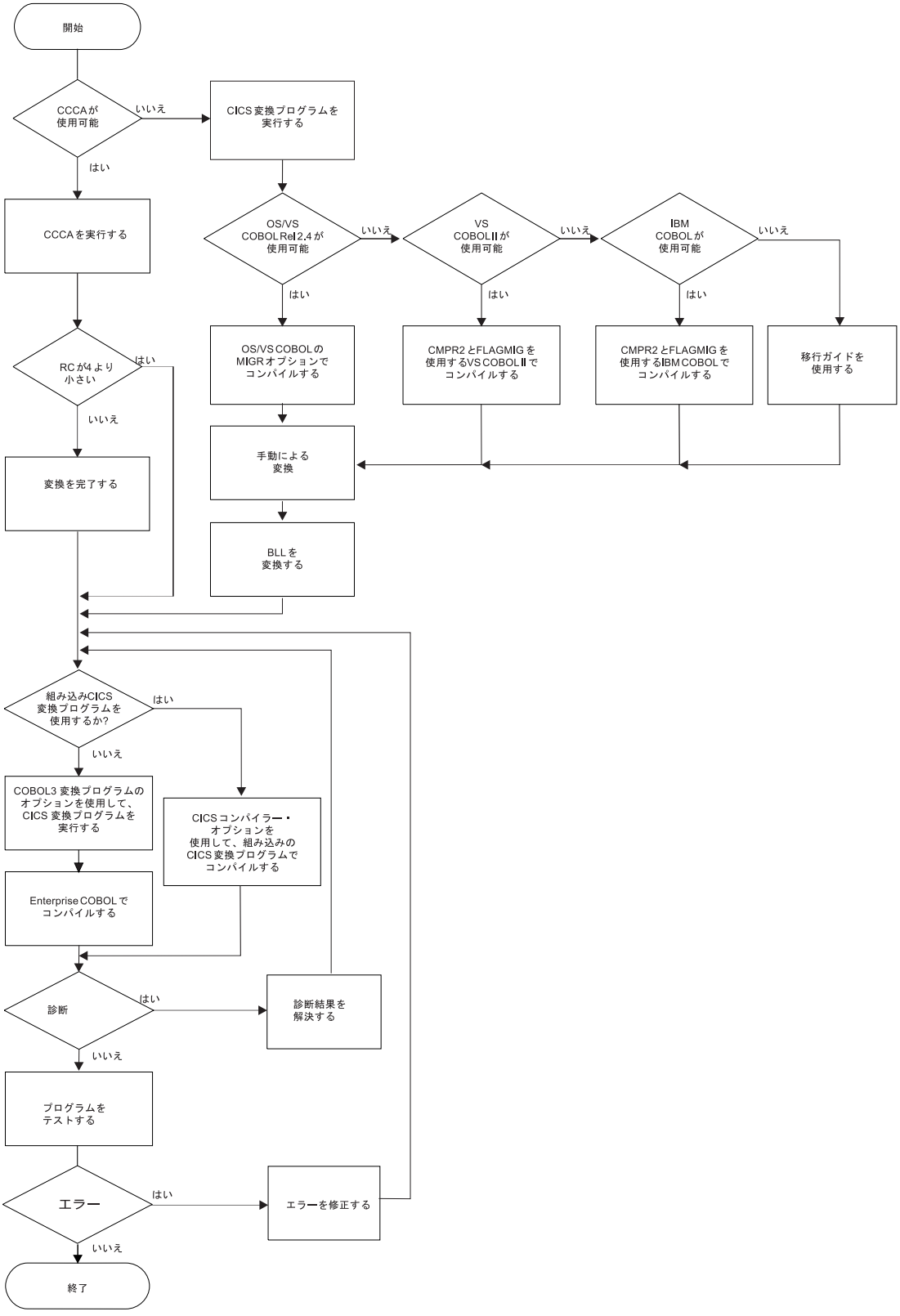

図 2. CICS コマンドを含む OS/VS COBOL プログラムを変換する手順

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

報告書作成プログラム・ステートメントが含まれている OS/VS COBOL プログラムを Enterprise COBOL に変換し、報告書作成プログラム・ステートメントを削除するには、以下のフローチャートに示されている手順を実行します。

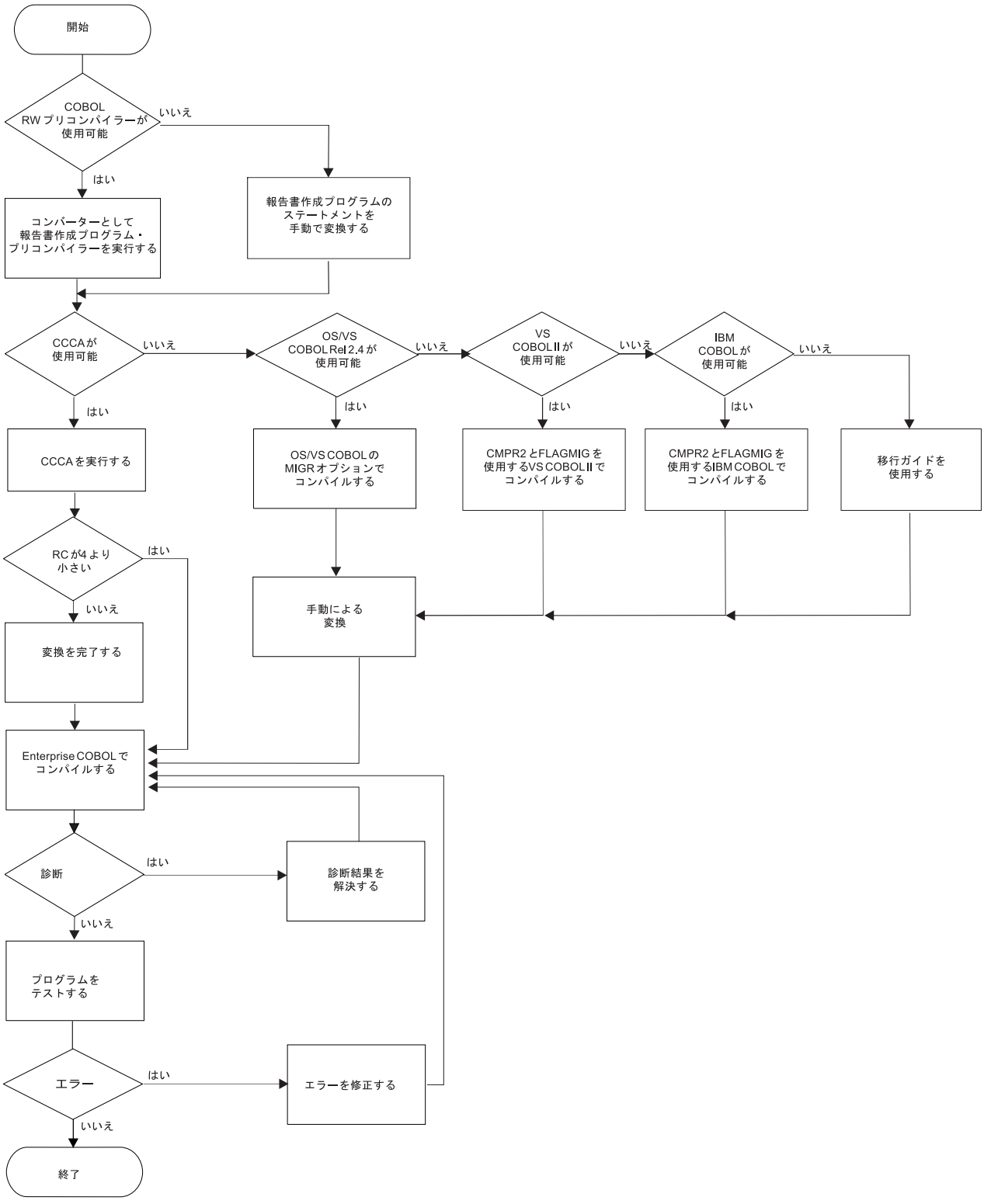

図 3. OS/VS COBOL プログラムに変換して、報告書作成プログラム・ステートメントを破棄する手順

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

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、以下のフローチャートに示したステップに従ってください。

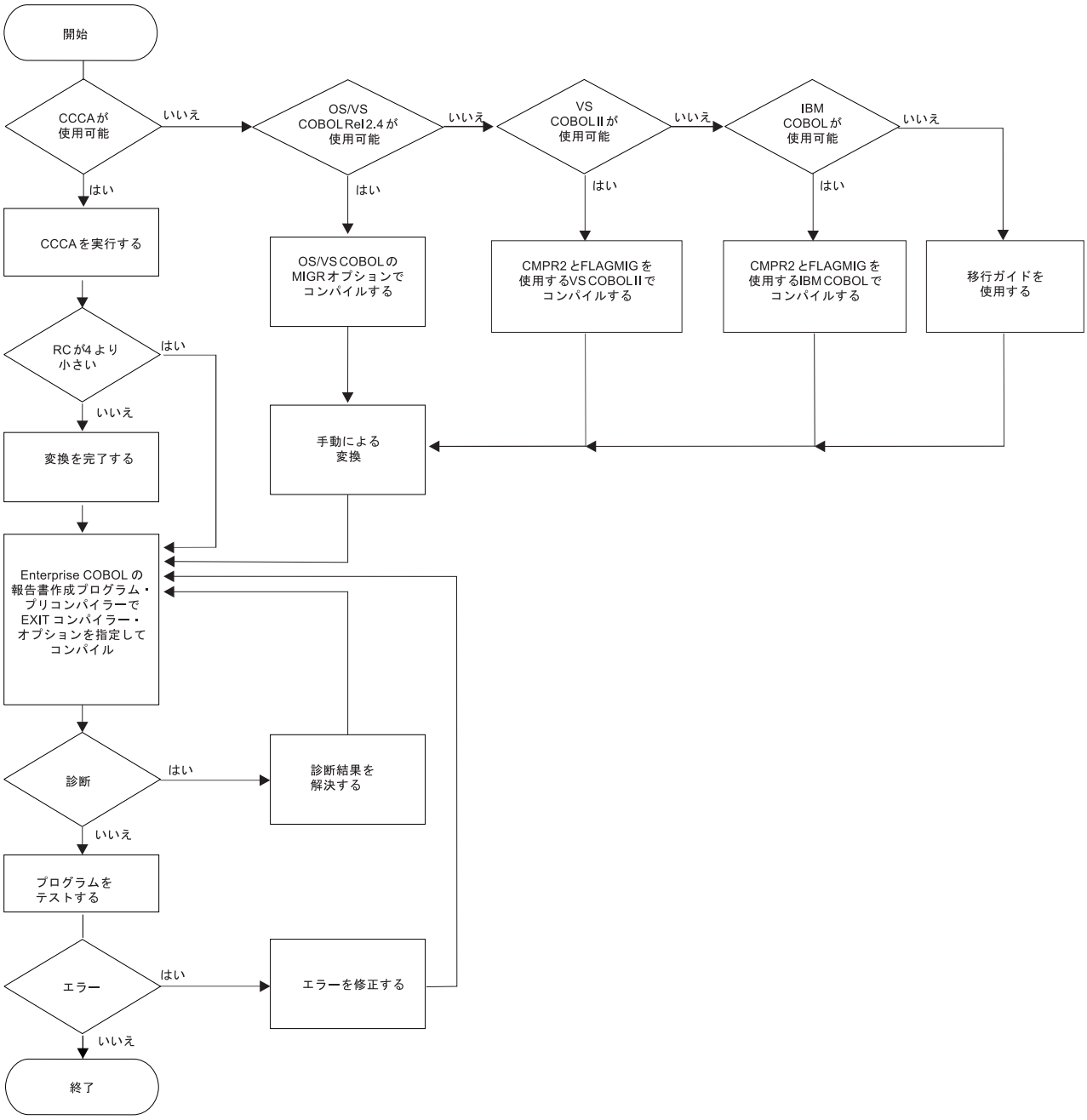

図 4. OS/VS COBOL プログラムに変換して、報告書作成プログラム・ステートメントを保持する手順

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。

既存のソースをバックアップとして保管してください (バックアップは、変換されたモジュールに問題がある場合、比較のベンチマークになり、リカバリーの元バージョンになります)。

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、 267 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、バインド (リンク・エディット)、および実行。

   アップグレードの際に、アプリケーション内のプログラムをすべて再コンパイルすることをお勧めします。これにより、発生する可能性のある問題がすべてなくなり、Enterprise COBOL V5 の最大のパフォーマンス上の利点が得られます。

   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
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、Debug Tool または Language Environment 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. 変換されたプログラムのテスト。

   アプリケーションの新たに再コンパイルされたバージョンの結果を既存のバージョンと比較し、結果が同一であることを確認します。一部のお客様は、プログラムの出力を比較するほか、Debug Tool のコード・カバレッジ機能を使用して、COBOL V5 でのプログラムの動作が旧 COBOL コンパイラーと同じになるようにしていました。

   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、 313 ページの『付録 G. 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。
   - 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関連させて

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。(これは、Language Environment への移行が完了していることを前提としています。)

   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 のもとでは、新バージョンの代わりに旧バージョンを最新の生産性チェックポイントから実行します。
   - 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 プログラムのアップグレード

第 5 章 OS/VS COBOL ソース・プログラムのアップグレード

OS/VS COBOL 言語と Enterprise COBOL 言語の間には、プログラムのアップグレードが必要となる可能性がある相違があります。

本書は、Enterprise COBOL へのアップグレードに適したアプリケーションを言語の観点から評価するために役立ちます。

このセクションにリストされている特定のトピックの他に、テープ・ユーザー Label サポートも変更されています。形式 2 宣言構文 USE...AFTER...LABEL PROCEDURE...、および構文 GO TO MORE-LABELS (オプション) のサポートが Enterprise COBOL V5 で廃止されました。

また、251 ページの『付録 B. COBOL 予約語の比較』で説明されている予約語の変更点についても考慮してください。

Enterprise COBOL では、85 COBOL 標準がサポートされます。 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 85 COBOL 標準プログラムに変換する必要があります。

このセクションは、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、SC43-0800*

ヒント:

1. *VS COBOL for OS/VS Reference* は、現在 IBM では提供していません。
2. CICS に関しては、特に考慮すべきことがあります。 OS/VS COBOL プログラムは、CICS のもとでは実行できなくなりました。CICS のもとで実行するすべての OS/VS プログラムは、Enterprise COBOL にアップグレードする必要があります。
3. 以下のセクションでは、68 COBOL 標準に対する言及はいずれも、IBM Full American National Standard COBOL バージョン 4 (プログラム 5734-CB2)、または OS/VS COBOL (プログラム 5740-CB1) の LANGLVL(1) でサポートされている COBOL 言語に対する言及です。
4. この「*移行ガイド*」全体に記載されている OS/VS COBOL に関する情報は、最新のサービス更新が適用された OS/VS COBOL リリース 2.4 に適用されます。

OS/VS COBOL と Enterprise COBOL の比較

OS/VS COBOL では、68 COBOL 標準 (LANGLVL(1)) および 74 COBOL 標準 (LANGLVL(2)) がサポートされていました。 Enterprise COBOL では 85 COBOL 標準がサポートされています。 74 COBOL 標準と Enterprise COBOL との間の言語の相違点のほか、ご使用の OS/VS COBOL プログラムには、文書化されていない OS/VS COBOL 拡張が含まれて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 8 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、このセクション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
- 文書化されていないサポートされない OS/VS COBOL 拡張

表 8. 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		66 ページの『簡略複合比較条件および括弧の使用』
ACCEPT ステートメント		67 ページの『ACCEPT ステートメント』
ALPHABETIC クラスの変更	CCCA	76 ページの『ALPHABETIC クラスの変更』
ALPHABET 文節の変更 - ALPHABET キーワード	CCCA	76 ページの『ALPHABET-NAME 節の変更: ALPHABET キーワード』
区域 A、ピリオド	CCCA	71 ページの『区域 A におけるピリオド』
算術ステートメントの変更		76 ページの『算術ステートメントの変更』
ASSIGN . . . OR	CCCA	59 ページの『ASSIGN . . . OR』
ASSIGN TO *integer system-name*	CCCA	59 ページの『ASSIGN . . . OR』
ASSIGN . . . FOR MULTIPLE REEL /UNIT	CCCA	59 ページの『ASSIGN . . . FOR MULTIPLE REEL/UNIT 』
ASSIGN 文節の変更 - *assignment-name* 形式	CCCA	76 ページの『ASSIGN 文節の変更』
PICTURE 文節内の B 記号 - 評価の変更		77 ページの『PICTURE 文節内の B 記号 : 評価の変更』
BDAM ファイル処理	CCCA*	58
BLANK WHEN ZERO 文節およびアスタリスク (*) のオーバーライド		67 ページの『BLANK WHEN ZERO 文節およびアスタリスク (*) のオーバーライド』

表 8. *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
CALL identifier ステートメント - PICTURE 文節内の B 記号		77 ページの『PICTURE 文節内の B 記号：評価の変更』
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		77 ページの『CALL ステートメントの変更』
CANCEL ステートメント - PICTURE 文節内の B 記号		77 ページの『PICTURE 文節内の B 記号：評価の変更』
CLOSE . . . FOR REMOVAL ステートメント		67 ページの『CLOSE . . . FOR REMOVAL ステートメント』
CLOSE ステートメント - WITH POSITIONING 句および DISP 句	CCCA	59 ページの『CLOSE ステートメント：WITH POSITIONING 句および DISP 句』
簡略複合比較条件の変更	CCCA	78 ページの『簡略複合比較条件の変更』
グループと数値パック 10 進項目の比較		67 ページの『グループと数値パック 10 進項目の比較』
関連した名前を指定した COPY ステートメント	CCCA	79 ページの『関連した名前を指定した COPY ステートメント』
通信機能		59
CURRENCY-SIGN 文節の変更 - 「/」、「=」、および「L」文字		80 ページの『CURRENCY-SIGN 文節の変更：「/」、「=」、および「L」文字』
CURRENT-DATE 特殊レジスター	CCCA	60 ページの『CURRENT-DATE 特殊レジスター』
DIVIDE . . . ON SIZE ERROR - 中間結果の変更		86 ページの『ON SIZE ERROR 句：中間結果の変更』
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		80 ページの『ENTRY ポイントへの動的 CALL ステートメント』
EXAMINE ステートメント	CCCA	60 ページの『EXAMINE ステートメント』
EXHIBIT ステートメント	CCCA	61 ページの『EXHIBIT NAMED に対する修正処置』
EXIT PROGRAM/GOBACK ステートメントの変更		80 ページの『EXIT PROGRAM/ GOBACK ステートメントの変更』
FILE STATUS 文節の変更	CCCA	80 ページの『FILE STATUS 文節の変更』

表 8. *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
FILE-CONTROL 段落の FILE-LIMIT 文節	CCCA	62 ページの『FILE-CONTROL 段落の FILE-LIMIT 文節』
制御のフロー (終了ステートメントなしの場合)		68 ページの『制御のフロー (終了ステートメントなしの場合)』
FOR MULTIPLE REEL/UNIT	CCCA	59 ページの『ASSIGN . . . FOR MULTIPLE REEL/UNIT 』
USE AFTER STANDARD ERROR 宣言の GIVING 句	CCCA	62 ページの『USE AFTER STANDARD ERROR 宣言の GIVING 句』
IF . . . OTHERWISE ステートメントの変更	CCCA	83 ページの『IF . . . OTHERWISE ステートメントの変更』
索引名 - 固有でない		68 ページの『索引名』
INSPECT ステートメント - PROGRAM COLLATING SEQUENCE 文節		87 ページの『PROGRAM COLLATING SEQUENCE 文節の変更』
オプション・ワードとしての IS		86 ページの『オプション・ワード IS 』
ISAM ファイル処理	CCCA	58
JUSTIFIED 文節の変更	CCCA	83 ページの『JUSTIFIED 文節の変更』
TOTALING/TOTALED AREA を持つ LABEL RECORDS 文節	CCCA	62 ページの『TOTALING/ TOTALED AREA 句を持つ LABEL RECORDS 文節』
LABEL RECORD IS ステートメント		68 ページの『LABEL RECORD IS ステートメント』
MOVE ステートメント - バイナリー値および DISPLAY 値		68 ページの『MOVE ステートメント - バイナリー値および DISPLAY 値』
MOVE ステートメントおよび比較 - 位取りの変更		84 ページの『MOVE ステートメントおよび比較 : 位取りの変更』
MOVE CORRESPONDING ステートメント	CCCA	69 ページの『MOVE CORRESPONDING ステートメント』
MOVE ステートメント - 複数の TO 指定		69 ページの『MOVE ステートメント - 複数の TO 指定』

表 8. *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
MOVE ALL—TO PIC 99		70 ページの『MOVE ALL - TO PIC 99』
MOVE ステートメント - 数値切り捨ての警告メッセージ		70 ページの『MOVE ステートメント - 数値切り捨ての警告メッセージ』
MULTIPLY ... ON SIZE ERROR - 中間結果の変更		86 ページの『ON SIZE ERROR 句 : 中間結果の変更』
固有でない Program-ID 名	CCCA	72 ページの『固有でない PROGRAM-ID 名』
NOTE ステートメント	CCCA	62 ページの『NOTE ステートメント』
グループ項目に対する数値クラス・テスト		84 ページの『グループ項目に対する数値クラス・テスト』
数値データの変更		84 ページの『数値データの変更』
数字編集の変更 (PICTURE 文節)		71 ページの『PICTURE ストリング』
OCCURS 文節 (句の順序)		70 ページの『OCCURS 文節』
OCCURS DEPENDING ON— ASCENDING 句および DESCENDING KEY 句		84 ページの『OCCURS DEPENDING ON 文節 : ASCENDING および DESCENDING KEY 句』
OCCURS DEPENDING ON - 受け取り項目の値の変更	CCCA	85 ページの『OCCURS DEPENDING ON 文節 : 受け取り項目の値の変更』
ON ステートメント	CCCA	62 ページの『ON ステートメント』
ON SIZE ERROR 句 - 中間結果の変更		86 ページの『ON SIZE ERROR 句 : 中間結果の変更』
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)		63 ページの『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)』
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)		63 ページの『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)』

表 8. 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
LEAVE、REREAD、および DISP 句を指定した OPEN ステートメント	CCCA	63 ページの『LEAVE、REREAD、および DISP 句を指定した OPEN ステートメント』
OPEN REVERSED ステートメント		70 ページの『OPEN REVERSED ステートメント』
OTHERWISE 文節の変更		83 ページの『IF . . . OTHERWISE ステートメントの変更』
パラメーターとして使用できない段落名		77 ページの『CALL ステートメントの変更』
PERFORM ステートメント - VARYING 句および AFTER 句の変更		86 ページの『PERFORM ステートメント: VARYING/AFTER 句の変更』
PERFORM ステートメント - 2 番目の UNTIL		70 ページの『PERFORM ステートメント - 第 2 の UNTIL』
任意の部における連続したピリオド		71 ページの『任意の部における連続したピリオド』
区域 A におけるピリオド	CCCA	71 ページの『区域 A におけるピリオド』
段落名で欠落しているピリオド	CCCA	71 ページの『段落名で欠落しているピリオド』
SD、FD、または RD の終わりで欠落しているピリオド		71 ページの『SD、FD、または RD の終わりで欠落しているピリオド』
PICTURE 文節 (数字編集の変更)		71 ページの『PICTURE ストリング』
PROGRAM COLLATING SEQUENCE 文節の変更		87 ページの『PROGRAM COLLATING SEQUENCE 文節の変更』
固有でない Program-ID 名	CCCA	72 ページの『固有でない PROGRAM-ID 名』
修飾 - 同じ句の反復使用		72 ページの『修飾 - 同じ句の反復使用』
READ ステートメント - KEY 句内の再定義されたレコード・キー		72 ページの『READ ステートメント - KEY 句内の再定義されたレコード・キー』
READ および RETURN ステートメントの変更 - INTO 句		87 ページの『READ および RETURN ステートメントの変更 : INTO 句』

表 8. *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
READY TRACE および RESET TRACE ステートメント	CCCA	63 ページの『READY TRACE および RESET TRACE ステートメント』
RECORD CONTAINS n CHARACTERS 文節		72 ページの『RECORD CONTAINS n CHARACTERS 文節』
RECORD KEY 句および ALTERNATE RECORD KEY 句		72 ページの『RECORD KEY 句および ALTERNATE RECORD KEY 句』
SD または FD 記入項目内の REDEFINES 文節	CCCA	73 ページの『SD または FD 記入項目内の REDEFINES 文節』
テーブルを指定した REDEFINES 文節		73 ページの『テーブルを指定した REDEFINES 文節』
比較条件	CCCA	73 ページの『比較条件』
REMARKS 段落	CCCA	65 ページの『REMARKS 段落』
RENAMES 文節 - 固有でない非修飾データ名		74 ページの『RENAMES 文節 - 固有でない非修飾データ名』
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	56
RERUN 文節の変更		87 ページの『RERUN 文節の変更』
RESERVE 文節の変更	CCCA	87 ページの『RESERVE 文節の変更』
予約語リストの変更	CCCA	88 ページの『予約語リストの変更』
SEARCH ステートメントの変更	CCCA	88 ページの『SEARCH ステートメントの変更』
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		89 ページの『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』
対応する FD のない SELECT ステートメント		74 ページの『対応する FD のない SELECT ステートメント』
SELECT OPTIONAL 文節の変更	CCCA	89 ページの『SELECT OPTIONAL 文節の変更』
SORT 特殊レジスター		89 ページの『SORT 特殊レジスター』

表 8. 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
SORT 動詞		74 ページの『SORT 動詞』
SORT または MERGE		74 ページの『SORT または MERGE』
ソース言語のデバッグの変更		89 ページの『ソース言語のデバッグの変更』
START . . . USING KEY ステートメント	CCCA	65 ページの『START . . . USING KEY ステートメント』
STRING ステートメント - PROGRAM COLLATING SEQUENCE 文節		87 ページの『PROGRAM COLLATING SEQUENCE 文節の変更』
STRING ステートメント - 送り出しフィールド ID		74 ページの『STRING ステートメント - 送り出しフィールド ID』
範囲外の添え字 - コンパイル時にフラグ設定		90 ページの『コンパイル時にフラグ設定される範囲外添え字』
ステートメント結合子としての THEN	CCCA	65 ページの『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	CCCA	65 ページの『TIME-OF-DAY 特殊レジスター』
LABEL RECORDS 文節内の TOTALING/TOTALED AREA 句	CCCA	62 ページの『TOTALING/ TOTALED AREA 句を持つ LABEL RECORDS 文節』
TRANSFORM ステートメント	CCCA	65 ページの『TRANSFORM ステートメント』
UNSTRING ステートメント - PROGRAM COLLATING SEQUENCE 文節		87 ページの『PROGRAM COLLATING SEQUENCE 文節の変更』
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	75 ページの『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』
UNSTRING ステートメント - 複数の INTO 句		75 ページの『UNSTRING ステートメント - 複数の INTO 句』
UNSTRING ステートメント - 添え字の評価の変更		90 ページの『UNSTRING ステートメント : 添え字の評価の変更』

表 8. *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
UPSI スイッチ	CCCA	91 ページの『UPSI スイッチ』
USE AFTER STANDARD ERROR - GIVING 句	CCCA	62 ページの『USE AFTER STANDARD ERROR 宣言の GIVING 句』
USE BEFORE STANDARD LABEL ステートメント	CCCA	66 ページの『USE BEFORE STANDARD LABEL』
VALUE 文節 - PICTURE 文節に関連した符号付き値	CCCA	75 ページの『VALUE 文節 - PICTURE 文節に関連した符号付き値』
VALUE 文節 - 条件名	CCCA	92 ページの『VALUE 文節の条件名』
WHEN-COMPILED 特殊レジスター	CCCA	92 ページの『WHEN-COMPILED 特殊レジスター』
WRITE AFTER POSITIONING ステートメント	CCCA	92 ページの『WRITE AFTER POSITIONING ステートメント』

* これは BDAM ファイル処理の一部についての移行です。

85 COBOL 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ために役立つよう、この*「移行ガイド」* 内の他の部分に記載されている情報を含め、複数の方法を使用できます。

続いて、役に立つ 2 つのメカニズム (CCCA および MIGR オプション) の要旨を示します。詳細については、267 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。

ヒント: IBM 以外のツールも、85 COBOL 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。

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、272 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 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。

267 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。

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

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・プリコンパイラーは、独立したプリコンパイラーとして実行でき、EXIT コンパイラー・オプションを使用して COBOL コンパイルに取り込むこともできます。

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

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 Language Environment のもとで実行

既存の OS/VS COBOL 報告書作成プログラム・アプリケーションは、Enterprise COBOL でコンパイルせずに Language Environment を使用して実行できますが、Enterprise COBOL V5 とは混用できません。 Enterprise COBOL V5 プログラムと OS/VS COBOL 報告書作成プログラムを混用したい場合は、Enterprise COBOL V5 を使用するようにすべてのプログラムを変換し、報告書作成プログラム・プリコンパイラーを使用する必要があります。

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

報告書作成プログラムのプリコンパイラーがインストール済みの場合にのみ、Enterprise COBOL で受け入れられる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、 267 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い言語エレメント

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、68 COBOL 標準言語エレメントの大部分に関する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 ファイルの処理も、z/OS V1.7 以降のリリースもサポートしません。 z/OS V1.7 以降に移行する前に、ISAM ファイルを VSAM/KSDS ファイルに変換する必要があります。

影響を受ける ISAM ファイル処理言語項目

Enterprise COBOL で受け入れられない 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

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。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。IDCAMS REPRO は z/OS V1.6 以前の ISAM ファイルに対してのみ作業します。 z/OS V1.7 以降に移行する前に、ISAM を VSAM/KSDS にマイグレーションする必要があります。

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、267 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。

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 で受け入れられない 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、267 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。

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、Language Environment 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 の中に RECEIVE ステートメントがある OS/VS COBOL プログラムでのみ使用されます。)

詳細については、*IBM VS COBOL for OS/VS*、および*IBM OS/VS COBOL Compiler and Library Programmer's Guide* 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信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信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 プログラムを変更する必要があります。

```
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.
```

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 プログラムに次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、 引き続き TALLY 特殊レジスターを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 に対する修正処置

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

```
WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).
      . . .
    EXHIBIT NAMED DAT-1 DAT-2
```

Enterprise COBOL

```
WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).
      . . .
    DISPLAY "DAT-1 = " DAT-1
        "DAT-2 = " DAT-2
```

EXHIBIT CHANGED に対する修正処置

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

```
WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).

      . . .
 EXHIBIT CHANGED 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
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 句を持つ 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。

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。

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を、COBOL プログラムまたは JCL と一致させなくても、OPEN ステートメントを正常に実行することができました。 Enterprise COBOL では、以下の条件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、313 ページの『付録 G. 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 プログラムと一致しなくても、OPEN ステートメントを正常に実行することができました。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、および RRDS ファイル定義に適用されます。

表 9. 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。 *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。 *n* は COBOL レコードの最大サイズ以上です。

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、各レコードを読み取り後に WORKING-STORAGE に移動するか、または READ INTO を使用します。

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、Debug Tool 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。

Debug Tool を使用する場合は、TEST 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 Debug Tool 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するか、または浮動コメント標識 *> を使用します。

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信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 プログラムを変更する必要があります。

```
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
```

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。

例えば、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。

例えば、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。

SEARCH ALL ステートメント

SEARCH ALL ステートメントを含み、OS/VS COBOL でコンパイルされたプログラムを使用している場合、SEARCH ALL ステートメントの動作の変更により、何らかの変更を行う必要が生じることがあります。

SEARCH ALL ステートメントの新しい動作については、112 ページの『SEARCH ALL ステートメントを含むプログラムのアップグレード』で説明されています。

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。

例えば、以下の比較の結果は、

```
"1 IS NOT > 0"
```

というメッセージが出され、数値的に正しくありません。

```
"1 > 0"
```

```
    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。

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す。

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。

索引名 OS/VS COBOL では、修飾された索引名を使用できました。

Enterprise COBOL では、修飾された索引名を使用できません。索引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。例えば、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。

例えば、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 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 で複数の受信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信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、受信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信側の前で予約語 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、固定数値受信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。Enterprise COBOL では、例を次のステートメントに変更すると、受け入れられます。

```
MOVE ALL ' ' TO num1(1:)
```

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信側を持つ 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.
```

コンパイラー・オプション DIAGTRUNC が有効であれば、Enterprise COBOL で同じ動作が可能になります。

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。

例えば、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、次のコード・シーケンス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 - 第 2 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)。

以下のコードの例の場合、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。

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

74 COBOL 標準との相違点として、OS/VS COBOL プログラムの RECORD CONTAINS *n* CHARACTERS 節は、FD 内で OCCURS DEPENDING ON 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。

QSAM RDW から取得するレコード長

OS/VS COBOL では、無効な負の添え字を使用することによって、可変長レコードを含むファイルのレコード長を RDW から取得できます。

Enterprise COBOL では、レコード内容の前にある領域内の可変ファイルの RDW は使用不可です。以前の COBOL 製品から移行するには、可変レコードの長さを、レコード自体の中にその情報がない場合は、FD 項目内でフォーマット 3 の RECORD 節を使用して設定または取得します。構文には、RECORD IS VARYING DEPENDING ON データ名 *1* が含まれます。データ名 *1* は WORKING-STORAGE に定義されます。コンパイラーが可変レコードを読み取った後、読み取られたデータの長さが自動的にデータ名 *1* に保管されます。以下に例を示します。

```
FILE  SECTION.
 FD  THE-FILE RECORD IS VARYING DEPENDING ON REC-LENGTH.
 01  THE-RECORD  PICTURE X(5000) .
 WORKING-STORAGE  SECTION.
 01  REC-LENGTH  PICTURE 9(5)  COMPUTATIONAL.
 01  SAVED-RECORD PICTURE X(5000).
 PROCEDURE DIVISION.
* Read a record of unknown length.
     READ THE-FILE.
     DISPLAY REC-LENGTH.
* or use REC-LENGTH to access the right amount of data:
     MOVE THE-RECORD (1:REC-LENGTH) TO SAVED-RECORD.
```

RECORD 文節について詳しくは、*Enterprise COBOL 言語解説書* を参照してください。

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。

例えば、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。例えば、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

OS/VS COBOL R2.3	Enterprise COBOL
< 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信フィールド ID を使用できました。Enterprise COBOL のもとでは、数値送信フィールド 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信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 - PICTURE 文節に関連した符号付き値

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

85 COBOL 標準に準拠するために、Enterprise COBOL ではいくつかの OS/VS COBOL 言語エレメントが変更されています。

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、セマンティクスが変更されたために実行結果が異なる可能性のあ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-NAME 節の変更: ALPHABET キーワード

OS/VS COBOL では、キーワード ALPHABET は ALPHABET-NAMES 節で許可されていませんでした。

Enterprise COBOL では、キーワード ALPHABET があり、これは必須です。

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 索引付きファイル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。

例えば、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、Language Environment 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
- オプション・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) を指定した 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
  ```

オプション・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ン・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、オプション・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。

Enterprise COBOL では、オプション・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。例えば、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。

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。

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、81 ページの表 10 を参照してください。
- VSAM ファイルについては、82 ページの表 11 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 10. 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	オプション・ファイルが使用できません。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	非オプション・ファイルが使用できません。
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。
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 11. 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	オプション・ファイルが使用できません。正常終了。
00	35	非オプション・ファイルが使用できません。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。
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。

表 11. 状況キーの値 : VSAM ファイル (続き)

OS/VS	Enterprise COBOL	意味
93 96	35	非オプション・ファイルが使用できません。
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信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)、受信フィールド (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。

例えば、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.
```

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。

例えば、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。

例えば、`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(NOPFD) は NUMPROC(PFD) に比べると、OS/VS COBOL の場合とよく似た処理をもたらしますが、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)。

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。

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、74 COBOL 標準と 85 COBOL 標準との間の変更点です。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。

オプション・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ン・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、オプション・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。

Enterprise COBOL では、オプション・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、以下の結果になります。

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、結果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

送信フィールドを、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、送信フィールドとして 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。

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。251 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 は次のステートメント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、「*Language Environment インストールおよびカスタマイズ*」を参照してください。

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 90 ページの表 12 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。Debug Tool 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 12. 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。
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)	詳しくは、「*IBM VS COBOL for OS/VS*」を参照してください。
ALL REFERENCES OF identifier-n	(なし)	詳しくは、「*IBM VS COBOL for OS/VS*」を参照してください。
cd-name-1	(なし)	詳しくは、「*IBM VS COBOL for OS/VS*」を参照してください。

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 : 添え字の評価の変更

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。

これらのフィールドの場合、Enterprise COBOL UNSTRING ステートメントで、関連した添え字付け、指標付け、または長さ計算の評価は (区切り文字送り出しフィールドの検査の直前に) 1 回だけ行われます。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信フィールド (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信フィールド (指定されている場合) に移動されます。

例えば、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
```

LANGLVL(2) を使用した OS/VS COBOL、および 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。

例えば、SPECIAL-NAMES 段落で条件名が定義されている場合、以下のコード例は同じ効果があります。

OS/VS COBOL

```
SPECIAL-NAMES.
    UPSI-0 IS MNUPO

        . . .
PROCEDURE DIVISION
        . . .
    IF UPSI-0 = 1 ...
    IF MNUPO = 0 ...
```

Enterprise COBOL

```
SPECIAL-NAMES.
    UPSI-0 IS MNUPO
     ON STATUS IS UPSI-0-ON
     OFF STATUS IS UPSI-0-OFF
       . .  .
PROCEDURE DIVISION
       . .  .
     IF UPSI-0-ON ...
     IF UPSI-0-OFF ...
```

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 では、 AFTER POSITIONING 句と一緒にサポートされていましたが、Enterprise COBOL ではサポートされていません。

Enterprise COBOL では、WRITE...AFTER ADVANCING ステートメントを使用して、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
```

制約事項: 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。例えば、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 6 章 移行済み OS/VS COBOL プログラムのコンパイル

このセクションには、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

OS/VS COBOL または Enterprise COBOL に関する特定の情報が記載されています。

移行済みプログラム用のコンパイラー・オプション

表 13 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 13. 移行済み OS/VS COBOL プログラム用のコンパイラー・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、AMODE 24 アセンブラー・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。
NOSTGOPT	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、NOSTGOPT 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、STGOPT を使用してください。
NUMPROC	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) とインストール・オプション 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符号表記についてのその他の代替手段については、「*Enterprise COBOL プログラミング・ガイド*」を参照してください。
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 Language Environment 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 Language Environment 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 Language Environment 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。

表 13. 移行済み OS/VS COBOL プログラム用のコンパイラー・オプション (続き)

コンパイラー・オプション	説明
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。
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信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 のデフォルトとして TRUNC を使用している場合、TRUNC(STD) を使用します。インストール先で OS/VS COBOL のデフォルトとして NOTRUNC を使用している場合、TRUNC(OPT) を使用します (バイナリー・データの切り捨てが起こらないようにする必要があるプログラムを選択する場合は除きます)。バイナリー・データの切り捨てが起こらないようにする必要があるプログラムでは、TRUNC(BIN) を使用します (特に、バイナリー・データ項目に移行されるデータの値が、バイナリー・データ項目の PICTURE 文節で定義された値より長くなる可能性がある場合)。個別のデータ項目に対して USAGE COMP-5 を指定し、バイナリー・データが確実に切り捨てられないようにすることができます。 **高位桁::** TRUNC(OPT) を使用してコンパイルされた Enterprise COBOL プログラムの結果は、NOTRUNC を使用してコンパイルされた OS/VS COBOL プログラムの結果とは異なることがあります。主な相違点は、プログラムが非ゼロの高位桁を失う可能性があるということです。高位桁の消失が起こる可能性のあるステートメントの場合、 Enterprise COBOL では、少なくとも以下の状態の 1 つを確実に満たす必要があることを示す診断メッセージが出されます。 • 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数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

表 14 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、 285 ページの『付録 E. オプションの比較』を参照してください。

表 14. 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。 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	Debug Tool 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、85 COBOL 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。

表 14. 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 (続き)

OS/VS COBOL オプション	Enterprise COBOL で同等のオプション
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・モジュールは常に、COBOL プログラムとリンク・エディットされるのではなく、ライブラリー・サブルーチンが実行時に動的に配置されるように作成されます。これは OS/VS COBOL の RES 動作に相当します。
STATE/NOSTATE	機能は、TEST オプションを介して使用できます。
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、Language Environment 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 環境でのみ実行され、 z/VM® では実行されません。 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 Language Environment 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。

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 7 章 VS COBOL II ソース・プログラムのアップデート

VS COBOL II 言語と Enterprise COBOL 言語の間には相違があるため、プログラムを変更しなければならない場合があります。

VS COBOL II プログラムは、以下の 1 つ以上の条件に該当しない限り、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。Enterprise COBOL は、CMPR2/NOCMPR2 コンパイラー・オプションをサポートしません。
- VS COBOL II リリース 3.x でコンパイルされたプログラムのうち、85 COBOL 標準の解釈の変更対象となった 3 つのマイナー 85 COBOL 標準機能のうちの 1 つ以上が含まれるプログラム
- VS COBOL II リリース 3.0 でコンパイルされたプログラムのうち、ACCEPT . . . FROM CONSOLE を使用するプログラム
- 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 拡張を使用するプログラム
- SEARCH ALL ステートメントを含むプログラム
- SIMVRD サポートを使用するプログラム
- 形式 2 宣言構文 `USE...AFTER...LABEL PROCEDURE...` および構文 `GO TO MORE-LABELS` (オプション) が含まれるプログラム。これらのプログラムのサポートは Enterprise COBOL バージョン 5 では廃止されました。

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 CMPR2 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。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。CMPR2 と NOCMPR2 (85 COBOL 標準) 間の言語の違いについては、 118 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、 267 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。

85 COBOL 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信側の参照変更

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表示さ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は表示されません。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 (例えば、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 の場合とは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信側である。
- 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。

例えば、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。

`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信側が含まれており、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、参照修飾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

Enterprise COBOL は、VS COBOL II と比較して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。

新しい予約語

ご使用のプログラムで新しい予約語のいずれかがユーザー定義語 (データ項目名や段落名など) として使用されている場合は、これらの語を変更する必要があります。 CCCA と類似の内容を行うことができ、また単に -85 などの接尾部を語のすべてのインスタンスに追加することができます。以下に例を示します。

```
77 VOLATILE PIC S9(9) BINARY.
Move 0 TO VOLATILE.
```

Enterprise COBOL V5 でコンパイルするには、これを次のように変更します。

```
77 VOLATILE-85 PIC S9(9) BINARY.
Move 0 TO VOLATILE-85.
```

Enterprise COBOL V5 での新しい予約語は XML-INFORMATION と VOLATILE です。

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、267 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。

CCCA は、PTF for APAR PM86253 によって、Enterprise COBOL バージョン 5.1 の予約語変換用に更新されています。バージョン 5.2 では、APAR PI32750 の PTF によって、予約語変換に関して CCCA が更新されています。

以下の表は 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、251 ページの『付録 B. COBOL 予約語の比較』を参照してください。

表 15. コンパイラー別の新規予約語

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 V3R1	JNIENVPTR, NATIONAL, XML, END-XML, XML-EVENT, XML-CODE, XML-TEXT, XML-NTEXT, FUNCTION-POINTER
Enterprise COBOL V3R4	NATIONAL-EDITED, GROUP-USAGE
Enterprise COBOL V4R1	XML-NAMESPACE, XML-NAMESPACE-PREFIX, XML-NNAMESPACE, XML-NNAMESPACE-PREFIX
Enterprise COBOL V4R2	XML-SCHEMA **注:** XML-INFORMATION が予約語として追加されました (APAR PM85035)。
Enterprise COBOL V5R1	XML-INFORMATION
Enterprise COBOL V5R2	VOLATILE

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 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 内のピリオドの前には、有効な区域 A 項目が必要です。

SEARCH ALL ステートメント

SEARCH ALL ステートメントを含み、VS COBOL II でコンパイルされたプログラムを使用している場合、SEARCH ALL ステートメントの動作の変更により、何らかの変更を行う必要が生じることがあります。

SEARCH ALL ステートメントの新しい動作については、112 ページの『SEARCH ALL ステートメントを含むプログラムのアップグレード』で説明されています。

SIMVRD サポートを使用するプログラムのアップグレード

このセクションでは、SIMVRD サポートを使用するプログラムをアップグレードするための処置について説明します。 *COBOL シミュレート可変長相対レコードデータ・セット (RRDS)* は、Enterprise COBOL バージョン 4 以降でコンパイルされたプログラムではサポートされません。これらのファイルは VSAM RRDS ファイルに変更する必要があります。

Enterprise COBOL バージョン 4 より前の、NOCMPR2 コンパイラー・オプションをサポートする COBOL コンパイラーでは、SIMVRD ランタイム・オプション・サポートを使用する場合、VSAM KSDS を使って *COBOL シミュレート可変長 RRDS* が使用できました。

COBOL プログラムで VSAM 可変長 RRDS および COBOL シミュレート可変長 RRDS を特定および記述するために使用するコーディングは、同様です。Enterprise COBOL バージョン 4 では、VSAM 可変長 RRDS サポートを使用しなければなりません。一般に、COBOL シミュレート可変長 RRDS から VSAM 可変長 RRDS サポートへマイグレーションするのに必要な処置は、ファイルの IDCAMS 定義を変更することだけです。

表 16. 可変長 RRDS を使用するステップ

ステップ	VSAM 可変長 RRDS	COBOL シミュレート可変長 RRDS
1	ORGANIZATION IS RELATIVE 節でファイルを定義します。	同じ
2	FD 項目を使って可変長サイズでレコードを記述します。	同じ。ただし、データ・セットにアクセスするすべての COBOL プログラムの FD 項目の中に、RECORD IS VARYING をコーディングしなければなりません。
3	NOSIMVRD ランタイム・オプションを使用します。	SIMVRD ランタイム・オプションを使用します。

4	アクセス方式サービス・プログラムを通じて、VSAM ファイルを RRDS として定義します。	アクセス方式サービス・プログラムを通じて、VSAM ファイルを以下のように定義します。
		`DEFINE CLUSTER INDEXED`
		`KEYS(4,0)`
		`RECORDSIZE(avg,m)`
		avg COBOL レコードの平均サイズです。必ず *m* 未満でなければなりません。
		m 最大サイズ COBOL レコード + 4 以上です。

シミュレート可変長 RRDS のステップ 2 で、可変長レコード・フォーマットを暗示するほかの言語エレメントをコーディングしても、COBOL シミュレート可変長 RRDS にはなりません。例えば、以下のエレメントだけでは、シミュレート可変長 RRDS アクセスを使用しませんでした。したがって SIMVRD ランタイム・オプションは必要ありませんでした。

- 長さが異なる複数の FD レコード
- レコード定義内の OCCURS . . . DEPENDING ON

- RECORD CONTAINS *integer-1* TO *integer-2* CHARACTERS

レコードを含み、出力用にオープンするファイルに対して、REUSE IDCAMS パラメーターを使用します。

- ORGANIZATION IS RELATIVE 節でファイルを定義します。
- FD 項目を使って可変長サイズでレコードを記述します。
- NOSIMVRD ランタイム・オプションを使用します。
- アクセス方式サービス・プログラムを通じて、VSAM ファイルを RRDS として定義します。

エラー: シミュレート可変長相対データ・セットと真の VSAM RRDS データ・セットを処理する場合、COBOL ファイル定義と VSAM データ・セット属性が一致しないと OPEN ファイル状況 39 が発生します。

可変長 RRDS を使用するためのコマンドに関する詳しい説明については、*z/OS DFSMS: カタログ用アクセス方式サービス・プログラム*を参照してください。

第 8 章 VS COBOL II プログラムのコンパイル

このセクションには、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 プログラム用のコンパイラー・オプション
- Prolog 形式の変更点

VS COBOL II または Enterprise COBOL に関する特定の情報が記載されています。

VS COBOL II プログラム用のコンパイラー・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。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、118 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 17 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 17. 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。
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。現在、TEST オプションには 2 つのサブオプションがあります。ソース・ファイルの情報をオブジェクト内に保管するかどうか、および JUMPTO コマンドと GOTO コマンドを Debug Tool で使用できるようにするかどうかを指定できます。 サブオプションを指定しない TEST は、TEST(NOEJPD, SOURCE) になります。TEST 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。

Enterprise COBOL でサポートされないコンパイラー・オプション

表 18 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 18. 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、85 COBOL 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 は、同等以上の機能を提供することができます。 Language Environment 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。TEST を指定すると、Language Environment 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 Language Environment の定様式ダンプを取得する方法については、「*Language Environment デバッグ・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。
NUMPROC(MIG)	Enterprise COBOL V5 は NUMPROC(MIG) オプションをサポートしていません。 NUMPROC(MIG) が指定された場合、Enterprise COBOL V5 は警告メッセージを発行し、コンパイルで NUMPROC のデフォルト設定が取得されます。これは、ユーザーがカスタマイズしたデフォルト、または IBM デフォルト (NUMPROC(NOPFD)) のいずれかです。
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 Language Environment 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 の形式を Enterprise COBOL の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 9 章 IBM COBOL ソース・プログラムのアップグレード

IBM COBOL と Enterprise COBOL では、COBOL 言語のサポートが異なります。

この情報は、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の修正が必要かを判別するうえで役立ちます。例えば、CMPR2 オプションを指定してコンパイルされた IBM COBOL プログラムでは、Enterprise COBOL が CMPR2/NOCMPR2 コンパイラー・オプションをサポートしないため、ソースの修正が必要です。

このセクションでは、IBM COBOL 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 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 (OO) COBOL プログラムのアップグレード
- サポートされない SOM ベースの OO COBOL 言語エレメント
- 変更された SOM ベースの OO COBOL 言語エレメント
- Enterprise COBOL の新しい予約語
- Language Environment ランタイムの考慮事項

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、118 ページの『CMPR2 コンパイラー・オプションから 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) 変換プログラムから組み込みの CICS 変換プログラムへのマイグレーションについては、226 ページの『単独の CICS 変換プログラムから組み込みの変換プログラム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

多くの IBM COBOL プログラムは、変更を行わずに Enterprise COBOL でコンパイルされます。

ただし、以下のプログラムは、Enterprise COBOL でコンパイルする前にアップグレードする必要があります。

- SEARCH ALL ステートメントを含むプログラム
- SIMVRD サポートを使用するプログラム
- 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 IBM COBOL 拡張機能を持つプログラム
- 形式 2 宣言構文 `USE...AFTER...LABEL PROCEDURE...` および構文 `GO TO MORE-LABELS` (オプション) が含まれるプログラム。これらのプログラムのサポートは Enterprise COBOL バージョン 5 では廃止されました。

- DATE FORMAT データ型、および 2000 年問題用の関数 (DATEVAL、UNDATE、YEARWINDOW) の一方または両方を使用するプログラム
- SOM ベースのオブジェクト指向 COBOL 構文を持つプログラム
- 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

SEARCH ALL ステートメントを含むプログラムのアップグレード

Enterprise COBOL では、SEARCH ALL ステートメントのインプリメンテーションでのエラーが修正されています。以前のリリースの COBOL の SEARCH ALL ステートメントには、キー比較論理にエラーがありました。このエラーが、意図していたものとは異なる結果を招きました。特に、比較では IF ステートメントまたは順次 SEARCH ステートメントと同じ結果が作成されませんでした。

長さの不一致の問題: 検索引数がテーブル・キーより長くなる

SEARCH ALL ステートメントの比較では、キーが SEARCH 引数より短いと、英数字キーにはブランクが埋め込まれ、数字キーには先行ゼロが加えられます。ただし、V3R3 とそれ以前のリリースでは場合によっては、SEARCH ALL で引数の余分の文字が無視されました。例えば、「ABCDEF」を含む `01 ARG PIC X(6)` の英数字検索引数は、値「ABCD」をともなう `05 MY-KEY PIC X(4)` のテーブルまたは配列のキーと、誤って一致してしまいます。「ABCD 」(ブランクあり) を含む検索引数は期待通りに一致となります。

数字の検索引数およびキーの場合にも同様の問題が発生しました。例えば、「123456」を含む`01 ARG PIC 9(6)` の検索引数は、値「3456」をともなう `05 MY-KEY PIC 9(4)` のテーブルまたは配列のキーと、誤って一致してしまいます。003456 を含む検索引数は期待通りに一致となります。

符号の不一致の問題: 符号付き数字引数と符号なし数字キー

第 2 の問題が発生するのは、検索引数が符号付き数字項目で、テーブル・キーが符号なし数字項目の場合です。検索引数のランタイム値が -1234 などの負数である場合、V3R3 と以前のリリースでコンパイルされたプログラムでは、1234 のテーブル・キーが一致となります。こうした比較は、通常の COBOL 比較条件の規則を使用して行われると、-1234 などの負の引数は符号なしのテーブル・キーと一致することはありません。

修正処置:

Enterprise COBOL ではこれらの問題は修正されました。ただし、以前のリリースでコンパイルされたアプリケーションには、間違った動作に依存するものもあります。これらのアプリケーションを特定し、修正してから、 Enterprise COBOL バージョン 4 以降に移行する必要があります。

こうした修正の影響を受けるプログラムおよび SEARCH ALL ステートメントの特定に役立つように、以下のコンパイラーおよびランタイム警告診断が出されます。

- コンパイラー・メッセージ: Enterprise COBOL コンパイラーは、以下のコンパイラー診断を生成します。実際に影響があるかどうかは、実行時の引数の内容によります。

- IGYPG3189-W。テーブル・キーより長い検索引数を持つ (つまり第 1 の問題が生じる可能性がある) すべての SEARCH ALL ステートメントについて出されます。
- IGYPG3188-W。検索引数が符号付き数字項目で、テーブル・キーが符号なし数字項目である (つまりプログラムで第 2 の問題が生じる可能性がある) 場合に出されます。

- ランタイム・メッセージ : 以下のランタイム・メッセージが生成されます。これらのランタイム・メッセージのいずれかを生成するプログラムは、変更の影響を受けます。
  - IGZ0194W。余分のバイトがブランクまたはゼロでない検索引数を持つすべての SEARCH ALL ステートメントについて出されます。
  - IGZ0193W。検索引数が負の値の符号付き数字項目で、テーブル・キーは符号なし数字項目の場合に出されます。

移行の手順

アプリケーションを Enterprise COBOL バージョン 4 以降に移行するには、以下の一連の手順のいずれかを行います。

- コンパイラー・メッセージについて処置を行います。
  1. プログラムを Enterprise COBOL でコンパイルします。
  2. コンパイラー・メッセージ IGYPG3188-W または IGYPG3189-W で示された SEARCH ALL ステートメントを検討します。このようなステートメントは影響を受けている可能性があります。

     ヒント: 非互換の結果が生じる可能性を最小化するために、これらのメッセージの重大度を E または S に変更することによって、これらの SEARCH ALL ステートメントの修正をプログラマーに強制することができます。メッセージの重大度を変更するには、EXIT コンパイラー・オプションの MSGEXIT サブオプションを使用します。この変更処置を行うと、これらのメッセージが発生するプログラムは、コードが修正されるまで実行できなくなります。IGYPG3188-W および IGYPG3189-W の重大度をそれぞれ IGYPG3188-S と IGYPG3189-S に変更するサンプル・コードが、サンプル・ユーザー出口 IGYMSGXT に含まれています。
- ランタイム・メッセージについて処置を行います。
  1. アプリケーションをテスト環境で実行します。
  2. ランタイム・メッセージ IGZ0193W または IGZ0194W を生成する SEARCH ALL ステートメントを検討します。

影響を受ける SEARCH ALL ステートメントを特定したら、以下の手順でアプリケーション・ロジックを適切に調整します。

- 検索引数がテーブル・キーより長い SEARCH ALL ステートメントの場合、以下のいずれかの処置を行います。
  - 必ず、キーの長さを超えている引数のバイトが、スペースまたはゼロになるようにします。

    ヒント: この調査を完了し、プログラムを変更しないことに決めた場合、EXIT コンパイラー・オプションの MSGEXIT サブオプションを使用して、

IGYPG3188-W および IGYPG3189-W の重大度をそれぞれ IGYPG3188-I と IGYPG3189-I に変更するか、またはこれらのメッセージを完全に抑制することができます。そうすると、プログラムは RC=0 でコンパイルされるようになります。サンプル・ユーザー出口 IGYMSGXT に、IGYPG3188-W および IGYPG3189-W の重大度を変更するサンプル・コードが含まれています。

– 引数をキーと同サイズの一時データ項目に移し、その一時項目を検索引数として使用します。
– 参照/修正によって比較範囲を制限します。以下に例を示します。
  - 検索引数 `01 ARG PIC X(6)` の英数字文字および `05 MY-KEY PIC X(4)` のキーでは、以下を使用します。

    ```
    WHEN MY-KEY (MY-INDEX) = ARG(1:4)
    ```

  - 検索引数 `01 ARG PIC 9(6)` の数字および `05 MY-KEY PIC 9(4)` の配列のキーでは、以下を使用します。

    ```
    WHEN MY-KEY (MY-INDEX) = ARG(3:4)
    ```

上記の 2 番目と 3 番目の処置によって、将来警告メッセージは表示されなくなります。

- 検索引数が符号付きで、テーブル・キーが符号なしである SEARCH ALL ステートメントの場合、必ず検索引数を正数値に正しく初期設定してから、SEARCH ステートメントを実行します。COBOL プログラム内の固有のアプリケーション・ロジックによっては、以下のいずれかの変更を行うことができます。
  – 引数のデータ記述を符号なしに変更します。
  – 検索引数を符号なしの一時変数に移し、その一時変数を SEARCH ALL ステートメントで使用します。

  いずれかの処置によって、将来警告メッセージは表示されなくなります。

SIMVRD サポートを使用するプログラムのアップグレード

このセクションでは、SIMVRD サポートを使用するプログラムをアップグレードするための処置について説明します。 *COBOL シミュレート可変長相対レコードデータ・セット (RRDS)* は、Enterprise COBOL バージョン 4 以降でコンパイルされたプログラムではサポートされません。これらのファイルは VSAM RRDS ファイルに変更する必要があります。

Enterprise COBOL バージョン 4 より前の、NOCMPR2 コンパイラー・オプションをサポートする COBOL コンパイラーでは、SIMVRD ランタイム・オプション・サポートを使用する場合、VSAM KSDS を使って *COBOL シミュレート可変長 RRDS* が使用できました。

COBOL プログラムで VSAM 可変長 RRDS および COBOL シミュレート可変長 RRDS を特定および記述するために使用するコーディングは、同様です。Enterprise COBOL バージョン 4 では、VSAM 可変長 RRDS サポートを使用しなければなりません。一般に、COBOL シミュレート可変長 RRDS から VSAM 可変長 RRDS サポートへマイグレーションするのに必要な処置は、ファイルの IDCAMS 定義を変更することだけです。

表 19. 可変長 RRDS を使用するステップ

ステップ	VSAM 可変長 RRDS	COBOL シミュレート可変長 RRDS
1	ORGANIZATION IS RELATIVE 節でファイルを定義します。	同じ
2	FD 項目を使って可変長サイズでレコードを記述します。	同じ。ただし、データ・セットにアクセスするすべての COBOL プログラムの FD 項目の中に、RECORD IS VARYING をコーディングしなければなりません。
3	NOSIMVRD ランタイム・オプションを使用します。	SIMVRD ランタイム・オプションを使用します。

4	アクセス方式サービス・プログラムを通じて、VSAM ファイルを RRDS として定義します。	アクセス方式サービス・プログラムを通じて、VSAM ファイルを以下のように定義します。
		`DEFINE CLUSTER INDEXED`
		`KEYS(4,0)`
		`RECORDSIZE(avg,m)`
		avg COBOL レコードの平均サイズです。必ず*m* 未満でなければなりません。
		m 最大サイズ COBOL レコード + 4 以上です。

シミュレート可変長 RRDS のステップ 2 で、可変長レコード・フォーマットを暗示するほかの言語エレメントをコーディングしても、COBOL シミュレート可変長 RRDS にはなりません。例えば、以下のエレメントだけでは、シミュレート可変長 RRDS アクセスを使用しませんでした。したがって SIMVRD ランタイム・オプションは必要ありませんでした。

- 長さが異なる複数の FD レコード
- レコード定義内の OCCURS . . . DEPENDING ON
- RECORD CONTAINS *integer-1* TO *integer-2* CHARACTERS

レコードを含み、出力用にオープンするファイルに対して、REUSE IDCAMS パラメーターを使用します。

- ORGANIZATION IS RELATIVE 節でファイルを定義します。
- FD 項目を使って可変長サイズでレコードを記述します。
- NOSIMVRD ランタイム・オプションを使用します。
- アクセス方式サービス・プログラムを通じて、VSAM ファイルを RRDS として定義します。

エラー: シミュレート可変長相対データ・セットと真の VSAM RRDS データ・セットを処理する場合、COBOL ファイル定義と VSAM データ・セット属性が一致しないと OPEN ファイル状況 39 が発生します。

可変長 RRDS を使用するためのコマンドに関する詳しい説明については、*z/OS DFSMS: カタログ用アクセス方式サービス・プログラム*を参照してください。

Language Environment ランタイムの考慮事項

Enterprise COBOL プログラムは、IBM COBOL でヒープ・ストレージが使用されていた一部のケースで Language Environment スタック・ストレージを使用します。このようなケースには、組み込み関数 UPPER-CASE や LOWER-CASE が含まれます。 Enterprise COBOL で再コンパイルを行うと、スタック・ストレージの使用量が著しく変わる可能性があります。 STACK が 16 MB 境界より下に割り振られている場合にラージ DSA (動的保存域) が必要になると、ストレージ不足エラーが起こる可能性があります。

必要なストレージの量を調べるには、コンパイラー・オプション MAP および LIST を指定してプログラムをコンパイルしてください。次のリスト行の下で FuncResultTemp を探してください: ***** AUTOMATIC MAP******

境界より上のストレージを使用するには、必要なストレージの量を減らすか、またはランタイム・オプションを STACK=(...ANYWHERE..) に変更しなければならない場合があります。

PICTURE P を持つ数値項目に関する考慮事項

Enterprise COBOL では、PICTURE 文字ストリングに記号 P が含まれているデータ項目を参照する際に、送信オペランドが数値である場合、記号 P で指定された桁位置にはゼロが含まれているとみなされます。

例えば、PICTURE 9P VALUE IS 10 を持つデータ項目を、PICTURE 99 を持つデータ項目と PICTURE XX を持つデータ項目に移動すると、どちらの場合も受信フィールドには 10 が含められます。しかし、項目を英数字項目と比較する場合のように、数値項目が必要ないときには、文字値が使用されます。例えば、PICTURE 9P と VALUE IS 10 を持つ項目は、PICTURE XX と VALUE IS "1 " (数字 1、その後にスペース) を持つ項目と等しくなります。

Enterprise COBOL の新しい予約語

Enterprise COBOL は、IBM COBOL と比較して多くの予約語が追加されています。既存の IBM COBOL 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。

新しい予約語

ご使用のプログラムで新しい予約語のいずれかがユーザー定義語 (データ項目名や段落名など) として使用されている場合は、これらの語を変更する必要があります。 CCCA と類似の内容を行うことができ、また単に -85 などの接尾部を語のすべてのインスタンスに追加することができます。以下に例を示します。

```
77 VOLATILE PIC S9(9) BINARY.
Move 0 TO VOLATILE.
```

Enterprise COBOL V5 でコンパイルするには、これを次のように変更します。

```
77 VOLATILE-85 PIC S9(9) BINARY.
Move 0 TO VOLATILE-85.
```

Enterprise COBOL V5 での新しい予約語は XML-INFORMATION と VOLATILE です。

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、 267 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。

CCCA は、PTF for APAR PM86253 によって、Enterprise COBOL バージョン 5.1 の予約語変換用に更新されています。バージョン 5.2 では、APAR PI32750 の PTF によって、予約語変換に関して CCCA が更新されています。

以下の表は 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、 251 ページの『付録 B. COBOL 予約語の比較』を参照してください。

表 20. コンパイラー別の新規予約語

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 V3R1	JNIENVPTR, NATIONAL, XML, END-XML, XML-EVENT, XML-CODE, XML-TEXT, XML-NTEXT, FUNCTION-POINTER
Enterprise COBOL V3R4	NATIONAL-EDITED, GROUP-USAGE
Enterprise COBOL V4R1	XML-NAMESPACE, XML-NAMESPACE-PREFIX, XML-NNAMESPACE, XML-NNAMESPACE-PREFIX
Enterprise COBOL V4R2	XML-SCHEMA **注:** XML-INFORMATION が予約語として追加されました (APAR PM85035)。
Enterprise COBOL V5R1	XML-INFORMATION
Enterprise COBOL V5R2	VOLATILE

SEARCH ALL ステートメント

SEARCH ALL ステートメントを含み、IBM COBOL でコンパイルされたプログラムを使用している場合、SEARCH ALL ステートメントの動作の変更により、何らかの変更を行う必要が生じることがあります。

SEARCH ALL ステートメントを含み、以下のいずれかのコンパイラーでコンパイルされた特定のプログラムに対して、何らかの処置を行う必要があります。

- COBOL (OS/390 および VM 版)
- COBOL (MVS および VM 版)
- COBOL/370

SEARCH ALL ステートメントの新しい動作については、112 ページの『SEARCH ALL ステートメントを含むプログラムのアップグレード』で説明されています。

CMPR2 コンパイラー・オプションから NOCMPR2 へのマイグレーション

COBOL プログラムが CMPR2 オプションを指定してコンパイルされた場合、Enterprise COBOL でコンパイルするには、そのプログラムを NOCMPR2 プログラムに変換する必要があります。CMPR2/NOCMPR2 オプションは、Enterprise COBOL ではサポートされていません。

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。

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

VS COBOL II リリース 3.0 以降、NOCMPR2 コンパイラー・オプションを使用して 85 COBOL 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 74 COBOL 標準の動作を選択することができるようになりました。ただし、Enterprise COBOL では、プログラムは85 COBOL 標準レベルでなければなりません。

CMPR2 オプションでは、VS COBOL II リリース 2 によってインプリメントされた標準 COBOL 74 の動作、および現在85 COBOL 標準でインプリメントされている*非標準の* リリース 2 拡張が提供されていました。NOCMPR2 オプションでは、85 COBOL 標準準拠の動作および IBM 拡張が提供されました。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 85 COBOL 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。

Enterprise COBOL では 85 COBOL 標準のサポートを提供しているのに対し、VS COBOL II リリース 2 では (85 COBOL 標準の一部の機能が追加された) 74 COBOL 標準のサポートを提供していました。 85 COBOL 標準の実装により、一部の言語エレメントの動作は、74 COBOL 標準の実装とは異なります。

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。

ヒント: 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 21. 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	120 ページの『SPECIAL-NAMES 段落の ALPHABET 文節』
ALPHABETIC クラス	121 ページの『ALPHABETIC クラス』
CALL ... ON OVERFLOW	121 ページの『CALL . . . ON OVERFLOW』
位取り整数と非数字との比較	123 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	124 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	126 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	126 ページの『ファイル状況コード』
固定ファイル属性および JCL の DCB= パラメーター	128 ページの『固定ファイル属性および JCL の DCB= パラメーター』
暗黙の EXIT PROGRAM	129 ページの『暗黙の EXIT PROGRAM』
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)	131 ページの『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)』

表 21. 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
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)	131 ページの『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)』
PERFORM から戻る方法	132 ページの『PERFORM から戻る方法』
PERFORM...VARYING...AFTER	134 ページの『PERFORM ... VARYING ... AFTER』
「A」および「B」を指定した PICTURE 文節	136 ページの『「A」および「B」を指定した PICTURE 文節』
PROGRAM COLLATING SEQUENCE	139 ページの『PROGRAM COLLATING SEQUENCE』
READ INTO と RETURN INTO	140 ページの『READ INTO と RETURN INTO』
RECORD CONTAINS n CHARACTERS	142 ページの『RECORD CONTAINS n CHARACTERS』
SET...TO TRUE	142 ページの『SET . . . TO TRUE』
SIZE ERROR、MULTIPLY と DIVIDE の	144 ページの『SIZE ERROR、MULTIPLY と DIVIDE の』
UNSTRING オペランドの評価	146 ページの『UNSTRING オペランドの評価』
UPSI スイッチ	152 ページの『UPSI スイッチ』
可変長グループ移動	153 ページの『可変長グループ移動』

SPECIAL-NAMES 段落の ALPHABET 文節

ALPHABET が、ALPHABET 文節で指定する必要がある予約語かどうかは、CMPR2/NOCMPR2 オプションの設定によって異なります。

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

ALPHABETIC クラスに 26 文字の小文字が含まれるかどうかは、CMPR2/NOCMPR2 オプションの設定によって異なります。

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

「ストレージ不足」エラー以外のエラーで ON OVERFLOW 条件が発生するかどうかは、CMPR2/NOCMPR2 オプションの設定によって異なります。

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。

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 85 COBOL 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。

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。実際に「ストレージ不足」エラーが原因で ON OVERFLOW 条件になる場合、プログラムは以前と同じ動作を示しますが、他のエラーのためにこの条件になる場合は、(ON OVERFLOW 句に) コーディングしたリカバリーのためのステートメントではエラーを訂正できない可能性があり、それ以降の CALL も失敗することがあります。

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

非数字項目と位取りされた数値項目との間の比較は、CMPR2/NOCMPR2 オプションの設定に応じて、異なる方法で処理されます。

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

ライブラリー・テキストまたは COPY...REPLACING ステートメント内の一部の非 COBOL 文字は、CMPR2/NOCMPR2 オプションの設定によって、異なる方法で扱われます。

「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: 85 COBOL 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 COPY & REPLACING     After COPY & 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...REPLACING が実行された後の CMPR2 と Enterprise COBOL との相違は、以下のようになります。

CMPR2	NOCMPR2
`MOVE  Z  TO  ID2.`	`MOVE  X:Y  TO  ID2.`

Enterprise COBOL のもとでは、「:」は区切り文字の 1 つなので、「A:B」は 3 つの別々のトークン (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.
```

ここでは、「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 レベルの診断メッセージを出します。

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 8 桁目で検出された非 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

COPY ステートメントのテキスト名およびライブラリー名において、文字 @、#、および $ をコーディングできるかどうかは、CMPR2/NOCMPR2 オプションの設定によって異なります。

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。例えば、`COPY X$3.` では項目がコピーされます。

NOCMPR2: コンパイラーは E レベルの診断メッセージを出します。

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。例えば、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/NOCMPR2 オプションの設定は、返されるファイル状況コードと、そのコードが示す、入出力操作に関する詳細情報の量に影響します。

CMPR2: 74 COBOL 標準に基づくファイル状況コードは、CMPR2 とともに返されます。

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関する訂正処置: 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 22 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包括的な定義については、『*ファイル状況キー*』 を参照してください。

表 22. 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		

表 22. CMPR2 と NOCMPR2 での QSAM および VSAM ファイル状況コード (続き)

VSAM ファイル状況コード		QSAM ファイル状況コード	
CMPR2	**NOCMPR2**	**CMPR2**	**NOCMPR2**
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		

固定ファイル属性および JCL の DCB= パラメーター

ブロック・サイズ、レコード・サイズ、およびその他の固定ファイル属性の処理は、CMPR2 と NOCMPR2 では異なります。 NOCMPR2 に移行するには、ご使用のプログラムおよび JCL を変更する必要があります。

CMPR2: CMPR2 プログラムでは、固定ファイル属性検査を行う場合は、読み取り/書き込み時のみに実行されます。(全く実行されてない場合)OPEN ステートメントは、一部の固定ファイル属性が矛盾していたとしても、正常に実行されます。例えば、以下のものにおけるレコード・サイズが異なっていても、OPEN は正常に実行されます。

- RECORD CONTAINS x 文節
- JCL DCB=(LRECL=y)
- 実際のデータ・セット・ラベル

NOCMPR2: NOCMPR2 プログラムでの85 COBOL 標準においては、固定ファイル属性検査を何度か実行する必要があります。その結果、矛盾する固定ファイル属性を持つプログラムは、後で問題を発生させるよりはむしろ OPEN 実行時に失敗することがあります。OPEN はランタイム・メッセージ IGZ0035S またはファイル状況 39 (ファイル状況文節が指定されている場合) のいずれかが原因で失敗する可能性があります。 QSAM ファイルでのファイル状況 39 の防止についての詳細は、313 ページの『付録 G. 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。

固定ファイル属性の矛盾に関する問題に共通する原因は、ご使用のファイルでの JCL DD ステートメントの DCB= パラメーターです。

メッセージ: これらの違いについての **MIGR** メッセージはありません。これは、固定ファイル属性がソース・プログラムの外側で指定されることがあるためです。

JCL の DCB= パラメーターに関する推奨事項:

システムでブロック・サイズを判別できるようにする DFSMS および COBOL の機能を活用することを強くお勧めします。 (通常、『*Enterprise COBOL プログラミング・ガイド*』に記載されているケース以外では、DCB= 属性を指定しないでください。)

以下に、推奨事項を示します。

- 新規ファイルでは、z/OS によってブロック・サイズを判別します。システムで判別されたブロック・サイズを活用するには、以下のようにします。
  - コード BLOCK CONTAINS 0 をソース・プログラムにコーディングするか、または BLOCK0 コンパイラー・オプションを使用します。
  - ご使用のソース・プログラムで RECORD CONTAINS 0 をコード化しないでください。
  - JCL DD ステートメントで BLKSIZE 値をコード化しないでください。
- 既存のブロック化されたデータ・セットの場合は、次の既存のファイル・ブロック・サイズを使用します。
  - コード BLOCK CONTAINS 0 をソース・プログラムにコーディングするか、または BLOCK0 コンパイラー・オプションを使用します。
  - DD 名定義で BLKSIZE 値をコード化しないでください。

JCL に BLKSIZE を置くことを検討する必要のある事例は、新規ファイルに特定のブロック・サイズを必要とし、そのブロック・サイズを使用中のプログラムを再コンパイルせずに変更する柔軟性が必要な場合です。この場合、以下のガイドラインに従ってください。

- コード BLOCK CONTAINS 0 をソース・プログラムにコーディングするか、または BLOCK0 コンパイラー・オプションを使用します。
- DD 名定義で BLKSIZE 値 (JCL DD ステートメントの DCB=(BLKSIZE=*xxx*)) をコード化してください。

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、プログラムに上記のステートメントのいずれも含まれない場合、コンパイラー警告診断メッセージが出されて、プログラムを分析し、そのプログラムが存在しうるものであるかどうかを検証するように提案されます。

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側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
プログラムで「STOP RUN」、「GOBACK」、または「EXIT PROGRAM」が見つかりませんでした。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、次のメッセージが発行されます。

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」が実行されます。
「OPTIMIZE」および「FLAGMIG」コンパイラー・オプションで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。

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)

CMPR2 と NOCMPR2 では、OPEN ステートメントに対する QSAM ファイルの固定ファイル属性の処理方法が異なります。

CMPR2: ファイルの OPEN を正常に実行するために、QSAM ファイルの固定ファイル属性を COBOL プログラム・ファイル定義、JCL、またはデータ・セット・ラベルと一致させる必要はありません。

NOCMPR2: 以下の項目が矛盾していると、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。

- データ・セット・ラベルからのファイルの固定ファイル属性
- ファイルについて JCL DD ステートメント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矛盾していると、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。

メッセージ: この違いについての **MIGR** メッセージはありません。これは、固定ファイル属性がソース・プログラムの外側で指定されることがあるためです。

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) の修正処置: よくあるファイル状況 39 の問題を防止するには、313 ページの『付録 G. 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)

CMPR2 と NOCMPR2 では、IDCAMS に関連した VSAM ファイル内で定義済みの RECORDSIZE を処理する方法が異なります。

CMPR2 では、ファイルの OPEN を正常に実行するために、IDCAMS に関連した VSAM ファイル内で定義済みの RECORDSIZE をご使用の COBOL プログラム・ファイル定義に一致させる必要はありませんでした。

CMPR2: ファイルの OPEN を正常に実行するために、IDCAMS に関連した VSAM ファイル内で定義済みの RECORDSIZE をご使用の COBOL プログラム・ファイル定義に一致させる必要はありませんでした。

NOCMPR2: ファイルの OPEN を正常に実行するには、IDCAMS に関連した VSAM ファイル内で定義済みの RECORDSIZE をご使用の COBOL プログラムのファイルに設定されたファイル定義に一致させる必要があります。

メッセージ: この違いについての **MIGR** メッセージはありません。これは、VSAM RECORDSIZE 属性がソース・プログラムの外側にあるためです。

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) の修正処置: プログラム・ファイル定義、または IDCAMS に関連した VSAM ファイル内で定義された RECORDSIZE のいずれかを変更して、以下の表の規則に一致させてください。以下の規則が VSAM ESDS、KSDS、および RRDS ファイル定義に適用されます。

表 23. 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。 *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。 *n* は COBOL レコードの最大サイズ以上です。

PERFORM から戻る方法

CMPR2 と NOCMPR2 では、修正措置を必要とするライン外 PERFORM ステートメントを処理する方法が異なります。

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、指定範囲の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".
```

メッセージ「Hi」を表示した後、コンパイラー生成コードは段落 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側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、以下の条件がすべて当てはまる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側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、テスト「N > 1」の結果に応じて、プログラムはただちに呼び出し側プログラムに戻るか、PERFORM に戻り N にゼロを移送してから呼び出し側プログラムに戻るか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、テスト「N > 1」の結果に応じて、プログラムはただちに呼び出し側プログラムに戻るか、段落 B に進んでメッセージを表示して処理を続けるか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 「EXIT PROGRAM」または「GOBACK」ステートメントの場合、「NOCMPR2」コンパイラー・オプションのもとでは、「PERFORM」範囲の終わりに達したとみなされます。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 「EXIT PROGRAM」または「GOBACK」ステートメントの場合、「NOCMPR2」コンパイラー・オプションのもとでは、「PERFORM」範囲の終わりに達したとみなされます。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 (LINE xx.xx)」で、「PERFORM」ステートメントの範囲内の「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

PERFORM ステートメントの VARYING 句における特定の ID は、CMPR2 が有効か NOCMPR2 が有効かに応じて、異なる方法で設定および増分されます。

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

AFTER 句を指定した PERFORM...VARYING だけが影響を受けます。

例えば、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 の訂正処置: 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。いずれかが索引名である場合は、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。例えば、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

PICTURE 文節内に記号 B があるデータ項目は、CMPR2 が有効か NOCMPR2 が有効かに応じて、英字項目または英字編集項目のいずれかとして扱われます。

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発行されます。

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発行されます。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ステートメント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".
```

重要: 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 の受信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。例えば、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。この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).
       または
01 CALLDN PIC X(7).
       または

01 CALLDN PIC AAAAABB
01 CALLDN1 REDEFINES CALLDN PIC A(7).

    MOVE "PROG1" TO CALLDN1.
    CALL CALLDN1.
    CANCEL CALLDN1.
```

PROGRAM COLLATING SEQUENCE

PROGRAM COLLATING SEQUENCE 文節によって判別される非数値の比較の真理値は、CMPR2 と NOCOMPR2 では異なることがあります。

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」ステートメントには影響を与えません。

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 と RETURN INTO

INTO 句を指定した READ (または RETURN) は、記述の少なくとも 1 つが数字または数字編集である複数の 01 レベルのレコード記述がある、固定長ファイルの CMPR2 と NOCMPR2 では、異なる方法で実行されることがあります。

コンパイラーは、暗黙の MOVE ステートメントの送信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、複数レコードを含む固定形式ファイル「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、次のメッセージが発行されます。

IGYPS2283-W

MIGR 「READ」または「RETURN」ステートメントの「INTO」句が、複数レコードを含むファイル「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 および 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。例えば、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に違いがあるとします。以下のステートメントを考えてください。

```
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」ではなく「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 が送信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) を送信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信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す。

その動作は、CMPR2 と NOCMPR2 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 および FILE2 の両方が可変長レコードを持つようになります。

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。

SET . . . TO TRUE

SET...TO TRUE は、CMPR2 が有効か NOCMPR2 が有効かによって異なる影響を与えます。

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、以下のように、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 = " / ")
```

SIZE ERROR、MULTIPLY と DIVIDE の

SIZE ERROR の動作は、CMPR2 が有効か NOCMPR2 が有効かによって異なります。

74 COBOL 標準および 85 COBOL 標準では、COMPUTE、DIVIDE、または MULTIPLY の各ステートメントに複数の受け取りフィールドがある場合、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記されています。例えば、
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。

中間結果の使用および定義については、「*Enterprise COBOL プログラミング・ガイド*」を参照してください。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、74 COBOL 標準規則に従って、36 桁の (即時) 結果が 30 桁 (または 31 桁) の (中間) 結果に移動されると、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、次のメッセージが発行されます。

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

UNSTRING ステートメントに関連した添え字付け、指標付け、および長さ計算では、CMPR2 が有効か NOCMPR2 が有効かによって、異なる結果が生成されることがあります。

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。

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。

例えば、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 オペランド内の受信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 オペランドの受信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 オペランド内の受信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 オペランド内の受信側として使用されている。

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 および送信側、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、標準 COBOL 74 と 85 COBOL 標準のいずれでも正しくないため、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。

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、以下の例のように、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」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 の受信側の 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 句内の受信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 受信側の添え字が最初の INTO 句の COUNT IN 受信側によって変更されるためです。さらに、3 番目の INTO 受信側の長さは、2 番目の INTO 句の COUNT IN 受信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、以下のルール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 句内の受信側に依存していなければなりません。

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 受信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 受信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 句内の受信側に依存していなければなりません。

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 受信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

UPSI スイッチの条件名は、CMPR2 が有効か NOCMPR2 が有効かに応じて、異なる方法で定義および参照する必要があります。

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

送信または受信 ODO オブジェクトの長さの計算は、CMPR2 が有効か NOCMPR2 が有効かによって異なる可能性があります。

CMPR2: グループ移動 (MOVE ステートメントなど) に関係する送信フィールドおよび受信フィールド内の 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。送信側および受信側の実際の長さは、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: 受信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、受信側が可変長であり、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 . . .

可変長グループ移行に対する修正処置:: 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、複合 ODO 項目ではない受信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信側を変更して、参照変更を使用して受信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).
```

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 ではサポートされません。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。

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、『サポートされない SOM ベースの OO COBOL 言語エレメント』および 156 ページの『変更された SOM ベースの OO 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 OO COBOL 言語エレメント

SOM ベースの OO プログラミングを使用する COBOL アプリケーションを Enterprise COBOL で Java ベースの OO プログラミングに移行する際には、以下のサポートされない SOM エレメントに注意してください。

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。例えば、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 OO COBOL 言語エレメント

SOM ベースの OO プログラミングを使用するアプリケーションを Enterprise COBOL で Java ベースの OO プログラミングに移行する際には、Enterprise COBOL で変更されている以下のエレメントに注意してください。

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。

第 10 章 IBM COBOL プログラムのコンパイル

このセクションには、以下のトピックに関する情報が記載されています。

- IBM COBOL からのデフォルトのコンパイラー・オプションの変更
- IBM COBOL プログラム用のコンパイラー・オプション
- Enterprise COBOL で使用できないコンパイラー・オプション

IBM COBOL または Enterprise COBOL に関する特定の情報が記載されています。

IBM COBOL プログラム用のデフォルトのコンパイラー・オプション

Enterprise COBOL コンパイラーには、IBM COBOL とはわずかに異なるコンパイラー・オプションがあります。 IBM から配送される際の製品構成では、コンパイラー・オプションは DBCS、FLAG(I,I)、RENT および XREF(FULL) がデフォルト値です。 IBM COBOL のデフォルトは NODBCS、 FLAG(I)、 NORENT および NOXREF でした。

COBOL2 CICS 変換プログラムのオプションを使用している場合は、DBCS オプションにより CICS プログラムに問題が発生することがあります。 COBOL3 変換プログラムのオプションを使用して修正してください。

IBM COBOL プログラム用のコンパイラー・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使用しているのと同じコンパイラー・オプションを使用するのであれば、いくつかの内部的な変更が影響することがありますが、基本的には動作は変わりません。

IBM COBOL アプリケーションで使用していた設定と異なるコンパイラー・オプションを設定する場合は、アプリケーションに与える可能性のある影響を十分考慮してください。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 には、IBM COBOL のコンパイラー・オプションに比較して、新しいコンパイラー・オプションがいくつかあります。表 24 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 24. IBM COBOL プログラム用のコンパイラー・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 24. IBM COBOL プログラム用のコンパイラー・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 Language Environment の開始日付を使用します。この開始日付は Language Environment 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 (例えば、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(DBCS) は、IBM COBOL および VS COBOL II の前のリリースとの互換性があります。

表 24. IBM COBOL プログラム用のコンパイラー・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。 例えば、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 で使用できないコンパイラー・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、以下のコンパイラー・オプションを除いて、Enterprise COBOL のコンパイルでも使用できます。

表 25. Enterprise COBOL で使用できないコンパイラー・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 でコンパイルするために、85 COBOL 標準に変換する必要があります。
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 25. *Enterprise COBOL* で使用できないコンパイラー・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。
NUMPROC(MIG)	Enterprise COBOL は、バージョン 4 より後のバージョンでは NUMPROC(MIG) オプションをサポートしていません。 NUMPROC(MIG) が指定された場合、Enterprise COBOL は警告メッセージを発行し、コンパイルで NUMPROC のデフォルト設定が取得されます。これは、ユーザーがカスタマイズしたデフォルト、または IBM デフォルト (NUMPROC(NOPFD)) のいずれかです。
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。 IGYCNOOO 予約語テーブルは、Enterprise COBOL コンパイラーに付属していません。

第 11 章 Enterprise COBOL バージョン 3 からプログラムのアップグレード

Enterprise COBOL バージョン 5 でコンパイルを行うには、いくつかの機能のいずれかを使用する Enterprise COBOL バージョン 3 プログラムの変更が必要になる場合があります。

以下のいずれかの言語機能を含むプログラムは、変更が必要になる場合があります。

- SEARCH ALL のあるプログラム
- XML PARSE を使用するプログラム
- XML GENERATE を使用するプログラム
- ユーザー・ワードとして新規予約語を使用するプログラム。詳細については、102 ページの『新しい予約語』を参照してください。
- SIMVRD 機能を使用するプログラム
- LABEL 宣言。形式 2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS (オプション) が含まれるプログラム。これらのプログラムのサポートは Enterprise COBOL バージョン 5 では廃止されました。
- DATE FORMAT とウィンドウ化日付関数を使用するプログラム。詳細については、182 ページの『IBM Enterprise COBOL for z/OS バージョン 5 における 2000 年言語拡張の変更』を参照してください。

SEARCH ALL ステートメント

SEARCH ALL ステートメントが含まれ、APAR PK16765 の PTF がインストールされる前の Enterprise COBOL V3R4、または V3R1 から V3R3 でコンパイルされたプログラムがある場合は、この情報を参照してください。

ヒント: Enterprise COBOL V3R4 コンパイラーにこの PTF がインストールされているかどうかは、コンパイラー・リストのページ・ヘッダーを見れば分かります。この PTF によってモディフィケーション・レベルが 0 から 1 に変更されたので、ヘッダー中の製品名が「Enterprise COBOL for z/OS 3.4.1」のようになっていれば、コンパイラーにこの PTF がインストールされています。

SEARCH ALL ステートメントを含むプログラムのアップグレード

Enterprise COBOL では、SEARCH ALL ステートメントのインプリメンテーションでのエラーが修正されています。以前のリリースの COBOL の SEARCH ALL ステートメントには、キー比較論理にエラーがありました。このエラーが、意図していたものとは異なる結果を招きました。特に、比較では IF ステートメントまたは順次 SEARCH ステートメントと同じ結果が作成されませんでした。

長さの不一致の問題: 検索引数がテーブル・キーより長くなる

SEARCH ALL ステートメントの比較では、キーが SEARCH 引数より短いと、英数字キーにはブランクが埋め込まれ、数字キーには先行ゼロが加えられます。ただし、V3R3 とそれ以前のリリースでは場合によっては、SEARCH ALL で引数の余分の文字が無視されました。例えば、「ABCDEF」を含む `01 ARG PIC X(6)` の英数字検索引数は、値「ABCD」をともなう `05 MY-KEY PIC X(4)` のテーブルまたは配列のキーと、誤って一致してしまいます。「ABCD 」(ブランクあり) を含む検索引数は期待通りに一致となります。

数字の検索引数およびキーの場合にも同様の問題が発生しました。例えば、「123456」を含む`01 ARG PIC 9(6)` の検索引数は、値「3456」をともなう `05 MY-KEY PIC 9(4)` のテーブルまたは配列のキーと、誤って一致してしまいます。003456 を含む検索引数は期待通りに一致となります。

符号の不一致の問題: 符号付き数字引数と符号なし数字キー

第 2 の問題が発生するのは、検索引数が符号付き数字項目で、テーブル・キーが符号なし数字項目の場合です。検索引数のランタイム値が -1234 などの負数である場合、V3R3 と以前のリリースでコンパイルされたプログラムでは、1234 のテーブル・キーが一致となります。こうした比較は、通常の COBOL 比較条件の規則を使用して行われると、-1234 などの負の引数は符号なしのテーブル・キーと一致することはありません。

修正処置:

Enterprise COBOL ではこれらの問題は修正されました。ただし、以前のリリースでコンパイルされたアプリケーションには、間違った動作に依存するものもあります。これらのアプリケーションを特定し、修正してから、 Enterprise COBOL バージョン 4 以降に移行する必要があります。

こうした修正の影響を受けるプログラムおよび SEARCH ALL ステートメントの特定に役立つように、以下のコンパイラーおよびランタイム警告診断が出されます。

- コンパイラー・メッセージ: Enterprise COBOL コンパイラーは、以下のコンパイラー診断を生成します。実際に影響があるかどうかは、実行時の引数の内容によります。
  - IGYPG3189-W。テーブル・キーより長い検索引数を持つ (つまり第 1 の問題が生じる可能性がある) すべての SEARCH ALL ステートメントについて出されます。
  - IGYPG3188-W。検索引数が符号付き数字項目で、テーブル・キーが符号なし数字項目である (つまりプログラムで第 2 の問題が生じる可能性がある) 場合に出されます。
- ランタイム・メッセージ : 以下のランタイム・メッセージが生成されます。これらのランタイム・メッセージのいずれかを生成するプログラムは、変更の影響を受けます。
  - IGZ0194W。余分のバイトがブランクまたはゼロでない検索引数を持つすべての SEARCH ALL ステートメントについて出されます。
  - IGZ0193W。検索引数が負の値の符号付き数字項目で、テーブル・キーは符号なし数字項目の場合に出されます。

移行の手順

アプリケーションを Enterprise COBOL バージョン 4 以降に移行するには、以下の一連の手順のいずれかを行います。

- コンパイラー・メッセージについて処置を行います。
  1. プログラムを Enterprise COBOL でコンパイルします。
  2. コンパイラー・メッセージ IGYPG3188-W または IGYPG3189-W で示された SEARCH ALL ステートメントを検討します。このようなステートメントは影響を受けている可能性があります。

     ヒント: 非互換の結果が生じる可能性を最小化するために、これらのメッセージの重大度を E または S に変更することによって、これらの SEARCH ALL ステートメントの修正をプログラマーに強制することができます。メッセージの重大度を変更するには、EXIT コンパイラー・オプションの MSGEXIT サブオプションを使用します。この変更処置を行うと、これらのメッセージが発生するプログラムは、コードが修正されるまで実行できなくなります。IGYPG3188-W および IGYPG3189-W の重大度をそれぞれ IGYPG3188-S と IGYPG3189-S に変更するサンプル・コードが、サンプル・ユーザー出口 IGYMSGXT に含まれています。
- ランタイム・メッセージについて処置を行います。
  1. アプリケーションをテスト環境で実行します。
  2. ランタイム・メッセージ IGZ0193W または IGZ0194W を生成する SEARCH ALL ステートメントを検討します。

影響を受ける SEARCH ALL ステートメントを特定したら、以下の手順でアプリケーション・ロジックを適切に調整します。

- 検索引数がテーブル・キーより長い SEARCH ALL ステートメントの場合、以下のいずれかの処置を行います。
  - 必ず、キーの長さを超えている引数のバイトが、スペースまたはゼロになるようにします。

    ヒント: この調査を完了し、プログラムを変更しないことに決めた場合、EXIT コンパイラー・オプションの MSGEXIT サブオプションを使用して、IGYPG3188-W および IGYPG3189-W の重大度をそれぞれ IGYPG3188-I と IGYPG3189-I に変更するか、またはこれらのメッセージを完全に抑制することができます。そうすると、プログラムは RC=0 でコンパイルされるようになります。サンプル・ユーザー出口 IGYMSGXT に、IGYPG3188-W および IGYPG3189-W の重大度を変更するサンプル・コードが含まれています。
  - 引数をキーと同サイズの一時データ項目に移し、その一時項目を検索引数として使用します。
  - 参照/修正によって比較範囲を制限します。以下に例を示します。
    - 検索引数 `01 ARG PIC X(6)` の英数字文字および `05 MY-KEY PIC X(4)` のキーでは、以下を使用します。

      ```
      WHEN MY-KEY (MY-INDEX) = ARG(1:4)
      ```

    - 検索引数 `01 ARG PIC 9(6)` の数字および `05 MY-KEY PIC 9(4)` の配列のキーでは、以下を使用します。

      ```
      WHEN MY-KEY (MY-INDEX) = ARG(3:4)
      ```

上記の 2 番目と 3 番目の処置によって、将来警告メッセージは表示されなくなります。

- 検索引数が符号付きで、テーブル・キーが符号なしである SEARCH ALL ステートメントの場合、必ず検索引数を正数値に正しく初期設定してから、SEARCH ステートメントを実行します。COBOL プログラム内の固有のアプリケーション・ロジックによっては、以下のいずれかの変更を行うことができます。
  - 引数のデータ記述を符号なしに変更します。
  - 検索引数を符号なしの一時変数に移し、その一時変数を SEARCH ALL ステートメントで使用します。

  いずれかの処置によって、将来警告メッセージは表示されなくなります。

XML PARSE ステートメントを含む Enterprise COBOL バージョン 3 プログラムのアップグレード

XML PARSE ステートメントがある Enterprise COBOL バージョン 3 プログラムをアップグレードするには、この情報を参照してください。

Enterprise COBOL バージョン 4 には、Enterprise COBOL バージョン 3 に比べて新しい XML PARSE サポートが導入されました。特に、z/OS System Services XML パーサーは、COBOL ランタイム・ライブラリーの一部である XML パーサーに対するデフォルトの代替としてサポートされるようになりました。バージョン 5 では、COBOL ランタイム・ライブラリー・パーサーと XML System Services パーサーのいずれかを選択できます。

当初、Enterprise COBOL V5.1 には XMLPARSE コンパイラー・オプションはなく、XMLSS パーサーが必要でした。しかし、最新のサービスが適用されていれば、V5.1 はこの点に関して V5.2 と同等であり、いずれにも XMLPARSE コンパイラー・オプションがあるため、旧バージョンの Enterprise COBOL で使用していたものと同じパーサーを V5 で使用できます。

- XMLPARSE(COMPAT) は、コンパイルされたコードが COBOL ランタイム・ライブラリー・パーサーを使用することを指定します。

  ほとんどの場合、XML PARSE ステートメントを持つ Enterprise COBOL バージョン 3 プログラムを変更して Enterprise COBOL バージョン 5 にアップグレードする必要はありません。 XMLPARSE(COMPAT) コンパイラー・オプションを指定することにより、互換性のある動作を得ることができます。ただし、Enterprise COBOL バージョン 5.2 での COMPAT XML パーサー実装は、バージョン 3 での実装とは異なる場合があります。変更は大半の既存プログラムには影響しませんが、相違が発生し得るこのようなまれなケースについて検討しておく必要があります。詳細については、 167 ページの『COMPAT XML パーサーの考慮事項』を参照してください。
- XMLPARSE(XMLSS) は、コンパイルされたコードが z/OS System Services XML パーサーを使用するように指定します。

  XMLPARSE(XMLSS) を使用するように変更したい場合は、XML PARSE ステートメントを使用する Enterprise COBOL バージョン 3 プログラムを変更する必要があります。

z/OS System Services XML パーサーには以下の利点があります。

- 最新の IBM 構文解析テクノロジー
- COBOL XML 構文解析を zAAP 専用プロセッサーにオフロードします。
- XML ネーム・スペースを使用する XML 文書の構文解析に対するサポートが改良されました。
- UTF-8 Unicode でエンコードされた XML 文書の構文解析が直接サポートされます。
- 大きな XML 文書の構文解析 (一度に 1 つのバッファー) がサポートされます。

オプションとして、Enterprise COBOL バージョン 5 で XMLPARSE(XMLSS) を使用するように Enterprise COBOL バージョン 3 プログラムを変更するには、新しい、変更された、および廃止された XML 構文解析イベントを反映するようにプログラムを変更します。詳細については、325 ページの『付録 L. XMLPARSE(COMPAT) から XMLPARSE(XMLSS) へのマイグレーション』を参照してください。

COMPAT XML パーサーの考慮事項

XML PARSE ステートメント実行中の XML 文書に対するユーザー変更

Enterprise COBOL V5 より前のバージョンでは、XML 処理プロシージャーの実行中には COMPAT XML パーサーがアクティブに進行していました。 V5 では、エンコード競合があれば解決され、その後に文書全体が構文解析され、XML イベントがバッファーに保管されます。構文解析の終了後、処理プロシージャーを実行する PERFORM ステートメントによって、XML イベントがこのバッファーからプログラムに送られます。このため、プログラムが処理プロシージャー・コード内で XML 文書を変更する場合、パーサーはこれらの変更を検出しません。旧バージョンの実装環境では、開始タグ名に合わせた終了タグ名の修正などの変更は、パーサーによって検出され、処理されていました。

継続可能 XML EXCEPTION イベント数の制限

範囲が 1 から 49 の XML-CODE 値を持つ XML EXCEPTION イベントの場合、XML-CODE をゼロに設定することによって継続を要求すると、COMPAT XML パーサーは、それ以降のエラーについては、検査を行うだけで XML EXCEPTION イベントを提示しません。 V5 COMPAT XML パーサーが EXCEPTION イベント後に継続されると、パーサーは XML バッファーを拡張しないため、XML バッファーを拡張する場合は発生する可能性のあるすべての EXCEPTION イベントを提示しない場合があります。初期バッファー・サイズでは、少なくとも 8192 個の XML イベントを収容できます。このサイズは、必要に応じて EXCEPTION 以外のイベント用に拡張することができます。

LE 条件処理による相違

Enterprise COBOL V5 より前のバージョンでは、処理プロシージャーは、アクティブ XML パーサーのスタック・フレームに従属するスタック・フレーム内で実行されていました。 V5 COMPAT パーサーの処理プロシージャーは、XML パーサーが

完了するまで実行された後、残りの COBOL プログラムと同じスタック・フレームで実行されます。この変更により、以下のような影響があります。

- 以前は、XML 処理プロシージャーに登録されている LE 条件ハンドラーは、COMPAT XML PARSE ステートメントが終了すると有効ではなくなっていました。 V5 実装環境では、これらのハンドラーは登録抹消されるまで有効です。
- 以前は、XML 処理プロシージャーの外部で設定された LE サービス再開点に分岐すると、COMPAT XML PARSE ステートメントが終了していました。 V5 では、XML PARSE ステートメントを終了するには、処理プロシージャーが正常終了する必要があります。そうしないと、プログラムが終了した場合 (IGZ0227S)、または別の XML PARSE ステートメントが実行された場合 (IGZ0228S) に、既にアクティブな XML PARSE ステートメントによってランタイム・エラーが発生します。

以下のプログラムで、この相違点を説明します。前述のとおり、このプログラムは Enterprise COBOL V5 より前のバージョンでは「正しく」実行されますが、Enterprise COBOL V5 ではランタイム・エラー IGZ0227S または IGZ0228S を引き起こします。 XML 処理プロシージャー内の示されたステートメントのコメントを外すと、プログラムはすべてのバージョンでエラーなく実行されます。

XMLPARSE(COMPAT) の処理

```
*****************************************************************
***  Function:                                                ***
***  Demonstate a difference between XML PARSE COMPAT on      ***
***  V3/V4 and V5 (or XMLSS on any version).                  ***
***                                                           ***
***  In V3/4, the logical branch out of the XML processing    ***
***  procedure by CEEMRCE terminates the XML PARSE. In V5,    ***
***  it does not, resulting in runtime messages such as:      ***
***    IGZ0227S There was an invalid attempt to end an        ***
***             XML PARSE statement.                          ***
***  when the program terminates (or attempts another parse).***
*****************************************************************
 Identification division.
   Program-id. XMLMIGR1.
 Data division.
  Working-storage section.
   1 XML-document pic x(4) value '<x/>'.
   1 zer0 comp pic 9 value 0.
  Local-storage section.
   1 routine procedure-pointer.
   1 token pointer.
   1 ceesrp-data.
    2 resume-point comp pic s9(9).
    2 state pic x value 'I'.
   1 fdbk-code.
    2 condition-token-value.
      88 fdbk-code-zero value low-value.
     3 pic xx.
     3 msg-no comp pic s9(4).
     3 pic x(4).
    2 pic x(4).
 Procedure division.  Main section.
     Perform register-user-handler
     Call 'CEE3SRP' using resume-point fdbk-code
     Service label.
   Repeat.
     If state = 'I'
       XML parse XML-document processing procedure XML-proc
       Display 'Back from XML parse...'
```

```
|                           Go to Repeat
|                         Else
|                           If state = 'R'
|                             Display 'Resumed after exception; in mainline code.'
|                           End-if
|                           Perform unregister-user-handler
|                           Display 'Another XML parse (P), or exit (E)?'
|                           Accept state
|                           If state = 'P'
|                             Move '<y/>' to XML-document
|                             XML parse XML-document processing procedure XML-proc.
|                         Goback.
|                        Register-user-handler.
|                          Set routine to entry 'USERHDLR'
|                          Set token to address of ceesrp-data
|                          Call 'CEEHDLR' using routine token fdbk-code
|                          If fdbk-code-zero
|                            Display 'Registered exception handler successfully.'
|                          Else
|                            Display 'Failed to register exception handler!' msg-no
|                            Move 16 to return-code
|                            Stop run.
|                        Unregister-user-handler.
|                          Set routine to entry 'USERHDLR'
|                          Call 'CEEHDLU' using routine fdbk-code
|                          If fdbk-code-zero
|                            Display 'Unregistered exception handler successfully.'
|                          Else
|                            Display 'Failed to unregister exception handler!' msg-no
|                            Move 16 to return-code
|                            Stop run.
|                       XML-proc section.
|                          Display XML-event '{' XML-text '}'
|                          If XML-event = 'START-OF-DOCUMENT'
|                            Display 'XML parse in progress...'
|                            Move 1 to xml-code
|                            Go to xp-srp.
|                          If XML-event = 'START-OF-ELEMENT' and XML-text = 'x'
|                            Compute tally = 1 / zer0.
|                          Go to xp-exit.
|                        Xp-srp.
|                       *** Uncomment the next two lines to move the resume point to ***
|                       *** within the XML processing procedure, thus allowing the   ***
|                       *** XML PARSE statement to terminate normally and correctly. ***
|                       *    Call 'CEE3SRP' using resume-point fdbk-code
|                       *    Service label
|                            If state = 'R'
|                              Display 'Resumed after exception; still in XML-proc.'
|                              Move 'X' to state.
|                        Xp-exit.
|                          Continue.
|                       End program XMLMIGR1.
|
|                       ****************************************************************
|                       *** LE user condition handler, invoked when the fixed-point  ***
|                       *** divide exception occurs (system completion code S0C9).   ***
|                       ****************************************************************
|                        Identification division.
|                          Program-id. USERHDLR.
|                        Data division.
|                         Working-storage section.
|                          1 fdbk-code.
|                           2 condition-token-value pic x(8).
|                             88 fdbk-code-zero value low-value.
|                           2 pic x(4).
|                         Linkage section.
|                          1 ceesrp-data.
```

```
          2 resume-point comp pic s9(9).
          2 state pic x.
        1 token pointer.
        1 result comp pic s9(9).
         88 resume value 10.
        1 curr-cond pic x(12).
        1 new-cond pic x(12).
      Procedure division using curr-cond token result new-cond.
          Display 'LE condition handler called...'
          Set address of ceesrp-data to token
          Call 'CEEMRCE' using resume-point fdbk-code
          If not fdbk-code-zero display 'Unable to resume execution!'
          Else Set resume to true Move 'R' to state.
          Goback.
      End program USERHDLR.
```

XML GENERATE ステートメントを含む Enterprise COBOL プログラムのアップグレード

Enterprise COBOL では、Enterprise COBOL バージョン 3 より後に 5 つの新しい XML GENERATE 例外コードが導入されました。

これらの例外コードを使用するプログラムは、新しいバージョンの Enterprise COBOL に移行するには変更する必要があります。

Enterprise COBOL に追加された XML GENERATE 例外コードは以下のとおりです。

415 受信側は国別だが、文書に指定されたエンコードが UTF-16 ではありませんでした。

416 XML 名前空間 ID に無効な XML 文字が含まれていました。

417 エレメント文字内容または属性値に、XML コンテンツでは正しくない文字が含まれていました。 XML の生成は続行されます。エレメント・タグ名または属性名には接頭部「hex.」が付けられ、元のデータ値は文書内では 16 進数で表されます。

418 置換文字がエンコード変換で生成されました。

419 XML 名前空間接頭部が無効でした。

新しい予約語を使用するプログラムの移行

Enterprise COBOL バージョン 3 以降、いくつかの予約語が追加されています。

ご使用のプログラムで新しい予約語のいずれかがユーザー定義語 (データ項目名や段落名など) として使用されている場合は、これらの語を変更する必要があります。 CCCA と類似の内容を行うことができ、また単に -85 などの接尾部を語のすべてのインスタンスに追加することができます。以下に例を示します。

```
77 VOLATILE PIC S9(9) BINARY.
Move 0 TO VOLATILE.
```

Enterprise COBOL V5 でコンパイルするには、これを次のように変更します。

```
77 VOLATILE-85 PIC S9(9) BINARY.
Move 0 TO VOLATILE-85.
```

新規の予約語は次のとおりです。

- VOLATILE
- XML-INFORMATION
- XML-NAMESPACE
- XML-NAMESPACE-PREFIX
- XML-NNAMESPACE
- XML-NNAMESPACE-PREFIX
- XML-SCHEMA

APAR PM86253 の PTF が Enterprise COBOL バージョン 5.1 に対してインストールされている場合、または APAR PI32750 の PTF が Enterprise COBOL バージョン 5.2 に対してインストールされている場合、変換ツール CCCA はこれらの予約語を自動的に変換します。 CCCA は、IBM Debug Tool 製品に組み込まれています。

異なる COBOL コンパイラーすべての予約語を比較する表は、 252 ページの表 37 を参照してください。

SIMVRD サポートを使用するプログラムのアップグレード

このセクションでは、SIMVRD サポートを使用するプログラムをアップグレードするための処置について説明します。 *COBOL シミュレート可変長相対レコードデータ・セット (RRDS)* は、Enterprise COBOL バージョン 4 以降でコンパイルされたプログラムではサポートされません。これらのファイルは VSAM RRDS ファイルに変更する必要があります。

Enterprise COBOL バージョン 4 より前の、NOCMPR2 コンパイラー・オプションをサポートする COBOL コンパイラーでは、SIMVRD ランタイム・オプション・サポートを使用する場合、VSAM KSDS を使って *COBOL シミュレート可変長 RRDS* が使用できました。

COBOL プログラムで VSAM 可変長 RRDS および COBOL シミュレート可変長 RRDS を特定および記述するために使用するコーディングは、同様です。Enterprise COBOL バージョン 4 では、VSAM 可変長 RRDS サポートを使用しなければなりません。一般に、COBOL シミュレート可変長 RRDS から VSAM 可変長 RRDS サポートへマイグレーションするのに必要な処置は、ファイルの IDCAMS 定義を変更することだけです。

表 26. 可変長 RRDS を使用するステップ

ステップ	VSAM 可変長 RRDS	COBOL シミュレート可変長 RRDS
1	ORGANIZATION IS RELATIVE 節でファイルを定義します。	同じ
2	FD 項目を使って可変長サイズでレコードを記述します。	同じ。ただし、データ・セットにアクセスするすべての COBOL プログラムの FD 項目の中に、RECORD IS VARYING をコーディングしなければなりません。

表 26. 可変長 RRDS を使用するステップ (続き)

ステップ	VSAM 可変長 RRDS	COBOL シミュレート可変長 RRDS
3	NOSIMVRD ランタイム・オプションを使用します。	SIMVRD ランタイム・オプションを使用します。

4	アクセス方式サービス・プログラムを通じて、VSAM ファイルを RRDS として定義します。	アクセス方式サービス・プログラムを通じて、VSAM ファイルを以下のように定義します。
		`DEFINE CLUSTER INDEXED`
		`KEYS(4,0)`
		`RECORDSIZE(avg,m)`
		avg COBOL レコードの平均サイズです。必ず*m* 未満でなければなりません。
		m 最大サイズ COBOL レコード + 4 以上です。

シミュレート可変長 RRDS のステップ 2 で、可変長レコード・フォーマットを暗示するほかの言語エレメントをコーディングしても、COBOL シミュレート可変長 RRDS にはなりません。例えば、以下のエレメントだけでは、シミュレート可変長 RRDS アクセスを使用しませんでした。したがって SIMVRD ランタイム・オプションは必要ありませんでした。

- 長さが異なる複数の FD レコード
- レコード定義内の OCCURS . . . DEPENDING ON
- RECORD CONTAINS *integer-1* TO *integer-2* CHARACTERS

レコードを含み、出力用にオープンするファイルに対して、REUSE IDCAMS パラメーターを使用します。

- ORGANIZATION IS RELATIVE 節でファイルを定義します。
- FD 項目を使って可変長サイズでレコードを記述します。
- NOSIMVRD ランタイム・オプションを使用します。
- アクセス方式サービス・プログラムを通じて、VSAM ファイルを RRDS として定義します。

エラー: シミュレート可変長相対データ・セットと真の VSAM RRDS データ・セットを処理する場合、COBOL ファイル定義と VSAM データ・セット属性が一致しないと OPEN ファイル状況 39 が発生します。

可変長 RRDS を使用するためのコマンドに関する詳しい説明については、*z/OS DFSMS: カタログ用アクセス方式サービス・プログラム*を参照してください。

第 12 章 Enterprise COBOL バージョン 3 プログラムのコンパイル

Enterprise COBOL バージョン 3 以降、コンパイラー・オプションおよびデバッグ動作がいくつか変更されています。

以下のトピックを読み終えたら、 189 ページの『第 15 章 IBM Enterprise COBOL for z/OS バージョン 5 における変更』も参照してください。

IBM Enterprise COBOL for z/OS バージョン 3 からのコンパイラー・オプションの変更

コンパイラー・オプションに多くの変更が行われました。

以下のオプションは廃止されました。

表 27. Enterprise COBOL バージョン 5 で使用できないコンパイラー・オプション

コンパイラー・オプション	説明
DATEPROC	2000 年問題対応用の拡張のサポートが廃止されました。
NOLIB	コンパイラーは、LIB が常に有効であるかのように動作します。
YEARWINDOW	2000 年問題対応用の拡張のサポートが廃止されました。
SIZE(MAX)	SIZE オプションは削除されました。
NUMPROC(MIG)	NUMPROC(PFD) および NUMPROC(NOPFD) は引き続き使用可能です。 NUMPROC(MIG) が指定された場合、Enterprise COBOL V5 は警告メッセージを発行し、コンパイルで NUMPROC のデフォルト設定が取得されます。これは、ユーザーがカスタマイズしたデフォルト、または IBM デフォルト (NUMPROC(NOPFD)) のいずれかです。

また、ランタイム・オプション `CHECK(OFF)` または `NOSSRANGE` を使用して、SSRANGE コンパイル範囲検査を実行時に無効にすることはできないことに注意してください。

Enterprise COBOL V5 の新しいオプションおよび変更されたオプションの説明については、 191 ページの『Enterprise COBOL バージョン 5 におけるコンパイラー・オプションの変更』を参照してください。

さまざまなコンパイラー・バージョンでサポートされるオプションの詳細なリストについては、 285 ページの『付録 E. オプションの比較』を参照してください。

TEST コンパイラー・オプションの相違

このセクションには、プログラムをアップグレードし、TEST コンパイラー・オプションを使用してコンパイルする際に知っておく必要がある、TEST コンパイラー・オプションの変更内容に関する情報があります。 Enterprise COBOL バージョン 5 の TEST コンパイラー・オプションは、以前のコンパイラーに比べて単純化

されています。 COBOL ソースの JCL または CBL/PROCESS ステートメントで TEST オプションが指定されている場合は、それらのテスト・オプションを変更してください。以下の TEST サブオプションは削除されましたが、一部は移行を容易にするために引き続き受け入れられます。そのオプションが使用された場合は、コンパイラー診断メッセージが発行されます。削除されたサブオプションは、同じ TEST オプション指定の中で新しいサブオプションとともに指定することはできません。

表 28. 削除された TEST サブオプション

削除されたサブオプション	コンパイラーで指定された場合の動作	診断メッセージのレベルまたはカテゴリー
ALL	診断が発行され、フックはオブジェクトで生成されません	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))
BLOCK	診断が発行され、フックはオブジェクトで生成されません	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))
PATH	診断が発行され、フックはオブジェクトで生成されません	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))
STMT	診断が発行され、フックはオブジェクトで生成されません	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))
NONE	診断が発行され、フックはオブジェクトで生成されません	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))
SYM	診断が発行され、シンボリック・デバッグ情報が必ず生成されます	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))
NOSYM	診断が発行され、シンボリック・デバッグ情報が必ず生成されます	エラー (オプション診断が無効です。このオプションは廃棄されます (Invalid option diagnostic, option discarded))

表 28. 削除された TEST サブオプション (続き)

削除されたサブオプション	コンパイラーで指定された場合の動作	診断メッセージのレベルまたはカテゴリー
HOOK	診断が発行され、フックはオブジェクトで生成されません	NOHOOK の動作に関する通知メッセージが必ず発行されます (サブオプションは許容されます。TEST は有効です (Suboption tolerated, TEST in effect))
NOHOOK	診断が発行され、フックはオブジェクトで生成されません	NOHOOK の動作に関する通知メッセージが必ず発行されます (サブオプションは許容されます。TEST は有効です (Suboption tolerated, TEST in effect))
SEPARATE	診断が発行され、デバッグ情報がオブジェクト・モジュールで必ず生成されます	NOSEPARATE の動作に関する通知メッセージが必ず発行されます (サブオプションは許容されます。TEST は有効です (Suboption tolerated, TEST in effect))
NOSEPARATE	診断が発行され、デバッグ情報がオブジェクト・モジュールで必ず生成されます	NOSEPARATE の動作に関する通知メッセージが必ず発行されます (サブオプションは許容されます。TEST は有効です (Suboption tolerated, TEST in effect))

注: 旧 TEST サブオプションはいずれも、インストール・デフォルト・オプションを設定するために IGYCDOPT で指定された場合は認識されません。

IBM Enterprise COBOL バージョン 5 におけるデバッグ情報の変更

IBM Enterprise COBOL バージョン 5 を使用してコンパイルされたプログラムのデバッグ情報は、以前のバージョンのコンパイラーを使用してコンパイルされたプログラムの動作とは異なります。

IBM Enterprise COBOL バージョン 5 では、デバッグ情報に関するジレンマが解決されます。従来は、以下の 2 つの選択肢がありました。

- 大きなロード占有スペースという代償を払っても、デバッグ・データを常に実行可能ファイルと一緒に保持する
- アプリケーションと同期し続けて必要なとき検索するという難題もあるが、デバッグ・データを分離して保持する

現在は、この両方をしのぐ状況にあります。プログラム・オブジェクトに NOLOAD デバッグ・セグメントを含めた場合、ロードされるプログラムのサイズがデバッグ・データによって増えることはなく、デバッグ・データは常に実行可能ファイルと合致し、いつでも使用できるため、データ・セットのリスト内を検索する必要がありません。

デバッグ可能バージョンのアプリケーションを生成するために使用される TEST コンパイラー・オプション、および NOTEST オプションが変更されました。

- TEST オプションを指定すると、DWARF デバッグ情報がアプリケーション・モジュールに含められます。
- SOURCE サブオプションが指定されている場合、DWARF デバッグ情報に拡張ソース・コードが含まれるため、IBM Debug Tool でコンパイラー・リストが不要になります。TEST(NOSOURCE) コンパイラー・オプションが指定されていると、生成された DWARF デバッグ情報に拡張ソース・コードは含まれません。
- NOTEST(DWARF) コンパイラー・オプションを使用すれば、基本 DWARF デバッグ情報をプログラム・オブジェクトに組み込むことができます。このようなプログラムは Debug Tool ではデバッグできませんが、それでも NOTEST 最適化を行い、CEEDUMP 出力や IBM Fault Analyzer などのアプリケーション障害分析ツールを有効化することはできます。
- プログラム・オブジェクトにデバッグ情報を含めないようにするには、NOTEST(NODWARF) オプションを使用します。

COBOL プログラムのデバッグ時に、向上した点および動作の変更が数多く Enterprise COBOL V5 に導入されていることが分かります。IBM Debug Tool でのデバッグの変更点について詳しくは、217 ページの『IBM Enterprise COBOL バージョン 5 における Debug Tool の変更』を参照してください。

第 13 章 Enterprise COBOL バージョン 4 からのアップグレード

Enterprise COBOL バージョン 5 でコンパイルするには、いくつかの機能を使用する Enterprise COBOL バージョン 4 プログラムのアップグレードが必要になる場合があります。

以下のいずれかの言語機能を含むプログラムは、変更が必要になる場合があります。

- XML PARSE と XMLPARSE(COMPAT) を併用するプログラム。
- XML PARSE と XMLPARSE(XMLSS) を併用する V4R1 プログラム。
- DATE FORMAT とウィンドウ化日付関数を使用するプログラム。詳細については、182 ページの『IBM Enterprise COBOL for z/OS バージョン 5 における 2000 年言語拡張の変更』を参照してください。
- LABEL 宣言。 Enterprise COBOL V5 でプログラムをコンパイルするには、形式 2 宣言構文 USE...AFTER...LABEL PROCEDURE... および構文 GO TO MORE-LABELS をすべて削除する必要があります。これらのプログラムのサポートは Enterprise COBOL バージョン 5 では廃止されました。
- ユーザー・ワードとして新規予約語を使用するプログラム。詳細については、102 ページの『新しい予約語』を参照してください。

Enterprise COBOL V5 への移行を支援するために、Enterprise COBOL V4.2 の APAR PM93450 により新規コンパイラー・オプション FLAGMIG4 が使用可能になります。Enterprise COBOL V4 プログラムに含まれている言語エレメントがサポートされなくなったものであったり、Enterprise COBOL V5 では異なる意味でサポートされるものであったりする場合に、FLAGMIG4 オプションは、そのような言語エレメントを特定します。コンパイラーは、そのようなすべての言語エレメントに対して警告診断メッセージを生成します。

XML PARSE ステートメントを含む Enterprise COBOL バージョン 4 プログラムのアップグレード

XML PARSE ステートメントを含む Enterprise COBOL バージョン 4 プログラムのアップグレードについては、以下のガイドラインを参照できます。

旧バージョンの Enterprise COBOL で COBOL ランタイム・ライブラリーの一部である COMPAT XML パーサーまたは z/OS System Services XML パーサーのどちらを使用していた場合でも、おそらく Enterprise COBOL バージョン 5 用にコード変更を行う必要はありません。

Enterprise COBOL バージョン 4.2 で z/OS System Services XML パーサーを使用していた場合、Enterprise COBOL バージョン 5.2 用にコード変更を行う必要はありません。当初、Enterprise COBOL V5.1 には XMLPARSE コンパイラー・オプションはなく、XMLSS パーサーが必要でした。しかし、最新のサービスが適用されていれば、V5.1 はこの点に関して V5.2 と同等であり、いずれにも XMLPARSE コンパ

イラー・オプションがあるため、旧バージョンの Enterprise COBOL で使用していたものと同じパーサーを V5 で使用できます。

Enterprise COBOL バージョン 4.1 で z/OS System Services XML パーサーを使用していた場合は、181 ページの『XML PARSE ステートメントを含み XMLPARSE(XMLSS) コンパイラー・オプションを使用する Enterprise COBOL バージョン 4 リリース 1 プログラムのアップグレード』の情報を検討してください。

Enterprise COBOL バージョン 4 およびバージョン 3 で COBOL ランタイム・ライブラリーの一部である COMPAT XML パーサーを使用する場合、おそらくコードを変更する必要はありません。 Enterprise COBOL バージョン 5 での COMPAT XML パーサー実装には、バージョン 3 およびバージョン 4 に比較すると、特殊なケースにおいて 2 つのわずかな相違点があります。このため、相違が発生し得るこのようなまれなケースに対して、特別な考慮事項を検討する必要があります。詳細については、167 ページの『COMPAT XML パーサーの考慮事項』を参照してください。

COMPAT XML パーサーの考慮事項

XML PARSE ステートメント実行中の XML 文書に対するユーザー変更

Enterprise COBOL V5 より前のバージョンでは、XML 処理プロシージャーの実行中には COMPAT XML パーサーがアクティブに進行していました。 V5 では、エンコード競合があれば解決され、その後に文書全体が構文解析され、XML イベントがバッファーに保管されます。構文解析の終了後、処理プロシージャーを実行する PERFORM ステートメントによって、XML イベントがこのバッファーからプログラムに送られます。このため、プログラムが処理プロシージャー・コード内で XML 文書を変更する場合、パーサーはこれらの変更を検出しません。旧バージョンの実装環境では、開始タグ名に合わせた終了タグ名の修正などの変更は、パーサーによって検出され、処理されていました。

継続可能 XML EXCEPTION イベント数の制限

範囲が 1 から 49 の XML-CODE 値を持つ XML EXCEPTION イベントの場合、XML-CODE をゼロに設定することによって継続を要求すると、COMPAT XML パーサーは、それ以降のエラーについては、検査を行うだけで XML EXCEPTION イベントを提示しません。 V5 COMPAT XML パーサーが EXCEPTION イベント後に継続されると、パーサーは XML バッファーを拡張しないため、XML バッファーを拡張する場合は発生する可能性のあるすべての EXCEPTION イベントを提示しない場合があります。初期バッファー・サイズでは、少なくとも 8192 個の XML イベントを収容できます。このサイズは、必要に応じて EXCEPTION 以外のイベント用に拡張することができます。

LE 条件処理による相違

Enterprise COBOL V5 より前のバージョンでは、処理プロシージャーは、アクティブ XML パーサーのスタック・フレームに従属するスタック・フレーム内で実行されていました。 V5 COMPAT パーサーの処理プロシージャーは、XML パーサーが

完了するまで実行された後、残りの COBOL プログラムと同じスタック・フレームで実行されます。この変更により、以下のような影響があります。

- 以前は、XML 処理プロシージャーに登録されている LE 条件ハンドラーは、COMPAT XML PARSE ステートメントが終了すると有効ではなくなっていました。 V5 実装環境では、これらのハンドラーは登録抹消されるまで有効です。
- 以前は、XML 処理プロシージャーの外部で設定された LE サービス再開点に分岐すると、COMPAT XML PARSE ステートメントが終了していました。 V5 では、XML PARSE ステートメントを終了するには、処理プロシージャーが正常終了する必要があります。そうしないと、プログラムが終了した場合 (IGZ0227S)、または別の XML PARSE ステートメントが実行された場合 (IGZ0228S) に、既にアクティブな XML PARSE ステートメントによってランタイム・エラーが発生します。

以下のプログラムで、この相違点を説明します。前述のとおり、このプログラムは Enterprise COBOL V5 より前のバージョンでは「正しく」実行されますが、Enterprise COBOL V5 ではランタイム・エラー IGZ0227S または IGZ0228S を引き起こします。 XML 処理プロシージャー内の示されたステートメントのコメントを外すと、プログラムはすべてのバージョンでエラーなく実行されます。

XMLPARSE(COMPAT) の処理

```
*****************************************************************
***  Function:                                                ***
***  Demonstate a difference between XML PARSE COMPAT on      ***
***  V3/V4 and V5 (or XMLSS on any version).                  ***
***                                                           ***
***  In V3/4, the logical branch out of the XML processing    ***
***  procedure by CEEMRCE terminates the XML PARSE. In V5,    ***
***  it does not, resulting in runtime messages such as:      ***
***    IGZ0227S There was an invalid attempt to end an        ***
***             XML PARSE statement.                          ***
***  when the program terminates (or attempts another parse).***
*****************************************************************
 Identification division.
   Program-id. XMLMIGR1.
 Data division.
  Working-storage section.
   1 XML-document pic x(4) value '<x/>'.
   1 zer0 comp pic 9 value 0.
  Local-storage section.
   1 routine procedure-pointer.
   1 token pointer.
   1 ceesrp-data.
    2 resume-point comp pic s9(9).
    2 state pic x value 'I'.
   1 fdbk-code.
    2 condition-token-value.
      88 fdbk-code-zero value low-value.
     3 pic xx.
     3 msg-no comp pic s9(4).
     3 pic x(4).
    2 pic x(4).
 Procedure division.  Main section.
     Perform register-user-handler
     Call 'CEE3SRP' using resume-point fdbk-code
     Service label.
   Repeat.
     If state = 'I'
       XML parse XML-document processing procedure XML-proc
       Display 'Back from XML parse...'
```

```
|                               Go to Repeat
|                             Else
|                               If state = 'R'
|                                 Display 'Resumed after exception; in mainline code.'
|                               End-if
|                               Perform unregister-user-handler
|                               Display 'Another XML parse (P), or exit (E)?'
|                               Accept state
|                               If state = 'P'
|                                 Move '<y/>' to XML-document
|                                 XML parse XML-document processing procedure XML-proc.
|                             Goback.
|                            Register-user-handler.
|                              Set routine to entry 'USERHDLR'
|                              Set token to address of ceesrp-data
|                              Call 'CEEHDLR' using routine token fdbk-code
|                              If fdbk-code-zero
|                                Display 'Registered exception handler successfully.'
|                              Else
|                                Display 'Failed to register exception handler!' msg-no
|                                Move 16 to return-code
|                                Stop run.
|                            Unregister-user-handler.
|                              Set routine to entry 'USERHDLR'
|                              Call 'CEEHDLU' using routine fdbk-code
|                              If fdbk-code-zero
|                                Display 'Unregistered exception handler successfully.'
|                              Else
|                                Display 'Failed to unregister exception handler!' msg-no
|                                Move 16 to return-code
|                                Stop run.
|                           XML-proc section.
|                              Display XML-event '{' XML-text '}'
|                              If XML-event = 'START-OF-DOCUMENT'
|                                Display 'XML parse in progress...'
|                                Move 1 to xml-code
|                                Go to xp-srp.
|                              If XML-event = 'START-OF-ELEMENT' and XML-text = 'x'
|                                Compute tally = 1 / zer0.
|                              Go to xp-exit.
|                             Xp-srp.
|                           *** Uncomment the next two lines to move the resume point to ***
|                           *** within the XML processing procedure, thus allowing the   ***
|                           *** XML PARSE statement to terminate normally and correctly. ***
|                           *    Call 'CEE3SRP' using resume-point fdbk-code
|                           *    Service label
|                                If state = 'R'
|                                  Display 'Resumed after exception; still in XML-proc.'
|                                  Move 'X' to state.
|                             Xp-exit.
|                                Continue.
|                            End program XMLMIGR1.
|
|                           ****************************************************************
|                           *** LE user condition handler, invoked when the fixed-point  ***
|                           *** divide exception occurs (system completion code S0C9).   ***
|                           ****************************************************************
|                            Identification division.
|                              Program-id. USERHDLR.
|                            Data division.
|                             Working-storage section.
|                              1 fdbk-code.
|                               2 condition-token-value pic x(8).
|                                 88 fdbk-code-zero value low-value.
|                               2 pic x(4).
|                             Linkage section.
|                              1 ceesrp-data.
```

```
       2 resume-point comp pic s9(9).
       2 state pic x.
      1 token pointer.
      1 result comp pic s9(9).
       88 resume value 10.
      1 curr-cond pic x(12).
      1 new-cond pic x(12).
    Procedure division using curr-cond token result new-cond.
        Display 'LE condition handler called...'
        Set address of ceesrp-data to token
        Call 'CEEMRCE' using resume-point fdbk-code
        If not fdbk-code-zero display 'Unable to resume execution!'
        Else Set resume to true Move 'R' to state.
        Goback.
    End program USERHDLR.
```

XML PARSE ステートメントを含み XMLPARSE(XMLSS) コンパイラー・オプションを使用する Enterprise COBOL バージョン 4 リリース 1 プログラムのアップグレード

Enterprise COBOL バージョン 4 リリース 1 と Enterprise COBOL バージョン 4 リリース 2 以降では、XMLPARSE(XMLSS) コンパイラー・オプションが有効になっている場合の XML PARSE の動作が異なります。 Enterprise COBOL バージョン 4 リリース 1 では、XMLPARSE(XMLSS) コンパイラー・オプションを使用して XML 文書を構文解析する際に、その文書のエンコード方式で表すことができなかった文字参照が文書内に含まれていた場合、その結果として、単一の ATTRIBUTE-CHARACTERS または CONTENT-CHARACTERS XML イベントが発生し、表現不能な文字参照がそれぞれ 1 個のハイフン/マイナスで置き換えられました。そのような置換が発生したことはプログラムに通知されませんでした。

例えば、以下の XML エレメントの内容を構文解析するとします。

```
<elem>abc&#x1234;xyz</elem>
```

これを Enterprise COBOL バージョン 4 リリース 1 のもとで、エンコード方式 CCSID 1140 で、XMLPARSE(XMLSS) コンパイラー・オプションを指定して構文解析した場合、その結果は単一の CONTENT-CHARACTERS XML イベントであり、特殊レジスター XML-TEXT には次のような (EBCDIC) ストリングが含まれました。

```
abc-xyz
```

また、特殊レジスター XML-CODE の内容はゼロになりました。

Enterprise COBOL バージョン 4 リリース 2 以降では、XMLPARSE(XMLSS) コンパイラー・オプションを使用して XML 文書を構文解析すると、単一の ATTRIBUTE-CHARACTERS または CONTENT-CHARACTERS イベントではなく、複数の XML イベントが発生します。表現不能な各文字参照は、以前はハイフン/マイナスで置き換えられていましたが、どのようなコンテキストで発生したのかに応じて、ATTRIBUTE-NATIONAL-CHARACTER または CONTENT-NATIONAL-CHARACTER XML イベントで表されるようになりました。これらのイベントは、XMLPARSE(XMLSS) コンパイラー・オプションの XML イベントです。

前記の XML エレメントの内容を構文解析の例として再び使用します。

```
<elem>abc&#x1234;xyz</elem>
```

これを Enterprise COBOL バージョン 4 リリース 2 以降のもとで構文解析すると、以下の一連の XML イベントが発生します。

- XML-TEXT に abc が入っている CONTENT-CHARACTERS
- XML-NTEXT に NX'1234' が入っている CONTENT-NATIONAL-CHARACTER
- XML-TEXT に xyz が入っている CONTENT-CHARACTERS

新しい予約語を使用するプログラムの移行

Enterprise COBOL バージョン 4 以降、いくつかの予約語が追加されています。

ご使用のプログラムで新しい予約語のいずれかがユーザー定義語 (データ項目名や段落名など) として使用されている場合は、これらの語を変更する必要があります。 CCCA と類似の内容を行うことができ、また単に -85 などの接尾部を語のすべてのインスタンスに追加することができます。以下に例を示します。

```
77 VOLATILE PIC S9(9) BINARY.
Move 0 TO VOLATILE.
```

Enterprise COBOL V5 でコンパイルするには、これを次のように変更します。

```
77 VOLATILE-85 PIC S9(9) BINARY.
Move 0 TO VOLATILE-85.
```

新規の予約語は次のとおりです。

- VOLATILE
- XML-INFORMATION

APAR PM86253 の PTF が Enterprise COBOL バージョン 5.1 に対してインストールされている場合、または APAR PI32750 の PTF が Enterprise COBOL バージョン 5.2 に対してインストールされている場合、変換ツール CCCA はこれらの予約語を自動的に変換します。 CCCA は、IBM Debug Tool 製品に組み込まれています。

異なる COBOL コンパイラーすべての予約語を比較する表は、 252 ページの表 37 を参照してください。

IBM Enterprise COBOL for z/OS バージョン 5 における 2000 年言語拡張の変更

2000 年言語拡張はサポートされなくなりました。

削除されたエレメントは以下のとおりです。

- DATE FORMAT 文節
- DATEVAL 組み込み関数
- UNDATE 組み込み関数
- YEARWINDOW 組み込み関数
- DATEPROC コンパイラー・オプション
- YEARWINDOW コンパイラー・オプション

Enterprise COBOL バージョン 5 を使用してコンパイルを行うには、これらの言語エレメントを削除する必要があります。

第 14 章 Enterprise COBOL バージョン 4 プログラムのコンパイル

Enterprise COBOL バージョン 4 プログラムのコンパイラー・オプションおよびデバッグ動作がいくつか変更されています。

以下のトピックを読み終えたら、 189 ページの『第 15 章 IBM Enterprise COBOL for z/OS バージョン 5 における変更』も参照してください。

IBM Enterprise COBOL for z/OS バージョン 4 からのコンパイラー・オプションの変更

コンパイラー・オプションに多くの変更が行われました。

以下のオプションは廃止されました。

表 29. Enterprise COBOL バージョン 5 で使用できないコンパイラー・オプション

コンパイラー・オプション	説明
DATEPROC	2000 年問題対応用の拡張のサポートが廃止されました。
NOLIB	コンパイラーは、LIB が常に有効であるかのように動作します。
YEARWINDOW	2000 年問題対応用の拡張のサポートが廃止されました。
SIZE	SIZE オプションは削除されました。
NUMPROC(MIG)	NUMPROC(PFD) および NUMPROC(NOPFD) は引き続き使用可能です。 NUMPROC(MIG) が指定された場合、Enterprise COBOL V5 は警告メッセージを発行し、コンパイルで NUMPROC のデフォルト設定が取得されます。これは、ユーザーがカスタマイズしたデフォルト、または IBM デフォルト (NUMPROC(NOPFD)) のいずれかです。

また、ランタイム・オプション `CHECK(OFF)` または `NOSSRANGE` を使用して、コンパイル範囲検査 (SSRANGE コンパイラー・オプションでコンパイルされたプログラムの場合) を実行時に無効にすることはできないことに注意してください。

Enterprise COBOL バージョン 5 の新規オプションおよび変更オプションについては、 191 ページの『Enterprise COBOL バージョン 5 におけるコンパイラー・オプションの変更』を参照してください。

さまざまなコンパイラー・バージョンでサポートされるオプションの詳細なリストについては、 285 ページの『付録 E. オプションの比較』を参照してください。

すべてのオプションの詳細な説明については、「*Enterprise COBOL for z/OS プログラミング・ガイド*」を参照してください。

IBM Enterprise COBOL バージョン 5 におけるデバッグ情報の変更

IBM Enterprise COBOL バージョン 5 を使用してコンパイルされたプログラムのデバッグ情報は、以前のバージョンのコンパイラーを使用してコンパイルされたプログラムの動作とは異なります。

IBM Enterprise COBOL バージョン 5 では、デバッグ情報に関するジレンマが解決されます。従来は、以下の 2 つの選択肢がありました。

- 大きなロード占有スペースという代償を払っても、デバッグ・データを常に実行可能ファイルと一緒に保持する
- アプリケーションと同期し続けて必要なとき検索するという難題もあるが、デバッグ・データを分離して保持する

現在は、この両方をしのぐ状況にあります。プログラム・オブジェクトに NOLOAD デバッグ・セグメントを含めた場合、ロードされるプログラムのサイズがデバッグ・データによって増えることはなく、デバッグ・データは常に実行可能ファイルと合致し、いつでも使用できるため、データ・セットのリスト内を検索する必要がありません。

デバッグ可能バージョンのアプリケーションを生成するために使用される TEST コンパイラー・オプション、および NOTEST オプションが変更されました。

- TEST オプションを指定すると、DWARF デバッグ情報がアプリケーション・モジュールに含められます。
- SOURCE サブオプションが指定されている場合、DWARF デバッグ情報に拡張ソース・コードが含まれるため、IBM Debug Tool でコンパイラー・リストが不要になります。TEST(NOSOURCE) コンパイラー・オプションが指定されていると、生成された DWARF デバッグ情報に拡張ソース・コードは含まれません。
- NOTEST(DWARF) コンパイラー・オプションを使用すれば、基本 DWARF デバッグ情報をプログラム・オブジェクトに組み込むことができます。このようなプログラムは Debug Tool ではデバッグできませんが、それでも NOTEST 最適化を行い、CEEDUMP 出力や IBM Fault Analyzer などのアプリケーション障害分析ツールを有効化することはできます。
- プログラム・オブジェクトにデバッグ情報を含めないようにするには、NOTEST(NODWARF) オプションを使用します。

COBOL プログラムのデバッグ時に、向上した点および動作の変更が数多く Enterprise COBOL V5 に導入されていることが分かります。IBM Debug Tool でのデバッグの変更点について詳しくは、217 ページの『IBM Enterprise COBOL バージョン 5 における Debug Tool の変更』を参照してください。

第 4 部 Enterprise COBOL バージョン 5 での新機能および相違点

Enterprise COBOL V5 を使用する場合は、以前のすべてのコンパイラーと異なる点がいくつかあり、それらを考慮する必要があります。現在使用しているコンパイラーからプログラムやアプリケーションを移行する作業に関するセクションを読んだ後に、このセクションを読んでください。

最新サービスが適用されている場合、Enterprise COBOL V5.1 は移行目的では Enterprise COBOL V5.2 と同等です。

第 15 章 IBM Enterprise COBOL for z/OS バージョン 5 における変更

Enterprise COBOL V5 では、コンパイラーおよびランタイム・ライブラリーに多くの変更が行われました。コンパイル、バインド (リンク・エディット)、実行、および Debug Tool の動作が変更されました。

変更は以下のカテゴリーに分類されます。

1. 前提ソフトウェアおよび前提サービス
2. ソース・コードの相違
3. コンパイラー・オプションの相違
4. コンパイル動作の相違
5. バインド (リンク・エディット) の相違
6. 実行時の相違
7. デバッグ情報および Debug Tool 動作の変更

Enterprise COBOL バージョン 5 の前提ソフトウェアおよび前提サービス

他の製品で Enterprise COBOL V5 を使用してプログラムをコンパイルし、またそれらのプログラムをバインド、実行、およびデバッグするには、更新が必要です。現在、Enterprise COBOL V5 では、FIXCAT を使用して、必要なサービスを見つけることができます。

関連ソフトウェア・プロダクトの前提条件レベル

Enterprise COBOL V5 とともに以下の製品を使用するには、以下の製品が以下のレベルになっていなければなりません。

- z/OS V1R13 以降
- CICS Transaction Server for z/OS V3 以降
- IBM DB2 V9 以降
- IBM IMS V11 以降
- PD Tools V12 以降 (Debug Tool、Fault Analyzer、Application Performance Analyzer)
- Rational Developer for System z V9 (RDz) 以降

必要なサービスを判別

PSP バケットで APAR や PTF のリストを探す必要はなくなりました。 Enterprise COBOL for z/OS V5.1 以降では、SMP/E FIXCAT を使用して、Enterprise COBOL V5 と連携する他の製品で必要な PTF を特定する必要があります。COBOL for z/OS V5 に必要となるサービス PTF は、この移行ガイドには記載されていませんし、PSP バケットに含まれていませんし、いずれの会議資料にも記載されていません。

SMP/E FIXCAT を使用すれば、Enterprise COBOL V5 の必要なサービスに関して最も正確な最新情報を取得できます。これは、必要なサービス PTF がすべてインストールされているかどうかを素早く判別するための最も簡単な方法です。 Enterprise COBOL V5 では、FIXCAT HOLDDATA に対する SMP/E V3R5 以降のサポートを使用して、プログラマチック・ターゲット・システム PTF 検証を行う必要があります。このような PTF は、拡張 HOLDDATA において、`IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R1` という FIXCAT で特定されます。

HOLDDATA タイプ FIXCAT (修正カテゴリー) は、必要なターゲット・システム PTF 用の修正の特定カテゴリーに APAR を関連付けるために使用されます。現行システムにおいて Enterprise COBOL V5 にアップグレードするために必要ではあるが、まだインストールされていない PTF を特定しやすくするためには、SMP/E **REPORT MISSINGFIX** コマンドを使用します。以下に、z/OS CSI に対して実行するために使用されるサンプル・コマンドを示します。

```
SET BDY(GLOBAL).
REPORT MISSINGFIX ZONES(ZOS13T)
FIXCAT(IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R1)
```

REPORT MISSINGFIX コマンドの完全な情報については、「*z/OS SMP/E コマンド*」を参照してください。

Enterprise COBOL V5 に移行するための Enterprise COBOL V4.2 援助プログラム

旧バージョンの Enterprise COBOL 用の修正は、FIXCAT では処理されません。以下の APAR 修正には、Enterprise COBOL V4.2 から Enterprise COBOL V5 への移行を支援するための援助プログラムが含まれています。

- PM93450 - FLAGMIG4: V5 でサポートされていない COBOL ステートメントがあるかどうかを識別するために役立ちます。
- PM85035 - XML-INFORMATION 特殊レジスターをサポートする新機能。XMLPARSE(XMLSS) に移行する (結果として V5 に移行する) ときに役立ちます。
- Language Environment V1.13 PM87347 これは、関連 Enterprise COBOL V4 APAR である PM85035 がインストールされた場合の実行時の XML-INFORMATION サポート用です。

Enterprise COBOL バージョン 5 における COBOL ソース・コードの相違

IBM Enterprise COBOL for z/OS V5 では、いくつかの言語エレメントが削除または変更されているため、ソース・プログラムを更新しなければならない場合があります。

2000 年言語拡張はサポートされなくなりました。ご使用のプログラムに以下のいずれかの言語エレメントが含まれている場合は、Enterprise COBOL V5 を使用してプログラムをコンパイルおよび実行するために、これらの言語エレメントを削除する必要があります。

- DATE FORMAT 文節
- DATEVAL 組み込み関数

- UNDATE 組み込み関数
- YEARWINDOW 組み込み関数

LABEL 宣言のサポートが変更されました。ご使用のプログラムに以下のいずれかの言語エレメントが含まれている場合は、Enterprise COBOL V5 を使用してプログラムをコンパイルおよび実行するために、これらの言語エレメントを削除する必要があります。

- 形式 2 宣言構文 `USE...AFTER...LABEL PROCEDURE...` はサポートされなくなりました。
- 構文 `GO TO MORE-LABELS` はサポートされなくなりました。

Enterprise COBOL V5.2 では、VOLATILE は新しい予約語です。 VOLATILE をユーザー定義語 (データ名や段落名など) として使用する既存のプログラムは、Enterprise COBOL V5.2 で S レベルの診断メッセージを受け取ります。VOLATILE のこれらのインスタンスは、VOLATILE-X などの他の語に変更する必要があります。あるいは、CCCA ユーティリティーを使用してこれを自動的に行うことができます。

Enterprise COBOL バージョン 5 におけるコンパイラー・オプションの変更

Enterprise COBOL V5 では、コンパイラー・オプションに対していくつか変更が行われました。

使用可能なオプションは次のとおりです。

表 30. Enterprise COBOL バージョン 5 の新規コンパイラー・オプション

コンパイラー・オプション	説明
AFP	新しいオプションです。 z/Architecture プロセッサーに備わる追加浮動小数点 (AFP) レジスターのコンパイラーでの使用方法を制御します。AFP(VOLATILE) がデフォルトです。
ARCH	新しいオプションです。実行可能プログラム命令の生成対象となるマシン・アーキテクチャーを指定します。 • Enterprise COBOL V5.1 では、ARCH(6) がデフォルトです。 • Enterprise COBOL V5.2 では、ARCH(6) は受け入れられなくなりました。新しい上位レベルの ARCH(11) が受け入れられ、ARCH(7) がデフォルトです。
COPYRIGHT	Enterprise COBOL V5.2 での新しいオプションです。オブジェクト・モジュールが生成される場合に、オブジェクト・モジュールにストリングを配置します。
DISPSIGN	新しいオプションです。符号付き数値項目の DISPLAY の出力形式設定を制御します。DISPSIGN(COMPAT) がデフォルトです。
HGPR	新しいオプションです。 z/Architecture プロセッサーに備わる 64 ビット・レジスターのコンパイラーでの使用方法を制御します。HGPR(PRESERVE) がデフォルトです。

表 30. *Enterprise COBOL バージョン 5 の新規コンパイラー・オプション (続き)*

コンパイラー・オプション	説明
MAXPCF	新しいオプションです。プログラムに *n* より大きい複雑度の因数が含まれている場合に、コードを最適化しないようコンパイラーに指示します。
QUALIFY	Enterprise COBOL V5.2 での新しいオプションです。修飾規則に作用し、COBOL 標準規則の下で参照できない一部のデータ項目を参照できるように修飾規則を拡張するかどうかを制御します。
SERVICE	Enterprise COBOL V5.2 での新しいオプションです。オブジェクト・モジュールが生成される場合に、オブジェクト・モジュールにストリングを配置します。
SQLIMS	新しいオプションです。新しい IMS SQL コプロセッサー (IMS では SQL ステートメント・コプロセッサーと呼ばれます) を有効にします。この新しいコプロセッサーは、組み込み SQLIMS ステートメントが入っているソース・プログラムを処理します。 当初、基本レベルの Enterprise COBOL V5.1 には SQLIMS オプションはありませんでしたが、最新サービスが適用されていると、V5.1 は V5.2 と同等となり、SQLIMS オプションを備えるようになります。
STGOPT	新しいオプションです。ストレージの最適化が制御されます。NOSTGOPT がデフォルトです。
VLR	新しいオプションです。長さに矛盾が生じている可変長レコードの READ ステートメント処理に作用します。 VLR(COMPAT) がデフォルトです。 当初、基本レベルの Enterprise COBOL V5.1 には VLR オプションはなく、いくつかの移行上の問題がありましたが、最新サービスが適用されていると、V5.1 は V5.2 と同等となり、VLR オプションを備えるようになります。詳細については、 202 ページの『可変長レコード - 不正な長さの READ』を参照してください。
XMLPARSE	新しいオプションです。このオプションは、構文解析に COBOL ライブラリーの互換モード COBOL XML パーサーを使用するか、z/OS XML System Services パーサーを使用するかの選択を可能にします。これにより、Enterprise COBOL V5 コンパイラーへの移行が容易になります。XMLPARSE(XMLSS) がデフォルトです。 当初、基本レベルの Enterprise COBOL V5.1 には XMLPARSE オプションはなく、いくつかの移行上の問題がありましたが、最新サービスが適用されていると、V5.1 は V5.2 と同等となり、XMLPARSE オプションを備えるようになります。

以下のオプションは変更されています。

表 31. *Enterprise COBOL バージョン 5 で変更されたコンパイラー・オプション*

コンパイラー・オプション	説明
EXIT	EXIT コンパイラー・オプションは、DUMP コンパイラー・オプションと相互に排他的ではなくなり、コンパイラー出口規則が更新されました。

表 31. Enterprise COBOL バージョン 5 で変更されたコンパイラー・オプション (続き)

コンパイラー・オプション	説明
MAP	新しいサブオプション HEX および DEC が MAP コンパイラー・オプションに追加されました。このサブオプションは、コンパイラー・リストの MAP 出力に、16 進または 10 進のどちらのオフセットを表示するかを制御します。 旧バージョンの Enterprise COBOL は MAP 出力に常に 16 進オフセットを表示していましたが、Enterprise COBOL V5.1 は当初、MAP 出力に常に 10 進オフセットを表示していました。 Enterprise COBOL V5.1 はサービスによって変更され、新しいサブオプション HEX および DEC が MAP オプションに追加されました。 Enterprise COBOL V5.2 にも同様に、両方の新しい MAP サブオプションがあります。サービス適用済みの V5.1 および V5.2 ではいずれも、MAP がサブオプションなしで指定された場合は、MAP(HEX) として受け入れられます。 これにより、旧 COBOL コンパイラーと同じ動作が Enterprise COBOL V5 で得られます。このため、Enterprise COBOL V5 コンパイラーへの移行が容易になります。
MDECK	コンパイラーは LIB オプションが常に有効であるかのように動作するため、MDECK オプションは LIB オプションに依存しなくなりました。
NORENT	NORENT は RMODE(ANY) と一緒には使用できなくなりました。 16 MB 境界より上での NORENT プログラムの実行はサポートされません。
OPTIMIZE	OPTIMIZE オプションは、アプリケーションに対して、より多くのレベルのパフォーマンス最適化を使用できるように変更されました。以前の OPTIMIZE オプション・フォーマットは非推奨ですが、互換性のための使用は許容されます。 **注:** OPT(0) は、以前のコンパイラーの NOOPTIMIZE オプションと同等ですが、以前は削除されなかった一部のコードを削除します。 古い OPT の FULL サブオプションによって提供されていたストレージ最適化は、新しいコンパイラー・オプション STGOPT で提供されるようになりました。
RMODE(ANY)	RMODE(ANY) は NORENT と一緒には使用できなくなりました。
SSRANGE	ランタイム・オプション CHECK(OFF) または NOSSRANGE を使用しても、コンパイルされた範囲検査を実行時に無効にすることはできません。 **注:** コンパイラー・オプション NOSSRANGE はサポートされています。

表 31. Enterprise COBOL バージョン 5 で変更されたコンパイラー・オプション (続き)

コンパイラー・オプション	説明
TEST	TEST コンパイラー・オプションのサブオプション HOOK | NOHOOK および SEPARATE | NOSEPARATE は削除されました。これらのサブオプションが指定された場合、 • HOOK - コンパイル・フックは使用できません。 • NOHOOK - NOHOOK 動作が常に有効になります。 • SEPARATE - コンパイラーは常にデバッグ情報をオブジェクトに入れます。 • NOSEPARATE - NOSEPARATE 動作が常に有効になります。 新しいサブオプション SOURCE および NOSOURCE が TEST コンパイラー・オプションに追加されました。 **注:** EJPD および NOEJPD サブオプションはサポートされています。APAR PM75819 が適用された Debug Tool V12、または Debug Tool V13 以降では、TEST(NOEJPD) オプションおよびゼロ以外の OPTIMIZE レベルを指定してコンパイルを行った場合でも、JUMPTO や GOTO を実行できます。ただし、Debug Tool コマンド SET WARNING OFF を使用する必要があります。また、予測不能な結果が生じる可能性があります。 NOTEST オプションが拡張され、DWARF および NODWARF サブオプションが組み込まれました。 **注:** DWARF デバッグ情報が常に NOLOAD セグメントとしてオブジェクト・プログラムに入れられるとしても、DWARF デバッグ・データを使用する Debug Tool、CEEDUMP、Fault Analyzer、Application Performance Analyzer、またはサード・パーティー・ベンダー・ツールが使用されていなければ、その NOLOAD セグメントは実行時にストレージを使用しません。

以下のオプションは削除されました。

表 32. Enterprise COBOL バージョン 5 で使用できないコンパイラー・オプション

コンパイラー・オプション	説明
DATEPROC	2000 年問題対応用の拡張のサポートが廃止されました。
NOLIB	コンパイラーは、LIB が常に有効であるかのように動作します。
NUMPROC(MIG)	NUMPROC(PFD) および NUMPROC(NOPFD) は引き続き使用可能です。 NUMPROC(MIG) が指定された場合、Enterprise COBOL V5 は警告メッセージを発行し、コンパイルで NUMPROC のデフォルト設定が取得されます。これは、ユーザーがカスタマイズしたデフォルト、または IBM デフォルト (NUMPROC(NOPFD)) のいずれかです。

表 32. Enterprise COBOL バージョン 5 で使用できないコンパイラー・オプション (続き)

コンパイラー・オプション	説明
SIZE	• Enterprise COBOL V5.1 では、SIZE オプション値は、COBOL コンパイルで使用される合計ストレージの上限ではなくなりました。また、SIZE サブオプション値 MAX はサポートされなくなりました。SIZE オプションのデフォルト値は SIZE(5000000) です。コンパイラーのメモリー要件について詳しくは、『Enterprise COBOL バージョン 5 におけるコンパイルの変更』を参照してください。 • Enterprise COBOL V5.2 では、SIZE オプションは削除されました。
YEARWINDOW	2000 年問題対応用の拡張のサポートが廃止されました。

さまざまなコンパイラー・バージョンでサポートされるオプションの詳細なリストについては、285 ページの『付録 E. オプションの比較』を参照してください。

すべてのオプションの詳細な説明については、「*Enterprise COBOL for z/OS プログラミング・ガイド*」を参照してください。

Enterprise COBOL バージョン 5 におけるコンパイルの変更

IBM Enterprise COBOL for z/OS では、異なる動作となる変更がいくつか行われています。

現在は、COBOL ランタイム・ライブラリー (z/OS の Language Environment コンポーネント) をコンパイル時に使用できるようになっていなければなりません。また、Enterprise COBOL バージョン 5 でプログラムをコンパイルするための APAR 修正 (PTF)、および Enterprise COBOL バージョン 5 でコンパイルされたプログラムを実行するための APAR 修正 (PTF) で、言語環境プログラムを更新する必要があります。前提ソフトウェア・レベルと、必要な保守について詳しくは、189 ページの『Enterprise COBOL バージョン 5 の前提ソフトウェアおよび前提サービス』を参照してください。

コンパイル時のストレージ要件は、旧バージョンの Enterprise COBOL に比べて大幅に増加しています。コンパイラーを実行するには、最低でも 200 M の領域サイズが必要です。 Enterprise COBOL バージョン 5.1 では、コンパイラー・オプション SIZE(MAX) はサポートされなくなりましたが、使用することは可能であり、SIZE(5000K) と解釈されます。 SIZE オプション設定は、V5.1 では 5000 K から 20000 K の範囲内でなければなりません。

大きなプログラムのそれぞれに大きな SIZE 値を指定する必要はありません。コンパイルの間にエラー・メッセージ「IGYPG5062-U コンパイラー処理のためのストレージが不十分でした (THERE WAS INSUFFICIENT STORAGE FOR COMPILER PROCESSING)」を受け取った場合のみ、デフォルトの SIZE 値を大きくする必要があります。このメッセージは、まだプログラムを処理している間にコンパイラー・フロントエンドでメモリー不足が発生し、そのフロントエンド用にメモリーをさらに割り振るために、SIZE オプションを使用しなければならないことを示しています。

ただし、SIZE オプションでフロントエンドに割り振られたメモリーは、それ以降のコンパイルのフェーズでは使用できなくなることに注意してください。そのため、メモリーのコード生成および最適化ステップが妨げられないよう、慎重に SIZE 値を調整してください。このようにしないと、それ以降のフェーズで、メッセージ「IGYCB7145-U コンパイルを続行するためには、コンパイラーのメモリーが不十分です (INSUFFICIENT MEMORY IN THE COMPILER TO CONTINUE COMPILATION)」でコンパイラーが異常終了する場合があります。

Enterprise COBOL バージョン 5.2 では、コンパイラー・オプション SIZE はサポートされなくなりました。領域サイズは 200 M 以上でなければなりません。特に、より高い最適化レベル (つまり、OPT(1) または OPT(2) のコンパイラー・オプションでコンパイルされたプログラム) では、領域サイズを大きくする必要があります。

注: 予期せずコンパイラーが異常終了する場合、または「IEW4000I 使用可能なストレージが不足していたため、DD 名 STEPLIB からモジュール IGYCBE をフェッチできませんでした。(IEW4000I FETCH FOR MODULE IGYCBE FROM DDNAME STEPLIB FAILED BECAUSE INSUFFICIENT STORAGE WAS AVAILABLE.)」というメッセージが表示される場合は、領域サイズが 200 M 以上であることを確認してください。 JCL で REGION=0M が設定されていると、システム・プログラマーが設定した JES システム・デフォルトで許可されている最大量が割り当てられます。これでは足らない場合があります。その場合、システム・プログラマーは領域サイズの上限を増やす必要があります。

以下の変更も考慮してください。

- Enterprise COBOL バージョン 5 リリース 1 の言語環境プログラム・メンバー ID は 4 です (以前は、すべての COBOL バージョンでメンバー ID は 5 でした)。
- コンパイル時の CPU 時間要件は、以前のバージョンの Enterprise COBOL と比べると、大幅に増加しています。新しいコンパイラーは、古いコンパイラーに比べて、コンパイルに 5 倍より多くの時間がかかることがあります。
- コンパイル時と実行時の診断メッセージは異なる場合があり、異なるタイミングまたは場所で生成されることがあります。
  - 情報レベルおよび警告レベルの診断メッセージの有無が異なる場合があります。
  - サポートされていない過度の量のストレージを定義するプログラムの診断メッセージは、コンパイル時にコンパイラーによって出されずに、バインド時にバインダーによって出されるか、または実行時に言語環境プログラムによって出されることがあります。
- コンパイラー出力の形式は GOFF 形式です。この形式では、コンパイラーはより効率的な生成済みコードを作成でき、また NOLOAD デバッグ情報 (DWARF) セグメントを出力できます。
- デバッグ情報に対して SYSDEBUG データ・セットは作成されません。
- コンパイラー・リストのフォーマットと内容は、以前のバージョンの Enterprise COBOL のものとは異なります。この変更に関する詳細は、「*Enterprise COBOL for z/OS プログラミング・ガイド*」を参照してください。

- Enterprise COBOL V5 では、いくつかのコンパイラー限界値が増やされました。詳細については、 307 ページの『付録 F. コンパイラー限界値の比較』を参照してください。

初期化されていないデータ・セットへのコンパイラー出力はサポートされない

コンパイラーが、初期化されていないデータ・セットに書き込もうとして失敗するケースがいくつかあります。

順次データ・セット

Enterprise COBOL バージョン 4 以前では、コンパイラーは、前のコンパイル・ステップからの特定属性なしで、事前に割り振られたオブジェクト・ファイルに書き込みを行うことができました。 Enterprise COBOL バージョン 5 では、これはできなくなりました。

例えば、Enterprise COBOL バージョン 4 では、コンパイラーは、前のステップからの指定属性 (DISP=MOD) なしで、事前に割り振られたデータ・セットに書き込みを行うことができました。コンパイラーがそのデータ・セットに書き込みを行うと、次の属性がそのデータ・セットに付与されました。

```
RECFM=FB LRECL=80 BLKSIZE=3200 DSORG=PS
```

Enterprise COBOL バージョン 5 では、属性は変更されず、ファイルへの書き込みの試行は失敗します。

ファイル属性は次のようになります:

```
RECFM=U LRECL=** BLKSIZE=6144 DSORG=PS
```

これは、バインダーへの入力としては無効です。

これに対処するには、事前割り振り時にデータ制御ブロック (DCB) 情報を次のように指定します。

```
DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
```

PDS または PDSE データ・セット

以前のバージョンの Enterprise COBOL では、コンパイラーは、前のコンパイル・ステップからの特定属性なしで、事前に割り振られた PDS オブジェクト・ファイルに書き込みを行うことができました。 Enterprise COBOL バージョン 5 では、これはサポートされていません。

例えば、Enterprise COBOL バージョン 4 では、コンパイラーは、前のステップからの指定属性 (DISP=MOD) なしで、事前に割り振られた PDS または PDSE に書き込みを行うことができました。今後、コンパイラーは、属性のオブジェクト・ファイルを作成します。

```
RECFM=FB LRECL=80 BLKSIZE=3200 DSORG=PO
```

Enterprise COBOL バージョン 5 では、PDS/PDSE データ・セットに対して DISP=MOD はサポートされません。

PDS に不定形式 (DCB を持たない前のステップからの出力など) が含まれている場合に、DISP=SHR または DISP=OLD を使用すると、Enterprise COBOL バージョン 5 は書き込みを行いますが、属性は変更しません。属性は次のままとなります:

```
RECFM=U LRECL=** BLKSIZE=6144 DSORG=PO
```

これは、バインダーへの入力としては無効です。

これを修正するには、割り振りステップにおいて次のように DCB 情報を指定します:

```
DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
```

DISP=MOD は使用しないでください。 DISP=SHR または DISP=OLD のみを使用してください。

Enterprise COBOL バージョン 5 における JCL およびパッケージ化の変更

Enterprise COBOL V5 では、パッケージ化、インストール、および JCL にいくつかの変更が行われています。

SIGYCOMP データ・セットは、以前のバージョンでは PDS データ・セットでしたが、現在は PDSE データ・セットになりました。

Enterprise COBOL V5 では追加のデータ・セットが必要になります。

- z/OS TSO またはバッチでコンパイルする場合、COBOL コンパイラーには現在、SYSUT1 から SYSUT15 までの 15 個のユーティリティー・データ・セットが必要です。
- SYSMDECK データ・セットは、すべてのコンパイルに必要となりました。 NOMDECK オプションが指定されている場合、SYSMDECK をユーティリティー (一時) データ・セットとして指定できます。MDECK を指定するときは、SYSMDECK DD 割り振りで永続データ・セットを指定する必要があります。
- 代替 DDNAME リスト・パラメーターは、COBOL コンパイラーがアセンブリー言語プログラムから呼び出されるときに使用され、追加の作業データ・セット用の項目により拡張されます。

以下の JCL カタログ式プロシージャーはサポートされなくなり、Enterprise COBOL V5 では削除されています。これらのプロシージャーはすべて、言語環境プログラム・プリリンカーや DFSMS ローダーを使用しますが、言語環境プログラム・プリリンカーも DFSMS ローダーも Enterprise COBOL V5 での使用がサポートされなくなったことが、その理由です。

- IGYWCG
- IGYWCPG
- IGYWCPL
- IGYWCPLG
- IGYWPL

Enterprise COBOL V5 に同梱されているカタログ式プロシージャーは変更されています。

- IGYWC
- IGYWCL
- IGYWCLG

Enterprise COBOL バージョン 5 でのユーザー作成条件ハンドラーに関するコンパイルの制約事項

Enterprise COBOL V5 でのユーザー作成条件ハンドラーの制約事項、および 5.1 と 5.2 の相違点について説明します。

Enterprise COBOL V5.1 でのユーザー作成条件ハンドラー

Enterprise COBOL V5.1 では、Language Environment サービス CEEHDLR を使用してユーザー作成条件ハンドラーを登録するアプリケーション内の COBOL プログラムはすべて、コンパイラー・オプションの以下のいずれかの構成を使用してコンパイルされていなければなりません。

- OPTIMIZE(0)
- OPTIMIZE(1) および TEST
- OPTIMIZE(2) および TEST

ユーザー作成条件処理サービスの使用は、OPTIMIZE(1) または OPTIMIZE(2) および NOTEST によって行われる高度な最適化とは両立せず、予期しない結果を引き起こすおそれがあります。 OPTIMIZE(1) または OPTIMIZE(2) とともに TEST オプションを指定する必要があります。これにより、実行される最適化量が削減されます。

Enterprise COBOL V5.2 でのユーザー作成条件ハンドラー

Enterprise COBOL V5.2 では、新しい VOLATILE 節が形式 1 データ記述項目に追加されました。この節は、Language Environment (LE) サービス CEEHDLR によって LE 条件ハンドラーを使用するプログラムに高レベルの最適化を使用することによる問題に対処するために役立ちます。このようなプログラムに対して、OPTIMIZE(1) または OPTIMIZE(2) を TEST コンパイラー・オプションなしで使用する場合には注意が必要です。特に、条件ハンドラー・プログラムが、条件ハンドラー・プログラム自体に対してローカルに定義されていないデータ項目 (例えば、アプリケーションで EXTERNAL として定義されているデータ項目) にアクセスする場合は、そのようなデータ項目を、条件が発生する可能性があるすべてのプログラムで、VOLATILE 節を使用して定義する必要があります。そうしないと、ハンドラー・プログラムはデータ項目の最新値を使用しない可能性があります。この場合、パフォーマンスに配慮して、VOLATILE 節の使用が TEST オプションの使用に優先されます。

再開ポイントを設定するために LE サービス CEE3SRP とともに SERVICE LABEL コンパイラー指示ステートメントも使用する条件ハンドラーのシナリオでは、このようなプログラムの最適化は大幅に削減される場合があります。

注: VOLATILE は現在、Enterprise COBOL での予約語になっています。VOLATILE をユーザー定義語 (データ名や段落名など) として使用する既存のプログラムは、Enterprise COBOL V5.2 で S レベルの診断メッセージを受け取ります。

VOLATILE のこれらのインスタンスは、VOLATILE-X などの他の語に変更する必要があります。あるいは、CCCA ユーティリティーを使用してこれを自動的に行うことができます。

VOLATILE 節について詳しくは、「*Enterprise COBOL 言語解説書*」の『*VOLATILE 節*』を参照してください。

Enterprise COBOL バージョン 5 でのバインド (リンク・エディット) の変更点

Enterprise COBOL バージョン 5 プログラムのバインド (リンク・エディット) がいくつか変更されています。

- Enterprise COBOL V5 アプリケーションをバインド (リンク・エディット) するには、DFSMS プログラム管理バインダーを使用する必要があります。言語環境プログラム・プリリンカーはサポートされなくなりました。
- 実行可能ファイルはプログラム・オブジェクトであり、ロード・モジュールではありません。プログラム管理ローダー (IEWBLDGO) はサポートされなくなりました。
- 実行可能ファイルは PDS データ・セット内に置くことはできません (PDSE 内のみ可能)。
- DWARF デバッグ・データを使用する Debug Tool、CEEDUMP、Fault Analyzer、Application Performance Analyzer、またはサード・パーティー・ベンダー・ツールが使用されていなければ、NOLOAD セグメントは実行時にストレージを使用しません。
- プログラム・オブジェクトに以下のいずれかのプログラムが含まれている場合は、バインダー・オプション RMODE(24) を指定する必要があります。
  - RMODE(24) または NORENT コンパイラー・オプションでコンパイルされた Enterprise COBOL プログラム。
  - NORENT オプションでコンパイルされた VS COBOL II プログラム。
  - RMODE 24 の CSECT が含まれているアセンブラー・プログラム。

Enterprise COBOL バージョン 5 の実行時の変更

Enterprise COBOL V5 の実行時の動作がいくつか変更されています。

Enterprise COBOL バージョン 5 プログラムをサポートするLanguage Environment PTF が z/OS システムにインストールされていない場合は、そのシステムで Enterprise COBOL バージョン 5 プログラムを実行することはできません。

- ランタイム・オプションが変更されました。詳細については、 201 ページの『Language Environment オプションの変更』を参照してください。
- インターオペラビリティー。Enterprise COBOL バージョン 5 には、旧バージョンの COBOL とのインターオペラビリティーにいくつかの制約事項があります。詳細については、 24 ページの『古いレベルの IBM COBOL プログラムとのインターオペラビリティー』を参照してください。
- Enterprise COBOL バージョン 4 でサポートされていた、すべての AMODE シナリオおよび RMODE シナリオは、NORENT コンパイラー・オプションでコンパ

イルされたプログラムが RMODE 24 でなければならないことを除いて、バージョン 5 でもサポートされるようになりました。バインドの後、実行可能 COBOL プログラムに、以下の AMODE 属性および RMODE 属性の組み合わせを指定できます。

- AMODE 31 と RMODE ANY
- AMODE ANY と RMODE 24
- AMODE 24 と RMODE 24

解決される AMODE 設定および RMODE 設定は、使用されている COBOL 言語構造、指定されたコンパイラー・オプション、指定されたバインダー・オプション、および実行可能モジュールにバインドされている入力オブジェクト・モジュールの AMODE 属性および RMODE 属性によって異なります。

- Enterprise COBOL V5 を使用してコンパイルされたアプリケーションでは、ランタイム・オプション `CHECK(OFF)` または `NOSSRANGE` を使用しても、コンパイルされた範囲検査を実行時に無効にすることはできません。
- 再使用可能な COBOL 環境を管理するための `IGZERRE` および `ILBOSTP0` インターフェースは、Enterprise COBOL V5 でコンパイルされたプログラムを含むアプリケーションではサポートされません。
- 静的呼び出しを動的呼び出しに変換するための `IGZBRDGE` マクロは、Enterprise COBOL V5 でコンパイルされたプログラムではサポートされません。
- 新しいコンパイラー・オプション `VLR(COMPAT|STANDARD)` は、Enterprise COBOL が可変長レコード・ファイルでの READ ステートメントのレコード長との矛盾を処理する方法を制御します。詳細については、 202 ページの『可変長レコード - 不正な長さの READ』を参照してください。
- 再入可能 COBOL プログラム用の VSAM レコード域は、デフォルトで 16 MB より上に割り振られます。 VSAM ファイル・レコード内のデータを `CALL ... USING` パラメーターとして `AMODE 24` サブプログラムに渡すプログラムが、影響を受けることがあります。このようなプログラムは、`DATA(24)` コンパイラー・オプションを指定して再コンパイルするか、言語環境プログラムの `HEAP()` オプションを使用すると、レコードが `AMODE 24` プログラムによってアドレッシング可能になります。
- CICS システム定義 (CSD) ファイルを更新し、Enterprise COBOL V5 ランタイム・モジュールを組み込む必要があります。詳細については、 223 ページの『CSD セットアップにおける Enterprise COBOL V5 との違い』を参照してください。
- COBOL プログラムが IEEE (10 進または 2 進) 浮動小数点ゼロ除算演算を実行すると、割り算演算により IEEE ゼロ除算例外が引き起こされます。詳細については、 206 ページの『オブジェクト指向 COBOL の使用または C プログラムとの相互運用』を参照してください。

Language Environment オプションの変更

Enterprise COBOL バージョン 5 プログラムのランタイム・オプションがいくつか変更されています。

以下のオプションに関しては、Enterprise COBOL バージョン 5 でコンパイルされたプログラムでの動作が異なります。

表 33. Enterprise COBOL バージョン 5 におけるランタイム・オプションの変更

オプション	説明
HEAP	バッチ下では WORKING-STORAGE スペース (RENT でコンパイルされたプログラム用) が HEAP から取得されない場合があるため、HEAP (および STORAGE) オプションは効力がありません。 バッチ下での WORKING-STORAGE は、COBOL V5 プログラムが静的に C、C++、または PL/I の各プログラムにリンクされ、プログラム・オブジェクトのメイン・エントリー・ポイントが COBOL V5 ではない場合、HEAP から取得されません。
CHECK(OFF)	CHECK(OFF) によって SSRANGE 検査が無効にされることはなくなりました。
NOSSRANGE	NOSSRANGE によって SSRANGE 検査が無効にされることはなくなりました。 **注:** NOSSRANGE コンパイラー・オプションは引き続き完全にサポートされています。
STORAGE	少数の特殊なケースでは、HEAP の STORAGE 初期値が WORKING-STORAGE 初期値に影響を与えることはなくなりました。このケースについて詳しくは、上記 HEAP オプションに関する説明を参照してください。

可変長レコード - 不正な長さの READ

当初、Enterprise COBOL V5.1 では、旧 COBOL コンパイラーに比べて不正な長さの READ に対する動作が変更されていましたが、最新サービスが適用された Enterprise COBOL V5.1、および V5.2 では、この動作は新しいコンパイラー・オプション `VLR(COMPAT|STANDARD)` によって変更できるようになりました。このオプションは、サービスが適用されていない COBOL V5.1 のオリジナルの標準適応動作、または旧 COBOL コンパイラーと互換性のある動作のどちらを行うかを制御するために導入されました。これにより、ご使用のプログラムにレコード長の矛盾を引き起こす `READ` ステートメントがある場合に、旧バージョンから Enterprise COBOL V5 への移行が容易になります。

85 COBOL 標準では、`READ` ステートメントの処理の一部として次の規則が指定されています: 「読み取られたレコード内の文字位置の数が、ファイルのレコード記述項目で指定されている最小サイズよりも小さい場合、最後に読み取られた有効文字の右側にあるレコード域の部分は未定義になります。読み取られたレコード内の文字位置の数が、file-name-1 のレコード記述項目で指定されている最大サイズよりも大きい場合、最大サイズより右にあるレコード部分は切り捨てられます。いずれの場合でも、READ ステートメントは正常に実行され、レコード長の矛盾が生じたことを示す入出力状況値 04 が設定されます。」

このロジックは、VS COBOL II、COBOL/370、COBOL for MVS & VM (以下の APAR 修正がインストールされたコンパイラーを除く)、およびサービスが適用されていない Enterprise COBOL V5.1 に正しく実装されていました。 `READ` ステートメントでレコード長の矛盾が検出されると、状況値 `04` を受け取ります。ただし、プログラムが以下のいずれかのコンパイラーでコンパイルされている場合は、状況値 `00` を受け取ります。これは、`READ` ステートメントの非標準の結果です。

- VS COBOL II V1.3 (APAR PN34704 用 PTF をインストール済み)
- VS COBOL II V1.4 (APAR PN38730 用 PTF をインストール済み)
- COBOL/370 V1.1 または V1.2 (APAR PN36445 用 PTF をインストール済み)
- COBOL for OS/390 & VM バージョン 2
- Enterprise COBOL V3 または V4

矛盾する動作により、Enterprise COBOL V5 への移行が妨げられる場合があります。このため、最新サービスが適用された Enterprise COBOL V5.1、および V5.2 では、VLR(COMPAT) コンパイラー・オプションによって、互換性がある非標準の動作を使用することを選択できます。

- VLR(COMPAT) を指定した場合は、READ ステートメントでレコード長の矛盾または「不正な長さの READ」が検出されると、File Status 00 を受け取ります。

  プログラムが「不正な長さの READ」を実行し、可変長レコード・ファイルの読み取り後にコードが File Status=0 を検査する場合、コードは、Enterprise COBOL V4 以前のバージョンと同様に「ゼロ」パスを取ります。

  注: この設定により、不正な長さの READ 状態で引き起こされる入出力問題を隠すことができます。 VLR(COMPAT) オプションの使用には注意が必要です。また、READ ステートメントが正しいかどうかを確認してください。
- VLR(STANDARD) を指定した場合は、READ ステートメントでレコード長の矛盾または「不正な長さの READ」が検出されると、File Status 04 を受け取ります。この設定を使用すると、FS=04 を検査することができ、レコード内の未定義データへのアクセスを回避し、またレコードの切り捨てられた部分への参照試行に対して保護例外を受け取ることを回避するコードを追加できます。

  プログラムが「不正な長さの READ」を実行し、可変長レコード・ファイルの読み取り後にコードが File Status=0 を検査する場合、コードは「非ゼロ」パスを取ります。 FS=0 をテストするようにコードを変更できます。 FS=4 およびその他の値はすべて失敗 READ になります。 FS=4 に関しては、変数内の不正データまたは保護例外を回避するコードを追加できます。

VLR(STANDARD) を使用すると、「不正な長さの READ」が発生した可能性がある場合にファイル状況によってそのことが示されるため、より信頼性の高いコードが生成され、入出力の問題は減少します。新しいコンパイラー・メッセージ MSGIGYP3178 も、「不正な長さの READ」の可能性がプログラムにあるかどうかを通知することにより、入出力問題の回避に役立ちます。このメッセージは、ファイル定義 (FD) を修正することで解決可能な「不正な長さの READ」の可能性を示すことにより、VLR(COMPAT) から VLR(STANDARD) への移行の支援に使用できます。また、実行のためにはプログラムの修正を必須とするように、メッセージの重大度を上げることもできます。これを行うには、EXIT コンパイラー・オプションの MSGEXIT サブオプションを使用して、メッセージ MSGIGYP3178 の重大度を I (RC=0) から S (RC=12)、E (RC=8)、または W (RC=4) に変更します。このメッセージの表示が不要な場合は、メッセージを完全に抑止することができます。

古いレベルの IBM COBOL プログラムとのインターオペラビリティー

Enterprise COBOL V5 のプログラムが、以前のバージョンの COBOL でコンパイルされたプログラムを呼び出したり、以前のバージョンの COBOL でコンパイルされたプログラムから呼び出されたりする (すなわち、V5 のプログラムが、以前のバージョンの COBOL でコンパイルされたプログラムと相互運用される) 場合に、いくつかの制約事項があります。

Enterprise COBOL V5 プログラムは、単一のアプリケーションにおいて OS/VS COBOL プログラムや VS COBOL II NORES プログラムと相互運用できません。Enterprise COBOL V5 でコンパイルされたプログラムを含む COBOL 実行単位 (言語環境プログラム・エンクレーブ) に OS/VS COBOL プログラムや VS COBOL II NORES プログラムを含めることはできません。

注: Enterprise COBOL V4 以前のバージョンでコンパイルされた COBOL プログラムのみを含む実行単位は、OS/VS COBOL プログラムや VS COBOL II NORES プログラムと相互運用できます。

Enterprise COBOL V5 でコンパイルされたプログラムは、以下の条件および呼び出しタイプに基づいて、VS COBOL II 以降でコンパイルされたプログラムと相互運用できます。

- 静的呼び出し。 Enterprise COBOL V5 でコンパイルされたプログラムは、単一のプログラム・オブジェクトを形成するために、以下のオブジェクト・モジュールまたはプログラムとバインド (リンク・エディット) したりできます。プログラム・オブジェクト内のプログラムは、相互に静的呼び出しを指定することができます。
  - RES コンパイラー・オプションを指定して VS COBOL II でコンパイルされたプログラム
  - VS COBOL II より後の任意の IBM COBOL コンパイラー・バージョンでコンパイルされたプログラム
  - Enterprise COBOL V3 または V4 でコンパイルされたプログラム

  注: NORES コンパイラー・オプションを指定して VS COBOL II でコンパイルされたプログラムは、Enterprise COBOL V5 でコンパイルされたプログラムと相互運用できません。
- 動的呼び出し。 RES オプションを指定して VS COBOL II でコンパイルされたプログラム、または VS COBOL II 以降のバージョンの COBOL でコンパイルされたプログラムを含むプログラム・モジュールも、動的 CALL ステートメントを使用して Enterprise COBOL V5 プログラム・オブジェクトと相互運用できます。
- DLL 呼び出し。 DLL リンケージをサポートしていた前のバージョンの COBOL でコンパイルされたプログラム・モジュールは、DLL リンケージを使用して Enterprise COBOL V5 プログラム・オブジェクトと相互運用できます。

正しくないプログラムのエラー動作変更

正しくない COBOL プログラムの動作は、Enterprise COBOL V5 とそれより前のバージョンでは異なる場合があります。 Enterprise COBOL V5 への移行に際しては、Enterprise COBOL V4 への移行時よりも積極的なテストを行うことを検討する必要があります。

- サポートされていない (まだ診断されていない) COBOL 言語構文を使用するプログラム。
- 実行時にデータ記述項目に PICTURE 節に準拠しない値を含むデータ項目との間でデータを移動するプログラム。 以下に例を示します。
  - サイズ超過値の +123456789 を含む、ピクチャー S9(6) USAGE BINARY が指定されたフルワード・バイナリー項目 (TRUNC(BIN) オプションが指定されている場合は除く)
  - サイズ超過値の 123 (例えば、16 進数の 123C) を含む、ピクチャー S99 PACKED-DECIMAL が指定された 2 バイトのパック 10 進数項目。
  - 無効または非優先の符号を含み、データ記述項目の符号要件に準拠しないパック 10 進数またはゾーン 10 進数の項目。
- 未診断の添え字範囲エラーが発生していて (SSRANGE コンパイラー・オプションが指定されていない場合)、基本データ項目に対するストレージ割り振り以外のストレージを参照するプログラム。
- 生成された特定のコード・シーケンス、レジスター規則、内部 IBM 制御ブロックに対する低レベルの依存関係を持つアプリケーションの Enterprise COBOL V5 での動作は、以前のバージョンにおける動作と異なる場合があります。
- 正しくないプログラムのすべてが正しくないものとして診断されるとは限りません。例えば、次のプログラムを見てください。このプログラムでは、ODO オブジェクトの値を正常範囲外に設定しています。

```
77 VAR1 COMP-3 PIC 9(3).
01 X.
02 VAR2 PIC X OCCURS 0 to 1 depending on VAR1.

MOVE 128 to VAR1
MOVE ALL 'C' to X *> This is illegal!
```

  結果:

  - V2、V3、および V4 の場合: 'C' の 128 バイトが移動されました
  - V5R1 の場合: 'C' の 1 バイトとジャンク 127 バイトが移動されました
- パラメーター長が一致しないプログラム:

```
WORKING-STORAGE SECTION.
77 GRP1 PIC X(100).
01 FILE-STATUS-DATA-ITEMS.
02 OTHER-SENSITIVE-DATA-ITEMS.
. . .
CALL 'SUBP' USING GRP1.

PROGRAM-ID. SUBP.
LINKAGE SECTION.
01 GRP2 PIC X(500).
PROCEDURE DIVISION USING GRP2
MOVE 'STUFF' TO GRP2(300:320) *> This is illegal!
```

  結果:

- V2、V3、および V4 の場合: 正しくないプログラムは失敗しません
- V5R1 の場合: CALLER でのファイル状況が変更され、フローが変更されました

- 数値比較において正しくないゾーン・ビットを持つゾーン 10 進数データ (USAGE DISPLAY のある数値) を使用するプログラム。この例では、VAR2 のバイト 3 は、ゾーン・ビット x'4' を持つ x'40' ですが、これは無効です。すべてのゾーン・ビットは x'F' でなければなりません。

```
WORKING-STORAGE SECTION.
01 VAR1 PIC X(5) VALUE '00 0'.
02 VAR2 REDEFINES VAR1 PIC 9(5).
. . .
IF VAR2 = ZERO
DISPLAY "EQUAL TO ZERO"
ELSE
DISPLAY "NOT EQUAL TO ZERO"
END-IF.
```

結果:

- V4 で NUMPROC(MIG) を指定した場合および V5.1 で OPT(0) を指定した場合、プログラムは「EQUAL TO ZERO」を表示します
- V4 で NUMPROC(PFD) または NUMPROC(NOPFD) を指定した場合および V5 で OPT(1) または OPT(2) を指定した場合、プログラムは「NOT EQUAL TO ZERO」を表示します。

ご使用のデータのゾーン 10 進数データ項目に正しいゾーン・ビットがあるとは限らない場合は、ゾーン・ビットが常に無視されるように、ZONEDATA(MIG) コンパイラー・オプションを使用してコンパイルしてください。

また、例の書式設定が正しいこと (つまり、字下げされたレベル 2 データ項目、正しい列など) を確認し、16 進値の内容を示すためのコメントを追加してください。

```
01 VAR1 PIC X(5) VALUE '00 0'. <* Value x'F0F040F0'
```

オブジェクト指向 COBOL の使用または C プログラムとの相互運用

Java や C などの一部のプログラミング言語では、ゼロ除算演算が無限大になることが予期されています。PL/I や COBOL などの他の言語では、ゼロ除算演算が例外を起こすことが予期されています。 COBOL プログラムは、ゼロ除算演算によって例外が発生するようなモードで実行するように、プロセッサーを設定します。COBOL プログラムがオブジェクト指向であり、Java メソッドを呼び出す場合、または COBOL プログラムが C プログラムと相互運用される場合、Java または C プログラムがゼロ除算演算を実行すると、プログラムは強制終了される可能性があります。

プログラムの強制終了を回避するには、IGZXDIVZ サンプルの手順に従って、条件ハンドラーをコンパイルおよびリンクして SCEERUN データ・セットに入れ、影響を受けるアプリケーションで Language Environment ランタイム・オプション USRHDLR(IGZXDIVZ) を使用します。

IBM Enterprise COBOL バージョン 5 におけるデバッグ情報の変更

IBM Enterprise COBOL バージョン 5 を使用してコンパイルされたプログラムのデバッグ情報は、以前のバージョンのコンパイラーを使用してコンパイルされたプログラムの動作とは異なります。

IBM Enterprise COBOL バージョン 5 では、デバッグ情報に関するジレンマが解決されます。従来は、以下の 2 つの選択肢がありました。

- 大きなロード占有スペースという代償を払っても、デバッグ・データを常に実行可能ファイルと一緒に保持する
- アプリケーションと同期し続けて必要なとき検索するという難題もあるが、デバッグ・データを分離して保持する

現在は、この両方をしのぐ状況にあります。プログラム・オブジェクトに NOLOAD デバッグ・セグメントを含めた場合、ロードされるプログラムのサイズがデバッグ・データによって増えることはなく、デバッグ・データは常に実行可能ファイルと合致し、いつでも使用できるため、データ・セットのリスト内を検索する必要がありません。

デバッグ可能バージョンのアプリケーションを生成するために使用される TEST コンパイラー・オプション、および NOTEST オプションが変更されました。

- TEST オプションを指定すると、DWARF デバッグ情報がアプリケーション・モジュールに含められます。
- SOURCE サブオプションが指定されている場合、DWARF デバッグ情報に拡張ソース・コードが含まれるため、IBM Debug Tool でコンパイラー・リストが不要になります。TEST(NOSOURCE) コンパイラー・オプションが指定されていると、生成された DWARF デバッグ情報に拡張ソース・コードは含まれません。
- NOTEST(DWARF) コンパイラー・オプションを使用すれば、基本 DWARF デバッグ情報をプログラム・オブジェクトに組み込むことができます。このようなプログラムは Debug Tool ではデバッグできませんが、それでも NOTEST 最適化を行い、CEEDUMP 出力や IBM Fault Analyzer などのアプリケーション障害分析ツールを有効化することはできます。
- プログラム・オブジェクトにデバッグ情報を含めないようにするには、NOTEST(NODWARF) オプションを使用します。

COBOL プログラムのデバッグ時に、向上した点および動作の変更が数多く Enterprise COBOL V5 に導入されていることが分かります。IBM Debug Tool でのデバッグの変更点について詳しくは、217 ページの『IBM Enterprise COBOL バージョン 5 における Debug Tool の変更』を参照してください。

第 16 章 Enterprise COBOL バージョン 5 プログラムを既存 COBOL アプリケーションに追加する

Enterprise COBOL V5 プログラムを既存のアプリケーションに追加する際は、既存のプログラムを Enterprise COBOL V5 で再コンパイルするか、新たに書かれた Enterprise COBOL V5 プログラムをインクルードします。

注: この移行ガイドは、Language Environment へのランタイムの移行を完了してある場合にのみ使用してください。これは、以下の条件が満たされていることを意味します。

- 言語環境プログラムのデータ・セット SCEERUN が、LNKLST または LPALST にインストールされている。
- LNKLST または LPALST 内に COBLIB、VSCLLIB、および COB2LIB のインスタンスがない。
- JCL STEPLIB/JOBLIB ステートメントや CICS 始動 JCL に COBLIB、VSCLLIB、COB2LIB のインスタンスがない。
- NORES でコンパイルされたプログラム用の、静的にバインドされたランタイム・ライブラリー・ルーチンがすべて、言語環境プログラムにあるルーチンで置き換えられている。
- RES でコンパイルされた VS COBOL II プログラム用の IGZEBST ブートストラップ・モジュールが、APAR PN74000 が適用された VS COBOL II ランタイム・バージョンの IGZEBST か、または言語環境プログラムにある IGZEBST で置き換えられた IGZEBST のいずれかにリンクしていた。CICS プログラムでは、COBOL V5 プログラムをアプリケーションに導入する場合、APAR PI33330 の PTF がインストールされた LE から IGZEBST を使用して再リンクすることが必要になります。

これらの手順が完了していない場合は、ここでの手順に従う前に、最初に「*Enterprise COBOL for z/OS コンパイラーおよびランタイム移行ガイド バージョン 4 リリース 2*」に記載されたランタイム移行作業をすべて完了しておいてください。

Enterprise COBOL V5 プログラムを既存のアプリケーションに追加する際には、以下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
- Language Environment の条件処理を使用する

C プログラム、C++ プログラム、または Enterprise PL/I プログラムにリンクした COBOL プログラムを含むプログラム・オブジェクトを使用している場合に、COBOL プログラムを Enterprise COBOL V5 に変更すると、プログラム・オブジェクトの動作が多少異なります。これは、このようなプログラム・オブジェクトを複数回フェッチした (つまり、C フェッチまたは PL/I フェッチのいずれかを使用した) 場合に発生します。後続のフェッチでは、他の LE 言語で使用された外部変数および静的変数が、その初期値を取得する代わりに、COBOL の規則に従って、最後に使用された状態を維持する可能性があります。以前のバージョンの COBOL と

リンクしている場合、C プログラム、C++ プログラム、および PL/I プログラムは C/C++ または PL/I の動作を保持します。

Enterprise COBOL バージョン 5 プログラムに関する AMODE 制約事項:

Enterprise COBOL V5.2 プログラムの AMODE 24 実行は、旧 Enterprise COBOL コンパイラーの場合と同様のすべてのケースでサポートされます。

注: COBOL プログラムを AMODE 24 で実行するには、すべての COBOL V5 プログラムを V5.1.1 以降のバージョンの Enterprise COBOL で、あるいは V4.2 以前のバージョンの Enterprise COBOL でコンパイルする必要があります。プログラム・オブジェクトのいずれかの構成要素が Enterprise COBOL V5.1.0 でコンパイルされている場合、そのプログラム・オブジェクトは AMODE 31 で動作しなければなりません。 AMODE 24 で実行される COBOL プログラムは、バインダー・オプション RMODE(24) でリンクされていなければなりません。

実行時の相違点:

Enterprise COBOL V5 プログラムは、以下のプログラムと混用できません。

- OS/VS COBOL プログラム。Enterprise COBOLにマイグレーションする必要があります。OS/VS COBOL プログラムを検出するには、以下のようにします。
  - Debug Tool の LMA ツールを使用し、ロード・ライブラリーを走査して OS/VS COBOL プログラムを探します。
  - Edge Portfolio Analyzer を使用し、ロード・ライブラリーを走査して OS/VS COBOL プログラムを探します。
  - ご使用の言語環境プログラムに APAR PM86742 用の修正をインストールして、実行時に検出された OS/VS COBOL プログラムに関する警告メッセージを探します。
- VS COBOL II NORES プログラム。Enterprise COBOLにマイグレーションする必要があります。

Enterprise COBOL バージョン 5 プログラムに関する RMODE 制約事項:

- 再入可能プログラムは RMODE 24 でも RMODE ANY でもかまいません。
- 再入不可プログラムは RMODE 24 でなければなりません。

AMODE および RMODE の考慮事項

Enterprise COBOL V5.1.0 プログラムでは、AMODE 24 プログラムと AMODE 31 プログラムとの間の静的呼び出しはサポートされていません。AMODE 24 プログラムと Enterprise COBOL V5.1.1 および V5.2 プログラムとの間での静的呼び出しは、Enterprise COBOL V5.1.1 および V5.2 プログラムで AMODE 24 がサポートされている場合はサポートされます。

注: V5.1.1 という Enterprise COBOL の正式なリリースはありませんでしたが、モディフィケーション・レベル V5.1.0 は、そのサービス・レベルのコンパイラーに AMODE 24 機能が追加されたことを示すために、サービスによって更新されました。

また、NORENT プログラムは境界よりも上に常駐できなくなりました。次の図に、動的または静的にすることができる呼び出しのタイプと、動的にしかできない呼び出しのタイプを示します。また、16 MB 境界を基準にしたデータおよびプログラムの位置に関する構成も示します。

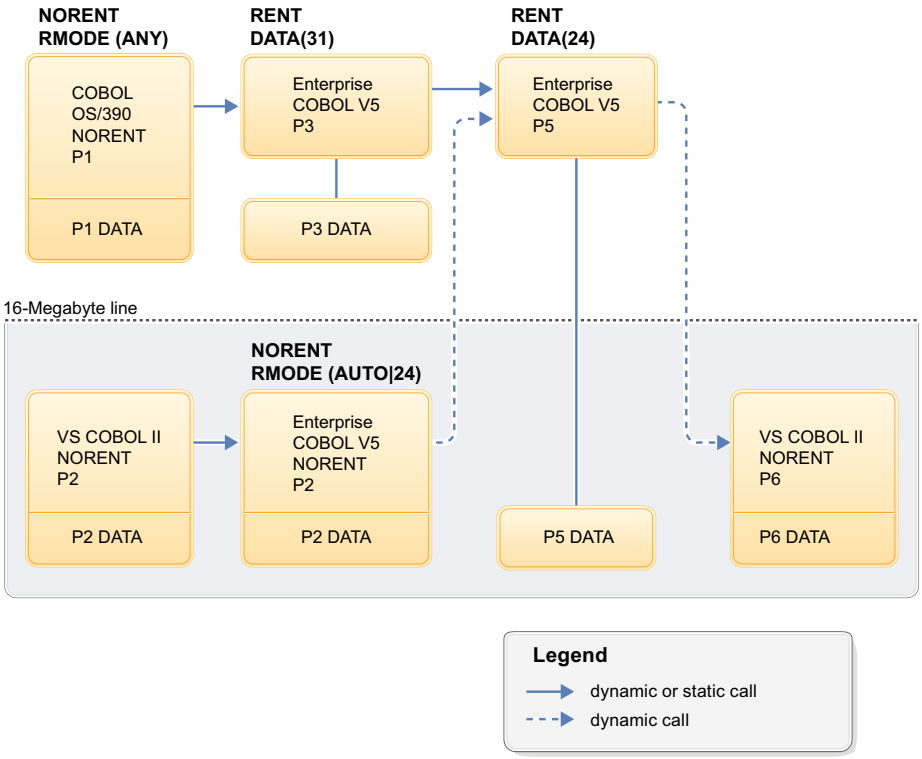

図 5. AMODE および RMODE の各種 COBOL プログラム間で有効な動的呼び出しと静的呼び出し

注: 他の AMODE 24 プログラムの場合は、Enterprise COBOL V5 プログラムと、OS/VS COBOL プログラムまたは VS COBOL II NORES プログラムのいずれかとの間で呼び出しは許可されません。

第 5 部 Enterprise COBOL の移行およびその他の IBM 製品

Enterprise COBOL for z/OS V5 では、CICS、DB2、IMS、およびその他のデータ・システムおよびトランザクション・システムにアクセスすることができます。また、Debug Tool と共に使用することができます。

第 17 章 Debug Tool

Debug Tool は、Language Environment 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。

Debug Tool は、VS COBOL II リリース 3.0、および後続のすべての COBOL コンパイラーに対してサポートを提供します。

Debug Tool の開始

Debug Tool 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、Language Environment の TEST ランタイム・オプションが Debug Tool の呼び出しを制御します。

Language Environment の呼び出し可能サービス CEETEST を使用することによって、Debug Tool 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

Language Environment 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 Language Environment と共に実行されているときに、Debug Tool 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、Debug Tool 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 Debug Tool を呼び出すことができます。

Language Environment 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、「*Language Environment プログラミング・リファレンス*」を参照してください。

CEETEST

Language Environment には、Debug Tool が制御を獲得することを可能にし、Debug Tool 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、Debug Tool が初期設定され、呼び出されます (Debug Tool 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 Debug Tool 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 Debug Tool 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、Debug Tool 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。

Language Environment の呼び出し可能サービス CEETEST の詳細および例については、「*Language Environment プログラミング・リファレンス*」を参照してください。

IBM Enterprise COBOL バージョン 5 におけるデバッグ情報の変更

IBM Enterprise COBOL バージョン 5 を使用してコンパイルされたプログラムのデバッグ情報は、以前のバージョンのコンパイラーを使用してコンパイルされたプログラムの動作とは異なります。

IBM Enterprise COBOL バージョン 5 では、デバッグ情報に関するジレンマが解決されます。従来は、以下の 2 つの選択肢がありました。

- 大きなロード占有スペースという代償を払っても、デバッグ・データを常に実行可能ファイルと一緒に保持する
- アプリケーションと同期し続けて必要なとき検索するという難題もあるが、デバッグ・データを分離して保持する

現在は、この両方をしのぐ状況にあります。プログラム・オブジェクトに NOLOAD デバッグ・セグメントを含めた場合、ロードされるプログラムのサイズがデバッグ・データによって増えることはなく、デバッグ・データは常に実行可能ファイルと合致し、いつでも使用できるため、データ・セットのリスト内を検索する必要がありません。

デバッグ可能バージョンのアプリケーションを生成するために使用される TEST コンパイラー・オプション、および NOTEST オプションが変更されました。

- TEST オプションを指定すると、DWARF デバッグ情報がアプリケーション・モジュールに含められます。
- SOURCE サブオプションが指定されている場合、DWARF デバッグ情報に拡張ソース・コードが含まれるため、IBM Debug Tool でコンパイラー・リストが不要になります。TEST(NOSOURCE) コンパイラー・オプションが指定されていると、生成された DWARF デバッグ情報に拡張ソース・コードは含まれません。
- NOTEST(DWARF) コンパイラー・オプションを使用すれば、基本 DWARF デバッグ情報をプログラム・オブジェクトに組み込むことができます。このようなプログラムは Debug Tool ではデバッグできませんが、それでも NOTEST 最適化を行い、CEEDUMP 出力や IBM Fault Analyzer などのアプリケーション障害分析ツールを有効化することはできます。
- プログラム・オブジェクトにデバッグ情報を含めないようにするには、NOTEST(NODWARF) オプションを使用します。

COBOL プログラムのデバッグ時に、向上した点および動作の変更が数多く Enterprise COBOL V5 に導入されていることが分かります。IBM Debug Tool でのデバッグの変更点について詳しくは、217 ページの『IBM Enterprise COBOL バージョン 5 における Debug Tool の変更』を参照してください。

IBM Enterprise COBOL バージョン 5 における Debug Tool の変更

Debug Tool を使用してデバッグを行う場合、IBM Enterprise COBOL バージョン 5 を使用してコンパイルされたプログラムには、以前のバージョンの COBOL を使用してコンパイルされたプログラムに比べて、多くのデバッグの利点があります。

COBOL アプリケーションとの Debug Tool インターフェースについて詳しくは、http://www-01.ibm.com/software/awdtools/debugtool/library/ にある資料を参照してください。

これらの相違の多くは、すべてのデバッグ・モード (フルスクリーン、バッチ、およびリモート) に当てはまります。 Debug Tool コマンドの完全な詳細は、「*Debug Tool References and Messages*」に記載されています。

DESCRIBE ATTRIBUTES コマンド

Debug Tool に示される PIC ストリングは、ソースで指定されているとおりに表示されます。このストリングは、Enterprise COBOL バージョン 5 よりも前のバージョンで行われていたように正規化されることはありません。

レベル・メンバーはソース・コードに書かれているとおりに表示されます。レベル・メンバーは、Enterprise COBOL バージョン 5 よりも前のバージョンで行われていたように正規化されることはありません。

データ記述がより明確になりました。例えば、現在の表示は次のようになります。

```
S9(5) SIGN LEAD SEP DISP
```

以前の表示は次のようになっていました。

```
S9(5) DSLS
```

DESCRIBE ATTRIBUTES は、Enterprise COBOL バージョン 5 では、シンボリック文字の長さとタイプを表示します。旧バージョンのコンパイラーでは、ゼロのみが表示されていました。

条件名 (レベル 88) で、アドレス 000000000 が表示されなくなりました。

配列および配列エレメントの出力がより簡潔で明確になりました。以下に例を示します。

- タイプに対しては IX ではなく INDEX が表示されます
- レベル 00 は表示されません
- 配列エレメントごとにタイプが繰り返し表示されることがなくなりました。エレメント・タイプは 1 回のみ表示されます。

レジスターにはアドレスがないため、Debug Tool ではレジスターの DESCRIBE ATTRIBUTES のアドレス (%GPR0 など) が表示されなくなりました。

LIST コマンドおよび AUTOMON 出力

表の LIST または AUTOMON では、新しい Debug Tool V12.1 オプション SET LIST BY SUBSCRIPT 形式が必ず表示されます。

長さがゼロの ODO 表を含むレコードまたはグループをリストする場合に、レコードまたはグループ内でその表の後にあるすべてのデータ項目が表示されます。以前は、これらのデータ項目は表示されませんでした。

AUTOMONITOR がアクティブな場合に、プログラム入り口に対してメッセージは表示されません。

英数字カテゴリーの Debug Tool 変数は、単一引用符で囲まれて表示されます。例えば、LIST %SYSTEM を実行した場合、%SYSTEM = 'MVS' と表示されるようになりました。

LIST %HEX の出力が改善されました。 LIST %HEX(var) の出力に、%HEX が表示されなくなりました。現在の出力は、%HEX (SBIN0_5) = X'00003039' ではなく、SBIN0_5 = X'00003039' です。 X' は 16 進表記を表します。

LIST varname の出力にブロック修飾が含まれなくなりました。例えば、結果は block_name::>varname = 5 ではなく varname = 5 になります。

LIST NAMES コマンドの出力に、01 および 77 レベルのデータ項目が表示されるようになりました。以前のバージョンでは、レコードまたはグループ階層内の従属データ項目を含む、すべてのデータ項目が表示されていました。展開された構造全体を表示するには、DESCRIBE ATTRIBUTES varname を使用します。

LIST NAMES LABEL は、ネストされたプログラム内のアクティブ・ブロックに含まれるラベルのみを表示するようになりました。以前は、操作対象のブロックに関係なく、プログラムのすべてのラベルが表示されていました。

ネストされたプログラムに対する LIST TITLED の出力が変更されました。現在は、アクティブ・ブロック内の変数のみが表示されます。

COBOL に関して、LIST コマンド後の配列の定様式表示が変更されました。配列のエレメントがグループの場合は、あるエレメントについて対象グループの全メンバーがまとめてリストされ、その後に次のエレメントについて対象グループのメンバーがリストされ、以降、その処理が繰り返されます。以前は、あるメンバーがすべての配列エレメントにわたってリストされ、その後に次のグループ・メンバーがすべての配列エレメントにわたってリストされていました。キーワード SUB は表示されなくなりました。

AUTOMONITOR の出力は、ADDRESS OF *var* および LENGTH OF *var* を単一の参照として表示します。

AT APPEARANCE および LIST NAMES CUS が変更されました。 Debug Tool は cus を認識します。例えば、アプリケーション内のメイン・プログラム・オブジェクトが MYMAIN であり、メインプログラムが MYMAIN であり、プログラム・オブジェクト内の 2 番目のプログラムが MYSUB1 である場合は、MYMAIN::>MYMAIN 1 で停止できます。その際、以下の新しい動作が示されます。

- LIST NAMES CUS を発行すると、プログラム・オブジェクト MYMAIN のほかに、メインプログラム MYMAIN とサブプログラム MYSUB1 の両方が表示されます。
- MYSUB1 に対して AT APPEARANCE 停止点を発行すると、その停止点が受け入れられます。

Enterprise COBOL V5 は、ネストされたプログラムごとに保存域を割り当てます。この保存域は、LIST CALL などのコマンドで参照できます。

MOVE コマンド、COMPUTE コマンド、IF コマンド

受信側のコンパイラーと送信側のコンパイラーで同じデータ型を使用できるように、Debug Tool の MOVE および COMPUTE コマンドが拡張されました。この機能拡張により、これらのコマンドの使用に関する従来の制限がなくなりました。

IF コマンドが拡張されました。 Enterprise COBOL V5 で Debug Tool を使用する場合に、関係条件に使用できる比較が拡張されました。関係条件 (データ項目、リテラル、および形象定数を含む) に使用できる比較は、「Enterprise COBOL 言語解説書」に従ってインプリメントされています。

また、添え字が変更されたことにより、これらのコマンドの使い勝手が向上しました。

- Enterprise COBOL バージョン 5 では、添え字名の相対的添え字付けが行われます。
- COBOL 言語規則に準拠するために、添え字データ項目を使用して配列に添え字を付けることはできなくなりました。
- COBOL 言語規則に準拠するために、添え字名に対して IN 修飾子も OF 修飾子も使用できなくなりました。

STEP コマンド

STEP を発行すれば、EVALUATE の WHEN 句に対して停止点を設定できます。

PERFORM とともに STEP OVER を使用することがサポートされるようになりました。

COBOL データ型のサポート

Debug Tool は、すべてのバイナリー・データ型で、正しい最大値をサポートするようになりました。例えば、符号なしの 8 バイト COMP-5 データ項目に 20 桁の最大値 18,446,744,073,709,551,615 を含めることができます。

INDEX (IX) および配列

Enterprise COBOL V5 では、タイプ INDEX のデータ項目を添え字として使用することはできません。例えば、データ項目を 77 IXDI1 USAGE IS INDEX として定義した場合は、LIST ARR(IXDI1) を実行することはできません。

添え字名にレベル番号はありませんが、添え字名は、トップレベル (01 または 77) のデータ項目と同じようにしてデバッグ情報に含まれています。以前のバージョンの Enterprise COBOL では、添え字名は、添え字名が含まれる配列の子と同じように、デバッグ情報にテーブル・エレメントとともに表示されていました。 Enterprise COBOL V5 では、表情報がリストされている場合は、添え字名は表示されません。添え字名は、名前で明示的にリストされる場合にのみ表示されます。以下のコマンドを使用すると、この変更が出力に反映されます。

- LIST NAMES

- LIST TITLED
- DESCRIBE ATTRIBUTES (引数なし)

Enterprise COBOL V5 では、INDEX 名が属している配列の名前を使用して、その INDEX 名を修飾することはできません。また、添え字名を含むグループやレコードの名前を、従属データ項目であるかのように修飾することはできなくなりました。例えば、REC1 の IX3 などがあります。以前のバージョンの Enterprise COBOL では、これが可能でした。

Enterprise COBOL V5 は、配列の添え字の一部として、増分 (+) 演算子や減分 (-) 演算子をサポートします。 Enterprise COBOL V4 では、これはサポートされていませんでした。

Enterprise COBOL V5 プログラムでは、配列の添え字をユーザーが指定しないと、Debug Tool のデフォルトである 1 が使用されます。以前のバージョンでは、Debug Tool は配列のメンバーをすべてリストしていました。次に示すように配列が宣言されているときに、LIST X が発行された場合、Debug Tool は、配列 ARR(1) 内の最初のエレメントのみを LIST X(1) として表示します。 LIST ARR(n) は、指定された添え字の X と Y を表示し、LIST ARR はすべてのメンバーの X と Y を表示します。

```
05 ARR OCCURS 10
10 X PIC 99
10 Y PIC 99
```

以前のバージョンの Enterprise COBOL では、配列の単一エレメントをリストするときに、出力がサイズ 1 の配列であるかのような形式になっていました。Enterprise COBOL V5 では、出力は、サイズ 1 の配列であるかのようにはなっておらず、指定のタイプの変数と同じになります。

その他の変更

AT CALL 入り口名は Enterprise COBOL V5 ではサポートされていません。

DESCRIBE CUS コマンドに対して複数の変更が実施されました。その変更は、以下のようなものです。

- 新しいコンパイラー名: IBM COBOL 5.2.0
- タイム・スタンプが表示されます: * Compiler: IBM COBOL 5.2.0 2014/11/27 13:08
- コンパイラー・オプションに多くの変更が行われました。
- リンケージのタイプが表示されます: * Its linkage is Language Environment FastLink。これがコンパイラーのデフォルト・リンケージです。

NUM オプションを使用した行番号付け、およびプログラムのシーケンスが、Enterprise COBOL V5 で変更されました。以前のバージョンでは、NUM および NOLIB を指定したバッチ・コンパイル (単一ソース内のプログラムのシーケンス) の場合、2 番目のプログラムでは、行番号が改めて最初から付け直されていました。Enterprise COBOL V5 では、NOLIB オプションが廃止されました。コンパイラーは、LIB が常に有効化されているかのように動作するため、シーケンス内の 2 番目のプログラムの行番号は、最初のプログラムの行番号から継続されます。

Enterprise COBOL V5 では、国別データ項目の表示に N が含まれます。例: 01 nat pic N(5) value "abcde" national のリスト表示は NAT= N'abcde' V4: NAT = 'abcde' です。

Enterprise COBOL V5 では、算術式に表示される桁数が変わりました。算術演算結果の桁数は「Enterprise COBOL プログラミング・ガイド」に定義されています。

Enterprise COBOL V5 では、符号の処理が変更されました。一方のオペランドに符号が付いている場合、算術式の結果に符合が付きます。以前のバージョンでは、結果の符合は答えによって異なっていました。ただし、減算および単項減算の結果の場合は別です。この場合は、結果の正確性を保証するために、常に符号が付きます。

デバッグ対象のプログラムが QUALIFY(EXTEND) オプションでコンパイルされていた場合、Debug Tool は、データ項目を参照するコマンド (例えば LIST、MOVE、COMPUTE、およびその他のコマンド) に新しい名前解決規則を適用します。これにより、データ項目参照の解決に関して、Debug Tool はコンパイラーと整合します。

Debug Tool は、VOLATILE キーワードで定義されたデータ項目を処理するように更新されました。これにより、このようなデータ項目を、不揮発性データ項目と同じ全コマンドで使用できるようになりました。

IBM Enterprise COBOL V5 でのフルスクリーン・モードの変更

フルスクリーン・モードのコマンドおよび機能には、以下の変更が適用されます。

以下のコマンドは、Enterprise COBOL V5 プログラムと、それより前のコンパイラーのプログラムとでは異なります。

- PANEL LISTINGS および PANEL SOURCES。この両方のコマンドは、プログラム名を示します。
- SET DEFAULT LISTINGS。 COBOL V5 プログラムでは、オブジェクトにソース・リスト情報が組み込まれます。
- SET DYNDEBUG OFF。 COBOL V5 コンパイラーは、コンパイル・フックをサポートしません。 COBOL V5 プログラム内で停止点を使用したり設定したりする場合は、SET DYNDEBUG ON が必要になります。
- SET LIST BY SUBSCRIPT。 COBOL V5 プログラムの場合、Debug Tool は、LIST BY SUBSCRIPT ON が常に有効であるかのように配列を表示します。 Enterprise COBOL V4 プログラムの場合、配列のデフォルト表示は SET LIST BY SUBSCRIPT OFF でした。
- SET PROGRAMMING LANGUAGE。 COBOL V5 のプログラミング言語は COBOL です。
- SET SOURCE。ソース・リスト情報はオブジェクトに組み込まれています。 COBOL V5 プログラムでこのコマンドを発行すると、エラー・メッセージが表示されます。

IBM Enterprise COBOL V5 でのリモート・モードに関する Debug Tool の変更

このセクションでは、リモート・デバッガー・インターフェースに適用される変更をリストします。

変更は以下のとおりです。

- Enterprise COBOL V5 では、モニター対象の式のツリーに含まれるノードにレベル番号 (例: `05 VAR1`) が示されます。 Enterprise COBOL V4 では、`VAR1` が示されていました。
- Enterprise COBOL V5 では、タイプ情報の一部として `PIC` が示されます (例: `05 SBIN1 PIC 99 COMP`)。
- Enterprise COBOL V4 の場合、配列型は ARRAY として示されていました。 Enterprise COBOL V5 では、これを示すために、該当する COBOL 用語 (`9 COMP OCCURS 2` など) が使用されます。これは、バッチ/フルスクリーン・モードの動作と一致します。
- Enterprise COBOL V5 では、レコード・タイプはその言語で認識されているとおりに示されます。例えば、`ALPHANUMERIC GROUP` や `NATIONAL GROUP` などです。 Enterprise COBOL V4 の場合、レコード・タイプは CHARACTER、STRUCT、または ARRAY として示されていました。
- Enterprise COBOL V5 でコンパイルされたプログラムでは、配列添え字をセミコロンで区切ることができます。これは、Enterprise COBOL V4 でコンパイルされたプログラムでは許可されていませんでした。これは、フルスクリーン・モードでは許可されていません。
- Enterprise COBOL V5 でコンパイルされたプログラムでは、ネストされたプログラムがデバッグ・ビューに表示されるようになりました。
- COBOL 言語には、例外条件を処理するための DECLARATIVES セクションが用意されています。 Enterprise COBOL V5 では、DECLARATIVES セクションが Debug Tool セッションにおける制御を持っている場合、そのための個別のフレームがデバッグ・ビューに表示されます。

第 18 章 COBOL ソースに関する CICS の移行の考慮事項

CICS のもとでプログラムを実行するには、必要なコンパイラー・オプションを十分理解している必要があります。また、CICS 環境で Enterprise COBOL プログラムを実行するための移行手順に従うか、組み込み CICS 変換プログラムを使用するようにプログラムをアップグレードする必要があります。

以下のトピックを考えてください。

- CSD セットアップにおける Enterprise COBOL V5 との違い
- DFHRPL セットアップにおける Enterprise COBOL V5 との違い
- CICS で実行されるプログラム用のコンパイラー・オプション
- 単独の CICS 変換プログラムから組み込みの CICS 変換プログラムへのマイグレーション
- CICS 下での COBOL V5 プログラムから VS COBOL II プログラムへの静的呼び出し

OS/VS COBOL の制約事項

OS/VS COBOL プログラムは、CICS のもとでは実行できなくなりました。CICS のもとで実行するすべての OS/VS COBOL プログラムは、Enterprise COBOL にアップグレードする必要があります。

CSD セットアップにおける Enterprise COBOL V5 との違い

CICS システム定義 (CSD) ファイルを更新し、Enterprise COBOL V5 ランタイム・モジュールを組み込む必要があります。

CICS をセットアップする一般的な手順には、CICS 環境で使用されるプログラム・モジュールを定義するための、CICS システム定義ファイルの更新が含まれます。新しいライブラリー・モジュールを Enterprise COBOL V5 用に追加する必要があります。Language Environment データ・セット `SCEERUN` に入るモジュールは以下のとおりです。

- `CEEEV004`
- `IGZXLPKA`
- `IGZXD24`
- `IGZXDMR`
- `IGZLLIBV`
- `IGZXLPKC`
- `IGZXLPIO`
- `IGZXAPI`
- `IEWBNDD`
- `IEWBIND`
- `CDAEEDE`

| • IGZXLPKB
| • IGZXLPKD
| • IGZXLPKE
| • IGZXLPKF
| • IGZXLPKG
| • IGZXPK2

Language Environment SCEESAMP データ・セットのメンバー CEECCSD に、この定義ファイルの例が用意されています。以下の行を既存の CSD ファイルに追加することもできます。

```
DEFINE PROGRAM(CEEEV004) GROUP(CEE)
DEFINE PROGRAM(IGZXLPKA) GROUP(CEE)
DEFINE PROGRAM(IGZXD24)  GROUP(CEE)
DEFINE PROGRAM(IGZXDMR)  GROUP(CEE)
DEFINE PROGRAM(IGZLLIBV) GROUP(CEE)
DEFINE PROGRAM(IGZXLPKC) GROUP(CEE)
DEFINE PROGRAM(IGZXLPIO) GROUP(CEE)
DEFINE PROGRAM(IGZXAPI)  GROUP(CEE)
DEFINE PROGRAM(IEWBNDD)  GROUP(CEE)
DEFINE PROGRAM(IEWBIND)  GROUP(CEE)
DEFINE PROGRAM(CDAEEDE)  GROUP(CEE)
DEFINE PROGRAM(IGZXLPKB) GROUP(CEE)
DEFINE PROGRAM(IGZXLPKD) GROUP(CEE)
DEFINE PROGRAM(IGZXLPKE) GROUP(CEE)
DEFINE PROGRAM(IGZXLPKF) GROUP(CEE)
DEFINE PROGRAM(IGZXLPKG) GROUP(CEE)
DEFINE PROGRAM(IGZXPK2) GROUP(CEE)
```

DFHRPL セットアップにおける Enterprise COBOL V5 との違い

プログラムを CICS 環境で実行するには、DFHRPL セットアップにおける違いを考慮してください。

CICS を始動する JCL を更新する必要があります。Debug Tool の hlq.SEQAMOD データ・セット、および言語環境プログラム・ランタイム・ライブラリー (SCEECICS、SCEERUN、またアプリケーションが必要としている場合は SCEERUN2) を DFHRPL 連結に組み込んでください。

TEST コンパイラー・オプションを使用して CICS 上でコンパイルされた Enterprise COBOL V5.1 以降のプログラムを実行している場合は、システム・ライブラリー MIGLIB および SIEAMIGE も DFHRPL DD 連結に追加する必要があります。DFHRPL 連結は、CICS 領域セットアップ JCL にあります。

CICS で実行されるプログラム用のコンパイラー・オプション

表 34 には、CICS で実行される 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 34. CICS で実行されるプログラム用のコンパイラー・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 変換プログラム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 変換プログラム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、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、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。
	
	CICS 変換プログラム・オプション COBOL3 が推奨されていますが、 COBOL2 もサポートされます。
	
	一時変数を使用する必要がある以前のプログラムを再変換する場合、COBOL2 オプションを選択します。特に注意すべき点は、一時変数を使用すると、プログラム内の引数値が正しく定義されていない場合に通常発生するようなエラーが回避されることがある点です。この COBOL2 オプションでは、一時変数の宣言が提供されます。この機能のために、COBOL2 で変換されたプログラム内では、実行時には気付かなくても、正しくない引数値の定義がある場合があります。このようなプログラムでは、COBOL3 オプションで変換するときに初めて、エラーが見つかることがあります。
	
	例えば、以下のようにコーディングしたとします。
	
	`EXEC CICS LINK PROGRAM('XXXXXXX')`
	`COMMAREA(WS-COMMAREA)`
	`LENGTH('1000')`
	`END-EXEC.`
	
	長さはバイナリー・ハーフワードにする規則になっていますが、引用符で囲まれているため、文字ストリングとして扱われます。 COBOL3 では、この文字ストリングは CALL の時点で直接 CICS に渡されるので、エラーが発生します。COBOL2 オプションでは、この長さは中間変数に移動され、さらに COBOL はそれを移動の一部として、文字ストリングからバイナリー・ハーフワードに変換します。COBOL2 オプションを使用すれば、プログラム内のエラーを修正せずに、その発生を従来どおり回避することができ、CICS の新しいリリースへのマイグレーションが容易になります。
	
	NOCICS オプションが有効な場合は、検出されたすべての CICS ステートメントは S レベル診断のフラグが立てられ、廃棄されます。

DBCS	DBCS オプションは Enterprise COBOL のデフォルトです。 COBOL2 CICS 変換プログラムのオプションを使用している場合は、CICS プログラムに問題が発生することがあります。COBOL3 変換プログラムのオプションを使用して修正してください。
NODYNAM	CICS 変換プログラム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。 **注:** CALL ステートメントの CALL identifier 形式を使用することによって、動的呼び出しは CICS 環境でサポートされます。

表 34. CICS で実行されるプログラム用のコンパイラー・オプション (続き)

コンパイラー・オプション	説明
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 および ELPA は読取専用ストレージなので、モジュールに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。例えば、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。また、TRUNC(OPT) を使用して特定の項目を COMP-5 として再定義すると、プログラム全体のランタイム・パフォーマンスを向上させることができます。

単独の CICS 変換プログラムから組み込みの変換プログラムへのマイグレーション

単独の CICS 変換プログラムを使用することから、組み込み変換プログラムを使用することにプログラムを移行するには、いくつかの変更を行う必要があります。

OS/VS COBOL CICS プログラムを Enterprise COBOL に移行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、47 ページの『第 5 章 OS/VS COBOL ソース・プログラムのアップグレード』の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、267 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。

組み込みの CICS 変換プログラムを使用するために COBOL アプリケーションを移行する場合は、以下を行います。

- 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 変換プログラム・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。例えば、単独の CICS 変換プログラムによって変換するプログラムに、以下のような CBL ステートメントが含まれているとします。

  ```
  CBL TEST(NOEJPD), XOPTS(LINKAGE,SEQ,SP)
  ```

  組み込みの CICS 変換プログラムの場合、以下のように変更する必要があります。

  ```
  CBL TEST(NOEJPD), CICS('LINKAGE,SEQ,SP')
  ```

- 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してください。組み込みの CICS 変換プログラムは、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。組み込み変換プログラム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 変換プログラム

組み込み変換プログラムは、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。

組み込み変換プログラムを使用すると、COBOL コンパイラーは、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 変換プログラムとインターフェースをとります。組み込みの CICS 変換プログラム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 変換プログラムは依然として Enterprise COBOL でサポートされますが、組み込みの CICS 変換プログラムを使用することをお勧めします。組み込みの CICS 変換プログラム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 変換プログラムを使用する利点は以下のとおりです。

- Debug Tool 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 変換プログラム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変換プログラム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 は最外部のプログラムに生成され、ネストされたプログラムでは PROCEDURE DIVISION USING の GLOBAL 属性で指定されます。
- 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 ステートメントは EXEC CICS ステートメントに影響を及ぼすことができる。
- 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変換プログラムを使用する 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。

注: CICS 資料には、組み込み CICS 変換プログラムでコンパイルされたプログラムでは EXCI 変換プログラム・オプションがサポートされていないことが記述されていますが、CICS におけるこの見解は取り消されました。現在は、EXCI 変換プログラム・オプションでコンパイルを実行でき、警告メッセージ DFH7006I を無視することができます。

組み込みの CICS 変換プログラム用のコンパイラー・オプション

表 35 には、組み込みの CICS 変換プログラムを使用する 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 35. 組み込みの CICS 変換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 変換プログラム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 変換プログラム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、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、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 S レベルのメッセージを受け取って、廃棄されます。
NODYNAM	CICS 変換プログラム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 および ELPA は読取専用ストレージなので、モジュールに上書きすることはできません。

CICS 下での COBOL V5 プログラムから VS COBOL II プログラムへの静的呼び出し

VS COBOL II プログラムのある既存のアプリケーションはおそらく、Enterprise COBOL ライブラリーより前のバージョンでリンクされています。モジュールに VS COBOL II プログラムと静的にリンクされた COBOL V5 プログラムが含まれており、そのモジュールを CICS 下で実行する必要がある場合は、APAR PI33330 の PTF によって更新された SCEELKED の LE ライブラリー・モジュールを使用して、アプリケーションを REPLACE IGZEBST で再リンクする必要があります。

第 19 章 DB2 コプロセッサー移行における考慮事項

DB2 プリコンパイラーを使用するプログラムをアップグレードして、代わりに DB2 コプロセッサーを使用する際には、言語エレメントおよびコード・ページ変換における相違を理解する必要があります。

以下のトピックを考えてください。

- DB2 コプロセッサーの組み込み
- 言語エレメント
- コード・ページ変換

DB2 バージョン 8 からご使用の場合は、 OS/VS COBOL プログラムの DB2 プリコンパイラーを使用することはできません。さらに、OS/VS COBOL を Enterprise COBOL と一緒に使用することはできません。したがって、プログラムを変更する必要がある場合は、Enterprise COBOL にアップグレードする必要があります。

DB2 コプロセッサーの組み込み

コプロセッサーによって、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。

コプロセッサーでは、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 コプロセッサーとインターフェースをとります。DB2 コプロセッサー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は現在でも DB2 および Enterprise COBOL でサポートされますが、コプロセッサー・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。コプロセッサー・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、コプロセッサー・ソリューションを使用すると、Debug Tool 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。

コプロセッサー・アプローチには以下のような利点があります。

- ソースに EXEC SQL (および EXEC CICS) ステートメントが含まれていても、単一のジョブ・ステップで COBOL プログラムをコンパイルできる。
- COPY ステートメントを使用して、EXEC SQL ステートメントを含むソース・コードを組み込む機能が使用できる。
- Debug Tool 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。 DB2 分離プリコンパイラ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。

- 出力リストは 2 つではなく 1 つだけである。
- 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。
- 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。

以下のジョブ・ストリームは、DB2 プリコンパイラーの使用例を示しています。

```
//DB2PRE JOB ...,
// NOTIFY=GTAO,MSGCLASS=A,CLASS=A,TIME=(1,0),
// REGION=200M,MSGLEVEL=(1,1)
//PC       EXEC PGM=DSNHPC,
//         PARM='HOST(COB2),QUOTE,APOSTSQL,SOURCE,XREF'
//DBRMLIB DD DSN=GTAO.DBRMLIB.DATA(COBTEST),DISP=SHR
//STEPLIB  DD DSN=DSN910.SDSNLOAD,DISP=SHR
//SYSCIN   DD DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
//            SPACE=(800,(500,500))
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT2   DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSIN    DD *

        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.
         GOBACK.

//COB EXEC PGM=IGYCRCTL,
//PARM=(NODYNAM,'BUF(12288)',SOURCE,NOXREF)
//STEPLIB DD DSN=IGY.V5R1M0.SIGYCOMP,DISP=SHR
//        DD DSN=CEE.SCEERUN,DISP=SHR
//        DD DSN=CEE.SCEERUN2,DISP=SHR
//SYSIN    DD DSN=&&DSNHOUT,DISP=(OLD,DELETE)
//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)
//SYSUT8 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT9 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT10 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT11 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT12 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT13 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
```

```
//SYSUT14 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT15 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSMDECK DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
```

組み込みの SQL コプロセッサーの例を以下に示します。

```
//DB2INT JOB (GTAO,F342,090,M49),'Gianni Tao',
//NOTIFY=GTAO,MSGCLASS=A,CLASS=A,TIME=(1,0),
//REGION=200M,MSGLEVEL=(1,1)
//COB EXEC PGM=IGYCRCTL,
//PARM=(NODYNAM,'BUF(12288)',SOURCE,NOXREF,SQL)
//STEPLIB DD DSN=IGY.V5R1M0.SIGYCOMP,DISP=SHR
//        DD DSN=CEE.SCEERUN,DISP=SHR
//        DD DSN=CEE.SCEERUN2,DISP=SHR
//        DD DSN=DSN910.SDSNLOAD,DISP=SHR
//DBRMLIB DD DSN=GTAO.DBRMLIB.DATA(COBTEST),DISP=SHR
//SYSIN DD *

       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.
        GOBACK.

//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)
//SYSUT8 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT9 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT10 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT11 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT12 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT13 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT14 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT15 DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSMDECK DD UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
```

言語エレメント

分離プリコンパイラーと組み込みコプロセッサーでは、SQL コードの特定のアスペクトの処理方法に多少の違いがあります。コプロセッサーの使用に変更する場合、以下の項目を表示して、これらの違いを考慮してください。

継続行 **プリコンパイラー:** EXEC SQL ステートメントは 12 から 72 桁目で始める必要があります。ステートメントの継続行は、8 から 72 桁目のどの位置からでも始めることができます。

コプロセッサー: EXEC SQL ステートメントのすべての行は、継続行も含め、12 から 72 桁目でコード化する必要があります。

コプロセッサーへのマイグレーションのための処置: 8 から 11 桁目で始まる EXEC SQL ステートメントの任意の継続行を 12 から 72 桁目で始まるように移動します。

SQL INCLUDE と使用する COPY REPLACING

プリコンパイラー: EXEC SQL INCLUDE ステートメントは、ネストされた COPY . . . REPLACING ステートメントを含むコピーブックを参照できます。

コプロセッサー: EXEC SQL INCLUDE ステートメントは、ネストされた COPY . . . REPLACING ステートメントを含むコピーブックを参照できません。これは、 EXEC SQL INCLUDE はコプロセッサーによって COPY と全く同様に処理され、ネストされた COPY ステートメントは REPLACING を使用することができないためです。COPY REPLACING の代わりに REPLACE を使用することを考慮しなければならない場合もあります。統合コプロセッサーでは、REPLACE はコピーブックに対しても有効です。これは、独立したプリコンパイラーの場合には当てはまりません。

コプロセッサーへのマイグレーションのための処置: COPY REPLACING をコピーブックではなく、元の COBOL ソース・プログラムの中にのみ存在するようにご使用のコードを変更してください。

FOR BIT DATA ホスト変数

プリコンパイラー: COBOL 英数字データ項目を、サブタイプ FOR BIT DATA をもつ DB2 文字データを保持するためのホスト変数として使用できます。プリコンパイラーでは、問題のホスト変数を FOR BIT DATA として宣言する明示的な EXEC SQL DECLARE VARIABLE ステートメントは必要ありません。

コプロセッサー: COBOL 英数字データ項目は、以下の場合にのみ、サブタイプ FOR BIT DATA をもつ DB2 文字データを保持するためのホスト変数として使用できます。

- NOSQLCCSID コンパイラー・オプションを指定したか、または、
- ホスト変数用の明示的な EXEC SQL DECLARE VARIABLE ステートメントを COBOL プログラムに指定した。以下に例を示します。

```
EXEC SQL DECLARE :HV1 VARIABLE FOR BIT DATA END-EXEC
```

テーブルの COBOL 宣言の生成に DB2 DCLGEN コマンドを使用する場合、EXEC SQL DECLARE ステートメントを自動的に作成できます。このためには、DCLGEN コマンドに `DCLBIT(YES)` オプションを指定します。

コプロセッサーへのマイグレーションのための処置:

- DCLGEN を使用して、明示的な EXEC SQL DECLARE VARIABLE FOR BIT DATA ステートメントを、文字データとしてのみでなく、ビット・データとして使用される任意のデータ項目のデータ宣言に追加する。
- 明示的な EXEC SQL DECLARE VARIABLE FOR BIT DATA ステートメントを手動でデータ宣言に追加する。
- NOSQLCCSID コンパイラー・オプションを使用する。

ホスト変数の多重定義

プリコンパイラー: ホスト変数参照が固有である必要はありません。

有効な DB2 データ型にマップされる最初の定義が使用されます。

コプロセッサー: すべてのホスト変数参照は固有でなければなりません。

ホスト変数参照が固有でない場合、コプロセッサーはそれを非固有参照と診断します。ユーザーは、そのホスト変数参照を固有にするために、それを完全に修飾しなければなりません。

コプロセッサーへのマイグレーションのための処置: 複数の定義が存在するホスト変数参照を完全修飾します。

EXEC SQL INCLUDE ステートメントの終わりのピリオド

プリコンパイラー: ピリオドは必要ありません。

ピリオドが指定されると、プリコンパイラーはそれをステートメントの一部として処理します。ピリオドが指定されないと、プリコンパイラーはそのステートメントをピリオドが指定されたかのように受け入れます。

コプロセッサー: ピリオドは必須です。(コプロセッサーは、EXEC SQL INCLUDE ステートメントを COPY ステートメントのように扱います。)

例:

```
IF A = B THEN
    EXEC SQL INCLUDE somecode END-EXEC.
ELSE
    ...
END-IF
```

IF ステートメントはピリオドで終わらないことに注意してください。

コプロセッサーへのマイグレーションのための処置: 各

```
EXEC SQL INCLUDE somecode END-EXEC
```

ステートメントの後にピリオドを追加してください。

REPLACE および EXEC SQL ステートメント

プリコンパイラー: COBOL REPLACE ステートメントと COPY ステートメントの REPLACING 句は EXEC SQL ステートメントから作成された拡張ソースに基づいて動作します。

コプロセッサー: COBOL REPLACE ステートメントおよび COPY ステートメントの REPLACING 句は、EXEC ステートメントを含む元のソース・プログラムに基づいて動作します。その結果、以下の例においては動作が異なることがあります。

```
REPLACE ==ABC ==By ==XYZ ==.
01 G.
02 ABC PIC X(10).
...
EXEC SQL SELECT *INTO :G.ABC FROM TABLE1 END-EXEC
```

プリコンパイラーでは、G.ABC への参照は拡張ソース内で ABC OF G として表示され、XYZ OF G で置き換えられます。コプロセッサーでは、ABC が元のソース・ストリング G.ABC において区切り文字によって区切られないために、置き換えは発生しません。

コプロセッサーへのマイグレーションのための処置: 以下のようにご使用のコードを変更して、非修飾参照だけでなく修飾参照 (例えば G.ABC) も置き換えます。

```
REPLACE ==ABC ==By ==XYZ ==
==G.ABC ==By ==G.XYZ ==.
```

または、修飾が不要になるようにコードを変更するか、そのようなデータ項目に対して REPLACE の使用を止めるか、 REPLACE によって変更される COBOL プログラムがきれいにコンパイルされるようにその他の方法を実行します。

END-EXEC に続くソース・コード

プリコンパイラー: 同じ行の END-EXEC に続くコードを無視します。

コプロセッサー: 同じ行の END-EXEC に続くコードを処理します。

コプロセッサーへのマイグレーションのための処置: 浮動コメント標識 *> を END-EXEC 句の後に追加します。

SQL-INIT-FLAG

プリコンパイラー: プログラムが複数回呼び出される際に、異なるアドレスにある可能性のあるホスト変数を渡す場合、呼び出し先プログラムが SQL-INIT-FLAG をリセットする必要があります。このフラグをリセットするということは、次の SQL ステートメントの実行時に、ストレージの初期設定が必要であることを DB2 に指示することを意味します。このフラグをリセットするには、呼び出し先プログラムの PROCEDURE DIVISION で、ホスト変数を使用するすべての実行可能な SQL ステートメントより前に、ステートメント `MOVE ZERO TO SQL-INIT-FLAG` を挿入します。

コプロセッサー: 呼び出し先プログラムは、SQL-INIT-FLAG をリセットする必要はありません。プログラムの移植性を高めるために、SQL-INIT-FLAG がプログラム内に自動的に定義されます。ただし、`MOVE ZERO TO SQL-INIT-FLAG` など、SQL-INIT-FLAG を変更するステートメントは、プログラムでの SQL 処理には何の影響もありません。

コプロセッサーへのマイグレーションのための処置: オプションで、SQL-INIT-FLAG への参照を除去します。この参照は使用しないため不要です。

コード・ページ変換

分離プリコンパイラーと組み込みコプロセッサーでは、文字変換の処理方法に違いがあります。コプロセッサーの使用に変更する場合、以下の項目を表示して、これらの違いを考慮してください。

SQL ステートメントのための COBOL および DB2 間のコード・ページ調整

プリコンパイラー: 調整はありません。SQL ステートメントの処理に対するコード・ページは、DB2 外部メカニズムおよびデフォルトによって決定されます。

コプロセッサー: SQL ステートメントのための COBOL および DB2 間のコード・ページ調整は、以下の SQLCCSID コンパイル・オプションによって異なります。

- SQLCCSID:
  - COBOL CODEPAGE(ccsid) コンパイラー・オプションは、COBOL ステートメントおよび SQL ステートメントのホスト変数の処理に影響を与えます。
  - CCSID 処理は、 Enterprise COBOL V3R4 での SQL コプロセッサーと互換性があります。
- NOSQLCCSID:
  - CODEPAGE(ccsid) コンパイラー・オプションは、COBOL ステートメントの処理に影響を与えるのみで、 SQL ステートメントの処理には使用されません。
  - SQL ステートメントの処理に対するコード・ページは、DB2 外部メカニズムおよびデフォルトによって決定されます。

SQLCCSID および NOSQLCCSID の詳細については、『*Enterprise COBOL for z/OS プログラミング・ガイド*』の「COBOL および DB2 CCSID 判別」のセクションを参照してください。

第 20 章 IMS プログラムを Enterprise COBOL V5 に移動する

COBOL を IMS 出口ルーチンに使用する場合、COBOL V5 におけるいくつかの制限について注意してください。

拡張サービスに対応している IMS 出口のみ、PDSE データ・セットに常駐できます。特に、COBOL ユーザーは一般に 2 つのタイプの出口を使用し、それらは PDSE データ・セットの外部での実行には対応していません。

```
DFSME127   The Input Message Segment Edit user exit
DFSME000   The Input Message Field Edit user exit
```

このようなタイプの IMS ユーザー出口として使用されている COBOL プログラムは、COBOL V5 ではコンパイルできません。実際の出口ルーチンが、PDSE データ・セット内の COBOL V5 プログラムをロードして呼び出す、PDS データ・セット内のアセンブラー・プログラムである場合は例外です。 COBOL V5 とこのようなユーザー出口を操作する場合、以下のように処理してください。

- 出口ルーチンが COBOL であれば、COBOL V5 で再コンパイルせずに、古いバージョンの COBOL を引き続き使用します。
- 出口ルーチンが COBOL であれば、COBOL V5 をロードするアセンブラー・プログラム使用するように変更するか、または出口ロジック用の COBOL V5 の動的呼び出しを行う古い COBOL を使用するように変更します。
- 出口ルーチンが、COBOL プログラムをロードするアセンブラーであれば、その COBOL プログラムを COBOL V5 で再コンパイルし、PDSE データ・セットにバインドし、その新しいデータ・セットを連結に追加します。

IMS では、さまざまなユーザー出口を拡張サービスで使用できるように対応を進めているところです。これにより、PDSE データ・セットの外部でユーザー出口を実行できるようになります。 IMS V11 において、新しいサービスに対応しているユーザー出口タイプのリスト:

```
ICQSEVNT(new)          The IMS CQS Event user exit
ICQSSEVT(new)          The IMS CQS Structure Event user exit
INITTERM(new)          The Initialization / Termination user exit
RESTART(new in IMS 10)The Restart user exit
PPUE (DSFSPPUE0)       The Partner Product user exit
```

IMS 12 で新たに有効になった出口はありません。

IMS 13 では、以下のユーザー出口タイプが有効です。

```
BSEX (DFSBSEX0)      The Build Security Environment user exit
LOGEDIT (DFSFLGE0)   The Log Edit user exit
LOGWRT (DFSFLGX0)    The Logger user exit
NDMX (DFSNDMX0)      The Non-Discardable Message user exit
OTMAIOED (DFSYIOE0)  The OTMA Input / Output Exit user exit
OTMARTUX (DFSYRTUX)  The OTMA Resume TPIPE Security user exit
OTMAYPRX (DFSYPRX0)  The OTMA Destination Resolution user exit
RASE (DFSRAS00)      The Resource Access Security user exit
```

IMS のもとで実行するため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には、 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 では、COBOL プログラムをプリロードしておくことができます。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 プログラムを含むプログラム・オブジェクトをプリロードするときは、そのプログラム・オブジェクト内のすべての 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、IBM COBOL、および VS COBOL II のプログラムが混在しているアプリケーションの場合、以下のコンパイラー・オプションが推奨されます。

表 36. 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
RENT	RENT	RENT および RES

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 または ELPA に入れることができます。そこでは、それらのプログラムを複数の IMS 従属領域で共有できます。

アプリケーション・プログラムは、16-MB 境界より上で実行するためには、RENT および RMODE(ANY) でコンパイルする必要があります。

IMS では、IMS アプリケーション・プログラム用のデータは 16-MB 境界より上に存在でき、IMS サービスを使用するプログラムには DATA(31) および RENT 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 プログラム・オブジェクト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プログラム・オブジェクトをリンクするには、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。

パフォーマンス用の LLA 管理ロード・ライブラリー

パフォーマンスのために Library Lookaside (LLA) を使用して COBOL ロード・モジュール (PDS ロード・ライブラリー) とプログラム・オブジェクト (PDSE ロード・ライブラリー) のキャッシングを管理する場合、COBOL V5 モジュールは LLA によってキャッシングされません。これは、結果として生じるプログラム・オブジェクトの構造が原因です。 IBM は、この制限を取り除くことは可能でした。その間に、PDSE ハイパースペース・キャッシングの有効化によるパフォーマンス向上が得られるようになりました。このキャッシングは、ロード・ライブラリー内の COBOL V5 プログラム・オブジェクトが LLA によって管理されている場合に役立ちます。

大きなプログラム・オブジェクトをロードするための手法の 1 つに、ページ不在主導型ローディングと呼ばれるものがあります。初期ロードは、プログラム・オブジェクトの一部にのみ適用されます。プログラム・オブジェクトの他の部分には、それらが参照される場合 (つまり、ページ不在が発生した場合) にのみ適用可能です。LLA によって管理されているライブラリーでは、ページ不在主導型ローディングを回避することで、パフォーマンスが向上する場合があります。プログラム・オブジェクトでバインダー・オプション `FETCHOPT=(NOPACK,PRIME)` を使用すると、システムはページ不在主導型ローディングを使用しません。デフォルトのバインダー・オプション `FETCHOPT=(NOPACK,NOPRIME)` では、ページ不在主導型ローディングが使用可能になります。

注: LLA によって管理されていないライブラリーにプログラム・オブジェクトがある場合は、デフォルトのバインダー・オプション `FETCHOPT=(NOPACK,NOPRIME)` によってパフォーマンスが向上する可能性があります。

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 Language Environment 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。

- 互換性
- Language Environment とのリンク・エディット
- Enterprise COBOL でのコンパイル
- Language Environment サービス
- Language Environment ランタイム・オプション
- サブシステム
- z/OS
- パフォーマンス
- サービス
- オブジェクト指向構文、Java 6 SDK、Java 7 SDK、および Java 8 SDK

互換性

OS/VS COBOL プログラムおよび 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?

CICS 以外では、OS/VS COBOL と Enterprise COBOL の間の呼び出しはサポートされていません。 CICS では、OS/VS COBOL プログラムはまったく実行できません。

CICS 以外では、VS COBOL II NORES プログラム (すなわち、NORES コンパイラー・オプションでコンパイルされたプログラム) と Enterprise COBOL の間の呼び出しはサポートされていません。 CICS では、VS COBOL II NORES プログラムはまったく実行できません。

非 CICS 呼び出しおよび CICS では、VS COBOL II RES プログラムと Enterprise COBOL プログラム間の呼び出しはすべてサポートされます。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 (Language Environment 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、279 ページの『CICS のもとでのアセンブラー 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。

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 ?

はい。ただし、アプリケーションに OS/VS COBOL プログラムが含まれる場合を除きます。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。

出力 DD のつづりに誤りがある COBOL プログラムを実行するとエラーが発生し、一時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、Language Environment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。旧コンパイラーで CMPR2 を使用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 85 COBOL 標準にアップグレードする必要があります。

詳細については、 118 ページの『CMPR2 コンパイラー・オプションから NOCMPR2 へのマイグレーション』を参照してください。

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*」に記載されているので、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。

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。

詳細については、 17 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。

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、 17 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 または VS COBOL II ソースを Enterprise COBOL ソースに移行するときには、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. IBM Debug Tool 製品に組み込まれている COBOL 移行援助プログラム (CCCA) は、OS/VS COBOL ソースおよび VS COBOL II ソースを Enterprise COBOL ソースに変換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。また、そのコードを Enterprise COBOL で継続して使用することも可能です。
3. Debug Tool ロード・モジュール・アナライザーは、プログラム・オブジェクト内の各オブジェクトの言語変換プログラムを決定できます。Debug Tool ロード・モジュール・アナライザーは、IBM Debug Tool 製品に組み込まれています。
4. Edge Portfolio Analyzer は、使用されたコンパイラー、コンパイラーのリリース、およびコンパイラー・オプションを報告することによって、既存プログラム・オブジェクトの目録の作成を支援します。

   Edge Portfolio Analyzer は現在 IBM では販売しておりません。Edge Portfolio Analyzer について詳しくは、www.edge-information.com にアクセスしてください。
5. Rational Asset Analyzer for System z (製品番号 5655-W57) は、目録を作成し、コード変更が企業の資産に及ぼす影響を分析するために役立ちます。

Enterprise COBOL は 85 COBOL 標準に適合していますか?

はい。Enterprise COBOL は、85 COBOL 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。

Enterprise COBOL は 2002 COBOL 標準に適合していますか?

Enterprise COBOL では、2002 COBOL 標準の多くの部分がサポートされています。

Enterprise COBOL プログラムのバインド (リンク・エディット)

オブジェクト・モジュール、ロード・モジュール、およびプログラム・オブジェクトの違いは何ですか?

オブジェクト・モジュールは、コンパイラーの出力であり、バインダーへの入力でもあります。ロード・モジュールは、Enterprise COBOL V4 以前のオブジェクト・モジュールでバインダーから出力される非 GOFF 実行可能ファイルです。プログラム・オブジェクトは、Enterprise COBOL V5.1 からのオブジェクト・モジュールのバインド時に、またはターゲット・データ・セット (SYSLMOD) が PDSE である場合は常に、バインダーから出力される新しいスタイルの GOFF 実行可能ファイルです。

Enterprise COBOL V5 でオブジェクト・モジュール・モジュールに対して PDS データ・セットおよび PDSE データ・セットは許可されていますか?

コンパイラー出力データ・セットは、オブジェクト・モジュールを含め、PDS または PDSE にすることができます。バインド・ステップの出力は PDSE にする必要があります。バインド (リンク・エディット) された COBOL オブジェクト・モジュールは、プログラム・オブジェクトになり、PDSE データ・セットに保管する必要があります。

Language Environment サービス

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働する複数のプログラムのサポートです。これは、pthread_create を呼び出す COBOL プログラム、または「pthread create」を実行する C プログラムによって開始できます。 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。

注: COBOL マルチスレッド化は、CICS の概念「スレッド・セーフ」には関連していません。

Language Environment ランタイム・オプション

Enterprise COBOL V5.1.1 は WORKING-STORAGE に HEAP を使用しますか?

COBOL プログラムが RENT オプションでコンパイルされていれば、以下のいずれかの場合、WORKING-STORAGE に HEAP が使用されます。

- DATA(24) コンパイラー・オプションでコンパイルされた
- CICS で稼働している
- COBOL V5.1.1 が、COBOL プログラム (V5.1.1、V4.2 以前) およびアセンブリー・プログラムのみが入っているプログラム・オブジェクトにある。プログラム・オブジェクト内に言語環境プログラム言語間呼び出しがなく、COBOL V5.1.0 プログラムもありません。
- COBOL V5 プログラムが、メイン・エントリー・ポイントが COBOL V5 であるプログラム・オブジェクトにある。この場合、COBOL が静的に C、C++、または PL/I にリンクしている状態で、プログラム・オブジェクトに言語環境プログラム言語間呼び出しが入ることが可能です。このようなプログラム・オブジェクト内のすべての COBOL V5 プログラムでは、(それらがメイン・エントリー・ポイントでなくても) WORKING-STORAGE がヒープ・ストレージから割り振られています。

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。

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、Language Environment 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、Language Environment 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。

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 変換プログラムは、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 変換プログラムによって変換されたプログラムの場合は NODYNAM コンパイラー・オプションが必要であり、)

CALL 'CEETDLI' は CICS プログラム内でサポートされますか ? Language Environment 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。) Language Environment 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 Language Environment サービスまたはユーザー作成の Language Environment 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。

Language Environment (および、COBOL プログラムと PL/I プログラムの混合に対するそのサポート) 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 IBM 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 に移行され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。

Language Environment 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。Language Environment 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。

CICS では、VS COBOL II と PL/I の混合は許可されません。

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サポートされません。IMS のもとで CBLTDLI を使用するときには、Language Environment 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。詳細については、「*Language Environment プログラミング・ガイド*」を参照してください。

z/OS

COBOL は 64 ビットの z/OS で稼働しますか ?

はい。COBOL は COBOL プログラムで 64 ビットのアドレッシングをサポートしていませんが、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?

はい。Enterprise COBOL V5 では、すべての古い COBOL コンパイラーと比べて、パフォーマンスが大幅に向上しています。特に、算術計算を多用するプログラムでパフォーマンスが向上します。

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、サポートされているランタイムと共に実行されている場合、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、23 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。

オブジェクト指向構文、Java 6 SDK、Java 7 SDK、および Java 8 SDK

Java 6、Java 7、および Java 8 を使用して既存の COBOL アプリケーションを実行するにはどうすればいいですか?

Java インターオペラビリティーのためにオブジェクト指向構文を使用する旧バージョンの Enterprise COBOL アプリケーションは、Java SDK 1.4.2 および Java 5 でサポートされていました。

これらのアプリケーションを Java 6、Java 7、または Java 8 で実行するには、以下の手順を実行します。

1. Enterprise COBOL V5.2 を使用してアプリケーションを再コンパイルおよび再リンクします。
2. Java 6、Java 7、または Java 8 の `javac` コマンドを使用して各オブジェクト指向 COBOL クラスに関連付けられている、生成済み Java クラスを再コンパイルします。

付録 B. COBOL 予約語の比較

下表に、OS/VS COBOL、VS COBOL II、IBM COBOL、および Enterprise COBOL での予約語における相違点を示します。

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。

- 47 ページの『第 5 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 99 ページの『第 7 章 VS COBOL II ソース・プログラムのアップデート』
- 111 ページの『第 9 章 IBM COBOL ソース・プログラムのアップグレード』
- 163 ページの『第 11 章 Enterprise COBOL バージョン 3 からプログラムのアップグレード』
- 177 ページの『第 13 章 Enterprise COBOL バージョン 4 からのアップグレード』

太字のテキストは、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) を示しています。

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 以降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。

X** Enterprise COBOL 列では、そのワードは Enterprise COBOL バージョン 4 リリース 1 以降でのみ予約されています。Enterprise COBOL バージョン 3 以前では予約されていません。

X*** Enterprise COBOL 列では、そのワードは Enterprise COBOL バージョン 4 リリース 2 以降で予約されています。Enterprise COBOL バージョン 4 リリース 1 以前では予約されていません。

X****** Enterprise COBOL 列では、そのワードは Enterprise COBOL バージョン 5 リリース 1 で予約されています。そのワードは Enterprise COBOL バージョン 4 リリース 2 以前では予約されていません。

X***
Enterprise COBOL 列では、そのワードは Enterprise COBOL バージョン 5 リリース 2 で予約されています。Enterprise COBOL バージョン 5 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。

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 85 COBOL 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 37. 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
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
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-NOT	RFD	RFD	RFD	-

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
BINARY	X	X	X	-
BINARY-CHAR	RFD	-	-	-
BINARY-DOUBLE	RFD	-	-	-
BINARY-LONG	RFD	-	-	-
BINARY-SHORT	RFD	-	-	-
BIT	RFD	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	-

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
COMMA	X	X	X	X
COMMON	X	X	X	-
COMMUNICATION	UNS	UNS	UNS	X
COMP	X	X	X	X
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	-
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	
COMPUTE	X	X	X	X
CONDITION	RFD	-	-	-
CONFIGURATION	X	X	X	X
CONSOLE	-	-	-	X
CONSTANT	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
CORRESPONDING	X	X	X	X
COUNT	X	X	X	X
CRT	RFD	-	-	-
CSP	-	-	-	X
CURRENCY	X	X	X	X
CURRENT-DATE	-	-	-	X
CURSOR	RFD	-	-	-
C01	-	-	-	X
C02	-	-	-	X
C03	-	-	-	X
C04	-	-	-	X

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
C05	-	-	-	X
C06	-	-	-	X
C07	-	-	-	X
C08	-	-	-	X
C09	-	-	-	X
C10	-	-	-	X
C11	-	-	-	X
C12	-	-	-	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	-
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
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
DISP	-	-	-	X
DISPLAY	X	X	X	X

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
DISPLAY-ST	-	-	-	X
DISPLAY-1	X	X	X	-
DIVIDE	X	X	X	X
DIVISION	X	X	X	X
DOWN	X	X	X	X
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
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-DISPLAY	RFD	-	-	-
END-DIVIDE	X	X	X	-
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-START	X	X	X	-
END-STRING	X	X	X	-
END-SUBTRACT	X	X	X	-
END-UNSTRING	X	X	X	-
END-WRITE	X	X	X	-

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
ERROR	X	X	X	X
ESI	UNS	UNS	UNS	X
EVALUATE	X	X	X	-
EVERY	X	X	X	X
EXAMINE	-	-	-	X
EXCEPTION	X	X	X	X
EXCEPTION-OBJECT	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
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
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

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
FUNCTION	X	X	-	-
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	X	-	-	-
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
INDEXED	X	X	X	X
INDICATE	UNS	UNS	UNS	X
INHERITS	X	X	-	-
INITIAL	X	X	X	X
INITIALIZE	X	X	X	-
INITIATE	UNS	UNS	UNS	X
INPUT	X	X	X	X
INPUT-OUTPUT	X	X	X	X
INSERT	CDW	CDW	CDW	X
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

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
JNIENVPTR	X	-	-	-
JUST	X	X	X	X
JUSTIFIED	X	X	X	X
KANJI	X	X	X	-
KEY	X	X	X	X
LABEL	X	X	X	X
LAST	UNS	UNS	UNS	X
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	-
LINE	X	X	X	X
LINE-COUNTER	UNS	UNS	UNS	X
LINES	X	X	X	X
LINKAGE	X	X	X	X
LOCAL-STORAGE	X	X	-	-
LOCALE	RFD	-	-	-
LOCK	X	X	X	X
LOW-VALUE	X	X	X	X
LOW-VALUES	X	X	X	X
MEMORY	X	X	X	X
MERGE	X	X	X	X
MESSAGE	UNS	UNS	UNS	X
METACLASS	-	X	-	-
METHOD	X	X	-	-
METHOD-ID	X	X	-	-
MINUS	RFD	-	-	-
MODE	X	X	X	X
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

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
NATIONAL	X	-	-	-
NATIONAL-EDITED	X	-	-	-
NATIVE	X	X	X	X
NEGATIVE	X	X	X	X
NESTED	RFD	-	-	-
NEXT	X	X	X	X
NO	X	X	X	X
NOMINAL	-	-	-	X
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
オフ	X	X	X	X
OMITTED	X	X	X	X
ON	X	X	X	X
OPEN	X	X	X	X
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	-	-
PACKED-DECIMAL	X	X	X	-
PADDING	X	X	X	-
PAGE	X	X	X	X
PAGE-COUNTER	UNS	UNS	UNS	X
PASSWORD	X	X	X	X

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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	-
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
PROGRAM-POINTER	RFD	-	-	-
PROPERTY	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
RECEIVE	UNS	UNS	UNS	X
RECORD	X	X	X	X
RECORD-OVERFLOW	-	-	-	X
RECORDING	X	X	X	X
RECORDS	X	X	X	X

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
RECURSIVE	X	X	-	-
REDEFINES	X	X	X	X
REEL	X	X	X	X
REFERENCE	X	X	X	-
REFERENCES	X	X	X	X
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
REPLACE	X	X	X	-
REPLACING	X	X	X	X
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	-	-	-
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
ROUNDED	X	X	X	X
RUN	X	X	X	X
SAME	X	X	X	X
SCREEN	RFD	-	-	-
SD	X	X	X	X

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
SEQUENCE	X	X	X	X
SEQUENTIAL	X	X	X	X
SERVICE	X	X	X	X
SET	X	X	X	X
SHARING	RFD	-	-	-
SHIFT-IN	X	X	X	-
SHIFT-OUT	X	X	X	-
SIGN	X	X	X	X
SIZE	X	X	X	X
SKIP1	CDW	CDW	CDW	X
SKIP2	CDW	CDW	CDW	X
SKIP3	CDW	CDW	CDW	X
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*	-	-

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
STANDARD	X	X	X	X
STANDARD-1	X	X	X	X
STANDARD-2	X	X	X	-
START	X	X	X	X
STATUS	X	X	X	X
STOP	X	X	X	X
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	-	-	-	X
SYSLIST	-	-	-	X
SYSOUT	-	-	-	X
SYSPUNCH	X	X	X	X
SYSTEM-DEFAULT	RFD	-	-	-
S01	-	-	-	X
S02	-	-	-	X
TABLE	UNS	UNS	UNS	X
TALLY	X	X	X	X
TALLYING	X	X	X	X
TAPE	X	X	X	X
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
THRU	X	X	X	X
TIME	X	X	X	X
TIME-OF-DAY	-	-	-	X

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
TRANSFORM	-	-	-	X
TRUE	X	X	X	-
TYPE	X	X*	-	-
TYPEDEF	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	-	-	-	X
UPSI-1	-	-	-	X
UPSI-2	-	-	-	X
UPSI-3	-	-	-	X
UPSI-4	-	-	-	X
UPSI-5	-	-	-	X
UPSI-6	-	-	-	X
UPSI-7	-	-	-	X
USAGE	X	X	X	X
USE	X	X	X	X
USER-DEFAULT	RFD	-	-	-
USING	X	X	X	X
VAL-STATUS	RFD	-	-	-
VALID	RFD	RFD	RFD	-
VALIDATE	RFD	RFD	RFD	-
VALIDATE-STATUS	RFD	-	-	-

表 37. 予約語の比較 (続き)

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
VALUE	X	X	X	X
VALUES	X	X	X	X
VARYING	X	X	X	X
VOLATILE	X*****	-	-	-
WHEN	X	X	X	X
WHEN-COMPILED	X	X	X	X
WITH	X	X	X	X
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-INFORMATION	X***	-	-	-
XML-NAMESPACE	X**	-	-	-
XML-NAMESPACE-PREFIX	X**	-	-	-
XML-NNAMESPACE	X**	-	-	-
XML-NNAMESPACE-PREFIX	X**	-	-	-
XML-NTEXT	X	-	-	-
XML-SCHEMA	X***	-	-	-
XML-TEXT	X	-	-	-
ZERO	X	X	X	X
ZEROES	X	X	X	X
ZEROS	X	X	X	X
_	X***	-	-	-
<	X	X	X	X
<>	RFD			
<=	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****	-	-	-
::	RFD			

付録 C. ソース・プログラム用の移行ツール

OS/VS COBOL、VS COBOL II、または IBM COBOL のソース・プログラムを Enterprise COBOL にアップグレードするために役立つ、いくつかの移行ツールが使用可能です。

この付録には、移行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)
- 移行をサポートするその他のプログラム

この付録は、使用すべきツール (ある場合) を判別し、それらを使用する方法を理解し、そ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ために役立ちます。

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、MIGR 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上で役立つため、計画されている将来の移行を容易に行うことができます。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 の機能のために導入された新規予約語 (以前は有効であったユーザー・ワードが今は正しくない可能性があります)
- 別の方法でサポートされる言語機能
- サポートされない言語機能

MIGR コンパイラー・オプションは、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

- 複合簡略比較条件の変更
- 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 . . .
- OCCURS DEPENDING ON 文節の変更
- ON SIZE ERROR オプションの中間結果における変更
- 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 または索引名 (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。
- 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
- 範囲外の添え字の評価における変更
- 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

下のリンクから、Enterprise COBOL においてより高い正確度でサポートされる OS/VS COBOL ステートメントを確認できます。これらのステートメントには、Enterprise COBOL ではより正確な結果が得られ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 ステートメントと比較の位取りにおける変更
- 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

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。274 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)
- I/O 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
- I/O 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
- 修飾された索引名

  (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
- 主語ではなく目的語として KEY 項目を使用する 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。

- Debug Tool ロード・モジュール・アナライザーは、プログラム・オブジェクト内の各オブジェクトの言語変換プログラムを決定できます。

  Debug Tool ロード・モジュール・アナライザー は、 Debug Tool に組み込まれています。
- 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は、 Debug Tool に組み込まれています。
- 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

Rational Asset Analyzer

Rational Asset Analyzer には、企業資産の目録を生成し、コード変更に必要となる相対労力の指標を返すツールが備わっています。

COBOL および CICS/VS コマンド・レベル移行援助プログラム (CCCA)

IBM Debug Tool 製品に組み込まれている COBOL および CICS/VS コマンド・レベル移行プログラム (CCCA) 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 は、PTF for APAR PM86253 によって、Enterprise COBOL バージョン 5.1 の予約語変換用に更新されています。バージョン 5.2 では、APAR PI32750 の PTF によって、予約語変換に関して CCCA が更新されています。

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 の主要な機能は以下のとおりです。

- 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 (リンケージ用ベース・ロケーター) セクションのメカニズムおよび参照の削除または変換

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。

詳しくは、「*COBOL および CICS/VS コマンド・レベル移行援助プログラム*」資料を参照してください。

CCCA を使用してよいとき

アプリケーションを OS/VS COBOL、VS COBOL II、または IBM COBOL 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 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 38 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 38. 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
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 ...

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 (製品番号 5798-DYR) を使用して、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ンパイルしたり、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久的に変換したり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 コンパイラーとの統合 (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統合する)。
- 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。

Debug Tool ロード・モジュール・アナライザー

デバッグ・ツール・ロード・モジュール・アナライザーは、プログラム・オブジェクトを分析して、各 CSECT のオブジェクトを生成するために使用された言語変換プログラム (コンパイラーまたはアセンブラー) を特定します。

このプログラムは、PDS または PDSE データ・セットの連結内のすべてまたは選択したプログラム・オブジェクトを処理できます。ロード・モジュール・アナライザーは、IBM Debug Tool 製品に組み込まれています。

Edge Portfolio Analyzer

Edge Portfolio Analyzer は、使用されたコンパイラー、コンパイラーのリリース、およびコンパイラー・オプションを報告することによって、既存プログラム・オブジェクトの目録の作成を支援します。

Edge Portfolio Analyzer は現在 IBM では販売しておりません。Edge Portfolio Analyzer について詳しくは、www.edge-information.com にアクセスしてください。

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

アプリケーションに COBOL プログラムとアセンブラー・プログラムが混在している場合、アプリケーションにいくつかの変更を行わなければならないことがあります。

必要に応じて以下の作業を行います。

- 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 のもとでサポートされるアセンブラー /COBOL 呼び出しの判別
- CICS のもとでサポートされるアセンブラー /COBOL 呼び出しの判別
- 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
- アセンブラーが COBOL プログラムのロード、呼び出し、または削除を行うアプリケーションの変更
- Enterprise COBOL V5 を呼び出す、または Enterprise COBOL V5 から呼び出されるアセンブラー・プログラムで汎用レジスター (GPR) の高位半分を保存および復元する

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一部の情報は、本書の他のセクションに記載されています。例えば、プロシージャー名を渡すアセンブラー・プログラムに関する情報は、76 ページの『OS/VS COBOL から変更された言語エレメント』に記載されています。

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存域に保管しなければなりません。特に、COBOL 保存域は、アセンブラー・プログラムの保存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 保存域のアドレスを 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
- 呼び出された時に使われていたものと同じ AMODE で、COBOL 呼び出し元へ戻す。

SVC LINK および COBOL 実行単位の境界

SVC LINK のターゲットが非 Language Environment 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、Language Environment 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 Language Environment 準拠のアセンブラー・プログラムである場合は、Language Environment 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 のもとでのアセンブラー COBOL 呼び出しのためのランタイム・サポート

次の表に、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組み合わせ、および非 CICS での Language Environment 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がリストされています。

サポートされない 呼び出しについては、ほとんどの場合に返される症状 (メッセージまたは異常終了コード) も表 39 にリストされています。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。

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 39. Language Environment でサポートされる、非 CICS での COBOL プログラムとアセンブラー・プログラム間の呼び出し。「はい」は、呼び出しがサポートされることを示します。

呼び出し元		発行先						
呼び出しタイプ	**発行側プログラム**	**Enterprise COBOL**	**IBM COBOL**	**VS COBOL II**	**OS/VS COBOL**	**LanEnv[1] Asm[2] メイン**	**LanEnv[1] Asm サブルーチン**	**非LanEnv Asm**
静的	Enterprise COBOL	はい	はい	はい	いいえ	いいえ[3]	はい	はい
	IBM COBOL	はい	はい	はい	はい	いいえ[3]	はい	はい
	VS COBOL II (RES あり)	はい	はい	はい	はい	いいえ[3]	はい[4]	はい
	VS COBOL II (NORES あり)	いいえ	はい	はい	はい	いいえ[3]	はい[4]	はい
	OS/VS COBOL	いいえ	はい	はい	はい	いいえ[3]	はい[4]	はい

表 39. Language Environment でサポートされる、非 CICS での COBOL プログラムとアセンブラー・プログラム間の呼び出し。「はい」は、呼び出しがサポートされることを示します。 (続き)

呼び出し元		発行先						
呼び出しタイプ	**発行側プログラム**	**Enterprise COBOL**	**IBM COBOL**	**VS COBOL II**	**OS/VS COBOL**	**LanEnv[1] Asm[2] メイン**	**LanEnv[1] Asm サブルーチン**	**非LanEnv Asm**
動的	Enterprise COBOL	はい	はい	はい	いいえ	いいえ[3]	はい	はい
	IBM COBOL	はい	はい	はい	はい	いいえ[3]	はい	はい
	VS COBOL II (RES あり)	はい	はい	はい	はい	いいえ[3]	はい	はい
	VS COBOL II (NORES あり)	いいえ	はい	はい	はい	いいえ[3]	はい	はい
	OS/VS COBOL	いいえ	はい	はい	はい	いいえ[3]	はい	はい
VCON	Asm (LanEnv)	はい	はい	はい	はい	いいえ[3]	はい	はい
	Asm (非LanEnv)	はい	はい	はい	はい	はい[5]	いいえ[6]	はい
LOAD	Asm (LanEnv)	はい	はい	はい	はい	いいえ[3]	はい	はい
BALR	Asm (非LanEnv)	はい	はい	はい	はい	はい[5]	いいえ[6]	はい
LINK	Asm (LanEnv)	はい	はい	はい	はい[7]	はい	いいえ[6]	はい
	Asm (非LanEnv)	はい	はい	はい	はい[7]	はい	いいえ[6]	はい

以下の注記で説明されている障害症状は、Language Environment 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に発生します。

1. (LanEnv は Language Environment を表します。) MAIN=YES を指定した CEEENTRY マクロは、Language Environment 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、Language Environment 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. (Asm はアセンブラーを表します。)
3. 確立された Language Environment エンクレーブ内から Language Environment 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働します。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。
4. 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。
5. 非 Language Environment アセンブラー呼び出し元が、確立された Language Environment エンクレーブ内で稼働している場合は、注 3 を参照してください。
6. 障害症状 0C1、0C4、または 0C5 異常終了。
7. OS/VS COBOL プログラムが、別の確立された Language Environment 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 のもとでのアセンブラー COBOL 呼び出しのためのランタイム・サポート

次の表に、COBOL プログラムおよびアセンブラー・プログラムに関係のある呼び出しの組み合わせ、および CICS での Language Environment 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がリストされています。

サポートされない 呼び出しについては、ほとんどの場合に返される症状 (メッセージまたは異常終了コード) も表 40 にリストされています。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。

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 40. Language Environment がサポートする、CICS で実行される COBOL プログラムとアセンブラー・プログラム間の呼び出し。「はい」は、呼び出しがサポートされることを示します。

呼び出し元		発行先					
呼び出しタイプ	**発行側プログラム**	**Enterprise COBOL**	**IBM COBOL**	**VS COBOL II**	**LanEnv[1] Asm[2] メイン**	**LanEnv[1] Asm サブルーチン**	**非LanEnv Asm**
静的	Enterprise COBOL	はい	はい	はい	いいえ[3]	はい	はい
	IBM COBOL	はい	はい	はい	いいえ[3]	いいえ[4]	はい
	VS COBOL II	はい	はい	はい	いいえ[3]	いいえ[4]	はい
動的	Enterprise COBOL	はい	はい	はい	いいえ[3]	はい	はい
	IBM COBOL	はい	はい	はい	いいえ[3]	はい	はい
	VS COBOL II	はい	はい	はい	いいえ[3]	はい	はい
EXEC CICS LINK	Enterprise COBOL	はい	はい	はい	いいえ[3]	いいえ[4]	はい
	IBM COBOL	はい	はい	はい	いいえ[3]	いいえ[4]	はい
	VS COBOL II	はい	はい	はい	いいえ[3]	いいえ[4]	はい
VCON	Asm (LanEnv)	はい	はい	いいえ[4]	いいえ[3]	はい	はい
	Asm (非LanEnv)	いいえ[4]	いいえ[4]	いいえ[4]	いいえ[3]	いいえ[4]	はい
EXEC CICS LINK	Asm (非LanEnv)	はい	はい	はい	いいえ[3]	いいえ[4]	はい
	Asm (非LanEnv)	はい	はい	はい	いいえ[3]	いいえ[4]	はい

以下の注記で説明されている障害症状は、Language Environment 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に発生します。

1. (LanEnv は Language Environment を表します。) MAIN=YES を指定した CEEENTRY マクロは、Language Environment 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、Language Environment 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. (Asm はアセンブラーを表します。)
3. CICS TS バージョン 3 より前のレベルの CICS のもとでは、Language Environment 準拠のアセンブラー・メインプログラムのため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 5 のプログラム・チェックによって発生する)。

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 (例えば、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。

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 Language Environment 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 Language Environment 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 Language Environment 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 Language Environment 準拠にする方法について詳しくは、「*Language Environment 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・ドライバーが IGZERRE または ILBOSTP0 のいずれかを使用する場合、ドライバーを変更する必要があります。

OS/VS COBOL ILBOSTP0 または IGZERRE ルーチンを Language Environment CEEPIPI INIT_SUB、CEEPIPI INIT_ MAIN、および CEEPIPI TERM 関数で置き換えます。これらの Language Environment 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。

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、Language Environment 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。

MAIN COBOL プログラムをロードし、そのプログラムに BALR を実行するアセンブラー・プログラム

Enterprise COBOL V5 より前には、アセンブラーから OS/VS COBOL メインプログラムに対して、LOAD および BALR を実行し、さらに BALR を再度実行することができました。ただし、NORENT オプションを指定して Enterprise COBOL (またはそれ以降のコンパイラー) でコンパイルされたメインプログラムに対して、LOAD および BALR を実行し、さらに再度 BALR を実行することはサポートされていません。 Enterprise COBOL で NORENT コンパイラー・オプションを使用して OS/VS COBOL プログラムを再コンパイルすると (上記の BALR を再度実行する場合)、プログラムはメッセージ IGZ0044S を出して異常終了します。次のように、いくつかの解決法が考えられます。

- RENT でコンパイルする。
- NORENT COBOL に後続の BALR を実行する前に、アセンブラー・コードを DELETE および再 LOAD に変更する。
- アセンブラー・プログラムを Language Environment 準拠に変更する。

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

Language Environment では、アセンブラー・プログラムは、以下のプログラムのいずれかを含んでいるプログラム・オブジェクトを SVC ロードし、SVC 削除することができます。

- 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

制約事項: Debug Tool は、アセンブラーが SVC 削除を使用して削除したプログラム・オブジェクトに含まれている COBOL プログラムをサポートしません。

Language Environment では、アセンブラー・プログラムは、以下のプログラムのいずれかを含んでいるプログラム・オブジェクトを 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プログラム・オブジェクトを 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プログラム・オブジェクト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、以下のタスク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。
- Language Environment 提供の CEEFETCH マクロおよび CEERELES マクロを使用する。

アセンブラー・プログラムで汎用レジスターの高位半分を保存および復元する

このトピックには、Enterprise COBOL V5 を呼び出す、または Enterprise COBOL V5 から呼び出されるアセンブラー・プログラムで汎用レジスター (GPR) の高位半分を保存および復元する方法に関する情報が付録に追加されました。

「*MVS プログラミング: アセンブラー・サービス・ガイド*」に記載されている F5SA または F8SA の保存域フォーマットは使用しないでください。

GPR の高位半分は、ユーザー・ストレージのどこにでも保存でき、またどこからでも復元できますが、HGPR(PRESERVE) が有効になっている場合は COBOL で使用されるモデルを選択したい場合があります。この場合、COBOL V5 コンパイラーは常に、レジスターの低位半分を保存するために使用されるものと同じ相対位置にあるストレージのブロックを使用します。GPR の高位半分を保存および復元するために、COBOL V5 で何が行われるかの例を以下に示します。

1. 入り口では以下が行われます。
   a. DSA 内の 72 バイトを予約します (現在オフセット +136 付近にあります)。
   b. STMH R1,R15,136(,R13) を指定します。
2. 出口では、LMH R1,R15,136(,R13) を指定します。

Enterprise COBOL V5 プログラムでのプログラム名およびコンパイル・タイム・スタンプの検出

COBOL V5 プログラムのプログラム名 (および PPA1) を実行時に見つけることができます。

1. 現行レジスター 13 から、逆チェーン・ポインター (R13 + 4) をたどります。
2. エントリー・ポイント・アドレス (EP@) が逆チェーンの R15 スロット (逆チェーン・アドレス + 16) にあります。
3. EP@ で、EP@+12 内のワードを確認します。エントリー・ポイントからこのプログラムの PPA1 へのオフセットを表す整数があります。
4. この整数を EP@ に加算します。これが PPA1 アドレスです。
5. プログラム名は PPA1 にあります。 (PPA1 の先頭バイト x 2 (byte *2) は、PPA1 内のプログラム名のオフセットを示しています。)
6. プログラム名の先頭 2 バイトは名前の長さで、その後に名前が続きます。

付録 E. オプションの比較

次の表で、Enterprise COBOL V5 コンパイラー・オプションおよびインストール・オプションについて記載します。また、これらのオプションが OS/VS COBOL、VS COBOL II、IBM COBOL および Enterprise COBOL V3 と V4 のオプションと比べてどうであるかについて説明します。

Enterprise COBOL V5 の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*」および「*Enterprise COBOL カスタマイズ・ガイド*」を参照してください。

表 41. オプションの比較

	適用対象					
オプション	**OS/VS**	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	**使用上の注意**
ADATA			X	X	X	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。
ADV	X	X	X	X	X	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。
AFP					X	z/Architecture プロセッサーに備わる追加浮動小数点 (AFP) レジスターのコンパイラーでの使用方法を制御します。AFP(VOLATILE) がデフォルトです。
ANALYZE			X**			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。
ALOWCBL		X	X	X	X	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。

表 41. 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
	OS/VS	VS II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	
APOST	X	X	X	X	X	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。
ARCH					X	実行可能プログラム命令の生成対象となるマシン・アーキテクチャーを指定します。ARCH(7) がデフォルトです。
ARITH			X	X	X	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) がデフォルトです。 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 への引数に 18 桁を、FACTORIAL への引数に 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 への引数に 31 桁を、FACTORIAL への引数に 29 桁を指定することができます。
AWO		X	X	X	X	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。
BLOCK0				X	X	ファイル記述に BLOCK CONTAINS も RECORDING MODE U も指定していないプログラム内のすべての順次ファイルに対して BLOCK CONTAINS 0 節をアクティブにします。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
BUF	X					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		X	X	X	X	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			X	X	X	組み込みの CICS (顧客情報管理システム) 変換プログラム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	X					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。
CMPR2		X	X			VS COBOL II リリース 2 またはその他の VS COBOL II CMPR2 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した。 デフォルト動作の NOCMPR2 は変更できません。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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
CODEPAGE				X	X	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。CODEPAGE(1140) がデフォルトです。
COMPILE		X	X	X	X	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。
COPYRIGHT					X	COPYRIGHT は、オブジェクト・モジュールが生成された場合に、オブジェクト・モジュール内にストリングを配置します。オブジェクトがプログラム・オブジェクトにリンクされている場合、ストリングはそのプログラム・オブジェクトとともにメモリーにロードされます。
COUNT	X					プログラム実行の終わりに動詞実行サマリーを作成します。各動詞がプロシージャー名とステートメント番号によって特定され、その使用回数が示されます。 同様の機能が Debug Tool によって提供されます。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
CURRENCY			X	X	X	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)		X	X	X	X	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で獲得されるかを指定します。 DATA(24) では、再入可能プログラム・データは 16MB 境界より下で獲得されます。DATA(31) では、再入可能プログラム・データは 16MB 境界より上で獲得されます。 DATA(31) がデフォルトです。
DATEPROC			X	X		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		X	X	X	X	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		X	X	X	X	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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
DECK	X	X	X	X	X	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			X	X	X	受信側が数値である 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) の診断メッセージを出すようにコンパイラーに指示します。NODIAGTRUNC がデフォルトです。
DISPSIGN					X	符号付き数値項目の DISPLAY の出力形式設定を制御します。DISPSIGN(COMPAT) がデフォルトです。
DLL			X	X	X	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	X					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	X	X	X	X	X	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。
DYNAM	X	X	X	X	X	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、サブプログラムはプログラム・オブジェクト内で静的にリンク・エディットされます。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
EXIT(INEXIT (IN-id))、EXIT(LIBEXIT (LIB-id))、EXIT(PRTEXIT (PRT-id))、EXIT(ADEXIT (ADT-id))、および EXIT(MSGEXIT (MSG-id))		X	X	X	X	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)。 ADEXIT サブオプションは、COBOL for MVS & VM 以降のコンパイラーでのみ使用可能です。 MSGEXIT サブオプションは、Enterprise COBOL V4.2 以降のコンパイラーでのみ使用可能です。 NOEXIT がデフォルトです。
EXPORTALL			X	X	X	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		X	X	X	X	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	X	X	X	X	X	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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
FLAGMIG		X	X	X		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。
FLAGMIG4				X****		Enterprise COBOL バージョン 4 リリース 2 用の APAR PM93450 によって、サポートされていない、または Enterprise COBOL バージョン 5 では異なる意味でサポートされる言語エレメントを Enterprise COBOL バージョン 4 プログラムで特定するため、オプション FLAGMIG4 が追加されます。コンパイラーは、このような言語エレメントのすべてについて警告診断メッセージを生成します。
FLAGSTD		X	X	X	X	85 COBOL 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		X				アプリケーションが異常終了するときに、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 オプションが使用されます。
HGPR					X	z/Architecture プロセッサーに備わる 64 ビット・レジスターのコンパイラーでの使用方法を制御します。HGPR(PRESERVE) がデフォルトです。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
IDLGEN			X			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			X	X	X	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。INTDATE(ANSI) では、85 COBOL 標準の開始日付 (Day 1 = January 1, 1601) が使用されます。INTDATE(LILIAN) では、Language Environment 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		X	X	X	X	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	X	X	X	X		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。
LINECNT=nn	X					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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
LINECOUNT		X	X	X	X	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		X	X	X	X	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	X					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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
MAP		X	X	X	X	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。 最新サービスがインストールされた Enterprise COBOL V5.1、および Enterprise COBOL V5.2 には、コンパイラー・リストの MAP 出力に 16 進または 10 進のどちらのオフセットを表示するかを制御する、新しいサブオプション HEX および DEC が追加されました。 基本レベルの Enterprise COBOL V5.1 は常に 10 進オフセットで MAP 出力を生成しますが、これより古いコンパイラーはすべて 16 進オフセットで MAP 出力を生成します。 サブオプションなしで MAP を指定すると、MAP(HEX) として受け入れられます。これにより、旧 COBOL コンパイラーと同じ動作が Enterprise COBOL で得られます。
MAXPCF(*n*)					X	プログラムに *n* より大きい複雑度の因数が含まれている場合に、コードを最適化しないようコンパイラーに指示します。デフォルトは MAXPCF(60000) です。
MDECK				X	X	ライブラリー処理 (COPY、BASIS、REPLACE、および EXEC SQL INCLUDE ステートメントの拡張) からの出力がファイルへ書き出されるようにします。NOMDECK がデフォルトです。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
NAME	X	X	X	X	X	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				X	X	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。
NUM	X					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		X	X	X	X	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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
NUMCLS		X	X	X	X	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		X	X	X	X	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、S/390® の標準の符号を持つものと見なされます。 **NUMPROC(NOPFD)** コンパイラーは、非優先的であるが有効な符号の、必要なすべての符号変換を行います。 **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。このサブオプションは、Enterprise COBOL V5 ではサポートされません。 NUMPROC(NOPFD) がデフォルトです。
OBJECT		X	X	X	X	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 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
OFFSET		X	X	X	X	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。
OPTFILE				X	X	コンパイラー・オプションが SYSOPTF DD ステートメントによって指定される別個のデータ・セットまたはファイルから読み込まれるように指定します。デフォルトでは、OPTFILE は有効になっていません。
OPTIMIZE	X	X	X	X	X	オブジェクト・プログラムを最適化します。 IBM COBOL、および V5 より前の Enterprise COBOL の場合、OPTIMIZE にはサブオプション (STD/FULL) がありました。デフォルトは NOOPTIMIZE でした。 Enterprise COBOL V5 の場合、OPTIMIZE にはサブオプション (*0 / 1 / 2*) があります。アプリケーション実行時のパフォーマンスを向上させるには、より高い最適化レベルを OPTIMIZE オプションに指定します。 `OPTIMIZE(0)` がデフォルトです。
OUTDD(SYSOUT) OUTDD(ddname)		X	X	X	X	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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
PGMNAME			X	X	X	長さおよび大/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、古いバージョンの COBOL コンパイラーと互換性のある方法で処理されます。 PGMNAME(COMPAT) がデフォルトです。
PMAP	X					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。 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。
QUALIFY					X	QUALIFY は、修飾の規則に作用し、COBOL 標準規則の下で参照できない一部のデータ項目を参照できるように修飾規則を拡張するかどうかを制御します。
QUOTE	X	X	X	X	X	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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
RES	X	X				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。RES はデフォルトの動作で、変更はできません。
RENT		X	X	X	X	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)			X	X	X	生成されるオブジェクト・プログラムの常駐モードを設定します。NORENT を指定してコンパイルされたプログラムには、RMODE(24) が設定されます。RENT を指定してコンパイルされたプログラムには、RMODE(ANY) が設定されます。RMODE(AUTO) がデフォルトです。
SEQ	X					ソース・ステートメントの行番号の昇順を検査します。 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		X	X	X	X	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。
SERVICE					X	SERVICE は、オブジェクト・モジュールが生成された場合に、オブジェクト・モジュール内にストリングを配置します。オブジェクト・モジュールがプログラム・オブジェクトにリンクされている場合、ストリングはこのプログラム・オブジェクトとともにメモリーにロードされます。言語環境プログラム・ダンプにトレースバックが含まれている場合は、このストリングがそのトレースバックに組み込まれます。
SIZE(MAX) SIZE(nnnnn) SIZE(nnnK)		X	X	X	X*****	コンパイルに使用される仮想記憶域を指定します。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
SOURCE	X	X	X	X	X	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	X	X	X	X	X	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			X	X	X	DB2 コプロセッサー能力を使用可能にし、DB2 サブオプションを指定します。NOSQL がデフォルトです。
SQLIMS					X	IMS SQL coprocessor 能力を使用可能にし、IMS サブオプションを指定します。NOSQLIMS がデフォルトです。
SQLSSCID				X	X	CODEPAGE コンパイラー・オプションが COBOL プログラム内の SQL ステートメントの処理に影響するかどうかを決定します。効力をもつのは、統合された DB2 コプロセッサー (SQL コンパイラー・オプション) を使用する場合のみです。NOSQLCCSID がデフォルトです。
SSRANGE		X	X	X	X	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。
STGOPT					X	ストレージ最適化を制御します。NOSTGOPT がデフォルトです。
SYSx	X					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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
STATE	X					アプリケーションが異常終了するときに、デバッグ情報を含んでいるダンプを生成します。 IBM Enterprise COBOL では、TEST オプションが、OS/VS COBOL の STATE オプションの代わりに使用されます。
SUPMAP SYNTAX CSYNTAX	X					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。
SYMDMP	X					シンボリック・ダンプを生成します。 異常終了ダンプと動的ダンプは、Language Environment 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・オプションによって入手できます。
SXREF	X					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。 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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
TERM	X					SYSTERM データ・セットに進行メッセージを送ります。 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。
TERMINAL		X	X	X	X	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	X	X	X	X	X	プロダクト用の Debug Tool で使用できるオブジェクト・コードを生成します。NOTEST(NODWARF) 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				X	X	複数の POSIX スレッドまたは PL/I タスクを持つ 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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
TRUNC	X	X	X	X	X	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 進数受信フィールドの 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。 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。
TYPECHK			X			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。 NOTYPECHK がデフォルトです。
VBREF VBSUM	X	X	X	X	X	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。 NOVBREF がデフォルトです。
VLR					X******	返されたレコード長がレコード記述と矛盾する場合に、可変長レコードの READ ステートメントから返されるファイル状況に作用します。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	
WORD		X	X	X	X	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。
XMLPARSE				X***	X******	Enterprise COBOL バージョン 4 以降のみ (サービスを介した Enterprise COBOL バージョン 5.1 で使用可能)。使用する XML パーサーとして、z/OS XML システム・サービス・パーサー (XMLSS) または Enterprise COBOL バージョン 3 で使用されていた COBOL 高速パーサーのいずれかを選択します。
XREF		X	X	X	X	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			X	X		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。YEARWINDOW(1900) がデフォルトです。
ZONEDATA					X	USAGE DISPLAY 数値比較に有効なゾーン・ビットを持つデータが数値データ項目 (ゾーン 10 進数) に含まれているかどうかをコンパイラーに指示します。
ZWB	X	X	X	X	X	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。

表 41. 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
	OS/VS	**VS II**	**IBM COBOL**	**Enterprise COBOLV3 と V4**	**Enterprise COBOL V5**	

- X* COBOL (OS/390 および VM 版) バージョン 2 リリース 2 でのみ使用可能です。
- X** COBOL (OS/390 および VM 版) バージョン 2 リリース 1 および 2 でのみ使用可能です。
- X*** Enterprise COBOL バージョン 4 リリース 1 および 2 でのみ使用可能です。
- X**** Enterprise COBOL バージョン 4 リリース 2 でのみ使用可能です。
- X***** SIZE(MAX) は Enterprise COBOL バージョン 5 リリース 1 ではサポートされません。 SIZE オプションは Enterprise COBOL バージョン 5 リリース 2 ではサポートされません。
- X****** Enterprise COBOL バージョン 5 リリース 2 でのみ使用可能です。

付録 F. コンパイラー限界値の比較

以下の表は、Enterprise COBOL V5、その他の Enterprise COBOL バージョン、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をリストしたものです。

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 V5	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
プログラムのサイズ	999,999 行	999,999 行	999,999 行	999,999 行
リテラルの数	4,194,303-B[1]	4,194,303-B[1]	4,194,303-B[1]	16,384-B
リテラルの全長	4,194,303-B[1]	4,194,303-B[1]	4,194,303-B[1]	OPT 後に 32,767-B
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	32,760-B	32,767-B	32,767-B	16,384-B
見出し部				
環境部				
構成セクション				
SPECIAL-NAMES 段落				
mnemonic-name IS	18	18	18	18
UPSI-*n* . . . (スイッチ)	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 . . . or ALSO . . .	256	256	256	256
INPUT-OUPUT SECTION				
FILE-CONTROL 段落				

言語エレメント	Enterprise COBOL V5	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
SELECT *ファイル名* . . .	最大 65,535 個のファイル名に外部名を割り当て可能	最大 65,535 個のファイル名に外部名を割り当て可能	最大 65,535 個のファイル名に外部名を割り当て可能	最大 65,535 個のファイル名に外部名を割り当て可能
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	32,767
RERUN *整数* RECORDS	16,777,215	16,777,215	16,777,215	16,777,215
SAME RECORD AREA	255	255	255	255
SAME RECORD AREA 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				
77 データ項目のサイズ	999,999,999	134,217,727	16,777,215	1,048,576
01 + 77 の合計 (データ項目)	N/L	N/L	N/L	255
88 条件名 . . .	N/L	N/L	N/L	N/L
66 RENAMES . . .	N/L	N/L	N/L	N/L
PICTURE 節、文字ストリング中の文字数	50	50	30	30
PICTURE 節、数値項目の桁位置	ARITH(COMPAT) を使用: 18 ARITH(EXTEND) を使用: 31	18 (または 31) [6]	IBM COBOL の場合: 18 (または 31) [6] VS COBOL II の場合: 18	18
PICTURE 節、数値編集項目の文字位置	249	249	249	127
PICTURE 記号複製 ( )	999,999,999	134,217,727	16,777,215	99,999
PICTURE 記号複製 ( )、クラス DBCS 項目	499,999,999	67,108,863	8,388,607	N/A
PICTURE 記号複製 ( )、クラス国別項目	499,999,999	67,108,863	N/A	N/A
PICTURE 記号複製 (編集)	32,767	32,767	32,767	99,999
基本項目のサイズ	134,217,727	134,217,727	16,777,215	32,767
OCCURS 整数	999,999,999	134,217,727	4,194,303	65,535

言語エレメント	Enterprise COBOL V5	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
テーブルのサイズ	999,999,999	134,217,727	8,388,607	32,767
ASC または DES KEY . . . (OCCURS 文節当たり)	12	12	12	12
キーの合計長さ (OCCURS 節当たり)	256B	256B	256B	256B
INDEXED BY . . . (OCCURS 節による指標名)	12	12	12	12
クラスまたはプログラム当たりの指標 (指標名) の総数	65,535	65,535	65,535	65,535
相対指標のサイズ	32,765	32,765	32,765	32,765
FILE SECTION				
FD レコード記述項目	1,048,575	1,048,575	1,048,575	1,048,575
FD *ファイル名* . . .	65,535	65,535	65,535	65,535
LABEL データ名 . . . (オプション文節がない場合)	255	255	255	185
ラベル・レコードの長さ	80-B	80-B	80-B	80-B
DATA RECORD データ名 . . .	N/L[4]	N/L[4]	N/L[4]	N/L[4]
BLOCK CONTAINS 整数	2,147,483,647[9]	2,147,483,647[9]	IBM COBOL の場合: 2,147,483,647 VS COBOL II 用: 1,048,575[5]	32,760
RECORD CONTAINS 整数	1,048,575[5]	1,048,575[5]	1,048,575[5]	32760
SD *ファイル名* . . .	65,535	65,535	65,535	65,535
DATA RECORD データ名 . . .	N/L[4]	N/L[4]	N/L[4]	N/L[4]
WORKING-STORAGE SECTION				
EXTERNAL 属性のない項目の合計サイズ	2,147,483,646-B	134,217,727-B	134,217,727-B	1,048,576
EXTERNAL 属性のある項目の合計サイズ	2,147,483,646-B	134,217,727-B	134,217,727-B	N/A
LINKAGE SECTION				
合計サイズ	N/L	134,213,631-B	134,217,727-B	1,048,576
手続き部				
プロシージャーおよび定数域	4,194,303[1]	4,194,303[1]	4,194,303[1]	1M+32-KB
PROCEDURE DIVISION USING *識別子* . . .	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 (FDUMP/TEST)	7	7	7	7
動詞当たりの添え字付けされたデータ名の数	32,767	32,767	32,767	511
ADD *識別子* . . .	N/L	N/L	N/L	N/L

言語エレメント	Enterprise COBOL V5	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
ALTER *プロシージャー名 1* TO *プロシージャー名 2* . . .	4,194,303[1]	4,194,303[1]	4,194,303[1]	64-KB[1]
CALL . . . BY CONTENT *識別子*	2,147,483,647	2,147,483,647	2,147,483,647	N/A
CALL *リテラル* . . .	4,194,303[1]	4,194,303[1]	4,194,303[1]	N/L
CALL *識別子またはリテラル* USING *識別子またはリテラル* . . .	16,380	16,380	16,380	N/L
実行単位内のアクティブ・プログラム数	32,767	32,767	32,767	32,767
呼び出される名前の数 (DYN オプション)	N/L	N/L	N/L	64-K
CANCEL *識別子またはリテラル* . . .	N/L	N/L	N/L	N/L
CLOSE *ファイル名* . . .	N/L	N/L	N/L	N/L
COMPUTE *識別子* . . .	N/L	N/L	N/L	N/L
DISPLAY *識別子またはリテラル* . . .	N/L	N/L	N/L	N/L
DIVIDE *識別子* . . .	N/L	N/L	N/L	N/L
ENTRY USING *識別子またはリテラル* . . .	N/L	N/L	N/L	N/L
EVALUATE . . . サブジェクト	64	64	64	N/L
EVALUATE . . . WHEN 節	256	256	256	N/L
GO *プロシージャー名* . . . DEPENDING	255	255	255	2031
INSPECT TALLYING および REPLACING 節	N/L	N/L	N/L	15
MERGE *ファイル名* ASC または DES KEY . . .	N/L	N/L	N/L	12
マージ・キーの合計長	4092-B[7]	4092-B[7]	4092-B[7]	256-B
MERGE USING *ファイル名* . . .	16[8]	16[8]	16[8]	16[8]
MOVE *識別子またはリテラル* TO *リテラル* . . .	N/L	N/L	N/L	N/L
MULTIPLY *識別子* . . .	N/L	N/L	N/L	N/L
OPEN *ファイル名* . . .	N/L	N/L	N/L	N/L
PERFORM	4,194,303	4,194,303	4,194,303	64-K
SEARCH . . . WHEN . . .	N/L	N/L	N/L	N/L
SET *指標または識別子* . . . TO	N/L	N/L	N/L	N/L
SET *指標* . . . UP または DOWN	N/L	N/L	N/L	N/L
SORT *ファイル名* ASC または DES KEY	N/L	N/L	N/L	12
ソート・キー合計長	4092-B[7]	4092-B[7]	4092-B[7]	256-B
SORT USING *ファイル名* . . .	16[8]	16[8]	16[8]	16[8]
STRING *識別子* . . .	N/L	N/L	N/L	N/L

言語エレメント	Enterprise COBOL V5	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
STRING DELIMITED *識別子*またはリテラル . . .	N/L	N/L	N/L	N/L
UNSTRING DELIMITED *識別子*またはリテラル . . .	N/L	255	255	15
UNSTRING INTO *識別子*またはリテラル . . .	N/L	N/L	N/L	N/L
USE . . . ON *ファイル名* . . .	N/L	N/L	N/L	N/L

1. プロシージャー + 定数域について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. 構文検査は行われますが、プログラム実行への影響はありません。無制限です。
5. 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。
6. COBOL (OS/390 および VM 版) V2R2 以降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。
7. QSAM および VSAM の場合、OPTION 制御ステートメントに EQUALS がコーディングされていれば限界値は 4088 バイトです。
8. QSAM および VSAM の SORT 限度。
9. OS/390 DFSMS バージョン 2 リリース 10.0 以降で提供されるラージ・ブロック・インターフェース (LBI) サポートが必要です。それより前のリリースの DFSMS を使用する OS/390 システムの場合、限度は 32,767 バイトです。ラージ・ブロック・サイズの使用について詳しくは、「*Enterprise COBOL プログラミング・ガイド*」を参照してください。

付録 G. QSAM ファイルでのファイル状況 39 の防止

QSAM ファイルでのファイル状況 39 を防止するには、プログラム内のファイルの記述と、データ・セットに対して定義された属性との間に不一致がないようにします。

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。例えば、次のように指定します。

ファイル形式	要件
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。
形式 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。
形式 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。

要確認: JCL の情報は、データ・セット・ラベル内の情報をオーバーライドします。

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 COBOL 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。例えば、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 6. 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 (例えば、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。

例えば、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 VARYING 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

注: このトピックは、QSAM ファイルのみを対象にしています。
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 H. バインダー (リンケージ・エディター) デフォルトのオーバーライド

AMODE と RMODE

RENT コンパイラー・オプションおよび RMODE コンパイラー・オプションの設定によっては、AMODE および RMODE の Enterprise COBOL デフォルト設定をオーバーライドしなければならない場合があります。

コンパイラーによって割り当てられている AMODE や RMODE はオーバーライドしないでください。特に、以下のことは行わないでください。

- Enterprise COBOL バージョン 5 の NORENT プログラムの RMODE を RMODE ANY に変更しないでください。
- Enterprise COBOL バージョン 5.1.0 プログラムの AMODE を AMODE 24 に変更しないでください。Enterprise COBOL バージョン 5.1.1 およびバージョン 5.2.0 のプログラムの AMODE は AMODE 24 に変更できます。

プログラム・オブジェクトがバインド後に AMODE 24 を割り当てられた場合、このオブジェクトの RMODE も RMODE 24 になります。バインダー・オプション RMODE(ANY) を指定することはできません。

RENT

RENT コンパイラー・オプションを使用してコンパイルする場合は、モジュールが REUS=RENT または代替 RENT オプションが指定された RENT であることをバインダーに通知する必要があります (RENT には REUS が含まれるため、REUS は不要)。属性 RENT は AMODE や RMODE とは異なり、コンパイラーによってプログラム・オブジェクトに設定されることはありません。

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、以下の説明にあるいずれかの手段を使用して AMODE または RMODE を指定します。

- リンク・エディット・ジョブ・ステップの EXEC ステートメント。
  programname は、IEWBLINK、IEWL、HEWL などのプログラム・バインダー (リンケージ・エディター) を指します。

```
//LKED     EXEC     PGM=programname,
//          PARM='AMODE=xx,RMODE=yy'
```

- リンケージ・エディター MODE 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)
```

許容可能な *xx* および *yy* ならびにバインダー MODE 制御ステートメントについて詳しくは、「*MVS プログラム管理: ユーザーズ・ガイドおよび解説書*」を参照してください。

バインダーによって設定された情報を使用するローダーは、ATTACH、LINK、XCTL、または LOAD/BASSM を使用して呼び出されたプログラムが 24 ビットと 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、プログラムの AMODE 属性を使用して決定します。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 I. 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があります。

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 Language Environment 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存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 J. z/OS UNIX に関する考慮事項

COBOL コンパイラーを z/OS UNIX 下で、またはコンパイラー入出力用の zFS ファイルとともに使用していて、z/OS V1.13 から z/OS V2.1 以降に移行中である場合は、z/OS の変更点が COBOL コンパイラーの使用に影響を与える可能性があります。

z/OS V2R1 以降、デフォルト OMVS セグメントは z/OS UNIX ではサポートされなくなりました。

z/OS V2R1 では、デフォルト OMVS セグメントは使用できなくなりました。すべての z/OS UNIX ユーザーまたはグループは、固有のユーザー ID (UID) およびグループ ID (GID) を使用して、ユーザー・プロファイルおよびグループ・プロファイルに OMVS セグメントを定義する必要があります。 1 つの解決策は、RACF® サポートを使用して、OMVS セグメントが定義されていないユーザーおよびグループの固有 UID および GID を要求時に自動的に生成することです。固有 UID および GID の自動生成サポートは、z/OS V1R11 以降に使用可能になっています。

cob2 インターフェースを介して z/OS UNIX 上のコンパイラーを呼び出す Enterprise COBOL ユーザーはすべて、影響を受ける可能性があります。バッチ・モードでコンパイラーを呼び出すユーザーも、入出力ファイルが z/OS UNIX ファイル・システム上にある場合は影響を受ける可能性があります。詳しくは、「z/OS V2R1 Migration guide」を参照してください。

次の例は、OMVS リソースへのアクセス試行時に OMVS セグメントを定義していない場合に発生するエラーを示しています。

```
16.26.26 JOB00126  $HASP373 T1F11010 STARTED - INIT 8    - CLASS A        - SYS
16.26.27 JOB00126  SMF000I  T1F11010    PC          DSNHPC      0004
16.26.27 JOB00126  ICH408I USER(USRT011 ) GROUP(SYS1    ) NAME(#################
   465               CL(PROCESS )
   465               OMVS SEGMENT NOT DEFINED
16.26.27 JOB00126  +CEE3798I ATTEMPTING TO TAKE A DUMP FOR ABEND U4093 TO DATA S
```

付録 K. JCL パラメーターへのアクセス

EXEC ステートメントの PARM= キーワードを使用すれば、パラメーター・ストリングを JCL から COBOL プログラムに渡すことができます。このパラメーターには、標準 COBOL コーディングによって、または CEE3PR2 Language Environment 呼び出し可能サービスを呼び出すことによって、アクセスできます。

COBOL コーディングの使用

PARM ストリングによって渡される *user_parameter* データを受け取る LINKAGE SECTION レコード (レベル 01) を、システムでそのストリングの前に挿入されるハーフワード長フィールドを考慮しながら、定義する必要があります。

プログラムは、このフィールド長がゼロではないことをテストし、PARM ストリング・データが実際に渡されていることを検証できます。以下に例を示します。

```
LINKAGE SECTION.
01 PARMDATA.
    05 STRINGLEN PIC 9(4) USAGE COMP.
    05 STRINGPARM PIC X(80).
PROCEDURE DIVISION USING PARMDATA.
    IF STRINGLEN > 0 . . .
```

詳しくは、「*Enterprise COBOL プログラミング・ガイド*」 の 『LINKAGE SECTION のコーディング』を参照してください。

CEE3PR2 の使用法

CEE3PR2 呼び出し可能サービスへのパラメーターを定義する必要があります。その際、パラメーターを PROCEDURE DIVISION USING ステートメントに追加する必要はありません。

```
WORKING-STORAGE SECTION.
01 PARMLEN PIC S9(9) BINARY.
01 PARMSTR.
   02 STR1-LENGTH PIC S9(4) BINARY.
   02 STR1-STRING.
      03 STR1-CHAR PIC X
                   OCCURS 0 TO 256 TIMES
                   DEPENDING ON STR1-LENGTH.
. . .
   CALL "CEE3PR2" USING PARMLEN,PARMSTR, FC.
```

CEE3PR2 呼び出し可能サービスについて詳しくは、「*Language Environment Programming Reference*」の『CEE3PR2』を参照してください。

付録 L. XMLPARSE(COMPAT) から XMLPARSE(XMLSS) へのマイグレーション

XMLPARSE 設定の XMLSS と COMPAT の相違点を理解した後で、ご使用のプログラムを、XMLPARSE(XMLSS) を使用するように移行することができます。これらの違いは、XMLPARSE(XMLSS) が有効になった時点で、新規、変更、無変更、および中止イベントとして説明されます。

- **ATTRIBUTE-CHARACTER イベント (中止)**
  - **XMLSS:** ATTRIBUTE-CHARACTER イベントはなくなりました。定義済みのものを含めてすべてのエンティティー参照は、未解決のエンティティー参照がない限り ATTRIBUTE-CHARACTERS イベントに含まれるようになりました。未解決のものがある場合は、例外イベントがシグナル通知されます。
  - **COMPAT:** ATTRIBUTE-CHARACTER イベントは、定義済みエンティティー参照に対してのみ発生します。5 つの定義済みエンティティー参照を、 332 ページの表 42 に示します。 XML-TEXT または XML-NTEXT には、属性値の定義済みエンティティー参照に対応する単一文字が含まれます。文字参照は、ATTRIBUTE-NATIONAL-CHARACTER イベントとしてシグナル通知されます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** ATTRIBUTE-CHARACTER イベントへの参照を削除し、このイベントに対するアクションを ATTRIBUTE-CHARACTERS イベント処理に統合します。
- **ATTRIBUTE-CHARACTERS イベント (変更)**
  - **XMLSS:** XML-TEXT または XML-NTEXT は、ATTRIBUTE-CHARACTERS イベントに対する値のサブストリングを持っていることがあります。XML-TEXT または XML-NTEXT も、値に文字参照またはエンティティー参照が含まれる場合でもその値の完全なストリングを含むことがあります。
  - **COMPAT** XML-TEXT または XML-NTEXT は、値に文字参照またはエンティティー参照が含まれている場合は、ATTRIBUTE-CHARACTERS イベントに対する値のサブストリングだけを持ちます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** 属性値に文字参照またはエンティティー参照が含まれない場合でも複数のイベントを処理するように、ATTRIBUTE-CHARACTERS イベントを処理するコードを変更しなければなりません。 コードが ATTRIBUTE-CHARACTERS をマルチ・イベントとして処理していた場合、ATTRIBUTE-CHARACTERS を単一イベントとして処理するようにコードを変更しなければなりません。
- **ATTRIBUTE-NAME イベント (変更)**
  - **XMLSS:** 名前空間にない属性名については、XML-TEXT または XML-NTEXT は属性名を含み、名前空間特殊レジスターはすべて空で長さはゼロです。 名前空間にある名前を持つ属性は、常に接頭部があり、以下の形式です。

    ```
    prefix:local-part = AttValue
    ```

XML-TEXT または XML-NTEXT にはローカル・パートが含まれ、XML-NAMESPACE または XML-NNAMESPACE には名前空間が含まれ、XML-NAMESPACE-PREFIX または XML-NNAMESPACE-PREFIX には接頭部が含まれます。

- **COMPAT:** 属性名に接頭部がある場合 (名前が名前空間に属することを示す) でも、すべての属性名について、 XML-TEXT または XML-NTEXT には完全な属性名が含まれます。
- **XMLPARSE(XMLSS) へのマイグレーション:** 名前空間の各部分を処理するようにコードを変更するか、 XML-TEXT、XML-NAMESPACE-PREFIX、および XML-NAMESPACE、あるいは、 XML-NTEXT、XML-NNAMESPACE-PREFIX、および XML-NNAMESPACE の各部分から完全な属性名を再構成するようにコードを変更します。

- **ATTRIBUTE-NATIONAL-CHARACTER イベント (変更)**
  - **XMLSS:** XML 文書の EBCDIC エンコード方式で表すことのできる文字参照は解決され、ATTRIBUTE-CHARACTERS イベントに組み込まれます。

    表すことのできない文字参照は、COMPAT の場合のように、ATTRIBUTE-NATIONAL-CHARACTER イベントとして表現されます。
  - **COMPAT:** XML PARSE ステートメント内の identifier-1 が指定する XML 文書のタイプに関わらず、XML-TEXT は空で、XML-NTEXT には (数字) 文字参照に対応する単一国別文字が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** おそらく変更は必要ありませんが、COMPAT の場合は国別文字に相当する EBCDIC が存在している可能性があり、それに反して XMLSS の場合は、文書の EBCDIC エンコード方式には国別文字に対応する表現がないことが分かっていることに注意してください。
- **COMMENT イベント (変更)**
  - **XMLSS:** XML-TEXT または XML-NTEXT は、COMMENT イベントに対する値のサブストリングを持っていることがあります。
  - **COMPAT** XML-TEXT または XML-NTEXT は、COMMENT イベントに対する値の完全なストリングを常に持っています。
  - **XMLPARSE(XMLSS) へのマイグレーション:** XML-TEXT または XML-NTEXT の COMMENT 値のサブストリングを受け取った場合でも複数のイベントを処理するように、COMMENT イベントを処理するコードを変更しなければなりません。 その場合、2 つ以上の COMMENT イベントを連続して受け取り、ストリングを連結して値の完全なストリングを再作成します。このように 1 つのコメントが分割されたものを、それぞれ別の複数のコメントからなるシーケンスと区別することはできません。
- **CONTENT-CHARACTER イベント (中止)**
  - **XMLSS:** CONTENT-CHARACTER イベントはなくなりました。事前定義済みのものを含めてすべてのエンティティー参照は、未解決のエンティティー参照がない限り CONTENT-CHARACTERS イベントに組み込まれるようになりました。未解決のものがある場合は、UNRESOLVED-REFERENCE イベントまたは EXCEPTION イベントがシグナル通知されます。
  - **COMPAT:** CONTENT-CHARACTER イベントは、定義済みエンティティー参照に対してのみ発生します。5 つの定義済みエンティティー参照を、 332 ページの表 42 に示します。 XML-TEXT または XML-NTEXT には、属性値の定

義済みエンティティー参照に対応する単一文字が含まれます。文字参照は、CONTENT-NATIONAL-CHARACTER イベントとしてシグナル通知されます。

  - **XMLPARSE(XMLSS) へのマイグレーション:** CONTENT-CHARACTER イベントへの参照を削除し、このイベントに対するアクションを CONTENT-CHARACTERS イベント処理に統合します。

- **CONTENT-CHARACTERS イベント (変更)**
  - **XMLSS:** XML-TEXT または XML-NTEXT は、CONTENT-CHARACTERS イベントに対する内容のサブストリングを持っていることがあります。XML-TEXT または XML-NTEXT も、内容に文字参照またはエンティティー参照が含まれる場合でもその内容の完全なストリングを含むことがあります。
  - **COMPAT** XML-TEXT または XML-NTEXT は、内容に文字参照またはエンティティー参照が含まれている場合は、CONTENT-CHARACTERS イベントに対する内容のサブストリングだけを持ちます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** 属性値に文字参照またはエンティティー参照が含まれない場合でも複数のイベントを処理するように、CONTENT-CHARACTERS イベントを処理するコードを変更しなければなりません。 コードが CONTENT-CHARACTERS をマルチ・イベントとして処理していた場合、CONTENT-CHARACTERS を単一イベントとして処理するようにコードを変更しなければなりません。
- **CONTENT-NATIONAL-CHARACTER イベント (変更)**
  - **XMLSS:** XML 文書の EBCDIC エンコード方式で表すことのできる文字参照は解決され、CONTENT-CHARACTERS イベントに組み込まれます。

    表すことのできない文字参照は、COMPAT の場合のように、CONTENT-NATIONAL-CHARACTER イベントとして表現されます。
  - **COMPAT:** XML PARSE ステートメント内の identifier-1 が指定する XML 文書のタイプに関わらず、XML-TEXT は空で、XML-NTEXT には (数字) 文字参照に対応する単一国別文字が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** おそらく変更は必要ありませんが、COMPAT の場合は国別文字に相当する EBCDIC が存在している可能性があり、それに反して XMLSS の場合は、文書の EBCDIC エンコード方式には国別文字に対応する表現がないことが分かっていることに注意してください。
- **DOCUMENT-TYPE-DECLARATION イベント (変更)**
  - **XMLSS:** XML-TEXT または XML-NTEXT には、文書タイプ宣言に指定されるように、ルート・エレメントの名前が含まれます。パーサーは内部 DTD サブセット内のエンティティー宣言とデフォルト属性値を処理し、文書タイプ宣言内の残りのテキストは無視します。
  - **COMPAT:** XML-TEXT または XML-NTEXT には、文書タイプ宣言全体が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** 文書タイプ宣言全体を保持することが重要な場合、XML 文書から情報を直接取得するように DOCUMENT-TYPE-DECLARATION イベントを処理するコードを変更しなければなりません。
- **ENCODING-DECLARATION イベント (変更)**

- **XMLSS:** XML-TEXT または XML-NTEXT には、エンコード方式の名前が含まれます。 パーサーはエンコード宣言を使用しないため、不正な文字が受け渡されることがあり、その場合パーサーは EXCEPTION イベントをシグナル通知しますが、これはリカバリー不能です。
- **COMPAT:** XML-TEXT または XML-NTEXT には、エンコード方式の名前が含まれます。 文書のエンコード方式にエラーがあると、EXCEPTION イベントを受け取ります。リカバリーして継続できる場合があります。
- **XMLPARSE(XMLSS) へのマイグレーション:** 構文解析の前に資料をチェックするか、または、CODEPAGE コンパイラー・オプションを使用するか、XML PARSE ステートメントで WITH ENCODING 句を使用して、エンコード方式を指定してください。

• END-OF-CDATA-SECTION イベント (変更)

- **XMLSS:** XML-EVENT、XML-CODE および XML-INFORMATION を除くすべての XML 特殊レジスターは空で、長さはゼロです。
- **COMPAT:** XML-TEXT または XML-NTEXT には、常にストリング "]]>"が含まれます。
- **XMLPARSE(XMLSS) へのマイグレーション:** END-OF-CDATA-SECTION イベントからストリング "]]>" が獲得される場合、値 "]]>" で初期設定されたデータ項目またはリテラルを使用して手動でこのストリングを戻すようにコードを変更してください。

• END-OF-DOCUMENT イベント (無変更)

- 2 つのパーサーは、END-OF-DOCUMENT イベントに対して同じ振る舞いをします。
- **XMLPARSE(XMLSS) へのマイグレーション:** 変更の必要はありません。

• END-OF-ELEMENT イベント (変更)

- **XMLSS:** XML-TEXT または XML-NTEXT には、エンド・エレメント・タグまたは空のエレメント・タグ名のローカル・パーツが含まれます。エレメント名が名前空間にある場合、XML-NAMESPACE または XML-NNAMESPACE には名前空間が含まれます。そうでない場合、これらの特殊レジスターは空で長さはゼロです。エレメント名が名前空間にあり、接頭部 ("prefix:local-part" の形式) がある場合、XML-NAMESPACE-PREFIX または XML-NNAMESPACE-PREFIX には接頭部が含まれます。そうでない場合、これらの特殊レジスターは空で長さはゼロです。
- **COMPAT** XML-TEXT または XML-NTEXT には、接頭部を含む完全なエレメント・タグ名が含まれます。エレメント名が名前空間にない場合、END-OF-ELEMENT に対する COMPAT と XMLSS に違いはありません。
- **XMLPARSE(XMLSS) へのマイグレーション:** エレメント名が名前空間にない場合、変更の必要はありません。エレメント名が名前空間にある場合は、完全なエレメント名を使用しないようにコードを変更するか、 XML テキスト中の各部分および名前空間特殊レジスターから完全なエレメント名を再構成するようにコードを変更してください。

• END-OF-INPUT イベント (新規)

- **XMLSS:** END-OF-INPUT イベントは、XML 文書のセグメントの終わりを示します。
- **COMPAT:** END-OF-INPUT イベントはなくなりました。

- **XMLPARSE(XMLSS): への移行** COMPAT では、文書は 1 セグメント内にあるため、XMLSS への移行には変更は不要です。

- **EXCEPTION イベント (変更)**
  - **XMLSS:** XML-CODE には、例外を特定する固有の戻りコードおよび理由コードが含まれます。 XML-CODE の違いの説明については、以下のセクション「その他の違い」を参照してください。 XML-TEXT または XML-NTEXT には、EXCEPTION イベントを発生させたエラーまたは異常の発生時点までの文書のフラグメントが含まれます。XML-EVENT および XML-INFORMATION 以外のすべての XML 特殊レジスターは空で、長さはゼロです。 EXCEPTION イベントからの継続はできません。
  - **COMPAT** XML-TEXT または XML-NTEXT には、EXCEPTION イベント発生時点までに解析された文書全体が含まれます。継続可能な EXCEPTION イベントもあります。
  - **XMLPARSE(XMLSS) へのマイグレーション:** コードまたは文書が、EXCEPTION イベントからリカバリーできるかどうかに依存する場合は、変更しなければなりません。

- **NAMESPACE-DECLARATION イベント (新規)**
  - **XMLSS:** XML-TEXT および XML-NTEXT はどちらも空で長さはゼロです。XML-NAMESPACE または XML-NNAMESPACE には、宣言された名前空間が含まれます。空ストリングを指定することによって、名前空間が「宣言されていない」場合は、 XML-NAMESPACE および XML-NNAMESPACE は空で長さがゼロです。名前空間の属性名が "xmlns:prefix" の形式の場合は、XML-NAMESPACE-PREFIX または XML-NNAMESPACE-PREFIX には、接頭部が含まれます。そうでない場合は、宣言がデフォルトの名前空間のものであり属性名が "xmlns" であれば、XML-NAMESPACE-PREFIX および XML-NNAMESPACE-PREFIX はいずれも空で長さがゼロです。
  - **COMPAT:** NAMESPACE-DECLARATION イベントはなくなりました。
  - **XMLPARSE(XMLSS) へのマイグレーション:** XMLSS へのマイグレーション後に NAMESPACE-DECLARATION イベントを受け取る場合は、この表の ATTRIBUTE-NAME、END-OF-ELEMENT、および START-OF-ELEMENT イベントの変更点を参照してください。

- **PROCESSING-INSTRUCTION-DATA イベント (変更)**
  - **XMLSS:** XML-TEXT または XML-NTEXT は、PROCESSING-INSTRUCTION-DATA イベントに対する値のサブストリングを持っていることがあります。
  - **COMPAT** XML-TEXT または XML-NTEXT は、PROCESSING-INSTRUCTION-DATA イベントに対する値の完全なストリングを常に持っています。
  - **XMLPARSE(XMLSS) へのマイグレーション:** XML-TEXT または XML-NTEXT の PROCESSING-INSTRUCTION-DATA 値のサブストリングを受け取った場合でも複数のイベントを処理するように、PROCESSING-INSTRUCTION-DATA イベントを処理するコードを変更しなければなりません。 その場合、 2 つ以上の PROCESSING-INSTRUCTION-DATA イベントを受け取り、それぞれのイベントには、対応する PROCESSING-INSTRUCTION-

TARGET イベントが先行します。それらの PROCESSING-INSTRUCTION-DATA サブストリングを連結して、完全なデータ・ストリングを再構築できます。

- **PROCESSING-INSTRUCTION-TARGET イベント (変更)**
  - **XMLSS:** 処理命令データがサブストリングに分割されている場合、処理命令の PROCESSING-INSTRUCTION-DATA イベントの各インスタンスの前に PROCESSING-INSTRUCTION-TARGET イベントが繰り返されます。
  - **COMPAT:** PROCESSING-INSTRUCTION-TARGET イベントは、1 つの処理命令について 1 回だけ発生します。
  - **XMLPARSE(XMLSS) への移行:** 処理命令データを累積中に PROCESSING-INSTRUCTION-TARGET イベントの複数のオカレンスに対処するようにコードを変更しなければならない場合があります。
- **STANDALONE-DECLARATION イベント (無変更)**
  - XMLSS および COMPAT は、STANDALONE-DECLARATION イベントに対して同じ振る舞いをします。
  - **XMLPARSE(XMLSS) へのマイグレーション:** 変更の必要はありません。
- **START-OF-CDATA-SECTION イベント (変更)**
  - **XMLSS:** XML-EVENT、XML-CODE および XML-INFORMATION を除くすべての XML 特殊レジスターは空で、長さはゼロです。
  - **COMPAT:** XML-TEXT または XML-NTEXT には、常にストリング "![CDATA["が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** START-OF-CDATA-SECTION イベントからストリング "![CDATA[" が獲得される場合、値 "![CDATA[" で初期設定されたデータ項目またはリテラルを使用して手動でこのストリングを戻すようにコードを変更してください。
- **START-OF-DOCUMENT イベント (変更)**
  - **XMLSS:** XML-EVENT、XML-CODE および XML-INFORMATION を除くすべての XML 特殊レジスターは空で、長さはゼロです。
  - **COMPAT:** XML-TEXT または XML-NTEXT には、文書全体が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** START-OF-DOCUMENT に対して文書全体が必要でないようにコードを変更してください。
- **START-OF-ELEMENT イベント (変更)**
  - **XMLSS:** XML-TEXT または XML-NTEXT には、スタート・エレメント名または空のエレメント名のローカル・パーツが含まれます。エレメント名が名前空間にある場合、XML-NAMESPACE または XML-NNAMESPACE には名前空間が含まれます。そうでない場合、これらの特殊レジスターは空で長さはゼロです。エレメント名が名前空間にあり、接頭部 ("prefix:local-part" の形式) がある場合、XML-NAMESPACE-PREFIX または XML-NNAMESPACE-PREFIX には接頭部が含まれます。そうでない場合、これらの特殊レジスターは空で長さはゼロです。
  - **COMPAT** XML-TEXT または XML-NTEXT には、接頭部を含む完全なスタート・エレメント名が含まれます。エレメント名が名前空間にない場合、START-OF-ELEMENT に対する COMPAT と XMLSS に違いはありません。

- **XMLPARSE(XMLSS) へのマイグレーション:** エレメント名が名前空間にない場合、変更の必要はありません。エレメント名が名前空間にある場合は、完全なエレメント名を使用しないようにコードを変更するか、 XML テキスト中の各部分および名前空間特殊レジスターから完全なエレメント名を再構成するようにコードを変更してください。

- **UNKNOWN-REFERENCE-IN-ATTRIBUTE イベント (中止)**
  - **XMLSS** なくなりました。パーサーは、属性値の処理中に、定義されていないエンティティーへの参照を検出すると、常に EXCEPTION イベントをシグナル通知します。
  - **COMPAT** XML-TEXT または XML-NTEXT には、"&" および ";" 区切り文字を含まない、エンティティー参照名が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** 属性値内の未定義のエンティティー参照が XML 文書に含まれていないようにしてください。

- **UNKNOWN-REFERENCE-IN-CONTENT イベント (中止)**
  - **XMLSS** なくなりました。代わりに、UNRESOLVED-REFERENCE イベントまたは EXCEPTION イベントが発生します。
  - **COMPAT** XML-TEXT または XML-NTEXT には、"&" および ";" 区切り文字を含まない、エンティティー参照名が含まれます。
  - **XMLPARSE(XMLSS) へのマイグレーション:** UNKNOWN-REFERENCE-IN-CONTENT を処理するコードを、UNRESOLVED-REFERENCE を処理するように変更してください。

    UNRESOLVED-REFERENCE イベントは、以下の条件がすべて満たされる場合のみシグナル通知されます。
    - 未解決の参照は、属性値ではなくエレメント・コンテント内にある。
    - XML 文書は、standalone="no" を指定する XML 宣言で始まっている。
    - XML 文書には、次の例のような文書タイプ宣言が含まれている。

      ```
      <!DOCTYPE rootElementName>
      ```

    - VALIDATING 句が XML PARSE ステートメントに指定されている場合、次の例のように文書タイプ宣言は外部 DTD サブセットも指定していなければなりません。

      ```
      <!DOCTYPE rootElementName SYSTEM "extSub.dtd">
      ```

    これらの条件が満たされない場合、パーサーは UNRESOLVED-REFERENCE の代わりに EXCEPTION イベントをシグナル通知します。

- **UNRESOLVED-REFERENCE イベント (新規)**
  - **XMLSS** XML-TEXT または XML-NTEXT には、"&" および ";" 区切り文字を含まない、エンティティー参照名が含まれます。
  - **COMPAT:** このイベントはなくなりました。代わりに、UNKNOWN-REFERENCE-IN-CONTENT イベントが発生します。
  - **XMLPARSE(XMLSS) へのマイグレーション:** UNKNOWN-REFERENCE-IN-CONTENT を参照してください。

- **VERSION-INFORMATION イベント (無変更)**
  - 両方のパーサーは、VERSION-INFORMATION イベントに対して同じ振る舞いをします。

– **XMLPARSE(XMLSS) へのマイグレーション:** 変更の必要はありません。

XMLPARSE(XMLSS) と XMLPARSE(COMPAT) とのその他の相違点:

- **XML-CODE**
  - **XMLSS:** パーサーによって、EXCEPTION イベントに対して XML-CODE がセットされている場合、最初のハーフワードが戻りコード、最後のハーフワードが理由コードです。この値は 16 進数に変換してください。共通戻りコードおよび理由コードは、「*z/OS XML System Services User's Guide and Reference*」にあります。また、COBOL 固有の戻りコードおよび理由コードは、「*Enterprise COBOL プログラミング・ガイド*」にあります。
  - **COMPAT:** XML-CODE 値は、「*Enterprise COBOL プログラミング・ガイド バージョン 4 リリース 2*」では 10 進数で記述されています。
  - **XMLPARSE(XMLSS) へのマイグレーション:** プログラムが、EXCEPTION イベントに対して特定の XML-CODE 値をテストする場合、ソース・プログラム内のこれらの値を変更する必要があります。
- **条件処理、RESUME、および XML PARSE ステートメント**
  - **XMLSS:** 実行中の処理プロシージャー内での例外が原因で条件処理ルーチン (CEEHDLR またはランタイム・オプション USERHDLR によって登録される) に制御が移り、 CEEMRCE がプログラム内の XML PARSE ステートメントの前の場所に再開カーソルを移動し、条件マネージャーから RESUME が要求された場合、2 回目の XML PARSE では重大度 3 の次のようなランタイム・エラー・メッセージが出されます。

    ```
    IGZ0228S XML PARSE ステートメントを開始しようとしましたが、無効です。
    ```

  - **COMPAT:** 実行中の処理プロシージャー内での例外が原因で条件処理ルーチン (CEEHDLR またはランタイム・オプション USERHDLR によって登録される) に制御が移り、CEEMRCE がプログラム内の XML PARSE ステートメントの前の場所に再開カーソルを移動し、条件マネージャーから RESUME が要求された場合、2 回目の XML PARSE は正常に開始します。
  - **XMLPARSE(XMLSS) へのマイグレーション:** CEE3SRP の呼び出しを処理プロシージャー内に移動してください。次に、再開ポイントで、条件処理ルーチンが例外からリカバリーできない場合、 -1 を XML-CODE に移すことで構文解析を終了させます。条件処理ルーチンが有効なリカバリーを実行できる場合は、 XML-CODE をそのままにすることで構文解析を続けることができます。

    あるいは、その代替方法として、実行が再開されたときに、処理プロシージャー内で例外が発生した XML PARSE ステートメントを含んでいたプログラムを呼び出したプログラム内に制御が移るように、 CEEMRCE の代わりに CEEMRCR を使用することもできます。

    上記の方法のどちらでも、例外に正しく対処できます。

次の表に、定義済みのエンティティー参照を示します。

表 42. 定義済みエンティティー参照

定義済みエンティティー	文字
&lt;	<
&gt;	>

表 42. 定義済みエンティティー参照 (続き)

定義済みエンティティー	文字
&amp;	&
&apos;	'
&quot;	"

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM もその関連会社も、本書に含まれている推奨事項の実施による結果に責任を負いません。推奨事項の実施は、実施者自身の責任でなされるものとします。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

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

プログラミング・インターフェース情報

本書は、IBM Enterprise COBOL for z/OS を使用してプログラムを作成する際に役立ちます。本書 (移行ガイド) は、IBM Enterprise COBOL for z/OS 用に提供され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 のサービスを使用するプログラムを作成することができます。

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

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。

この用語集には、以下の資料からの用語および定義も含まれます。

- *ANSI INCITS 23-1985, Programming Languages - COBOL* (以下の資料による修正箇所あり)
  - *ANSI INCITS 23a-1989, Programming Languages - Intrinsic Function Module for COBOL*
  - *ANSI INCITS 23b-1993, Programming Language - Correction Amendment for COBOL*
- *ANSI INCITS 172-2002 American National Standard Dictionary of Information Technology*

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。

A

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。

異常終了 (abend)
プログラムの異常終了。

16MB 境界より上 (above the 16-MB line)
いわゆる 16MB 境界より上だが、2GB バーより下のストレージ。このストレージは 31 ビット・モード (AMODE 31) でのみアドレッシング可能です。 1980 年代に IBM が MVS/XA アーキテクチャーを導入するまで、プログラムの仮想ストレージは 16MB に制限されていました。 24 ビット・モード (AMODE 24) でコンパイルされたプログラムは、想像上のストレージ境界の下に保持されているかのように、16MB のスペースのみにアドレッシングできます。VS COBOL II 以来、AMODE 31 を持つプログラムは 16MB 境界より上のデータをアドレッシングできます。

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。

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 (,) によるデータ項目における小数点位置の物理表現。

英字名 (* alphabet-name)
環境部の SPECIAL-NAMES 段落内で、特定の文字セットまたは照合シーケンスに名前を割り当てるユーザー定義語。

英字 (* alphabetic character)
文字またはスペース文字。

英数字 (* alphanumeric character)
コンピューターの 1 バイト文字セット内の任意の文字。

英数字データ項目 (alphanumeric data item)
USAGE DISPLAY として暗示的または明示的に記述され、英数字、英数字編集、または数字編集のカテゴリーを持つデータ項目を指す一般的用語。

英数字編集データ項目 (alphanumeric-edited data item)
少なくとも 1 つの A または X シンボルのインスタンスおよび、少なくとも 1 つの挿入記号 B、0、または / を含む、PICTURE 文字ストリングで記述されるデータ項目。英数字編集データ項目には USAGE DISPLAY があります。

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。

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。

AMODE
リンケージ・エディターによって提供さ

れ、プログラム・オブジェクトが入る必要のあるアドレッシング・モードを示すプログラム・オブジェクトの属性。

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。

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。

引数 (* 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。

算術式 (* 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。

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。

配列 (array)
Language Environment 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。

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。

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)。

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。

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。

AT END 条件 (* AT END condition)
以下の操作のいずれかが原因で発生する条件：

1. 順次アクセス・ファイル用の READ ステートメントについて、以下の条件のいずれかが検出されたとき。
   - ファイルに次の論理レコードが存在しない。
   - 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。

- オプションの入力ファイルが存在しない。

2. RETURN ステートメントについて、関連のソートまたはマージ・ファイル用の次の論理レコードがないとき。
3. SEARCH ステートメントについて、検索動作が、関連の WHEN 句のいずれかに指定された条件を満足させることなく終了したとき。

B

基本的な文書のエンコード方式 (basic document encoding)
XML 文書で、XML パーサーが文書の最初の数バイトを調べることによって判定する、以下のエンコード方式カテゴリーのうちの 1 つ。

- ASCII
- EBCDIC
- Unicode UTF-16、ビッグ・エンディアンまたはリトル・エンディアン。
- その他のサポートされないエンコード方式。
- 認識できないエンコード方式。

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・データおよび UTF-16 文字を保存するときに使用するデフォルト形式。この形式では、バイナリー・データ項目の最下位バイトが最高位アドレスにあり UTF-16 文字の最下位バイトが最高位アドレスにあります。*リトル・エンディアン (little-endian)* と対比。

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。

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。

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。

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。

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。

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。

組み込み関数 (built-in function)
「組み込み関数 (intrinsic function)」を参照。

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。

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。

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。

呼び出し可能サービス (callable services)
Language Environment 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、Language Environment の規則を順守するすべてのプログラムから使用できる。

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。

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。

呼び出し側プログラム (* calling program)
別のプログラムへの CALL を実行するプログラム。

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。

CEEDUMP
Language Environment 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、Language Environment 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。

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。

世紀ウィンドウ (century window)
100 年の間隔であり、Language Environment 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。Language Environment 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。

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。

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。

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。

CICS 顧客情報管理システム (CICS)。

CICS 変換プログラム (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。

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)
クラスを表すランタイム・オブジェクト。

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。

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
?	スペース
+	正符号
-	負符号 (-)
*****	アスタリスク
/	斜線 (スラッシュ)
=	等号
$	通貨記号 (currency sign)
,	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。

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。例えば、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。

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。

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。

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。

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。

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。

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。

コンパイラー・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プログラム・オブジェクトの性質、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。コンパイル時オプション (compile-time options) も参照。

コンパイル時オプション (compile-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プログラム・オブジェクトの性質、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。

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。)

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。

条件 (condition)
Language Environment 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。

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。

CONSOLE
オペレーター・コンソールと関連した COBOL 環境名。

連続項目 (* contiguous items)
手続き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。

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。

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。

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。

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。

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。

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。

D

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。

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。

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。

宣言文 (* declarative sentence)
1 つの USE ステートメントから構成され、分離文字ピリオドで終了するコンパイラー指示ステートメント。

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。

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。

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。

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。

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。

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。

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。

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。

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、識別部、環境部、データ部、および手続き部の 4 つの部がある。

部のヘッダー (* division header)
ワードとその後に続く、部の先頭を示す分離文字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.
```

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。

文書タイプ宣言 (document type declaration)
文書のクラスに対して文法を提供するマークアップ宣言を含むか指し示す XML エレメント。この文法は、文書タイプ定義または DTD として知られています。

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。

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。

動的アクセス (* dynamic access)
特定の論理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。

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可能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。

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、レジスター保存域、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、Language Environment が管理する STACK セグメント内に割り振られる。

E

EBCDIC (拡張 2 進化 10 進交換コード) (* EBCDIC (Extended Binary-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。

編集済みデータ項目 (edited data item)
ゼロの抑制または編集文字の挿入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
?	スペース
0	ゼロ
+	正符号
-	負符号
CR	貸方
DB	借方
Z	ゼロ抑止
*	チェック・プロテクト
$	通貨記号
,	コンマ (小数点)
.	ピリオド (小数点)
/	斜線 (スラッシュ)

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。

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。

エンクレーブ (enclave)
Language Environment 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。

クラス終了マーカー (* end class marker)
ワードの組み合わせに分離文字ピリオドが続いたもので、COBOL クラス定義の終わりを示す。クラス終了マーカーは次のとおり。

```
END CLASS class-name.
```

メソッド終了マーカー (* end method marker)
ワードの組み合わせに分離文字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。メソッド終了マーカーは次のとおり。

```
END METHOD method-name.
```

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。

プログラム終了マーカー (* end program marker)
ワードの組み合わせに分離文字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。プログラム終了マーカーは次のとおり。

```
END PROGRAM program-name.
```

記入項目 (* entry)
一連の連続する記述の文節であり、分離文字ピリオドで終了し、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。

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。

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。

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 (runtime environment)」を参照。

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。

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。

式 (* expression)
演算式または条件式。

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。

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。

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。単一のプログラム・オブジェクト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 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。例えば、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。

例えば、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。

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。

F

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。

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・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。

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。

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。

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。

フォーマット (* format)
一連のデータの特定の配置。

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 には参照修飾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。

G

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。

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。

H

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。

高位桁 (* high order end)
文字ストリングの左端の文字。

HLL 高水準言語。

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。

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。

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、または参照修飾子 (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、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。これは、COBOL (MVS お

よび VM 版) リリース 2、COBOL (OS/390 および VM 版) または Enterprise COBOL 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。

IGZEBST
VS COBOL II のブートストラップ・ルーチン。これは、VS COBOL II 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。

ILC 言語間通信。言語間通信は、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。

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。

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分離文字ピリオド、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。

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。

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。

索引付け (indexing)
索引名を使用しての添え字付けと同義。

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。

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。

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。

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。

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。

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。

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。

入力プロシージャー (* input procedure)
形式 1 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。

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。

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。

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。

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。

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 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。例えば、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。

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。

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)。

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。

*** 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。

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 フェーズを指定せずに 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。

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。

K

K
記憶容量を表すときの 2 の 10 乗。10 進表記では 1024。

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。

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。

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)。

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。

K バイト (KB) (kilobyte (KB))
1K バイトは 1024 バイト。

L

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。

Language Environment
z/OS Language Environment の省略名。C、C++、COBOL、FORTRAN、および PL/I 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アーキテクチャー構造およびインターフェース。 Language Environment に準拠するコンパイラーでコンパイルされたプログラムおよび、Java アプリケーションに必要です。

Language Environment 準拠 (Language Environment-conforming)
Language Environment の共通インターフェース規則に準拠していること。

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。

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

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。

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。

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。

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。

*** 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。

リンク・エディット (link-edit)
ロード可能なコンピューター・プログラムをバインダー (リンケージ・エディター) によって作成すること。

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、呼び出し側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、呼び出し側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。

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。

リトル・エンディアン (little-endian)
PC 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、最大重み数字が最高位アドレスにある。「ビッグ・エンディアン (big-endian)」と対比。

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。例えば、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 (local)

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。

低位桁 (* low order end)
文字ストリングの右端の文字。

M

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。

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。例えば、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。

M バイト (M) (* megabyte (M))
1M バイトは 1,048,576 バイト。

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。

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。

メソッド定義 (* Method Definition)
メソッドを定義する COBOL ソース単位。

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。

メソッド名 (* method-name)
メソッドを識別するユーザー定義語。

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。

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。Language Environment 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。

MVS 多重仮想記憶 (Multiple Virtual Storage) オペレーティング・システム。

N

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。

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。

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。

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。

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。

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。

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。

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。

非数字項目 (* nonnumeric item)
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して記述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。

ヌル (null)
空。意味を持たないこと。

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。

数値項目 (* numeric item)
その内容の記述が、「0」から「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。

O

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・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。

*** 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。

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・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生成され、バインダー (リンケージ・エディター) 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。

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。

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。

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。

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。

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入力フォーマット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。

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。

オプション・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。

オプション・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。

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。

出力プロシージャー (* output procedure)
形式 1 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。

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。

P

パック 10 進数項目 (packed decimal item)
「内部 10 進項目 (internal decimal item)」を参照。

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。

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、内部論理要件または出力メディアの外部特性に基づいて分割される。

ページ本体 (* page body)
論理ページの中で、行の書き込みまたはスペース埋め込みを行うことができる部分。

段落 (* paragraph)
手続き部では、段落名とその後の分離文字ピリオド、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)
予約語とその後の分離文字ピリオドであり、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 ID DIVISION)
CLASS-ID. (Class ID DIVISION)
METHOD-ID. (Method ID 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。

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 で使用される仮引数という用語に対して他の言語で使用される用語。

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。

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。

物理レコード (* physical record)
「ブロック (block)」を参照。

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。

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。

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。

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。

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。

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・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。

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。

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・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。

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。

プログラム名 (program-name)
見出し部およびプログラム終了マーカーにおいて、COBOL ソース・プログラムを識別するユーザー定義語または英数字リテラル。

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 (==)。

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
?	スペース
=	等号

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。

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。

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。

R

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。

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。

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。

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。

レコード名 (* record-name)
COBOL プログラムのデータ部のレコード記述記入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。

レコード番号 (* record number)
順次編成ファイル内でのレコードの序数。

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。

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。

再入可能 (reentrant)
複数のユーザーがプログラム・オブジェクト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。

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。

参照修飾子 (* 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。

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

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

比較条件 (* relation condition)
演算式、データ項目、非数値リテラル、または索引名の値が、別の演算式、データ項目、非数値リテラル、または索引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。)

相対ファイル (* relative file)
相対編成のファイル。

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。

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。

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。

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。

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。

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。Language Environment 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。

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 (runtime environment)
COBOL プログラムが実行される環境。

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。Language Environment では、実行単位はエンクレーブと同義である。

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。

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 (例えば、END-ADD) も暗黙のもの (分離文字ピリオド) もある。ステートメントの終わりの変数。

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・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分離文字ピリオド。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分離文字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 と分離文字ピリオドで構成される。

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。

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。

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分離文字ピリオドで終了する。

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。

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。

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、後に 1 つのスペースが続く 1 つのピリオド (.)。

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。

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。

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。

逐次検索 (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。

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。

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。

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

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。

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。

ソート・ファイル (* sort file)
形式 1 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。

*** 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。

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。

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および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、プログラム終了マーカーで終わる (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。

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 (-)
*	アスタリスク
/	斜線 (スラッシュ)
=	等号
$	通貨記号
,	コンマ
;	セミコロン
.	ピリオド (小数点、終止符)
"	引用符
'	アポストロフィ
(	左括弧
)	右括弧
>	より大きい
<	より小さい
:	コロン
_	下線
*>	浮動コメント標識

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。

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。

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。

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。

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。

STL STL File System。 COBOL および PL/I のネイティブ・ワークステーション・ファイル・システムおよび PC ファイル・システム。例外として明示的に記載されているものを除き、完全な ANSI 85 COBOL 標準入出力言語、および「*COBOL 言語解説書*」に記載されているすべての拡張機能を含め、順次ファイル、相対ファイル、索引ファイルをサポートする。

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。

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。

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。

サブプログラム (* subprogram)
「呼び出し先プログラム (called program)」を参照。

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、可変数の引数が許可される関数の関数引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。

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。

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。

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。

T

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。

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・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開始区切り文字と終了区切り文字。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。

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。

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。

U

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。

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。

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。

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。例えば、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。

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。

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。

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。

V

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。

*** 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。

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。

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。

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)
ファイルの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。

VSAM (仮想記憶アクセス方式) (VSAM (Virtual Storage Access Method))
高性能の大容量記憶アクセス方式。入力順データ・セット (ESDS)、キー順データ・セット (KSDS)、および相対レコード・データ・セット (RRDS) の 3 種類のデータ編成を使用できる。COBOL でこれらに相当するものは、それぞれ、順次編成、索引編成、および相対編成。

W

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。

*** 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。

X

XML Extensible Markup Language。SGML から派生し、SGML のサブセットであるマークアップ言語を定義するための標準メタ言語。XML では、SGML の複雑で使用頻度の低い部分が省略され、文書タイプや作成者の処理、構造化された情報の管理、多様なコンピューター・システム間での構造化された情報の送信や共有を行うアプリケーションを、より容易に作成することができます。XML を使用するには、SGML に必要な堅固なアプリケーションおよび処理は必要ありません。XML は、World Wide Web Consortium (W3C) の後援のもとで開発されました。

XML データ (XML data)
XML エレメントが階層構造に編成されたデータ。データ定義は XML エレメント型宣言で定義されます。

XML 宣言 (XML declaration)
使用される XML のバージョンや文書のエンコード方式などの、XML 文書の特徴を指定する XML テキスト。

XML 文書 (XML document)
W3C XML 仕様による定義に従ってよく形成されたデータ・オブジェクト。

XML 名前空間 (XML namespace)
W3C XML 名前空間仕様によって定義され、エレメント名および属性名の集合の有効範囲を制限するメカニズム。一意的に選択された XML 名前空間により、複数の XML 文書または XML 文書内の複数のコンテキストに渡って、エレメント名または属性名が固有であることが保証されます。

XML スキーマ (XML schema)
W3C によって定義された、XML 文書の構造および内容の記述と制限のためのメカニズム。XML スキーマ (これ自体も XML 中に表される) が、任意のタイプの XML 文書のクラスを効果的に定義する (例えば購入注文など)。

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。例えば、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。

Z

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。

ゾーン 10 進数項目 (zoned decimal item)
「外部 10 進数項目 (external decimal item)」を参照。

#

85 COBOL 標準 (85 COBOL Standard)
以下の標準によって定義された COBOL 言語。

- 「*ANSI INCITS 23-1985, Programming languages - COBOL*」は「*ANSI INCITS 23a-1989, Programming Languages - COBOL - Intrinsic Function Module for COBOL*」および「*ANSI INCITS 23b-1993, Programming Languages - Correction Amendment for COBOL*」に改訂されました。
- 「*ISO 1989:1985, Programming languages - COBOL*」は「*ISO/IEC 1989/AMD1:1992, Programming languages - COBOL: Intrinsic function module*」および「*ISO/IEC 1989/AMD2:1994, Programming languages - Correction and clarification amendment for COBOL*」に改訂されました。

2002 COBOL 標準 (2002 COBOL Standard)
以下の標準によって定義された COBOL 言語。

- INCITS/ISO/IEC 1989-2002, Information Technology - Programming Languages - COBOL
- ISO/IEC 1989:2002, Information technology -- Programming languages -- COBOL

資料名リスト

IBM Enterprise COBOL for z/OS

以下の資料が「Enterprise COBOL for z/OS library」にあります。

- *カスタマイズ・ガイド*、SA88-5309
- *言語解説書*、SA88-5311
- *プログラミング・ガイド*、SA88-5310
- *移行ガイド*、GA88-5312
- *Program Directory*、GI11-9180
- *Licensed Program Specifications*、GI11-9181

関連資料

z/OS ライブラリー資料

以下の資料が「z/OS Internet Library」にあります。

ランタイム・ライブラリー拡張機能

- *DWARF/ELF エクステンション ライブラリー・リファレンス*
- *共通デバッグ・アーキテクチャー ライブラリー・リファレンス*
- *共通デバッグ・アーキテクチャー ユーザーズ・ガイド*

z/Architecture

- *解説書*

z/OSDFSMS

- *カタログのためのアクセス方式サービス・プログラム*
- *Checkpoint/Restart*
- *Macro Instructions for Data Sets*
- *データ・セットの使用法*
- *Utilities*

z/OS DFSORT

- *Application Programming Guide*
- *インストールおよびカスタマイズ*

z/OS ISPF

- *ダイアログ開発者 ガイドとリファレンス*
- *ユーザーズ・ガイド 第 I 巻*
- *ユーザーズ・ガイド 第 II 巻*

z/OS 言語環境プログラム

- *概念*
- *カスタマイズ*
- *デバッグのガイド*
- *プログラミング・ガイド*
- *プログラミング・リファレンス*
- *ランタイム・メッセージ*
- *ランタイム・アプリケーション マイグレーション・ガイド*
- *ILC (言語間通信) アプリケーションの作成*

z/OS MVS

- *JCL 解説書*
- *JCL ユーザーズ・ガイド*
- *プログラム管理: ユーザーズ・ガイドおよび解説書*
- *システム・コマンド*
- *z/OS Unicode Services ユーザーズ・ガイドおよび解説書*
- *z/OS XML System Services User's Guide and Reference*

z/OS TSO/E

- *コマンド解説書*
- *入門*
- *ユーザーズ・ガイド*

z/OS UNIX システム・サービス

- *コマンド解説書*
- *プログラミング : アセンブラー呼び出し可能サービス 解説書*
- *ユーザーズ・ガイド*

z/OS XL C/C++

- *プログラミング・ガイド*
- *ランタイム・ライブラリー・リファレンス*

CICS Transaction Server for z/OS

以下の資料が「CICS」にあります。

- *Application Programming Guide*
- *Application Programming Reference*
- *カスタマイズ・ガイド*
- *External Interfaces Guide*

DB2 for z/OS

以下の資料が「DB2 for z/OS Library」にあります。

- *アプリケーション・プログラミングおよび SQL ガイド*
- *コマンド解説書*
- *SQL 解説書*

Debug Tool

以下の資料が「Debug Tool for z/OS」にあります。

- *リファレンスおよびメッセージ*
- *ユーザーズ・ガイド*

以下の資料は、「IBM Publications Center」で資料番号を検索することで見つけることができます。

COBOL 報告書作成プログラム・プリコンパイラー

- *Programmer's Manual*、SC26-4301

IMS

- *Application Programming API Reference*、SC18-9699
- *Application Programming Guide*、SC18-9698

z/OS のソフトコピー資料

以下のコレクション・キットには、z/OS および関連製品資料が含まれます。

- *z/OS CD Collection Kit*、SK3T-4269

Java

- *IBM SDK for Java - Tools Documentation*、publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp
- *Java 2 Enterprise Edition Developer's Guide*、download.oracle.com/javaee/1.2.1/devguide/html/DevGuideTOC.html
- *Java 2 SDK*、*Standard Edition Documentation*、download.oracle.com/javase/1.4.2/docs/
- *Java EE 5 Tutorial*、download.oracle.com/javaee/5/tutorial/doc/
- *The Java Language Specification, Third Edition*、by Gosling et al.、java.sun.com/docs/books/jls/
- *Java Native Interface*、download.oracle.com/javase/1.5.0/docs/guide/jni/
- *Java Technology Edition SDK User Guides*、www.ibm.com/developerworks/java/jdk/aix/service.html

Unicode および文字表現

- *Unicode*、www.unicode.org/
- *Character Data Representation Architecture Reference and Registry*、SC09-2190

XML

- *Extensible Markup Language (XML)*、www.w3.org/XML/
- *Namespaces in XML 1.0*、www.w3.org/TR/xml-names/
- *Namespaces in XML 1.1*、www.w3.org/TR/xml-names11/
- *XML specification*、www.w3.org/TR/xml/

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]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

プログラム番号: 5655-W32

Printed in Japan

GC43-0801-03

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21