Enterprise COBOL for z/OS 6.3

移行ガイド

GC43-3369-02

注記

本書および本書で紹介する製品をご使用になる前に、357 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 6 リリース 3 (プログラム番号 5655-EC6) 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、Enterprise COBOL for z/OS ライブラリーから無料で参照またはダウンロードできます。 Enterprise COBOL for z/OS は継続的デリバリー (CD) モデルをサポートしており、資料は CD モデルで配布されるフィーチャーを記述するために更新されるので、2 カ月ごとに更新の有無を確認することをお勧めします。

このリリースに関する製品資料は、注文番号を更新せずに定期的に更新する予定です。 製品資料の版を個別に参照する必要がある場合は、更新日付で注文番号を参照してください。

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 またはバージョン 6 に移行するときに役立つ情報が記載されています。

バージョン 5 からバージョン 6 への移行において必要となる変更は少ないため、本書では、ユーザーはバージョン 4 以前のコンパイラーからバージョン 5 またはバージョン 6 に移行するということを前提にしています。

また本書では、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
- Enterprise COBOL Value Unit Edition for z/OS バージョン 5 リリース 2
- IBM Enterprise COBOL for z/OS バージョン 6 リリース 1
- IBM Enterprise COBOL Value Unit Edition for z/OS バージョン 6 リリース 1
- IBM Enterprise COBOL for z/OS バージョン 6 リリース 2
- IBM Enterprise COBOL Value Unit Edition for z/OS バージョン 6 リリース 2
- IBM Enterprise COBOL for z/OS バージョン 6 リリース 1
- IBM Enterprise COBOL Value Unit Edition for z/OS バージョン 6 リリース 3

注 : Enterprise COBOL Value Unit Edition for z/OS は、異なる製品番号および価格設定メトリックで入手可能になった Enterprise COBOL for z/OS と同一です。

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、xxxiii 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、製品番号、*GA* および *EOS* の日付

コンパイラー	リリース・レベル	製品番号	一般出荷開始 (GA) の日付 (年-月-日)	サポート終了 (EOS) の日付 (年-月-日)
OS/VS COBOL	バージョン 1 リリース 2 モディフィケーション 3	5740-CB1	1974-09-23	1999-12-31
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	1976-09-23	1999-12-31
VS COBOL II	バージョン 1 リリース 3	5668-958	1988-12-16	1996-06-30
VS COBOL II	バージョン 1 リリース 4	5668-958	1993-03-12	2001-03-31
COBOL/370	バージョン 1 リリース 1	5688-197	1991-12-20	1997-09-30
COBOL (MVS および VM 版)	バージョン 1 リリース 2	5688-197	1995-10-27	2001-12-31
COBOL (OS/390 および VM 版)	バージョン 2 リリース 1	5648-A25	1997-05-23	2004-12-31
COBOL (OS/390 および VM 版)	バージョン 2 リリース 2	5648-A25	2000-09-29	2004-12-31
Enterprise COBOL for z/OS	バージョン 3 リリース 1	5655-G53	2001-11-30	2004-04-04
Enterprise COBOL for z/OS	バージョン 3 リリース 2	5655-G53	2002-09-27	2005-10-03
Enterprise COBOL for z/OS	バージョン 3 リリース 3	5655-G53	2004-02-27	2007-04-30
Enterprise COBOL for z/OS	バージョン 3 リリース 4	5655-G53	2005-07-01	2015-04-30
Enterprise COBOL for z/OS	バージョン 4 リリース 1	5655-S71	2007-12-14	2014-04-30
Enterprise COBOL for z/OS	バージョン 4 リリース 2	5655-S71	2009-08-28	2022-04-30
Enterprise COBOL for z/OS	バージョン 5 リリース 1	5655-W32	2013-06-21	2020-04-30

表 *1. COBOL* コンパイラー名、バージョン、リリース、製品番号、*GA* および *EOS* の日付 *(*続き*)*

コンパイラー	リリース・レベル	製品番号	一般出荷開始 (GA) の日付 (年-月-日)	サポート終了 (EOS) の日付 (年-月-日)
Enterprise COBOL for z/OS	バージョン 5 リリース 2	5655-W32	2015-02-27	2020-04-30
Enterprise COBOL Value Unit Edition for z/OS[1]	バージョン 5 リリース 2	5697-ECV	2015-10-06	2020-04-30
Enterprise COBOL for z/OS	バージョン 6 リリース 1	5655-EC6	2016-03-18	未発表
Enterprise COBOL Value Unit Edition for z/OS[1]	バージョン 6 リリース 1	5697-V61	2016-03-18	未発表
Enterprise COBOL for z/OS	バージョン 6 リリース 2	5655-EC6	2017-09-08	未発表
Enterprise COBOL Value Unit Edition for z/OS[1]	バージョン 6 リリース 2	5697-V61	2017-09-08	未発表
Enterprise COBOL for z/OS	バージョン 6 リリース 3	5655-EC6	2019-09-06	未発表
Enterprise COBOL Value Unit Edition for z/OS[1]	バージョン 6 リリース 3	5697-V61	2019-09-06	未発表

注:

1. Enterprise COBOL Value Unit Edition for z/OS は、異なる製品番号および価格設定メトリックで入手可能になった Enterprise COBOL for z/OS と同一です。

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。

旧 COBOL コンパイラーから Enterprise COBOL への移行にかかる労力の見積もりには、以下の製品や成果物も役立ちます。

- IBM Enterprise COBOL Developer Trial for z/OS V6.3 (5655-TY6)。

Enterprise COBOL Developer Trial for z/OS には、無料の 90 日間の評価ライセンスが付属しています。Enterprise COBOL Developer Trial for z/OS V6.3 を使用すれば、実動の移行プロジェクトに時間とリソースを投入する前に、非実稼働環境で IBM Enterprise COBOL for z/OS V6.3 の最新の能力を評価できます。

- IBM Enterprise COBOL for z/OS Migration Assistant.

  COBOL Migration Assistant を使用すれば、Enterprise COBOL V4 以前から Enterprise COBOL V5 または V6 へのコンパイラー移行プロセス全体を通してナビゲートすることができます。

- IBM Enterprise COBOL for z/OS Migration Portal。

  事例研究、COBOL のエキスパートへのインタビューのビデオ、クラウド・ベースの COBOL Migration Assistant、COBOL のマイグレーションとパフォーマンス・チューニングに関する無料の Web セミナー、FAQ、移行をサポートする他の IBM 製品、Enterprise COBOL V4 以前のバージョンから Enterprise COBOL V5 または V6 への移行の労力を軽減するのに役立つ他の多くのリソースを確認してください。

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。

関連資料

Enterprise COBOL および Language Environment で提供される情報は、Enterprise COBOL のインストールとカスタマイズ、および z/OS 用の COBOL アプリケーションの作成に役立つように設計されています。

Enterprise COBOL for z/OS の資料

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) 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 for z/OS* 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 for z/OS* 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 for z/OS* 言語解説書

表 2. *Enterprise COBOL for z/OS* の資料 (続き)

作業	資料
問題を診断するために COBOL コンパイラーのメッセージと戻りコードを理解する	*Enterprise COBOL for z/OS* メッセージおよびコード

z/OS の Language Environment エレメントの資料

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
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行われた主な変更を示します。 最新の技術的な変更は、HTML 版では >| と |< の間に囲まれ、PDF 版では左側の余白にある縦棒 (|) で示されています。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2021 年 5 月)

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2021 年 5 月)

- SOURCE オプションに新しいサブオプション DEC | HEX が追加されました。このサブオプションは、ソースのリストのシーケンス番号を 10 進形式と 16 進形式のどちらで生成するかを制御します。詳しくは、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。この変更は、Enterprise COBOL バージョン 6 リリース 3 (APAR PH35643 用 PTF インストール済み) で導入されています。
- 非推奨の ZONEDATA コンパイラー・オプションが新しい INVDATA オプションに置き換えられ、無効なデータが含まれた USAGE DISPLAY および USAGE PACKED-DECIMAL のデータ項目を処理するためのコードをコンパイラーで生成する方法を、ユーザーがきめ細かく制御できるようになりました。詳しくは、

192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。この変更は、Enterprise COBOL バージョン 6 リリース 3 (APAR PH37328 用 PTF インストール済み) で導入されています。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2021 年 3 月)

- 新しい TUNE オプションが追加され、実行可能プログラムの最適化の対象にするアーキテクチャーを指定できるようになりました。詳しくは、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。この変更は、Enterprise COBOL バージョン 6 リリース 3 (APAR PH34804 用 PTF インストール済み) で導入されています。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2021 年 1 月)

- 各 COBOL コンパイラー・バージョンの一般出荷開始 (GA) とサポート終了 (EOS) の日付が、xvi ページの『IBM COBOL コンパイラー、名前およびバージョン別』に追加されました。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2020 年 11 月)

- メッセージ IGZ0268W と IGZ0269W が発行された場合は TRAP(ON) を有効にする必要があるということが、33 ページの『アプリケーションの目録の作成』に明記されました。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2020 年 7 月)

- 新しいオプション QSAMBUFFINITCHAR が IGZUOPT モジュールに追加され、QSAM バッファーの初期化に使用される先頭文字を制御できるようになりました。詳しくは、353 ページの『付録 L QSAM バッファーの初期化の制御』を参照してください。このサポートは、Enterprise COBOL バージョン 6 リリース 3 で、APAR PH25917 に対応するランタイム LE PTF をインストールすることで取得できます。
- 181 ページの『WORKING-STORAGE SECTION の変更』で PPA4 のレイアウトと WORKING-STORAGE の場所についての説明が更新されました。
- 294 ページの『COBOL 報告書作成プログラム・プリコンパイラー』に、IBM 報告書作成プログラムの背景が明記され、サポートの詳細とバージョン要件が追加されました。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2020 年 5 月)

- INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。 詳しくは、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。この変更は、Enterprise COBOL バージョン 6 リリース 3 (APAR PH22581 用 PTF インストール済み) で導入されています。
- INSPECT...TALLYING の動作の変更点が、190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』に明記されました。
- PDS データ・セットから PDSE データ・セットへの移行に関する詳細が、27 ページの『第 3 章 コンパイラーのアップグレード・チェックリスト』に記載されました。
- CICS® の下で TEST(...,NOSEPARATE) を使用してコンパイルしたプログラムには、DFHRPL ライブラリーに対する読み取りアクセス権限が必要であるということが、238 ページの『DFHRPL セットアップにおける Enterprise COBOL V5 および V6 との違い』に明記されました。
- COBOL の移行に関するすべての情報を一元的に検索できる場所である COBOL Migration Information Portal へのリンクが追加されました。詳しくは、xvi ページの『IBM COBOL コンパイラー、名前およびバージョン別』を参照してください。
- 355 ページの『付録 M Enterprise COBOL for z/OS のアクセシビリティー機能』で、「IBM Developer for z Systems®」が「IBM Developer for z/OS」に更新されました。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2020 年 1 月)

- ランタイム LE PTF UI67483(V2R2)/UI67485(V2R3)/UI67486(V2R4) をインストールすると、Enterprise COBOL V5 以降のバージョンでの NOBLKSET と従来のマージ方式のサポートが取得されます。詳しくは、214 ページの『DFSORT オプション NOBLKSET の使用』を参照してください。
- APAR PH20724 に対応する PTF をインストールすると、CALL ステートメントの USING 句を使用してサブプログラムに *file-name* を渡す方法が復元されます。詳しくは、179 ページの『CALL ... USING ステートメントでの file-name の使用』を参照してください。
- 移行前によく尋ねられる質問とその回答が、移行前の FAQ に追加されました。

GC43-3369-02 (英文原典: GC27-8715-02) での変更点 (2019 年 9 月)

- Enterprise COBOL V6.3 の変更点に関する情報が 173 ページの『第 16 章 Enterprise COBOL バージョン 6 における変更』の章に追加されました。 変更点は主に、以下のトピックに該当するものです。
  - 前提ソフトウェア・レベルの変更
  - コンパイラー・オプションの変更
  - コンパイラーの動作の変更 (リストの変更、カタログ式プロシージャーの変更、共有ストレージでのコンパイラー・フェーズの変更、CALL ... USING *file-name* の変更)
  - ベンダー・ツールに影響する可能性がある変更 (PPA1 の変更)

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2021 年 2 月)

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2021 年 2 月)

- 共有ストレージにコンパイラー・フェーズを配置するためのインストール・カスタマイズが除去されました。
- 各 COBOL コンパイラー・バージョンの一般出荷開始 (GA) とサポート終了 (EOS) の日付が、xvi ページの『IBM COBOL コンパイラー、名前およびバージョン別』に追加されました。

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2020 年 12 月)

- メッセージ IGZ0268W と IGZ0269W が発行された場合は TRAP(ON) を有効にする必要があるということが、33 ページの『アプリケーションの目録の作成』に明記されました。

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2020 年 8 月)

- 190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』で `INSPECT...TALLYING` の動作の例が更新されました。
- よくあるご質問 (FAQ) とその回答が、254 ページの『互換性』、256 ページの『Enterprise COBOL でのコンパイル』、260 ページの『サブシステム』、262 ページの『パフォーマンス』に追加されました。

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2020 年 6 月)

- 新しいオプション `QSAMBUFFINITCHAR` が `IGZUOPT` モジュールに追加され、QSAM バッファーの初期化に使用される先頭文字を制御できるようになりました。詳しくは、353 ページの『付録 L QSAM バッファーの初期化の制御』を参照してください。このサポートは、Enterprise COBOL バージョン 6 リリース 2 で、APAR PH25917 に対応するランタイム LE PTF をインストールすることで取得できます。
- 181 ページの『WORKING-STORAGE SECTION の変更』で PPA4 のレイアウトと WORKING-STORAGE の場所についての説明が更新されました。
- 294 ページの『COBOL 報告書作成プログラム・プリコンパイラー』に、IBM 報告書作成プログラムの背景が明記され、サポートの詳細とバージョン要件が追加されました。

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2020 年 4 月)

- INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。 詳しくは、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。この変更は、Enterprise COBOL バージョン 6 リリース 2 (APAR PH24413 用 PTF インストール済み) で導入されています。
- INSPECT...TALLYING の動作の変更点が、190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』に明記されました。
- PDS データ・セットから PDSE データ・セットへの移行に関する詳細が、27 ページの『第 3 章 コンパイラーのアップグレード・チェックリスト』に記載されました。
- CICS の下で TEST(...,NOSEPARATE) を使用してコンパイルしたプログラムには、DFHRPL ライブラリーに対する読み取りアクセス権限が必要であるということが、238 ページの『DFHRPL セットアップにおける Enterprise COBOL V5 および V6 との違い』に明記されました。
- COBOL の移行に関するすべての情報を一元的に検索できる場所である COBOL Migration Information Portal へのリンクが追加されました。詳しくは、xvi ページの『IBM COBOL コンパイラー、名前およびバージョン別』を参照してください。
- 355 ページの『付録 M Enterprise COBOL for z/OS のアクセシビリティー機能』で、「IBM Developer for z Systems」が「IBM Developer for z/OS」に更新されました。

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2020 年 2 月)

- ランタイム LE PTF UI67483(V2R2)/UI67485(V2R3)/UI67486(V2R4) をインストールすると、Enterprise COBOL V5 以降のバージョンでの NOBLKSET と従来のマージ方式のサポートが取得されます。詳しくは、214 ページの『DFSORT オプション NOBLKSET の使用』を参照してください。
- 移行前によく尋ねられる質問とその回答が、移行前の FAQ に追加されました。

GC43-3369-01 (英語原典: GC27-8715-01) における変更 (2019 年 3 月)

- 新規コンパイラー・オプション INITIAL に関する情報が 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』に追加されました。 変更は、Enterprise COBOL バージョン 6 リリース 2 (APAR PH05855 用 PTF インストール済み) で導入されています。
- 258 ページの『Enterprise COBOL プログラムのバインド (リンク・エディット)』において、Enterprise COBOL V4 から V5 または V6 の間の、コンパイラー生成シンボルの変更に関する FAQ と回答を追加しました。
- Enterprise COBOL V5 または V6 への移行をサポートする Enterprise COBOL V4 PTF がリストされているサポート・ページのリンクを追加しました。 詳細については、161 ページの『第 14 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。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2018 年 9 月)

- Enterprise COBOL V5 および V6 で WORKING-STORAGE SECTION を見つける方法が更新されて、さらに明確にされました。 詳細については、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2018 年 7 月)

- NUMCHECK(ZON) コンパイラー・オプションの 2 つの新規サブオプション ALPHNUM および NOALPHNUM に関する情報が追加されました。 これらのサブオプションは、英数字データ項目、英数字リテラル、または英数字形象定数と比較されるゾーン 10 進数データ項目に対する暗黙的数値クラス・テスト用のコードをコンパイラーが生成するのかどうかを制御します。 変更は、Enterprise COBOL バージョン 6 リリース 2 (APAR PI98480 用 PTF インストール済み) で導入されています。

- Edge Portfolio Analyzer が、無料でオープン・ソースの COBOL アナライザーで置き換えられました。 詳しくは、295 ページの『無料でオープン・ソースの COBOL アナライザー』 を参照してください。
- NORES でコンパイルされた OS/VS COBOL プログラムおよび VS COBOL II プログラムにコンパイラー・マイグレーションが必要であることが 22 ページの『コンパイラー移行』 に明記されました。
- 181 ページの『WORKING-STORAGE SECTION の変更』 において PPA4 レイアウトのオフセット X'2C' に関する説明が訂正されました。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2018 年 5 月)

- 237 ページの『CSD セットアップにおける Enterprise COBOL V5 および V6 との違い』 にある情報が更新されました。 特定の CICS TS バージョンに LE プログラム用のシステム自動インストール 機能が備わっていて、そのプログラムの初回ロード時に CICS がプログラム定義を自動的に作成するためです。
- 別の CICS 変換プログラムが依然として現行 CICS 製品に同梱されていますが今後拡張されることがないことが 240 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』 に明記されました。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2018 年 3 月)

- 187 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 の前提ソフトウェアおよび前提サービス』 から OMVS セグメント要件が削除されました。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2018 年 1 月)

- 新規コンパイラー・オプション COPYLOC に関する情報が 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』に追加されました。 変更は、Enterprise COBOL バージョン 6 リリース 2 (APAR PI91584 用 PTF インストール済み) で導入されています。
- RULES コンパイラー・オプションの新規サブオプションに関する情報が 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』 に追加されました。 Enterprise COBOL バージョン 6 リリース 2 (APAR PI91585 および PI91586 用 PTF インストール済み) で変更が導入されています。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2017 年 11 月)

- 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』 で、ZONEDATA コンパイラー・オプションの動作が更新されました。この変更は、Enterprise COBOL バージョン 6 リリース 2 (APAR PI90571 用 PTF インストール済み) で導入されています。
- IGZEBST の要件が 3 ページの『第 1 章 新しいコンパイラーとランタイムの紹介』 と他のいくつかのトピックで明記されました。

GC43-3369-01 (英語原典: GC27-8715-01) での変更点 (2017 年 9 月)

- Enterprise COBOL V6.2 の変更点に関する情報が 173 ページの『第 16 章 Enterprise COBOL バージョン 6 における変更』 の章に追加されました。 変更点は主に、以下のトピックに該当するものです。
  - 前提ソフトウェア・レベルの変更
  - COBOL ソース・コードの相違
  - コンパイラー・オプションの変更
  - リストの変更
- Enterprise COBOL V6 での z/OS MEMLIMIT の変更点に関する情報が更新されました。 (201 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 でのコンパイルの変更点』)
- WORKING-STORAGE SECTION の見つけ方に関する情報が更新されました。 (181 ページの『WORKING-STORAGE SECTION の変更』)

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2021 年 2 月)

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2021 年 2 月)

- 共有ストレージにコンパイラー・フェーズを配置するためのインストール・カスタマイズが除去されました。
- 各 COBOL コンパイラー・バージョンの一般出荷開始 (GA) とサポート終了 (EOS) の日付が、xvi ページの『IBM COBOL コンパイラー、名前およびバージョン別』に追加されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2020 年 12 月)

- メッセージ IGZ0268W と IGZ0269W が発行された場合は TRAP(ON) を有効にする必要があるということが、33 ページの『アプリケーションの目録の作成』に明記されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2020 年 8 月)

- 190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』で INSPECT...TALLYING の動作の例が更新されました。
- よくあるご質問 (FAQ) とその回答が、254 ページの『互換性』、256 ページの『Enterprise COBOL でのコンパイル』、260 ページの『サブシステム』、262 ページの『パフォーマンス』に追加されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2020 年 6 月)

- 新しいオプション QSAMBUFFINITCHAR が IGZUOPT モジュールに追加され、QSAM バッファーの初期化に使用される先頭文字を制御できるようになりました。詳しくは、353 ページの『付録 L QSAM バッファーの初期化の制御』を参照してください。このサポートは、Enterprise COBOL バージョン 6 リリース 1 で、APAR PH25917 に対応するランタイム LE PTF をインストールすることで取得できます。
- 181 ページの『WORKING-STORAGE SECTION の変更』で PPA4 のレイアウトと WORKING-STORAGE の場所についての説明が更新されました。
- 294 ページの『COBOL 報告書作成プログラム・プリコンパイラー』に、IBM 報告書作成プログラムの背景が明記され、サポートの詳細とバージョン要件が追加されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2020 年 4 月)

- INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。 詳しくは、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。この変更は、Enterprise COBOL バージョン 6 リリース 1 (APAR PH24414 用 PTF インストール済み) で導入されています。
- INSPECT...TALLYING の動作の変更点が、190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』に明記されました。
- PDS データ・セットから PDSE データ・セットへの移行に関する詳細が、27 ページの『第 3 章 コンパイラーのアップグレード・チェックリスト』に記載されました。
- CICS の下で TEST(...,NOSEPARATE) を使用してコンパイルしたプログラムには、DFHRPL ライブラリーに対する読み取りアクセス権限が必要であるということが、238 ページの『DFHRPL セットアップにおける Enterprise COBOL V5 および V6 との違い』に明記されました。
- COBOL の移行に関するすべての情報を一元的に検索できる場所である COBOL Migration Information Portal へのリンクが追加されました。詳しくは、xvi ページの『IBM COBOL コンパイラー、名前およびバージョン別』を参照してください。
- 355 ページの『付録 M Enterprise COBOL for z/OS のアクセシビリティー機能』で、「IBM Developer for z Systems」が「IBM Developer for z/OS」に更新されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2020 年 2 月)

- ランタイム LE PTF UI67483(V2R2)/UI67485(V2R3)/UI67486(V2R4) をインストールすると、Enterprise COBOL V5 以降のバージョンでの NOBLKSET と従来のマージ方式のサポートが取得されます。詳しくは、214 ページの『DFSORT オプション NOBLKSET の使用』を参照してください。
- 移行前によく尋ねられる質問とその回答が、移行前の FAQ に追加されました。

GC43-3028-04 (英文原典: GC27-8715-00) における変更 (2019 年 3 月)

- 258 ページの『Enterprise COBOL プログラムのバインド (リンク・エディット)』において、Enterprise COBOL V4 から V5 または V6 の間の、コンパイラー生成シンボルの変更に関する FAQ と回答を追加しました。
- Enterprise COBOL V5 または V6 への移行をサポートする Enterprise COBOL V4 PTF がリストされているサポート・ページのリンクを追加しました。 詳細については、161 ページの『第 14 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。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2018 年 9 月)

- NUMCHECK(ZON) コンパイラー・オプションの 2 つの新規サブオプション ALPHNUM および NOALPHNUM に関する情報が追加されました。 これらのサブオプションは、英数字データ項目、英数字リテラル、または英数字形象定数と比較されるゾーン 10 進数データ項目に対する暗黙的数値クラス・テスト用のコードをコンパイラーが生成するのかどうかを制御します。 変更は、Enterprise COBOL バージョン 6 リリース 1 (APAR PH01251 用 PTF インストール済み) で導入されています。
- Enterprise COBOL V5 および V6 で WORKING-STORAGE SECTION を見つける方法が更新されて、さらに明確にされました。 詳細については、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2018 年 7 月)

- Edge Portfolio Analyzer が、無料でオープン・ソースの COBOL アナライザーで置き換えられました。 詳しくは、295 ページの『無料でオープン・ソースの COBOL アナライザー』を参照してください。
- NORES でコンパイルされた OS/VS COBOL プログラムおよび VS COBOL II プログラムにコンパイラー・マイグレーションが必要であることが 22 ページの『コンパイラー移行』に明記されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2018 年 5 月)

- 新規コンパイラー・オプション COPYLOC に関する情報が 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』に追加されました。 変更は、Enterprise COBOL バージョン 6 リリース 1 (APAR PI96231 用 PTF インストール済み) で導入されています。
- 237 ページの『CSD セットアップにおける Enterprise COBOL V5 および V6 との違い』にある情報が更新されました。 特定の CICS TS バージョンに LE プログラム用のシステム自動インストール 機能が備わっていて、そのプログラムの初回ロード時に CICS がプログラム定義を自動的に作成するためです。
- 別の CICS 変換プログラムが依然として現行 CICS 製品に同梱されていますが今後拡張されることがないことが 240 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』に明記されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2018 年 3 月)

- 187 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 の前提ソフトウェアおよび前提サービス』から OMVS セグメント要件が削除されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2017 年 11 月)

- 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』で、ZONEDATA コンパイラー・オプションの動作が更新されました。 この変更は、Enterprise COBOL バージョン 6 リリース 1 (APAR PI88271 用 PTF インストール済み) で導入されています。

- IGZEBST の要件が 3 ページの『第 1 章 新しいコンパイラーとランタイムの紹介』と他のいくつかのトピックで明記されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2017 年 6 月)

- 87 ページの『移行済みプログラム用のコンパイラー・オプション』で、NOSTGOPT コンパイラー・オプションの動作が更新されました。 この変更は、Enterprise COBOL バージョン 6 リリース 1 (APAR PI81838 用 PTF インストール済み) で導入されています。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2017 年 4 月)

- パラメーターの不一致 (つまり、プログラムが引数をサブプログラムに渡した後で、それらの引数がパラメーターとして誤用される場合) を検出できる、新しい PARMCHECK コンパイラー・オプションに関する情報が追加されました。 PARMCHECK オプションは、Enterprise COBOL V6 へのマイグレーションに、また適切なプログラミング方式を確認するために役立ちます。 PARMCHECK オプションは、Enterprise COBOL バージョン 6 リリース 1 (APAR PI78089 用 PTF インストール済み) に導入されています。
- ソース・プログラム内の PERFORM ステートメントによって参照されているプロシージャー (段落またはセクション) のインライン化が許可されるかどうかを制御する、新しい INLNE コンパイラー・オプションに関する情報が追加されました。 INLINE オプションは、Enterprise COBOL バージョン 6 リリース 1 (APAR PI77981 用 PTF インストール済み) に導入されています。
- プロセス間通信 (IPC) メッセージ・キューに関する情報が、158 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるデバッグ情報の変更』に追加されました。
- 211 ページの『正しくないプログラムのエラー動作変更』において、コード・サンプルを更新し、パラメーター長に不一致があるプログラムに対して Enterprise COBOL V5 および V6 で行われた動作変更を分かりやすく記述しました。
- 「V4 において、COMP-5 データ項目値 (符号付き/符号なし) が PIC X(n) データ項目に移されると、誤った値が移される」という情報が追加されました。 これは、Enterprise COBOL V5 および V6 で修正されました。 詳しくは、190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』を参照してください。
- 206 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点』において、ILBOABN0 呼び出し可能サービスを Enterprise COBOL V5 以降のバージョン で呼び出すときの情報が更新されました。
- XML System Services パーサーを使用して、EBCDIC 文書を解析するときに、いくつかの文字または文字の組み合わせを x'15' に変換するときの情報が、345 ページの『付録 K XMLPARSE(COMPAT) から XMLPARSE(XMLSS) へのマイグレーション』に追加されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2017 年 2 月)

- データ項目が送信データ項目として使用される場合に、それらのデータ項目を検証するために追加のコードを生成するかどうかを制御する、新しい NUMCHECK コンパイラー・オプションに関する情報が追加されました。 NUMCHECK オプションは、Enterprise COBOL バージョン 6 リリース 1 (APAR PI71625 用 PTF インストール済み) に導入されています。
- ソース・プログラムで ILBOABN0 呼び出し可能サービスの呼び出しが検出されたときに出される新しい警告メッセージに関する情報が、206 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点』に追加されました。
- 範囲検査が失敗した場合の COBOL プログラムの実行時動作を制御する、2 つの新しいサブオプション MSG および ABD が SSRANGE コンパイラー・オプションに追加されました。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2016 年 9 月)

- Enterprise COBOL V6 の変更点に焦点を当てた情報が、173 ページの『第 16 章 Enterprise COBOL バージョン 6 における変更』に追加されました。
- 未初期化データ項目を検査し、それらが初期化されていないまま使用されている場合に警告メッセージを出すかどうかを制御する、新しい INITCHECK コンパイラー・オプションに関する情報が追加されまし

た。 INITCHECK オプションは、Enterprise COBOL バージョン 6 リリース 1 (APAR PI68226 用 PTF インストール済み) に導入されています。

GC43-3028-04 (英文原典: GC27-8715-00) での変更点 (2016 年 3 月)

- Enterprise COBOL V6 の変更に関する情報が 171 ページの『第 4 部 Enterprise COBOL バージョン 5 およびバージョン 6 での新機能および相違点』に追加されました。V6 の変更は主に、以下のトピックに対するものです。
  - 前提ソフトウェア・レベルの変更
  - COBOL ソース・コードの相違
  - コンパイラー・オプションの変更
  - 大きなプログラムにおけるシステム MEMLIMIT 設定への依存性
  - ランタイムの変更
  - ベンダー・ツールに影響する可能性がある変更
- 以下の新しい予約語が追加されました。
  - ALLOCATE
  - DEFAULT
  - END-JSON
  - FREE
  - JSON
  - JSON-CODE
- 以前のバージョンから Enterprise COBOL V5 および V6 に移行する際の特別な考慮事項を含んだ、トピック 29 ページの『第 4 章 Enterprise COBOL V6 へのマイグレーションに関する推奨事項』が追加されました。

GC43-0801-03 (英語原典: GC14-7383-03) における変更 (2019 年 3 月)

GC43-0801-03 (英語原典: GC14-7383-03) における変更 (2019 年 3 月)

- Enterprise COBOL V5 または V6 への移行をサポートする Enterprise COBOL V4 PTF がリストされているサポート・ページのリンクを追加しました。詳細については、161 ページの『第 14 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。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2018 年 9 月)

- NUMCHECK(ZON) コンパイラー・オプションの 2 つの新規サブオプション ALPHNUM および NOALPHNUM に関する情報が追加されました。 これらのサブオプションは、英数字データ項目、英数字リテラル、または英数字形象定数と比較されるゾーン 10 進数データ項目に対する暗黙的数値クラス・テスト用のコードをコンパイラーが生成するのかどうかを制御します。 変更は、Enterprise COBOL バージョン 5 リリース 2 (APAR PH01241 用 PTF インストール済み) で導入されています。
- Enterprise COBOL V5 で WORKING-STORAGE SECTION を見つける方法に関する情報が追加されました。 詳細については、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2018 年 7 月)

- Edge Portfolio Analyzer が、無料でオープン・ソースの COBOL アナライザーで置き換えられました。 詳しくは、295 ページの『無料でオープン・ソースの COBOL アナライザー』を参照してください。
- NORES でコンパイルされた OS/VS COBOL プログラムおよび VS COBOL II プログラムにコンパイラー・マイグレーションが必要であることが 22 ページの『コンパイラー移行』に明記されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2018 年 5 月)

- 237 ページの『CSD セットアップにおける Enterprise COBOL V5 および V6 との違い』にある情報が更新されました。 特定の CICS TS バージョンに LE プログラム用のシステム自動インストール 機能が備わっていて、そのプログラムの初回ロード時に CICS がプログラム定義を自動的に作成するためです。
- 別の CICS 変換プログラムが依然として現行 CICS 製品に同梱されていますが今後拡張されることがないことが 240 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』に明記されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2017 年 11 月)

- 192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更』で、ZONEDATA コンパイラー・オプションの動作が更新されました。この変更は、Enterprise COBOL バージョン 5 リリース 2 (APAR PI90458 用 PTF インストール済み) で導入されています。
- IGZEBST の要件が 3 ページの『第 1 章 新しいコンパイラーとランタイムの紹介』と他のいくつかのトピックで明記されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2017 年 9 月)

- ファイル整合性検査の検証を必要とする正常実行された VSAM OPEN ステートメントから報告されるユーザー・ファイル状況に作用する新規 VSAMOPENFS コンパイラー・オプションに関する情報が追加されました。 VSAMOPENFS オプションは Enterprise COBOL バージョン 5 リリース 2 (APAR PI85868 用 PTF インストール済み) で導入されました。
- SSRANGE コンパイラー・オプションの 2 つの新規サブオプション MSG および ABD に関する情報が追加されました。 これらのサブオプションは、範囲検査が失敗したときの COBOL プログラムの実行時動作を制御します。 MSG および ABD は Enterprise COBOL バージョン 5 リリース 2 (APAR PI86343 用 PTF インストール済み) で導入されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2017 年 6 月)

- データ項目が送信データ項目として使用される場合に、それらのデータ項目を検証するために追加のコードを生成するかどうかを制御する、新しい NUMCHECK コンパイラー・オプションに関する情報が追加されました。NUMCHECK オプションは Enterprise COBOL バージョン 5 リリース 2 (APAR PI81006 用 PTF インストール済み) で導入されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2017 年 4 月)

- コード・サンプルを更新し、パラメーター長に不一致があるプログラムに対して Enterprise COBOL V5 および V6 で行われた動作変更を分かりやすく記述しました。 詳しくは、211 ページの『正しくないプログラムのエラー動作変更』を参照してください。
- 「V4 において、COMP-5 データ項目値 (符号付き/符号なし) が PIC X(n) データ項目に移されると、誤った値が移される」という情報が追加されました。 これは、Enterprise COBOL V5 および V6 で修正されました。 詳しくは、190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』を参照してください。
- 206 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点』において、`ILBOABN0` 呼び出し可能サービスを Enterprise COBOL V5 以降のバージョン で呼び出すときの情報が更新されました。
- XML System Services パーサーを使用して、EBCDIC 文書を解析するときに、いくつかの文字または文字の組み合わせを x'15' に変換するときの情報が、345 ページの『付録 K XMLPARSE(COMPAT) から XMLPARSE(XMLSS) へのマイグレーション』に追加されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2017 年 1 月)

- ソース・プログラムで `ILBOABN0` 呼び出し可能サービスの呼び出しが検出されたときに出される新しい警告メッセージに関する情報が、206 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点』に追加されました。

GC43-0801-03 (英語原典: GC14-7383-03) での変更点 (2016 年 9 月)

- 未初期化データ項目を検査し、それらが初期化されていないまま使用されている場合に警告メッセージを出すかどうかを制御する、新しい INITCHECK コンパイラー・オプションに関する情報が追加されました。 INITCHECK オプションは Enterprise COBOL バージョン 5 リリース 2 (APAR PI69197 用 PTF インストール済み) で導入されました。

GC43-0801-03 (英語原典: GC14-7383-03) における変更 (2015 年 7 月)

コンパイラー・オプションの変更に関する情報が追加されました。

- 新しいオプション: ZONECHECK(MSG|ABD)
- 変更されたオプション: ZONEDATA。 新しいサブオプション NOPFD が ZONEDATA コンパイラー・オプションに追加されました。ZONEDATA(NOPFD) を使用すると、COBOL V4 で NUMPROC(NOPFD|PFD) を使用した場合に COBOL V4 が実行するのと同じ方法で、ゾーン 10 進数データの比較を実行するコードをコンパイラーが生成できるようになります。

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

- 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。

GC43-0801-02 (英語原典: GC14-7383-02) における変更 (2019 年 3 月)

GC43-0801-02 (英語原典: GC14-7383-02) における変更 (2019 年 3 月)

- Enterprise COBOL V5 または V6 への移行をサポートする Enterprise COBOL V4 PTF がリストされているサポート・ページのリンクを追加しました。詳細については、161 ページの『第 14 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。

GC14-7383-02 での変更点 (2018 年 9 月)

- Enterprise COBOL V5 で WORKING-STORAGE SECTION を見つける方法に関する情報が追加されました。 詳細については、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。

GC14-7383-02 での変更点 (2018 年 5 月)

- 237 ページの『CSD セットアップにおける Enterprise COBOL V5 および V6 との違い』にある情報が更新されました。 特定の CICS TS バージョンに LE プログラム用のシステム自動インストール 機能が備わっていて、そのプログラムの初回ロード時に CICS がプログラム定義を自動的に作成するためです。

- 別の CICS 変換プログラムが依然として現行 CICS 製品に同梱されていますが今後拡張されることがないことが 240 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』に明記されました。

GC14-7383-02 での変更点 (2017 年 4 月)

- 206 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点』において、`ILBOABN0` 呼び出し可能サービスを Enterprise COBOL V5 以降のバージョン で呼び出すときの情報が更新されました。

GC14-7383-02 での変更点 (2017 年 2 月)

- ソース・プログラムで `ILBOABN0` 呼び出し可能サービスの呼び出しが検出されたときに出される新しい警告メッセージに関する情報が、206 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点』に追加されました。

GC14-7383-02 での変更点 (2014 年 3 月)

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 (英文原典: GC14-7383-00) 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、227 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、161 ページの『第 14 章 Enterprise COBOL バージョン 4 からのアップグレード』、167 ページの『第 15 章 Enterprise COBOL バージョン 4 プログラムのコンパイル』、および 187 ページの『第 17 章 Enterprise COBOL バージョン 5 およびバージョン 6 での変更点』を参照してください。

GC88-4746-01 (英文原典: GC23-8527-01) での変更点 (2009 年 8 月)

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、265 ページの『付録 B COBOL 予約語の比較』 および 329 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。

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

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。

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

- 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。

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

- 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。

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

- 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

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

- 本書に適用されたサービス更新

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

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。

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

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。

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

コンパイラー

- 47 ページの『第 6 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 Enterprise COBOL for z/OS 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 6 リリース 3 (PTF インストール済み) での変更点

変更されたステートメント

- ランタイム APAR PH20569 (V2R2) および PH21261 (V2R3/V2R4): Enterprise COBOL V5 以降のバージョンでの DFSORT オプション NOBLKSET と従来のマージ方式のサポートを取得するための、MERGE ステートメントの新しいランタイム・オプション (IGZCOMPAT) が導入されています。(214 ページの『DFSORT オプション NOBLKSET の使用』)
- PH18641: 新しい「NAME is OMITTED」句が JSON GENERATE ステートメントに追加され、最上位の親の名前が生成されていない匿名 JSON オブジェクトを生成できるようになりました。
- PH20724: CALL ステートメントの USING 句を使用してサブプログラムに *file-name* を渡す方法が復元されました。(179 ページの『CALL ... USING ステートメントでの file-name の使用』)
- PH26789: 新しい「CONVERTING」句が JSON GENERATE ステートメントおよび JSON PARSE ステートメントに 追加され、JSON ブール値の生成と解析ができるようになりました。

  注 : この新機能を利用するプログラムがリンクされたり実行されたりするすべてのシステムに COBOL Runtime LE APAR PH26698 も適用する必要があります。
- PH30975: 新しい「when-phrase」と「generic-suppression-phrase」が JSON GENERATE ステートメントに追加され、JSON GENERATE の際に条件付きでデータ項目を抑止できるようになりました。

  注 : この新機能を利用するプログラムがリンクされたり実行されたりするすべてのシステムに COBOL Runtime LE APAR PH31172 も適用する必要があります。

変更された組み込み関数

- PH20997: UUID4 組み込み関数が導入されています。

  注 : この新機能を利用するプログラムがリンクされたり実行されたりするすべてのシステムに COBOL Runtime LE PTF UI66560(V2R2)/UI66555(V2R3)/UI66557(V2R4) も適用する必要があります。
- PH31047: ISO 8601 で指定された形式との間での日時情報のエンコードとデコードをサポートする新しい日時組み込み関数と、算術計算に適した整数との間での日時情報のエンコードとデコードをサポートする新しい日時組み込み関数が導入されています。

注 : これらの新しい日時組み込み関数を利用するプログラムがリンクされたり実行されたりするすべてのシステムに COBOL Runtime LE APAR PH31133 も適用する必要があります。

以下の組み込み関数が 2002 COBOL 標準の一部として追加されました。

- TEST-DATE-YYYYMMDD: TEST-DATE-YYYYMMDD 関数は、標準的な日付形式 (YYYYMMDD) の日付がグレゴリオ暦で有効な日付かどうかをテストします。
- TEST-DAY-YYYYDDD: TEST-DAY-YYYYDDD 関数は、年間通算日形式 (YYYYDDD) の日付がグレゴリオ暦で有効な日付かどうかをテストします。

以下の組み込み関数が 2014 COBOL 標準の一部として追加されました。

- COMBINED-DATETIME: COMBINED-DATETIME 関数は、整数の日付形式の日付と標準的な数値の時刻形式の時刻を組み合わせて、日付と時刻の両方のコンポーネントの派生元にできる、単一の数値項目にします。
- FORMATTED-CURRENT-DATE: FORMATTED-CURRENT-DATE 関数は、この関数が評価されるシステムによって提供される現在の日時を表す文字ストリングを戻します。
- FORMATTED-DATE: FORMATTED-DATE 関数は、日付を整数の日付形式から要求された形式に変換します。
- FORMATTED-DATETIME: FORMATTED-DATETIME 関数は、日付と時刻の組み合わせの形式を使用して、整数の日付形式の日付と午前 0 時からの経過秒数で表される数値の時刻を変換して組み合わせ、その日付と時刻の組み合わせの形式に応じた定様式の日時表記にします。
- FORMATTED-TIME: FORMATTED-TIME 関数は、1 つの形式を使用し、午前 0 時からの経過秒数を表す値を、要求された形式の定様式の時刻に変換します。
- INTEGER-OF-FORMATTED-DATE: INTEGER-OF-FORMATTED-DATE 関数は、指定された形式の日付を、整数の日付形式に変換します。
- SECONDS-FROM-FORMATTED-TIME: SECONDS-FROM-FORMATTED-TIME 関数は、指定された形式の時刻を、午前 0 時からの経過秒数を表す数値に変換します。
- SECONDS-PAST-MIDNIGHT: SECONDS-PAST-MIDNIGHT 関数は、この関数が評価されるシステムによって提供される現在の現地時間を表す標準的な数値の時刻形式の値を戻します。
- TEST-FORMATTED-DATETIME: TEST-FORMATTED-DATETIME 関数は、日付、時刻、または日時の組み合わせを表すデータ項目が有効かどうかを、指定された形式に従ってテストします。

新しいコンパイラー・オプションおよび変更されたコンパイラー・オプション

- PH22581: INITCHECK: INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスのうち 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。
- PH27536: NUMCHECK(ZON): NUMCHECK(ZON) オプションに新しいサブオプション LAXREDEF | STRICTREDEF が追加されました。このサブオプションは、再定義された項目をコンパイラーで検査してそれについての警告メッセージを出すかどうかを制御します。
- PH29542: NUMCHECK(BIN): NUMCHECK(BIN) オプションに新しいサブオプション TRUNCBIN | NOTRUNCBIN が追加されました。このサブオプションは、バイナリー・データ項目を検査するコードをコンパイラーで生成するかどうかを制御します。
- PTF UI71591 (APAR 番号なし): 数値の受信側に移動されているコンテンツの所有者である英数字の送信側を検査する新しい機能が NUMCHECK に追加されました。数値の受信側に移動されているコンテンツの所有者である英数字の送信側については、コンパイラーはこの送信側を整数の数値として扱うため、NUMCHECK は英数字の送信側ごとに暗黙的な数値のクラス・テストを生成します。
- ランタイム APAR PH29755(V2R3/V2R4) および PH30338(V2R3/V2R4 AMODE 64): TEST: DWARF 診断情報が含まれる LLA/VLF 管理プログラムのための新しいサポートが追加されました。
- PH33122: RULES: RULES オプションに新しいサブオプション LAXREDEF | NOLAXREDEF が追加されました。このサブオプションは、長さが一致しない再定義済みの項目をユーザーに通知します。

- PH34804: TUNE: 新しい TUNE オプションで、実行可能プログラムの最適化の対象となるアーキテクチャーが指定されるようになりました。
- PH35643: SOURCE: SOURCE オプションに新しいサブオプション DEC | HEX が追加されました。このサブオプションは、ソースのリストのシーケンス番号を 10 進形式と 16 進形式のどちらで生成するかを制御します。
- PH35652: OFFSET: OFFSET の動作が改善されました。COBOL コードの 1 行に対して複数の命令ブロックがある場合は、指定の COBOL ステートメントのオフセット・テーブル内に複数の項目が存在するようになります。
- PH37328: INVDATA: 非推奨の ZONEDATA コンパイラー・オプションが新しい INVDATA コンパイラー・オプションに置き換えられ、無効なデータが含まれた USAGE DISPLAY および USAGE PACKED-DECIMAL のデータ項目を処理するためのコードをコンパイラーで生成する方法を、ユーザーがきめ細かく制御できるようになりました。

移行支援

- ランタイム APAR PH25917: IGZUOPT モジュールに新しいオプション QSAMBUFFINITCHAR が追加され、QSAM バッファーの初期化に使用される先頭文字を制御できるようになりました。(353 ページの『付録 L QSAM バッファーの初期化の制御』)

インストール済み環境のカスタマイズの変更

- PH37331: COBOL カスタマイズ・マクロ内で誤ってコーディングされているオプションや OPTION= ではなく OPTION() としてコーディングされているオプションの診断のためのサポートが追加されました。

IBM Enterprise COBOL for z/OS バージョン 6 リリース 3 での変更点

コンパイラー・オプションの変更

- 新しいコンパイラー・オプションは以下のとおりです。
  - LP: 新しい LP コンパイラー・オプションは、関連する言語機能を有効にして、AMODE 31 (31 ビット) または AMODE 64 (64 ビット) のどちらのプログラムを生成するかを指示するために使用できます。LP(32) がデフォルトです。
- 変更されたコンパイラー・オプション:
  - ARCH: ARCH(7) は現在受け入れられません。 新しい上位レベルの ARCH(13) が受け入れられます。ARCH(8) がデフォルトです。
  - NUMCHECK: NUMCHECK(MSG) と NUMCHECK(ABD) のどちらが有効であるかにかかわりなく、コンパイル時に無効なデータが見つかるとコンパイル時エラー・メッセージが生成され、検査は除去されます。

AMODE 64 サポート

Enterprise COBOL を使用して、AMODE 31 (31 ビット) または AMODE 64 (64 ビット) アプリケーションを開発することができます。 アプリケーションで 64 ビット環境をサポートするために、コードを適宜、適合させてください。

言語エレメントの変更

- 動的な長さの基本項目を指定するため、DYNAMIC LENGTH 節がサポートされています。 動的な長さの基本項目は、実行時に長さが変わる可能性のあるデータ項目です。 これは 2014 COBOL 標準の一部です。
- 新しい UTF-8 データ型を示すため、UTF-8 句が USAGE 節に追加されています。 UTF-8 文字データを示すピクチャー・シンボルの 'U' も追加されました。 新しい USAGE は、データの新規クラス (UTF-8) と、データの新規カテゴリー (UTF-8) を示します。

- POINTER-32 句が USAGE 節に追加されました。これを使用して、ポインター・データ項目またはデータ・ポインターを定義できます。 POINTER-32 は、LP コンパイラー・オプション設定に関係なく 4 バイトの基本データ項目であり、LP(32) と LP(64) の両方で使用できます。
- REPOSITORY 段落の FUNCTION 指定子 INTRINSIC では、語 FUNCTION を指定せずに使用できる組み込み関数名の宣言が可能です。 これは 2002 COBOL 標準の一部です。

リストの変更

リストの用語が次のように変更されています。

- `static map` は `initial heap storage map` に変更されました。
- `writeable static area (WSA)` は `storage` に変更されました。
- `WSA24` は `below the line storage` に変更されました。
- `automatic map` は `stack storage map` に変更されました。

共有ストレージ変更におけるコンパイラー・フェーズ

- 共有ストレージにコンパイラー・フェーズを配置するためのインストール・カスタマイズが除去されました。

CALL ... USING ステートメントでの *file-name* の使用

CALL ステートメントの USING 句を使ってプログラムが *file-name* をサブプログラムに渡すことはできなくなりました。

IBM Enterprise COBOL for z/OS バージョン 6 リリース 2 (PTF インストール済み) での変更点

新しいコンパイラー・オプションおよび変更されたコンパイラー・オプション

- 新しいコンパイラー・オプションは以下のとおりです。
  - PI91584: `COPYLOC`: 新規コンパイラー・オプション `COPYLOC` を使用すれば、ライブラリー・フェーズでコピー・メンバーを検索する追加ロケーションとして PDSE (または PDS) データ・セットあるいは z/OS UNIX ディレクトリーを追加できます。
  - PH05855: `INITIAL`: 新しい `INITIAL` コンパイラーによって、`IS INITIAL` 節を `PROGRAM-ID` 段落に追加することなく、また動的 `CALL` ステートメントも動的 `CANCEL` ステートメントも使用することなく、呼び出されるたびに初期値がデータ項目に入るプログラムを得ることができます。
  - PH37328: `INVDATA`: 現在非推奨の ZONEDATA コンパイラー・オプションが新しい `INVDATA` コンパイラー・オプションに置き換えられ、無効なデータが含まれた USAGE DISPLAY および USAGE PACKED-DECIMAL のデータ項目を処理するためのコードをコンパイラーで生成する方法を、ユーザーがきめ細かく制御できるようになりました。
- 変更されたコンパイラー・オプション:
  - PI90571: `ZONEDATA`: `ZONEDATA` オプションが更新されました。 このオプションは、無効な数字、無効な符号、または無効なゾーン・ビットを含んでいる可能性がある `USAGE DISPLAY` データ項目または `PACKED-DECIMAL` データ項目に対する `MOVE` ステートメント、比較、および計算の動作に作用するようになりました。
  - PI91585: `RULES`: 新規サブオプション `OMITODOMIN | NOOMITODOMIN` が `RULES` 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、*integer-1* (最小出現回数) なしで指定されているすべての `OCCURS DEPENDING ON` 節に関してコンパイラーが警告メッセージを発行するかどうかを制御します。
  - PI91586: `RULES`: 新規サブオプション `UNREF | NOUNREFALL | NOUNREFSOURCE` が `RULES` 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、コンパイラーが未参照データ項目を報告するか

どうかを制御したり、報告が行われるのがコピー・メンバーで宣言されていないデータ項目に関してのみ (NOUNREFSOURCE) なのかすべてのデータ項目に関して (NOUNREFALL) なのかを制御したりします。

- PI96135: NUMCHECK(PAC): 桁数が偶数であるパック 10 進数 (COMP-3) データ項目に関して、未使用ビットがゼロになっているかどうかが検査されるようになりました。
- PI98480: NUMCHECK(ZON): 新規サブオプション ALPHNUM | NOALPHNUM が NUMCHECK(ZON) 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、英数字データ項目、英数字リテラル、または英数字形象定数と比較されるゾーン 10 進数データ項目に対する暗黙的数値クラス・テスト用のコードをコンパイラーが生成するのかどうかを制御します。
- PH04369: RULES(NOEVENPACK) は、名前が DFH、DSN、EYU、または SQL で始まる偶数桁 PACKED-DECIMAL データ項目 (これらは CICS および Db2 用に、または CICS および Db2 によって生成されたデータ項目です) についてメッセージを出しません。
- PH04485: TEST： 新しいサブオプション DSNAME | NODSNAME が TEST|NOTEST(SEPARATE) オプションに追加されました。 このサブオプションは、外部ファイル名 (コンパイル中に使用される SYSDEBUG データ・セット名) がオブジェクト・プログラムに格納されるかどうかを制御します。
- PH08642: NUMCHECK: NUMCHECK オプションによって追加されていた冗長性検査が削除され、パフォーマンスが向上します。いくつかの検査をコンパイル時に実行できます。 NUMCHECK を指定することによって、コンパイラーは、ランタイムではなくコンパイル時にいくつかのメッセージを生成できるようになります。
- PH09225: INITCHECK: INITCHECK オプションを OPTIMIZE(0) と一緒に指定できます。
- PH11667: NUMCHECK(BIN): NUMCHECK(BIN) は、2 進数データ項目 (COMP、COMP-4、および USAGE BINARY) について、TRUNC(BIN) が有効な場合でも 検査します。
- PH24340: NUMCHECK(ZON): NUMCHECK(ZON) オプションに新しいサブオプション LAXREDEF | STRICTREDEF が追加されました。このサブオプションは、再定義された項目をコンパイラーで検査してそれについての警告メッセージを出すかどうかを制御します。
- PH24413: INITCHECK: INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。
- PH26794: NUMCHECK(BIN): NUMCHECK(BIN) オプションに新しいサブオプション TRUNCBIN | NOTRUNCBIN が追加されました。このサブオプションは、バイナリー・データ項目を検査するコードをコンパイラーで生成するかどうかを制御します。
- ランタイム APAR PH20569 (V2R2/V2R3/V2R4): TEST(NOSEPARATE) が有効な場合に組み込まれる DWARF 診断情報を、LLA/VLF 管理プログラムから抽出できるようになっています。

- 以下のコンパイラー・オプションは非推奨です。
  - PH7328: INVDATA: このコンパイラー・オプションは非推奨ですが、使用が許容されており、同等の形式の新しい INVDATA コンパイラー・オプションに自動的にマップされます。

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

- PI91584: 新規コンパイラー・オプション COPYLOC が導入されると、COPY ステートメントが更新されます。
- PI95081: 取得される動的ストレージの位置を制御するために新規 LOC(24|31) 句が ALLOCATE ステートメントに追加されました。 この句は、取得される動的ストレージの位置を判別するときに DATA コンパイラー・オプションの影響をオーバーライドします。
- PI97160: SET TO FALSE および WHEN SET TO FALSE が 2002 COBOL 標準の定義に従って導入されました。 これにより、PROCEDURE DIVISION における無効値を明示的に参照しなくてもすむようになります。

- ランタイム APAR PH20569 (V2R2) および PH21261 (V2R3/V2R4): Enterprise COBOL V5 以降のバージョンでの DFSORT オプション NOBLKSET と従来のマージ方式のサポートを取得するための、MERGE ステートメントの新しいランタイム・オプション (IGZCOMPAT) が導入されています。
- PH28546: 新しい「CONVERTING」句が JSON GENERATE ステートメントおよび JSON PARSE ステートメントに追加され、JSON ブール値の生成と解析ができるようになりました。
  この新機能を利用するプログラムがリンクされたり実行されたりするすべてのシステムに COBOL Runtime LE APAR PH26698 も適用する必要があることに注意してください。

IBM 提供 CICS 予約語テーブルの変更

- PI91589: 新しい COBOL ワードが IBM 提供の CICS 予約語テーブルに追加されました。

組み込み関数の機能拡張

- PI97434: 以下の組み込み関数を使用して国別データ項目を処理するためのサポートが追加されました。
  - REVERSE
  - ULENGTH
  - UPOS
  - USUBSTR
  - UWIDTH

  この PTF の前提条件として Language Environment (LE) APAR PI97224 (z/OS V2R1/V2R2) および APAR PI97712 (z/OS V2R3) 用の PTF が必要となります。 PI97434 用の PTF がインストールされたコンパイラーを使用する前に、COBOL プログラムが実行されるすべてのシステムで Language Environment (LE) に APAR PI97224 または APAR PI97712 用の PTF がインストールされていることを確認してください。

移行支援

- ランタイム APAR PH25917: IGZUOPT モジュールに新しいオプション QSAMBUFFINITCHAR が追加され、QSAM バッファーの初期化に使用される先頭文字を制御できるようになりました。(353 ページの『付録 L QSAM バッファーの初期化の制御』)

IBM Enterprise COBOL for z/OS バージョン 6 リリース 2 での変更点

新規コンパイラー・オプション、変更されたコンパイラー・オプション、および削除されたコンパイラー・オプション

- 新しいコンパイラー・オプション:
  - DEFINE
  - INITCHECK
  - INLINE
  - NUMCHECK
  - PARMCHECK
- 変更されたコンパイラー・オプション:
  - AFP: デフォルト値が AFP(NOVOLATILE) に変更されました。
  - ARCH: 新しい上位レベルの ARCH(12) が受け入れられました。 デフォルトは引き続き ARCH(7) です。
  - MAXPCF: V6 コンパイラー・キャパシティーの増加を反映して、デフォルト値が MAXPCF(100000) に変更されました。

  - NOSTGOPT: 以前のバージョンでは、NOSTGOPT が有効になっていてもデータ項目を OPT(2) で最適化できました。 このバージョンでは、OPT(2) が指定されていてもストレージやデータ項目が最適化されないように NOSTGOPT が変更されました。 これは特に WORKING-STORAGE の目印に役立ちます。
  - SSRANGE: 新しいサブオプション MSG | ABD および ZLEN | NOZLEN が SSRANGE コンパイラー・オプションに追加されました。 これらにより、以下がそれぞれ可能になります。
    - 単一実行における範囲外条件の追加報告に関する異常終了および継続処理に代わるメッセージ。
    - メッセージまたは異常終了なしに進行するためのゼロ長の参照変更。
  - TEST: デバッグ能力を保持しながらディスク上のプログラム・オブジェクト・サイズを制御するために新規サブオプション SEPARATE および NOSEPARATE が TEST コンパイラー・オプションに追加されました。 また、TEST(NODWARF)、TEST(SEPARATE)、NOTEST(DWARF,SOURCE) など、サブオプションの新しい組み合わせが TEST コンパイラー・オプションと NOTEST コンパイラー・オプションの両方でサポートされるようになりました。
- 以下のコンパイラー・オプションは削除されました。
  - ZONECHECK は非推奨ですが、互換性のために使用できるようにはなっています。 これは NUMCHECK(ZON) で置き換えられます。

新規ステートメント

- 新規 JSON PARSE ステートメントは JSON テキストを COBOL データ・フォーマットに変換します。

新しい特殊レジスターと変更された特殊レジスター

- 新しい JSON-STATUS 特殊レジスターは、JSON PARSE ステートメントが正常に実行されたか、または非例外状態が発生したかを示すために使用されます。
- また、JSON-CODE 特殊レジスターは、JSON PARSE ステートメントが正常に実行されたか、または例外状態が発生したかを示すために使用されます。

新しいディレクティブ

- 2002 COBOL 標準 で定義されているように、条件付きコンパイルをサポートするために以下の新しいコンパイラー・ディレクティブが追加されました。
  - DEFINE ディレクティブはコンパイル変数を定義/定義解除します。
  - EVALUATE ディレクティブは、コンパイル・グループに組み込むソース行を選択する分岐方式を提供します。
  - IF ディレクティブは、片方向または両方向の条件付きコンパイルを提供します。
- 新規 INLINE ディレクティブの指定により、コンパイラーは、ソース・プログラムにおいて PERFORM ステートメントによって参照されるプロシージャーをインライン化するかどうかを判断できます。

デバッグの変更

- TEST(SEPARATE) で、デバッグ能力を保持しながらモジュール・サイズを制御するためにデバッグ情報をサイド・ファイルに生成できるようになりました。

リストの変更

- Enterprise COBOL V5 より前の COBOL コンパイラーの場合と同様に、コンパイラー診断メッセージがリストの末尾に表示されるようになりました。
- MD5 シグニチャーがプログラム・オブジェクトとデバッグ・データに追加されたため、プログラムが再コンパイルされた場合でもデバッグ･データを実行可能ファイルと突き合わせることができるようになりました。
- PPA4 の終わりに 3 つの新規フィールドが追加されました。
  - WORKING-STORAGE における最初のユーザー定義データ項目のオフセット。

- WORKING-STORAGE におけるユーザー定義データ項目の合計長。
- 外部データ項目が存在するかどうかを示すビット。

IBM Enterprise COBOL for z/OS バージョン 6 リリース 1 (PTF インストール済み) での変更点

新規コンパイラー・オプション、変更されたコンパイラー・オプション、および削除されたコンパイラー・オプション

- 新しいコンパイラー・オプション:
  - PI68226: INITCHECK
  - PI71625: NUMCHECK
  - PI78089: PARMCHECK
  - PI77981: INLINE
  - PI96231: COPYLOC
- 変更されたコンパイラー・オプション:
  - PI68023 および PI81838: NOSTGOPT: NOSTGOPT コンパイラー・オプションのデフォルト動作が変更されました。
  - PI74933: SSRANGE: コンパイラーによる参照変更長の検査方法を制御する新しいサブオプション ABD および SSRANGE が、SSRANGE コンパイラー・オプションに追加されました。
  - PI98996: NUMCHECK(PAC): 桁数が偶数であるパック 10 進数 (COMP-3) データ項目に関して、未使用ビットがゼロになっているかどうかが検査されるようになりました。
  - PH01251: NUMCHECK(ZON): 新規サブオプション ALPHNUM | NOALPHNUM が NUMCHECK(ZON) 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、英数字データ項目、英数字リテラル、または英数字形象定数と比較されるゾーン 10 進数データ項目に対する暗黙的数値クラス・テスト用のコードをコンパイラーが生成するのかどうかを制御します。
  - PH13943: NUMCHECK(BIN): NUMCHECK(BIN) は、2 進数データ項目 (COMP、COMP-4、および USAGE BINARY) について、TRUNC(BIN) が有効な場合でも 検査します。
  - PH24414: INITCHECK: INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。
  - ランタイム APAR PH20569 (V2R2/V2R3/V2R4): TEST(NOSEPARATE) が有効な場合に組み込まれる DWARF 診断情報を、LLA/VLF 管理プログラムから抽出できるようになっています。
- 以下のコンパイラー・オプションは削除されました。
  - PI71625: ZONECHECK は非推奨ですが、互換性のために使用できるようにはなっています。これは NUMCHECK(ZON) で置き換えられます。

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

- PI71621: JSON GENERATE ステートメントが再設計され、改善されました。
- PI92944: 取得される動的ストレージの位置を制御するために新規 LOC(24|31) 句が ALLOCATE ステートメントに追加されました。 この句は、取得される動的ストレージの位置を判別するときに DATA コンパイラー・オプションの影響をオーバーライドします。
- PI96231: 新規コンパイラー・オプション COPYLOC が導入されると、COPY ステートメントが更新されます。

- ランタイム APAR PH20569 (V2R2) および PH21261 (V2R3/V2R4): Enterprise COBOL V5 以降のバージョンでの DFSORT オプション NOBLKSET と従来のマージ方式のサポートを取得するための、MERGE ステートメントの新しいランタイム・オプション (IGZCOMPAT) が導入されています。

移行支援

- ランタイム APAR PH25917: IGZUOPT モジュールに新しいオプション QSAMBUFFINITCHAR が追加され、QSAM バッファーの初期化に使用される先頭文字を制御できるようになりました。(353 ページの『付録 L QSAM バッファーの初期化の制御』)

IBM Enterprise COBOL for z/OS バージョン 6 リリース 1 の変更点

新しいオプションおよび変更されたオプション

- 新しいコンパイラー・オプション:
  - SUPPRESS
  - VSAMOPENFS
  - ZONECHECK
- 変更されたコンパイラー・オプション:
  - LANGUAGE
  - SSRANGE

削除されたオプション

LVLINFO インストール・オプションが除去されました。以前は LVLINFO であった場所にビルド・レベル情報が入り、LVLINFO の代わりに、SERVICE コンパイラー・オプションをユーザー・サービス・レベル情報に使用できます。

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 ALLOCATE ステートメントは動的ストレージを取得します。これに対し、新しい FREE ステートメントは、ALLOCATE ステートメントで以前に取得された動的ストレージを解放します。どちらのステートメントも、2002 COBOL 標準の一部です。
- 2002 COBOL 標準 の一部として、INITIALIZE ステートメントに以下の機能拡張が行われました。
  - 新しい FILLER 句が追加され、FILLER データ項目を INITIALIZE ステートメントで初期設定できるようになりました。
  - 新しい VALUE 句が追加され、基本データ項目を VALUE 節で指定されたリテラルに初期設定できるようになりました。
- 新しい JSON GENERATE ステートメントは、データを JSON 形式に変換します。

コンパイラー動作の変更

Enterprise COBOL V6 では、コンパイラーはプログラムが大きくない場合でも、それらのプログラムをコンパイルするために 2 GB 境界より上のストレージを使用して始動します。 つまり、z/OS MEMLIMIT パラメーターをゼロでない値に設定しなければならない可能性があります。 MEMLIMIT の z/OS デフォルトは 2 GB ですが、プログラムをコンパイルするときに MEMLIMIT の z/OS 設定が十分な大きさではない場合、コンパイラー・メッセージ `IGYCB7145-U Insufficient memory in the compiler to continue compilation` が返される可能性があります。 このエラー・メッセージが出された場合は、ジョブ・カードで `REGION=0M` および `MEMLIMIT=3G` を設定し、プログラムを再コンパイルしてください。 これが正常に行われた場合は、IEFUSI、SMFPRMxx、または SMFLIMxx で設定されたシステム MEMLIMIT デフォルトを、2 GB 以上に変更することを検討してください。

注 : SMFLIMxx PARMLIB メンバーは、z/OS V2.2 以降のバージョンにのみ用意されています。

デバッグの変更

WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。

JCL の変更

コンパイラー・メッセージの言語を指定するためには、コンパイル時に LANGUAGE コンパイラー・オプションを使用し、さらに、Language Environment ランタイム・オプション NATLANG を設定する必要があります。 コンパイル JCL で CEEOPTS DD を使用するようお勧めします。

IBM Enterprise COBOL for z/OS バージョン 5 リリース 2 (PTF インストール済み) での変更点

新規コンパイラー・オプション、変更されたコンパイラー・オプション、および削除されたコンパイラー・オプション

- 新しいコンパイラー・オプション:
  - PI40822: ZONECHECK
  - PI69197: INITCHECK
  - PI81006: NUMCHECK
  - PI85868: VSAMOPENFS
- 変更されたコンパイラー・オプション:
  - PI40853: ZONEDATA: 新しいサブオプション NOPFD が ZONEDATA コンパイラー・オプションに追加されました。ZONEDATA=NOPFD を使用すると、COBOL V4 で NUMPROC=NOPFD|PFD を使用した場合に COBOL V4 が実行するのと同じ方法で、ゾーン 10 進数データの比較を実行するコードをコンパイラーが生成できるようになります。
  - PI53044: SSRANGE: コンパイラーによる参照変更長の検査方法を制御する新しいサブオプション ZLEN および NOZLEN が、SSRANGE コンパイラー・オプションに追加されました。
  - PI86343: SSRANGE: コンパイラーによる参照変更長の検査方法を制御する新しいサブオプション ABD および SSRANGE が、SSRANGE コンパイラー・オプションに追加されました。
  - PI90458: ZONEDATA: 無効な数値、無効な符号、または無効なゾーン・ビットが含まれている可能性がある USAGE DISPLAY データ項目または PACKED-DECIMAL データ項目に関する MOVE ステートメント、比較、および計算の動作に影響するように ZONEDATA オプションが更新されました。
  - PI97835: NUMCHECK(PAC): 桁数が偶数であるパック 10 進数 (COMP-3) データ項目に関して、未使用ビットがゼロになっているかどうかが検査されるようになりました。
  - PH01241: NUMCHECK(ZON): 新規サブオプション ALPHNUM | NOALPHNUM が NUMCHECK(ZON) 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、英数字データ項目、英数字リテラル、または英数字形象定数と比較されるゾーン 10 進数データ項目に対する暗黙的数値クラス・テスト用のコードをコンパイラーが生成するのかどうかを制御します。
- 以下のコンパイラー・オプションは削除されました。
  - PI81006: ZONECHECK は推奨されなくなり、IGYCDOPT に指定できなくなりました。NUMCHECK=(ZON) を使用すると、ZONECHECK と同じ結果を得られます。

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

- 新しいコンパイラー・オプション:
  - 著作権
  - QUALIFY(COMPAT|EXTEND)
  - SERVICE
  - SQLIMS
  - VLR(COMPAT|STANDARD)
  - XMLPARSE(XMLSS|COMPAT)
  - ZONEDATA(PFD|MIG)
- 変更されたコンパイラー・オプション:
  - ARCH: ARCH(6) は現在受け入れられません。 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。
- 以下のコンパイラー・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 ステートメントを含むソース・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

- 新しいコンパイラー・オプション:
  - AFP(*VOLATILE* | *NOVOLATILE*)
  - ARCH(*n*)
  - DISPSIGN(*SEP* | *COMPAT*)
  - HGPR(*PRESERVE* | *NOPRESERVE*)
  - MAXPCF(*nnn*)
  - STGOPT | NOSTGOPT
- 変更されたコンパイラー・オプション:
  - 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ていません。
  - 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。TEST コンパイラー・オプションには、新たに SOURCE および NOSOURCE サブオプションが追加されました。
  - 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・オプションと相互排他的ではなくなり、コンパイラー終了規則が更新されました。
  - OPTIMIZE オプションが変更され、いくつかのレベルの最適化が可能になりました。 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 には、コンパイル時に言語環境プログラムが必要となっています。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、23 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 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。 これらはいずれも、現在サポートされていない Language Environment Prelinker または DFSMS Loader を使用しているためです。

- 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、249 ページの『第 22 章 IMS プログラムを Enterprise COBOL V5 または V6 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。
- 下線文字 (_) が、データ名やプログラム名などのユーザー定義語で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 (PTF インストール済み) での変更点

- 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、「Enterprise COBOL for z/OS 言語解説書」を参照してください。) このサポートにより、大量のデータを使用するプログラミングが可能になります。例として以下のようなものがあります。
  - Db2 BLOB および CLOB データ・タイプを使用する 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 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 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。このサポートによって、STRING、UNSTRING、および INSPECT などのステートメントでの国別グループの使用が容易になります。
  - 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 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 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と、パフォーマンスが改善されます。
  - 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。
  - 混合 COBOL-Java 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することができます。

  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 for z/OS Debug Tool Utilities and Advanced Functions for z/OS* 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。
  - 再生サポートを利用すると、アプリケーションの実行パスとデータ値を 記録して再生できます。
  - 自動モニター・サポート機能は、デバッグ中に現在のステートメントで 参照されている変数の値を表示します。
  - 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、TEST(NONE,. . .) を 指定してコンパイルされたプログラムに対しては、GOTO コマンドは サポートされていませんでした。.)

  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 for z/OS Debug Tool Utilities and Advanced Functions for z/OS* 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 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 コマンドを使用して起動できます。
  - 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 節が追加されました。これにより、数字リテラルは、PICTURE 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 以外のエントリー・ポイントのアドレスを入れることができ、これにより、C 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型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。

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 またはバージョン 6 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、メンバーを PDSE データ・セットにコピーし、PDSE データ・セットをロード・ライブラリーのために使用します。IBM は、RECFM=U、DSNTYPE=LIBRARY、 および VERSION=2 を指定して PDSE ロード・ライブラリーのデータ・セットを割り振り、その他のすべての属性をブランクのままにすることを推奨しています。 以下に ISPF の例を示します。

```
Record format . . . . U
Record length . . . .
Block size  . . . . .
Data set name type    LIBRARY
Data set version  . : 2
```

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。
- 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。

これら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。

Language Environment を Enterprise COBOL V5 または V6 および VS COBOL II プログラムとともに使用

VS COBOL II プログラムと Enterprise COBOL V5 または V6 プログラムを一緒に実行するときは、以下の項目が必要となります。

- 現行バージョンの IGZEBST が必要となります。
  - CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換える必要があります。

    注 : APAR PI33330 の PTF がインストールされた LE からの IGZEBST は、COBOL V5 または V6 プログラムがなくても任意の COBOL プログラム VS COBOL II 以降と一緒に使用することもできます。
  - 動的な CALL の対象の CICS プログラムで必要なのは、APAR PI25079 用の PTF を SCEERUN にインストールすることのみです。

    注 : 非 CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換えると、パフォーマンスが向上します。 これは必須ではありません。 COBOL V5 または V6 プログ

ラムから VS COBOL II プログラムを呼び出すために非 CICS で動的に呼び出されるプログラムに関しては、IGZEBST の問題はありません。

- 現行バージョンの CEEBETBL (Language Environment 外部テーブル) が必要となります。 COBOL V5 または V6 の新規オブジェクト・コードと先程バインドしたオブジェクト・コードを組み込むと、古いバージョンの CEEBETBL が間接的に組み込まれてしまう場合があります。

  バインドする CEEBETBL の長さが x'28' (または現行 SCEELKED ライブラリーにおける CEEBETBL の長さ) よりも短い場合、その CEEBETBL は古い CEEBETBL であり、置き換える必要があります。 そうしないと、ランタイム異常終了が発生したり、強制終了ランタイム・メッセージが発行されたりします。

  移行の一環として COBOL V5 または V6 で古いオブジェクト・コードを再バインドする場合は、不用意に CEEBETBL を入り口点にすることがないように注意して、古いオブジェクト・コードを組み込む前に CEEBETBL の現行コピーを明確に組み込むことをお勧めします。

これらの条件を理解していて、すべてを満たしている場合、31 ページの『第 5 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*」(http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf)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
- 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

詳しくは、xxxiii 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、IBM z® 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、および JSON 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 およびバージョン 6 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

5 ページの表 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: XML、 Java インターオペラビリティー、 OO、 統合 CICS 変換プログラム、 マルチスレッド、 Unicode、 JSON
			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
		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 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 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 COBOL 標準 の改訂、 以下のサポート: 言語 環境 デバッグ・ツール

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
	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、2002 COBOL 標準および 2014 COBOL 標準からの選択機能、構造化プログラミング、DBCS 各国語、CICS インターフェース向上、31 ビット・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。以下に例を示します。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 が必要です。 Enterprise COBOL for z/OS バージョン 5.1 および 5.2 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。 Enterprise COBOL for z/OS バージョン 6.1 および 6.2 では、最小レベルとして z/OS バージョン 2 リリース 1 が必要です。 Enterprise COBOL for z/OS バージョン 6.3 では、最小レベルとして z/OS バージョン 2 リリース 2 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。7 ページの表 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	Notes®	利点を備えていない製品		
		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	*
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。	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 2 GB 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	*
JSON サポート	Enterprise COBOL V6.1 では JSON テキストの生成のサポート Enterprise COBOL V6.2 以降では JSON テキストの生成および解析のサポート	X	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 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

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

利点	Notes®	利点を備えていない製品		
		OS/VS COBOL	VS COBOL II	IBM COBOL
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	
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	

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

利点	Notes®	利点を備えていない製品		
		OS/VS COBOL	VS COBOL II	IBM COBOL
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		
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		

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

利点	Notes®	利点を備えていない製品		
		OS/VS COBOL	VS COBOL II	IBM COBOL
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		
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 • IBM Developer for z/OS 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

表 5. *Enterprise COBOL* および *Language Environment* の利点 (続き)				
利点	**Notes®**	**利点を備えていない製品**		
		OS/VS COBOL	**VS COBOL II**	**IBM COBOL**
ランタイム・オプション	ABTERMENC および TERMTHDACT - エラー動作を制御できます。	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	
Enterprise COBOL バージョン 5 およびバージョン 6 のコンパイラー・オプション	Enterprise COBOL バージョン 5 およびバージョン 6 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。 それらの変更について詳しくは、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。	X	X	X
Enterprise COBOL バージョン 5 およびバージョン 6 のコンパイラー・オプション (続き)	• SQLIMS オプションと VLR オプションは、Enterprise COBOL V5.1 (サービス PTF 適用済み)、V5.2、および V6 で使用できます。 • SUPPRESS オプションおよび VSAMOPENFS オプションは Enterprise COBOL V6.1 および V6.2 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 および V6 に組み込まれています。 • ZONECHECK オプションは、Enterprise COBOL V5.1 (サービス PTF 適用済み) および V5.2 (サービス PTF 適用済み) で使用できます。 ZONECHECK は、V6.1 (APAR PI71625 用 PTF インストール済み) では推奨されなくなり、NUMCHECK で置き換えられています。 • ZONEDATA オプションは、Enterprise COBOL V5.1 (サービス PTF 適用済み)、V5.2 (サービス PTF 適用済み)、および V6 で使用できます。	X	X	X

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

利点	Notes®	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 および V6 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	Notes®	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 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	**Notes®**	**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 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

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 6 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 91 ページの『第 8 章 VS COBOL II ソース・プログラムのアップデート』
- 101 ページの『第 10 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

FLAGMIG オプションは、Enterprise COBOL の下でコンパイルを行うために変換しなければならないソース・ステートメントを識別する場合に役立ちます。 FLAGMIG は、CMPR2 オプションをサポートする Enterprise COBOL より前のコンパイラーで使用できます。 Enterprise COBOL V4.2 を既に使用している場合は、Enterprise COBOL V5 または V6 への移行に役立つ FLAGMIG4 オプション (現行サービスが適用された Enterprise COBOL V4.2 で使用可能) を使用することをお勧めします。

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、292 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)』、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。

Enterprise COBOL への移行に役立つ FLAGMIG オプションおよび CMPR2 オプションの使用法について詳しくは、106 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。

Enterprise COBOL V4.2 を既に使用していて Enterprise COBOL V5 または V6 への移行を望む場合は、FLAGMIG4 オプションを使用して、Enterprise COBOL V5 または V6 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ます。 詳細については、15 ページの『FLAGMIG4 コンパイラー・オプション』を参照してください。

FLAGMIG4 コンパイラー・オプション

FLAGMIG4 オプションは、Enterprise COBOL V5 または V6 へのマイグレーションに役立ちます。FLAGMIG4 は、Enterprise COBOL V4.2 (APAR PM93450 用 PTF インストール済み) で使用可能です。 また、APAR PI12240、PI26838、および PI58762 の PTF には FLAGMIG4 オプションに対する更新が含まれているため、これらの PTF もインストールすることをお勧めします。

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 または V6 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。

注 : COBOL V5 および V6 におけるソース・コードの変更が COBOL 言語機能で使用されることはほとんどないため、それらの変更は COBOL ユーザーの 99% には影響しません。

SOM ベースのオブジェクト指向 COBOL

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、139 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 Db2 コプロセッサー

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、243 ページの『第 21 章 Db2 コプロセッサー移行における考慮事項』を参照してください。

組み込みの CICS 変換プログラム

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・プログラムを組み込み 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、237 ページの『第 20 章 CICS 移行における考慮事項』を参照してください。

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、プログラムの変更時に一度に 1 つのプログラムずつアプリケーションを段階的に再コンパイルするか、または既存のすべてのアプリケーションを同時に Enterprise COBOL で再コンパイルします。 また、両方の戦略の一部のみを使用することにしてもかまいません。

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、54 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 またはバージョン 6 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、265 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、91 ページの『第 8 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、106 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、287 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、101 ページの『第 10 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 またはバージョン 6 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。

詳細については、147 ページの『第 12 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 またはバージョン 6 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。

詳細については、161 ページの『第 14 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 プログラムを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、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。

この手順が完了していない場合は、ここにある手順を実行する前にまず、「*Enterprise COBOL V4.2 Compiler and Runtime Migration Guide*」 (http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf) に記載されているランタイム移行作業をすべて完了しておいてください。

VS COBOL II プログラムと Enterprise COBOL V5 または V6 プログラムを一緒に実行するときは、以下の項目が必要となります。

- 現行バージョンの IGZEBST が必要となります。
  - CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換える必要があります。

    注 : APAR PI33330 の PTF がインストールされた LE からの IGZEBST は、COBOL V5 または V6 プログラムがなくても任意の COBOL プログラム VS COBOL II 以降と一緒に使用することもできます。
  - 動的な CALL の対象の CICS プログラムで必要なのは、APAR PI25079 用の PTF を SCEERUN にインストールすることのみです。

    注 : 非 CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換えると、パフォーマンスが向上します。 これは必須ではありません。 COBOL V5 または V6 プログラムから VS COBOL II プログラムを呼び出すために非 CICS で動的に呼び出されるプログラムに関しては、IGZEBST の問題はありません。
- 現行バージョンの CEEBETBL (Language Environment 外部テーブル) が必要となります。 COBOL V5 または V6 の新規オブジェクト・コードと先程バインドしたオブジェクト・コードを組み込むと、古いバージョンの CEEBETBL が間接的に組み込まれてしまう場合があります。

  バインドする CEEBETBL の長さが x'28' (または現行 SCEELKED ライブラリーにおける CEEBETBL の長さ) よりも短い場合、その CEEBETBL は古い CEEBETBL であり、置き換える必要があります。 そうしないと、ランタイム異常終了が発生したり、強制終了ランタイム・メッセージが発行されたりします。

  移行の一環として COBOL V5 または V6 で古いオブジェクト・コードを再バインドする場合は、不用意に CEEBETBL を入り口点にすることがないように注意して、古いオブジェクト・コードを組み込む前に CEEBETBL の現行コピーを明確に組み込むことをお勧めします。

Enterprise COBOL プログラムを既存のアプリケーションに追加する場合は、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 および V6 との違い

詳細については、223 ページの『第 18 章 Enterprise COBOL バージョン 5 またはバージョン 6 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 V5 または V6 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？

プログラムはサポート対象コンパイラー (現在サポートされているのは IBM Enterprise COBOL for z/OS のみ) でコンパイルし、サポート対象ランタイム・ライブラリー (z/OS のサポート対象バージョンに関しては 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。

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、16 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。NORES でコンパイルされた OS/VS COBOL プログラムおよび VS COBOL II プログラムにはコンパイラー・マイグレーションが必要です。

ほとんどのプログラムの場合、Enterprise COBOL V5 または V6 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 V5 または V6 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 V5 または V6 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 および V6 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、287 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 または V6 プログラムと相互運用される OS/VS COBOL プログラム
- Enterprise COBOL V5 または V6 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 または V6 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 または V6 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、287 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 および V6 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 および V6 プログラムの相互運用性

Enterprise COBOL V5 および V6 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 または V6 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 または V6 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 または V6 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・オプションを有効にして VS COBOL II でコンパイルされたプログラムは、Enterprise COBOL V5 または V6 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 または V6 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 または V6 プログラム・オブジェクトと相互運用できます。

OS/VS COBOL プログラムを持っているかどうかの確認方法

OS/VS COBOL プログラムを持っているかどうかを確認するには、以下を行います。

- Debug Tool ロード・モジュール・アナライザー 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。
- http://cbttape.org/cbtdowns.htm にある無料の COBOL アナライザーを使用して、ロード・ライブラリーをスキャンし、OS/VS COBOL プログラムを探します。 このアナライザーは、その Web ページ上では「*File # 321 COBOL Analyzer from Roland Schiradin & post processor*」 として示されています。
- 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。

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. COBOL 実行可能ファイル (ロード・モジュールまたはプログラム・オブジェクト) が PDS データ・セット (ロード・ライブラリーとも呼ばれる) 内にある場合は、 それらのメンバーを PDSE データ・セットにコピーし、PDSE データ・セットをロード・ライブラリーのために使用します。IBM は、RECFM=U、DSNTYPE=LIBRARY、 および VERSION=2 を指定して PDSE ロード・ライブラリーのデータ・セットを割り振り、その他のすべての属性をブランクのままにすることを推奨しています。 以下に ISPF の例を示します。

   ```
   Record format . . . . U
   Record length . . . .
   Block size  . . . . .
   Data set name type    LIBRARY
   Data set version  . : 2
   ```

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。
   - 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。

   VS COBOL II プログラムと Enterprise COBOL V5 または V6 プログラムを一緒に実行するときは、以下の項目が必要となります。

   - 現行バージョンの IGZEBST が必要となります。
     - CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換える必要があります。

       注 : APAR PI33330 の PTF がインストールされた LE からの IGZEBST は、COBOL V5 または V6 プログラムがなくても任意の COBOL プログラム VS COBOL II 以降と一緒に使用することもできます。
     - 動的な CALL の対象の CICS プログラムで必要なのは、APAR PI25079 用の PTF を SCEERUN にインストールすることのみです。

       注 : 非 CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換えると、パフォーマンスが向上します。 これは必須ではありません。 COBOL V5 または V6 プログラムから VS COBOL II プログラムを呼び出すために非 CICS で動的に呼び出されるプログラムに関しては、IGZEBST の問題はありません。
   - 現行バージョンの CEEBETBL (Language Environment 外部テーブル) が必要となります。 COBOL V5 または V6 の新規オブジェクト・コードと先程バインドしたオブジェクト・コードを組み込むと、古いバージョンの CEEBETBL が間接的に組み込まれてしまう場合があります。

     バインドする CEEBETBL の長さが x'28' (または現行 SCEELKED ライブラリーにおける CEEBETBL の長さ) よりも短い場合、その CEEBETBL は古い CEEBETBL であり、置き換える必要があります。

そうしないと、ランタイム異常終了が発生したり、強制終了ランタイム・メッセージが発行されたりします。

移行の一環として COBOL V5 または V6 で古いオブジェクト・コードを再バインドする場合は、不用意に CEEBETBL を入り口点にすることがないように注意して、古いオブジェクト・コードを組み込む前に CEEBETBL の現行コピーを明確に組み込むことをお勧めします。

3. IBM Software Product Compatibility Reports 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。
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。 COBOL V5 および V6 に必要な PTF を見つけるには、SMP/E の FIXCAT 機能を使用します。187 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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。
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。CICS において、CSD セットアップの変更と DFHRPL セットアップの変更を行うには、237 ページの『第 20 章 CICS 移行における考慮事項』を参照してください。 APAR のリストについては、187 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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 V6 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 SSRANGE、NUMCHECK、PARMCHECK、INITCHECK、および OPT(0) を使用して COBOL V6 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。 SSRANGE、NUMCHECK、PARMCHECK、または INITCHECK のいずれのエラーも発生せず、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を確認したら、NOSSRANGE、NONUMCHECK、NOPARMCHECK、および OPT(2) を使用して再コンパイルを行い、最終テストを実行して、アプリケーションを実動に移行します。 SSRANGE、NUMCHECK、および PARMCHECK を使用してテストを行う理由は、COBOL V6 で様々な結果を引き起こす無効な COBOL データがあることを一部のお客様が認識しているためです。 INITCHECK では、データが初期化されていないために移行問題も発生する可能性があるケースを (コンパイル時に) 検出できます。

    少し間をおいてから、SSRANGE、NUMCHECK、PARMCHECK、または INITCHECK のいずれのエラーも見つからなかった場合、将来的な移行のために、このステップをスキップすることが可能です。無効なデータ使用が行われていないと考えられます。 また、このステップは、初めてプログラムが V6 でコンパイルされるときにのみ推奨されます。COBOL V6 を使用してプログラムをコンパイルした場合は、今後のコンパイルで、SSRANGE、NUMCHECK、PARMCHECK、または INITCHECK を指定したリグレッション・テストをスキップできます。
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。

第 4 章 Enterprise COBOL V6 へのマイグレーションに関する推奨事項

Enterprise COBOL V5 および V6 へのマイグレーションは、以前の COBOL コンパイラー・マイグレーション (OS/VS COBOL から Enterprise COBOL へのマイグレーションを除く) より複雑であるため、Enterprise COBOL V5 または V6 へのマイグレーションを行う前に、このセクションをお読みになるようお勧めします。

リグレッション・テスト

Enterprise COBOL V6 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 SSRANGE、NUMCHECK、PARMCHECK、INITCHECK、および OPT(0) を使用して COBOL V6 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。SSRANGE、NUMCHECK、PARMCHECK、または INITCHECK のいずれのエラーも発生せず、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を確認したら、NOSSRANGE、NONUMCHECK、NOPARMCHECK、および OPT(2) を使用して再コンパイルを行い、最終テストを実行して、アプリケーションを実動に移行します。 SSRANGE、NUMCHECK、および PARMCHECK を使用してテストを行う理由は、COBOL V6 で様々な結果を引き起こす無効な COBOL データがあることを一部のお客様が認識しているためです。 INITCHECK では、データが初期化されていないために移行問題も発生する可能性があるケースを (コンパイル時に) 検出できます。

注 : この追加のテストは、Enterprise COBOL V5 または V6 で既にコンパイルされたプログラムに対して実行する必要はありません。

ベスト・プラクティス

通常の業務どおり Enterprise COBOL V5 および V6 を扱う代わりに、1 つまたは 2 つのプログラムを一度にマイグレーションするのではなく、アプリケーションそれぞれを完全にマイグレーションすることをチームに認識させることを検討してください。このようにすると、アプリケーションの中のすべてのプログラムがマイグレーションされるため、将来的な更新または修正でマイグレーションが行われることはありません。 より多くのプログラムが、Enterprise COBOL V5 および V6 での「Million Instructions Per Second (MIPS)」節約の利点を得られるため、長期的にはコストを節約できます。

共通問題と解決策

一部のお客様から、Enterprise COBOL V5 または V6 へのマイグレーション中の、ソース・コードを調べただけでは見つからない「無効な COBOL データ」による問題が報告されています。最もよくある問題は、以下のとおりです。

- 数値 USAGE DISPLAY データ項目における無効なデータ
- パラメーター/引数のサイズ不一致
- TRUNC(STD) または TRUNC(OPT) でコンパイルされたプログラムにおいて、データ定義で定義されている桁数より多い桁数を持つ値が入っている、過剰な数の 2 進データ項目
- 初期化されていないデータ項目、または最初に設定されずに使用されているデータ項目

このような問題をより簡単に識別するには、以下の解決策を検討してください。

- 常に RULES(NOEVENPACK) および DIAGTRUNC でコンパイルする。
- IBM Developer for z/OS 以降のバージョンの「互換性のための COBOL プログラムのスキャン」機能を使用してパラメーターおよび引数を確認する。詳しくは、COBOL プログラムの互換性のスキャンを参照してください。
- 初期コード変更および単体テストの場合は SSRANGE、NUMCHECK、PARMCHECK、INITCHECK、および OPT(0) でコンパイルを行う。

- 品質保証テストおよび実動の場合は NOSSRANGE、NONUMCHECK、NOPARMCHECK、および OPT(2) で再コンパイルする。

第 5 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。

OS/VS COBOL でコンパイルされたプログラム、または CMPR2 コンパイラー・オプションを使用した IBM COBOL による VS COBOL II でコンパイルされたプログラムでは、Enterprise COBOL V5 または V6 で再コンパイルを行う前にソースの変更が必要になります。 このようなプログラムを使用している場合、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。

注 : ご使用のプログラムが NOCMPR2 で、あるいは Enterprise COBOL V3 または V4 でコンパイルされている場合、それらを Enterprise COBOL V5 または V6 でコンパイルするためには、ソースの変更は (ほとんどの場合) 必要ありません。 このマニュアルの『プログラムのアップグレード』で、該当する章を参照してください。

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
- 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。 (「*Program Directory for Enterprise COBOL*」は、Enterprise COBOL for z/OS ライブラリーから入手してください。)必ず、Enterprise COBOL バージョン 5 またはバージョン 6 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。

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 ソース・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、Debug Tool に組み込まれています。

CCCA の詳細については、287 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 または V6 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。また、APAR PI12240、PI26838、および PI58762 の PTF には FLAGMIG4 オプションに対する更新が含まれているため、これらの PTF もインストールすることをお勧めします。

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 または V6 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。

注 : COBOL V5 および V6 におけるソース・コードの変更が COBOL 言語機能で使用されることはほとんどないため、それらの変更は COBOL ユーザーの 99% には影響しません。

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、287 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 (33 ページの『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、295 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。

無償でオープン・ソースの COBOL アナライザーは、使用されたコンパイラー、コンパイラー・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できます。 詳しくは、295 ページの『無料でオープン・ソースの COBOL アナライザー』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。

注 : 以下の警告メッセージが発行された後もプログラムを実行し続けるには、 TRAP (ON) ランタイム・オプションを指定して実行する必要があります。 詳しくは、「*z/OS Language Environment* プログラミング・リファレンス」の TRAP を参照してください。

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/V/S 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/V/S COBOL program "program-name")

IBM Application Discovery and Delivery Intelligence および IBM Rational® Asset Analyzer for System z® 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、291 ページの『IBM Application Discovery and Delivery Intelligence および IBM Rational Asset Analyzer for System z』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 (34 ページの『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。35 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	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。

35 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		
	数値 USAGE DISPLAY	5		

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		
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
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		

1. 詳細については、335 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。

36 ページの表 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。

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
3	7 〜 8	組織にとっての重要度が高い。 移行ツールを用いての移行処置が大きい。
	3 〜 6	組織にとっての重要度が中位。 移行ツールを用いての移行処置が中位。
	0 〜 3	組織にとっての重要度が低い。 移行ツールを用いての移行処置が小さい。
4	9 〜 10	組織にとっての重要度が高い。 移行処置が非常に大きい。
	7 〜 8	組織にとっての重要度が中位。 移行ツールを用いての移行処置が大きい。
	3 〜 6	組織にとっての重要度が低い。 移行ツールを用いての移行処置が中位。
5	9 〜 10	組織にとっての重要度が中位。 移行処置が非常に大きい。
	7 〜 8	組織にとっての重要度が低い。 移行ツールを用いての移行処置が大きい。
6	9 〜 10	組織にとっての重要度が低い。 移行処置が非常に大きい。

移行優先順位を決定するときには、以下の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。223 ページの『第 18 章 Enterprise COBOL バージョン 5 またはバージョン 6 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。

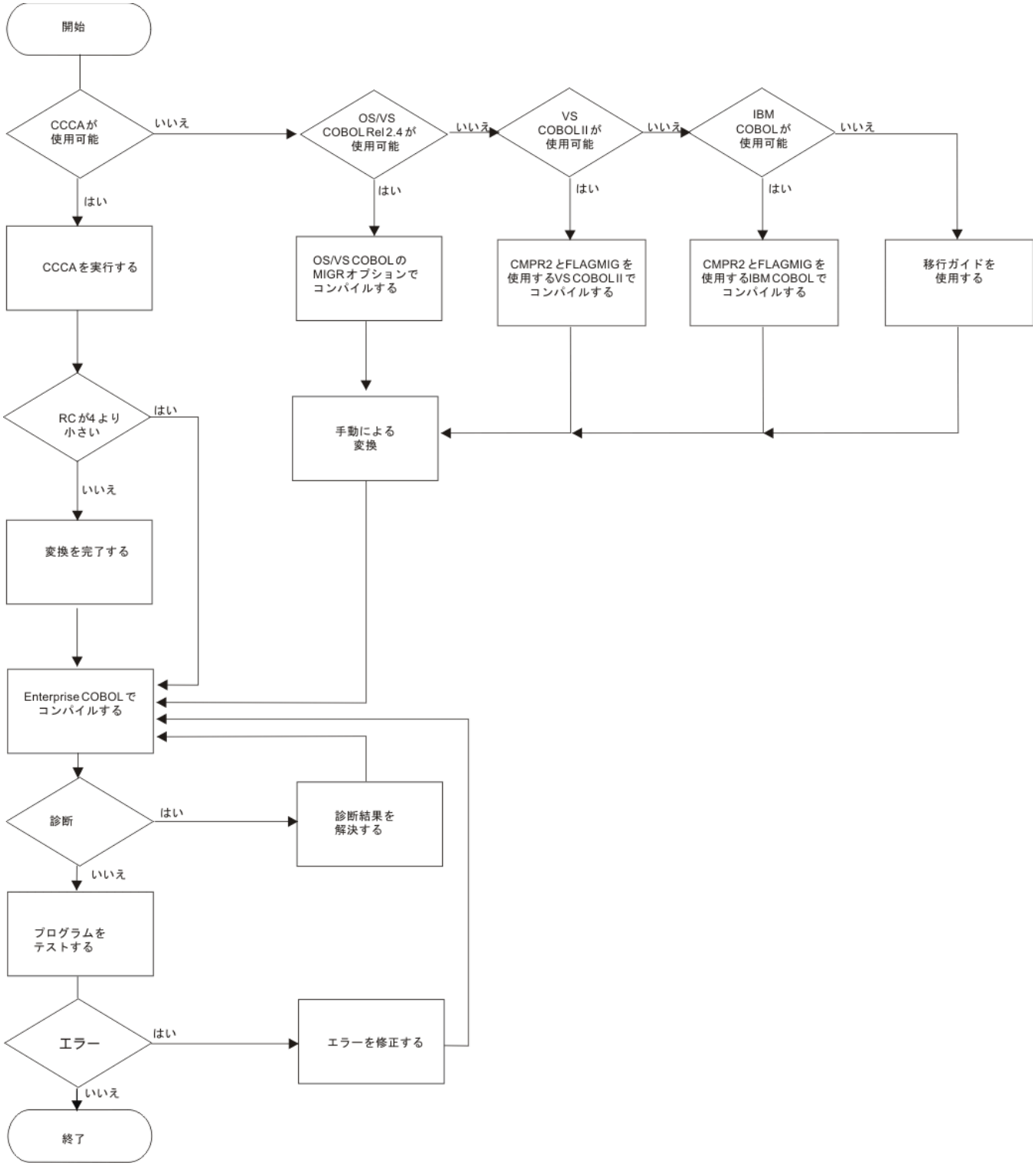


図 *1. OS/VS COBOL* プログラムを *Enterprise COBOL* プログラムに変換する手順

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。

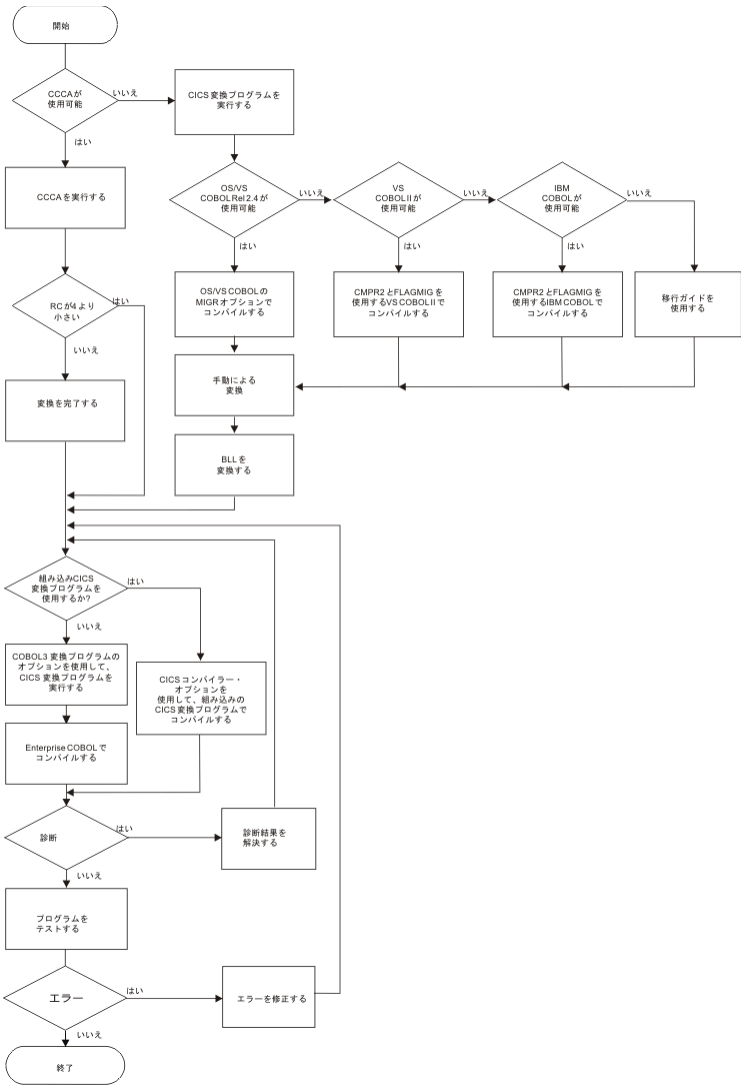


図 *2. CICS* コマンドを含む *OS/VS COBOL* プログラムを変換する手順

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。

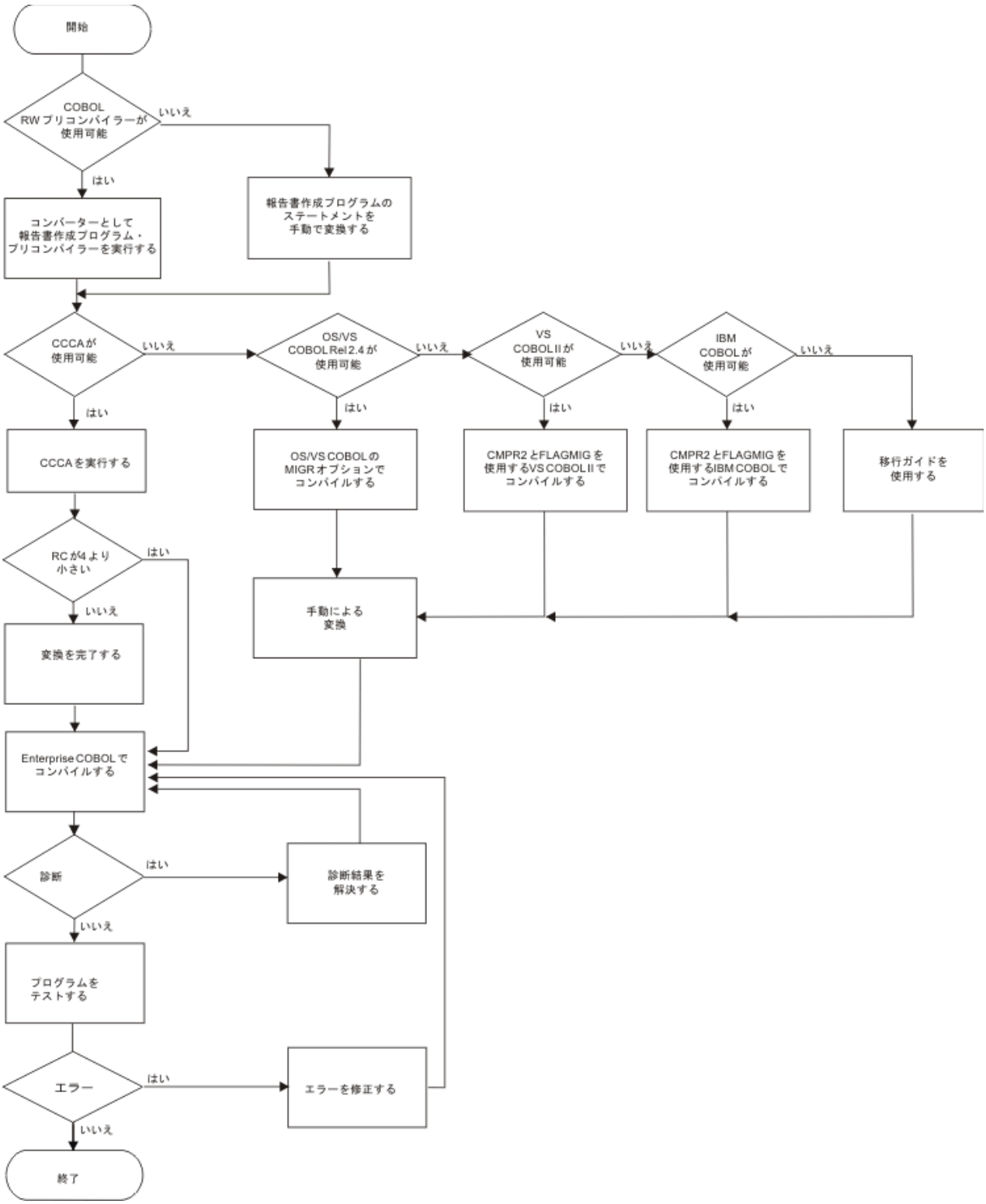


図 *3. OS/VS COBOL* プログラムに変換して、報告書作成プログラム・ステートメントを破棄する手順

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。

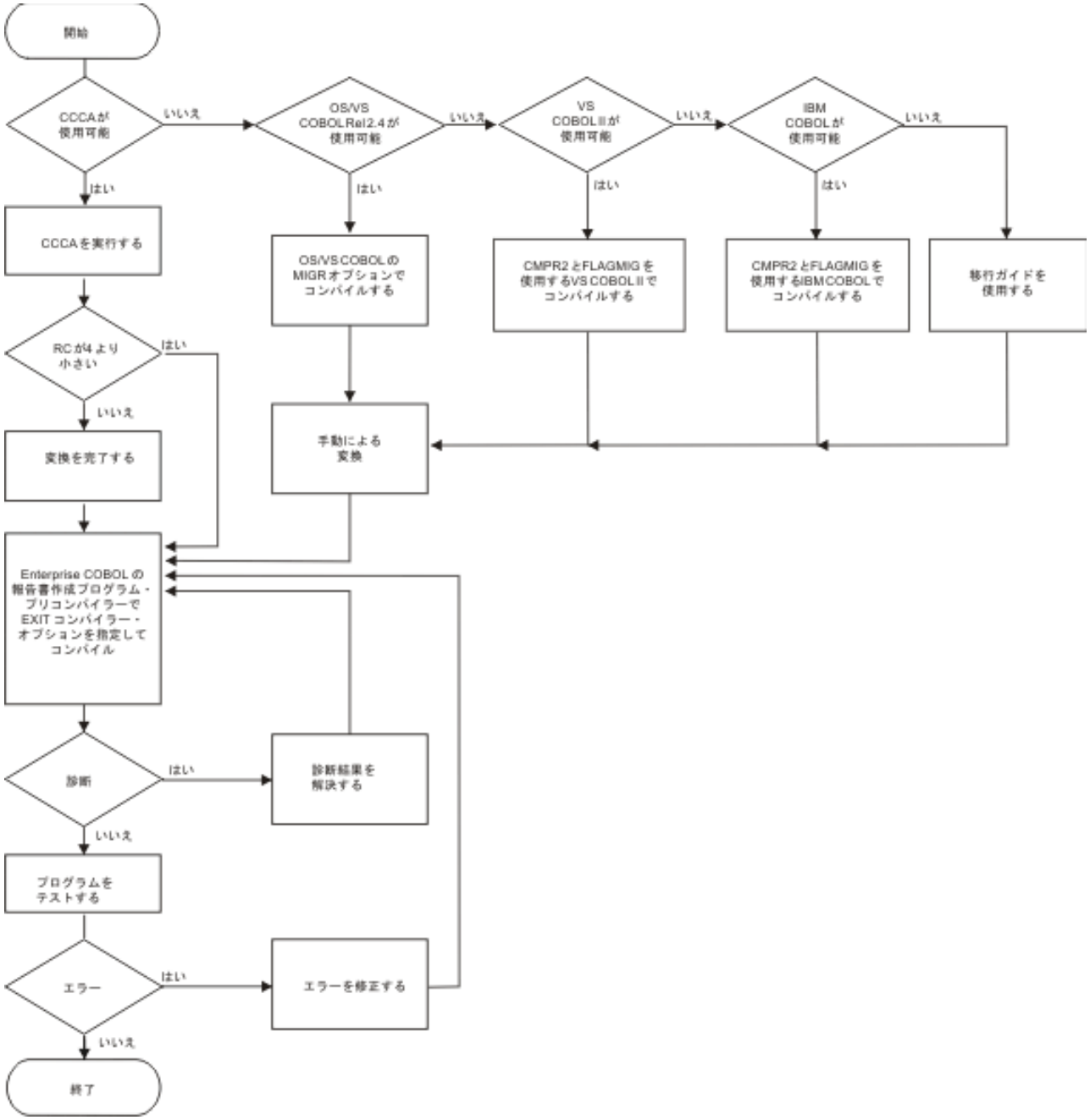


図 *4. OS/VS COBOL* プログラムに変換して、報告書作成プログラム・ステートメントを保持する手順

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、287 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。

   コンパイルとテストは 2 回ずつ行うようお勧めします。 まず最初に、SSRANGE、NUMCHECK、PARMCHECK、INITCHECK、および OPT(0) を使用してコンパイルを行い、テストが正常に完了した後で実動のために NOSSRANGE、NONUMCHECK、NOPARMCHECK、および OPT(2) を使用して再コンパイルを行います。

   また、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 または V6 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 または V6 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、335 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 6 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、ご使用の OS/VS COBOL プログラムを Enterprise COBOL にアップグレードするときに役立ちます。

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、265 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 *(SC43-3031)*

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 標準への参照は、IBM 完全版米国標準規格 COBOL バージョン 4 (プログラム 5734-CB2) によってサポートされる COBOL 言語への参照、または OS/VS COBOL (プログラム 5740-CB1) の LANGLVL(1) への参照です。
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)

48 ページの表 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		64 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		64 ページの『ACCEPT ステートメント』
ALPHABETIC クラスの変更	CCCA	71 ページの『ALPHABETIC クラスの変更 』
ALPHABET 節の変更 - ALPHABET キーワード	CCCA	71 ページの『ALPHABET-NAME 節の変更: ALPHABET キーワード』
区域 A、ピリオド	CCCA	67 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		71 ページの『算術ステートメントの変更 』
ASSIGN . . . OR	CCCA	58 ページの『ASSIGN . . . OR』
ASSIGN TO *integer system-name*	CCCA	58 ページの『ASSIGN . . . OR』
ASSIGN . . . FOR MULTIPLE REEL /UNIT	CCCA	58 ページの『ASSIGN . . . FOR MULTIPLE REEL/ UNIT 』
ASSIGN 節の変更 - *assignment-name* 形式	CCCA	71 ページの『ASSIGN 節の変更』
PICTURE 節内の B 記号 - 評価の変更		72 ページの『PICTURE 節内の B 記号 : 評価の変更』
BDAM ファイル処理	CCCA*	57 ページの『#unique_163/ unique_163_Connect_42_BDAMfile』
BLANK WHEN ZERO 節およびアスタリスク (*) のオーバーライド		64 ページの『BLANK WHEN ZERO 節およびアスタリスク (*) のオーバーライド 』
CALL identifier ステートメント - PICTURE 節内の B 記号		72 ページの『PICTURE 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		72 ページの『CALL ステートメントの変更』
CANCEL ステートメント - PICTURE 節内の B 記号		72 ページの『PICTURE 節内の B 記号 : 評価の変更』
CLOSE . . . FOR REMOVAL ステートメント		64 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	58 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	73 ページの『簡略複合比較条件の変更 』

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
グループと数値パック 10 進項目の比較		64 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	74 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		57 ページの『#unique_164/unique_164_Connect_42_comfeature』
CURRENCY-SIGN 節の変更 - 「/」、「=」、および「L」文字		74 ページの『CURRENCY-SIGN 節の変更 : 「/」、「=」、および「L」文字』
CURRENT-DATE 特殊レジスター	CCCA	58 ページの『CURRENT-DATE 特殊レジスター』
DIVIDE . . . ON SIZE ERROR - 中間結果の変更		80 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		75 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	59 ページの『EXAMINE ステートメント』
EXHIBIT ステートメント	CCCA	59 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		75 ページの『EXIT PROGRAM/GOBACK ステートメントの変更』
FILE STATUS 節の変更	CCCA	75 ページの『FILE STATUS 節の変更』
FILE-CONTROL 段落の FILE-LIMIT 文節	CCCA	60 ページの『FILE-CONTROL 段落の FILE-LIMIT 節』
制御のフロー (終了ステートメントなしの場合)		65 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	58 ページの『ASSIGN . . . FOR MULTIPLE REEL/UNIT』
USE AFTER STANDARD ERROR 宣言の GIVING 句	CCCA	60 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	77 ページの『IF . . . OTHERWISE ステートメントの変更』
索引名 - 固有でない		65 ページの『索引名』
INSPECT ステートメント - PROGRAM COLLATING SEQUENCE 節		81 ページの『PROGRAM COLLATING SEQUENCE 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
オプション・ワードとしての IS		80 ページの『オプション・ワード IS』
ISAM ファイル処理	CCCA	56 ページの『#unique_165/unique_165_Connect_42_isam』
JUSTIFIED 節の変更	CCCA	78 ページの『JUSTIFIED 節の変更』
LABEL 宣言		60 ページの『LABEL 宣言』
TOTALING/TOTALED AREA を持つ LABEL RECORDS 節	CCCA	60 ページの『TOTALING/TOTALED AREA 句を持つ LABEL RECORDS 節』
LABEL RECORD IS ステートメント		65 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		65 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		78 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	65 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		66 ページの『MOVE ステートメント - 複数の TO 指定』
MOVE ALL—TO PIC 99		66 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		66 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		80 ページの『ON SIZE ERROR 句 : 中間結果の変更』
固有でない Program-ID 名	CCCA	68 ページの『固有でない PROGRAM-ID 名』
NOTE ステートメント	CCCA	60 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		78 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		79 ページの『数値データの変更』
数字編集の変更 (PICTURE 節)		68 ページの『PICTURE ストリング』
OCCURS 節 (句の順序)		67 ページの『OCCURS 節』

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
OCCURS DEPENDING ON— ASCENDING 句および DESCENDING KEY 句		79 ページの『OCCURS DEPENDING ON 節 : ASCENDING および DESCENDING KEY 句』
OCCURS DEPENDING ON - 受け取り項目の値の変更	CCCA	79 ページの『OCCURS DEPENDING ON 節 : 受け取り項目の値の変更』
ON ステートメント	CCCA	60 ページの『ON ステートメント』
ON SIZE ERROR 句 - 中間結果の変更		80 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)		61 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)		61 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)』
LEAVE、REREAD、および DISP 句を指定した OPEN ステートメント	CCCA	61 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		67 ページの『OPEN REVERSED ステートメント』
OTHERWISE 節の変更		77 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		72 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		80 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		67 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		67 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	67 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	68 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		68 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 節 (数字編集の変更)		68 ページの『PICTURE ストリング』

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
PROGRAM COLLATING SEQUENCE 文節の変更		81 ページの『PROGRAM COLLATING SEQUENCE 節の変更』
固有でない Program-ID 名	CCCA	68 ページの『固有でない PROGRAM-ID 名』
修飾 - 同じ句の反復使用		68 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		68 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		81 ページの『READ および RETURN ステートメントの変更 : INTO 句』
READY TRACE および RESET TRACE ステートメント	CCCA	61 ページの『READY TRACE および RESET TRACE ステートメント』
RECORD CONTAINS n CHARACTERS 節		68 ページの『RECORD CONTAINS n CHARACTERS 節』
RECORD KEY 句および ALTERNATE RECORD KEY 句		68 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 節	CCCA	69 ページの『SD または FD 記入項目内の REDEFINES 節』
テーブルを指定した REDEFINES 節		69 ページの『テーブルを指定した REDEFINES 節』
比較条件	CCCA	69 ページの『比較条件』
REMARKS 段落	CCCA	62 ページの『REMARKS 段落』
RENAMES 節 - 固有でない非修飾データ名		69 ページの『RENAMES 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	55 ページの『#unique_166/unique_166_Connect_42_reportwr』
RERUN 節の変更		81 ページの『RERUN 節の変更』
RESERVE 節の変更	CCCA	81 ページの『RESERVE 節の変更』
予約語リストの変更	CCCA	81 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	82 ページの『SEARCH ステートメントの変更』

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
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		82 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		69 ページの『対応する FD のない SELECT ステートメント』
SELECT OPTIONAL 節の変更	CCCA	82 ページの『SELECT OPTIONAL 節の変更』
SORT 特殊レジスター		82 ページの『SORT 特殊レジスター』
SORT ステートメント		70 ページの『SORT ステートメント』
SORT または MERGE		70 ページの『SORT または MERGE』
ソース言語のデバッグの変更		83 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	62 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 節		81 ページの『PROGRAM COLLATING SEQUENCE 節の変更』
STRING ステートメント - 送り出しフィールド ID		70 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		83 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	62 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	62 ページの『TIME-OF-DAY 特殊レジスター』
LABEL RECORDS 節内の TOTALING/TOTALED AREA 句	CCCA	60 ページの『TOTALING/TOTALED AREA 句を持つ LABEL RECORDS 節』
TRANSFORM ステートメント	CCCA	63 ページの『TRANSFORM ステートメント』
UNSTRING ステートメント - PROGRAM COLLATING SEQUENCE 節		81 ページの『PROGRAM COLLATING SEQUENCE 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	70 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』

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
UNSTRING ステートメント - 複数の INTO 句		71 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		83 ページの『UNSTRING ステートメント : 添え字の評価の変更』
UPSI スイッチ	CCCA	84 ページの『UPSI スイッチ』
USE AFTER STANDARD ERROR - GIVING 句	CCCA	60 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	63 ページの『USE BEFORE STANDARD LABEL』
VALUE 節 - PICTURE 節に関連した符号付き値	CCCA	71 ページの『VALUE 節 - PICTURE 節に関連した符号付き値』
VALUE 節 - 条件名	CCCA	84 ページの『VALUE 節の条件名』
WHEN-COMPILED 特殊レジスター	CCCA	85 ページの『WHEN-COMPILED 特殊レジスター』
WRITE AFTER POSITIONING ステートメント	CCCA	85 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、287 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、292 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。

287 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。

- 55 ページの『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』
- 55 ページの『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』
- 56 ページの『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』
- 56 ページの『影響を受ける報告書作成プログラム言語項目』

注 : Enterprise COBOL V5.1 以降では、 コンパイラー・アーキテクチャーの変更により、COBOL 報告書作成プログラム・プリコンパイラー V1.6.01 以降が必要です。 COBOL 報告書作成プログラムに対する更新は、SPC Systems のサポート契約に従って、SPC Systems から入手する必要があります。 プログラム・サービスおよび技術サポートについては、SPC Systems (https://www.spc-systems.com) にお問い合わせください。

COBOL 報告書作成プログラム・プリコンパイラー製品について詳しくは、294 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。

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 または V6 とは混用できません。 Enterprise COBOL V5 または V6 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 または V6 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
英数字リテラル 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 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、287 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 節
編成パラメーター I
TRACK-AREA 節

START ステートメントの USING KEY 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、287 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 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 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、287 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 節を受け入れました。この節を 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* 節を受け入れました。この節を 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                         Enterprise COBOL

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

EXHIBIT CHANGED に対する修正処置

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                         Enterprise COBOL

WORKING-STORAGE SECTION.          WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).              77  DAT-1 PIC X(8).
77  DAT-2 PIC X(8).              77  DAT-2 PIC X(8).
                                  77  DAT1-CMP PIC X(8).
                                  77  DAT2-CMP PIC X(8).
      . . .                             . . .
 EXHIBIT CHANGED DAT-1 DAT-2      IF DAT-1 NOT EQUAL TO DAT1-CMP
                                    DISPLAY DAT-1
                                  END-IF
                                  IF DAT-2 NOT EQUAL TO DAT2-CMP
                                    DISPLAY DAT-2
                                  END-IF
                                  MOVE DAT-1 TO DAT1-CMP
                                  MOVE DAT-2 TO DAT2-CMP
```

EXHIBIT CHANGED NAMED に対する修正措置

EXHIBIT CHANGED NAMED ステートメントは、以下のように、IF および DISPLAY ステートメントで置き換えることができます。

1. データ項目の新しい値が以前の値と異なるかどうかを調べるために、IF ステートメントを指定します。
2. IF ステートメントの *statement-1* として DISPLAY ステートメントを指定します。

この変更により、新しい値が以前の値と異なるときにだけ、指定されたデータ項目の値が表示されます。

```
    OS/VS COBOL                          Enterprise COBOL

WORKING-STORAGE SECTION.          WORKING-STORAGE SECTION.
77  DAT-1 PIC X(8).              77  DAT-1    PIC X(8).
77  DAT-2 PIC X(8).              77  DAT-2    PIC X(8).
                                  77  DAT1-CMP PIC X(8).
                                  77  DAT2-CMP PIC X(8).
        . . .                           . . .
  EXHIBIT CHANGED NAMED            IF DAT-1 NOT EQUAL TO DAT1-CMP
      DAT-1 DAT-2                      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 節

OS/VS COBOL は、FILE-LIMIT 節を受け入れて、それをコメントとして扱いました。Enterprise COBOL はこの節を受け入れません。したがって、FILE-LIMIT 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 節を使用してください。FILE STATUS 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。

LABEL 宣言

Enterprise COBOL V5 以降、以下の LABEL 宣言はサポートされなくなりました。

- 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。

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 および V6 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。

TOTALING/TOTALED AREA 句を持つ LABEL RECORDS 節

OS/VS COBOL では、LABEL RECORDS 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 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、335 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。

例えば、次のように指定します。

```
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、101 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。例えば、次のように指定します。

```
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 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 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 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 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 節を使用できました。この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 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 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。

例えば、次のように指定します。

```
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 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 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。例えば、次のように指定します。

```
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。例えば、次のように指定します。

```
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。例:

```
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。例えば、次のように指定します。

```
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 節

OS/VS COBOL では、OCCURS 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。例えば、次のように指定します。

```
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 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 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 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 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。 例えば、次のように指定します。

```
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 節について詳しくは、*Enterprise COBOL for z/OS* 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 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 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 節を削除してください。

テーブルを指定した REDEFINES 節

OS/VS COBOL では、REDEFINES 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
< THAN	< または LESS THAN

RENAMES 節 - 固有でない非修飾データ名

OS/VS COBOL プログラム内の RENAMES 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 ステートメント

以前の保守レベルでは、OS/VS COBOL コンパイラーは SORT ステートメント内の 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。例えば、次のように指定します。

```
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 節 - PICTURE 節に関連した符号付き値

OS/VS COBOL では、PICTURE 節が無符号である場合に、VALUE 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 節のリテラルは PICTURE 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 節の変更

Enterprise COBOL は、以下の形式の ASSIGN 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 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 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。例えば、次のように指定します。

  ```
  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)。例えば、次のように指定します。

  ```
  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。例えば、次のように指定します。

```
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。例えば、次のように指定します。

```
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。例えば、次のように指定します。

```
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 節の変更 :「/」、「=」、および「L」文字

LANGLVL(1) を用いる OS/VS COBOL は、CURRENCY-SIGN 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 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 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、75 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、76 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 for z/OS* 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。

表 *10.* 状況キーの値 *: QSAM* ファイル *(*続き*)*

OS/VS	Enterprise COBOL	意味
92	42	OPEN モードでないファイルに対して CLOSE が試みられました。
92	43	最後の入出力ステートメントが READ でないときに REWRITE が試みられました。
92	44	異なるサイズのレコードで順次ファイル・レコードの再書き込みが試みられました。
92	46	有効な次のレコードがない状況で順次 READ が試みられました。
92	47	ファイルが OPEN INPUT または I-O モードでないときに READ が試みられました。
92	48	ファイルが OPEN OUTPUT、I-O、または EXTEND モードでないときに WRITE が試みられま した。
00	48	ファイルが OPEN I-O モードのときに WRITE が試みられました。
92	49	ファイルが OPEN I-O モードでないときに DELETE または REWRITE が試みられました。
92	92	論理エラー。

表 *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。

表 *11.* 状況キーの値 *: VSAM* ファイル *(*続き*)*

OS/VS	Enterprise COBOL	意味
30	30	永続エラー。
90	37	大容量記憶装置上にないファイルのオープンが試みられました。
90	90	その他のエラー (これ以上の情報はありません)。
91	91	VSAM パスワード障害。
92	41	OPEN モードのファイルに対して OPEN が試みられました。
92	42	OPEN モードでないファイルに対して CLOSE が試みられました。
92	43	最後の入出力ステートメントが READ または DELETE でないときに REWRITE が試みられました。
92	47	ファイルが OPEN INPUT または I-O モードでないときに READ が試みられました。
92	48	ファイルが OPEN OUTPUT、I-O、または EXTEND モードでないときに WRITE が試みられま した。
92	49	ファイルが OPEN I-O モードでないときに DELETE または REWRITE が試みられました。
93	93	VSAM リソースが使用可能ではありません。
93 96	35	非オプション・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 (VSAMOPENFS(COMPAT) (デフォルト) が有効になっている場合)	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。
	00 (VSAMOPENFS(SUCC) が有効になっている場合)	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 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 節と一緒 に JUSTIFIED 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 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。例えば、次のように指定します。

```

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。例えば、次のように指定します。

```

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、次の MOVE ステートメントが実行された後:

```
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。

Enterprise COBOL V5.2 以降では、ZONEDATA(MIG) (APAR PH31500 用の PTF がインストールされた Enterprise COBOL V6.2 では INVDATA(FORCENUMCMP) に置き換わっています)V5 または V6 を使用して、COBOL V5 または V6 へのマイグレーションを容易にすることができます。

INVDATA(FORCENUMCMP) オプションが有効な場合、コンパイラーは、ゾーン 10 進数データ項目内の各桁のゾーン・ビットを無視する数値比較を行うための命令を生成します。さらに、ゾーン 10 進数データ項目に無効なゾーン・ビットが含まれている場合、コンパイラーは、COBOL V4 とは異なる結果を生む可能性があることが分かっている最適化を実行しません。詳しくは、INVDATA を参照してください。

OCCURS DEPENDING ON 節 : ASCENDING および DESCENDING KEY 句

OS/VS COBOL は、OCCURS DEPENDING ON 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 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 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。

例えば、次のように指定します。

```
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。例えば、次のように指定します。

```
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。例えば、次のように指定します。

```
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 節の変更

OS/VS COBOL では、PROGRAM COLLATING SEQUENCE 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 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 節の変更

RERUN 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 節の変更

OS/VS COBOL は、以下の形式の FILE CONTROL 段落 RESERVE 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 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 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 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。265 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 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 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 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 83 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。例えば、次のように指定します。

```
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                    Enterprise COBOL

SPECIAL-NAMES.                 SPECIAL-NAMES.
    UPSI-0 IS MNUPO                UPSI-0 IS MNUPO
                                    ON STATUS IS UPSI-0-ON
                                    OFF STATUS IS UPSI-0-OFF
        . . .                       . .  .
PROCEDURE DIVISION             PROCEDURE DIVISION
      . . .                         . . .
    IF UPSI-0 = 1 ...              IF UPSI-0-ON ...
    IF MNUPO = 0 ...               IF UPSI-0-OFF ...
```

VALUE 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 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。例えば、次のように指定します。

```
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 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 7 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

87 ページの表 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	コメント
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。 未使用のデータ項目を必要としない場合、または未使用のデータ項目が VOLATILE 節で定義されている場合にのみ、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 for z/OS* 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	コメント
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 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 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

88 ページの表 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、305 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 for z/OS* 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 8 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 NOCMPR2 プログラムへ 変換する必要があります。CMPR2/NOCMPR2 コンパイラー・オプションは、Enterprise COBOL ではサポートされていません。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、106 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、287 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 と比較して少し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 または V6 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 V6 における新しい予約語は ALLOCATE、DEFAULT、END-JSON、FREE、JSON、JSON-CODE、および JSON-STATUS 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、287 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、APAR PM86253 の PTF によって、Enterprise COBOL V5.1 の予約語変換用に更新されています。V5.2 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。V6.1 では、APAR PI55980 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 の各後続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、265 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しい予約語

コンパイラー	予約語
COBOL/370 V1R1	FUNCTION, PROCEDURE-POINTER
COBOL (MVS および VM 版) V1R2	CLASS-ID、 METACLASS、 RECURSIVE、 END-INVOKE、 METHOD、 REPOSITORY、 INHERITS、 METHOD-ID、 RETURNING、 INVOKE、 OBJECT、 SELF、 SUPER、 LOCAL-STORAGE、 OVERRIDE
COBOL (OS/390 および VM 版) V2R1	COBOL (MVS および VM 版) と同じ
COBOL (OS/390 および VM 版) V2R2	COMP-5、 COMPUTATIONAL-5、 EXEC、 END-EXEC、 SQL、 TYPE、 FACTORY
COBOL (OS/390 および VM 版) V2R2、PQ49375 適用済み	EXECUTE
Enterprise COBOL 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
Enterprise COBOL V6R1	ALLOCATE、DEFAULT、END-JSON、FREE、JSON、JSON-CODE
Enterprise COBOL V6R2	JSON-STATUS
Enterprise COBOL V6R3	BYTE-LENGTH、JAVA、LIMIT、POINTER-32、UTF-8

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、101 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 をコーディングする必要もあります。
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・プログラムを使用して、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・プログラムを使用して、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
		は、COBOL レコードの平均サイズで、厳密に *m* より小さくなります。
		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 9 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、106 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 for z/OS* 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

97 ページの表 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	**コメント**
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。 • Enterprise COBOL V5 および V6.1 では TEST オプションにサブオプション EJPD | NOEJPD および SOURCE | NOSOURCE 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。 • Enterprise COBOL V6.2 では、デバッグ機能を保持しながらディスク上のプログラム・オブジェクト・サイズを制御するために新規サブオプション SEPARATE および NOSEPARATE が TEST コンパイラー・オプションに追加されました。 また、TEST(NODWARF)、TEST(SEPARATE)、NOTEST(DWARF,SOURCE) など、サブオプションの新しい組み合わせが TEST コンパイラー・オプションと NOTEST コンパイラー・オプションの両方でサポートされるようになりました。 TEST オプションについて詳しくは、『TEST』(*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 でサポートされないコンパイラー・オプション

98 ページの表 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	**コメント**
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。

表 *18. Enterprise COBOL* でサポートされないコンパイラー・オプション *(*続き*)*

VS COBOL II コンパイラー・オプション	**コメント**
NUMPROC(MIG)	Enterprise COBOL V5 および V6 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 または V6 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。 NUMPROC(MIG) でコンパイルしたプログラムを Enterprise COBOL V6 にマイグレーションするには、NUMPROC(PFD) にマイグレーションするときに役立つ NUMCHECK コンパイラー・オプションの使用を検討してください。 1. プログラムを NUMCHECK(ZON,PAC) および NUMPROC(PFD) でコンパイルします。 2. 適切な大きさの入力データで完全なリグレッション・テストを実行します。 アプリケーションが NUMCHECK メッセージも NUMCHECK 異常終了も引き起こさなければ、NUMPROC(PFD) および NONUMCHECK による実動用のコンパイルを安全に実行できます。 これは無効なデータの問題を解決するだけではありません。NUMPROC(PFD) は、NUMPROC コンパイラー・オプションのための最も有効な設定です。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。
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 または NORES が検出されると、Enterprise COBOL はエラー・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 10 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、106 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詳しくは、240 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、変更の影響を受ける可能性があります。
  - 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、適宜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 をコーディングする必要もあります。
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・プログラムを使用して、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・プログラムを使用して、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
		は、COBOL レコードの平均サイズで、厳密に *m* より小さくなります。
		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: `*****  STACK STORAGE  MAP******`

必要なストレージの量を減らすか、または境界より上のストレージを使用するようにランタイム・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。

Enterprise COBOL の新しい予約語

Enterprise COBOL は、IBM COBOL と比較して少し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 または V6 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 V6 における新しい予約語は ALLOCATE、DEFAULT、END-JSON、FREE、JSON、JSON-CODE、および JSON-STATUS 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、287 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、APAR PM86253 の PTF によって、Enterprise COBOL V5.1 の予約語変換用に更新されています。V5.2 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。V6.1 では、APAR PI55980 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 の各後続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、265 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しい予約語

コンパイラー	予約語
COBOL/370 V1R1	FUNCTION, PROCEDURE-POINTER
COBOL (MVS および VM 版) V1R2	CLASS-ID、 METACLASS、 RECURSIVE、 END-INVOKE、 METHOD、 REPOSITORY、 INHERITS、 METHOD-ID、 RETURNING、 INVOKE、 OBJECT、 SELF、 SUPER、 LOCAL-STORAGE、 OVERRIDE
COBOL (OS/390 および VM 版) V2R1	COBOL (MVS および VM 版) と同じ
COBOL (OS/390 および VM 版) V2R2	COMP-5、 COMPUTATIONAL-5、 EXEC、 END-EXEC、 SQL、 TYPE、 FACTORY
COBOL (OS/390 および VM 版) V2R2、PQ49375 適用済み	EXECUTE
Enterprise COBOL 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

表 20. コンパイラー別の新しい予約語 (続き)

コンパイラー	予約語
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
Enterprise COBOL V6R1	ALLOCATE、DEFAULT、END-JSON、FREE、JSON、JSON-CODE
Enterprise COBOL V6R2	JSON-STATUS
Enterprise COBOL V6R3	BYTE-LENGTH、JAVA、LIMIT、POINTER-32、UTF-8

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、101 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 節	108 ページの『SPECIAL-NAMES 段落の ALPHABET 節』
ALPHABETIC クラス	109 ページの『ALPHABETIC クラス』
CALL ... ON OVERFLOW	110 ページの『CALL . . . ON OVERFLOW』
位取り整数と非数字との比較	111 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	112 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	114 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	114 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	116 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	117 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)	118 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)	119 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	120 ページの『PERFORM から戻る方法』
PERFORM...VARYING...AFTER	122 ページの『PERFORM ... VARYING ... AFTER』
「A」および「B」を指定した PICTURE 節	123 ページの『「A」および「B」を指定した PICTURE 節』
PROGRAM COLLATING SEQUENCE	126 ページの『PROGRAM COLLATING SEQUENCE』
READ INTO と RETURN INTO	127 ページの『READ INTO と RETURN INTO』
RECORD CONTAINS n CHARACTERS	128 ページの『RECORD CONTAINS n CHARACTERS』
SET...TO TRUE	129 ページの『SET . . . TO TRUE』
SIZE ERROR、MULTIPLY と DIVIDE の	131 ページの『SIZE ERROR、MULTIPLY と DIVIDE の』
UNSTRING オペランドの評価	132 ページの『UNSTRING オペランドの評価』
UPSI スイッチ	137 ページの『UPSI スイッチ』
可変長グループ移動	138 ページの『可変長グループ移動』

SPECIAL-NAMES 段落の ALPHABET 節

ALPHABET が、ALPHABET 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 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。

例えば、次のように指定します。

```
SPECIAL-NAMES.
   ALPHA-NAME IS STANDARD-1.
```

NOCMPR2

ALPHABET 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。

例えば、次のように指定します。

```
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 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 節に対する修正処置:

キーワード ALPHABET を ALPHABET 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。

例えば、次のように指定します。

```
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。

例えば、次のように指定します。

```
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。

例えば、次のように指定します。

```
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 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、英数字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、115 ページの表 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 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、335 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 for z/OS* 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 (CEE3ABD 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、335 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。

表 *23. VSAM* ファイル定義に関する規則 (続き)

ファイル・タイプ	規則
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 に対する修正処置

FLAGMIG によってフラグが立てられた PERFORM...VARYING ステートメントは変換する必要があります。上記の *4* つの依存関係がすべて該当する PERFORM...VARYING ステートメントの変換方法 の 1 つを以下に示します。

```
PERFORM xx
   VARYING id-2 FROM id-3 BY id-4 UNTIL cond-1
     AFTER  id-5 FROM id-6 BY id-7 UNTIL cond-2
     AFTER id-8 FROM id-9 BY id-10 UNTIL cond-3.
```

これは、以下のように変換します。

```
MOVE id-3 TO id-2.
MOVE id-6 TO id-5
MOVE id-9 TO id-8

PERFORM UNTIL cond-1
  PERFORM UNTIL cond-2
    PERFORM UNTIL cond-3
      PERFORM xx
      ADD id-10 TO id-8
    END-PERFORM
    MOVE id-9 TO id-8
    ADD id-7 TO id-5
  END-PERFORM
  MOVE id-6 TO id-5
  ADD id-4 TO id-2
END-PERFORM
```

この例では、すべての id-x が ID であると仮定しています。いずれかが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 節

PICTURE 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 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 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 ステートメントについては、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」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 ステートメント

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 ステートメントに対する修正処置

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 ステートメント

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 ステートメントのこの位置に記号 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」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 ステートメントに対する修正処置

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* ステートメント

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 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 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」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 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 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 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 節で記述されている場合
- データ項目が BLANK WHEN ZERO 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」節の規則に従って実行されます。

JUSTIFIED 節

JUSTIFIED 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 節では、初期設定は JUSTIFIED 節による影響を受けません。つまり、VALUE 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 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 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 節

BLANK WHEN ZERO 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 節では、初期設定は BLANK WHEN ZERO 節によ

る影響を受けません。つまり、VALUE 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 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 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 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 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 for z/OS* 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。 その後で、*temp* は C および 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。

例えば、次のように指定します。

```
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。

例えば、次のように指定します。

```
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。

例えば、次のように指定します。

```
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。

例えば、次のように指定します。

```
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。影響を受けるステートメント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、データを移動するステートメントすべてに影響を与えます。影響のあるステートメントは次のとおりです。

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。例えば、次のように指定します。

```
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、139 ページの『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 140 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 節

- CLASS-ID 段落の INHERITS 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 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 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 節

- CLASS-ID 段落の METACLASS IS 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 節の METACLASS OF 節はサポートされません。

METHODS

- METHOD-ID 段落の OVERRIDE 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 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 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 11 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 for z/OS* 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。143 ページの表 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	コメント
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	コメント
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 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。
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	コメント
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。
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。 NUMPROC(MIG) でコンパイルしたプログラムを Enterprise COBOL V6 にマイグレーションするには、NUMPROC(PFD) にマイグレーションするときに役立つ NUMCHECK コンパイラー・オプションの使用を検討してください。 1. プログラムを NUMCHECK(ZON,PAC) および NUMPROC(PFD) でコンパイルします。 2. 適切な大きさの入力データで完全なリグレッション・テストを実行します。 アプリケーションが NUMCHECK メッセージも NUMCHECK 異常終了も引き起こさなければ、NUMPROC(PFD) および NONUMCHECK による実動用のコンパイルを安全に実行できます。 これは無効なデータの問題を解決するだけではありません。NUMPROC(PFD) は、NUMPROC コンパイラー・オプションのための最も有効な設定です。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。
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 12 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 またはバージョン 6 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、93 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、165 ページの『IBM Enterprise COBOL for z/OS バージョン 5 およびバージョン 6 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、変更の影響を受ける可能性があります。
  - 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、適宜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 およびバージョン 6 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 および V6 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 V5 または V6 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 V5.2 以降での 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、150 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 で XMLPARSE(XMLSS) を使用するように Enterprise COBOL バージョン 5 またはバージョン 6 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、345 ページの『付録 K 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
受け取り側は国別でしたが、文書に対して指定されたエンコードは 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 または V6 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。

- ALLOCATE
- DEFAULT
- END-JSON
- FREE
- JSON
- JSON-CODE
- JSON-STATUS
- 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、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、または APAR PI55980 の PTF が Enterprise COBOL バージョン 6.1 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、265 ページの表 50 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 をコーディングする必要もあります。
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・プログラムを使用して、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・プログラムを使用して、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
		は、COBOL レコードの平均サイズで、厳密に *m* より小さくなります。
		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 13 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、187 ページの『第 17 章 Enterprise COBOL バージョン 5 およびバージョン 6 での変更点』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	コメント
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 または V6 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。 NUMPROC(MIG) でコンパイルしたプログラムを Enterprise COBOL V6 にマイグレーションするには、NUMPROC(PFD) にマイグレーションするときに役立つ NUMCHECK コンパイラー・オプションの使用を検討してください。 1. プログラムを NUMCHECK(ZON,PAC) および NUMPROC(PFD) でコンパイルします。 2. 適切な大きさの入力データで完全なリグレッション・テストを実行します。 アプリケーションが NUMCHECK メッセージも NUMCHECK 異常終了も引き起こさなければ、NUMPROC(PFD) および NONUMCHECK による実動用のコンパイルを安全に実行できます。 これは無効なデータの問題を解決するだけではありません。NUMPROC(PFD) は、NUMPROC コンパイラー・オプションのための最も有効な設定です。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。

表 28. *Enterprise COBOL Version 6* では無効なコンパイラー・オプション

コンパイラー・オプション	コメント
LVLINFO	インストール・オプションが除去されました。 以前は LVLINFO であった場所にビルド・レベル情報が入り、LVLINFO の代わりに、SERVICE コンパイラー・オプションをユーザー・サービス・レベル情報に使用できます。

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 および V6 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、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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、305 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 およびバージョン 6 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 29. 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	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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))

表 29. 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
NONE	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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	診断メッセージが発行されます。 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	診断メッセージが発行されます。 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))
HOOK	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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	診断メッセージが発行されます。 オブジェクトにおいてフックは生成されません。	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	**Enterprise COBOL V5 および V6.1:** 診断メッセージが発行されます。 オブジェクト・プログラムでデバッグ情報が必ず生成されます。 **Enterprise COBOL V6.2:** TEST(SEPARATE) を指定すれば、生成された DWARF デバッグ情報はオブジェクト・プログラムではなく SYSDEBUG データ・セットに書き込まれます。	**Enterprise COBOL V5 および V6.1:** 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)) **Enterprise COBOL V6.2:** 診断メッセージは発行されません。
NOSEPARATE	**Enterprise COBOL V5 および V6.1:** 診断メッセージが発行されます。 オブジェクト・プログラムでデバッグ情報が必ず生成されます。 **Enterprise COBOL V6.2:** TEST(NOSEPARATE) を指定すれば、生成された DWARF デバッグ情報はオブジェクト・プログラムに書き込まれます。	**Enterprise COBOL V5 および V6.1:** 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)) **Enterprise COBOL V6.2:** 診断メッセージは発行されません。

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。

Enterprise COBOL バージョン 5 およびバージョン 6 におけるデバッグ情報の変更

Enterprise COBOL V5 または V6 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 およびバージョン 6 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。

TEST オプションの変更

Enterprise COBOL V5 および V6.1 では、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。

Enterprise COBOL V6.2 では、TEST(SEPARATE) オプションでプログラムをコンパイルすることにより、サイド・ファイルへのデバッグ情報の生成がサポートされます。

TEST の変更について詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『*TEST*』を参照してください。

リスト情報の変更

Enterprise COBOL V5 および V6.1 では、診断メッセージはリストの下部にありません。リストの診断メッセージ部分を表示するには、以下の手順に従ってください。

1. コマンド行に **F 'end of c'** と入力します (ヘッダー: End of compilation を見つけるには、ISPF **FIND** コマンドを使用します)。
2. Enter を押します。
3. (オプション) 「戻る (Page back)」を押します。

Enterprise COBOL V6.2 では、Enterprise COBOL V4 以前のコンパイラーと同様に、診断メッセージが再びリストの下部に示されるようになりました。

Enterprise COBOL V6 にのみ適用される変更

- WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。詳しくは、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。
- Enterprise COBOL V6 では、コンパイラー内でプロセス間通信 (IPC) メッセージ・キューが使用されます。そのため、cob2 を使用して z/OS UNIX でコンパイルを行っているときにコンパイラーで内部エラーが発生し、コンパイラーが KILL シグナルによって終了した場合、終了した時点で残っているメッセージ・キューを照会し、失効したメッセージ・キューを除去する必要があります。失効したメッセージ・キューは、以下の z/OS UNIX コマンドで削除できます。

  1. **ipcs -q** を入力し、キューのリストを表示します。
  2. ご使用のユーザー ID に関連付けられているキューを見つけます。
  3. **ipcrm -q** を入力し、キューを削除します。

  z/OS バッチでコンパイルを行っている場合、コンパイラー・エラーの後で、失効したメッセージ・キューを削除する必要はありません。
- Enterprise COBOL V6.3 における PPA1 の変更点

  Enterprise COBOL V6.3 以降、PPA1 の flag3 のビット 30 (オフセット X'1C') は、拡張フラグ・フィールドの存在を示すために設定することができます。 このビットを設定した場合、拡張フラグは、ベクトル・レジスター域がオプション領域にあることを示すビット 0 を持つことになります。 このことは、Language Environment インターフェースに従って PPA1 にアクセスするツールやプログラム・コードには影響しないはずです。 PPA1 の詳細については、*z/OS Language Environment Vendor Interfaces* を参照してください。

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 および V6 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、231 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 で加えられた Debug Tool の変更』を参照してください。

第 14 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 またはバージョン 6 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。

- 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、165 ページの『IBM Enterprise COBOL for z/OS バージョン 5 およびバージョン 6 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 または V6 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、93 ページの『新しい予約語』を参照してください。

Enterprise COBOL V5 または V6 への移行に役立つ新しいコンパイラー・オプション FLAGMIG4 があり、Enterprise COBOL V4.2 用の APAR PM93450 の PTF によって使用可能になります。 また、APAR PI12240、PI26838、および PI58762 の PTF には FLAGMIG4 オプションに対する更新が含まれているため、これらの PTF もインストールすることをお勧めします。 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 または V6 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。

注 : COBOL V5 および V6 におけるソース・コードの変更が COBOL 言語機能で使用されることはほとんどないため、それらの変更は COBOL ユーザーの 99% には影響しません。

ヒント : Enterprise COBOL V5 または V6 への移行をサポートする Enterprise COBOL V4 PTF を検討し、適用するようお勧めします。 詳細については、http://www.ibm.com/support/docview.wss?uid=swg21982146 を参照してください。

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 またはバージョン 6 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、COBOL V5.2 および V6 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 および V6 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、164 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 およびバージョン 6 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。詳しくは、150 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 N'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 または V6 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。

- ALLOCATE
- DEFAULT
- END-JSON
- FREE
- JSON
- JSON-CODE
- JSON-STATUS
- 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、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、または APAR PI55980 の PTF が Enterprise COBOL バージョン 6.1 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、265 ページの表 50 を参照してください。

IBM Enterprise COBOL for z/OS バージョン 5 およびバージョン 6 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 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 またはバージョン 6 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 15 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、187 ページの『第 17 章 Enterprise COBOL バージョン 5 およびバージョン 6 での変更点』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 *30. 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	コメント
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 または V6 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。 NUMPROC(MIG) でコンパイルしたプログラムを Enterprise COBOL V6 にマイグレーションするには、NUMPROC(PFD) にマイグレーションするときに役立つ NUMCHECK コンパイラー・オプションの使用を検討してください。 1. プログラムを NUMCHECK(ZON,PAC) および NUMPROC(PFD) でコンパイルします。 2. 適切な大きさの入力データで完全なリグレッション・テストを実行します。 アプリケーションが NUMCHECK メッセージも NUMCHECK 異常終了も引き起こさなければ、NUMPROC(PFD) および NONUMCHECK による実動用のコンパイルを安全に実行できます。 これは無効なデータの問題を解決するだけではありません。NUMPROC(PFD) は、NUMPROC コンパイラー・オプションのための最も有効な設定です。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。

表 31. *Enterprise COBOL Version 6* では無効なコンパイラー・オプション

コンパイラー・オプション	コメント
LVLINFO	インストール・オプションが除去されました。 以前は LVLINFO であった場所にビルド・レベル情報が入り、LVLINFO の代わりに、SERVICE コンパイラー・オプションをユーザー・サービス・レベル情報に使用できます。

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 およびバージョン 6 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、192 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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、305 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 およびバージョン 6 におけるデバッグ情報の変更

Enterprise COBOL V5 または V6 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 およびバージョン 6 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。

TEST オプションの変更

Enterprise COBOL V5 および V6.1 では、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。

Enterprise COBOL V6.2 では、TEST(SEPARATE) オプションでプログラムをコンパイルすることにより、サイド・ファイルへのデバッグ情報の生成がサポートされます。

TEST の変更について詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『*TEST*』を参照してください。

リスト情報の変更

Enterprise COBOL V5 および V6.1 では、診断メッセージはリストの下部にありません。リストの診断メッセージ部分を表示するには、以下の手順に従ってください。

1. コマンド行に **F 'end of c'** と入力します (ヘッダー: `End of compilation` を見つけるには、ISPF **FIND** コマンドを使用します)。
2. Enter を押します。
3. (オプション) 「戻る (Page back)」を押します。

Enterprise COBOL V6.2 では、Enterprise COBOL V4 以前のコンパイラーと同様に、診断メッセージが再びリストの下部に示されるようになりました。

Enterprise COBOL V6 にのみ適用される変更

- WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。詳しくは、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。
- Enterprise COBOL V6 では、コンパイラー内でプロセス間通信 (IPC) メッセージ・キューが使用されます。そのため、cob2 を使用して z/OS UNIX でコンパイルを行っているときにコンパイラーで内部エラーが発生し、コンパイラーが KILL シグナルによって終了した場合、終了した時点で残っているメッセージ・キューを照会し、失効したメッセージ・キューを除去する必要があります。失効したメッセージ・キューは、以下の z/OS UNIX コマンドで削除できます。

  1. **ipcs -q** を入力し、キューのリストを表示します。
  2. ご使用のユーザー ID に関連付けられているキューを見つけます。
  3. **ipcrm -q** を入力し、キューを削除します。

  z/OS バッチでコンパイルを行っている場合、コンパイラー・エラーの後で、失効したメッセージ・キューを削除する必要はありません。
- Enterprise COBOL V6.3 における PPA1 の変更点

  Enterprise COBOL V6.3 以降、PPA1 の flag3 のビット 30 (オフセット X'1C') は、拡張フラグ・フィールドの存在を示すために設定することができます。 このビットを設定した場合、拡張フラグは、ベクトル・レジスター域がオプション領域にあることを示すビット 0 を持つことになります。 このことは、Language Environment インターフェースに従って PPA1 にアクセスするツールやプログラム・コードには影響しないはずです。 PPA1 の詳細については、*z/OS Language Environment Vendor Interfaces* を参照してください。

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 および V6 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、231 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 で加えられた 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 およびバージョン 6 での新機能および相違点

Enterprise COBOL V5 および V6 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。

- V6 に特定の変更を調べるには、173 ページの『第 16 章 Enterprise COBOL バージョン 6 における変更』を参照してください。
- V5 および V6 における変更を調べるには、187 ページの『第 17 章 Enterprise COBOL バージョン 5 およびバージョン 6 での変更点』を参照してください。

第 16 章 Enterprise COBOL バージョン 6 における変更

Enterprise COBOL V6 に特定の変更について詳しくは、このセクションをお読みください。

V6 における変更は、主に以下のカテゴリーに分類されます。

- 前提ソフトウェア・レベルの変更
- COBOL ソース・コードの相違
- コンパイラー・オプションの変更
- 大きなプログラムにおけるシステム MEMLIMIT 設定への依存性
- ランタイムの変更
- ベンダー・ツールに影響する可能性がある変更

Enterprise COBOL バージョン 6 の前提条件ソフトウェア・レベル変更

Enterprise COBOL V6.1 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。

- IBM z/OS V2R1 以降
- IBM CICS Transaction Server for z/OS V4 以降
- IBM DB2 V10 以降
- IBM IMS V12 以降
- IBM Problem Determination Tools for z/OS V13 以降 (Debug Tool、Fault Analyzer、File Manager、Application Performance Analyzer)
- IBM Rational Developer for z Systems (RDz) V9.5.1.1 以降
- IBM Developer for z Systems (IDz) Enterprise Edition V14.0 以降

Enterprise COBOL V6.2 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。

- IBM z/OS V2R1 以降
- IBM CICS Transaction Server for z/OS V4.2 以降
- IBM DB2 V11 以降
- IBM IMS V13 以降
- IBM Problem Determination Tools for z/OS V13 以降 (Debug Tool、Fault Analyzer、File Manager、Application Performance Analyzer)
- IBM Developer for z/OS (IDz) Enterprise Edition V14.1 以降

注 : デバッグ情報が別のサイド・ファイルにあるプログラム・オブジェクトを (Enterprise COBOL V6.2 の新しい TEST(SEPARATE) オプションを使用して) デバッグするには、IBM Debug for z Systems V14.1 (5655-Q50) 以降、IBM Developer for z Systems V14.1 (5724-T07) 以降、または IBM Application Delivery Foundation for z Systems V3.1 (5655-AC6) 以降が必要です。

Enterprise COBOL V6.3 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。

- IBM z/OS V2R2 以降

  注 : AMODE 64 (64 ビット) の COBOL アプリケーションを実行するには、z/OS V2.3 (5650-ZOS) 以降が必要です。
- IBM CICS Transaction Server for z/OS V5.3 以降
- IBM DB2 V11 以降
- IBM IMS V14 以降

- IBM Application Delivery Foundation for z Systems V3.2 以降
- IBM Application Performance Analyzer for z/OS V14.1.8 以降
- IBM Fault Analyzer for z/OS V14.1.8 以降
- IBM File Manager for z/OS V14.1.8 以降
- IBM Developer for z/OS V14.2 以降
- IBM Debug for z/OS V14.2 以降

注 : 前提ソフトウェア・レベルは、Enterprise COBOL V6 リリースの一般出荷開始日の時点で有効でした。今後、一部の製品はサポート終了の予定を発表していくことになります。 現在サポートされているバージョンについては、ソフトウェア・ライフサイクルのページを確認してください。

Enterprise COBOL バージョン 6 における COBOL ソース・コードの相違

以下の相違点は、特に Enterprise COBOL V6 に該当します。

予約語

Enterprise COBOL V6.1 以降では、ALLOCATE、DEFAULT、END-JSON、FREE、JSON、および JSON-CODE が新しい予約語です。 これらの語をユーザー定義語 (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 V6 で S レベルの診断メッセージを受け取ります。 これらの予約語のインスタンスを ALLOCATE-X や JSON-Y 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の変更を自動的に行うことができます。

Enterprise COBOL V6.2 以降、JSON-STATUS が予約語になりました。JSON-STATUS 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 で S レベルの診断メッセージを受け取ります。JSON-STATUS のこれらのインスタンスは、JSON-STATUS-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 V6.3 から、BYTE-LENGTH、JAVA、LIMIT、POINTER-32、および UTF-8 が新しい予約語になりました。 これらの語をユーザー定義語 (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 V6.3 で S レベルの診断メッセージを受け取ります。これらの予約語のインスタンスを BYTE-LENGTH-X や BYTE-LENGTH-Y 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の変更を自動的に行うことができます。

VALUE 節

Enterprise COBOL V5 以前のバージョンでは、LINKAGE SECTION または FILE SECTION 内の 88 以外のレベルの VALUE 節は、コメントとして扱われ、無視されていました。

Enterprise COBOL V5 以前のバージョンでのコンパイルの例:

```
   000224          LINKAGE SECTION.
   000225            01 ALPH-ITEM  PIC X(4)  VALUE 1234.


==000225==> IGYDS1158-I A non-level-88 "VALUE" clause was found in the
"FILE SECTION" or "LINKAGE SECTION". The "VALUE" clause was treated as comments.
```

しかし、Enterprise COBOL V6.1 以降、LINKAGE SECTION 項目および FILE SECTION 項目の VALUE 節は構文検査の対象になり、意味を持つようになりました 。

Enterprise COBOL V6 でのコンパイル:

```
   000224          LINKAGE SECTION.
   000225            01 ALPH-ITEM  PIC X(4)  VALUE 1234.


==000225==> IGYGR1080-S A "VALUE" clause literal was not compatible with the data
category of the subject data item. The "VALUE" clause was discarded.
```

V6 では、以下のようになります。

- データ VALUE リテラルに PICTURE 節との互換性がない場合は、上記の例に示されているように、IGYGR1080-S エラー・メッセージが出されます。
- データ VALUE リテラルに PICTURE 節との互換性がある場合は、LINKAGE SECTION 内のデータ項目の初期化に使用されます。

要約すると、RC=0 でコンパイルされた、LINKAGE SECTION 内に 88 以外のレベルの VALUE 節がある COBOL V5 プログラムは、VALUE 節のリテラルの妥当性に応じて、COBOL V6 で RC=12 を受け取るか、LINKAGE データ項目が初期化されます。

CALL...USING *file-name* ステートメント

CALL ステートメントの USING 句を使用してサブプログラムに *file-name* を渡す方法は、Enterprise COBOL V6.3 で削除されましたが、Enterprise COBOL V6.3 に APAR PH20724 に対応する PTF をインストールすると復元されます。

Enterprise COBOL V6 におけるコンパイラー・オプションの変更

以下のオプションが追加されました。

表 *32. Enterprise COBOL* バージョン *6* 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	コメント
COPYLOC	Enterprise COBOL V6.1 (サービス PTF 適用済み)、V6.2 (サービス PTF 適用済み)、および V6.3 以降の新しいオプションです。 これを使用すれば、ライブラリー・フェーズでコピー・メンバーを検索する追加ロケーションとして PDSE (または PDS) データ・セットあるいは z/OS UNIX ディレクトリーを追加できます。
DEFINE	Enterprise COBOL V6.2 からの新しいオプションです。これは、PARAMETER 句を持つ DEFINE ディレクティブを使用して、プログラムに定義されているコンパイル変数にリテラル値を割り当てます。
INITCHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 未初期化データ項目を検査し、それらが初期化されていないまま使用されている場合に警告メッセージを出すかどうかを制御します。
INITIAL	Enterprise COBOL V6.2 (サービス PTF 適用済み)、および V6.3 以降の新しいオプションです。これにより、プログラムとそのネスト・プログラムは、IS INITIAL 節が PROGRAM-ID 段落に指定されたかのように動作します。
INLINE	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 これは、ソース・プログラム内の PERFORM ステートメントによって参照されるプロシージャーをインライン化するかどうかを、コンパイラーが考慮するように制御します。
LP	Enterprise COBOL V6.3 からの新しいオプションです。関連する言語機能を有効にして、AMODE 31 (31 ビット) または AMODE 64 (64 ビット) のどちらのプログラムを生成するかを指示するために使用できます。
NUMCHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 これは、送信データ項目として使用するゾーン 10 進データ項目およびパック 10 進データ項目に対して、暗黙的な数値のクラス・テストを生成するかどうか、また 2 進データ項目に対して SIZE ERROR 検査を生成するかどうかを制御します。

表 *32. Enterprise COBOL* バージョン *6* 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	コメント
PARMCHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 これは、WORKING-STORAGE にある最後の項目の後に追加のデータ項目を生成するよう、コンパイラーに指示します。 このバッファー・データ項目は、呼び出されたサブプログラムが WORKING-STORAGE の終わりを超えてデータを破損させたかどうかを検査するために、ランタイムに使用されます。
SUPPRESS	Enterprise COBOL V6.1 からの新しいオプションです。COPY ステートメントの SUPPRESS 句を無視するかどうかを制御します。
TUNE	Enterprise COBOL V6.3 (サービス PTF 適用済み) 以降の新しいオプションです。これは、実行可能プログラムの最適化の対象となるアーキテクチャーを指定します。 ARCH を指定する場合は、デフォルトの TUNE レベルと ARCH レベルが一致している必要があります。ARCH を指定しない場合は、ARCH と TUNE は両方ともデフォルトで 8 に設定されます。
VSAMOPENFS	Enterprise COBOL V6.1 からの新しいオプションです。ファイル整合性検査の確認を必要とする、正常に実行された VSAM OPEN ステートメントから報告されるユーザー・ファイル状況に作用します。

以下のオプションは変更されています。

表 *33. Enterprise COBOL* バージョン *6* 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	コメント
AFP	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。 • Enterprise COBOL V6.1 では、AFP(VOLATILE) がデフォルトです。 • Enterprise COBOL V6.2 から、AFP(NOVOLATILE) がデフォルトです。
ARCH	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 V6.1 では、ARCH(7) がデフォルトです。 • Enterprise COBOL V6.2 では、新しい上位レベルの ARCH(12) が受け入れられ、以前と同じく ARCH(7) がデフォルトです。 • Enterprise COBOL V6.3 では、ARCH(7) が削除され、新しい上位レベルの ARCH(13) が受け入れられます。 ARCH(8) がデフォルトです。
INITCHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、V6.2 (サービス PTF 適用済み)、および V6.3 (サービス PTF 適用済み) 以降では、INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。

表 33. *Enterprise COBOL* バージョン 6 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	コメント

LANGUAGE	COBOL V6 において、大文字英語または日本語のコンパイラー・メッセージに変更するためには、コンパイル時に LANGUAGE コンパイラー・オプションを使用し、さらに、Language Environment ランタイム・オプション NATLANG を設定する必要があります。 コンパイル JCL で CEEOPTS DD を使用するようお勧めします。
	例えば、メッセージを日本語に変更するためには、コンパイル時に LANGUAGE(JA) コンパイラー・オプションを使用し、さらに、NATLANG LE ランタイム・オプションを指定してください。
	`//CEEOPTS DD *`
	`NATLANG(JPN)`
	`/*`

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。 • Enterprise COBOL V6.1 では、MAXPCF(60000) がデフォルトです。 • Enterprise COBOL V6.2 以降では、MAXPCF(100000) がデフォルトです。
NOSTGOPT	Enterprise COBOL V6.1 では、NOSTGOPT が有効になっていてもデータ項目を OPT(2) で最適化できました。Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、NOSTGOPT が変更され、OPT(2) を指定してもストレージやデータ項目の最適化は行われなくなりました。これは特に WORKING-STORAGE の目印に役立ちます。
NUMCHECK	Enterprise COBOL V6.3 から、コンパイル時に無効なデータが見つかった場合、NUMCHECK(MSG) と NUMCHECK(ABD) のどちらが指定されていても、エラー・レベル・メッセージが生成され、検査は除去されます。
RULES	Enterprise COBOL V6.2 (サービス PTF 適用済み)、および V6.3 以降では、RULES コンパイラー・オプションに以下の新しいサブオプションが追加されました。 • OMITODOMIN | NOOMITODOMIN は、*integer-1* (最小出現回数) なしで指定されているすべての OCCURS DEPENDING ON 節に関して警告メッセージを発行するかどうかをコンパイラーに指示します。 • UNREF | NOUNREFALL | NOUNREFSOURCE は、未参照データ項目に対して警告メッセージを発行するのかどうかをコンパイラーに指示したり、報告が行われるのがコピー・メンバーで宣言されていないデータ項目に関してのみ (NOUNREFSOURCE) なのかすべてのデータ項目に関して (NOUNREFALL) なのかを制御するようにコンパイラーに指示したりします。 • LAXREDEF | NOLAXREDEF は、任意のレベルでデータ項目がさらに小さい項目に再定義される際に警告メッセージを出すかどうかをコンパイラーに指示します。
SOURCE	Enterprise COBOL V6.3 (サービス PTF 適用済み) 以降では、 SOURCE コンパイラー・オプションに新しいサブオプション DEC | HEX が追加されました。 SOURCE(DEC) が有効な場合、ソースのリストの行番号は 10 進形式になります。 SOURCE(HEX) が有効な場合は、ソースのリストの行番号は 16 進形式になります。
SSRANGE	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、SSRANGE コンパイラー・オプションに新しいサブオプション MSG | ABD が追加されました。このサブオプションは、範囲検査が失敗したときの COBOL プログラムのランタイム動作を制御します。

表 33. *Enterprise COBOL* バージョン 6 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	コメント
TEST	Enterprise COBOL V6.2 から、デバッグ機能を保持しながらディスク上のプログラム・オブジェクト・サイズを制御するために新規サブオプション SEPARATE | NOSEPARATE が TEST コンパイラー・オプションに追加されました。 また、TEST(NODWARF)、TEST(SEPARATE)、NOTEST(DWARF,SOURCE) など、サブオプションの新しい組み合わせが TEST コンパイラー・オプションと NOTEST コンパイラー・オプションの両方でサポートされるようになりました。

以下のオプションは削除されました。

表 34. *Enterprise COBOL Version 6* では無効なコンパイラー・オプション

コンパイラー・オプション	コメント
LVLINFO	Enterprise COBOL V6.1 から、LVLINFO インストール・オプションが除去されました。以前は LVLINFO であった場所にビルド・レベル情報が入り、LVLINFO の代わりに、SERVICE コンパイラー・オプションをユーザー・サービス・レベル情報に使用できます。
ZONECHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、ZONECHECK は非推奨ですが、互換性のために使用は許容されています。また、この代わりに NUMCHECK(ZON) が使用されるようになっています。

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、305 ページの『付録 E オプションの比較』を参照してください。

パフォーマンスに影響する可能性のあるコンパイラー・オプションの詳細なリストについては、「*Enterprise COBOL for z/OS* パフォーマンス・チューニング・ガイド」の『*V6* を最大限に活用するためのコンパイラー・オプションのチューニング方法』を参照してください。

すべてのコンパイラー・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」の『コンパイラー・オプション』を参照してください。

Enterprise COBOL バージョン 6 におけるコンパイルの変更

z/OS MEMLIMIT changes

Enterprise COBOL V6 では、コンパイラーはプログラムが大きくない場合でも、それらのプログラムをコンパイルするために 2 GB 境界より上のストレージを使用して始動します。 つまり、z/OS MEMLIMIT パラメーターをゼロでない値に設定しなければならない可能性があります。 MEMLIMIT の z/OS デフォルトは 2 GB ですが、プログラムをコンパイルするときに MEMLIMIT の z/OS 設定が十分な大きさではない場合、コンパイラー・メッセージ `IGYCB7145-U Insufficient memory in the compiler to continue compilation` が返される可能性があります。 このエラー・メッセージが出された場合は、ジョブ・カードで `REGION=0M` および `MEMLIMIT=3G` を設定し、プログラムを再コンパイルしてください。 これが正常に行われた場合は、IEFUSI、SMFPRMxx、または SMFLIMxx で設定されたシステム MEMLIMIT デフォルトを、2 GB 以上に変更することを検討してください。

注 : SMFLIMxx PARMLIB メンバーは、z/OS V2.2 以降のバージョンにのみ用意されています。

リストの変更

- Enterprise COBOL V6.1 以降、ビルド・レベル情報 (形式 PYYMMDD) が常に、リスト・ファイルのヘッダーに組み込まれます。この情報は、コンパイラーの保守レベルを調べるために役立ちます。 リスト・ヘッダーの例:

```
PP 5655-EC6 IBM Enterprise COBOL for z/OS 6.3.0 PXXXXXX
```

  Enterprise COBOL V5 および V6.1 では、診断メッセージはリストの中央部にあります。 Enterprise COBOL V6.2 以降のバージョンでは、Enterprise COBOL V4 以前のコンパイラーと同様に、診断メッセージがリストの下部に示されるようになりました。

- Enterprise COBOL V6.3 以降、リストの用語は次のように変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの `STATIC MAP` は `INITIAL HEAP STORAGE MAP` に変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの書き込み可能静的領域 (WSA) は `storage` に変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの `WSA24` は `BELOW THE LINE STORAGE` に変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの `AUTOMATIC MAP` は `STACK STORAGE MAP` に変更されました。

JCL の変更

COBOL V6 において、大文字英語または日本語のコンパイラー・メッセージに変更するためには、コンパイル時に LANGUAGE コンパイラー・オプションを使用し、さらに、Language Environment ランタイム・オプション NATLANG を設定する必要があります。 コンパイル JCL で CEEOPTS DD を使用するようお勧めします。

例えば、メッセージを日本語に変更するためには、コンパイル時に LANGUAGE(JA) コンパイラー・オプションを使用し、さらに、NATLANG LE ランタイム・オプションを指定してください。

```
//CEEOPTS DD *
          NATLANG(JPN)
/*
```

Enterprise COBOL V6.3 以降、COBOL AMODE 64 (64 ビット) プログラムの開発を支援するために、コンパイルを行うための新しいカタログ式プロシージャーが提供されています。 AMODE 64 サポートは Enterprise COBOL V6.3 に導入された新規フィーチャーです。 AMODE 64 サポートの詳細については、AMODE 64 プログラムの開発 (*Enterprise COBOL for z/OS* プログラミング・ガイド) を参照してください。

共有ストレージ変更におけるコンパイラー・フェーズ

Enterprise COBOL V6.3 以降、コンパイラー・フェーズを共有ストレージに入れるためのインストール・カスタマイズはなくなりました。 最新のシステムでは、多くの場合ユーザーが使用できるストレージが増えたため、コンパイラー・フェーズを共有ストレージに入れることによってストレージを節約する必要がないからです。 コンパイラーに対するこの変更の結果として、コンパイラー・フェーズを共有ストレージに入れるための言語はサポートされなくなりました。 したがって、コンパイラー・フェーズが共有ストレージの `IN` または `OUT` であることを指定する IGYCDOPT カスタマイズの保存済みコピーがある場合、その言語を除去してからでなければ IGYCDOPT をアセンブルできません。 コンパイラー・フェーズの `IN` または `OUT` を指定する IGYCDOPT 内のステートメントがない場合には、この変更によって影響を受けることはありません。

CALL ... USING ステートメントでの *file-name* の使用

CALL ステートメントの USING 句を使用してサブプログラムに *file-name* を渡す方法は、Enterprise COBOL V6.3 で削除されましたが、Enterprise COBOL V6.3 に APAR PH20724 に対応する PTF をインストールすると復元されます。

Enterprise COBOL バージョン 6 の実行時の変更

- 始動時に STORAGE ランタイム・オプションを使用して WORKING-STORAGE を選択した値に初期化できない場合があります。 以下のようなケースがあります。
  - スパン (RECORDING MODE S) ファイルを持つ COBOL V6 プログラム
  - DATA(31) でコンパイルされた非 CICS COBOL V5 プログラム
- V6 でのファイル状況の変更:
  - レコード・サイズが一致しない行順次ファイルに対する WRITE ステートメント

    以前のリリースの Enterprise COBOL では、レコード・サイズが一致しない行順次ファイルにレコードを書き込もうと試みた場合、ファイル状況 48 が誤って返されました。 これは Enterprise COBOL V6 で修正され、ファイル状況 44 が返されるようになりました。
  - UNIX ファイル属性が「書き込みのみ」である場合の、行順次ファイルに対する OPEN INPUT

    以前のリリースの Enterprise COBOL では、「書き込みのみ」属性を持つ行順次ファイル (DD PATHOPTS=(OWRONLY,...) を持つ z/OS UNIX ファイルや、書き込みアクセス権のみを持つ COBOL プログラムなど) に対する、INPUT 句を持つ OPEN ステートメントが、誤ってファイル状況 0 (正常) を返しました。 オープン・アクセス・モードをサポートしないファイルに対して OPEN ステートメントを試みた場合は、ファイル状況 37 が返されなければなりません。

    注 : この「書き込みのみ」は、行順次ファイルに適用可能ではない APPLY WRITE-ONLY 節を意味するわけではありません。 行順次ファイルは、z/OS UNIX ファイル・システムで作成されたファイルです。

    Enterprise COBOL V6 では、この OPEN ステートメントはファイル状況 37 で検出されます。
  - ファイル属性が一致しない VSAM ファイルに対する OPEN INPUT、I-O、EXTEND

    以前のリリースの Enterprise COBOL では、OPTIONAL と定義されていない VSAM ファイルに対して OPEN INPUT ステートメント、I-O ステートメント、または EXTEND ステートメントを試みたときにファイル属性の不一致が検出された場合、ファイル状況 35 が誤って返されました。 これは Enterprise COBOL V6 で修正され、ファイル状況 39 が返されるようになりました。

    注 :

    - OPEN OUTPUT における、また VSAM ファイルが OPTIONAL と定義されている場合の OPEN INPUT、I-O、および EXTEND における、同じようなファイル属性不一致条件は、既にファイル状況 39 として正しく報告されるように修正されています。
    - Enterprise COBOL V6.3 以降では、LP=64 コンパイラー・オプションを使用すると、コンパイル・プロセスには POSIX(ON) モードで実行されるコンポーネントが組み込まれます。 必然的に、このオプションを指定してコンパイラーを実行するユーザーごとに RACF® で OMVS セグメント (RACF 代替製品で同等のもの) が設定されていなければならないということになります。

ベンダー・ツールに影響する可能性がある、Enterprise COBOL バージョン 6 における変更

- WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。詳細については、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。
- Enterprise COBOL V6 では、コンパイラー内でプロセス間通信 (IPC) メッセージ・キューが使用されます。そのため、cob2 を使用して z/OS UNIX でコンパイルを行っているときにコンパイラーで内部エラーが発生し、コンパイラーが KILL シグナルによって終了した場合、終了した時点で残っているメッセージ・キューを照会し、失効したメッセージ・キューを除去する必要があります。失効したメッセージ・キューは、以下の z/OS UNIX コマンドで削除できます。

  1. **ipcs -q** を入力し、キューのリストを表示します。
  2. ご使用のユーザー ID に関連付けられているキューを見つけます。

3. **ipcrm -q** を入力し、キューを削除します。

z/OS バッチでコンパイルを行っている場合、コンパイラー・エラーの後で、失効したメッセージ・キューを削除する必要はありません。

- Enterprise COBOL V6.3 における PPA1 の変更点

  Enterprise COBOL V6.3 以降、PPA1 の flag3 のビット 30 (オフセット X'1C') は、拡張フラグ・フィールドの存在を示すために設定することができます。 このビットを設定した場合、拡張フラグは、ベクトル・レジスター域がオプション領域にあることを示すビット 0 を持つことになります。 このことは、Language Environment インターフェースに従って PPA1 にアクセスするツールやプログラム・コードには影響しないはずです。 PPA1 の詳細については、*z/OS Language Environment Vendor Interfaces* を参照してください。

WORKING-STORAGE SECTION の変更

WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。次の方法で Enterprise COBOL V5 および V6 プログラムにおいて実行時に WORKING-STORAGE を見つけることができます。

COBOL V5 および V6 において WORKING-STORAGE の開始を見つけるには、ダンプで PPA4 (Program Prologue Area 4) を見つける方法を知る必要があります。

A: AMODE 31 の場合

以下の説明は、プログラムを LP(32) でコンパイルした場合に当てはまります。

ダンプで PPA4 (Program Prolog Area 4) を見つける方法

1. ダンプにおいてトレースバックからプログラムの開始を見つけます。
2. 開始アドレス + x'0C' の位置にオフセット値があります。 これがプログラムの開始から PPA1 へのオフセットです。
3. 開始アドレス + PPA1 オフセット = PPA1。
4. ダンプ内のその位置に移動します。
5. PPA1 + x'04' の位置にオフセット値があります。 これがプログラムの開始から PPA2 へのオフセットです。
6. 開始アドレス + PPA2 オフセット = PPA2。
7. ダンプ内のその位置に移動します。
8. PPA2 + x'08' の位置にオフセット値があります。 これが PPA2 から PPA4 へのオフセットです。
9. PPA2 + PPA4 オフセット = PPA4。
10. ダンプ内のその位置に移動します。現在の位置が PPA4 です。

次に PPA4 のレイアウトについて知る必要があります。

PPA4 のレイアウト

PPA4 のレイアウト、および各 PPA4 のオフセット、長さ、説明について詳しくは、「*z/OS Language Environment Vendor Interfaces*」の『COBOL V5+ 32-bit PPA4 layout』を参照してください。

次に、いくつかの用語について知る必要があります。

知るべき用語

NORENT 静的領域
このストレージ域は、実行可能ファイルにおいて、NORENT でコンパイルされたプログラムごとに割り振られます。 NORENT プログラムの WORKING-STORAGE はここにあります。

LE の書き込み可能静的領域 (WSA)
COBOL V5/V6 プログラム・オブジェクト (実行可能ファイル) ごとに、このストレージ域があります。

RENT 静的領域
このストレージ域は、静的に実行可能ファイルにバインドされて RENT でコンパイルされたプログラムごとに WSA 内で割り振られます。 プログラムごとに独自の RENT 静的領域があります。 プログラムの WORKING-STORAGE はここに指定されることもあれば、ここには指定されないこともあります。

プログラム静的領域
このストレージ域は、特定の条件が満たされた場合にのみ WSA の外側で割り振られます。 その場合、プログラムの WORKING-STORAGE の場所は RENT 静的領域ではなくここです。

次に、WORKING-STORAGE が含まれている可能性がある場所が 3 つあることを理解する必要があります。

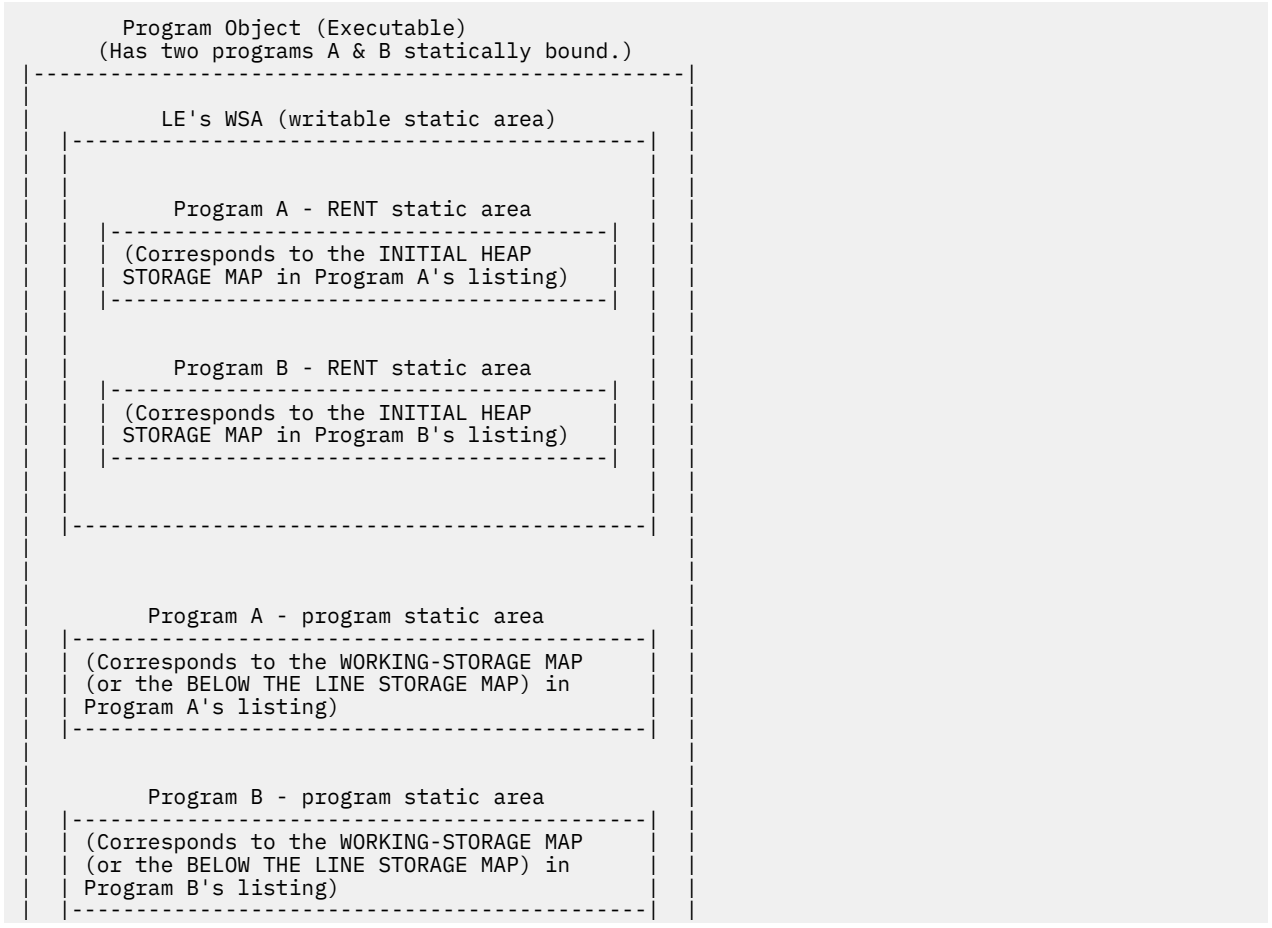
WORKING-STORAGE が含まれている可能性がある領域の説明

WORKING-STORAGE が含まれている可能性がある場所は 3 つあり、それぞれ別の場所です。

- プログラム・オブジェクト (実行可能ファイル) 内。 NORENT オプションでコンパイルされたすべてのプログラムには、実行可能プログラム内で予約された NORENT 静的領域があり、WORKING-STORAGE はここにあります。
- RENT オプションでコンパイルされたすべてのプログラムでは、RENT 静的領域は LE の WSA (書き込み可能静的領域) 内で割り振られます。 WORKING-STORAGE は、ここにある可能性があります。
- RENT 静的領域に置かれる代わりに、一部の COBOL V5 以降の RENT プログラムの WORKING-STORAGE は、LE の WSA の外で、プログラム静的領域と呼ばれる領域に割り振られます。

どこに WORKING-STORAGE が置かれるかを決定するための規則は、次のセクションにあります。

下の図は、その WORKING-STORAGE がプログラム静的領域にある RENT プログラム用に、どのようにストレージがレイアウトされているかを示しています。



```
|                                                |
|------------------------------------------------|
```

WORKING-STORAGE が含まれている可能性がある 3 つの領域を理解したら、プログラムの WORKING-STORAGE が実際はどこにあるのかを判別する方法を知る必要があります。 詳しくは、「*z/OS Language Environment Vendor Interfaces*」の『IGZXAPI』の 『*WORKING-STORAGE location*』および『*Layout of the Language Environment WSA, STATIC, PROGRAM STATIC, and User Working Storage*』を参照してください。

WORKING-STORAGE がある領域の判別方法

表 *35. WORKING-STORAGE* がある領域

COBOL のバージョン	**コンパイラー・オプション**	**WORKING-STORAGE がある場所**
COBOL V5	NORENT	プログラムの NORENT 静的領域
	RENT、DATA(31)	WSA 内におけるプログラムの RENT 静的領域
	RENT、DATA(24)。または RENT、WSOPT[1]	WSA の外におけるプログラムのプログラム静的領域
COBOL V6 以降のバージョン	NORENT	プログラムの NORENT 静的領域
	RENT、DATA(31)、および SPANNED RECORDS (つまり WSOPT ビットはオフ)[2]	WSA 内におけるプログラムの RENT 静的領域
	RENT、DATA(24) (つまり WSOPT ビットはオン)[2]	WSA の外におけるプログラムのプログラム静的領域
	RENT、DATA(31)、および NO SPANNED RECORDS (つまり WSOPT ビットはオン)[2]	WSA の外におけるプログラムのプログラム静的領域

注:

1. COBOL V5 には WSOPT コンパイラー・オプションがあります。 COBOL V6 には WSOPT コンパイラー・オプションはなくなりましたが、コンパイラーによって自動的に設定される WSOPT に関するシグニチャー情報ビットがあります。
2. SPANNED RECORDS の場合、WSOPT シグニチャー情報ビットはオフです。 NO SPANNED RECORDS の場合、WSOPT シグニチャー情報ビットはオンです。
   - 「RECORDING MODE」を求めてプログラムをスキャンし、「S」に設定されているすべてのファイルを探せば、SPANNED RECORDS が使用されているかどうかを判別できます。
   - また、リスト内のシグニチャー情報バイトを調べて WSOPT ビットを探すという方法もあります (シグニチャー・バイト 8、ビット 3)。 例えば、リストから次の情報を取得します。

     ```
     =X'001000000000'   INFO. BYTES 7-12
     ```

     バイト 8 は x'10' (b'00010000') です。 ビットの番号は 01234567 のように左から右に並びます。 ビット 3 がオンなので WSOPT はオンです。

WORKING-STORAGE がある領域が分かれば、WORKING-STORAGE を見つける方法が分かります。

WORKING-STORAGE をダンプ内で見つける方法

表 *36. PPA4*、*NORENT* 静的領域、*LE* の *WSA*、*RENT* 静的領域、プログラム静的領域をダンプ内で見つける方法

見つける対象	ダンプ内で見つける方法
PPA4	上記の説明を参照してください。
NORENT 静的領域	当該アドレスがストレージ内の <PPA4 + x'08'> にあります。
LE の WSA	当該アドレスがストレージ内の <CEECAA (または R12) + x'1F4'> にあります。 これはダンプ内で CEECAARENT と記述されています。
RENT 静的領域	当該アドレスが以下のストレージ内の場所にあります。 <ストレージ内の CEECAA (または R12) + x'1F4' にあるアドレス> <プログラムの PPA4 + x'0C' 内のオフセット>
プログラム静的領域	当該アドレスが以下のストレージ内の場所にあります。 <ストレージ内の CEECAA (または R12) + x'1F4’ にあるアドレス> <プログラムの PPA4 + x'0C' 内のオフセット> <プログラムの PPA4 + x'10' 内のオフセット>

この領域がダンプで見つかったら、それをコンパイル・リストと比較できます。

COBOL リストでは、以下のようになっています。

- INITIAL HEAP STORAGE MAP は RENT 静的領域または NORENT 静的領域のレイアウトを示します。
- WORKING-STORAGE MAP または BELOW THE LINE STORAGE MAP はプログラム静的領域のレイアウトを示します。

B: AMODE 64 の場合

以下の情報は、プログラムを LP(64) でコンパイルした場合に当てはまります。

WORKING-STORAGE SECTION に関する情報は、ヒープ・ストレージ・アドレス・テーブルとともに、プログラムの PPA4 にあります。 それらを見つけるには、以下の手順に従ってください。

1. ダンプにおいて、トレースバックからプログラムのエントリー・ポイント・アドレスを見つけます。 LE CEEDUMP トレースバックでは、これはプログラムの行に対応する "E Addr" 列の下のアドレスです。

   下記の例では、HELLO が COBOL プログラムです。 そのエントリー・ポイント・アドレスは X'260000A8' です。 このアドレスには、プログラムの最初の実行可能命令、すなわち STMG 命令が含まれるはずです。

```
Traceback:
  DSA     Entry        E  Offset    …
  1       CEEHDSP      +00003F3C
  2       CELQHROD     +00000266
  3       HELLO        +00000224
  4       CELQINIT     +00001D0C

  DSA     DSA Addr           E  Addr
  1       00000050082FBC60   0000000026B0A3D0
  2       00000050082FEDA0   0000000026B1DD18
  3       00000050082FEFA0   00000000260000A8
  4       00000050082FF220   0000000026903010
```

2. プログラム・エントリー・ポイントのオフセット -x'08' (つまりエントリー・ポイントの前) に、整数値があります。 この値は、エントリー・ポイント・アドレスから PPA1 へのオフセットです。
3. PPA1+x'04' に、オフセット値があります。 これは、エントリー・ポイント・アドレスから PPA2 へのオフセットです。
4. PPA2+x'08' に、オフセット値があります。 これは、PPA2 から PPA4 へのオフセットです。
5. PPA4+x'7C' に、オフセット値があります。 これは、プログラムの環境 から、ヒープ・ストレージ・アドレス・テーブルへのオフセットです。

ここで、環境 はプログラムの XPLINK 環境を指します。 これは、プログラムへの入り口にあるレジスター R5 内のアドレスです。 プログラムの最初の命令である STMG では、レジスターをスタックに格納します。 R5 のコンテンツはダンプに見つかります。

ヒープ・ストレージ・アドレス・テーブル

プログラムの環境 からのこのテーブルのオフセットは PPA4+X'7C' にあります。

LP(64) における WORKING-STORAGE SECTION 内のデータ項目は、デフォルトで 2 GB 境界より上に割り振られます。 それらは COBOL の ABOVE THE BAR HEAP にあります。 その開始アドレスは、ヒープ・ストレージ・アドレス・テーブル の最初のフィールド (このテーブルのオフセット X'00') にあります。 なお、このアドレスはコンパイル・リストの ABOVE THE BAR HEAP MAP セクションにも対応しています。 それは、WORKING-STORAGE SECTION 内のレベル 77 および 01 データ項目に関する情報を提供します。

ABOVE THE BAR HEAP に割り振られる COBOL 制御域とコンパイラー内部変数もあります。 プログラム内の最初の WORKING-STORAGE データ項目は、初めから存在しているとは限りません。 プログラムの WORKING-STORAGE SECTION 内の最初のデータ項目のオフセットは、PPA4 のオフセット +X'40' で見つけられます。

表 *37.* ヒープ・ストレージ・アドレス・テーブル

	長さ	**説明**
X'00'	8	COBOL の ABOVE THE BAR HEAP (64 ビット・ストレージ域) の開始アドレス
X'08'	8	予約済み
X'10'	8	予約済み

WOKRING-STORAGE SECTION に関連する情報は、以下のように要約できます。

表 *38. WORKING-STORAGE SECTION* のまとめ

説明	**見つけられる場所**
R5 からのヒープ・ストレージ・アドレス・テーブルのオフセット	PPA4+x'7C'
WORKING-STORAGE の開始アドレス	ヒープ・ストレージ・アドレス・テーブル + x'00'
WORKING-STORAGE からの最初のユーザー 64 ビット・データ項目のオフセット	PPA4+x'40'
すべてのユーザー WORKING-STORAGE 64 ビット・データ項目を含む領域の長さ	PPA4+x'48'

PPA4 のレイアウト、および各 PPA4 のオフセット、長さ、説明について詳しくは、「*z/OS Language Environment Vendor Interfaces*」の『COBOL 64-bit PPA4 layout』を参照してください。

C: LE ベンダー・インターフェース IGZXAPI を使用した WORKING-STORAGE アドレスの照会

COBOL 固有のベンダー・インターフェース・ルーチン IGZXAPI を使用して、WORKING-STORAGE アドレスを照会することもできます。 Enterprise COBOL V6.1 では、LE ベンダー・インターフェース IGZXAPI が新しい機能コード 8 で機能拡張され、COBOL プログラムの WORKING-STORAGE SECTION の長さとアドレスに関する情報を要求できるようになりました。 返されるアドレスは、COBOL コンパイラー・リストの INITIAL HEAP STORAGE MAP セクションに対応しています。PROGRAM-ID からのプログラム名やシグニチャー情報バイトといった他の情報 (コンパイラー・リストの「Compiler Options and Program Information Section」に対応) も返されます。

この機能拡張は、COBOL Runtime LE PTF (APAR PI49703) で導入されます。 IGZXAPI について詳しくは、「*z/OS Language Environment Vendor Interfaces*」で IGZXAPI の説明を参照してください。

関連タスク
『*LIST* 出力の読み取り』 (*Enterprise COBOL for z/OS* プログラミング・ガイド)

関連参照
例*:* プログラム *Prolog* 領域
(*Enterprise COBOL for z/OS* プログラミング・ガイド)
Common interfaces and conventions (*z/OS Language Environment Vendor Interfaces*)

第 17 章 Enterprise COBOL バージョン 5 およびバージョン 6 での変更点

Enterprise COBOL V5 または V6 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。

Enterprise COBOL V5 および V6 での変更点は、主に以下のカテゴリーに分類されます。

- 前提ソフトウェアおよび前提サービスの変更
- COBOL ソース・コードの相違
- コンパイラー・オプションの変更
- コンパイル動作の相違
- バインディング (リンク・エディット) の変更
- 実行時の変更点
- デバッグ情報および Debug Tool 動作の変更

COBOL V5 または V6 への移行は、それ以前の COBOL 移行とは次の点で異なります。この移行では、ご使用のプログラムが無効データを使用していて、COBOL V5 または V6 で異なる結果が返されるかどうかを調べるためのリグレッション・テストが推奨されます。 以前の COBOL コンパイラーでは、同じコードおよび同じデータ・レイアウトが生成されていたため、無効データによって、バージョンが異なっていても同じ結果が返されました。

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。

バージョン 5 のコンパイラーのパフォーマンス特性は、バージョン 6 の場合と同様です。特に注釈がある場合を除いて、このセクションにおける変更および移行推奨は、バージョン 5 およびバージョン 6 のどちらにも適用されます。

Enterprise COBOL バージョン 5 およびバージョン 6 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 および V6 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 および V6 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。

- IBM z/OS V1R13 以降
- IBM CICS Transaction Server for z/OS V3 以降
- IBM DB2 V9 以降
- IBM IMS V11 以降
- IBM Problem Determination Tools for z/OS V12 以降 (Debug Tool、Fault Analyzer、File Manager、Application Performance Analyzer)
- IBM Rational Developer for System z (RDz) V9.5 以降

Enterprise COBOL V6.1 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。

- IBM z/OS V2R1 以降

- IBM CICS Transaction Server for z/OS V4 以降
- IBM DB2 V10 以降
- IBM IMS V12 以降
- IBM Problem Determination Tools for z/OS V13 以降 (Debug Tool、Fault Analyzer、File Manager、Application Performance Analyzer)
- IBM Rational Developer for z Systems (RDz) V9.5.1.1 以降
- IBM Developer for z Systems (IDz) Enterprise Edition V14.0 以降

Enterprise COBOL V6.2 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。

- IBM z/OS V2R1 以降
- IBM CICS Transaction Server for z/OS V4.2 以降
- IBM DB2 V11 以降
- IBM IMS V13 以降
- IBM Problem Determination Tools for z/OS V13 以降 (Debug Tool、Fault Analyzer、File Manager、Application Performance Analyzer)
- IBM Developer for z/OS (IDz) Enterprise Edition V14.1 以降

注 : デバッグ情報が別のサイド・ファイルにあるプログラム・オブジェクトを (Enterprise COBOL V6.2 の新しい TEST(SEPARATE) オプションを使用して) デバッグするには、IBM Debug for z Systems V14.1 (5655-Q50) 以降、IBM Developer for z Systems V14.1 (5724-T07) 以降、または IBM Application Delivery Foundation for z Systems V3.1 (5655-AC6) 以降が必要です。

Enterprise COBOL V6.3 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。

- IBM z/OS V2R2 以降

  注 : AMODE 64 (64 ビット) の COBOL アプリケーションを実行するには、z/OS V2.3 (5650-ZOS) 以降が必要です。
- IBM CICS Transaction Server for z/OS V5.3 以降
- IBM DB2 V11 以降
- IBM IMS V14 以降
- IBM Application Delivery Foundation for z Systems V3.2 以降
- IBM Application Performance Analyzer for z/OS V14.1.8 以降
- IBM Fault Analyzer for z/OS V14.1.8 以降
- IBM File Manager for z/OS V14.1.8 以降
- IBM Developer for z/OS V14.2 以降
- IBM Debug for z/OS V14.2 以降

注 : 前提ソフトウェア・レベルは、Enterprise COBOL V6 リリースの一般出荷開始日の時点で有効でした。今後、一部の製品はサポート終了の予定を発表していくことになります。 現在サポートされているバージョンについては、ソフトウェア・ライフサイクルのページを確認してください。

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 および V6 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 および V6 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 および V6 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 および V6 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、HOLDDATA において、FIXCAT カテゴリー名で特定されます。 COBOL では現在、5 つあります。

- COBOL V5.1: `IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R1`
- COBOL V5.2: `IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R2`
- COBOL V6.1: `IBM.TargetSystem-RequiredService.Enterprise-COBOL.V6R1`
- COBOL V6.2: `IBM.TargetSystem-RequiredService.Enterprise-COBOL.V6R2`
- COBOL V6.3: `IBM.TargetSystem-RequiredService.Enterprise-COBOL.V6R3`

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 または V6 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。COBOL V6.3 (およびこれ以前のバージョン) において、z/OS CSI に対して実行するために使用されるサンプル・コマンドは次のとおりです。

```
SET BDY(GLOBAL).
REPORT MISSINGFIX ZONES(ZOS13T)
FIXCAT(IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R1,
       IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R2,
       IBM.TargetSystem-RequiredService.Enterprise-COBOL.V6R1,
       IBM.TargetSystem-RequiredService.Enterprise-COBOL.V6R2,
       IBM.TargetSystem-RequiredService.Enterprise-COBOL.V6R3)
```

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 または V6 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 または V6 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 または V6 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。また、APAR PI12240、PI26838、および PI58762 の PTF には FLAGMIG4 オプションに対する更新が含まれているため、これらの PTF もインストールすることをお勧めします。

  注 : COBOL V5 および V6 におけるソース・コードの変更が COBOL 言語機能で使用されることはほとんどないため、それらの変更は COBOL ユーザーの 99% には影響しません。
- PM85035 - XML-INFORMATION 特殊レジスターをサポートする新機能。これは、XMLPARSE が追加される前に、COBOL V5.1 への移行に必要な XMLPARSE(XMLSS) への移行に役立っていました。 COBOL V5.1 (サービス適用済み) および COBOL V5.2 以降のコンパイラーでは、XMLPARSE コンパイラー・オプションが追加されたため、XMLPARSE(XMLSS) への移行は不要になりました。
- PI40323 - ZONECHECK。 このオプションは、数値 DISPLAY ゾーン 10 進データ項目において、無効な COBOL データのケースを見つけるために役立ちます。 無効データによって、COBOL V5 または V6 では、以前の COBOL コンパイラー・リリースと比較して異なる結果が返される可能性があります。
- 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 およびバージョン 6 における COBOL ソース・コードの相違

Enterprise COBOL V5 および V6 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

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 または V6 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 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

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 または V6 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。

予約語 VOLATILE

Enterprise COBOL V5.2 以降、VOLATILE が予約語になりました。 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 および V6 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。

INSPECT...TALLYING の動作

`INSPECT...TALLYING` に対して、以前のバージョンのコンパイラーは、INSPECT を実行する前に、符号付き数値表示の検査対象項目にゾーン・ニブルを挿入します。これによって、例えば、スペースがゼロに変更されます。COBOL V5 以降のバージョンでは、このゾーン正規化は行われなくなっています。 `REPLACING` 節の更新なしで INSPECT が実行されると、検査される項目は予期しないものになっていましたが、COBOL V5 以降のバージョンではこれは行われません。

COBOL V4 と同様の予期しないかたちでの動作は、COBOL V5 以降のバージョンでは発生しません。 REPLACING を追加したり除去したりしても、エラーと見なされる COBOL V4 以前の動作は再現されません。COBOL V4 で予期しない動作を回避するには、INSPECT の前に NUMERIC クラスのテストを追加するか、検査される項目内にスペースまたは英数字データを移動しないようにするか、検査される項目内にゼロを移動することができます。COBOL V5 以降のバージョンでは、検査されるデータ項目が予期しない方法で変更されることはありません。

以下に例を示します。

```
01 TEST-DATA.
  02  NUM-DISP  PIC S9(9).

. . .

MOVE 0 To TALLY
MOVE SPACES TO TEST-DATA
INSPECT NUM-DISP TALLYING TALLY FOR ALL ZEROES
IF TALLY > 0 THEN
   DISPLAY 'This is COBOL V4 or earlier'
```

```
ELSE
   DISPLAY 'This is COBOL V6'
```

ご使用のプログラムがこの動作に依存している場合、COBOL V6 では、スペースのゼロへの置き換えなどのために REPLACING を使用して INSPECT を実行するようにプログラムを変更できます。

```
INSPECT NUM-DISP REPLACING ALL SPACES BY '0'
```

この例は実質的に COBOL V4 で行われたことと同じですが、符号付き数値表示のデータ項目のその他の非数値コンテンツの置き換えが必要になることもあります。

移動の命令

x'8000' 以上の値を持つ 16 ビット COMP-5 送信側 (PICTURE 節 PIC 9(2) から PIC 9(4) まで) を英数字データ項目に移動するとき、Enterprise COBOL V4.2 は、高位 16 ビットがすべて 1 である状態の 32 ビット値として値をロードする命令を誤って使用します。 これで、PIC X(9) 受信側に移動される値が誤って変更されます。 Enterprise COBOL V5 および V6 は、高位 16 ビットがすべてゼロである状態の 32 ビット値として 16 ビット値をロードします。これは正しい動作であって、Enterprise COBOL V4.2 とは異なります。

Enterprise COBOL V6 にのみ適用される変更

以下の相違点は、特に Enterprise COBOL V6 に該当します。

予約語

Enterprise COBOL V6.1 以降では、ALLOCATE、DEFAULT、END-JSON、FREE、JSON、および JSON-CODE が新しい予約語です。 これらの語をユーザー定義語 (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 V6 で S レベルの診断メッセージを受け取ります。 これらの予約語のインスタンスを ALLOCATE-X や JSON-Y 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の変更を自動的に行うことができます。

Enterprise COBOL V6.2 以降、JSON-STATUS が予約語になりました。JSON-STATUS 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 で S レベルの診断メッセージを受け取ります。JSON-STATUS のこれらのインスタンスは、JSON-STATUS-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 V6.3 から、BYTE-LENGTH、JAVA、LIMIT、POINTER-32、および UTF-8 が新しい予約語になりました。 これらの語をユーザー定義語 (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 V6.3 で S レベルの診断メッセージを受け取ります。これらの予約語のインスタンスを BYTE-LENGTH-X や BYTE-LENGTH-Y 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の変更を自動的に行うことができます。

VALUE 節

Enterprise COBOL V5 以前のバージョンでは、LINKAGE SECTION または FILE SECTION 内の 88 以外のレベルの VALUE 節は、コメントとして扱われ、無視されていました。

Enterprise COBOL V5 以前のバージョンでのコンパイルの例:

```
  000224          LINKAGE SECTION.
  000225            01 ALPH-ITEM  PIC X(4)  VALUE 1234.


==000225==> IGYDS1158-I A non-level-88 "VALUE" clause was found in the
"FILE SECTION" or "LINKAGE SECTION". The "VALUE" clause was treated as comments.
```

しかし、Enterprise COBOL V6.1 以降、LINKAGE SECTION 項目および FILE SECTION 項目の VALUE 節は構文検査の対象になり、意味を持つようになりました 。

Enterprise COBOL V6 でのコンパイル:

```
  000224          LINKAGE SECTION.
  000225            01 ALPH-ITEM  PIC X(4)  VALUE 1234.
```

```
==000225==> IGYGR1080-S A "VALUE" clause literal was not compatible with the data
category of the subject data item. The "VALUE" clause was discarded.
```

V6 では、以下のようになります。

- データ VALUE リテラルに PICTURE 節との互換性がない場合は、上記の例に示されているように、IGYGR1080-S エラー・メッセージが出されます。
- データ VALUE リテラルに PICTURE 節との互換性がある場合は、LINKAGE SECTION 内のデータ項目の初期化に使用されます。

要約すると、RC=0 でコンパイルされた、LINKAGE SECTION 内に 88 以外のレベルの VALUE 節がある COBOL V5 プログラムは、VALUE 節のリテラルの妥当性に応じて、COBOL V6 で RC=12 を受け取るか、LINKAGE データ項目が初期化されます。

CALL...USING *file-name* ステートメント

CALL ステートメントの USING 句を使用してサブプログラムに *file-name* を渡す方法は、Enterprise COBOL V6.3 で削除されましたが、Enterprise COBOL V6.3 に APAR PH20724 に対応する PTF をインストールすると復元されます。

Enterprise COBOL バージョン 5 およびバージョン 6 におけるコンパイラー・オプションの変更

Enterprise COBOL V5 および V6 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 *39. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
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。 • Enterprise COBOL V5.1 から、AFP(VOLATILE) がデフォルトです。 • Enterprise COBOL V6.2 から、AFP(NOVOLATILE) 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 および V6.1 では、ARCH(6) は受け入れられなくなり、ARCH(7) がデフォルトです。 • Enterprise COBOL V6.2 では、新しい上位レベルの ARCH(12) が受け入れられます。 デフォルトは引き続き ARCH(7) です。 • Enterprise COBOL V6.3 では、ARCH(7) が削除され、新しい上位レベルの ARCH(13) が受け入れられます。 ARCH(8) がデフォルトです。
COPYLOC	Enterprise COBOL V6.1 (サービス PTF 適用済み)、V6.2 (サービス PTF 適用済み)、および V6.3 以降の新しいオプションです。 これを使用すれば、ライブラリー・フェーズでコピー・メンバーを検索する追加ロケーションとして PDSE (または PDS) データ・セットあるいは z/OS UNIX ディレクトリーを追加できます。
COPYRIGHT	Enterprise COBOL V5.2 から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。

表 *39. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
DEFINE	Enterprise COBOL V6.2 からの新しいオプションです。これは、PARAMETER 句を持つ DEFINE ディレクティブを使用して、プログラムに定義されているコンパイル変数にリテラル値を割り当てます。
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。
INITCHECK	Enterprise COBOL V5.2 (サービス PTF 適用済み)、V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 未初期化データ項目を検査し、それらが初期化されていないまま使用されている場合に警告メッセージを出すかどうかを制御します。
INITIAL	Enterprise COBOL V6.2 (サービス PTF 適用済み)、および V6.3 以降の新しいオプションです。これにより、プログラムとそのネスト・プログラムは、IS INITIAL 節が PROGRAM-ID 段落に指定されたかのように動作します。
INLINE	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 これは、ソース・プログラム内の PERFORM ステートメントによって参照されるプロシージャーをインライン化するかどうかを、コンパイラーが考慮するように制御します。
INVDATA	Enterprise COBOL V6.2 (サービス PTF 適用済み)、および V6.3 以降の新しいオプションです。このオプションは、非推奨の ZONEDATA オプションの代わりとなるオプションです。これは、USAGE DISPLAY データ項目および PACKED-DECIMAL データ項目におけるデータが有効であるかどうかをコンパイラーに伝え、そのデータが有効ではない場合、コンパイラーの動作はどうあるべきかをコンパイラーに指示します。 COBOL V5 または V6 へのマイグレーションを簡単に行うには、以下のようにします。 • 数字、符号、およびゾーン・ビットが有効な場合は、NOINVDATA を使用し、さらに COBOL V5 または V6 の使用時に COBOL V4 で使用した設定と同じ NUMPROC 設定を使用します。 **注 :** NUMPROC (MIG) は COBOL V5 または V6 では使用できなくなっているため、V4 でこれを使用していた場合、このシナリオでは代わりに NUMPROC (NOPFD) を使用する必要があります。 • 無効な数値、無効な符号、または無効なゾーン・ビットがある場合は、以下のようにします。 – COBOL V4 で NUMPROC(MIG) を使用していた場合、COBOL V5 や V6 では INVDATA(FORCENUMCMP,NOCLEANSIGN) と NUMPROC(NOPFD) を使用します。 – COBOL V4 で NUMPROC(NOPFD) を使用していた場合、COBOL V5 や V6 では INVDATA(NOFORCENUMCMP,CLEANSIGN) を (または単純に INVDATA を) 使用します。 – COBOL V4 で NUMPROC(PFD) を使用していた場合、COBOL V5 や V6 では INVDATA(NOFORCENUMCMP,CLEANSIGN) を (または単純に INVDATA を) 使用します。

表 *39. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
LP	Enterprise COBOL V6.3 からの新しいオプションです。関連する言語機能を有効にして、AMODE 31 (31 ビット) または AMODE 64 (64 ビット) のどちらのプログラムを生成するかを指示します。LP(32) がデフォルトです。
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。 • Enterprise COBOL V5.1 以降では、MAXPCF(60000) がデフォルトです。 • Enterprise COBOL V6.2 以降では、MAXPCF(100000) がデフォルトです。
NUMCHECK	Enterprise COBOL V5.2 (サービス PTF 適用済み)、V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 これは、送信データ項目として使用するゾーン 10 進データ項目およびパック 10 進データ項目に対して、暗黙的な数値のクラス・テストを生成するかどうか、また 2 進データ項目に対して SIZE ERROR 検査を生成するかどうかを制御します。
PARMCHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降の新しいオプションです。 これは、WORKING-STORAGE にある最後の項目の後に追加のデータ項目を生成するよう、コンパイラーに指示します。 このバッファー・データ項目は、呼び出されたサブプログラムが WORKING-STORAGE の終わりを超えてデータを破損させたかどうかを検査するために、ランタイムに使用されます。
QUALIFY	Enterprise COBOL V5.2 から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。
RULES	Enterprise COBOL V5.2 からの新しいオプションです。これは、コンパイル時に特定タイプのソース・コードにフラグを立てることにより、プログラムを改善するためにプログラムに関する情報をコンパイラーから要求します。
SERVICE	Enterprise COBOL V5.2 から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	Enterprise COBOL V5.1 (サービス PTF 適用済み)、および V5.2 以降の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 ステートメントを含むソース・プログラムを処理します。
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。 Enterprise COBOL V5.1、V5.2、および V6.1 では、NOSTGOPT が有効になっていても、OPT(2) でデータ項目を最適化できます。 Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、NOSTGOPT が変更され、OPT(2) を指定してもストレージやデータ項目の最適化は行われなくなりました。これは特に WORKING-STORAGE の目印に役立ちます。
SUPPRESS	Enterprise COBOL V6.1 からの新しいオプションです。COPY ステートメントの SUPPRESS 句を無視するかどうかを制御します。
TUNE	Enterprise COBOL V6.3 (サービス PTF 適用済み) 以降の新しいオプションです。これは、実行可能プログラムの最適化の対象となるアーキテクチャーを指定します。 ARCH を指定する場合は、デフォルトの TUNE レベルと ARCH レベルが一致している必要があります。ARCH を指定しない場合は、ARCH と TUNE は両方ともデフォルトで 8 に設定されます。

表 *39. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
VLR	Enterprise COBOL V5.1 (サービス PTF 適用済み)、および V5.2 以降の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(STANDARD) がデフォルトです。 詳細については、210 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。
VSAMOPENFS	Enterprise COBOL V6.1 からの新しいオプションです。ファイル整合性検査の確認を必要とする、正常に実行された VSAM OPEN ステートメントから報告されるユーザー・ファイル状況に作用します。
XMLPARSE	Enterprise COBOL V5.1 (サービス PTF 適用済み)、および V5.2 以降の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 または V6 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。
ZONECHECK	Enterprise COBOL V5.2 (サービス PTF 適用済み) および V6.1 の新しいオプションです。送信データ項目として使用するゾーン 10 進データ項目に対して IF NUMERIC クラス・テストを生成するようコンパイラーに指示します。 Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、ZONECHECK は非推奨ですが、互換性のために使用は許容されています。代わりに、NUMCHECK(ZON) の使用を検討してください。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。

表 *39. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
ZONEDATA	Enterprise COBOL V5.2 からの新しいオプションです。これは、USAGE DISPLAY データ項目および PACKED-DECIMAL データ項目におけるデータが有効であるかどうかをコンパイラーに伝え、そのデータが有効ではない場合、コンパイラーの動作はどうあるべきかをコンパイラーに指示します。 最初、ベース・レベルの Enterprise COBOL V5.2 には、NOPFD サブオプションがありませんでした。 V5.2 (サービス PTF 適用済み)、および V6.1 以降では、COBOL V4 で NUMPROC(NOPFD|PFD) を使用した場合に COBOL V4 が実行するのと同じようにしてゾーン 10 進数データの比較を実行するコードをコンパイラーで生成するための、NOPFD サブオプションが追加されました。 COBOL V5 または V6 へのマイグレーションを容易にするには、以下のようにします。 • 数字、符号、およびゾーン・ビットが有効な場合は、ZONEDATA(PFD) を使用し、さらに COBOL V5 または V6 の使用時に COBOL V4 で使用した設定と同じ NUMPROC 設定を使用します。 • 無効な数値、無効な符号、または無効なゾーン・ビットがある場合は、以下のようにします。 – COBOL V4 で NUMPROC(MIG) を使用した場合は、ZONEDATA(MIG) と NUMPROC(NOPFD) を COBOL V5 または V6 で使用します。 – COBOL V4 で NUMPROC(NOPFD) を使用した場合は、ZONEDATA(NOPFD) と NUMPROC(NOPFD) を COBOL V5 または V6 で使用します。 – COBOL V4 で NUMPROC(PFD) を使用した場合は、ZONEDATA(NOPFD) と NUMPROC(PFD) を COBOL V5 または V6 で使用します。 **注 :** Enterprise COBOL V6.2、および V6.3 (サービス PTF 適用済み) では、ZONEDATA は非推奨ですが、互換性のために使用は許容されています。代わりに、INVDATA の使用を検討してください。詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『INVDATA』を参照してください。

Enterprise COBOL バージョン 6 の新規コンパイラー・オプションについては、Enterprise COBOL バージョン 6 の新規コンパイラー・オプションを参照してください。

以下のオプションは変更されています。

表 *40. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
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。
INITCHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、V6.2 (サービス PTF 適用済み)、および V6.3 (サービス PTF 適用済み) 以降では、INITCHECK オプションに新しいサブオプション LAX | STRICT が追加されました。このサブオプションは、データ項目がステートメントへの論理パスの 1 つ以上で初期化されているという条件と、ステートメントへのすべての論理パスで初期化されているという条件の、どちらの条件に当てはまらない場合にコンパイラーでデータ項目に関する警告メッセージを出すかを制御します。

表 *40. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント

LANGUAGE	COBOL V6 において、大文字英語または日本語のコンパイラー・メッセージに変更するためには、コンパイル時に LANGUAGE コンパイラー・オプションを使用し、さらに、Language Environment ランタイム・オプション NATLANG を設定する必要があります。 コンパイル JCL で CEEOPTS DD を使用するようお勧めします。
	例えば、メッセージを日本語に変更するためには、コンパイル時に LANGUAGE(JA) コンパイラー・オプションを使用し、さらに、NATLANG LE ランタイム・オプションを指定してください。
	`//CEEOPTS DD *`
	`NATLANG(JPN)`
	`/*`

MAP	Enterprise COBOL V5.1 (サービス PTF 適用済み)、および V5.2 以降では、MAP コンパイラー・オプションに新しいサブオプション HEX | DEC が追加されました。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 進と 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、基本レベルの Enterprise COBOL V5.1 は当初、MAP 出力に 10 進オフセットを表示していました。Enterprise COBOL V5.1 (サービス PTF 適用済み) 以降では、MAP オプションに新しいサブオプション HEX および DEC が追加されました。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 V5 または V6 で得られます。このため、Enterprise COBOL V5 または V6 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。
NOSTGOPT	• Enterprise COBOL V5.1 以降では、NOSTGOPT が有効になっていても、OPT(2) でデータ項目を最適化できます。 • Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、NOSTGOPT が変更され、OPT(2) を指定してもストレージやデータ項目の最適化は行われなくなりました。これは特に WORKING-STORAGE の目印に役立ちます。
NUMCHECK	Enterprise COBOL V6.3 以降では、コンパイル時に無効なデータが見つかった場合、`NUMCHECK(MSG)` と `NUMCHECK(ABD)` のどちらが有効であるかにかかわらず、エラー・レベル・メッセージが生成され、検査は除去されます。

表 *40. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
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。
RULES	Enterprise COBOL V6.2 (サービス PTF 適用済み)、および V6.3 以降では、RULES コンパイラー・オプションに以下の新しいサブオプションが追加されました。 • OMITODOMIN | NOOMITODOMIN は、*integer-1* (最小出現回数) なしで指定されているすべての OCCURS DEPENDING ON 節に関して警告メッセージを発行するかどうかをコンパイラーに指示します。 • UNREF | NOUNREFALL | NOUNREFSOURCE は、未参照データ項目に対して警告メッセージを発行するのかどうかをコンパイラーに指示したり、報告が行われるのがコピー・メンバーで宣言されていないデータ項目に関してのみ (NOUNREFSOURCE) なのかすべてのデータ項目に関して (NOUNREFALL) なのかを制御するようにコンパイラーに指示したりします。 • LAXREDEF | NOLAXREDEF は、任意のレベルでデータ項目がさらに小さい項目に再定義される際に警告メッセージを出すかどうかをコンパイラーに指示します。
SOURCE	Enterprise COBOL V6.3 (サービス PTF 適用済み) 以降では、 SOURCE コンパイラー・オプションに新しいサブオプション DEC | HEX が追加されました。 SOURCE(DEC) が有効な場合、ソースのリストの行番号は 10 進形式になります。 SOURCE(HEX) が有効な場合は、ソースのリストの行番号は 16 進形式になります。
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。 Enterprise COBOL V5.2 (サービス PTF 適用済み)、および V6.1 以降では、コンパイラーによる参照変更長の検査方法を制御する新しいサブオプション ZLEN | NOZLEN が、SSRANGE コンパイラー・オプションに追加されました。 Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、SSRANGE コンパイラー・オプションに新しいサブオプション MSG | ABD が追加されました。このサブオプションは、範囲検査が失敗したときの COBOL プログラムのランタイム動作を制御します。 **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 *40. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
TEST	• Enterprise COBOL V5.1 から、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 | 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。 • Enterprise COBOL V6.2 から、デバッグ機能を保持しながらディスク上のプログラム・オブジェクト・サイズを制御するために新規サブオプション SEPARATE | NOSEPARATE が TEST コンパイラー・オプションに追加されました。また、TEST(NODWARF)、TEST(SEPARATE)、NOTEST(DWARF,SOURCE) など、サブオプションの新しい組み合わせが TEST コンパイラー・オプションと NOTEST コンパイラー・オプションの両方でサポートされるようになりました。

Enterprise COBOL バージョン 6 で変更されたコンパイラー・オプションについては、Enterprise COBOL バージョン 6 で変更されたコンパイラー・オプションを参照してください。

以下のオプションは削除されました。

表 *41. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
DATEPROC	2000 年問題対応用の拡張のサポートが廃止されました。
LVLINFO	Enterprise COBOL V6.1 から、LVLINFO インストール・オプションが除去されました。以前は LVLINFO であった場所にビルド・レベル情報が入り、LVLINFO の代わりに、SERVICE コンパイラー・オプションをユーザー・サービス・レベル情報に使用できます。
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。

表 *41. Enterprise COBOL* バージョン *5* およびバージョン *6* 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	コメント
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 または V6 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。 NUMPROC(MIG) でコンパイルしたプログラムを Enterprise COBOL V6 にマイグレーションするには、NUMPROC(PFD) にマイグレーションするときに役立つ NUMCHECK コンパイラー・オプションの使用を検討してください。 1. プログラムを NUMCHECK(ZON,PAC) および NUMPROC(PFD) でコンパイルします。 2. 適切な大きさの入力データで完全なリグレッション・テストを実行します。 アプリケーションが NUMCHECK メッセージも NUMCHECK 異常終了も引き起こさなければ、NUMPROC(PFD) および NONUMCHECK による実動用のコンパイルを安全に実行できます。 これは無効なデータの問題を解決するだけではありません。 NUMPROC(PFD) は、NUMPROC コンパイラー・オプションのための最も有効な設定です。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。
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、201 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 でのコンパイルの変更点』を参照してください。 • Enterprise COBOL V5.2 以降では、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。
ZONECHECK	Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、ZONECHECK は非推奨ですが、互換性のために使用は許容されています。また、この代わりに NUMCHECK(ZON) が使用されるようになっています。

Enterprise COBOL バージョン 6 で除去されたコンパイラー・オプションについては、Enterprise COBOL バージョン 6 では無効なコンパイラー・オプションを参照してください。

以下のオプションは Enterprise COBOL V4 でサポートされなくなりましたが、V3 以前のバージョンからのマイグレーションを容易にするために、通知または警告メッセージで許容されていました。 Enterprise COBOL V5 および V6 では、これらのオプションは許容されません。いずれかを指定すると、結果としてエラー・メッセージが出されます。

- CMPR2
- EVENTS
- FDUMP
- FLAGSAA
- PFDSIGN
- RES

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、305 ページの『付録 E オプションの比較』を参照してください。

パフォーマンスに影響する可能性のあるコンパイラー・オプションの詳細なリストについては、「*Enterprise COBOL for z/OS* パフォーマンス・チューニング・ガイド」の『*V6* を最大限に活用するためのコンパイラー・オプションのチューニング方法』を参照してください。

すべてのコンパイラー・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」の『コンパイラー・オプション』を参照してください。

Enterprise COBOL バージョン 5 およびバージョン 6 でのコンパイルの変更点

Enterprise COBOL V5 および V6 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 V5 または V6 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 V5 または V6 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・レベルおよび必要な保守について詳しくは、187 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 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 V5.2 以降では、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。

Enterprise COBOL V6 では、コンパイラーはプログラムが大きくない場合でも、それらのプログラムをコンパイルするために 2 GB 境界より上のストレージを使用して始動します。 つまり、z/OS MEMLIMIT パラメーターをゼロでない値に設定しなければならない可能性があります。 MEMLIMIT の z/OS デフォルトは 2 GB ですが、プログラムをコンパイルするときに MEMLIMIT の z/OS 設定が十分な大きさではない場合、コンパイラー・メッセージ IGYCB7145-U Insufficient memory in the compiler to continue

compilation が返される可能性があります。 このエラー・メッセージが出された場合は、ジョブ・カードで REGION=0M および MEMLIMIT=3G を設定し、プログラムを再コンパイルしてください。 これが正常に行われた場合は、IEFUSI、SMFPRMxx、または SMFLIMxx で設定されたシステム MEMLIMIT デフォルトを、2 GB 以上に変更することを検討してください。

注 : SMFLIMxx PARMLIB メンバーは、z/OS V2.2 以降のバージョンにのみ用意されています。

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 以降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。これらの変更について詳しくは、「*Enterprise COBOL* プログラミング・ガイド」を参照してください。
- Enterprise COBOL V6.1 以降、ビルド・レベル情報 (形式 PYYMMDD) が常に、リスト・ファイルのヘッダーに組み込まれます。この情報は、コンパイラーの保守レベルを調べるために役立ちます。 リスト・ヘッダーの例:

```
PP 5655-EC6 IBM Enterprise COBOL for z/OS 6.3.0 PXXXXXX
```

- Enterprise COBOL V5 および V6.1 では、診断メッセージはリストの中央部にあります。 Enterprise COBOL V6.2 以降のバージョンでは、Enterprise COBOL V4 以前のコンパイラーと同様に、診断メッセージがリストの下部に示されるようになりました。
- Enterprise COBOL V5 および V6 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、329 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。
- Enterprise COBOL V6.3 以降、リストの用語は次のように変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの STATIC MAP は INITIAL HEAP STORAGE MAP に変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの書き込み可能静的領域 (WSA) は storage に変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの WSA24 は BELOW THE LINE STORAGE に変更されました。
  - Enterprise COBOL V6.2 以前のバージョンの AUTOMATIC MAP は STACK STORAGE MAP に変更されました。
- Enterprise COBOL V6.3 以降、コンパイラー・フェーズを共有ストレージに入れるためのインストール・カスタマイズはなくなりました。 最新のシステムでは、多くの場合ユーザーが使用できるストレージが増えたため、コンパイラー・フェーズを共有ストレージに入れることによってストレージを節約する必要がないからです。 コンパイラーに対するこの変更の結果として、コンパイラー・フェーズを共有ストレージに入れるための言語はサポートされなくなりました。 したがって、コンパイラー・フェーズが共有ストレージの IN または OUT であることを指定する IGYCDOPT カスタマイズの保存済みコピーがある場合、その言語を除去してからでなければ IGYCDOPT をアセンブルできません。 コンパイラー・フェーズ

の IN または OUT を指定する IGYCDOPT 内のステートメントがない場合には、 この変更によって影響を受けることはありません。

- CALL ステートメントの USING 句を使用してサブプログラムに *file-name* を渡す方法は、Enterprise COBOL V6.3 で削除されましたが、Enterprise COBOL V6.3 に APAR PH20724 に対応する PTF をインストールすると復元されます。

IBM Enterprise COBOL Value Unit Edition (VUE) for z/OS V5.2 以降のバージョンを使用している場合、同じタスクから複数回 (例えば、MVS LINK マクロを使用して) コンパイラーを起動することはできません。

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 およびバージョン 6 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 およびバージョン 6 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。 これは、Enterprise COBOL バージョン 5 およびバージョン 6 で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 およびバージョン 6 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 およびバージョン 6 における JCL およびパッケージ化の変更

Enterprise COBOL V5 および V6 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。

Enterprise COBOL V5 および V6 に適用される変更

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 および V6 は、以下の追加のデータ・セットを必要とします。

- 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 および V6 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 および V6 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 および V6 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 V6 に適用される変更

COBOL V6 において、大文字英語または日本語のコンパイラー・メッセージに変更するためには、コンパイル時に LANGUAGE コンパイラー・オプションを使用し、さらに、Language Environment ランタイム・オプション NATLANG を設定する必要があります。 コンパイル JCL で CEEOPTS DD を使用するようお勧めします。

例えば、メッセージを日本語に変更するためには、コンパイル時に LANGUAGE(JA) コンパイラー・オプションを使用し、さらに、NATLANG LE ランタイム・オプションを指定してください。

```
//CEEOPTS DD *
         NATLANG(JPN)
/*
```

Enterprise COBOL V6.3 以降、COBOL AMODE 64 (64 ビット) プログラムの開発を支援するために、コンパイルを行うための新しいカタログ式プロシージャーが提供されています。 AMODE 64 サポートは Enterprise COBOL V6.3 に導入された新規フィーチャーです。 AMODE 64 サポートの詳細については、AMODE 64 プログラムの開発 (*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 およびバージョン 6 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 および V6 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 および V6 でのユーザー作成条件ハンドラー

Enterprise COBOL V5.2 および V6 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 および V6 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、『*VOLATILE* 節』(*Enterprise COBOL for z/OS* 言語解説書) を参照してください。

Enterprise COBOL バージョン 5 およびバージョン 6 におけるバインド (リンク・エディット) の変更

Enterprise COBOL V5 または V6 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 または V6 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。
- The Language Environment プリリンカーは、GOFF オブジェクト・フォーマットを認識しないため、使用できなくなりました。 プリリンカーが前のオブジェクト・モジュール・フォーマットに対して行う変換は、プログラム・オブジェクト・フィーチャーと互換性がありません。 Enterprise COBOL V5 または V6 アプリケーションを含む実行可能プログラムのコンポーネントに Language Environment プリリンカーを使用すると、未定義の動作が生じます。
- 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。 バッチ・ローダー (IEWBLDG0) は、 GOFF オブジェクト・フォーマットやプログラム・オブジェクト・フォーマットを認識しないため、 区分データ・セットまたは PDSE にプログラム・モジュールを作成するためには使用できません。 代わりに、プログラム管理ローダーが PDSE データ・セット内のプログラムをサポートできます。
- 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。
  - V5 より前のコンパイラーでコンパイルされた COBOL プログラム。 AMODE 24 とともに実行され、COBOL V5 以降でコンパイルされた COBOL プログラムを静的に呼び出すもの。

Enterprise COBOL バージョン 5 およびバージョン 6 での実行時の変更点

Enterprise COBOL V5 および V6 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 またはバージョン 6 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 またはバージョン 6 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、209 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 V5 および V6 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、23 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。
- 無効データによって、COBOL V5 および V6 では、それ以前の COBOL バージョンの場合とは異なる結果が生じる可能性があります。 より新しいコンパイラーでは以前のコンパイラーとは異なる結果が得られたり、各種の OPT または ARCH 設定で異なる結果が得られたりすることを、一部のユーザーは確認しています。通常、これは無効データが実行時に COBOL プログラムに送られたことが原因です。 ご使用のプログラムでこの問題が発生するかどうかを調べる 1 つの方法は、以下の新しい移行推奨に従うことです。
  1. SSRANGE、NUMCHECK、PARMCHECK、INITCHECK、および OPT(0) を使用してコンパイルし、リグレッション・テストを実行する。
  2. 問題があるかどうかを確認する。

- 問題が見つからなければ、NOSSRANGE、NONUMCHECK、NOPARMCHECK、および OPT(2) で再コンパイルしてから、最終テストを実行してアプリケーションを実動に移行する。
- 問題が見つかった場合、プログラム/データを修正するか、または不正なデータがゾーン 10 進データ項目にあれば、INVDATA コンパイラー・オプションを使用して無効データを受け入れる。

INITCHECK オプションと NUMCHECK オプションはいずれも、Enterprise COBOL V6.2、V6.1、および V5.2 (最新サービス適用済み) で使用可能です。

注: この追加のテストは、Enterprise COBOL V5 または V6 で既にコンパイルされたプログラムに対して実行する必要はありません。

- まれなシナリオとして、Enterprise COBOL V5 および V6 では、COMPUTE と ROUNDED を使用する有効なデータが、旧 COBOL バージョンとは異なる結果を返すことがあります。

  このシナリオは次のようなものです。ROUNDED を指定した COMPUTE ステートメントでの最後の演算が加算または減算である場合、その演算のオペランドの小数部精度がその計算の中間結果の小数部精度と違う場合は、オペランドにシフトが必要となる可能性があります。このときに右シフトが実行され、その結果としてオペランドの小数部精度が失われると、そのオペランドの値は演算の実行の前に丸められることになります。そのデータ値に基づいて、この結果が、Enterprise COBOL と旧コンパイラーとで異なるものになることがあります。常に推奨されることとして、移行後に計算結果が正しいか確認するためのリグレッション・テストを行うことをお勧めします。
- 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 およびバージョン 6 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 または AMODE 31 のどちらか、および 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。
- AMODE 24 実行がサポートされていないため、アプリケーションを AMODE 31 で実行しなければならない場合があります。 詳細については、209 ページの『AMODE の制約事項』を参照してください。
- Enterprise COBOL V5 または V6 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。
- COBOL 環境で ABEND を要求するための ILBOABN0 インターフェースは、Enterprise COBOL V5 以降のバージョン で動的に呼び出すことができます。 Enterprise COBOL コンパイラーでコンパイルされたプログラムによって呼び出されると、ACTION コード 1 を使用して CEE3ABD を呼び出したときと同じ結果になります。

  CEE3ABD インターフェースは、作成された CEEDUMP で提供される詳細のレベルを制御するための高い柔軟性を備えているため、CEE3ABD インターフェースを移行して使用するよう強くお勧めします。

  ご使用のアプリケーションが Enterprise COBOL プログラムによって呼び出される場合、より古いバージョンの ILBOABN0 (LE の SCEELKED より前) が静的にリンクされていれば、予期しない ABEND が発生する可能性があります。 予期しない ABEND を修正するには、以下の提案のいずれかに従ってください。
  - CEE3ABD に移行してください。
  - LE の SCEELKED に対する LINK ステップで、アプリケーションを REPLACE ILBOABN0 に再リンクしてください。
  - ILBOABN0 の動的呼び出しを使用するよう、COBOL プログラムを変更してください。
- 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 または V6 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 または V6 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、210 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) ファイルを更新して Enterprise COBOL V5 および V6 ランタイム・モジュールを組み込まなければならない場合があります。 詳細については、237 ページの『CSD セットアップにおける Enterprise COBOL V5 および V6 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、213 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。

COBOL V5 以降では、プロシージャーと関数のポインターは、エントリー・ポイントを直接指すのではなく、関数記述子を指します。これらのポインターが直接 (データ専用モジュールなどの) エントリー・ポイントを指すことがコードで予期されている場合は、コードの変更が必要です。詳しくは、プロシージャー・ポインターと関数ポインターの使用 (*Enterprise COBOL for z/OS* プログラミング・ガイド) を参照してください。

COBOL V5 以降では、プロシージャーと関数のポインターに対する呼び出しは、LE スタック・フレームのあるモジュールから行わなければなりません。これは高水準プログラミング言語の場合になります。この種の呼び出しをアセンブラー・モジュールから行う場合は、CEEENTRY マクロと CEETERM マクロを使用して LE スタック・フレームを提供し、必要とされる関連するレジスターの内容も提供する必要があります。

Enterprise COBOL V6 にのみ適用される変更

- 始動時に STORAGE ランタイム・オプションを使用して WORKING-STORAGE を選択した値に初期化できない場合があります。 以下のようなケースがあります。
  - スパン (RECORDING MODE S) ファイルを持つ COBOL V6 プログラム
  - DATA(31) でコンパイルされた非 CICS COBOL V5 プログラム
- V6 でのファイル状況の変更:
  - レコード・サイズが一致しない行順次ファイルに対する WRITE ステートメント

    以前のリリースの Enterprise COBOL では、レコード・サイズが一致しない行順次ファイルにレコードを書き込もうと試みた場合、ファイル状況 48 が誤って返されました。 これは Enterprise COBOL V6 で修正され、ファイル状況 44 が返されるようになりました。
  - UNIX ファイル属性が「書き込みのみ」である場合の、行順次ファイルに対する OPEN INPUT

    以前のリリースの Enterprise COBOL では、「書き込みのみ」属性を持つ行順次ファイル (DD PATHOPTS=(OWRONLY,...) を持つ z/OS UNIX ファイルや、書き込みアクセス権のみを持つ COBOL プログラムなど) に対する、INPUT 句を持つ OPEN ステートメントが、誤ってファイル状況 0 (正常) を返しました。 オープン・アクセス・モードをサポートしないファイルに対して OPEN ステートメントを試みた場合は、ファイル状況 37 が返されなければなりません。

    注 : この「書き込みのみ」は、行順次ファイルに適用可能ではない APPLY WRITE-ONLY 節を意味するわけではありません。 行順次ファイルは、z/OS UNIX ファイル・システムで作成されたファイルです。

    Enterprise COBOL V6 では、この OPEN ステートメントはファイル状況 37 で検出されます。
  - ファイル属性が一致しない VSAM ファイルに対する OPEN INPUT、I-O、EXTEND

    以前のリリースの Enterprise COBOL では、OPTIONAL と定義されていない VSAM ファイルに対して OPEN INPUT ステートメント、I-O ステートメント、または EXTEND ステートメントを試みたときに

ファイル属性の不一致が検出された場合、ファイル状況 35 が誤って返されました。 これは Enterprise COBOL V6 で修正され、ファイル状況 39 が返されるようになりました。

注 :

- OPEN OUTPUT における、また VSAM ファイルが OPTIONAL と定義されている場合の OPEN INPUT、I-O、および EXTEND における、同じようなファイル属性不一致条件は、既にファイル状況 39 として正しく報告されるように修正されています。
- Enterprise COBOL V6.3 以降では、LP=64 コンパイラー・オプションを使用すると、コンパイル・プロセスには POSIX(ON) モードで実行されるコンポーネントが組み込まれます。 必然的に、このオプションを指定してコンパイラーを実行するユーザーごとに RACF で OMVS セグメント (RACF 代替製品で同等のもの) が設定されていなければならないということになります。

Language Environment オプションの変更

Enterprise COBOL バージョン 5 およびバージョン 6 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 またはバージョン 6 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 *42. Enterprise COBOL* バージョン *5* およびバージョン *6* におけるランタイム・オプションの変更

オプション	コメント
HEAP	• COBOL V5 では、いくつかの CICS 以外の環境 (バッチ、TSO、IMS など) において、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。 CICS 以外の環境での 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。 • COBOL V6 以降では、WORKING-STORAGE スペース (RENT でコンパイルされたプログラム用) が HEAP から取得されます。 そのため、HEAP (および STORAGE) オプションに効力があります。
CHECK(OFF)	CHECK(OFF) は、SSRANGE でコンパイルされた COBOL V5 または V6 プログラムのランタイム添え字範囲検査を無効にしません。
NOSSRANGE	NOSSRANGE ランタイム・オプションは、SSRANGE でコンパイルされた COBOL V5 または V6 プログラムのランタイム添え字範囲検査を無効にしません。 **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	COBOL V5 における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。

AMODE の制約事項

AMODE 24 の実行は以下のケースではサポートされていません。 アプリケーションは AMODE 31 で実行されなければなりません。 これは、COBOL V3 および V4 と同じ AMODE 24 制約事項のセットです。

- XML PARSE ステートメントを含むプログラム
- XML GENERATE ステートメントを含むプログラム
- C プログラム、C++ プログラム、または PL/I プログラムにバインドされた COBOL を含み、静的 CALL を通じて通信を行うプログラム・オブジェクト

- INVOKE ステートメントなどのオブジェクト指向言語構文や、オブジェクト指向クラス定義を含むプログラム
- 以下のいずれかのコンパイラー・オプションを使用してコンパイルされたプログラム
  - DLL
  - PGMNAME(LONGUPPER)
  - PGMNAME(LONGMIXED)
- マルチスレッド・アプリケーション

  注: THREAD オプションを使用してコンパイルされたプログラムは AMODE 24 で実行できますが、 実行先は複数のスレッドや複数の PL/I タスクを持たないアプリケーションに限られます。
- z/OS UNIX ファイル・システムから実行されるプログラム

  注: z/OS UNIX ファイル・システムに常駐する AMODE 31 ドライバー・プログラムは、MVS PDSE に常駐する AMODE 24 プログラム・モジュールに対する動的呼び出しを含むことができます。
- EXIT コンパイラー・オプションで指定された COBOL コンパイラー出口モジュールとして使用されるプログラム
- XPLINK を使用する言語環境プログラム・エンクレーブ。XPLINK コンパイラー・オプションを使用してコンパイルされた非 COBOL プログラムを含むエンクレーブ、または XPLINK ランタイム・オプションを使用して実行されるエンクレーブが含まれます。

注: COBOL プログラムをアドレッシング・モード 24 で実行するには、 すべての COBOL プログラムを、Enterprise COBOL V5.1.1 以降のバージョン、または Enterprise COBOL V4.2 以前のバージョンを使用してコンパイルする必要があります。 プログラム・オブジェクトに含まれるいずれかのコンポーネントが Enterprise COBOL V5.1.0 を使用して コンパイルされたものである場合、そのプログラム・オブジェクトはアドレッシング・モード 31 で実行されなければなりません。 アドレッシング・モード 24 を使用して実行される COBOL プログラムは、バインダー・オプション RMODE(24) を使用して リンクされなければなりません。

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 以降のバージョンでは、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 および V6 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 および V6 への移行が妨げられる場合があります。このため、Enterprise COBOL V5.1、および V5.2 以降のバージョンでは、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。

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 および V6 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 または V6 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 および V6 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。Enterprise COBOL V4 以前の初期化コードに含まれる PROGRAM-ID、COMPILED TIME、COMPILED DATE などの情報は、Enterprise COBOL V5 以降の初期化コードには含まれ

ません。 そのため、その情報が依存するプログラムの動作は Enterprise COBOL V5 および V6 で異なります。

- 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
  – V5 および V6 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). *> The last item in WORKING-STORAGE SECTION
PROCEDURE DIVISION.
. . .
   CALL 'SUBP' USING GRP1.

PROGRAM-ID. SUBP.
LINKAGE SECTION.
01 GRP2 PIC X(500).
PROCEDURE DIVISION USING GRP2
   MOVE 'STUFF' TO GRP2(300:20) *> This is illegal!
```

  結果:

  上の例では、GRP1 の長さと GRP2 の長さは一致していません。 GRP2 への MOVE によって、呼び出し側プログラムの WORKING-STORAGE にある最後のデータ項目に続いて、ストレージのオーバーレイが発生します。

  – V2、V3、および V4: 通常、WORKING-STORAGE (CALLER を参照) にある最後のデータ項目の後に未使用のストレージがあり、上書きは検出されなかったために、不正な MOVE によって障害は発生しませんでした。
  – V5 および V6: ファイル制御ブロックが、WORKING-STORAGE にある最後のデータ項目の直後に続きます。 そのため、CALLER のファイル状況情報はオーバーレイされ、その後でプログラムのフローが変わる可能性があります。

- 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'.              <* Value x'F0F040F0'
   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。

データに無効な数字、無効な符号コード、または無効なゾーン・ビットがある場合は、プログラムが実行時に数値データ項目に無効なデータを持たないように、プログラムまたはシステムを変更してください。

プログラムまたはシステムを修正したら、お好みの NOINVDATA オプションを使用できます。この作業を含めることができず、引き続き無効データを使用して処理を続行しなければならない場合にのみ、INVDATA オプションに対して以下の選択項目を検討してください。

- COBOL V4 で NUMPROC(MIG) を使用した場合は、INVDATA(FORCENUMCMP,NOCLEANSIGN) と NUMPROC(NOPFD) を COBOL V6 で使用します。
- COBOL V4 で NUMPROC(NOPFD) を使用した場合は、COBOL V6 では INVDATA(NOFORCENUMCMP,CLEANSIGN) (または単純に INVDATA) と NUMPROC(NOPFD) を使用します。
- COBOL V4 で NUMPROC(PFD) を使用した場合は、COBOL V6 では INVDATA(NOFORCENUMCMP,CLEANSIGN) (または単純に INVDATA) と NUMPROC(PFD) を使用します。

注:

- 過去に COBOL V4 以前のバージョンから COBOL V5 または V6 へのマイグレーションを完了し、COBOL V5 または V6 で非推奨の ZONEDATA (MIG) オプションを使用していて、その動作に問題がなければ、 ZONEDATA(MIG) の代わりに INVDATA(FORCENUMCMP, CLEANSIGN) を使用してください。
- IBM は、プログラム、データ、またはその両方を訂正して、NOINVDATA オプションを使用することをお勧めします。

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、INVDATA(FORCENUMCMP) 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。

- 無効なゾーン・ビットを持つゾーン 10 進数データを使用しているプログラムに対して、SEARCH ALL ステートメントは、Enterprise COBOL V4 とそれ以降 (V5 および V6.1) とでは異なる結果を生成する場合があります。 Enterprise COBOL V6.2 では、SEARCH ALL ステートメントは、前項で説明されているように INVDATA オプションに従って動作します。 正しい形式の INVDATA オプションを使用して、Enterprise COBOL V4 以前のバージョンと同じ動作が生じるようにします。

  COBOL V6.2 ランタイム・ライブラリー使用可能化 PTF が Language Environment に適用されている場合、SEARCH ALL ステートメントは、前項で説明されているように INVDATA オプションに従って動作します。これは、再コンパイルなしにすべての COBOL V5 および V6 プログラムで使用可能です。 正しい形式の INVDATA オプションを使用して、Enterprise COBOL V4 以前のバージョンと同じ動作が生じるようにします。

  注 : 明らかに無効なデータが見つかった場合、これらのオプションを使用しても、常に古いコンパイラーの動作を完全に一致させることができるわけではありません。

  このオプションは、無効なゾーン 10 進数データを持つプログラムにのみ作用します。 無効なゾーン・ビットがない場合、SEARCH ALL ステートメントは、INVDATA オプションの設定とは無関係に、同じ結果を生成します。

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。

ILBOABN0 に関する考慮事項

COBOL 環境で ABEND を要求するための ILBOABN0 インターフェースは、Enterprise COBOL V5 以降のバージョン で動的に呼び出すことができます。 Enterprise COBOL コンパイラーでコンパイルされたプログ

ラムによって呼び出されると、ACTION コード 1 を使用して CEE3ABD を呼び出したときと同じ結果になります。

CEE3ABD インターフェースは、作成された CEEDUMP で提供される詳細のレベルを制御するための高い柔軟性を備えているため、CEE3ABD インターフェースを移行して使用するよう強くお勧めします。

ご使用のアプリケーションが Enterprise COBOL プログラムによって呼び出される場合、より古いバージョンの ILBOABN0 (LE の SCEELKED より前) が静的にリンクされていれば、予期しない ABEND が発生する可能性があります。 予期しない ABEND を修正するには、以下の提案のいずれかに従ってください。

- CEE3ABD に移行してください。
- LE の SCEELKED に対する LINK ステップで、アプリケーションを REPLACE ILBOABN0 に再リンクしてください。
- ILBOABN0 の動的呼び出しを使用するよう、COBOL プログラムを変更してください。

DFSORT オプション NOBLKSET の使用

BLKSET オプションは、DFSORT を呼び出す際のデフォルトです。これを使用する場合、DFSORT で効率的な BLOCKSET 手法を使用できます。NOBLKSET が有効な場合は、DFSORT はフォールバックして従来の手法を使用します。推奨されている設定は BLKSET です。

DFSORT では、以下の場合にも従来の手法が使用されます。

1. 中間作業用ストレージにテープ装置が使用されている場合。テープの代わりにディスク装置を使用すると、この制限を回避できます。中間ストレージを指定するには、SORTWKdd ステートメントを使用します。
2. DFSORT OPTION コントロールの RECORD ステートメント内で L5 が使用されている場合。L5 は、平均レコード長を指定します。L5 を使用する代わりに、AVGRLEN=n ステートメントを使用して同様に指定できます。

DFSORT での従来の技法の使用は、COBOL V4.2 以前のバージョンで許可されています。プログラムを COBOL V5 以降のバージョンにアップグレードしようとしている場合は、その機会を利用して DFSORT の従来の手法から移行することを強くお勧めします。

COBOL V5 以降のバージョンで、COBOL V4.2 以前のバージョンとの互換性を提供してこの手法の使用を許容するには、ランタイム LE PTF UI67483(V2R2)/UI67485(V2R3)/UI67486(V2R4) を適用し、プログラムの実行時に指示に従って JCL をセットアップします。

注 : これらのランタイム PTF は AMODE 31 にのみ適用されます。DFSORT の従来の手法は、AMODE 64 で実行されている COBOL プログラムではサポートされません。

Enterprise COBOL バージョン 5 およびバージョン 6 におけるデバッグ情報の変更

Enterprise COBOL V5 または V6 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 およびバージョン 6 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。

TEST オプションの変更

Enterprise COBOL V5 および V6.1 では、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。

Enterprise COBOL V6.2 では、TEST(SEPARATE) オプションでプログラムをコンパイルすることにより、サイド・ファイルへのデバッグ情報の生成がサポートされます。

TEST の変更について詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『*TEST*』を参照してください。

リスト情報の変更

Enterprise COBOL V5 および V6.1 では、診断メッセージはリストの下部にありません。リストの診断メッセージ部分を表示するには、以下の手順に従ってください。

1. コマンド行に **F 'end of c'** と入力します (ヘッダー: End of compilation を見つけるには、ISPF **FIND** コマンドを使用します)。
2. Enter を押します。
3. (オプション) 「戻る (Page back)」を押します。

Enterprise COBOL V6.2 では、Enterprise COBOL V4 以前のコンパイラーと同様に、診断メッセージが再びリストの下部に示されるようになりました。

Enterprise COBOL V6 にのみ適用される変更

- WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。詳しくは、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。
- Enterprise COBOL V6 では、コンパイラー内でプロセス間通信 (IPC) メッセージ・キューが使用されます。そのため、cob2 を使用して z/OS UNIX でコンパイルを行っているときにコンパイラーで内部エラーが発生し、コンパイラーが KILL シグナルによって終了した場合、終了した時点で残っているメッセージ・キューを照会し、失効したメッセージ・キューを除去する必要があります。失効したメッセージ・キューは、以下の z/OS UNIX コマンドで削除できます。

  1. **ipcs -q** を入力し、キューのリストを表示します。
  2. ご使用のユーザー ID に関連付けられているキューを見つけます。
  3. **ipcrm -q** を入力し、キューを削除します。

  z/OS バッチでコンパイルを行っている場合、コンパイラー・エラーの後で、失効したメッセージ・キューを削除する必要はありません。
- Enterprise COBOL V6.3 における PPA1 の変更点

  Enterprise COBOL V6.3 以降、PPA1 の flag3 のビット 30 (オフセット X'1C') は、拡張フラグ・フィールドの存在を示すために設定することができます。 このビットを設定した場合、拡張フラグは、ベクトル・レジスター域がオプション領域にあることを示すビット 0 を持つことになります。 このことは、Language Environment インターフェースに従って PPA1 にアクセスするツールやプログラム・コードに

は影響しないはずです。 PPA1 の詳細については、*z/OS Language Environment Vendor Interfaces* を参照してください。

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 および V6 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、231 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 で加えられた Debug Tool の変更』を参照してください。

WORKING-STORAGE SECTION の変更

WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。次の方法で Enterprise COBOL V5 および V6 プログラムにおいて実行時に WORKING-STORAGE を見つけることができます。

COBOL V5 および V6 において WORKING-STORAGE の開始を見つけるには、ダンプで PPA4 (Program Prologue Area 4) を見つける方法を知る必要があります。

A: AMODE 31 の場合

以下の説明は、プログラムを LP(32) でコンパイルした場合に当てはまります。

ダンプで PPA4 (Program Prolog Area 4) を見つける方法

1. ダンプにおいてトレースバックからプログラムの開始を見つけます。
2. 開始アドレス + x'0C' の位置にオフセット値があります。 これがプログラムの開始から PPA1 へのオフセットです。
3. 開始アドレス + PPA1 オフセット = PPA1。
4. ダンプ内のその位置に移動します。
5. PPA1 + x'04' の位置にオフセット値があります。 これがプログラムの開始から PPA2 へのオフセットです。
6. 開始アドレス + PPA2 オフセット = PPA2。
7. ダンプ内のその位置に移動します。
8. PPA2 + x'08' の位置にオフセット値があります。 これが PPA2 から PPA4 へのオフセットです。
9. PPA2 + PPA4 オフセット = PPA4。
10. ダンプ内のその位置に移動します。現在の位置が PPA4 です。

次に PPA4 のレイアウトについて知る必要があります。

PPA4 のレイアウト

PPA4 のレイアウト、および各 PPA4 のオフセット、長さ、説明について詳しくは、「*z/OS Language Environment Vendor Interfaces*」の『COBOL V5+ 32-bit PPA4 layout』を参照してください。

次に、いくつかの用語について知る必要があります。

知るべき用語

NORENT 静的領域
このストレージ域は、実行可能ファイルにおいて、NORENT でコンパイルされたプログラムごとに割り振られます。 NORENT プログラムの WORKING-STORAGE はここにあります。

LE の書き込み可能静的領域 (WSA)
COBOL V5/V6 プログラム・オブジェクト (実行可能ファイル) ごとに、このストレージ域があります。

RENT 静的領域
このストレージ域は、静的に実行可能ファイルにバインドされて RENT でコンパイルされたプログラムごとに WSA 内で割り振られます。 プログラムごとに独自の RENT 静的領域があります。 プログラムの WORKING-STORAGE はここに指定されることもあれば、ここには指定されないこともあります。

プログラム静的領域
このストレージ域は、特定の条件が満たされた場合にのみ WSA の外側で割り振られます。 その場合、プログラムの WORKING-STORAGE の場所は RENT 静的領域ではなくここです。

次に、WORKING-STORAGE が含まれている可能性がある場所が 3 つあることを理解する必要があります。

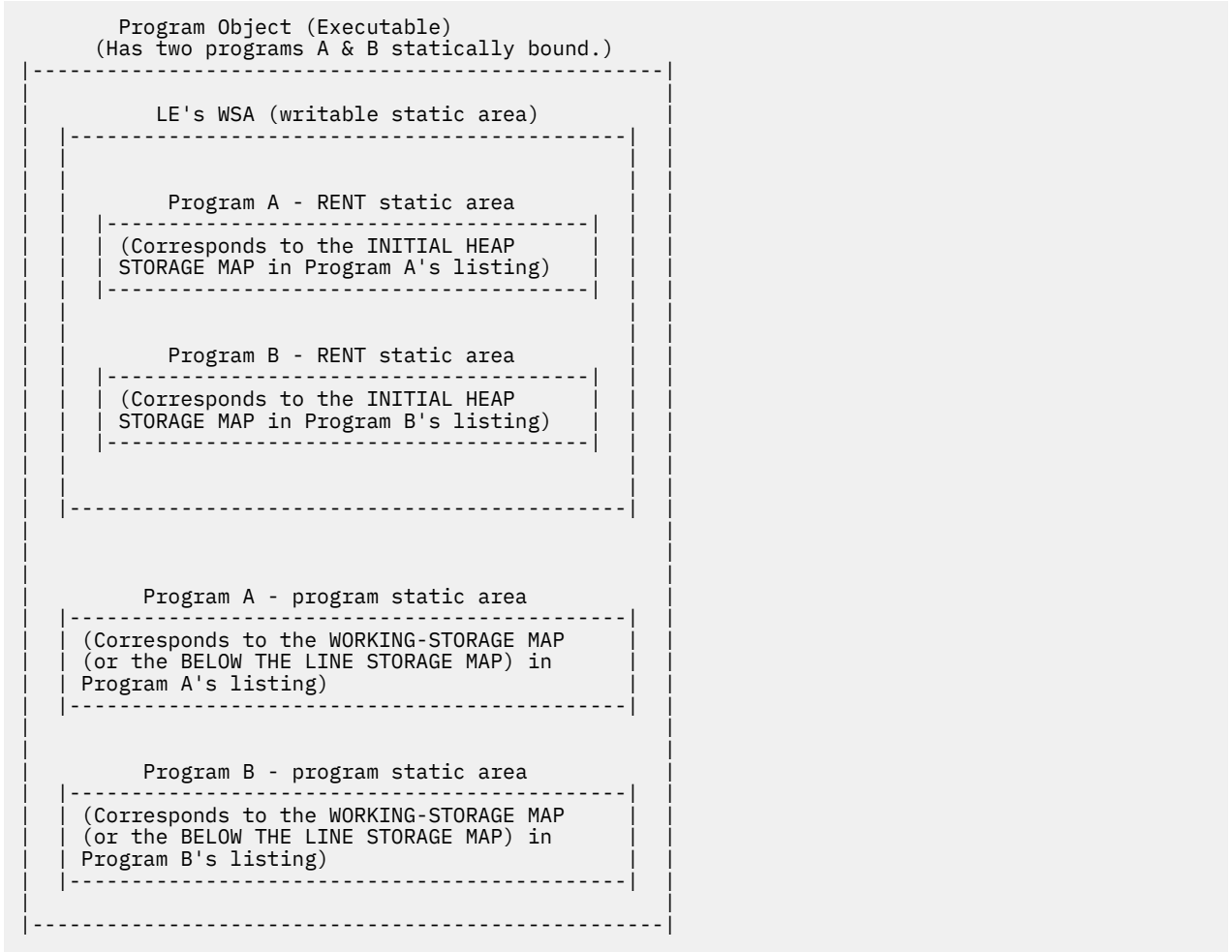
WORKING-STORAGE が含まれている可能性がある領域の説明

WORKING-STORAGE が含まれている可能性がある場所は 3 つあり、それぞれ別の場所です。

- プログラム・オブジェクト (実行可能ファイル) 内。 NORENT オプションでコンパイルされたすべてのプログラムには、実行可能プログラム内で予約された NORENT 静的領域があり、WORKING-STORAGE はここにあります。
- RENT オプションでコンパイルされたすべてのプログラムでは、RENT 静的領域は LE の WSA (書き込み可能静的領域) 内で割り振られます。 WORKING-STORAGE は、ここにある可能性があります。
- RENT 静的領域に置かれる代わりに、一部の COBOL V5 以降の RENT プログラムの WORKING-STORAGE は、LE の WSA の外で、プログラム静的領域と呼ばれる領域に割り振られます。

どこに WORKING-STORAGE が置かれるかを決定するための規則は、次のセクションにあります。

下の図は、その WORKING-STORAGE がプログラム静的領域にある RENT プログラム用に、どのようにストレージがレイアウトされているかを示しています。



WORKING-STORAGE が含まれている可能性がある 3 つの領域を理解したら、プログラムの WORKING-STORAGE が実際はどこにあるのかを判別する方法を知る必要があります。 詳しくは、「*z/OS Language Environment Vendor Interfaces*」の『IGZXAPI』の 『*WORKING-STORAGE location*』および『*Layout of the Language Environment WSA, STATIC, PROGRAM STATIC, and User Working Storage*』を参照してください。

WORKING-STORAGE がある領域の判別方法

表 *43. WORKING-STORAGE* がある領域

COBOL のバージョン	**コンパイラー・オプション**	**WORKING-STORAGE がある場所**
COBOL V5	NORENT	プログラムの NORENT 静的領域
	RENT、DATA(31)	WSA 内におけるプログラムの RENT 静的領域
	RENT、DATA(24)。または RENT、WSOPT[1]	WSA の外におけるプログラムのプログラム静的領域
COBOL V6 以降のバージョン	NORENT	プログラムの NORENT 静的領域
	RENT、DATA(31)、および SPANNED RECORDS (つまり WSOPT ビットはオフ)[2]	WSA 内におけるプログラムの RENT 静的領域
	RENT、DATA(24) (つまり WSOPT ビットはオン)[2]	WSA の外におけるプログラムのプログラム静的領域
	RENT、DATA(31)、および NO SPANNED RECORDS (つまり WSOPT ビットはオン)[2]	WSA の外におけるプログラムのプログラム静的領域

注:

1. COBOL V5 には WSOPT コンパイラー・オプションがあります。 COBOL V6 には WSOPT コンパイラー・オプションはなくなりましたが、コンパイラーによって自動的に設定される WSOPT に関するシグニチャー情報ビットがあります。
2. SPANNED RECORDS の場合、WSOPT シグニチャー情報ビットはオフです。 NO SPANNED RECORDS の場合、WSOPT シグニチャー情報ビットはオンです。
   - 「RECORDING MODE」を求めてプログラムをスキャンし、「S」に設定されているすべてのファイルを探せば、SPANNED RECORDS が使用されているかどうかを判別できます。
   - また、リスト内のシグニチャー情報バイトを調べて WSOPT ビットを探すという方法もあります (シグニチャー・バイト 8、ビット 3)。 例えば、リストから次の情報を取得します。

     ```
     =X'001000000000'   INFO. BYTES 7-12
     ```

     バイト 8 は x'10' (b'00010000') です。 ビットの番号は 01234567 のように左から右に並びます。 ビット 3 がオンなので WSOPT はオンです。

WORKING-STORAGE がある領域が分かれば、WORKING-STORAGE を見つける方法が分かります。

WORKING-STORAGE をダンプ内で見つける方法

表 *44. PPA4*、*NORENT* 静的領域、*LE* の *WSA*、*RENT* 静的領域、プログラム静的領域をダンプ内で見つける方法

見つける対象	**ダンプ内で見つける方法**
PPA4	上記の説明を参照してください。

表 *44. PPA4*、*NORENT* 静的領域、*LE* の *WSA*、*RENT* 静的領域、プログラム静的領域をダンプ内で見つける方法 (続き)

見つける対象	ダンプ内で見つける方法
NORENT 静的領域	当該アドレスがストレージ内の <PPA4 + x'08'> にあります。
LE の WSA	当該アドレスがストレージ内の <CEECAA (または R12) + x'1F4'> にあります。 これはダンプ内で CEECAARENT と記述されています。
RENT 静的領域	当該アドレスが以下のストレージ内の場所にあります。 <ストレージ内の CEECAA (または R12) + x'1F4' にあるアドレス> <プログラムの PPA4 + x'0C' 内のオフセット>
プログラム静的領域	当該アドレスが以下のストレージ内の場所にあります。 <ストレージ内の CEECAA (または R12) + x'1F4’ にあるアドレス> <プログラムの PPA4 + x'0C' 内のオフセット> <プログラムの PPA4 + x'10' 内のオフセット>

この領域がダンプで見つかったら、それをコンパイル・リストと比較できます。

COBOL リストでは、以下のようになっています。

- INITIAL HEAP STORAGE MAP は RENT 静的領域または NORENT 静的領域のレイアウトを示します。
- WORKING-STORAGE MAP または BELOW THE LINE STORAGE MAP はプログラム静的領域のレイアウトを示します。

B: AMODE 64 の場合

以下の情報は、プログラムを LP(64) でコンパイルした場合に当てはまります。

WORKING-STORAGE SECTION に関する情報は、ヒープ・ストレージ・アドレス・テーブルとともに、プログラムの PPA4 にあります。 それらを見つけるには、以下の手順に従ってください。

1. ダンプにおいて、トレースバックからプログラムのエントリー・ポイント・アドレスを見つけます。 LE CEEDUMP トレースバックでは、これはプログラムの行に対応する "E Addr" 列の下のアドレスです。

   下記の例では、HELLO が COBOL プログラムです。 そのエントリー・ポイント・アドレスは X'260000A8' です。 このアドレスには、プログラムの最初の実行可能命令、すなわち STMG 命令が含まれるはずです。

```
Traceback:
  DSA     Entry        E  Offset    …
  1       CEEHDSP      +00003F3C
  2       CELQHROD     +00000266
  3       HELLO        +00000224
  4       CELQINIT     +00001D0C

  DSA     DSA Addr           E  Addr
  1       000000050082FBC60  0000000026B0A3D0
  2       000000050082FEDA0  0000000026B1DD18
  3       000000050082FEFA0  000000002600000A8
  4       000000050082FF220  0000000026903010
```

2. プログラム・エントリー・ポイントのオフセット -x'08' (つまりエントリー・ポイントの前) に、整数値があります。 この値は、エントリー・ポイント・アドレスから PPA1 へのオフセットです。
3. PPA1+x'04' に、オフセット値があります。 これは、エントリー・ポイント・アドレスから PPA2 へのオフセットです。

4. PPA2+x'08' に、オフセット値があります。 これは、PPA2 から PPA4 へのオフセットです。
5. PPA4+x'7C' に、オフセット値があります。 これは、プログラムの環境 から、ヒープ・ストレージ・アドレス・テーブルへのオフセットです。

ここで、環境 はプログラムの XPLINK 環境を指します。 これは、プログラムへの入り口にあるレジスター R5 内のアドレスです。 プログラムの最初の命令である STMG では、レジスターをスタックに格納します。 R5 のコンテンツはダンプに見つかります。

ヒープ・ストレージ・アドレス・テーブル

プログラムの環境 からのこのテーブルのオフセットは PPA4+X'7C' にあります。

LP(64) における WORKING-STORAGE SECTION 内のデータ項目は、デフォルトで 2 GB 境界より上に割り振られます。 それらは COBOL の ABOVE THE BAR HEAP にあります。 その開始アドレスは、ヒープ・ストレージ・アドレス・テーブル の最初のフィールド (このテーブルのオフセット X'00') にあります。 なお、このアドレスはコンパイル・リストの ABOVE THE BAR HEAP MAP セクションにも対応しています。 それは、WORKING-STORAGE SECTION 内のレベル 77 および 01 データ項目に関する情報を提供します。

ABOVE THE BAR HEAP に割り振られる COBOL 制御域とコンパイラー内部変数もあります。 プログラム内の最初の WORKING-STORAGE データ項目は、初めから存在しているとは限りません。 プログラムの WORKING-STORAGE SECTION 内の最初のデータ項目のオフセットは、PPA4 のオフセット +X'40' で見つけられます。

表 *45.* ヒープ・ストレージ・アドレス・テーブル

	長さ	**説明**
X'00'	8	COBOL の ABOVE THE BAR HEAP (64 ビット・ストレージ域) の開始アドレス
X'08'	8	予約済み
X'10'	8	予約済み

WOKRING-STORAGE SECTION に関連する情報は、以下のように要約できます。

表 *46. WORKING-STORAGE SECTION* のまとめ

説明	**見つけられる場所**
R5 からのヒープ・ストレージ・アドレス・テーブルのオフセット	PPA4+x'7C'
WORKING-STORAGE の開始アドレス	ヒープ・ストレージ・アドレス・テーブル + x'00'
WORKING-STORAGE からの最初のユーザー 64 ビット・データ項目のオフセット	PPA4+x'40'
すべてのユーザー WORKING-STORAGE 64 ビット・データ項目を含む領域の長さ	PPA4+x'48'

PPA4 のレイアウト、および各 PPA4 のオフセット、長さ、説明について詳しくは、「*z/OS Language Environment Vendor Interfaces*」の『COBOL 64-bit PPA4 layout』を参照してください。

C: LE ベンダー・インターフェース IGZXAPI を使用した WORKING-STORAGE アドレスの照会

COBOL 固有のベンダー・インターフェース・ルーチン IGZXAPI を使用して、WORKING-STORAGE アドレスを照会することもできます。 Enterprise COBOL V6.1 では、LE ベンダー・インターフェース IGZXAPI が新しい機能コード 8 で機能拡張され、COBOL プログラムの WORKING-STORAGE SECTION の長さとアドレスに関する情報を要求できるようになりました。 返されるアドレスは、COBOL コンパイラー・リストの INITIAL HEAP STORAGE MAP セクションに対応しています。PROGRAM-ID からのプログラム名やシグニ

チャー情報バイトといった他の情報 (コンパイラー・リストの「Compiler Options and Program Information Section」に対応) も返されます。

この機能拡張は、COBOL Runtime LE PTF (APAR PI49703) で導入されます。 IGZXAPI について詳しくは、「*z/OS Language Environment Vendor Interfaces*」で IGZXAPI の説明を参照してください。

関連タスク
『*LIST* 出力の読み取り』 (*Enterprise COBOL for z/OS* プログラミング・ガイド)

関連参照
例*:* プログラム *Prolog* 領域
(*Enterprise COBOL for z/OS* プログラミング・ガイド)
Common interfaces and conventions (*z/OS Language Environment Vendor Interfaces*)

第 18 章 Enterprise COBOL バージョン 5 またはバージョン 6 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 または V6 プログラムを既存のアプリケーションに追加する場合、既存のプログラムを Enterprise COBOL V5 または V6 で再コンパイルするか、または新たに作成した Enterprise COBOL V5 または V6 プログラムを組み込みます。

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。

この手順が完了していない場合は、ここにある手順を実行する前にまず、「*Enterprise COBOL V4.2 Compiler and Runtime Migration Guide*」 (http://publibfp.dhe.ibm.com/epubs/pdf/igy3mg50.pdf) に記載されている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 または V6 プログラムを既存のアプリケーションに追加すると、以下のことを行うことが できます。

- 既存のプログラムをインストール先の要件に応じて漸進的にアップグレードする
- 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 または V6 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。

Language Environment を Enterprise COBOL V5 または V6 および VS COBOL II プログラムとともに使用

VS COBOL II プログラムと Enterprise COBOL V5 または V6 プログラムを一緒に実行するときは、以下の項目が必要となります。

- 現行バージョンの IGZEBST が必要となります。
  - CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換える必要があります。

    注 : APAR PI33330 の PTF がインストールされた LE からの IGZEBST は、COBOL V5 または V6 プログラムがなくても任意の COBOL プログラム VS COBOL II 以降と一緒に使用することもできます。
  - 動的な CALL の対象の CICS プログラムで必要なのは、APAR PI25079 用の PTF を SCEERUN にインストールすることのみです。

注 : 非 CICS で静的に呼び出されたプログラムに関しては、VS COBOL II プログラムを持つアプリケーションにおける IGZEBST を、APAR PI33330 の PTF がインストールされた LE からの IGZEBST で置き換えると、パフォーマンスが向上します。 これは必須ではありません。 COBOL V5 または V6 プログラムから VS COBOL II プログラムを呼び出すために非 CICS で動的に呼び出されるプログラムに関しては、IGZEBST の問題はありません。

- 現行バージョンの CEEBETBL (Language Environment 外部テーブル) が必要となります。 COBOL V5 または V6 の新規オブジェクト・コードと先程バインドしたオブジェクト・コードを組み込むと、古いバージョンの CEEBETBL が間接的に組み込まれてしまう場合があります。

  バインドする CEEBETBL の長さが x'28' (または現行 SCEELKED ライブラリーにおける CEEBETBL の長さ) よりも短い場合、その CEEBETBL は古い CEEBETBL であり、置き換える必要があります。 そうしないと、ランタイム異常終了が発生したり、強制終了ランタイム・メッセージが発行されたりします。

  移行の一環として COBOL V5 または V6 で古いオブジェクト・コードを再バインドする場合は、不用意に CEEBETBL を入り口点にすることがないように注意して、古いオブジェクト・コードを組み込む前に CEEBETBL の現行コピーを明確に組み込むことをお勧めします。

Enterprise COBOL バージョン 5 またはバージョン 6 プログラムに関する AMODE 制約事項

Enterprise COBOL V5.2 または V6 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 または V6 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。

Enterprise COBOL バージョン 5 またはバージョン 6 プログラムでの実行時の相違点

Enterprise COBOL V5 または V6 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 ロード・モジュール・アナライザー 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。
  - http://cbttape.org/cbtdowns.htm にある無料の COBOL アナライザーを使用して、ロード・ライブラリーをスキャンし、OS/VS COBOL プログラムを探します。 このアナライザーは、その Web ページ上では「*File # 321 COBOL Analyzer from Roland Schiradin & post processor*」 として示されています。
  - 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。

COBOL 環境で ABEND を要求するための `ILBOABN0` インターフェースは、Enterprise COBOL V5 以降のバージョン で動的に呼び出すことができます。 Enterprise COBOL コンパイラーでコンパイルされたプログラムによって呼び出されると、ACTION コード 1 を使用して CEE3ABD を呼び出したときと同じ結果になります。

`CEE3ABD` インターフェースは、作成された `CEEDUMP` で提供される詳細のレベルを制御するための高い柔軟性を備えているため、`CEE3ABD` インターフェースを移行して使用するよう強くお勧めします。

ご使用のアプリケーションが Enterprise COBOL プログラムによって呼び出される場合、より古いバージョンの `ILBOABN0` (LE の SCEELKED より前) が静的にリンクされていれば、予期しない ABEND が発生する可能性があります。 予期しない ABEND を修正するには、以下のアドバイスのいずれかに従ってください。

- `CEE3ABD` に移行してください。

- LE の SCEELKED に対する LINK ステップで、アプリケーションを REPLACE ILBOABN0 に再リンクしてください。
- ILBOABN0 の動的呼び出しを使用するよう、COBOL プログラムを変更してください。

Enterprise COBOL バージョン 5 またはバージョン 6 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 V4.2 以前のプログラムとの間の静的呼び出しは、Enterprise COBOL V4.2 以前のプログラムで AMODE 24 がサポートされている場合はサポートされます。

注 : AMODE 64 (LP(64)) プログラムと、異なる AMODE のプログラムとの間の呼び出しはサポートされていません。

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。

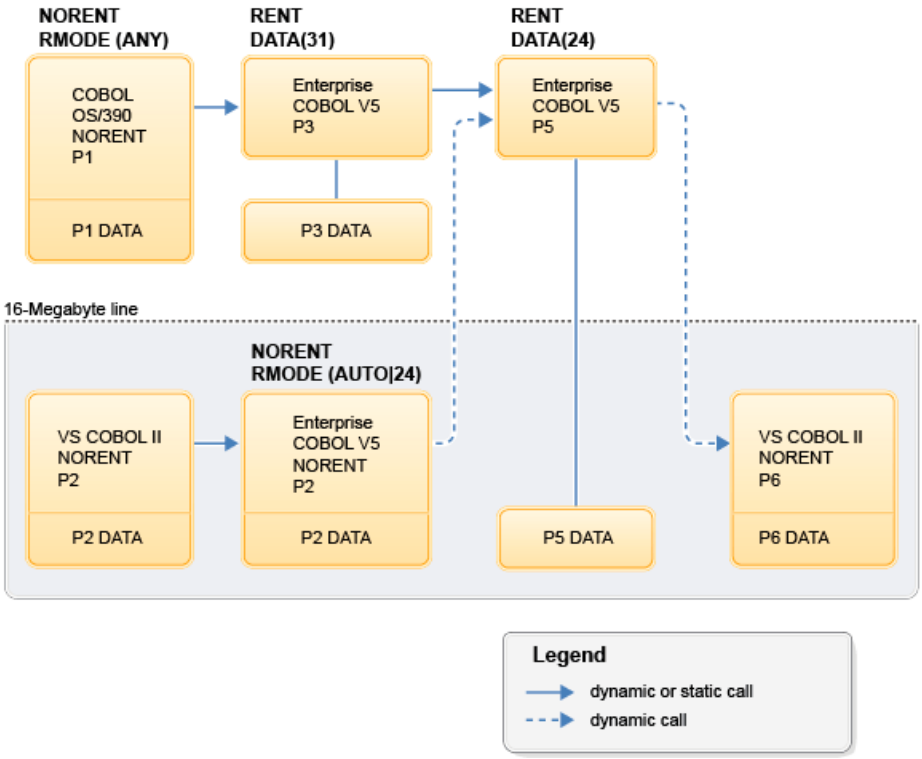


図 *5. AMODE* および *RMODE* の各種 *COBOL* プログラム間で有効な動的呼び出しと静的呼び出し

注 : 他の AMODE 24 プログラムの場合は、Enterprise COBOL V5 または V6 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 V5 および V6 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 19 章 IBM Debug for z/OS

IBM Debug for z/OS (以前の IBM Debug for z Systems および IBM Debug Tool for z/OS) 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。 本書では、

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 が呼び出す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。

Enterprise COBOL バージョン 5 およびバージョン 6 におけるデバッグ情報の変更

Enterprise COBOL V5 または V6 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 およびバージョン 6 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。

TEST オプションの変更

Enterprise COBOL V5 および V6.1 では、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。

Enterprise COBOL V6.2 では、TEST(SEPARATE) オプションでプログラムをコンパイルすることにより、サイド・ファイルへのデバッグ情報の生成がサポートされます。

TEST の変更について詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『*TEST*』を参照してください。

リスト情報の変更

Enterprise COBOL V5 および V6.1 では、診断メッセージはリストの下部にありません。リストの診断メッセージ部分を表示するには、以下の手順に従ってください。

1. コマンド行に **F 'end of c'** と入力します (ヘッダー: `End of compilation` を見つけるには、ISPF **FIND** コマンドを使用します)。
2. Enter を押します。
3. (オプション) 「戻る (Page back)」を押します。

Enterprise COBOL V6.2 では、Enterprise COBOL V4 以前のコンパイラーと同様に、診断メッセージが再びリストの下部に示されるようになりました。

Enterprise COBOL V6 にのみ適用される変更

- WORKING-STORAGE SECTION の割り振りおよび管理が、Enterprise COBOL V5 以降変更されました。これは、COBOL プログラムの実行には影響しません。 WORKING-STORAGE SECTION の開始アドレスを見つける必要があるツールまたはプログラムに影響する可能性があります。詳しくは、181 ページの『WORKING-STORAGE SECTION の変更』を参照してください。
- Enterprise COBOL V6 では、コンパイラー内でプロセス間通信 (IPC) メッセージ・キューが使用されます。そのため、cob2 を使用して z/OS UNIX でコンパイルを行っているときにコンパイラーで内部エラーが発生し、コンパイラーが KILL シグナルによって終了した場合、終了した時点で残っているメッセージ・キューを照会し、失効したメッセージ・キューを除去する必要があります。失効したメッセージ・キューは、以下の z/OS UNIX コマンドで削除できます。

  1. **ipcs -q** を入力し、キューのリストを表示します。
  2. ご使用のユーザー ID に関連付けられているキューを見つけます。
  3. **ipcrm -q** を入力し、キューを削除します。

z/OS バッチでコンパイルを行っている場合、コンパイラー・エラーの後で、失効したメッセージ・キューを削除する必要はありません。

- Enterprise COBOL V6.3 における PPA1 の変更点

  Enterprise COBOL V6.3 以降、PPA1 の flag3 のビット 30 (オフセット X'1C') は、拡張フラグ・フィールドの存在を示すために設定することができます。 このビットを設定した場合、拡張フラグは、ベクトル・レジスター域がオプション領域にあることを示すビット 0 を持つことになります。 このことは、Language Environment インターフェースに従って PPA1 にアクセスするツールやプログラム・コードには影響しないはずです。 PPA1 の詳細については、*z/OS Language Environment Vendor Interfaces* を参照してください。

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 および V6 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、231 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 で加えられた Debug Tool の変更』を参照してください。

Enterprise COBOL バージョン 5 およびバージョン 6 で加えられた Debug Tool の変更

Debug Tool を使用してデバッグを行う場合、Enterprise COBOL バージョン 5 およびバージョン 6 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、https://www.ibm.com/support/docview.wss?uid=swg27048563#debugger 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 V5 および V6 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。 次に例を示します。

- 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 1 対のアポストロフィ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 および V6 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 および V6 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 for z/OS 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 V5 および V6 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 および V6 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 および V6 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 および V6 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 および V6 は、配列の INDEX 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 および V6 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 および V6 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 および V6 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 および V6 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 および V6 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 および V6 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 および V6 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 for z/OS プログラミング・ガイド」に定義されています。

Enterprise COBOL V5 および V6 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。

Enterprise COBOL バージョン 5 およびバージョン 6 におけるフルスクリーン・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 および V6 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 および V6 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 および V6 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 または V6 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 および V6 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 および V6 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 または V6 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。

Enterprise COBOL バージョン 5 およびバージョン 6 における 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 および V6 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 および V6 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 および V6 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 および V6 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 および V6 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 および V6 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 および V6 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 20 章 CICS 移行における考慮事項

COBOL V5 以降のプログラムを CICS 環境で実行するためには、必要な CSD 更新および DFHRPL 更新について十分理解しておく必要があります。 また、CICS 用のコンパイラー・オプションについても十分理解しておく必要があります。 最終的には、統合 CICS 変換プログラムへの移行を実行するようお勧めします。

以下のトピックを考えてください。

- 237 ページの『CSD セットアップにおける Enterprise COBOL V5 および V6 との違い』
- 238 ページの『DFHRPL セットアップにおける Enterprise COBOL V5 および V6 との違い』
- 239 ページの『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』
- 240 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』
- 242 ページの『CICS 下での COBOL V5 または V6 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 および V6 との違い

以下の CICS TS バージョンでは、CICS がシステム自動インストール を使用して必要な Enterprise COBOL V5 および V6 ランタイム・モジュールをインストールするため、CICS システム定義 (CSD) ファイルを自分で更新する必要はありません。

- CICS TS V5.4 以降
- CICS TS V5.3 (APAR PI60389 用 PTF 適用済み)
- CICS TS V5.1 および V5.2 (APAR PI60388 および PI73184 用 PTF 適用済み)

対象の PTF が適用されていない場合や、以前の CICS TS バージョンに関しては、以下のように、Enterprise COBOL V5 および V6 ランタイム・モジュールを組み込むように CSD ファイルを更新する必要があります。

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、CSD 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 および V6 用に追加する必要があります。以下のモジュールの多く は、Language Environment データ・セット SCEERUN に含まれています。

- CEEEV004
- IGZXLPKA
- IGZXD24
- IGZXDMR
- IGZLLIBV
- IGZXLPKC
- IGZXLPIO
- IGZXAPI
- IEWBNDD
- IEWBIND
- CDAEEDE
- IGZXLPKB

- IGZXLPKD
- IGZXLPKE
- IGZXLPKF
- IGZXLPKG
- IGZXPK2
- IGZUOPT*
- IGZXCDA

* デフォルトでは、IGZUOPT モジュールは SCEERUN に含まれていません。 IGZUOPT は、Language Environment データ・セット SCEESAMP にあるサンプル JCL IGZ1OPT を使って作成できる、オプションのモジュールです。 ジョブ・カードおよびロード・ライブラリー名を変更し、IGZ1OPT JCL を実行して IGZUOPT を作成します。

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)
DEFINE PROGRAM(IGZUOPT) GROUP(CEE)*
DEFINE PROGRAM(IGZXCDA) GROUP(CEE)
```

* デフォルトでは、IGZUOPT はこのサンプルに含まれていませんが、使用中であれば追加することができます。

DFHRPL セットアップにおける Enterprise COBOL V5 および V6 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 (領域内でデバッグする場合)、および Language Environment ランタイム・ライブラリー (SCEECICS、SCEERUN、加えて、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。 DFHRPL 連結は、CICS 領域セットアップ JCL にあります。

TEST コンパイラー・オプションを使用して CICS の下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、 TEST(...,SEPARATE) を排他的に使用している場合を除いて、 アプリケーション内のデバッグ・セグメントへのバインダー・アクセスをサポートするために、 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。

バインダーがデバッグ・セグメントにアクセスするには、CICS 領域のユーザー ID に DFHRPL DD 連結内のライブラリーに対する読み取り権限が必要です。 読み取りアクセスを提供しないと、以下の診断メッセージが生成されます。

```
IEW2716S D801 OPEN FAILED FOR DDNAME DFHRPL.
IEW2146S 02D9 CONFLICTING INPUT SPECIFICATIONS ON AN INCLUDE CALL.
```

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

239 ページの表 47 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 *47. 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	コメント

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、または >>CALLINTERFACE DYNAM ディレクティブ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 47. *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	コメント
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 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 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。

CICS 変換プログラムのオプションの完全なリストについては、 *CICS* アプリケーションの開発の変換プログラムのオプションの定義を参照してください。

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

分離型 CICS 変換プログラムは、浮動コメント区切り、JSON GENERATE および JSON PARSE、コンパイラー・ディレクティブなど、より新しい COBOL 言語に対して更新されていません。 COBOL コンパイラーの最新機能を使用するには、統合 CICS 変換プログラムを使用してください。

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 V5.2 以降では、統合 CICS 変換プログラムが用意された EXCI 変換プログラム・オプションがサポートされています。 CICS V5.2 より前のバージョンでは、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。 EXCI 変換プログラム・オプションでも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

241 ページの表 48 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 *48.* 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	コメント
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。

表 48. 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 (続き)

コンパイラー・オプション	コメント
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、または >>CALLINTERFACE DYNAM ディレクティブ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。
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 または V6 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 または V6 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 21 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 以降では、 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

組み込みの SQL 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・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 の側面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。

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。例えば、次のように指定します。

```
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。)

注 : ピリオドは、他の EXEC SQL ステートメント (EXEC SQL INCLUDE ステートメントは除く) の後には必要ありません。

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。

注 : END-EXEC と同じ行にピリオドがある SQL ステートメントでの組み込み SQL コプロセッサーの使用が変更されるよう、ここには、END-EXEC の後に続くピリオドも含まれます。 プリコンパイラーの場合、ピリオドは無視されるため、削除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: 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。

SYSLIB と COPY 対 EXEC SQL INCLUDE

プリコンパイラー: プリコンパイラーでは、コンパイラーとは異なる SYSLIB 連結が使用されるため、例えば 'member1' のような特定のメンバー名は、プリコンパイルでは EXEC SQL INCLUDE 'member1'、コンパイルでは COPY 'member1' を使用して、2 つの異なるメンバーを参照できます。

コプロセッサー: SYSLIB 連結は、EXEC SQL INCLUDE ステートメントと COPY ステートメントの両方に 1 つだけです。 この場合、1 つの名前を使用して 2 つの異なるメンバーを参照することはできません。

コプロセッサーにマイグレーションするための処置: SQL INCLUDE メンバーまたはコピーブック・メンバーを名前変更し、対応するソース・コード変更を、関係する SQL INCLUDE ステートメントまたは COPY ステートメントに加えます。

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。
  - CODEPAGE コンパイラー・オプションに指定する CCSID は、データベース・エンコード用の DSNHDECP の CCSID に一致している必要があります。
- NOSQLCCSID (推奨):
  - 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 (DSNHDECP など) 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 22 章 IMS プログラムを Enterprise COBOL V5 または V6 に移動する

COBOL を IMS 出口ルーチンに使用する場合、COBOL V5 および V6 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 または V6 ではコンパイルできません。実際の出口ルーチンが、PDSE 内の COBOL V5 または V6 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 または V6 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 または V6 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 または V6 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 または V6 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 または V6 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 *49. 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、基本バージョンの z/OS では、COBOL V5 または V6 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 はこの問題を、APAR OA45127 によるサービス・ストリームで修正しました。 あるいは、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 または V6 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。 初期ロードでは、プログラム・オブジェクトの一部のみが取り込まれます。プログラム・オブジェクトの他の部分は、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) にのみ取り込まれます。 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。

また、プログラム・オブジェクトに RMODE 24 CSECT が含まれている場合に LLA を使用するときの考慮事項については、「*Enterprise COBOL for z/OS* パフォーマンス・チューニング・ガイド」の『ランタイム・パフォーマンスに影響するその他の製品関連要因』 も参照してください。

付録 A よくある質問 (FAQ) と回答

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。

- マイグレーション前
- 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

マイグレーション前

このトピックでは、マイグレーション前のよくある質問について説明します。

COBOL マイグレーションとは何ですか?

ライセンス交付およびシステム・プログラマーの観点からすると、 マイグレーションとは、ビルド・コンパイラーを新しいバージョンに更新することを意味します。 これには、必要なソース、オプション、テスト、および環境の変更が含まれます。 最新のマイグレーション・ターゲットは Enterprise COBOL V6 です。

開発の観点からすると、マイグレーションとは、パフォーマンス上のメリットを得て COBOL の新機能にアクセスするために、新しいコンパイラーを使用してプログラムを再コンパイルすることを意味します。

COBOL V4 以前から COBOL V6 へのマイグレーションは、COBOL V3 から V4 へのマイグレーションとは異なりますか?

はい。新しい世代の COBOL コンパイラーはさまざまな命令を使用して COBOL プログラムをより効率的にするため、実行時の無効なデータの使用についての隠れた根本的な問題を明らかにすることができます。 無効なデータがあると異なる動作が発生することがあるため、追加のテストが必要になります。 ほとんどのお客様はこのような問題を経験しませんが、問題に直面するかどうかは事前には把握できないため、新しい推奨のマイグレーション・プロセスがあります。

以前のバージョンの COBOL でコンパイルされたロード・モジュールのサポートは終了しますか?

いいえ。 サポート対象の z/OS レベルを使用している限り、z/OS LE を使用して既にコンパイルされたモジュールは実稼働環境で引き続きサポートされます。

マイグレーションする必要があるのはなぜですか?

それが必要なのは、 Enterprise COBOL V4 および V5 の EOS (サービス終了) が発表されているためです。 詳しくは、IBM ソフトウェア・ライフサイクルの Web サイトを参照してください。

実現可能性は、以下の点にあります。

- z/Architecture の十分な活用: Enterprise COBOL V6 は、 最新の IBM メインフレームを高度に最適化し、そのハードウェアを十分に活用することにより、パフォーマンスを改善します。
- アプリケーションの最新化: JSON、条件付きコンパイル、新しい組み込み関数、64 ビット COBOL アプリケーションなどの、新しい COBOL 言語の機能と生産性機能を活用できます。

マイグレーションの前に学習しておくべきベスト・プラクティスはありますか?

はい。実行時に COBOL データ項目に無効データが存在することが原因と考えられるマイグレーションの失敗を回避するために、 マイグレーションを支援するためにコンパイラーに追加されたマイグレーション・オプションを使用して、 2 コンパイルと 2 テストのプロセスを実行できます。 また、IBM DevOps ツール (以下の FAQ で紹介する) を使用して、自動的にビルドとテストを行うこともできます。 これにより、マイグレーションとその後のアプリケーションの保守とアップブレードは簡単になります。

マイグレーションを高速化するためのツールはありますか?

はい。IBM DevOps ツールを使用すると、アプリケーションの管理がずっと簡単になる可能性があります。 これには、COBOL V4 以前のコンパイラーから COBOL V6 へのマイグレーションで重要となる自動テストも含まれます。

アプリケーションのビルドとリリースを改善するための DevOps ツールには、次のようなものがあります。

- IBM UrbanCode Build
- IBM UrbanCode Deploy
- IBM Dependency Based Build
- IBM UrbanCode Velocity
- IBM UrbanCode Release

編集とデバッグを改善するための DevOps ツールには、次のようなものがあります。

- IBM Developer for z/OS
- IBM Z Open Development

自動テストとテスト効率のための DevOps ツールには、次のようなものがあります。

- IBM Z Open Unit Test
- IBM Rational Test Workbench
- Rational Test Virtualization Server

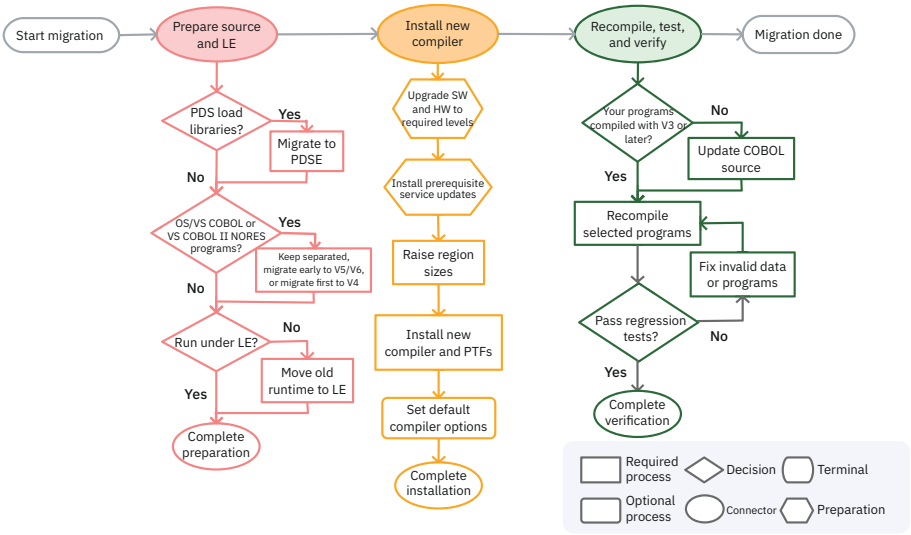
マイグレーションのためには、すべてを再コンパイルする必要がありますか？

新しいビルド・コンパイラーを使用してすべてのプログラムを再コンパイルすることは推奨されていますが、必須ではありません。 ただし、COBOL V6 でパフォーマンスが向上するのは、再コンパイルされたプログラムだけです。

ほとんどの COBOL ユーザーは、マイグレーション作業を行うとき、力を入れて開発しているアプリケーションやプログラムに焦点を当てます。 まず、アプリケーションの開発アクティビティーを分析して、再コンパイルの主な候補を判別することができます。 定期的に更新されていないアプリケーションやプログラムは、すぐに再コンパイルする必要はありません。 アプリケーションやプログラムのソースが変更されている場合は、そのアプリケーションやプログラムを再コンパイルする必要がありますが、 そうでない場合には、ただ IBM Automatic Binary Optimizer for z/OS (ABO) を使用してパフォーマンスを向上させることができます。

マイグレーション・プロセス全体の概要はどこで参照できますか?

以下のマイグレーション・プロセスの図は全体的なマイグレーション・プロセスを示しており、COBOL Migration Assistant はマイグレーションのガイダンスを提供します。



最初のマイグレーション・ターゲットとして CPU を最も消費している要素を選択するには、どうすればよいですか?

ビデオ「How to Identify Top CPU Consuming COBOL Modules」(https://mediacenter.ibm.com/media/1_ygabnyso) では、 ピーク使用量の識別、最も消費量の多いジョブとプログラムのペアの識別、最も消費量の多いモデルの識別、COBOL CSECTS のスキャン、という 4 つのステップで、 CPU を最も消費している要素を識別する方法が紹介されています。

マイグレーションについて詳しく知るために利用できる教育パッケージはありますか?

IBM Enterprise COBOL for z/OS V6 Migration Webinar および Performance Tuning Webinar は、どちらもお客様に無料で提供されています。 これらの Web セミナーは、Web キャストを介して定期的に開催されています。 COBOL Migration & Performance Webinars で今後の開催予定を確認し、参加登録をしてください。

マイグレーションについての過去の Web セミナーは Enterprise COBOL Migration Webinars on Box にあり、パフォーマンスについての過去の Web セミナーは Enterprise COBOL Performance Webinars on Box にあります。

マイグレーションの問題が発生した場合、どのようにしてサポートを受けることができますか?

COBOL compilers support portal で Case を開いて、マイグレーションの問題を報告することができます。

V5 または V6 にマイグレーションした後は、無効なデータについて心配する必要がありますか?

以前の無効データをすべて訂正し、 プログラムが使用するデータを検証した場合は、「いいえ」となります。まだ無効なデータがある場合は、将来のマイグレーションの後や、ARCH オプションまたは OPT オプションの変更したときに、動作に違いが生じる可能性があります。

実行時にプログラムで無効なデータが使用されていることが判明しても、アプリケーションの結果が OK であれば、その無効データの問題は修正しなくてもよいですか?

いいえ。 一部のお客様は、無効データのテストを行わないで COBOL V6.1 に正常にマイグレーションしましたが、 COBOL V6.2 に移行し、z14 ハードウェア用の ARCH(12) を使用してコンパイルすると、プログラムが無効データで異常終了するようになりました。 無効データは長年にわたり問題になることがあるので、訂正すべきです。

マイグレーション後に実動システムで COBOL プログラムのパフォーマンス向上を測定できますか?

はい、できます。 ただし、ハードウェア、ワークロード、およびコードはすべてマイグレーション時に変更される可能性があるので、そのようにすることは推奨されていません。 最適で最も正確な方法は、直後に COBOL V4 のモジュールと COBOL V6 のモジュールを、同じマシン上で同じデータを使用して測定することです。測定を行っておくか、ロード・モジュールを保存しておいてください。そうしない場合、マイグレーションが進行すると、以前の数値は取得できなくなります。

互換性

このトピックでは、互換性に関するよくある質問について説明します。

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 以外のもとでは、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 のもとでは、OS/VS COBOL プログラムはまったく実行できません。

CICS 以外のもとでは、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 のもとでは、VS COBOL II NORES プログラムはまったく実行できません。

非 CICS においても CICS においても、VS COBOL II RES プログラムと Enterprise COBOL プログラムの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 for z/OS* プログラミング・ガイド」を参照してください。

COBOL とアセンブラーの間の呼び出しの完全なリスト (Language Environment 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、299 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。

Enterprise COBOL V3 または V4 でコンパイルされたプログラムは、Enterprise COBOL V6 でコンパイルされたプログラムを呼び出すことができますか?

はい。Enterprise COBOL V3、V4、および V6 のプログラムを使用するアプリケーションを作成し、それらを正常に連動させることができます。Enterprise COBOL V6 プログラムは、OS/VS COBOL または VS COBOL II で NORES オプションを指定してコンパイルされたプログラムを除く、すべての COBOL プログラムを呼び出すことができます。 つまり、さまざまな 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をミックス・アンド・マッチすることができます。 また、既存のアプリケーションのいくつかのプログラムを Enterprise COBOL V6 で再コンパイルし、 残りのプログラムは以前のコンパイラーでコンパイルされたままにするということも可能です。

再コンパイルしないで COBOL プログラムのパフォーマンスを向上させるには、IBM Automatic Binary Optimizer for z/OS (ABO) を使用して、「*IBM Automatic Binary Optimizer for z/OS* ユーザーズ・ガイド」の『Eligible compilers』にリストされている COBOL バージョンでコンパイルされた古いロード・モジュールを最適化します。

プログラムを選択的に Enterprise COBOL に移行すること はできますか?

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 ?

はい。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、106 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。しかし、シグニチャー域のマップを使用して、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。 詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」で『*LIST* 出力の読み取り』を参照してください。

Enterprise COBOL V6 と以前のバージョンの違い

Enterprise COBOL V6 における変更は、主に以下のカテゴリーに分類されます。

- 前提ソフトウェア・レベルの変更
- COBOL ソース・コードの相違
- コンパイラー・オプションの変更
- 大きなプログラムにおけるシステム MEMLIMIT 設定への依存性
- ランタイムの変更
- ベンダー・ツールに影響する可能性がある変更

上記のリンクをご確認ください。 これらの変更について詳しくは、「*Enterprise COBOL for z/OS* 移行ガイド」を参照してください。

Enterprise COBOL V6 から削除された機能

削除された機能は、2000 年言語拡張と LABEL 宣言です。 詳しくは、190 ページの『Enterprise COBOL バージョン 5 およびバージョン 6 における COBOL ソース・コードの相違』を参照してください。

Enterprise COBOL でのコンパイル

このトピックでは、Enterprise COBOL を使用したコンパイルに関するよくある質問について説明します。

無効なデータを検出したり、パフォーマンスをチューニングしたりするためのコンパイラー・オプションはありますか?

無効なデータを検出するには、 まず初期コード変更と単体テストのために SSRANGE、NUMCHECK、PARMCHECK、INITCHECK、および OPT(0) を使用してコンパイルしてから、 品質保証テストと実動のた

めに NOSSRANGE、NONUMCHECK、NOPARMCHECK、および OPT(2) を使用して再コンパイルします (2 ステップ・コンパイル・プロセス)。

- 「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『SSRANGE』: SSRANGE を使用して、範囲外のストレージ参照を検査するコードを、コンパイラーで生成します。
- 「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『NUMCHECK』: NUMCHECK を使用して、送信データ項目として使用されているデータ項目を検証する追加のコードを、コンパイラーで生成します。
- 「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『PARMCHECK』: PARMCHECK を使用して、WORKING-STORAGE 内の最後の項目の後に追加のデータ項目をコンパイラーで生成し、その後、呼び出されたサブプログラムによって WORKING-STORAGE の終わりを超えたデータが破壊されていないかどうかを検査します。
- 「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『INITCHECK』: INITCHECK は、初期化されていないデータ項目をコンパイラーで検査し、 データ項目が初期化されずに使用されている場合にコンパイラーから警告メッセージを発行するために使用します。

Enterprise COBOL V6 では、パフォーマンス・チューニングのためのいくつかの新しいコンパイラー・オプションや大幅に変更されたコンパイラー・オプションが提供されており、それらはパフォーマンスに影響する可能性があります。最適なパフォーマンスを得るために推奨されるコンパイラー・オプションのセットは、OPT(2) と ARCH(x) です。パフォーマンス・チューニングについて詳しくは、「*Enterprise COBOL for z/OS* パフォーマンス・チューニング・ガイド」の『V6 を最大限に活用するためのコンパイラー・オプションのチューニング方法』を参照してください。

マイグレーションのベスト・プラクティスについて詳しくは、29 ページの『第 4 章 Enterprise COBOL V6 へのマイグレーションに関する推奨事項』を参照してください。

無効データについてのテストは実行時に行われますか?

ほとんどのテストは実行時に行われますが、特定のテストはコンパイル時に行われます。一般的に、問題が検出されるのは実行時です。データが 1 つのプログラム内に含まれる場合を除き、コンパイラーは無効データの出どころを認識しないためです。

2 ステップ・コンパイルは一回限りのプロセスとして実行されますか?

マイグレーション中にすべての無効データを検出して訂正できれば、2 ステップ・コンパイルは一回限りのプロセスとなります。 無効データの存続を許容する場合は、将来のコンパイラー・バージョンやオプションの変更により、その無効データの処理時に動作が変更される結果になる可能性があります。

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、16 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、16 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. 無償でオープン・ソースの COBOL アナライザーは、使用されたコンパイラー、コンパイラー・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できます。

   無料の COBOL アナライザーを http://cbttape.org/cbtdowns.htm からダウンロードします。 このアナライザーは、その Web ページ上では「*File # 321 COBOL Analyzer from Roland Schiradin & post processor*」 として示されています。
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 V6 のみ: メモリー不足に関するメッセージをコンパイラーが出した場合、何をすべきですか?

Enterprise COBOL V6 では、メッセージ「`IGYCB7145-U コンパイルを続行するためのメモリーがコンパイラーに不足しています (Insufficient memory in the compiler to continue compilation)`」を受け取った場合、コンパイル・ジョブで使用可能な領域サイズを大きくするか、またはご使用のシステムの MEMLIMIT 設定を大きくしてください。 1 GB 以上を既に使用していて、これでも十分ではない場合は、ご使用のプログラムが非常に大きいため、2 GB 境界より上のストレージが必要であると考えられます。 つまり、システムの MEMLIMIT 設定が 2 GB 以上でなければならない可能性があります。 システム・プログラマーに、現在の MEMLIMIT 設定を調べてもらってください。 非常に大きなプログラムでは、MEMLIMIT を 3 GB または 4 GB 以上に設定しなければならない場合があります。

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 標準および 2014 COBOL 標準に適合していますか?

Enterprise COBOL では、2002 COBOL 標準 および 2014 COBOL 標準の多くの部分がサポートされています。 詳しくは、『*Enterprise COBOL バージョン 3 以降に実装されている 2002/2014 COBOL 標準機能*』(*Enterprise COBOL for z/OS* 言語解説書) を参照してください。

2 GB 境界より上で実行されるようにパラメーターを指定してプログラムをコンパイルする方法はありますか?

z/OS には、高水準言語での 2 GB 境界より上での実行 (RMODE 64) のサポートはありませんが、Enterprise COBOL V6.3 には LP(64) コンパイラー・オプションがあり、プログラムでこれを使用することにより 2 GB 境界より上のデータをアドレッシングすること (AMODE64) が可能です。 LP について詳しくは、*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 プログラムのバインド (リンク・エディット)

このトピックでは、Enterprise COBOL プログラムのバインド (リンク・エディット) に関するよくある質問について説明します。

- 259 ページの『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?』
- 259 ページの『Enterprise COBOL V5 または V6 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?』

- 259 ページの『Enterprise COBOL V4 から V5 または V6 の間に、コンパイラー生成シンボルの変更はありますか?』

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 以降のバージョン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 または V6 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。

Enterprise COBOL V4 から V5 または V6 の間に、コンパイラー生成シンボルの変更はありますか?

バインダー・ステートメント (CHANGE など) を使用して V4 コンパイラー生成シンボルを検査する場合、AMBLIST ユーティリティーを使用して、COBOL V5 および V6 の GOFF 実行可能プログラムで生成されるシンボルについて理解しておく必要があります。 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」の『AMBLIST』を参照してください。

V5 コンパイラーおよび V6 コンパイラーは一般に、新しい CSECT (C_WSA や C_CODE など) を生成します。これらは通常、プログラムごとに 3 つの関連するラベル・シンボルを生成します。例えば、プログラム P でラベル・シンボル P、P#C、および P#S が生成されます。 バインダー・ステートメントを使用して、このようなシンボルのいずれかを変更する場合、同様のステートメントでそれらすべてを変更する必要があります。

Language Environment ランタイム・オプション

このトピックでは、Language Environment ランタイム・オプションに関するよくある質問とその回答について説明します。

- 259 ページの『Enterprise COBOL は WORKING-STORAGE に HEAP を使用しますか?』
- 260 ページの『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 ?』
- 260 ページの『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 ?』
- 260 ページの『Enterprise COBOL は STACK ストレージを使用しますか ?』
- 260 ページの『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 ?』
- 260 ページの『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 ?』

Enterprise COBOL 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。

- Enterprise COBOL V4.2 以前のリリースでコンパイルされた
- DATA(24) コンパイラー・オプションを使用してコンパイルされた
- CICS での実行
- COBOL プログラム (COBOL 5.1.0 を除く) およびアセンブラーのみを含むプログラム・オブジェクトにおける COBOL V5.1.1 以降のプログラム。 そのプログラム・オブジェクト内には言語環境プログラムの言語間呼び出しがない。また、COBOL V5.1.0 プログラムもない。
- メインエントリー・ポイントが COBOL V5 のプログラム・オブジェクト内にある、COBOL V5 プログラムである。 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できる。そ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 (メイン・エントリー・ポイントではない場合も含め) では、WORKING-STORAGE はヒープ・ストレージから割り振られる。
- COBOL V6.1 以降のプログラム

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許容されるエラー、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

このトピックでは、サブシステムに関するよくある質問とその回答について説明します。

- 261 ページの『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 ?』

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・プログラムは、Enterprise COBOL V6 プログラムで引き続き正常に作動しますか?

現在 Enterprise COBOL V4 以前のプログラムと連携しているアセンブラー・プログラムは、 Enterprise COBOL V6 プログラムで引き続き正常に作動します。 それらは LE 準拠である必要はありません。

Enterprise COBOL V6 には、Enterprise COBOL V4 ほどには、LE 準拠でないアセンブラーとの互換性がないのでしょうか?

それは、アセンブラー・プログラムの実行内容によって異なります。 アセンブラー・プログラムが COBOL 実行可能ファイルで情報を検索する (COBOL プログラム・レイアウトに依存する) 場合や、アセンブラー・プログラムが LE 対応である (LE で正常に実行される) ものでない場合は、問題が発生することがあります。多数のアセンブラーを使用している場合、発生する可能性のある問題を調査してください。 アセンブラー・プログラムは、LE 準拠である (CEEENTRY と CEETERM マクロを使用している) 必要はありませんが、LE 対応でなければなりません。

z/OS

COBOL は 64 ビットの z/OS で稼働しますか？

はい。COBOL は、COBOL プログラムでの AMODE 64 アドレッシングをサポートするようになったため、COBOL プログラム内で AMODE 64 の機能を十分に活用することができます。AMODE 31 であっても、AMODE 64 z/OS に移行するだけで、その利点の一部を得ることができます。AMOE 64 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 AMODE 64 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

このトピックでは、パフォーマンスに関するよくある質問とその回答について説明します。

- 262 ページの『OS/VS COBOL から Enterprise COBOL に変換すると、パフォーマンスは向上しますか?』
- 263 ページの『Enterprise COBOL V6 を使用して再コンパイルした後のパフォーマンス向上はどのようにして確認できますか?』
- 263 ページの『マイグレーションの前と後に COBOL プログラムのパフォーマンスを測定する最良の方法はどれですか?』

OS/VS COBOL から Enterprise COBOL に変換すると、パフォーマンスは向上しますか?

はい。Enterprise COBOL V5 および V6 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。

Enterprise COBOL V6 使用時の重要なパフォーマンス利点およびチューニング考慮事項について詳しくは、「*Enterprise COBOL for z/OS* パフォーマンス・チューニング・ガイド」を参照してください。

Enterprise COBOL V6 を使用して再コンパイルした後のパフォーマンス向上はどのようにして確認できますか?

パフォーマンス向上を確認する唯一の方法は、 マイグレーションの前と後でパフォーマンスを測定することです。 したがって、パフォーマンスの向上を確認する必要がある場合は、ビルド・コンパイラーをマイグレーションする前に、 現行の Enterprise COBOL V4 のモジュールをバックアップすることをお勧めします。 IBM Automatic Binary Optimizer for z/OS (ABO) によるマイグレーション、 必要な再コンパイル、または最適化が完了したら、テスト環境をセットアップし、実際の代表的なワークロードを準備し、 バックアップした Enterprise COBOL V4 のモジュールと、新しい COBOL V6 のモジュールまたは ABO で最適化したモジュールでパフォーマンスを測定します。

マイグレーションの前と後に COBOL プログラムのパフォーマンスを測定する最良の方法はどれですか?

それは、アプリケーションおよびワークロードによって異なります。

- トランザクション・アプリケーションの場合は、スループットを見て、一定の期間にアプリケーションが処理できるトランザクション数 (1 秒当たりのトランザクション数など) を確認します。
- バッチ・アプリケーションの場合は、バッチ・ジョブの CPU 時間または経過時間を測定します。

マイグレーションの前と後の COBOL プログラムのパフォーマンスを測定するには、サンプリングを使用してパフォーマンスを測定する IBM Application Performance Analyzer for z/OS、 Compuware STROBE、または Computer Associates (CA) MAT などのプロファイル・ツールを使用します。 IBM RMF を使用し、SMF レコードを使用してジョブ・リソースの使用量を測定することもできます。 RMF およびその他のツールの使用方法について詳しくは、ホワイト・ペーパー COBOL Applications: Techniques to make them more Efficient を参照してください。 パフォーマンス・チューニングでは、CPU 時間を最も多く消費しているホット・スポットと特定のモジュールを見つける必要があります。 マイグレーション前の COBOL V4 のパフォーマンスとマイグレーション後の COBOL V6 のパフォーマンスを比較することはベスト・プラクティスではありません。 ハードウェア、ワークロード、およびコードはすべてマイグレーション中に変更される可能性があるためです。 比較は、同じマシン上で、同じ時刻に、同じ入力データを使用して行う必要があります。

サービス

アプリケーションについて IBM サービス・サポートを得るためには 、すべてのプログラムを再コンパイルする必要がありますか ?

プログラムが、サポートされ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、22 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 以降の SDK

Java 6 以降を使用して既存の 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。

このようなアプリケーションを Java 6 以降で実行するには、以下の手順を実行します。

1. Enterprise COBOL V5.2 以降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 以降の `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 6 章 OS/VS COBOL ソース・プログラムのアップグレード』
- 91 ページの『第 8 章 VS COBOL II ソース・プログラムのアップデート』
- 101 ページの『第 10 章 IBM COBOL ソース・プログラムのアップグレード』
- 147 ページの『第 12 章 Enterprise COBOL バージョン 3 から プログラムのアップグレード』
- 161 ページの『第 14 章 Enterprise COBOL バージョン 4 からのアップグレード』

キー:

X
このワードはプロダクトで予約されています。

-
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 *50.* 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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
ALLOCATE[1]	X Enterprise COBOL V6.1 以降でのみ予約済み	-	-	-
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	-
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	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
BYTE-LENGTH	X Enterprise COBOL V6.3 以降でのみ予約済み	-	-	-
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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
COLUMNS	RFD	-	-	-
COM-REG	X	X	X	-
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 COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	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 COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	RFD	-
COMPUTE	X	X	X	X
CONDITION	RFD	-	-	-
CONFIGURATION	X	X	X	X
CONSOLE	-	-	-	X
CONSTANT	RFD	-	-	-
CONTAINS	X	X	X	X

表 *50.* 予約語の比較 *(続き)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
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	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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[1]	X Enterprise COBOL V6.1 以降でのみ予約済み	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
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	-	-	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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 COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	-	-
END-IF	X	X	X	-
END-INVOKE	X	X	-	-
END-JSON[1]	X Enterprise COBOL V6.1 以降でのみ予約済み	-	-	-
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	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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	-
END-XML[1]	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 COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	-	-
EXECUTE	X	X COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	-	-

表 *50.* 予約語の比較 *(続き)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
EXHIBIT	-	-	-	X
EXIT	X	X	X	X
EXTEND	X	X	X	X
EXTERNAL	X	X	X	-
FACTORY	X	X COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	-	-
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[1]	X Enterprise COBOL V6.1 以降でのみ予約済み	RFD	RFD	-
FROM	X	X	X	X
FUNCTION	X	X	-	-
FUNCTION-ID	RFD	-	-	-
FUNCTION-POINTER[1]	X	-	-	-
GENERATE	UNS	UNS	UNS	X

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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[1]	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	-	-

表 *50.* 予約語の比較 *(続き)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
IS	X	X	X	X
JAVA	X Enterprise COBOL V6.3 以降でのみ予約済み	-	-	-
JNIENVPTR[1]	X	-	-	-
JSON[1]	X Enterprise COBOL V6.1 以降でのみ予約済み	-	-	-
JSON-CODE[1]	X Enterprise COBOL V6.1 以降でのみ予約済み	-	-	-
JSON-STATUS[1]	X Enterprise COBOL V6.2 以降でのみ予約済み	-	-	-
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	X Enterprise COBOL V6.3 以降でのみ予約済み	UNS	UNS	X
LIMITS	UNS	UNS	UNS	X
LINAGE	X	X	X	X
LINAGE-COUNTER	X	X	X	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
NATIONAL[1]	X	-	-	-
NATIONAL-EDITED[1]	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	-

表 *50.* 予約語の比較 *(続き)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
NULLS	X	X	X	-
NUMBER	UNS	UNS	UNS	X
NUMERIC	X	X	X	X
NUMERIC-EDITED	X	X	X	-
OBJECT	X	X	-	-
OBJECT-COMPUTER	X	X	X	X
OBJECT-REFERENCE	RFD	-	-	-
OCCURS	X	X	X	X
OF	X	X	X	X
OFF	X	X	X	X
OMITTED	X	X	X	X
ON	X	X	X	X
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
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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
POINTER	X	X	X	X
POINTER-24	RFD	-	-	-
POINTER-31	RFD	-	-	-
POINTER-32	X Enterprise COBOL V6.3 以降でのみ予約済み	-	-	-
POINTER-64	RFD	-	-	-
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

表 *50.* 予約語の比較 *(続き)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
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	-	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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 COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	-	-
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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
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
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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
TRACK-AREA	-	-	-	X
TRACK-LIMIT	-	-	-	X
TRACKS	-	-	-	X
TRAILING	X	X	X	X
TRANSFORM	-	-	-	X
TRUE	X	X	X	-
TYPE	X	X COBOL (OS/390 および VM 版) V2.2 以降でのみ予約済み	-	-
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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
UTF-8	X Enterprise COBOL V6.3 以降でのみ予約済み	-	-	-
VAL-STATUS	RFD	-	-	-
VALID	RFD	RFD	RFD	-
VALIDATE	RFD	RFD	RFD	-
VALIDATE-STATUS	RFD	-	-	-
VALUE	X	X	X	X
VALUES	X	X	X	X
VARYING	X	X	X	X
VOLATILE[1]	X Enterprise COBOL V5.2 以降でのみ予約済み	-	-	-
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[1]	X	-	-	-
XML-CODE[1]	X	-	-	-
XML-EVENT[1]	X	-	-	-
XML-INFORMATION[1]	X Enterprise COBOL V4.2 以降でのみ予約済み	-	-	-
XML-NAMESPACE[1]	X Enterprise COBOL V4.1 以降でのみ予約済み	-	-	-

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
XML-NAMESPACE-PREFIX[1]	X Enterprise COBOL V4.1 以降でのみ予約済み	-	-	-
XML-NNAMESPACE[1]	X Enterprise COBOL V4.1 以降でのみ予約済み	-	-	-
XML-NNAMESPACE-PREFIX[1]	X Enterprise COBOL V4.1 以降でのみ予約済み	-	-	-
XML-NTEXT[1]	X	-	-	-
XML-SCHEMA[1]	X Enterprise COBOL V4.2 以降でのみ予約済み	-	-	-
XML-TEXT[1]	X	-	-	-
ZERO	X	X	X	X
ZEROES	X	X	X	X
ZEROS	X	X	X	X
_	X Enterprise COBOL V4.2 以降でのみ予約済み	-	-	-
<	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

表 *50.* 予約語の比較 *(*続き*)*

予約語	Enterprise COBOL	IBM COBOL	VS COBOL II	OS/VS COBOL
*>[1]	X Enterprise COBOL V5.1 以降でのみ予約済み	-	-	-
::	RFD	-	-	-

注:

1. これは IBM COBOL 以降に追加された新規予約語です。

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)
- FLAGMIG コンパイラー・オプション (VS COBOL II、COBOL for MVS & VM、COBOL for OS/390 & VM)
- FLAGMIG4 コンパイラー・オプション (現行サービスが適用された Enterprise COBOL V4.2)
- 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 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 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 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 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 節の変更
- RESERVE 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 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 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 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 ステートメント (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 ステートメント
- LINE-COUNTER、PAGE-COUNTER、および PRINT-SWITCH 特殊レジスター
- SPECIAL NAMES における英数字リテラル IS 簡略名
- FD の REPORT 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。294 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 ステートメント - 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 節
- 編成パラメーター"I"
- TRACK-AREA 節
- START ステートメントの USING KEY 節

BDAM

- ACTUAL KEY 節
- APPLY RECORD-OVERFLOW (BDAM)
- BDAM ファイル宣言
- I/O ステートメント - 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 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 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

FLAGMIG コンパイラー・オプション

FLAGMIG オプションは、Enterprise COBOL の下でコンパイルを行うために変換しなければならないソース・ステートメントを識別する場合に役立ちます。 FLAGMIG は、CMPR2 オプションをサポートする Enterprise COBOL より前のコンパイラーで使用できます。 Enterprise COBOL V4.2 を既に使用している場合は、Enterprise COBOL V5 または V6 への移行に役立つ FLAGMIG4 オプション (現行サービスが適用された Enterprise COBOL V4.2 で使用可能) を使用することをお勧めします。

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、292 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)』、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。

Enterprise COBOL への移行に役立つ FLAGMIG オプションおよび CMPR2 オプションの使用法について詳しくは、106 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。

Enterprise COBOL V4.2 を既に使用していて Enterprise COBOL V5 または V6 への移行を望む場合は、FLAGMIG4 オプションを使用して、Enterprise COBOL V5 または V6 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ます。 詳細については、15 ページの『FLAGMIG4 コンパイラー・オプション』を参照してください。

FLAGMIG4 コンパイラー・オプション

FLAGMIG4 オプションは、Enterprise COBOL V5 または V6 へのマイグレーションに役立ちます。 FLAGMIG4 は、Enterprise COBOL V4.2 (APAR PM93450 用 PTF インストール済み) で使用可能です。 また、APAR PI12240、PI26838、および PI58762 の PTF には FLAGMIG4 オプションに対する更新が含まれているため、これらの PTF もインストールすることをお勧めします。

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 または V6 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。

注 : COBOL V5 および V6 におけるソース・コードの変更が COBOL 言語機能で使用されることはほとんどないため、それらの変更は COBOL ユーザーの 99% には影響しません。

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

IBM Application Discovery and Delivery Intelligence および IBM Rational Asset Analyzer for System z

IBM Application Discovery and Delivery Intelligence および IBM Rational Asset Analyzer for System z にはツールが用意されていて、企業資産の目録を生成したり、コード変更に必要となる相対労力の指標を返したりできます。

詳しくは、以下を参照してください。

- IBM Application Discovery and Delivery Intelligence: https://www.ibm.com/products/app-discovery-and-delivery-intelligence
- IBM Rational Asset Analyzer for System z: https://www.ibm.com/support/knowledgecenter/SS3JHP/raa_family_welcome.html

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、APAR PM86253 の PTF によって、Enterprise COBOL V5.1 の予約語変換用に更新されています。V5.2 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。V6.1 では、APAR PI55980 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。例えば、次のように指定します。

```
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。

例えば、次のように指定します。

```
EXEC CICS TWA(BLL).
```

これは、次のように置き換えられます。

```
    EXEC CICS ADDRESS 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

293 ページの表 51 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 *51. 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

表 *51. 1* 次 *BLL* を扱う *COBOL* ステートメント *(*続き*)*

元のソース	変換後のソース
MOVE 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

COBOL 報告書作成プログラム・プリコンパイラーを使用して、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。

IBM は、SPC Systems から COBOL 報告書作成プログラム・プリコンパイラーのライセンス交付を受け、これを以下のプログラム番号で販売しています。

- 5798-DYR: COBOL 報告書作成プログラム・プリコンパイラーおよびライブラリー
- 5798-DZX: COBOL 報告書作成プログラムのライブラリーのみ

注 : Enterprise COBOL V5.1 以降では、 コンパイラー・アーキテクチャーの変更により、COBOL 報告書作成プログラム・プリコンパイラー V1.6.01 以降が必要です。 COBOL 報告書作成プログラムに対する更新は、SPC Systems のサポート契約に従って、SPC Systems から入手する必要があります。 プログラム・サービスおよび技術サポートについては、SPC Systems (https://www.spc-systems.com) にお問い合わせください。

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* と *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。

無料でオープン・ソースの COBOL アナライザー

無料でオープン・ソースの COBOL Analyzer は、使用されたコンパイラー、コンパイラー・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。

無料の COBOL アナライザーを http://cbttape.org/cbtdowns.htm からダウンロードします。 このアナライザーは、その Web ページ上では「*File # 321 COBOL Analyzer from Roland Schiradin & post processor*」 として示されています。

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変換
- 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、71 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 298 ページの表 52 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 *52. 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
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

表 52. *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**

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 299 ページの表 53 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 53. *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

表 53. *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。 *(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**	**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 [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変換

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 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。 いくつかの解決法が考えられます。

- RENT でコンパイルする。
- 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ロード・モジュ ールを SVC ロードし、SVC 削除することができます。

- 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

制約事項 : Language Environment は SVC 削除を追跡できませんが、プログラムに関連付けられているストレージ、制御域、およびファイル入出力域を解放することはできます。 プログラムでクローズされていないファイルや、プログラムに対して割り振られているストレージはいずれも、Language Environment や COBOL ライブラリーでは適切には解放されません。 このようなリソースに後からアクセスすると、予測不能な結果になる可能性があります。 また、Debug Tool は、アセンブラーが SVC 削除を使用して削除したロード・モジュールに含まれている COBOL プログラムをサポートしていません。

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ロード・モジュールを SVC ロードすることができますが、SVC 削除することはできません。

- RENT オプションを指定してコンパイルされた VS COBOL II プログラム
- RENT オプションを指定してコンパイルされた IBM COBOL プログラム
- Enterprise COBOL プログラム

アセンブラー・プログラムが、3 種類のプログラムを含んでいるロード・モジュール を SVC 削除すると、予測できない結果になる可能性があります。

通常、COBOL プログラムが含まれているロード・モジュールをロードしたり削除したりしなければならないアセンブラー・プログラムの場合は、以下のいずれかの方式が推奨されます。 これは、RENT オプションまたは NORENT オプションが指定された COBOL プログラムに適用されます。

- 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。

COBOL V5 および V6 プログラムはプログラム・オブジェクトであるため、COBOL V5 または V6 プログラムをロードするためには、アセンブラー・プログラムは CEELOAD ではなく CEEFETCH を使用する必要があります。

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 または V6 を呼び出す、または Enterprise COBOL V5 または V6 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。

「*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 または V6 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 または V6 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 または V6 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。

現在の COBOL V5 または V6 プログラムを呼び出したプログラムの名前を見つける

LE サービス CEETBCK を使用すると、COBOL V5 または V6 プログラムの実行時に呼び出し側プログラムの名前を見つけることができます。詳しくは、「*z/OS Language Environment Vendor Interfaces*」 を参照してください。

付録 E オプションの比較

次の表で、Enterprise COBOL V5 および V6 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 および V6 のオプションの詳細については、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の 『コンパイラー・オプション』を参照してください。

キー:

X
コンパイラー・オプションを使用できます。

-
コンパイラー・オプションを使用できません。

表 *54.* オプションの比較

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
ADATA	-	-	X	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	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	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。 • Enterprise COBOL V5.1、V5.2、および V6.1 では AFP(VOLATILE) がデフォルトです。 • Enterprise COBOL V6.2 では、AFP(NOVOLATILE) がデフォルトです。
ANALYZE	-	-	X COBOL (OS/390 および VM 版) V2.1 以降でのみ使用可能	-	-	-	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
ALOWCBL	-	X	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。
APOST/QUOTE	X	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 QUOTE 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	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(8) がデフォルトです。
ARITH	-	-	X	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 節、固定小数点数値リテラル、NUMVAL、NUMVAL-C、および NUMVAL-F 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 節、固定小数点数値リテラル、NUMVAL、NUMVAL-C、および NUMVAL-F 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	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	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 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
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	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	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。

表 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
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。 CMPR2 オプションは Enterprise COBOL V4 ではサポートされなくなりましたが、V3 以前のバージョンからの移行を容易にするために、通知メッセージや警告メッセージが発行されるものの、使用は容認されていました。Enterprise COBOL V5 および V6 では、CMPR2 オプションは使用できなくなりました。 このバージョンで CMPR2 オプションを指定すると、エラー・メッセージが発行されます。
CODEPAGE	-	-	-	X	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	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
COPYLOC	-	-	-	-	-	X Enterprise COBOL V6.1 (サービス適用済み) 以降でのみ使用可能	COPYLOC コンパイラー・オプションは、ライブラリー・フェーズでコピー・メンバーを検索する追加ロケーションとして PDSE (または PDS) データ・セットあるいは z/OS UNIX ディレクトリーを追加する場合に使用します。
COPYRIGHT	-	-	-	-	X	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ステートメント実行サマリーを作成します。各ステートメント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。
CURRENCY	-	-	X	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 節の両方が使用されると、CURRENCY SIGN 節に指定された記号が、PICTURE 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	-	X	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。

表 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
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	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	-	X	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。
DECK	X	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。
DEFINE	-	-	-	-	-	X Enterprise COBOL V6.1 (サービス適用済み) 以降でのみ使用可能	これは、PARAMETER 句を持つ DEFINE ディレクティブを使用して、プログラムに定義されているコンパイル変数にリテラル値を割り当てます。
DIAGTRUNC	-	-	X	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	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
DLL	-	-	X	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	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	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。
EXIT	-	X	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	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。

表 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
FASTSRT	-	X	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	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。
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 V4.2 (サービス適用済み) でのみ使用可能	-	-	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 またはバージョン 6 では異なる形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、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。また、APAR PI12240、PI26838、および PI58762 の PTF には FLAGMIG4 オプションに対する更新が含まれているため、これらの PTF もインストールすることをお勧めします。 **注 :** COBOL V5 および V6 におけるソース・コードの変更が COBOL 言語機能で使用されることはほとんどないため、それらの変更は COBOL ユーザーの 99% には影響しません。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
FLAGSTD	-	X	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	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。
INITCHECK	-	-	-	-	-	X Enterprise COBOL V6.1 (サービス適用済み) 以降でのみ使用可能	未初期化データ項目を検査し、それらが初期化されていないまま使用されている場合に警告メッセージを出すかどうかを制御します。
INITIAL	-	-	-	-	-	X Enterprise COBOL V6.2 (サービス適用済み) 以降でのみ使用可能	これにより、プログラムとそのネスト・プログラムは、IS INITIAL 節が PROGRAM-ID 段落に指定されたかのように動作します。

表 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
INLINE	-	-	-	-	-	X Enterprise COBOL V6.1 (サービス適用済み) 以降でのみ使用可能	ソース・プログラム内の PERFORM ステートメントによって参照されているプロシージャーのインライン化 (段落またはセクション) のコンパイラー使用を制御します。 NOINLINE を指定すると、コンパイラーは、PERFORM ステートメントによって参照されているプロシージャーをインライン化しません。
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	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。
INVDATA	-	-	-	-	-	Enterprise COBOL V6.2 (サービス適用済み) 以降でのみ使用可能	これは、USAGE DISPLAY データ項目および PACKED-DECIMAL データ項目におけるデータが有効であるかどうかをコンパイラーに伝え、そのデータが有効ではない場合に、コンパイラーはどのように動作すべきかをコンパイラーに指示します。
LANGUAGE	-	X	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
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	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。
LINECOUNT	-	X	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	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。
LP	-	-	-	-	-	X Enterprise COBOL V6.3 以降でのみ使用可能	関連する言語機能を有効にして、31 ビットまたは 64 ビットのどちらのプログラムを生成するかを指示します。LP(32) がデフォルトです。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
MAP	-	X	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 および V6 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 V5 および V6 で得られます。
MAXPCF	-	-	-	-	X	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(100000) です。
MDECK	-	-	-	X	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。
NAME	X	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
NSYMBOL	-	-	-	X	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	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。
NUMCHECK	-	-	-	-	-	X Enterprise COBOL V6.1 (サービス適用済み) 以降でのみ使用可能	送信データ項目として使用するゾーン 10 進データ項目およびパック 10 進データ項目に対して、暗黙的な数値のクラス・テストを生成するかどうか、また 2 進データ項目に対して SIZE ERR0R 検査を生成するかどうかを制御します。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。
NUMCLS	-	X	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、*Enterprise COBOL for z/OS* カスタマイズ・ガイド を参照してください。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
NUMPROC	-	X	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 および V6 ではサポートされません。 NUMPROC(MIG) でコンパイルしたプログラムを Enterprise COBOL V6 にマイグレーションするには、NUMPROC(PFD) にマイグレーションするときに役立つ NUMCHECK コンパイラー・オプションの使用を検討してください。 1. プログラムを NUMCHECK(ZON,PAC) および NUMPROC(PFD) でコンパイルします。 2. 適切な大きさの入力データで完全なリグレッション・テストを実行します。 アプリケーションが NUMCHECK メッセージも NUMCHECK 異常終了も引き起こさなければ、NUMPROC(PFD) および NONUMCHECK による実動用のコンパイルを安全に実行できます。これは無効なデータの問題を解決するだけではありません。NUMPROC(PFD) は、NUMPROC コンパイラー・オプションのための最も有効な設定です。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。 NUMPROC(NOPFD) がデフォルトです。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
OBJECT	-	X	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。OBJECT 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。
OFFSET	-	X	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	X	コンパイラー・オプションが SYSOPTFDD 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	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 および V6 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	-	X	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 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
PARMCHECK	-	-	-	-	-	X Enterprise COBOL V6.1 (サービス適用済み) 以降でのみ使用可能	WORKING-STORAGE にある最後の項目の後に追加のデータ項目を生成するよう、コンパイラーに指示します。 次にこのバッファー・データ項目は、呼び出されたサブプログラムが WORKING-STORAGE の端を超えてデータを破壊していないかどうかを検査するために実行時に使用されます。 NOPARMCHECK がデフォルトです。
PGMNAME	-	-	X	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	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
QUOTE	X	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 QUOTE のどちらが有効であるかには関係なく、リテラルを 引用符とアポストロフィのどちらで区切っても構いません。QUOTE を使用すると、形象定数 QUOTE/QUOTES は 1 つ以上のアポストロフィ (') 文 字を表します。
RES	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	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	-	-	X	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。
RULES	-	-	-	-	X Enterprise COBOL V5.1 (サービス適用済み) 以降でのみ使用可能	X	これは、コンパイル時に特定タイプのソース・コードにフラグを立てることにより、プログラムを向上させるためにプログラムに関する情報をコンパイラーに要求します。
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。

表 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
SEQUENCE	-	X	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	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	-	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。 SIZE(MAX) は、Enterprise COBOL V5.1 ではサポートされていません。SIZE オプションは、Enterprise COBOL V5.2 ではサポートされていません。
SOURCE	X	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(DEC) がデフォルトです。 最新サービスがインストールされた Enterprise COBOL V6.3 以降では、新しいサブオプション HEX および DEC が追加されています。SOURCE(DEC) が有効な場合、ソースのリストの行番号は 10 進形式になります。 SOURCE(HEX) が有効な場合、ソースのリストの行番号は 16 進形式になります。
SPACE	X	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
SQL	-	-	X	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。
SQLCCSID	-	-	-	X	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。 SQLCCSID がデフォルトです。 is the default.
SQLIMS	-	-	-	-	X	X	IMS SQL コプロセッサー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。
SSRANGE	-	X	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。 Enterprise COBOL V6.2 では、範囲検査が失敗したときの COBOL プログラムのランタイム動作を制御するために、新しいサブオプション MSG および ABD が追加されています。 Enterprise COBOL V6.1 で、コンパイラーによる参照変更長の検査方法を制御する新しいサブオプション ZLEN および NOZLEN が追加されました。 NOSSRANGE がデフォルトです。
STGOPT	-	-	-	-	X	X	ストレージ最適化を制御します。NOSTGOPT がデフォルトです。
SUPPRESS	-	-	-	-	-	X	COPY ステートメントの SUPPRESS 句を無視するかどうかを制御します。
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 54. オプションの比較 (続き)

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
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。
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	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
TEST	X	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, NOSOURCE, NOSEPARATE) がデフォルトです。 詳しくは、「*Enterprise COBOL for z/OS* プログラミング・ガイド」の『TEST』を参照してください。
THREAD	-	-	-	X	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。
TRUNC	X	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 for z/OS* プログラミング・ガイド」を参照してください。
TUNE	-	-	-	-	-	X Enterprise COBOL V6.3 (サービス適用済み) 以降でのみ使用可能	実行可能プログラムが、どのアーキテクチャーを対象にして最適化されるのかを指定します。 ARCH が指定されている場合、デフォルトの TUNE レベルは ARCH レベルと一致する必要があります。 ARCH を指定しない場合は、ARCH と TUNE は両方ともデフォルトで 8 に設定されます。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
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	-	X	X	X	X	X	プログラム内のすべてのステートメント・タイプの相互参照リストを生成します。 NOVBREF がデフォルトです。
VBSUM	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。
VLR	-	-	-	-	X Enterprise COBOL V5.1 (サービス適用済み) 以降でのみ使用可能	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。
VSAMOPENFS	-	-	-	-	X Enterprise COBOL V5.2 (サービス適用済み) 以降でのみ使用可能	X	ファイル整合性検査の確認を必要とする、正常に実行された VSAM OPEN ステートメントから報告されるユーザー・ファイル状況に作用します。
WORD	-	X	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。

表 *54.* オプションの比較 *(*続き*)*

オプション	OS/VS COBOL	VS COBOL II	IBM COBOL	Enterprise COBOLV3 と V4	Enterprise COBOL V5	Enterprise COBOL V6	使用上の注意
XMLPARSE	-	-	-	X Enterprise COBOL V4.1 以降でのみ使用可能	X Enterprise COBOL V5.1 (サービス適用済み) 以降でのみ使用可能	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	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。
ZONECHECK	-	-	-	-	X Enterprise COBOL V5.1 (サービス適用済み) 以降でのみ使用可能	X	送信データ項目として使用するゾーン 10 進データ項目に対して IF NUMERIC クラス・テストを生成するようコンパイラーに指示します。 Enterprise COBOL V6.1 (サービス PTF 適用済み)、および V6.2 以降では、ZONECHECK は非推奨ですが、互換性のために使用は許容されています。代わりに、NUMCHECK(ZON) の使用を検討してください。 詳しくは、*Enterprise COBOL for z/OS* プログラミング・ガイド内の *NUMCHECK* を参照してください。
ZWB	X	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。

付録 F コンパイラー限界値の比較

以下の表は、Enterprise COBOL V5 および V6、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 および V6	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 から 7	0 から 7	0 から 7	0 から 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				
SELECT ファイル名 . . .	最大 65,535 のファイル名を外部名に割り当て可能	最大 65,535 のファイル名を外部名に割り当て可能	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

言語エレメント	Enterprise COBOL V5 および V6	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
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	LP(32) が有効な場合: 999,999,999 -B LP(64) が有効な場合: 2,147,483,646 -B	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: 18	18
PICTURE 節、数値編集項目の文字位置	249	249	249	127
PICTURE 記号複製 ( )	LP(32) が有効な場合: 999,999,999 LP(64) が有効な場合: 2,147,483,646	134,217,727	16,777,215	99,999

言語エレメント	Enterprise COBOL V5 および V6	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
PICTURE 記号複製 ( )、クラス DBCS 項目	LP(32) が有効な場合: 499,999,999 LP(64) が有効な場合: 1,073,741,823	67,108,863	8,388,607	N/A
PICTURE 記号複製 ( )、クラス国別項目	LP(32) が有効な場合: 499,999,999 LP(64) が有効な場合: 1,073,741,823	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	LP(32) が有効な場合: 999,999,999 LP(64) が有効な場合: 2,147,483,646	134,217,727	16,777,215	32,767
OCCURS 整数	LP(32) が有効な場合: 999,999,999 LP(64) が有効な場合: 2,147,483,646	134,217,727	4,194,303	65,535
テーブルのサイズ	LP(32) が有効な場合: 999,999,999 LP(64) が有効な場合: 2,147,483,646	134,217,727	8,388,607	32,767
テーブル・エレメントのサイズ	LP(32) が有効な場合: 999,999,999 LP(64) が有効な場合: 2,147,483,646	134,217,727		
ASC または DES KEY . . . (OCCURS 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

言語エレメント	Enterprise COBOL V5 および V6	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
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: 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	LP(32) が有効な場合: 2,147,483,646-B LP(64) が有効な場合: 無制限、マシンの使用可能な 64 ビット・アドレッシング容量まで	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	LP(32) が有効な場合: 2,147,483,646-B LP(64) が有効な場合: 無制限、マシンの使用可能な 64 ビット・アドレッシング容量まで	134,217,727-B	134,217,727-B	N/A
LINKAGE SECTION				
合計サイズ	LP(32) が有効な場合: 2,147,483,646-B LP(64) が有効な場合: 無制限、マシンの使用可能な 64 ビット・アドレッシング容量まで	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

言語エレメント	Enterprise COBOL V5 および V6	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
プロシージャー名	1,048,575[1]	1,048,575[1]	1,048,575[1]	64-KB[1]
行当たりのステートメントの数 (FDUMP/TEST)	7	7	7	7
ステートメントごとの添え字付きデータ名	32,767	32,767	32,767	511
ADD 識別子 . . .	N/L	N/L	N/L	N/L
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

言語エレメント	Enterprise COBOL V5 および V6	その他の Enterprise COBOL バージョン	IBM COBOL および VS COBOL II	OS/VS COBOL
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
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 for z/OS* 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 for z/OS* 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 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 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 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 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 およびバージョン 6 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 以降のプログラムの 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、このトピックで説明されている手段のいずれかを使用して 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 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 for z/OS* 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 K 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、351 ページの表 55 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、 351 ページの表 55 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。

注 : XML System Services パーサーは、EBCDIC 文書を解析するとき、以下の文字または文字の組み合わせを x'15' に変換します。

x'0D' CR
x'15' NL
x'25' LF
x'0D15' (これらの 2 バイトを一緒に)
x'0D25' (これらの 2 バイトを一緒に)

ASCII 文書のメモリー内イメージが EBCDIC に変換される場合、これらの文字の一部が生成される可能性があります。 COMPAT パーサーは、これらの変換をいずれも実行しません。 これらが実行されないことに依存するアプリケーションは、XMLPARSE(XMLSS) を使用するときに、該当する変更を必要とします。

- **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 for z/OS* プログラミング・ガイド」にあります。
  – **COMPAT:** XML-CODE 値は、「*Enterprise COBOL for z/OS* 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 *55.* 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;	>
&amp;	&
&apos;	'
&quot;	"

付録 L QSAM バッファーの初期化の制御

QSAM ファイルの先頭文字は、プリンターのスペーシング制御の制御文字 (CC) として使用できます。 場合によっては、LINAGE 節または WRITE AFTER...LINE(S) ステートメントのセマンティック要件に応じて、 制御文字だけを持つブランク・レコードが出力されます。

CC の後のバイトは遊びバイトと呼ばれます。 Enterprise COBOL V5.1 以前のバージョンでは、これらの遊びバイトの値は未定義です。 EBCDIC SPACE または BINARY ZERO のいずれかとなる可能性がありますが、COBOL 処理ルーチン中に明示的な初期化は行われません。 CC を処理するときにはプリンターが別の行に移動していた可能性があるため、これらの遊びバイトには技術的な意味はありません。 しかし、一部のプリンターは、特定の値で誤動作する場合があります。

COBOL 処理中に QSAM バッファーを何で初期化するかを制御し、遊びバイトの値に一貫性があるようにするには、 APAR PH25917 に対応する COBOL ランタイム LE PTF を適用し、以下のステップを実行します。

1. Language Environment サンプル・データ・セット .SCEESAMP で、 サンプル JCL IGZ4OPT をコピーしてカスタマイズします。 必要に応じて QSAMBUFFINITCHAR オプションを指定します。 ジョブ・カードおよびロード・ライブラリー名を変更し、この JCL を実行して IGZUOPT を作成します。
2. このモジュールは、アプリケーションの実行時に STEPLIB 連結内のデータ・セットに入れます。

サンプル JCL IGZ4OPT を使用して IGZUOPT を作成するとき、 JCL の JOB STEP1 は、IGZXOPT という MACRO を呼び出すアセンブラー・プログラムをアセンブルします。 このマクロは、特殊な COBOL ランタイム・オプションを指定するために使用されます。 以下の構文によって、QSAMBUFFINITCHAR オプションを指定します。

```
IGZXOPT QSAMBUFFINITCHAR=DEFAULT | SPACE | BINZERO
```

QSAMBUFFINITCHAR に設定できるのは DEFAULT、SPACE、または BINZERO で、デフォルト値は DEFAULT です。 QSAM バッファーは、指定した値に基づいて初期化されます。

- DEFAULT を指定すると、LINAGE 節または WRITE AFTER...LINE(S) ステートメントに使用される QSAM バッファーは、 Enterprise COBOL V5.1 以前のバージョンと同じ動作を使用します。
- SPACE を指定すると、LINAGE 節または WRITE AFTER...LINE(S) ステートメントに使用される QSAM バッファーは、 EBCDIC SPACE (X'40') で初期化されます。
- BINZERO を指定すると、LINAGE 節または WRITE AFTER...LINE(S) ステートメントに使用される QSAM バッファーは、 BINARY ZERO (X'00') で初期化されます。

付録 M Enterprise COBOL for z/OS のアクセシビリティー機能

アクセシビリティー機能は、運動や視覚などに障害を持つユーザーが情報技術製品を快適に使用できるように支援します。 z/OS のアクセシビリティー機能は、Enterprise COBOL for z/OS のアクセスを支援します。

アクセシビリティー機能

z/OS は、以下のような主要アクセシビリティー機能を備えています。

- スクリーン・リーダーおよび画面拡大機能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・サイズなど表示属性のカスタマイズ機能

z/OS では、US Section 508 (https://www.access-board.gov/ict/) および Web Content Accessibility Guidelines (WCAG) 2.0 (http://www.w3.org/TR/WCAG20/) に確実に準拠するために、最新の W3C 標準である WAI-ARIA 1.0 (http://www.w3.org/TR/wai-aria/) を使用しています。アクセシビリティー機能を利用するには、最新リリースのスクリーン・リーダーを、この製品でサポートされる最新の Web ブラウザーと併用してください。

IBM Knowledge Center の Enterprise COBOL for z/OS オンライン製品資料は、アクセシビリティーに対応しています。 IBM Knowledge Center のアクセシビリティー機能については、http://www.ibm.com/support/knowledgecenter/en/about/releasenotes.html に説明があります。

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。

ユーザーは、 IBM Developer for z/OS 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* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ikj4p120)
- *z/OS TSO/E User's Guide* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ikj4c240/APPENDIX1.3)
- *z/OS ISPF User's Guide Volume I* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ispzug70)
- IBM Developer for z/OS Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSQ2R2/rdz_welcome.html?lang=en)

上記の資料には、キーボード・ショートカットやファンクション・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。

インターフェース情報

Enterprise COBOL for z/OS のオンライン製品資料は、IBM Knowledge Center で入手でき、標準の Web ブラウザーで表示できます。

PDF ファイルでのアクセシビリティー・サポートは限定的です。 PDF 資料では、オプションのフォント拡大機能およびハイコントラスト表示設定を使用でき、キーボードのみでナビゲートできます。

スクリーン・リーダーで、ピリオドやコンマなどの PICTURE 記号を含む構文図、ソース・コード例、およびテキストを正確に読み上げるには、すべての句読点を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。

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。

関連アクセシビリティー情報

標準の IBM ヘルプ・デスクとサポート Web サイトに加え、IBM は、聴覚が不自由なお客様が営業やサポート・サービスにアクセスするために使用できる TTY 電話サービスを立ち上げました。

TTY サービス
800-IBM-3383 (800-426-3383)
(北米内)

IBM およびアクセシビリティー

IBM のアクセシビリティーへの取り組みについて詳しくは、IBM Accessibility (www.ibm.com/able) を参照してください。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。 本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

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

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

Intellectual Property Dept. for Rational Software
IBM Corporation
5 Technology Park Drive
Westford, MA 01886
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。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。 これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、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. 1991, 2020.

プライバシー・ポリシーに関する考慮事項:

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。 ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項をご確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンド・ユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および「IBM Software Products and Software-as-a-Service Privacy Statement」(http://www.ibm.com/software/info/product-privacy) を参照してください。

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。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

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、他の言語では同じ意味を持つことも、持たないこともあります。

この用語集には、以下の資料からの用語および定義が記載されています。

- 「*ANSI INCITS 23-1985, Programming languages - COBOL*」(「*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*」で改訂)
- 「*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。
- 「*ANSI X3.172-2002, American National Standard Dictionary for Information Systems*」
- *INCITS/ISO/IEC 1989-2002, Information technology - Programming languages - COBOL*
- *INCITS/ISO/IEC 1989:2014, Information technology - Programming languages, their environments and system software interfaces - Programming language COBOL*

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の中で、 共通のサブジェクトまたは共通のサブジェクトと共通の比較演算子を明示的に省略したことにより生ずる複合条件。

異常終了 (abend)
プログラムの異常終了。

16 MB 境界より上 (above the 16 MB line)
いわゆる 16 MB 境界より上で、2 GB バーより下のストレージ。 このストレージは、31 ビット・モードでのみアドレス可能である。 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・モードでコンパイルされたプログラムは、架空のストレージ境界線の下に保持されているかのように、16MB のスペースに対してのみアドレッシングが可能です。 VS COBOL II 以降、31 ビット・モードでコンパイルされたプログラムは、16 MB 境界より上に配置することができます。

アクセス・モード (* access mode)
ファイル内のレコードを操作するに当たって使用する方式。

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。

実際の文書エンコード (actual document encoding)
XML 文書のエンコード・カテゴリーで、以下のいずれかとなる。XML パーサーは文書の最初の数バイトを調べて判別する。

- ASCII
- EBCDIC
- UTF-8
- UTF-16 (ビッグ・エンディアンまたはリトル・エンディアンのいずれか)
- これ以外のサポートされないエンコード
- 認識不能なエンコード

英字名 (* alphabet-name)
ENVIRONMENT DIVISION の SPECIAL-NAMES 段落のユーザー定義語であり、特定の文字セットまたは照合シーケンス (あるいはその両方) に名前を割り当てるもの。

英字 (* alphabetic character)
英字またはスペース文字。

英数字位置 (alphanumeric character position)
「文字位置 *(character position)*」を参照。

英字データ項目 (alphabetic data item)
記号 A のみを含む PICTURE 文字ストリングが記述されたデータ項目。英字データ項目は USAGE DISPLAY を持ちます。

英数字 (* alphanumeric character)
コンピューターの 1 バイト文字セットの任意の文字。

英数字データ項目 (alphanumeric data item)
暗黙的または明示的に USAGE DISPLAY として記述された、カテゴリー英数字、英数字編集、または数字編集を持つデータ項目を 指す一般的な呼び方。

英数字編集データ項目 (alphanumeric-edited data item)
少なくとも 1 つの記号 A または X のインスタンスおよび少なくとも 1 つの単純挿入記号 B、0、または / を含んでいる、PICTURE 文字ストリングで記述されたデータ項目。 英数字編集データ項目は USAGE DISPLAY を持ちます。

英数字関数 (* alphanumeric function)
コンピューターの英数字セットからの 1 つ以上の文字のストリングで値が構成されている関数。

英数字グループ項目 (alphanumeric group item)
GROUP-USAGE NATIONAL 節なしで定義されたグループ項目。 INSPECT、STRING、および UNSTRING などの操作の場合、英数字グループ項目は、実際のグループの内容にかかわらず、その内容すべてが USAGE DISPLAY として記述されているかのように処理されます。 グループ内の基本項目を処理する必要のある操作 (MOVE CORRESPONDING、ADD CORRESPONDING、または INITIALIZE など) の場合、英数字グループ項目はグループ・セマンティクスを使用して処理されます。

英数字リテラル (alphanumeric literal)
次のセットからの開始区切り文字を有するリテラル。'、"、X'、X"、Z'、または Z"。 この文字ストリングには、コンピューターの有する文字セットの任意の文字を含めることができる。

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、 その内容が索引付きファイルの中のレコードを識別するもの。

米国規格協会 (American National Standards Institute: ANSI)
米国で認定された組織が自発的工業規格を作成して維持する手順を設定する組織であり、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) ID、リテラル、算術式、または関数 ID で、これにより関数の評価に使用する値を指定する。 (2) CALL または INVOKE ステートメント の USING 句のオペランドであり、呼び出されたプログラムまたは起動されたメソッドに値を渡すのに使用されます。

算術式 (* arithmetic expression)
数値リテラル、数値基本項目 (算術演算子で区切られた ID とリテラルなど) を表す ID、1 つの算術演算子で区切られた 2 つの算術式、または括弧で囲まれた算術式。

算術演算 (* arithmetic operation)
ある算術ステートメントが実行されることにより、 またはある算術式が計算されることにより生じるプロセスで、 そこで指定された引数に対して数学的に正しい解が求められる。

算術演算子 (* arithmetic operator)
次に示す集合に属する 1 文字、または 2 文字で構成された固定した組み合わせ。

文字	意味
+	加算
-	減算
*	乗算
/	除算

文字	意味
**	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 の各ステートメントがある。

配列 (array)
データ・オブジェクトで構成される集合体。それぞれの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。 配列は、COBOL ではテーブルに類似する。

昇順キー (* ascending key)
データ項目を比較する際の規則に一致するように、最低のキー値から始めて最高のキー値へとデータを順序付けている値に即したキー。

ASCII
情報交換用米国標準コード。 7 ビットのコード化文字をベースとする 1 つのコード化文字セット (パリティー・チェックを含む 8 ビット) を 使用する標準コードであり、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に使用される。 ASCII セットは、制御文字と図形文字から構成されている。

IBM は、ASCII に対する拡張 (文字 128 から 255) を定義している。

ASCII DBCS
「*2 バイト ASCII (double-byte ASCII)*」を参照。

割り当て名 (assignment-name)
COBOL ファイルの編成を識別する名前で、 システムがこれを認識する際に使用する。

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。 想定小数点には、論理的な意味があり、物理的には表現されない。

AT END 条件 (AT END condition)
次のような特定の条件のもとで、READ、RETURN、または SEARCH ステートメントを実行した場合に引き起こされる条件。

- 順次アクセス・ファイルに対して READ ステートメントを実行中に、そのファイル内に次の論理レコードが存在しない場合、または相対レコード番号中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場合、またはオプションの入力ファイルが使用可能でない場合。
- RETURN ステートメントの実行中に、関連するソート・ファイルまたはマージ・ファイルについての次の論理レコードが存在しない場合。
- SEARCH ステートメントの実行中に、関連する WHEN 句のいずれかで指定された条件を満足することなく、検索操作が終了した場合。

B

基本文字セット (basic character set)
言語のワード、文字ストリング、および区切り文字の作成時に使用される基本的な文字セット。 基本文字セットは 1 バイトの EBCDIC でインプリメントされる。 拡張文字セットには DBCS 文字が含まれる。これは、コメント、リテラル、およびユーザー定義語で使用できる。

85 COBOL 標準 における「*COBOL* 文字セット *(COBOL character set)*」と同義。

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)* と比較。

バイナリー項目 (binary item)
2 進表記 (基数 2 の数体系) で表される数値データ項目。 等価の 10 進数は、10 進数字 0 から 9 に演算符号を加えたもので構成される。 項目の左端のビットは、演算符号。

二分探索 (binary search)
二分探索の各段階では、 データ・エレメント集合が 2 つに分割される。 数が奇数の場合は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、ある論理レコードの一部がブロックに入ることがある。 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。「物理レコード *(physical record)*」と同義。

ブール条件 (boolean condition)
ブール条件は、ブール・リテラルが true であるか false であるかを決定する。 ブール条件は定数条件式でのみ使用できる。

ブール・リテラル (boolean literal)
true 値を示す B'1'、または false 値を示す B'0' のどちらか。 ブール・リテラルは定数条件式でのみ使用できる。

停止点 (breakpoint)
通常は命令によって指定されるコンピューター・プログラムの場所であり、プログラムの実行は外部からの介入またはモニター・プログラムによって割り込まれる場合がある。

バッファー (buffer)
入力データまたは出力データを一時的に保持するために使用されるストレージの一部分。

組み込み関数 (built-in function)
組み込み関数 *(intrinsic function)* を参照。

ビジネス・メソッド (business method)
ビジネス・ロジックまたはアプリケーションのルールをインプリメントする Enterprise Bean のメソッド。 (Oracle)

バイト (byte)
特定の数のビット (通常 8 ビット) から成るストリングであり、1 つの単位として処理され、1 つの文字または制御機能を表す。

バイト・オーダー・マーク (BOM) (byte order mark (BOM))
UTF-16 または UTF-32 テキストの先頭に使用して、後続テキストのバイト・オーダーを示す Unicode 文字。バイト・オーダーには、「ビッグ・エンディアン (big-endian)」または「リトル・エンディアン (little-endian)」がある。

バイトコード (bytecode)
Java コンパイラーによって生成され、Java インタープリターによって実行される、マシンから独立したコード。(Oracle)

C

呼び出し可能サービス (callable services)
言語環境プログラムでは、従来の言語環境プログラム定義の呼び出しインターフェースを使用して COBOL プログラムが呼び出すことができる一連のサービス。 言語環境プログラムの規約を共有するすべてのプログラムがこれらのサービスを使用できる。

呼び出し先プログラム (called program)
CALL ステートメントの対象となるプログラム。 呼び出し先プログラムと呼び出し側プログラムが実行時に結合されて、1 つの実行単位が作成される。

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。

標準分解 (canonical decomposition)
複数の Unicode 文字を使用して単一の合成済み Unicode 文字を表す方法。 通常、標準分解は、ラテン語文字と発音区別符号が個別に表されるように発音区別符号付きのラテン語文字を分離するために使用される。 合成済み Unicode 文字とその標準分解を表す例については、合成済み文字 *(precomposed character)* を参照。

ケース構造 (case structure)
結果として生じた多数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。

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・セットに置かれた一連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 をコーディングする時間を節約して、エラーを減らすことができる。

CCSID
コード化文字セット *ID (coded character set identifier)* を参照。

世紀ウィンドウ (century window)
2 桁年号が固有に決まる 100 年間のこと。 COBOL プログラマーが使用できる世紀ウィンドウには、いくつかのタイプがある。

- ウィンドウ操作組み込み関数 DATE-TO-YYYYMMDD、DAY-TO-YYYYDDD、および YEAR-TO-YYYY については、引数*-2 (argument-2)* によって世紀ウィンドウを指定する。
- 言語環境プログラム呼び出し可能サービスについては、CEESCEN で世紀ウィンドウを指定する。

文字 (* character)
言語のそれ以上分割できない基本単位。

文字エンコード・ユニット (character encoding unit)
コード化文字セット内の 1 つのコード・ポイントに相当するデータの単位。 1 つ以上の文字エンコード・ユニットを使用して、コード化文字セットの文字が表現される。 エンコード・ユニット とも呼ばれる。

USAGE NATIONAL の場合、文字エンコード・ユニットは、 UTF-16 の 2 バイト・コード・ポイントに対応している。

USAGE DISPLAY の場合、文字エンコード・ユニットは、1 つのバイトに対応している。

USAGE DISPLAY-1 の場合、文字エンコード・ユニットは、 DBCS 文字セットの 2 バイト・コード・ポイントに対応している。

文字位置 (character position)
1 文字を保持または表示するために必要な物理ストレージまたは表示スペースの量。 この用語はどのような文字のクラスにも適用される。 文字の特定のクラスについては、以下の用語が適用される。

- 英数字文字位置。USAGE DISPLAY を使用して表される DBCS 文字。
- *DBCS* 文字位置。USAGE DISPLAY-1 を使用して表される DBCS 文字。
- 国別文字位置。USAGE NATIONAL を使用して表される文字。UTF-16 の 文字エンコード・ユニットと同義。

文字セット (character set)
テキスト情報を表すために使用されるエレメントの集合。 ただし、コード化表現は想定されていません。 コード化文字セット *(coded character set)* も参照。

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連続した文字。 文字ストリングは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・ステップを後で再始動することができるように、 ジョブとシステムの状況に関する情報を記録しておくことができるところ。

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。 同じ具体化を共用するオブジェクトは、同じクラスのオブジェクトとみなされる。 クラスは階層として定義でき、あるクラスを別のクラスから継承することができる。

クラス (オブジェクト指向) (class (object-oriented))
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。 同じ具体化を共用するオブジェクトは、同じクラスのオブジェクトとみなされる。

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、 すべて DBCS であるか、すべて漢字であるか、あるいはクラス名の定義においてリストされた文字だけで構成されるかという命題で、それに関して真の値を判別することができる。

クラス定義 (* class definition)
クラスを定義する COBOL ソース単位。

クラス階層 (class hierarchy)
オブジェクト・クラス間の関係を示すツリーのような構造。 最上部に 1 つのクラスが置かれ、その下に 1 つ以上のクラスの層が置かれる。 「継承階層 *(inheritance hierarchy)*」と同義。

クラス識別記入項目 (* class identification entry)
IDENTIFICATION DIVISION の CLASS-ID 段落内の 記入項目であり、クラス名を指定する節と、選択した属性をクラス定義に割り当てる節を含む。

クラス名 (オブジェクト指向) (class-name (object-oriented))
オブジェクト指向 COBOL クラス定義の名前。

クラス名 (データの) (* class-name (of data))
ENVIRONMENT DIVISION の SPECIAL-NAMES 段落で定義されるユーザー定義語であり、真理値を定義できる命題に名前を割り当てる。データ項目の内容は、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る。

クラス・オブジェクト (class object)
クラスを表す実行時オブジェクト。

節 (* clause)
記入項目の属性を指定するという目的で順番に並べられた連続する COBOL 文字ストリング。

クライアント (client)
オブジェクト指向プログラミングにおいて、クラス内の 1 つ以上のメソッドからサービスを要求するプログラムまたはメソッド。

COBOL 文字セット (COBOL character set)
COBOL 構文を作成する際に使用される文字セット。 完全な COBOL 文字セットは、以下の文字で構成される。

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 (-) (ハイフン)
*	アスタリスク
/	斜線 (スラッシュ)
=	等号
$	通貨符号
,	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

文字	意味
)	右括弧
>	より大きい
<	より小さい
:	コロン
_	下線

COBOL ワード (* COBOL word)
ワード *(word)* を参照。

コード・ページ (code page)
すべてのコード・ポイントに図形文字および制御機能の意味を割り当てるもの。 例えば、あるコード・ページでは、8 ビット・コードに対して 256 コード・ポイントに文字と意味を割り当て、別のコード・ページでは、7 ビット・コードに対して 128 コード・ポイントに文字と意味を割り当てることができる。 ワークステーション上の英語の IBM コード・ページは IBM-1252 で、ホストは IBM-1047 である。コード化文字セット *(coded character set)*。

コード・ポイント (code point)
コード化文字セット (コード・ページ) に定義する固有のビット・パターン。 コード・ポイントには、グラフィック・シンボルおよび制御文字が割り当てられる。

コード化文字セット (coded character set)
文字セットを設定し、その文字セットの文字とコード化表現との間の関係を設定する明確な規則の集まり。 コード化文字セットの例として、ASCII もしくは EBCDIC コード・ページで、または Unicode 対応の UTF-16 エンコード・スキームで表す文字セットがある。

コード化文字セット ID (CCSID) (coded character set identifier (CCSID))
特定のコード・ページを識別する 1 から 65,535 までの IBM 定義番号。

照合シーケンス (* collating sequence)
コンピューターに受け入れ可能な文字のシーケンスで、 ソート、マージ、比較、および索引付きファイルの順次処理を目的として順序付けしたもの。

列 (* column)
印刷行または参照形式行におけるバイト位置。 列は、行の左端の位置から始めて行の右端の位置まで、1 から 1 ずつ増やして番号が付けられる。 列は 1 つの 1 バイト文字を保持する。

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。 条件 *(condition)* および 複合否定条件 *(negated combined condition)* も参照。

結合文字 (combining characters)
他の前後の Unicode 文字を変更するために使用される Unicode 文字。 通常、結合文字は、ラテン語文字を変更するために使用される Unicode 発音区別符号。 合成済み文字 *(precomposed character)* を参照して、ラテン語文字 U+0061 (a) とともに使用される結合文字 U+0308 (¨) の例を確認すること。

コメント記入項目 (* comment-entry)
ドキュメンテーションに使用される IDENTIFICATION DIVISION 内の項目で、実行に影響しない。

コメント行 (comment line)
行の標識区域のアスタリスク (*) と、または、プログラム・テキスト区域 (区域 A および区域 B) の最初の文字ストリングとしてのアスタリスクと大なり記号 (*>) と、 その行の区域 A および区域 B に続くコンピューターの文字セットの任意の文字によって表されるソース・プログラム行。 コメント行は、文書化にのみ役立つ。 行の標識区域では斜線 (/)、そしてその行の区域 A および B ではコンピューター文字セット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改ページが行われる。

共通プログラム (* common program)
別のプログラムに直接的に含まれているにもかかわらず、その別のプログラムに直接的または間接的に含まれている任意のプログラムから呼び出す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) ある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、プログラムの全体的なロジック構造を利用することによって、または 1 つの記号ステートメントから複数のコンピューター命令を作り出すことによって、またはアセンブラーの機能のようにこれら両方を使用することによって、マシン言語プログラムを生成すること。

コンパイル変数 (compilation variable)
ある特定のリテラル値のシンボル名、あるいは DEFINE ディレクティブまたは DEFINE コンパイラー・オプションによって指定されたコンパイル時演算式のシンボル名。

コンパイル時 (* compile time)
COBOL コンパイラーによって、COBOL ソース・コードが COBOL オブジェクト・プログラムに変換される時間。

コンパイル時演算式 (compile-time arithmetic expression)
DEFINE ディレクティブおよび EVALUATE ディレクティブに、または定数条件式に指定されている算術式のサブセット。 コンパイル時演算式における、正規演算式との違い:

- 指数演算子を指定することはできません。
- オペランドはすべて、整数リテラルか、すべてのオペランドが整数リテラルである演算式でなければなりません。
- 式はゼロによる除算が発生しないように指定する必要があります。

コンパイラー (compiler)
高水準言語で記述されたソース・コードをマシン言語のオブジェクト・コード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)
コンパイル時にコンパイラーに特定の処置を行わせるステートメント。 標準コンパイラー指示ステートメントには、 COPY、REPLACE、および USE がある。

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。 条件 *(condition)*、単純否定条件 *(negated simple condition)*、および 複合否定条件 *(negated combined condition)* も参照。

複合 ODO (complex ODO)
次のような OCCURS DEPENDING ON 節の特定の形式。

- 可変位置項目またはグループ: DEPENDING ON オプションを指定した OCCURS 節によって記述されたデータ項目の後に、非従属データ項目またはグループが続く。 グループは英数字グループでも国別グループでも構いません。
- 可変位置テーブル: DEPENDING ON オプションを指定した OCCURS 節によって記述されたデータ項目の後に、OCCURS 節に よって記述された非従属データ項目が続く。
- 可変長エレメントを持つテーブル: OCCURS 節によって記述されたデータ項目に、DEPENDING ON オプションを指定した OCCURS 節によって記述された従属データ項目が含まれている。
- 可変長エレメントを持つテーブルの指標名。
- 可変長エレメントを持つテーブルのエレメント。

コンポーネント (component)
(1) 関連ファイルからなる機能グループ化。 (2) オブジェクト指向プログラミングでは、特定の機能を実行し、他のコンポーネントやアプリケーションと連携するように設計されている、再使用可能なオブジェクトまたはプログラム。 JavaBeans は、コンポーネントを作成するための、Oracle が提供するアーキテクチャーである。

合成形式 (composed form)
標準分解による合成済み Unicode 文字の表記。 詳しくは、合成済み文字 *(precomposed character)* を参照。

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 (例外) (condition (exception))
言語環境プログラムによって使用可能にされる、あるいは認識される例外。したがって、ユーザー条件処理ルーチンと言語条件処理ルーチンの活動化に適している。 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、条件は言語特定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できる。

条件 (式) (condition (expression))
ある真の値が決定される実行時のデータの状況。 条件という用語が本書で 一般形式の「条件」(条件-*1*、条件-*2*、...) の中、またはこれに関連して使用された場合は、. .) 次のいずれかである。オプションとして括弧で囲まれた単純条件からなる条件式、あるいは、単純条件、論理演算子、および括弧の構文的に正しい組み合わせ (真理値を判別できる) からなる複合条件。 単純条件 *(simple condition)*、複合条件 *(complex condition)*、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。

条件式 (* conditional expression)
EVALUATE、IF、PERFORM、または SEARCH ステートメントの中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。

条件句 (* conditional phrase)
ある条件ステートメントが実行された結果得られる条件の真理値の判別に基づいてとられるべき処置を指定する句。

条件ステートメント (* conditional statement)
条件の真理値を判別することと、オブジェクト・プログラムの次の処理がこの真理値によって決まることを指定するステートメント。

条件変数 (* conditional variable)
1 つまたは複数の値を持つデータ項目であり、これらの値が、そのデータ項目に割り当てられた条件名を持つ。

条件名 (* condition-name)
条件変数が想定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るユーザー定義語。

条件名条件 (* condition-name condition)
真理値を判別できる命題で、かつ、条件変数の値が、その条件変数と関連する条件名に属する一連の値のメンバーである命題。

*** CONFIGURATION SECTION**
ENVIRONMENT DIVISION のセクションであり、ソース・プログラムとオブジェクト・プログラムの全体的な仕様およびクラス定義を記述する。

CONSOLE
オペレーター・コンソールに関連する COBOL 環境名。

定数条件式 (constant conditional expression)
IF ディレクティブで、または EVALUATE ディレクティブの WHEN 句で使用される可能性がある条件式のサブセット。

定数条件式は、以下の項目のいずれかでなければなりません。

- 両方のオペランドがリテラルであるか、リテラル項のみを含む算術式である比較条件。条件は比較条件の規則に従う必要があり、以下の追加事項があります。
  - オペランドは同じカテゴリーでなければなりません。算術式は数値カテゴリーです。
  - リテラルが指定され、それらが数値リテラルではない場合、関係演算子は “IS EQUAL TO”、“IS NOT EQUAL TO”、“IS =”、“IS NOT =”、または “IS <>” でなければなりません。

  比較条件 *(relation condition)* も参照。
- 定義済み条件。定義済み条件 *(defined condition)* も参照。
- ブール条件。 ブール条件 *(boolean condition)* も参照。

- 前述の単純条件の形式を、AND、OR、および NOT を使用して複合条件に結合することによって形成した複合条件。 簡略複合比較条件を指定することはできません。複合条件 *(complex condition)* も参照。

含まれているプログラム (contained program)
別の COBOL プログラムにネストされている COBOL プログラム。

連続項目 (* contiguous items)
DATA DIVISION 内の連続する記入項目によって記述され、相互に一定の階層関係を持っている項目。

コピーブック (copybook)
一連のコードが含まれたファイルまたはライブラリー・メンバーであり、コンパイル時に COPY ステートメントを使用してソース・プログラムに組み込まれる。 ファイルはユーザーが作成する場合、COBOL によって提供される場合、 または他の製品によって供給される場合とがある。 「コピー・ファイル *(copy file)*」と同義。

カウンター (* counter)
他の数字を使ってその数字分だけ増減したり、 あるいは 0 または任意の正もしくは負の値に変更またはリセットしたりできるようにした、 数または数表現を収めるために使用されるデータ項目。

相互参照リスト (cross-reference listing)
コンパイラー・リストの一部であり、プログラム内においてファイル、フィールド、および標識が定義、参照、および変更される場所に関する情報が入る。

通貨記号値 (currency-sign value)
数字編集項目に保管される通貨単位を識別する文字ストリング。 典型的な例としては、$、USD、EUR などがある。 通貨記号値は、CURRENCY コンパイラー・オプションで定義するか、ENVIRONMENT DIVISION の SPECIAL-NAMES 段落内の CURRENCY SIGN 節によって定義することができる。 CURRENCY SIGN 節が指定されない場合、NOCURRENCY コンパイラー・オプションが有効であれば、ドル記号 ($) がデフォルトの通貨記号値として使用される。 通貨記号 *(currency symbol)* も参照。

通貨記号 (currency symbol)
数字編集項目内の通貨記号値の部分を示すために、PICTURE 節で使用される文字。 通貨記号は、CURRENCY コンパイラー・オプションで定義するか、ENVIRONMENT DIVISION の SPECIAL-NAMES 段落内の CURRENCY SIGN 節によって定義することができる。 CURRENCY SIGN 節が指定されない場合、NOCURRENCY コンパイラー・オプションが有効であれば、ドル記号 ($) がデフォルトの通貨記号値および通貨記号として使用される。 通貨記号と通貨符号値は複数定義可能。 通貨記号値 *(currency sign value)* も参照。

現行レコード (* current record)
ファイル処理において、 ファイルに関連するレコード域の中で使用可能なレコード。

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している概念上のエンティティー。

D

データ節 (* data clause)
COBOL プログラムの DATA DIVISION のデータ記述記入項目に現れる節で、データ項目の特定の属性を記述する情報を提供する。

データ記述項目 (* data description entry)
COBOL プログラムの DATA DIVISION 内の記入項目であり、レベル番号の後に必要に応じてデータ名が続き、その後に必要に応じて一連のデータ節で構成されるもの。

DATA DIVISION
COBOL プログラムまたはメソッドの 1 つの部 (division)。 使用するファイルとファイルに含まれるレコード、必要となる内部 WORKING-STORAGE レコード、COBOL 実行単位内の複数のプログラムで使用可能なデータなど、プログラムまたはメソッドで処理するデータを記述する。

データ項目 (* data item)
COBOL プログラムにより、または関数評価の規則により、定義されたデータの単位 (リテラルを除く)。

データ・セット (data set)
「ファイル *(file)*」の同義語。

データ名 (* data-name)
データ記述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た場合、データ名は、その形式の規則で特に許可されていない限り、参照変更、添え字付け、または修飾してはならないワードを表す。

DBCS
2 バイト文字セット *(double-byte character set (DBCS))* を参照

DBCS 文字 (DBCS character)
IBM 2 バイト文字セット (DBCS) に定義された任意の文字。

DBCS 文字位置 (DBCS character position)
文字位置 *(character position)* を参照。

DBCS データ項目 (DBCS data item)
少なくとも 1 つの記号 G または少なくとも 1 つの記号 N (NSYMBOL(DBCS) コンパイラー・オプションが有効なとき) を含んでいる PICTURE 文字ストリングで記述されたデータ項目。DBCS データ項目は USAGE DISPLAY-1 を持っています。

デバッグ行 (* debugging line)
行の標識区域に文字 D がある行のこと。

デバッグ・セクション (* debugging section)
USE FOR DEBUGGING ステートメントが含まれているセクション。

宣言文 (* declarative sentence)
区切り記号のピリオドによって終了する 1 つの USE ステートメントから構成されるコンパイラー指示文。

宣言部分 (* declaratives)
PROCEDURE DIVISION の先頭に書き込まれた 1 つ以上の特殊目的セクションの集合であり、その先頭にはキーワード DECLARATIVE が付き、その最後にはキーワード END DECLARATIVES が続いている。宣言部分は、セクション・ヘッダー、USE コンパイラー指示文、および 0 個、1 個、または複数個の関連する段落で構成される。

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。

定義済み条件 (defined condition)
コンパイル変数が定義されているかどうかをテストするコンパイル時条件。 定義済み条件は、IF ディレクティブで、または EVALUATE ディレクティブの WHEN 句で指定される。

範囲区切りステートメント (* delimited scope statement)
明示的範囲終了符号を含んでいるステートメント。

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、これを使用して区切られる文字ストリングの一部ではない。

従属領域 (dependent region)
IMS において、メッセージ・ドリブン・プログラム、バッチ・プログラム、またはオンライン・ユーティリティーを含む MVS 仮想記憶領域。

降順キー (* descending key)
データ項目を比較する際の規則に一致するように、 最高のキー値から始めて最低のキー値へとデータを順序付けている値に付けられるキー。

数字 (digit)
0 から 9 までの任意の数字。COBOL では、この用語を用いて他の記号を参照することはない。

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)
前回アクセスしたデータへの参照には依存せず、 プロセスがデータの位置にのみ依存する方法で、 データを記憶装置から取得したり、データを記憶装置に入れる機能。

表示浮動小数点データ項目 (display floating-point data item)
暗黙的または明示的に USAGE DISPLAY として記述されており、外部浮動小数点データ項目を記述する PICTURE 文字ストリングを持っている、データ項目。

部 (* division)
部の本体と呼ばれる、0 個、1 個、または複数個のセクションまたは段落の集合であり、特定の規則に従って形成および結合されたもの。 それぞれの部は、部のヘッダーおよび関連した部の本体で構成される。 COBOL プログラムには、見出し部、環境部、データ部、および手続き部の 4 つの部がある。

部の見出し (* 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 *(DLL) (dynamic link library (DLL))* を参照。

DLL アプリケーション (DLL application)
インポートしたプログラム、関数、または変数を参照するアプリケーション。

DLL リンケージ (DLL linkage)
DLL および NODYNAM オプションを 使用してコンパイルされたプログラム内の CALL。CALL は、別個のモジュール内のエクスポートされた名前に解決されるか、または、別個のモジュールに定義されたメソッドの INVOKE に解決される。

Do 構造 (do construct)
構造化プログラミングでは、DO ステートメントを使えば、プロシージャー内の複数のステートメントをグループ化できる。 COBOL では、インライン PERFORM ステートメントが同様に機能する。

do-until
構造化プログラミングにおいて、do-until ループは、少なくとも 1 回は実行され、所定の条件が真になるまで実行される。 COBOL では、TEST AFTER 句を PERFORM ステートメントで使用すれば、同様に機能する。

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、TEST BEFORE 句を PERFORM ステートメントで使用すれば、同様に機能する。

文書タイプ宣言 (document type declaration)
あるクラスの文書に対する文法を規定するマークアップ宣言を含む、または 指示する XML エレメント。この文法は、文書タイプ定義または DTD とも呼ばれる。

文書タイプ定義 (document type definition (DTD))
XML 文書のクラスの文法。 文書タイプ宣言を参照。

2 バイト ASCII (double-byte ASCII)
DBCS 文字および 1 バイト ASCII 文字を含む IBM の文字セット (「ASCII DBCS」とも呼ばれる)。

2 バイト EBCDIC (double-byte EBCDIC)
DBCS 文字および 1 バイト EBCDIC 文字を含む IBM の文字セット (「EBCDIC DBCS」とも呼ばれる)。

2 バイト文字セット (double-byte character set (DBCS))
それぞれの文字が 2 バイトで表現される 1 組の文字。 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 文字の入力、表示、および印刷には、DBCS を受け入れ可能なハードウェアおよびサポートされるソフトウェアが必要。

DWARF
DWARF は UNIX International Programming Languages Special Interest Group (SIG) で開発された。言語に依存しないデバッグ情報を提供することにより、さまざまな言語の、統一された方法でのシンボリックなソース・レベル・デバッグのニーズを満たすように設計されている。 DWARF ファイルには、さまざまなエレメントに編成されたデバッグ・データが含まれている。 詳しくは、「*DWARF/ELF エクステンション ライブラリー・リファレンス*」の『*DWARF program information*』を参照。

動的アクセス (* dynamic access)
1 つの OPEN ステートメントの実行範囲内において、特定の論理レコードを、大容量記憶ファイルからは順次アクセス以外の方法で取り出したりそのファイルに入れたりでき、またファイルからは順次アクセスの方法で取り出せるアクセス・モード。

動的 CALL (dynamic CALL)
DYNAM オプションおよび NODLL オプションを使用してコンパイルされたプログラム内の CALL *literal* ステートメント、または NODLL オプションを使用してコンパイルされたプログラム内の CALL *identifier* ステートメント。

動的長 (dynamic-length)
実行時に変化する可能性がある論理長を持つ項目を記述する形容詞。

動的長基本項目 (dynamic-length elementary item)
データ宣言項目に DYNAMIC LENGTH 節が含まれる基本データ項目。

動的長グループ (dynamic-length group)
従属動的長基本項目を含むグループ項目。

ダイナミック・リンク・ライブラリー (DLL) (dynamic link library (DLL))
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バインドされる実行可能コードおよびデータが入ったファイル。 複数のアプリケーションが DLL 内のコードおよびデータを同時に共用することができる。 DLL はプログラムの実行可能ファイルの一部ではないが、実行可能ファイルを正しく実行するためには必要となる可能性がある。

動的ストレージ域 (dynamic storage area (DSA))
動的に獲得されるストレージであり、レジスター保管域、および動的ストレージ割り振 りに使用可能な区域 (プログラム変数など) から構成される。 DSA は、プログラムまたは関数が呼び出されるときに割り振られ、 呼び出しインスタンスの継続時間の間持続します。 DSA は通常、言語環境プログラムによって管理されるスタック・セグメント内に割り振られる。

E

EBCDIC (拡張 2 進化 10 進コード) (* EBCDIC (Extended Binary-Coded Decimal Interchange Code))
8 ビット・コード化文字をベースとするコード化文字セット。

EBCDIC 文字 (EBCDIC character)
EBCDIC (拡張 2 進化 10 進コード) セットに含まれているいずれかの記号。

EBCDIC DBCS
「*2 バイト EBCDIC (double-byte EBCDIC)*」を参照。

編集データ項目 (edited data item)
0 の抑止または編集文字の挿入、あるいはその両方を行うことによって変更されたデータ項目。

編集用文字 (* editing character)
次に示す集合に属する 1 文字、または 2 文字で構成される固定した組み合わせ。

文字	意味
	スペース
0	ゼロ
+	正符号
-	負符号
CR	貸方
DB	借方

文字	意味
Z	ゼロの抑止
*	小切手変造防止
$	通貨符号
,	コンマ (小数点)
.	ピリオド (小数点)
/	斜線 (スラッシュ)

EGCS
拡張図形文字セット *(extended graphic character set) (EGCS)* を参照。

EJB
Enterprise JavaBeans を参照。

EJB コンテナー (EJB container)
J2EE アーキテクチャーの EJB コンポーネント契約を実装するコンテナー。 この契約は、セキュリティー、並行性、ライフ・サイクル管理、トランザクション、デプロイメント、およびその他のサービスを含んでいるエンタープライズ Bean 用のランタイム環境を指定します。 EJB コンテナーは、EJB サーバーまたは J2EE サーバーによって提供される。 (Oracle)

EJB サーバー (EJB server)
EJB コンテナーにサービスを提供するソフトウェア。 EJB サーバーは、1 つ以上の EJB コンテナーをホストできる。 (Oracle)

エレメント (テキスト・エレメント) (element (text element))
1 つのデータ項目または動詞の記述などのようなテキスト・ストリングの 1 つの論理単位で、その前にエレメント・タイプを識別する固有のコードが付けられたもの。

基本項目 (* elementary item)
論理的にそれ以上細分できないものとして記述されているデータ項目。

カプセル化 (encapsulation)
オブジェクト指向プログラミングでは、オブジェクトの固有の詳細を隠すのに使用される技法。 オブジェクトは、基礎構造を露出しなくても、データの照会と操作を行うインターフェースを提供する。「情報隠蔽 *(information hiding)*」と同義。

エンクレーブ (enclave)
言語環境プログラムのもとで実行される場合、エンクレーブは実行単位に類似している。 エンクレーブは、LINK および C の system() 関数の使用によって、他のエンクレーブを作成できる。

エンコード・ユニット (encoding unit)
文字エンコード・ユニット *(character encoding unit)* を参照。

END CLASS マーカー (end class marker)
語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 クラス名.
```

メソッド終了マーカー (end method marker)
語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.
```

PROCEDURE DIVISION の終わり (* end of PROCEDURE DIVISION) (* end of PROCEDURE DIVISION)
COBOL ソース・プログラムにおいて、それ以後にはプロシージャーが存在しない物理的な位置。

プログラム終了マーカー (* end program marker)
語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、次のように記述する。

```
END PROGRAM program-name.
```

Enterprise Bean
ビジネス・タスクを実装し、EJB コンテナーに存在するコンポーネント。 (Oracle)

Enterprise JavaBeans
オブジェクト指向の分散エンタープライズ・レベル・アプリケーションの開発とデプロイメントのために、Oracle によって定義されたコンポーネント・アーキテクチャー。

項目 (* entry)
分離文字ピリオドで終了させられる連続する節の記述セットであり、COBOL プログラムの IDENTIFICATION DIVISION、ENVIRONMENT DIVISION、または DATA DIVISION に書き込まれる。

環境節 (* environment clause)
ENVIRONMENT DIVISION 記入項目の一部として現れる節。

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。 ENVIRONMENT DIVISION 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と、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する。 この部では、ファイルの論理概念とそのレコードの間のリンケージ、およびファイルが保管される装置の物理的局面を提供する。

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の制御文字、報告書コード、またはプログラム・スイッチ、あるいはそれらの組み合わせを識別する。環境名が ENVIRONMENT DIVISION の簡略名と関連付けられている場合は、その簡略名を、置換が有効な任意の形式で置き換えることができる。

環境変数 (environment variable)
コンピューター環境の一部の局面を定義する多数の変数のいずれかであり、その環境で動作するプログラムからアクセス可能。 環境変数は、動作環境に依存するプログラムの動作に影響を与える。

実行時 (execution time)
実行時 *(run time)* を参照。

実行時環境 (execution-time environment)
ランタイム環境 *(runtime environment)* を参照。

明示的範囲終了符号 (* explicit scope terminator)
特定の PROCEDURE DIVISION ステートメントの有効範囲を終わらせる予約語。

指数 (exponent)
別の数 (底) をべき乗する指数を示す数。 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)
算術式または条件式。

拡張モード (* extend mode)
ファイルに対する EXTEND 句の指定のある OPEN ステートメントが実行されてから、そのファイルに対する REEL または UNIT 句の指定のない CLOSE ステートメントが実行される前までの、ファイルの状態。

拡張図形文字セット (extended graphic character set) (EGCS)
それぞれの図形文字を表すために 2 バイトを必要とする図形文字セット (漢字文字セットなど)。 現在は、*2* バイト文字セット *(DBCS)* と呼ばれるようになった。

Extensible Markup Language
XML を参照。

拡張 (extensions)
85 COBOL 標準 に記述されているものの他に、IBM コンパイラーでサポートされる COBOL 構文およびセマンティクス。

外部コード・ページ (external code page)
XML 文書では、CODEPAGE コンパイラー・オプションによって 指定された値。

外部データ (* external data)
プログラムの中で外部データ項目および外部ファイル結合子として記述されるデータ。

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任意のプログラムから参照することができる。

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任意のプログラムから参照できる。

外部 10 進数データ項目 (external decimal data item)
ゾーン *10* 進数データ項目 *(zoned decimal data item)* および 国別 *10* 進数データ項目 *(national decimal data item)* を参照。

外部ファイル結合子 (* external file connector)
実行単位の 1 つ以上のオブジェクト・プログラムにアクセス可能なファイル結合子。

外部浮動小数点データ項目 (external floating-point data item)
表示浮動小数点データ項目 *(display floating-point data item)* および 国別浮動小数点データ項目 *(national floating-point data item)* を参照。

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、名前が付けられたハードウェア装置またはソフトウェアで、2 つの選択的な状態のうち一方が存在することを示すために使用される。

F

ファクトリー・データ (factory data)
いったんクラスに割り振られ、クラスのすべてのインスタンスに共用されるデータ。 ファクトリー・データは、クラス定義の FACTORY 段落の DATA DIVISION の WORKING-STORAGE SECTION 内に宣言される。Java private 静的データと同義。

ファクトリー・メソッド (factory method)
オブジェクト・インスタンスとは無関係に、クラスによってサポートされる メソッド。 ファクトリー・メソッドは、クラス定義の FACTORY 段落に宣言される。Java public 静的メソッドと同義。これらは通常、オブジェクトの作成をカスタマイズするために使用されます。

形象定数 (* figurative constant)
ある予約語を使用することによって参照されるコンパイラー生成の値。

ファイル (* file)
論理レコードの集合。

ファイル属性対立条件 (* file attribute conflict condition)
あるファイルで入出力操作を実行する試みが失敗し、 プログラムの中でそのファイルに対して指定したファイル属性が、 そのファイルの固定属性と一致していないこと。

ファイル節 (* file clause)
DATA DIVISION の記入項目であるファイル記述項目 (FD 記入項目) およびソート・マージ・ファイル記述項目 (SD 記入項目) のいずれかの一部として現れる節。

ファイル結合子 (* file connector)
ファイルに関する情報が入っており、ファイル名と物理ファイルの間のリンケージとして、さらに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ストレージ域。

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。

ファイル制御項目 (* file control entry)
SELECT 節と、ファイルの関連物理属性を宣言するすべての従属節。

FILE-CONTROL 段落 (FILE-CONTROL paragraph)
ENVIRONMENT DIVISION 内の段落であり、この中では、特定のソース単位で使用されるデータ・ファイルが宣言される。

ファイル記述項目 (* file description entry)
DATA DIVISION の FILE SECTION の中にある記入項目。レベル標識 FD と、それに続くファイル名、および、必要に応じて、次に続く一連のファイル節から構成される。

ファイル名 (* file-name)
DATA DIVISION の FILE SECTION の中のファイル記述項目またはソート・マージ・ファイル記述項目で記述されるファイル結合子に名前を付けるユーザー定義語。

ファイル編成 (* file organization)
ファイルの作成時に確立される永続的な論理ファイル構造。

ファイル位置標識 (file position indicator)
概念的エンティティーであり、索引付きファイルの場合は参照キー内の現行キーの値、順次ファイルの場合は現行レコードのレコード番号、相対ファイルの場合は現行レコードの相対レコード番号が入っている。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使用可能でないことを示すか、AT END 条件が既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。

*** FILE SECTION**
DATA DIVISION のセクションであり、ファイル記述項目、ソート・マージ・ファイル記述項目、および関連するレコード記述が入っている。

ファイル・システム (file system)
データ・レコードおよびファイル記述プロトコルの特定のセットに準拠するファイルの集合、およびこれらのファイルを管理する一連のプログラム。

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設定され、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、指標付き)、 基本レコード・キー、代替レコード・キー、コード・セット、 最小および最大のレコード・サイズ、レコード・タイプ (固定長、可変長)、 索引付き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、 レコード区切り文字がある。

固定長レコード (* fixed-length record)
ファイル記述項目またはソート・マージ記述項目が、 すべてのレコードのバイトの個数が同じであるように要求しているファイルに関連付けられたレコード。

固定小数点項目 (fixed-point item)
PICTURE 節で定義される数値データ項目であり、オプションの符号の位置、その中に含まれる桁数、およびオプションの小数点の位置を指定するもの。 2 進数、パック 10 進数、または外部 10 進数のいずれかのフォーマットをとることができる。

浮動コメント標識 (*>) (floating comment indicators (*>))
浮動コメント標識がプログラムのテキスト域 (領域 A プラス領域 B) 内の最初の文字ストリングである場合は、この行がコメント行であることを示します。 また、浮動コメント標識がプログラムのテキスト領域内の 1 つ以上の文字ストリングの後にある場合は、インライン・コメントを示します。

浮動小数点 (floating point)
実数を 1 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、固定小数点部分 (最初の数表示) と、暗黙浮動小数点の底を指数で表される数だけ累乗して得られる値 (2 番目の数表示) との積が、実数になります。 例えば、数値 0.0001234 の浮動小数点表記は 0.1234 -3 です (ここで、0.1234 は 小数部であり、-3 は指数です)。

浮動小数点データ項目 (floating-point data item)
小数部と指数が入っている数値データ項目。 その値は、小数部に、指数で指定されただけ累乗された数字データ項目の底を乗算することによって得られる。

フォーマット (* format)
データの集合の特定の配列。

機能 (* function)
ステートメントの実行中に参照された時点で決定される値を持つ、 一時的なデータ項目。

関数 ID (* function-identifier)
関数を参照する文字ストリングと分離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 は、参照修飾子を含むことができる。 英数字関数を参照する関数 ID は、 一定の制限に従いつつ ID が指定できる一般フォーマットの中ならばどこにでも指定できる。 整数関数または数字関数を参照する関数 ID は、算術式が指定できる一般フォーマットの中ならばどこにおいても指定できる。

機能名 (function-name)
必要な引数を伴って関数の値を決定する呼び出しを行うメカニズムに付けられる名前を表すワード。

関数ポインター・データ項目 (function-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 FUNCTION-POINTER 節で定義されるデータ項目に、関数入り口点のアドレスが含められる。 一般的に、C および Java プログラムと通信するために使用される。

G

ガーベッジ・コレクション (garbage collection)
参照されなくなったオブジェクト用のメモリーが Java ランタイム・システムによって自動的に解放されること。

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、またはそのプログラム内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、およびいくつかの特殊レジスターが、グローバル名となり得る。

グローバル参照 (global reference)
メソッドの有効範囲外にあるオブジェクトの参照。

グループ項目 (group item)
(1) 複数の従属データ項目で構成されるデータ項目。 英数字グループ項目 *(alphanumeric group item)* および 国別グループ項目 *(national group item)* を参照。 (2) 国別グループまたは英数字グループとして明示的に (またはコンテキスト で) 限定されていない場合、この用語は一般のグループを指します。

グループ区切り文字 (grouping separator)
読みやすさのために数値を何桁かまとめて区切るのに使用される文字。 デフォルトの文字はコンマです。

H

ヘッダー・ラベル (header label)
(1) 記録メディア・ユニットのデータ・レコードの前にある、データ・セットのラベル。 (2)「ファイル開始ラベル *(beginning-of-file label)*」の同義語。

隠蔽 (メソッド) (hide (a method))
親クラスで同じメソッド名により定義されたファクトリーまたは静的メソッドを (サブクラスで) 再定義すること。 したがって、サブクラスのメソッドは親クラスのメソッドを隠蔽する。

高位終了 (* high-order end)
文字ストリングの左端の文字。

ハイパースペース (hiperspace)
z/OS 環境で、プログラムがバッファーとして使用できる最大 2 GB までの連続する仮想記憶アドレス範囲。

I

IBM COBOL 拡張部分 (IBM COBOL extension)
85 COBOL 標準 に記述されているものの他に、IBM コンパイラーでサポートされる COBOL 構文およびセマンティクス。

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。
IDENTIFICATION DIVISION では、プログラム、クラス、またはメソッドを識別する。
IDENTIFICATION DIVISION には、作成者名、インストール、または日付を含めることができる。

ID (* identifier)
データ項目に名前を付けるための文字ストリングと分離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際には、関数 ID が使われる。

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)
命令の動詞で開始して、行うべき無条件の処置を指定するステートメント。または明示範囲終了符号によって区切られた条件ステートメント (範囲区切りステートメント)。 1 つの命令ステートメントは、一連の命令ステートメントから構成することができる。

暗黙の範囲終了符号 (* implicit scope terminator)
終了していないステートメントが前にある場合、その範囲を区切る分離文字ピリオド。 または、前にある句の中に含まれるステートメントがある場合、 そのステートメントの範囲の終わりをそれが現れること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。

指標 (* index)
コンピューターのストレージ域またはレジスター。 ここにはテーブル内の個々のエレメントを識別するものを表現する内容が入る。

指標データ項目 (* index data item)
指標名に関連する値を、インプリメントする人が指定した形式で収めることができるデータ項目。

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 つまたは複数の指標名から構成される ID。

索引付きファイル (* indexed file)
指標付き編成のファイル。

指標付き編成 (* indexed organization)
各レコードがそのレコードの中にある 1 つまたは複数のキー値によって識別される永続的な論理ファイル構造。

指標付け (indexing)
指標名を使用した添え字付け と同義。

索引名 (* index-name)
特定のテーブルに関連付けられた指標に付ける名前を表すユーザー定義語。

継承 (inheritance)
クラスのインプリメンテーションを、別のクラスを基にして使用するメカニズム。 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。 Enterprise COBOL は 多重継承 をサポートしない。サブクラスは、必ず 1 つの即時スーパークラスを有する。

継承階層 (inheritance hierarchy)
クラス階層 *(class hierarchy)* を参照。

初期設定プログラム (* initial program)
実行単位内にプログラムが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)
プログラムが実行単位の中に呼び出された最初期のそのプログラムの状態。

インライン (inline)
ルーチン、サブルーチン、または他のプログラムに分岐せずに、 連続的に実行されるプログラム内の命令。

インライン・コメント (inline comments)
インライン・コメントは、プログラムのテキスト域にある、前に 1 つ以上の文字ストリングが付いた浮動コメント標識 (*>) で示され、コンパイル・グループの任意の行に書き込むことができます。 浮動コメント標識に続く、領域 B の終わりまでの文字はすべて、コメント・テキストです。

入力ファイル (* input file)
入力モードでオープンされるファイル。

入力モード (* input mode)
ファイルに対する INPUT 句の指定のある OPEN ステートメントが実行されてから、そのファイルに対する REEL または UNIT 句の指定のない CLOSE ステートメントが実行される前までの、ファイルの状態。

入出力ファイル (* input-output file)
I-O モードでオープンされるファイル。

*** INPUT-OUTPUT SECTION**
ENVIRONMENT DIVISION 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付け、実行時にデータの伝送および処理に必要な情報を提供する。

入出力ステートメント (* input-output statement)
個々のレコードに対して操作を行うことにより、またはファイルを 1 つの単位として操作することにより、ファイルの処理を行うステートメント。 入出力ステートメントには、ACCEPT (ID 句付き)、CLOSE、DELETE、DISPLAY、OPEN、READ、REWRITE、SET (TO ON または TO OFF 句付き)、START、および WRITE がある。

入力プロシージャー (* input procedure)
ソートすべき特定のレコードの解放を制御する目的で、形式 1 SORT ステートメントの実行時に制御が渡されるステートメントの集合。

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 OBJECT 段落の DATA DIVISION の WORKING-STORAGE SECTION に定義される。 オブジェクトの状態には、クラスが導入した、現行クラスによって継承されているインスタンス変数の状態も含まれる。 インスタンス・データの個々のコピーは、各オブジェクト・インスタンスごとに作成される。

整数 (* integer)
(1) 小数点の右側に桁位置がない数値リテラル。 (2) DATA DIVISION 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を含まないもの。 (3) 関数の起こりうるすべての評価の戻り値で、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)
カテゴリーが数字で、その定義が小数点の右側に桁位置を持たない関数。

対話式システム生産性向上機能 (ISPF) (Interactive System Productivity Facility (ISPF))
TSO または VM ユーザーに対してメニュー方式のインターフェースを提供する IBM ソフトウェア・プロダクト。ISPF には、ライブラリー・ユーティリティー、強力なエディター、およびダイアログ管理が組み込まれています。

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、各種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簡単に構築することができる。

中間結果 (intermediate result)
連続して行われる算術演算の結果を収める中間フィールド。

内部データ (* internal data)
プログラムの中で記述されるデータで、すべての外部データ項目および外部ファイル結合子を除いたもの。 プログラムの LINKAGE SECTION で記述された項目は、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 つのプログラムの中で記述されるデータ項目。 内部データ項目は、グローバル名を持つことができる。

内部 10 進数データ項目 (internal decimal data item)
USAGE PACKED-DECIMAL または USAGE COMP-3 として記述されており、項目を数値として定義する PICTURE 文字ストリング (記号 9、S、P、または V の有効な組み合わせ) を持っている、データ項目。 「パック *10* 進数データ項目 *(packed-decimal data item)*」と同義。

内部ファイル結合子 (* internal file connector)
実行単位内にあるただ 1 つのオブジェクト・プログラムのみがアクセスできるファイル結合子。

内部浮動小数点データ項目 (internal floating-point data item)
USAGE COMP-1 または USAGE COMP-2 として記述されているデータ項目。 COMP-1 は、単精度浮動小数点データ項目を定義します。 COMP-2 は、倍精度浮動小数点データ項目を定義します。 内部浮動小数点データ項目に関連した PICTURE 節はありません。

レコード内データ構造 (* intrarecord data structure)
連続したデータ記述記入項目のサブセットによって定義される、1 つの論理レコードから得られるグループ・データ項目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、レコード内データ構造を記述している最初のデータ記述記入項目のレベル番号より大きいレベル番号を持つすべての記入項目が含まれる。

組み込み関数 (intrinsic function)
よく使用される算術関数のような事前定義関数で、組み込み関数参照によって呼び出される。

無効キー条件 (* invalid key condition)
索引付きファイルまたは相対ファイルに関連するキーの特定値が無効であると判別された場合に生じる、実行時の条件。

*** I-O-CONTROL**
ENVIRONMENT DIVISION 段落の名前。この段落では、再実行開始点についてのオブジェクト・プログラム要件、複数データ・ファイルによる同じ区域の共用、および単一入出力装置上の複数のファイル・ストレージが指定される。

I-O-CONTROL 記入項目 (* I-O-CONTROL entry)
ENVIRONMENT DIVISION の I-O-CONTROL 段落内の記入項目であり、プログラム実行中に指定のファイルへのデータの伝送と処理を行うために必要な情報を提供する節が入っている。

入出力モード (* I-O mode)
ファイルに対する I-O 句の指定のある OPEN ステートメントが実行されてから、そのファイルに対する REEL または UNIT 句の指定のない CLOSE ステートメントが実行される前までの、ファイルの状態。

入出力状況 (* I-O status)
入出力操作の結果としての状況を示す 2 文字の値を収める概念上のエンティティー。 この値は、そのファイルについてのファイル制御記入項目で FILE STATUS 節を使用することによって、プログラムに使用可能にされる。

is-a
継承階層におけるクラスおよびサブクラスの特徴を表す関係。 あるクラスに対して is-a 関係を持つサブクラスは、そのクラスから継承する。

ISPF
対話式システム生産性向上機能 *(ISPF) (Interactive System Productivity Facility (ISPF))* を参照。

反復構造 (iteration structure)
ある条件が真である間、あるいはある条件が真になるまで、一連のステートメントが繰り返して実行されるプログラムの処理ロジック。

J

J2EE
Java 2 Platform, Enterprise Edition (J2EE) を参照。

Java 2 Platform, Enterprise Edition (J2EE)
エンタープライズ・アプリケーションの開発とデプロイメントのために、Oracle によって定義された環境。 J2EE プラットフォームは、多層の Web ベース・アプリケーションを開発するための機能を提供するサービス、アプリケーション・プログラミング・インターフェース (API)、およびプロトコルで構成されている。 (Oracle)

Java Batch Launcher and Toolkit for z/OS (JZOS)
従来のバッチ環境で実行されて z/OS システム・サービスにアクセスする z/OS Java アプリケーションの開発を支援するツールのセット。

Java バッチ処理プログラム (JBP) (Java batch-processing program (JBP))
オンライン・データベースおよび出力メッセージ・キューにアクセスできる IMS バッチ処理プログラム。JBP はオンラインで実行されますが、バッチ環境のプログラムと同様に、JCL を使用して、または TSO セッション内で開始されます。

Java バッチ処理領域 (Java batch-processing region)
Java バッチ処理プログラムだけがスケジュールされる IMS 従属領域。

Java Database Connectivity (JDBC)
Java プログラムのデータベースへのアクセスを可能にする API を定義する、Oracle の仕様。

Java メッセージ処理プログラム (JMP) (Java message-processing program (JMP))
トランザクションによって駆動され、オンライン IMS データベースとメッセージ・キューにアクセスできる、Java アプリケーション・プログラム。

Java メッセージ処理領域 (Java message-processing region)
Java メッセージ処理プログラムだけがスケジュールされる IMS 従属領域。

Java Native Interface (JNI)
Java 仮想マシン (JVM) 内で実行される Java コードが、他のプログラム言語で記述されたアプリケーションおよびライブラリーと連携できるようにするプログラミング・インターフェース。

Java 仮想マシン (JVM) (Java virtual machine (JVM))
コンパイル済みの Java プログラムを実行する中央演算処理装置のソフトウェア・インプリメンテーション。

JavaBeans
移植可能で、プラットフォームに依存しない、再使用可能なコンポーネント・モデル。 (Oracle)

JBP
Java バッチ処理プログラム (JBP) (Java batch-processing program (JBP)) を参照。

JDBC
Java Database Connectivity (JDBC) を参照。

JMP
Java メッセージ処理プログラム (JMP) (Java message-processing program (JMP)) を参照。

ジョブ制御言語 (JCL) (job control language (JCL))
ジョブをオペレーティング・システムに識別させ、ジョブの要件を記述するために使われる制御言語。

JSON
JSON (JavaScript Object Notation) とは、単純なデータ交換フォーマットである。

JVM
Java 仮想マシン *(JVM) (Java virtual machine (JVM))* を参照。

JZOS
Java Batch Launcher と *Toolkit for z/OS* を参照。

K

K
記憶容量に関連して使用されるときは、2 の 10 乗。 10 進表記では 1024。

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)
索引付きファイルの中のレコードをアクセスするために現在使用されている基本キーまたは代替キー。

キーワード (* keyword)
コンテキスト・センシティブ語または予約語。その語の表示フォーマットがソース単位で使用されるときは、その語は必須である。

キロバイト (KB) (kilobyte (KB))
1 キロバイトは 1024 バイトに相当する。

L

言語名 (* language-name)
特定のプログラミング言語を指定するシステム名。

Language Environment
z/OS 言語環境プログラム 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。 言語環境プログラム 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。

言語環境プログラム 準拠 (Language Environment-conforming)
言語環境プログラム の規約に準拠したオブジェクト・コードを生成するコンパイラー製品 (Enterprise COBOL、COBOL (OS/390 および VM 版)、COBOL (MVS および VM 版)、C/C++ for MVS & VM、PL/I for MVS & VM など) の特性。

最後に使われた状態 (last-used state)
内部値がプログラム終了時と同じままで、初期値にリセットされない、プログラムの状態を言う。

文字 (* letter)
以下の 2 つのセットのいずれかに属する文字。

1. 英大文字: 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. 英小文字: 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。 DATA DIVISION 内のレベル標識には、CD、FD、および SD がある。

レベル番号 (* level-number)
階層構造におけるデータ項目の位置を示すか、またはデータ記述記入項目の特性を示す、2 桁の数字で表されたユーザー定義語。 1 から 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 から 9 のレベル番号は、1 桁の数字として書き込むことも、 0 の後に有効数字を書き込むこともできる。 レベル番号 66、77、および 88 は、データ記述項目の特性を識別する。

ライブラリー名 (* library-name)
COBOL ライブラリーの名前を表すユーザー定義語。与えられたソース・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を識別する。

ライブラリー・テキスト (* library text)
COBOL ライブラリーの中にある一連のテキスト・ワード、コメント行、 インライン・コメント、区切り文字のスペース、または区切り文字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。 リリアン日フォーマットは、 グレゴリオ暦の考案者であるルイジ・リリオにちなんだ名称。

*** LINAGE-COUNTER**
ページ本体内の現在位置を指す値を収めた特殊レジスター。

リンク (link)
(1) リンク接続 (伝送メディア) と、それぞれがリンク接続の終端にある 2 つのリンク・ステーションの組み合わせ。1 つのリンクは、マルチポイントまたはトークンリング構成において、複数のリンク間で共用できる。 (2) データ項目あるいは 1 つ以上のコンピューター・プログラムの部分を相互接続すること。例えば、リンケージ・エディターによってオブジェクト・プログラムをリンクして実行可能ファイルを作成すること。

LINKAGE SECTION
呼び出し先のプログラムまたはメソッドの DATA DIVISION 内のセクションであり、呼び出し側プログラムまたはメソッドから使用可能なデータ項目が記述される。 これらのデータ項目は、呼び出し側プログラムまたはメソッドおよび呼び出し先プログラムまたはメソッドの両方から参照できる。

リンカー (linker)
z/OS バインダー (リンケージ・エディター) を指す用語。

リテラル (literal)
ストリングを構成するために配列された文字によって、または形象定数を使用することによって、その値が決められる文字ストリング。

リトル・エンディアン (little-endian)
Intel プロセッサーが 2 進データおよび UTF-16 文字を保管するために使用するデフォルト形式。 この形式では、2 進数データ項目の最上位バイトが最上位のアドレスになり、 UTF-16 文字の最上位バイトが最上位のアドレスになる。 ビッグ・エンディアン *(big-endian)* と比較。

ローカル参照 (local reference)
メソッドの有効範囲内にあるオブジェクトの参照。

ロケール (locale)
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、通貨表記、数値表記、または言語など。

*** LOCAL-STORAGE SECTION**
DATA DIVISION のセクションであり、VALUE 節で割り当てられた値に応じて、呼び出し単位で割り振りまたは解放が行わ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、基本項目またはグループ項目のどちらでもよい。 「レコード *(record)*」と同義。

下位終了 (* 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)
プログラムとサブルーチンからなる階層において、プロセス内でプログラムが実行されたときに最初に制御を受け取るプログラム。

Make ファイル (makefile)
アプリケーションに必要なファイルのリストが収められたテキスト・ファイル。 make ユーティリティーはこのファイルを使用して、ターゲット・ファイルを最新の変更で更新する。

大容量記憶 (* mass storage)
データを順次と非順次の 2 つの方法で編成して保管しておくことができるストレージ・メディア。

大容量記憶装置 (* mass storage device)
磁気ディスクなど、大きな記憶容量を持つ装置。

大容量記憶ファイル (* mass storage file)
大容量記憶メディアに格納されたレコードの集合。

メガバイト、MB (* megabyte (MB))
1 メガバイトは 1,048,576 バイトに相当する。

マージ・ファイル (* merge file)
MERGE ステートメントによってマージされるレコードの集まり。 マージ・ファイルは、マージ機能により作成され、マージ機能によってのみ使用できる。

メッセージ処理プログラム (MPP) (message-processing program (MPP))
トランザクションによって駆動され、オンライン IMS データベースとメッセージ・キューにアクセスできる、IMS アプリケーション・プログラム。

メッセージ・キュー (message queue)
メッセージがアプリケーション・プログラムによって処理されたり端末に送信されたりする前に、 キューに入れられるデータ・セット。

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 つを定義し、そのオブジェクトに対する INVOKE ステートメントによって実行されるプロシージャー・コード。

メソッド定義 (* method definition)
メソッドを定義する COBOL ソース・コード。

メソッド見出し記入項目 (* method identification entry)
IDENTIFICATION DIVISION の METHOD-ID 段落内の記入項目。この記入項目には、メソッド名を指定する節が入っている。

メソッドの起動 (method invocation)
あるオブジェクトから別のオブジェクトへの通信で、受信オブジェクトにメソッドを実行するように要求するもの。

メソッド名 (method-name)
オブジェクト指向操作の名前。 メソッドを起動するのに使用する場合、名前は、英数字リテラル、国別リテラル、カテゴリー英数字データ項目、またはカテゴリー国別データ項目にすることができます。メソッドを定義する METHOD-ID 段落で使用する場合、名前は英数字リテラルまたは国別リテラルにする必要があります。

メソッド隠蔽 (method hiding)
「隠蔽 *(hide)*」を参照。

メソッド多重定義 (method overloading)
「多重定義 *(overload)*」を参照。

メソッドのオーバーライド (method overriding)
「オーバーライド *(override)*」を参照。

簡略名 (* mnemonic-name)
ENVIRONMENT DIVISION 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。

モジュール定義ファイル (module definition file)
プログラム・オブジェクト内のコード・セグメントを記述するファイル。

MPP
メッセージ処理プログラム *(MPP) (message-processing program (MPP))* を参照。

マルチタスキング (multitasking)
2 つ以上のタスクの並行実行またはインターリーブド実行を可能にする操作モード。

マルチスレッド化 (multithreading)
コンピューター内で複数のパスを使用して実行を行う並行操作。 「マルチプロセッシング *(multiprocessing)*」と同義。

N

名前 (name)
COBOL オペランドを定義する 30 文字を超えないで構成されたワード。

ネーム・スペース (namespace)
XML 名前空間 *(XML namespace)* を参照。

国別文字 (national character)
(1) 国別リテラルまたは USAGE NATIONAL の UTF-16 文字。 (2) UTF-16 で表される任意の文字。

国別文字データ (national character data)
UTF-16 で表されるデータの一般参照。

国別文字位置 (national character position)
文字位置 *(character position)* を参照。

国別データ (national data)
「国別文字データ *(national character data)*」を参照。

国別データ項目 (national data item)
カテゴリー国別、国別編集、または USAGE NATIONAL の数字編集のデータ項目。

国別 10 進数データ項目 (national decimal data item)
暗黙的または明示的に USAGE NATIONAL として記述されており、PICTURE の記号 9、S、P、および V の有効な組み合わせを含んでいる、外部 10 進数データ項目。

国別編集データ項目 (national-edited data item)
少なくとも 1 つの N のインスタンスおよび単純挿入記号 B、0、または / の少なくとも 1 つを含んでいる PICTURE 文字ストリングで記述されている、データ項目。 国別編集データ項目は USAGE NATIONAL を持ちます。

国別浮動小数点データ項目 (national floating-point data item)
暗黙的または明示的に USAGE NATIONAL として記述されており、浮動小数点データ項目を記述する PICTURE 文字ストリングを持っている、外部浮動小数点データ項目。

国別グループ項目 (national group item)
明示的または暗黙的に GROUP-USAGE NATIONAL 節で記述されたグループ項目。 国別グループ項目は、INSPECT、STRING、および UNSTRING などの操作で、カテゴリー国別の基本データ項目として定義されているかのように処理されます。 英数字グループ項目内で USAGE NATIONAL データ項目を定義するのとは対照的に、この処理により、国別文字の埋め込みおよび切り捨てが確実に正しく行われます。 グループ内の基本項目を処理する必要のある操作 (MOVE CORRESPONDING、ADD CORRESPONDING、および INITIALIZE など) の場合、国別グループは グループ・セマンティクスを使用して処理されます。

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。

ネイティブ・メソッド (native method)
COBOL などの別のプログラム言語で記述されたインプリメンテーションを備える Java メソッド。

複合否定条件 (* negated combined condition)
論理演算子 NOT とその直後に括弧で囲んだ複合条件を続けたもの。 条件 *(condition)* および 複合条件 *(combined condition)* も参照。

単純否定条件 (* negated simple condition)
論理演算子 NOT とその直後に単純条件を続けたもの。 条件 *(condition)* および 単純条件 *(simple condition)* も参照。

ネストされたプログラム (nested program)
他のプログラムの中に直接的に含まれているプログラム。

次の実行可能文 (* next executable sentence)
現在のステートメントの実行完了後に制御が移される次の文。

次の実行可能なステートメント (* next executable statement)
現在のステートメントの実行完了後に制御が移される次のステートメント。

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)
WORKING-STORAGE SECTION および LINKAGE SECTION 内の基本データ項目で、他のデータ項目と階層上の関係を持たないもの。

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 。

ヌル (null)
無効なアドレスの値をポインター・データ項目に割り当てるために使用される形象定数。 NULL を使えるところならばどこでも、NULLS を使用できる。

数字 (* numeric character)
次のような数字に属する文字。 0、1、2、3、4、5、6、7、8、9。

数値データ項目 (numeric data item)
(1) 記述により内容が数字 0 から 9 より選ばれた文字で表される値に制限されるデータ項目。符号付きである場合、この項目は +、-、または他の表記の演算符号も含むことができます。 (2) カテゴリー数値、内部浮動小数点、または外部浮動小数点のデータ項目。 数値データ項目は、USAGE DISPLAY、NATIONAL、PACKED-DECIMAL、BINARY、COMP、COMP-1、COMP-2、COMP-3、COMP-4、または COMP-5 を持つことができます。

数字編集データ項目 (numeric-edited data item)
印刷出力の際に使用するのに適したフォーマットの数値データを含むデータ項目。 データ項目は、外部 10 進数字の 0 から 9 の数字、小数点、コンマ、通貨符号、符号制御文字、その他の編集記号から構成される。 数字編集項目は、USAGE DISPLAY または USAGE NATIONAL のいずれかで表すことができる。

数字関数 (* numeric function)
クラスとカテゴリーは数字だが、考えられる評価のいくつかにおいて整数関数の要件を満たさないような関数。

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)
1 つ以上の数字から構成されるリテラルで、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。 小数点は右端の文字であってはならない。 代数符号がある場合には、それが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手段である。 クラス内の各オブジェクトは、そのクラスの 1 つのインスタンスであると言われる。

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。 それ自体が実行可能なマシン・コードか、 またはその種のコードの作成を目的としての処理に適する。

*** OBJECT-COMPUTER**
ENVIRONMENT DIVISION にある段落の名前であり、ここではオブジェクト・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)
ENVIRONMENT DIVISION の OBJECT-COMPUTER 段落内の記入項目。この記入項目には、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を記述する節が入っている。

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・インスタンス (object instance)
単一の、場合によっては多数のオブジェクト。COBOL クラス定義の Object 段落における指定に基づいて インスタンス生成される。 オブジェクト・インスタンスは、そのクラス定義に記述されたデータおよびすべての継承データのコピーを保持する。 オブジェクト・インスタンスに関連付けられたメソッドには、そのクラス定義で定義されたメソッドおよびすべての 継承メソッドが含まれる。

オブジェクト・インスタンスは Java クラスのインスタンスにすることができる。

オブジェクト・モジュール (object module)
オブジェクト・デック *(object deck)* または テキスト・デック *(text deck)* と同義。

項目のオブジェクト (* object of entry)
COBOL プログラムの DATA DIVISION 記入項目内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。

オブジェクト指向プログラミング (object-oriented programming)
カプセル化および継承の概念に基づいたプログラミング・アプローチ。 プロシージャー型プログラミング技法とは異なり、オブジェクト指向プログラミングでは、何かが達成される方法ではなく、問題を含むデータ・オブジェクトとその操作方法に重点を置く。

オブジェクト・プログラム (object program)
問題を解決するためにデータと相互に作用することを目的とする実行可能なマシン言語命令とその他の要素の集合または グループ。 このコンテキストでは、オブジェクト・プログラムとは一般に、COBOL コンパイラーがソース・プログラムまたはクラス定義を操作した 結果得られるマシン言語である。 あいまいになる危険がない場合には、オブジェクト・プログラム という用語の代わりに プログラム というワードだけが使用される。

オブジェクト・リファレンス (object reference)
クラスのインスタンスを識別する値。 クラスが指定されなかった場合、オブジェクト参照は一般的なものとなり、任意のクラスのインスタンスに適用できる。

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。 実行時 *(run time)* と同義。

廃止される言語エレメント (* obsolete element)
2002 COBOL 標準 から削除された 85 COBOL 標準 の COBOL 言語エレメント。

ODO オブジェクト (ODO object)
次の例では、X が OCCURS DEPENDING ON 節のオブジェクト (ODO オブジェクト) である。

```
WORKING-STORAGE SECTION.
01  TABLE-1.
    05  X                   PIC S9.
    05  Y OCCURS 3 TIMES
          DEPENDING ON X    PIC X.
```

ODO オブジェクトの値によって、テーブル内の ODO サブジェクトの数が決まる。

ODO 対象 (ODO subject)
上記の例では、Y が OCCURS DEPENDING ON 節の サブジェクト (ODO サブジェクト) である。 テーブル内の ODO サブジェクトの数である Y の値は、X の値によって決まる。

オープン・モード (* open mode)
OPEN ステートメントが実行されてから、REEL および UNIT 句の指定のない CLOSE ステートメントが実行される前までのファイルの状態。 個々のオープン・モードは、OPEN ステートメントの中で、INPUT、OUTPUT、I-O、または EXTEND のいずれかとして指定する。

オペランド (* operand)
(1) オペランドの一般的な定義は、「操作の対象となるコンポーネント」である。 (2) 本書の目的に沿った言い方をすれば、ステートメントや記入項目の形式中に現れる小文字または日本語で書かれた語 (または語群) はオペランドと見なされ、そのオペランドによって指示されたデータに対して暗黙の参照を行う。

演算、操作 (operation)
オブジェクトに関して要求できるサービス。

演算符号 (* operational sign)
値が正であるか負であるかを示すために数字データ項目または数値リテラルに付けられる代数符号。

オプション・ファイル (optional file)
オブジェクト・プログラムが実行されるたびに必ずしも使用可能でなくてもよいものとして宣言されているファイル。

オプショナル・ワード (* optional word)
言語を読みやすくする目的でのみ特定の形式で含められる予約語。 このようなワードが表示されている形式をソース単位内で使用する場合、そのワードの有無はユーザーが選択できる。

出力ファイル (* output file)
出力モードまたは拡張モードのいずれかでオープンされるファイル。

出力モード (* output mode)
OUTPUT または EXTEND 句の指定のある OPEN ステートメントが実行されてから、 REEL および UNIT 句の指定のない CLOSE ステートメントが実行される前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が完了した後で制御が渡されるステートメントの集合、または MERGE ステートメントの実行中に、要求があればマージ機能がマージ済みの順序になっているレコードのうち次のレコードを選択できるようになった後で制御が渡されるステートメントの集合。

オーバーフロー条件 (overflow condition)
ある演算結果の一部が意図した記憶単位の容量を超えたときに起こる条件。

多重定義 (overload)
同じクラスで使用可能な別のメソッドと同一の名前を使い (ただし、異なるシグニチャーを使用して)、メソッドを定義すること。 シグニチャー *(signature)* も参照。

指定変更 (override)
サブクラスのインスタンス・メソッド (親クラスから継承された) を再定義すること。

P

パッケージ (package)
関連する Java クラスの集まり。個々に、または全体としてインポートすることができる。

パック 10 進数データ項目 (packed-decimal data item)
内部 *10* 進数データ項目 *(internal decimal data item)* を参照。

埋め込み文字 (padding character)
物理レコード内の未使用文字位置を埋めるのに使用される英数字または国別文字。

ページ (page)
データの物理的分離を表す、出力データの垂直分割。 分離は、内部論理要件または出力メディアの外部特性、あるいはその両方に基づいて行われる。

ページ本体 (* page body)
行を記述できる、または行送りすることができる (またはその両方ができる) 論理ページの部分。

段落 (* paragraph)
PROCEDURE DIVISION では、段落名の後に分離文字ピリオドが続き、その後に 0 個以上の文が続く。 IDENTIFICATION DIVISION および ENVIRONMENT DIVISION では、段落ヘッダーの後に 0 個以上の記入項目が続く。

段落ヘッダー (* paragraph header)
予約語の後に分離文字ピリオドが付いたもので、IDENTIFICATION DIVISION および ENVIRONMENT DIVISION において段落の始まりを示すもの。 IDENTIFICATION DIVISION で許可されている段落ヘッダーは次のとおり。

```
PROGRAM-ID. (Program IDENTIFICATION
   DIVISION)
CLASS-ID. (Class IDENTIFICATION DIVISION)
METHOD-ID. (Method IDENTIFICATION
   DIVISION)
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
```

ENVIRONMENT DIVISION で許可されてい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)
PROCEDURE DIVISION の中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)
(1) 呼び出し側プログラムと呼び出し先プログラムの間で受け渡されるデータ。 (2) メソッド呼び出しの USING 句内のデータ・エレメント。引数によって、呼び出されたメソッドが要求された操作を実行するために使用できる追加情報を与える。

Persistent Reusable JVM
トランザクション間で JVM をリセットすることによりトランザクション処理用にシリアルに再利用できる JVM。 リセット・フェーズでは、JVM が既知の初期状態に復元される。

句 (* phrase)
連続する 1 つ以上の COBOL 文字ストリングを配列したセットで、COBOL プロシージャー・ステートメントまたは COBOL 節の一部を構成する。

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。 これらのデータ項目は、USAGE IS POINTER 節を使用して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ている。 ポインター・データ項目は、 他のポインター・データ項目と等しいかどうかを比較したり、 他のポインター・データ項目に内容を移動することができる。

移植する、ポート (port)
(1) 異なるプラットフォームで実行できるようにコンピューター・プログラムを変更すること。 (2) インターネット・プロトコルでは、Transmission Control Protocol (TCP) プロトコルまたは User Datagram Protocol (UDP) プロトコルと高水準のプロトコルまたはアプリケーションの間の特定の論理結合子。 ポートはポート番号によって識別される。

可搬性 (portability)
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・プラットフォームに、ソース・プログラムに比較的わずかな変更を加えるだけでアプリケーション・プログラムを移行できる能力。

合成済み文字 (precomposed character)
標準分解により複数の Unicode 文字を使用して表すことができる単一の Unicode 文字。 合成済み文字は合成文字形式と同じ物理表記を持たない。 例えば、Unicode 文字 U+00E4 (ä) は、Unicode 文字 U+0061 + U+0308 (ä) (ラテン語小文字 a + 結合発音区別符号) の組み合わせとして表すことができる

合成済み文字。 通常、合成済み文字は、発音区別符号を持つラテン語文字や他の結合文字を表すために使用される。

事前初期設定 (preinitialization)
プログラム (特に非 COBOL プログラム) からの複数の呼び出しの準備として の COBOL ランタイム環境の初期設定。 この環境は、明示的に終了されるまで終了されない。

基本レコード・キー (* prime record key)
索引付きファイルのレコードを固有なものとして識別する内容を持つキー。

優先順位番号 (* priority-number)
セグメンテーションの目的で、PROCEDURE DIVISION 内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 から 9 までの文字だけしか使用できない。セグメント番号は 1 桁または 2 桁として表すことができる。

private
ファクトリー・データまたはインスタンス・データに適用されるため、そのデータを定義するクラスのメソッドだけがアクセス可能である。

プロシージャー (* procedure)
PROCEDURE DIVISION 内にある 1 つの段落または論理的に連続する段落のグループ、あるいは 1 つのセクションまたは論理的に連続するセクションのグループ。

プロシージャー・ブランチ・ステートメント (* procedure branching statement)
ソース・コードの中にステートメントが書かれている順番どおりに次の実行可能ステートメントに制御の移動をせず、別のステートメントに明示的に制御の移動を引き起こすステートメント。 プロシージャー分岐ステートメントは次のとおり。ALTER、CALL、EXIT、EXIT PROGRAM、GO TO、MERGE (OUTPUT PROCEDURE 句付き)、PERFORM および SORT (INPUT PROCEDURE または OUTPUT PROCEDURE 句付き)、XML PARSE。

PROCEDURE DIVISION
COBOL の部の 1 つで、問題を解決するための命令を記述する。

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 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によって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呼び出しがプログラム・コードによって置き換えられるプロセスのこと。

プロシージャー名 (* procedure-name)
PROCEDURE DIVISION の中にある段落またはセクションに名前を付けるために使用される ユーザー定義語。 プロシージャー名は、段落名 (これは修飾することができる) またはセクション名から構成される。

プロシージャー・ポインター (procedure pointer)
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 節を付けて定義したデータ項目が、 プロシージャーへの入り口点のアドレスを収める。

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 節で定義されるデータ項目には、プロシージャー入り口点のアドレスが入っている。一般的に、COBOL および言語環境プログラムのプログラムと通信するために使用される。

プロセス (process)
プログラムの全部または一部の実行中に発生する一連のイベント。 複数のプロセスを並行して実行することができ、1 つのプロセス内で実行されるプログラムはリソースを共用することができる。

プログラム (program)
(1) コンピューターによる処理に適した一連の命令。 処理には、コンパイラーを使用してプログラムの実行準備をすることやランタイム環境を使用してプログラムを実行することが含まれます。 (2) 1 つ以上の相互に関係のあるモジュールの論理アセンブリー。 同じプログラムの複数のコピーを異なるプロセスで実行することができます。

プログラム名 (program-name)
IDENTIFICATION DIVISION と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。

プログラム識別記入項目 (* program identification entry)
IDENTIFICATION DIVISION の PROGRAM-ID 段落内の記入項目であり、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節が入っている。

プログラム名 (program-name)
IDENTIFICATION DIVISION 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。

プロジェクト (project)
ダイナミック・リンク・ライブラリー (DLL) や他の実行可能ファイル (EXE) などのターゲットを作成するのに必要な、データおよびアクションの完全セット。

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、インライン・コメント、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隣接した 2 つの等号文字 (==)。

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、処理を待機する入力データ・ブロック、 または処理が終了して補助ストレージまたは出力装置への転送を待機する出力データ・ブロックで形成される。

修飾されたデータ名 (* qualified data-name)
データ名と、その後に連結語の OF または IN とデータ名修飾子を続けたものが 1 つ以上のセットで続いて構成される ID。

修飾子 (* qualifier)
(1) レベル標識と関連付けられるデータ名または名前であり、参照の際に、別のデータ名 (修飾子に従属する項目の名前) と 一緒に、または条件名と一緒に使用される。 (2) セクション名。 そのセクションの中で指定されている段落名と共に参照する際に使用される。 (3) ライブラリー名。 そのライブラリーと関連付けられたテキスト名と共に参照する際に使用される。

R

ランダム・アクセス (* random access)
キー・データ項目のプログラム指定値を使って、 相対ファイルまたは索引付きファイルから取り出したり、削除したり、 またはそこに入れたりす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)
DATA DIVISION の FILE SECTION 内のレコード記述項目で記述されるレコードを処理する目的で割り振られるストレージ域。 FILE SECTION では、レコード域の現行の文字位置の数は、明示または暗黙の RECORD 節によって決められる。

レコード記述 (* record description)
レコード記述項目 *(record description entry)* を参照。

レコード記述項目 (* record description entry)
特定のレコードに関連したデータ記述項目全体。 「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。 レコード・モードは、F (固定長)、V (可変長)、S (スパン)、 または U (不定フォーマット) とすることができる。

レコード・キー (record key)
索引付きファイル内のレコードを識別する内容を持つキー。

レコード名 (* record-name)
COBOL プログラムの DATA DIVISION 内のレコード記述項目で記述されるレコードに名前を付けるユーザー定義語。

レコード番号 (* record number)
編成が順次であるファイル内のレコードの順序数。

レコード・モード (recording mode)
ファイル内の論理レコードの形式。 記録モードは、F (固定長)、V (可変長)、S (スパン)、または U (不定形式) とすることができる。

再帰 (recursion)
それ自体を呼び出すプログラム、または、それ自体で呼び出したプログラムのいずれかによって直接あるいは間接に呼び出されるプログラム。

再起可能 (recursively capable)
PROGRAM-ID ステートメントで RECURSIVE 属性が指定されていれば、プログラムは再帰可能である (再帰的に呼び出すことができる)。

リール (reel)
ストレージ・メディアの個別部分。その大きさはインプリメントする人によって決定され、1 つのファイルの一部、1 つのファイルの全部、または任意の個数のファイルが収容される。 「ユニット *(unit)*」および「ボリューム *(volume)*」と同義。

再入可能 (reentrant)
プログラムまたはルーチンの属性。この属性によって、プログラム・オブジェクトの 1 つのコピーを複数のユーザーが共用できる。

参照形式 (* reference format)
COBOL ソース・プログラムを記述するに際して標準的な方式を提供する形式。

参照変更 (reference modification)
新規のカテゴリー英数字、カテゴリー DBCS、またはカテゴリー国別のデータ 項目を定義する方法であり、USAGE DISPLAY、DISPLAY-1、または NATIONAL データ項目の左端文字および左端文字位置を基準にした長さを指定して定義する方法です。

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分離文字の構文的に正しい組み合わせ。 区切り用の左括弧区切り文字、左端の文字位置、区切り文字のコロン、 任意指定の長さ、および区切り用の右括弧区切り文字を含む。

関係 (* relation)
関係演算子 *(relational operator)* または 比較条件 *(relation condition)* を参照。

比較文字 (* relation character)
以下のセットに属する文字。

文字	意味
>	より大きい
<	より小さい
=	に等しい

比較条件 (* relation condition)
ある算術式、データ項目、英数字リテラル、または索引名の値が、他の算術式、データ項目、英数字リテラル、または索引名の 値と特定の関係があるという命題 (それに対して真理値を判別する)。 関係演算子 *(relational operator)* も参照。

比較演算子 (* 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。

文字	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	に等しい
IS =	に等しい
IS NOT EQUAL TO	に等しくない
IS NOT =	に等しくない
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

相対ファイル (* relative file)
相対編成のファイル。

相対キー (* relative key)
相対ファイルの中の論理レコードを識別するための内容を持つキー。

相対編成 (* relative organization)
各レコードが、レコードのファイル内における論理的順序位置を指定する 0 より大きい整数値によって、固有なものとして識別される永続的な論理ファイル構造。

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。 この数値は、整数の数値リテラルとして扱われる。

予約語 (* reserved word)
COBOL ソース・プログラムの中で使用することができるが、ユーザー定義語またはシステム名としてプログラムの中で使用されてはならないワードのリスト中に挙げられている COBOL ワード。

リソース (* resource)
オペレーティング・システムの制御下に置かれており、実行中のプログラムによって使用できる機能またはサービス。

結果の ID (* resultant identifier)
算術演算の結果が収められるユーザー定義のデータ項目。

再使用可能環境 (reusable environment)
事前初期設定用の古い COBOL インターフェース (RTEREUS ランタイム・オプション)、または言語環境プログラム・インターフェース CEEPIPI 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設定するときに、再使用可能環境が作成されます。

ルーチン (routine)
コンピューターに操作または一連の関連操作を実行させる、COBOL プログラム内の一連のステートメント。 言語環境プログラムでは、プロシージャー、関数、またはサブルーチンのいずれかを指す。

ルーチン名 (* routine-name)
COBOL 以外の言語で記述されたプロシージャーを識別するユーザー定義語。

実行時 (* run time)
オブジェクト・プログラムが実行されるとき。 「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)
1 つの独立型オブジェクト・プログラム、あるいは COBOL の CALL または INVOKE ステートメントによって相互作用し、実行時に 1 つのエンティティーとして機能する複数のオブジェクト・プログラム。

S

SBCS
1 バイト文字セット *(SBCS) (single-byte character set (SBCS))* を参照。

範囲終了符号 (scope terminator)
PROCEDURE DIVISION の特定のステートメントの終わりを示す COBOL 予約語。これは明示的なもの (例えば、END-ADD など) であることもあれば、暗黙のもの (分離文字ピリオド) であることもある。

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・ヘッダーとそれに関連付けられたセクション本体から構成される。

セクション・ヘッダー (* section header)
後ろに分離文字ピリオドが付いたワードの組み合わせで あり、ENVIRONMENT、DATA、または PROCEDURE の各部において、セクションの始まりを示すもの。 ENVIRONMENT DIVISION および DATA DIVISION では、 セクション・ヘッダーは、予約語の後に分離文字ピリオドを続けたものから構成される。 ENVIRONMENT DIVISION で許可されているセクション・ヘッダーは次のとおり。

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

DATA DIVISION で許可されているセクション・ヘッダーは次のとおり。

```
FILE SECTION.
WORKING-STORAGE SECTION.
LOCAL-STORAGE SECTION.
LINKAGE SECTION.
```

PROCEDURE DIVISION では、セクション・ヘッダーは、セクション名、その後に続く予約語 SECTION、およびその後の分離文字ピリオドから構成される。

セクション名 (* section-name)
PROCEDURE DIVISION の中にあるセクションに名前を付けるユーザー定義語。

セグメント化 (segmentation)
85 COBOL 標準 分割モジュールに基づく Enterprise COBOL の機能。 セグメンテーション機能は、セクション・ヘッダーの優先順位番号を使用して、セクションを固定セグメントまたは独立セグメントに割り当てる。 セグメント種別は、セグメントに含まれるプロシージャーが初期状態の制御を受け取るか、最後に使われた状態の制御を受け取るかに影響を及ぼす。

選択構造 (selection structure)
条件が真であるか偽であるかに応じて、ある一連のステートメントか、または別の一連のステートメントが実行されるというプログラムの処理ロジック。

文 (* sentence)
1 つ以上のステートメントの並びで、その最後のものは、分離文字ピリオドで終了する。

別々にコンパイルされたプログラム (* separately compiled program)
あるプログラムをそこに含まれたプログラムと共に、 他のすべてのプログラムとは別個にコンパイルしたときのそのプログラム。

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 文字または連続する 2 文字以上。

区切り文字のコンマ (* separator comma)
文字ストリングを区切るために使われる、後ろに 1 つのスペースが続く 1 つのコンマ (,)。

分離文字ピリオド (* separator period)
文字ストリングを区切るために使われる、後ろに 1 つのスペースが続く 1 つのピリオド (.)。

区切り文字のセミコロン (* separator semicolon)
文字ストリングを区切るために使われる、後ろに 1 つのスペースが続く 1 つのセミコロン (;)。

順序構造 (sequence structure)
一連のステートメントが、順序どおりに実行されるプログラムの処理ロジック。

順次アクセス (* sequential access)
ファイル内のレコードの並び方によって規定されている、 論理レコードの連続した前後関係順に、論理レコードをファイルから取り出したり、 ファイルに書き込んだりするアクセス・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書き込まれるときに確定されたレコードの前後関係によって識別されるような永続的な論理ファイル構造。

逐次探索 (serial search)
最初のメンバーから始めて最後のメンバーで終わるように、 ある集合のメンバーが連続的に検査される探査方法。

Session Bean
EJB において、クライアントによって作成され、通常は 1 つのクライアント/サーバー・セッションの期間だけ存在する Enterprise Bean。 (Oracle)

77 レベル記述記入項目 (77-level-description-entry)
レベル番号 77 を持つ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)
データ項目や算術式の代数値が、0 より小さいか、大きいか、または等しいかという命題で、それに関して真理値が判別できる。

シグニチャー (signature)
(1) ある操作とそのパラメーターの名前。 (2) あるメソッドの名前と その仮パラメーターの数と型。

単純条件 (* simple condition)
以下のセットから選択される任意の単一条件。

- 比較条件
- クラス条件

- 条件名条件
- スイッチ状況条件
- 符号条件

条件 *(condition)* および 単純否定条件 *(negated simple condition)* も参照。

1 バイト文字セット (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。 *ASCII* および *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 (レコード内) (slack bytes (within records))
複数の基本データ項目を正しく位置合わせするために、コンパイラーによってデータ項目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。 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 節によって、コン パイラーに遊びバイトを挿入させる。

遊びバイト (レコード間) (slack bytes (between records))
複数の基本データ項目を正しく位置合わせするために、プログラマーによってファイルのブロック化論理レコードの間に挿入されるバイト。 場合によっては、レコード間に遊びバイトを挿入することによってバッファー内で処理されるレコードのパフォーマンスが改善される。

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まり。ソート・ファイルは、ソート機能によってのみ作成され使用される。

ソート・マージ・ファイル記述項目 (* sort-merge file description entry)
DATA DIVISION の FILE SECTION の中にある記入項目。レベル標識 SD と、それに続くファイル名、および、必要に応じて、次に続く一連のファイル節から構成される。

*** SOURCE-COMPUTER**
ENVIRONMENT DIVISION にある段落の名前であり、ここではソース・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)
ENVIRONMENT DIVISION の SOURCE-COMPUTER 段落内の記入項目であり、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節が入っている。

ソース項目 (* source item)
SOURCE 節によって指定される ID で、印刷可能な項目の値を提供する。

ソース・プログラム (source program)
ソース・プログラムは、他の形式や記号を使用して表現することができるが、本書では、構文的に正しい COBOL ステートメントの集合を常に指している。 COBOL ソース・プログラムは、IDENTIFICATION DIVISION または COPY ステートメントで開始され、指定された場合はプログラム終了マーカーで終了するか、または追加のソース・プログラム行なしで終了する。

ソース単位 (source unit)
COBOL ソース・コードの 1 単位で、個別にコンパイルできる。プログラムまたはクラス定義。 コンパイル単位 とも呼ばれる。

特殊文字 (special character)
以下のセットに属する文字。

文字	意味
+	正符号
-	負符号 (-) (ハイフン)
*	アスタリスク
/	斜線 (スラッシュ)
=	等号
$	通貨符号
,	コンマ

文字	意味
;	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

SPECIAL-NAMES
ENVIRONMENT DIVISION にある段落の名前。この段落では、環境名がユーザー指定の簡略名と関連付けられる。

特殊名記入項目 (* special names entry)
ENVIRONMENT DIVISION の SPECIAL-NAMES 段落内の記入項目。この記入項目は、通貨記号を指定したり、小数点を選択したり、シンボリック文字を指定したり、インプリメントする人の名前をユーザー指定の簡略名と関連付けたり、英字名を文字セットまたは照合シーケンスと関連付けたり、クラス名を一連の文字と関連付けたりするための手段を提供する。

特殊レジスター (* special registers)
コンパイラーの生成する特定のストレージ域のことで、 その基本的な使用法は、具体的な COBOL 機能を使用したときに作り出される情報を記憶することである。

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、動詞を冒頭に置いた、ワード、リテラル、および区切り記号の構文的に正しい組み合わせ。

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、構造の下方へと渡され、 階層内のより上位レベルへの無条件ブランチは行われない。

サブクラス (* subclass)
別のクラスから継承するクラス。 継承関係にある 2 つのクラスをまとめて考える場合、継承する側、つまり継承先のクラスをサブクラスといい、継承される側、つまり継承元のクラスをスーパークラスという。

項目のサブジェクト (* subject of entry)
DATA DIVISION の記入項目内において、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、(オプションで演算子 + または - 付きの整数が後ろにある) データ名、あるいは (オプションで演算子 + または - 付きの整数が後ろにある) 索引名のいずれかによって表されるオカレンス番号。これによりテーブル内の特定のエレメントを識別する。 可変数の引数を認める関数では、添え字付き ID を関数引数として使用する場合は、添え字に ALL を使用することができる。

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 つ以上の添え字から構成される ID。

置換文字 (substitution character)
ソース・コード・ページからターゲット・コード・ページへの変換の際に、ターゲット・コード・ページで定義されていない文字を表すのに使用される文字。

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。

サロゲート・ペア (surrogate pair)
UTF-16 形式のユニコードで、共に 1 つのユニコード図形文字を表すエンコード方式ペアの単位。 ペアの最初の単位は上位サロゲート と呼ばれ、第 2 の単位は下位サロゲート と 呼ばれる。 上位サロゲートのコード値の範囲は、X'D800' から X'DBFF' である。 下位サロゲートのコード値の範囲は、X'DC00' から X'DFFF' である。 サロゲート・ペアは、Unicode 16 ビット・コード化 文字セットに適合する文字を 65,536 文字を超えて提供する。

スイッチ状況条件 (switch-status condition)
オンまたはオフに設定可能な UPSI スイッチが、特定の状況に設定されているという命題で、これに関して真理値を判別することができる。

シンボリック文字 (* 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)
(1) 意味や解釈および使用の方法に依存しない、文字同士または文字のグループ同士の間の関係。 (2) 言語における表現の構造。 (3) 言語構造を支配する規則。 (4) 記号相互の関係。 (5) ステートメントの構築にかかわる規則。

システム名 (* system-name)
オペレーティング環境と連絡し合うために使用される COBOL ワード。

T

テーブル (* table)
DATA DIVISION の中で OCCURS 節によって定義される、論理的に連続するデータ項目の集合。

テーブル・エレメント (* table element)
テーブルを構成する繰り返し項目の集合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)
以下のいずれかの文字から成る COBOL 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マージン A およびマージン R の間の、1 文字または連続した文字のシーケンス。

- スペース以外の区切り記号、疑似テキスト区切り文字、英数字リテラルの開始と終了の区切り文字。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。英数字リテラルの場合には、そのリテラルの境界となる開始の引用符と終了の引用符を含むリテラル。
- コメント行および区切り記号によって囲まれたワード COPY を除く、その他の連続する一連の COBOL 文字で、区切り記号でもリテラルでもないもの。

スレッド (thread)
プロセスの制御下にあるコンピューター命令のストリーム (プロセス内のアプリケーションによって開始される)。

トークン (token)
COBOL エディターでは、プログラムにおける意味の単位。 トークンには、データ、言語キーワード、ID、またはその他の言語構文の一部を含めることができる。

トップダウン設計 (top-down design)
関連付けられた諸機能が、構造の各レベルで実行されるようにする階層構造を使ったコンピューター・プログラムの設計。

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・ユニットのデータ・レコードの後にある、データ・セットのラベル。 (2)「ファイル終わりラベル *(end-of-file label)*」の同義語。

トラブルシューティング (troubleshoot)
コンピューター・ソフトウェアの使用中に問題を検出し、突き止め、除去すること。

真の値 (* truth value)
2 つの値 (真または偽) のどちらか一方によって、 条件評価の結果を表したもの。

型式化オブジェクト・リファレンス (typed object reference)
指定されたクラスまたはそのサブクラスのオブジェクトだけを参照できるデータ名。

U

単項演算子 (* unary operator)
正符号 (+) または負符号 (-)。 算術式の変数や算術式の左括弧の前に置き、それぞれ +1 または -1 を式に乗算する。

無制限テーブル (unbounded table)
上限として 整数-*2* を指定するのではなく、`OCCURS 整数-1 to UNBOUNDED` によるテーブル。

Unicode
現代世界の各国の言語で記述されるテキストの交換、処理、表示をサポートする汎用文字エンコード標準。UTF-8、UTF-16、UTF-32 など、Unicode を表現する複数のエンコード・スキームが ある。Enterprise COBOL では、国別データ・タイプの表記としてビッグ・エンディアン・フォーマットの UTF-16 を使用して Unicode をサポートしている。

URI (Uniform Resource Identifier (URI))
リソースを一意に指す文字のシーケンスのことで、Enterprise COBOL では、名前空間の ID。URI 構文は、文書「*Uniform Resource Identifier (URI): Generic Syntax*」で定義されています。.

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 (universal object reference)
どのクラスのオブジェクトでも参照できるデータ名。

非制限ストレージ (unrestricted storage)
2 GB バーより下のストレージ。 16 MB 境界より上または下がある。 16 MB 境界より上では、31 ビット・モードでのみ、アドレス可能。

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。 あるステートメントの実行不成功は、そのステートメントによって参照されるデータには影響を及ぼさないが、状況表示には影響を与える可能性がある。

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 から UPSI-7 の 8 つのスイッチがある。

URI
URI を参照。

ユーザー定義語 (* user-defined word)
節やステートメントの形式を満たすためにユーザーが提供する必要のある COBOL ワード。

V

変数 (* variable)
オブジェクト・プログラムの実行によって変更を受ける可能性のある値を持つデータ項目。 算術式で使われる変数は、数字基本項目でなければならない。

可変長項目 (variable-length item)
OCCURS 節の DEPENDING 句で記述された表を含んだグループ項目。

可変長レコード (* variable-length record)
ファイル記述項目またはソート・マージ・ファイル記述項目が、 文字位置の数が可変であるレコードを許容しているファイルに関連付けられているレコード。

可変オカレンス・データ項目 (* variable-occurrence data item)
可変オカレンス・データ項目とは、反復される回数が可変であるテーブル・エレメントを言う。 そのような項目は、そのデータ記述記入項目内に OCCURS DEPENDING ON 節を持っているか、またはそのような項目に従属していなければならない。

可変位置グループ (* variably located group)
同じレコード内の可変長テーブルに続くグループ項目 (可変長テーブルに従属するわけではない)。 グループ項目は、英数字グループでも国別グループでも構いません。

可変位置項目 (* variably located item)
同じレコード内の可変長テーブルに続くデータ項目 (可変長テーブルに従属するわけではない)。

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とられる処置を表すワード。

ボリューム (volume)
外部ストレージのモジュール。 テープ装置の場合はリール、直接アクセス装置の場合はユニット。

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、自動的に実行されるシステム固有の処理手順。

VSAM ファイル・システム (VSAM file system)
COBOL の順次編成、相対編成、および索引編成をサポートするファイル・システム。

W

Web サービス (web service)
特定のタスクを実行し、HTTP や SOAP といったオープン・プロトコルを介してアクセス可能なモジュラー・アプリケーション。

空白文字 (white space)
文書にスペースを挿入する文字。 空白文字には以下のものがある。

- スペース
- 水平タブ
- 復帰
- 改行
- 次の行

Unicode 標準では上記のように呼ばれる。

ワード (* word)
ユーザー定義語、システム名、予約語、または関数名を形成する、30 文字を超えない文字ストリング。

*** WORKING-STORAGE SECTION**
独立項目または WORKING-STORAGE レコード、あるいはその両方から構成される、WORKING-STORAGE データ項目を記述する DATA DIVISION のセクション。

ワークステーション (workstation)
コンピューターの総称 (パーソナル・コンピューター、3270 端末、インテリジェント・ワークステーション、および UNIX 端末を含む)。ワークステーションはメインフレームまたはネットワークに接続されることがよくある。

ラッパー (wrapper)
オブジェクト指向コードとプロシージャー指向コード間のインターフェースを提供するオブジェクト。 ラッパーを使用すると、他のシステムがプログラムを再利用したり、プログラムにアクセスしたりできるようになる。

X

x
PICTURE 節内の記号であり、コンピューターの有する文字セットの任意の文字を含めることができる。

XML
Extensible Markup Language。 マークアップ言語を定義するための標準メタ言語。SGML から派生した、SGML のサブセットである。 XML では、SGML の複雑で使用頻度の低い部分が省略され、文書タイプを扱うアプリケーションの作成、構造化情報の作成および管理、異種コンピューター・システム間での 構造化情報の伝送および共用がはるかに容易になっている。 XML を使用するときに、SGML で必要とされるような堅固なアプリケーションや 処理は不要である。 XML は、World Wide Web Consortium (W3C) の主導で開発された。

XML データ (XML data)
XML エレメントを持つ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・タイプ宣言で定義される。

XML 宣言 (XML declaration)
使用している XML のバージョンや文書のエンコードなど、XML 文書の特性を指定する XML テキスト。

XML 文書 (XML document)
W3C XML 規格で定義されているとおり正しい形式のデータ・オブジェクト。

XML ネーム・スペース (XML namespace)
W3C XML ネーム・スペース仕様によって定義されたメカニズムで、エレメント名および属性名の集まりの有効範囲を制限する。 一意的に選択された XML 名前空間によって、複数の XML 文書または XML 文書内の複数のコンテキストでエレメント名または属性名が一意的に識別されます。

XML スキーマ (XML schema)
W3C によって定義されたメカニズムで、XML 文書の構造と内容を記述し、制約する。 XML スキーマは、それ自体が XML で表され、特定タイプ (購入注文など) の XML 文書のクラスを効率的に定義する。

Z

z/OS UNIX ファイル・システム (z/OS UNIX file system)
階層構造で編成されたファイルとディレクトリーの集合であり、z/OS UNIX を使用してアクセスできる。

ゾーン 10 進数データ項目 (zoned decimal data item)
暗黙的または明示的に USAGE DISPLAY として記述されており、PICTURE の記号 9、S 、P、および V の有効な組み合わせを含んでいる、外部 10 進数データ項目。 ゾーン 10 進数データ項目の内容は、文字 0 から 9 で表され、必要に応じて符号が付きます。 PICTURE ストリングが符号を指定しており、SIGN IS SEPARATE 節が指定されている場合、符号は 文字 + または - として表されます。 SIGN IS SEPARATE が指定されていない場合、符号は、符号位置の最初の 4 ビットをオーバーレイする 1 つの 16 進数字です (先行または末尾)。

#

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*

2014 COBOL 標準 (2014 COBOL Standard)
以下の標準によって定義された COBOL 言語。

- *INCITS/ISO/IEC 1989:2014, Information technology - Programming languages, their environments and system software interfaces - Programming language COBOL*

リソース・リスト

Enterprise COBOL for z/OS

COBOL for z/OS の資料

以下の資料が「Enterprise COBOL for z/OS ライブラリー」にあります。

- *What's new*
- カスタマイズ・ガイド (SC43-3366-02)
- 言語解説書 (SC43-3367-02)
- プログラミング・ガイド (SC43-3368-02)
- 移行ガイド (GC43-3369-02)
- パフォーマンス・チューニング・ガイド (SC43-4104-01)
- メッセージおよびコード (SC43-4107-01)
- *Program Directory* (GI13-4526-02)
- *Licensed Program Specifications* (GI13-4532-02)

ソフトコピー資料

次のコレクション・キットには、Enterprise COBOL およびその他の製品資料が含まれます。 これらは http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss にあります。

- *z/OS Software Products Collection*
- *z/OS and Software Products DVD Collection*

サポート

Enterprise COBOL for z/OS のご使用の際に問題がある場合は、サイト: https://www.ibm.com/support/home/product/B984385H82239E03/Enterprise_COBOL_for_z/OS を参照してください。そこでは最新のサポート情報が提供されています。

関連資料

z/OS ライブラリー資料

以下の資料が「z/OS ライブラリー」にあります。

ランタイム・ライブラリー拡張機能

- 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
- *DWARF/ELF* エクステンション ライブラリー・リファレンス

z/Architecture

- *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

- アプリケーション・プログラミング・ガイド
- 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 *1* 巻
- ユーザーズ・ガイド 第 *2* 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
- *Language Environment Vendor Interfaces*
- 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 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
- プログラミング*:* 高水準言語向け呼び出し可能サービス
- 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* ユーザーズ・ガイドおよび解説書

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 ライブラリー」にあります。

- *CICS* アプリケーションの開発
- *API (EXEC CICS)* リファレンス
- *CICS* システム・プログラムの開発
- グローバル・ユーザー出口リファレンス
- *XPI Reference*
- *CICS* での *EXCI* の使用

COBOL 報告書作成プログラム・プリコンパイラー

- *Programmer's Manual*、SC26-4301
- *Installation and Operation*、SC26-4302

Db2 for z/OS

以下の資料が「Db2 ライブラリー」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

IBM Debug for z/OS (以前の IBM Debug for z Systems および IBM Debug Tool for z/OS)

IBM Debug for z/OS については、IBM Debug for z/OS ライブラリーを参照してください。

IBM Debug for z Systems および IBM Debug Tool for z/OS は、IBM Debug for z/OS に置き換えられています。COBOL 文書ライブラリーで、IBM Debug for z Systems および IBM Debug Tool for z/OS の参照がすべて変更されているわけではありません。 デバッガーを最新レベルにアップグレードして、デバッグ機能の全範囲を使用できるようにすることをお勧めします。場合によっては、COBOL アプリケーションの作成に使用している Enterprise COBOL のレベルに応じて、デバッガーを特定のバージョンにアップグレードする必要があります。

- IBM Debug Tool V13.1 は、Enterprise COBOL V5.1 以前のバージョンをサポートしています。
- IBM Debug for z Systems V14.0 は、Enterprise COBOL V6.1 以前のバージョンをサポートしています。
- IBM Debug for z Systems V14.1 は、Enterprise COBOL V6.2 以前のバージョンをサポートしています。
- IBM Debug for z/OS V14.2 は、Enterprise COBOL V6.3 以前のバージョンをサポートしています。

お客様のニーズに最も適している IBM デバッグ製品を見つけるには、https://www.ibm.com/support/knowledgecenter/SSQ2R2_14.2.0/com.ibm.debug.cg.doc/common/dcompo.html?sc=SSQ2R2_latest を参照してください。

IBM Developer for z/OS (以前の IBM Developer for z Systems)

IBM Developer for z Systems に関する情報は、IBM Developer for z/OS ライブラリーにあります。

注 : IBM Developer for z Systems および Rational Developer for z Systems は IBM Developer for z/OS に置き換えられています。

以下の資料が「IBM Publications Center」にあり、資料番号で検索できます。

IMS

- *Application Programming API Reference*、SC18-9699
- *Application Programming Guide*、SC18-9698

WebSphere® Application Server for z/OS

- *Applications* , SA22-7959

Softcopy publications for z/OS

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
- *The Java 2 Enterprise Edition Developer's Guide*、download.oracle.com/javaee/1.2.1/devguide/html/DevGuideTOC.html
- *Java 2 on z/OS*、www.ibm.com/servers/eserver/zseries/software/java/
- *The Java EE 5 Tutorial*、download.oracle.com/javaee/5/tutorial/doc/
- *The Java Language Specification, Third Edition* (Gosling 他著)、java.sun.com/docs/books/jls/
- *The Java Native Interface*、download.oracle.com/javase/1.5.0/docs/guide/jni/
- *JDK 5.0 Documentation*、download.oracle.com/javase/1.5.0/docs/

JSON

- JavaScript Object Notation (JSON)、www.json.org

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]

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

プログラム番号: 5655-EC6

GC43-3369-02