システムz最新化に役立つ移行情報セミナー　2015

Enterprise COBOL V5.2最新情報

日本アイ･ビー･エム株式会社
沖野 英紀

目次

- Enterprise COBOL V5のご紹介
- Enterprise COBOL V5の新機能
  - Enterprise COBOL V5.1で提供された新機能
  - Enterprise COBOL V5.2の新機能
- Enterprise COBOL V5への移行
  - 移行の主な考慮点
  - 移行時の制約の緩和

Enterprise COBOL V5のご紹介

PGM No	製品名	VRM	前提OS	発表	出荷開始	サポート終了
5655-W32	Enterprise COBOL for z/OS	V5R2	Z/OS V1.13～	2015/01/13	2015/02/27 予定	-
		V5R1	z/OS V1.13～	2013/04/23	2013/06/21	-
5655-S71	Enterprise COBOL for z/OS	V4R2M0	z/OS V1.9～	2009/08/26	2009/08/28	-
		V4R1M0	z/OS V1.7～	2007/12/12	2007/12/14	2014/04/30
5655-G53	Enterprise COBOL for z/OS	V3R4M0	z/OS V1.4～	2005/06/21	2005/07/01	2015/04/30
		V3R3M0	OS/390 V2.10 、z/OS V1.1～	2004/02/17	2004/02/27	2007/04/30
	Enterprise COBOL for z/OS and OS/390	V3R2M0		2002/08/20	2002/09/27	2005/10/03
		V3R1M0		2001/11/27	2001/12/28	2004/04/04

- V5R2は、2013年6月に出荷されたIBMの画期的なCOBOLコンパイラーの2ndリリース
- V5では高度な最適化技術を導入して、V4までのコンパイラーを刷新

Enterprise COBOL V5の新機能

Enterprise COBOL V5.1で提供された新機能サマリー

新機能	機能概要
稼動ハードウェアに最適なコードの生成	• ARCHオプションによりコンパイラー生成命令の制御 • HGPR、AFPオプションの追加
最適化機能の強化	• CやJavaで実績のある最適化技術を活用 • OPT(0|1|2)オプションによる最適化制御 • STGOPTオプションの追加
デバッグ機能の強化	• 業界標準DWARFをベースにしたデバッグ情報の出力 • 実行時にデバッグ情報をメモリーに展開しないNOLOAD属性 • TESTオプションの変更
新しいCOBOL言語機能	• インラインコメント（浮動コメント）のサポート • WORKING-STORAGEサイズ制限を2GBに拡張 • データ項目のサイズ制限を999,999,999バイトに拡張 • UNBOOUNDEDテーブルのサポート
Open系対応の新しいCOBOL言語機能	• UTF-8データ処理のための組み込み関数の拡張 • XML GENERATE命令の拡張 （NAME OF句、TYPE OF句、SUPPRESS句） • XML PARSE命令の拡張 （XML-INFORMATION特殊レジスター）
使いやすさ向上のための新規オプション	• DISPSIGNオプション、LVLINFO導入時オプション

Enterprise COBOL V5.2の新機能サマリー

新機能	機能概要
新しいハードウェアへの対応：z13	• SIMD命令の利用 • DFP(10進浮動小数点）命令利用の強化
新規コンパイルオプション	• プログラム開発支援とアプリケーション管理の強化 • 移行支援サポート強化
2002 COBOL言語規格への対応強化	• テーブルSORT命令 • EXIT命令の機能強化（EXIT PERFORM命令など） • COPY REPLACINGおよびREPLACE命令の機能強化
新しいCOBOL言語機能	• ユーザー条件処理を使用するアプリケーションのOPTIMIZE • >>CALLINTERFACEコンパイラー指示 • XML GENERATE命令SUPPRESS指定の強化
JSON serviceへのアクセスサポートの発表	• COBOLからz/OS JSON serviceの利用

新しいハードウェアへの対応:z13

- 初めて、新しいハードウェア発表と同時にCOBOLコンパイラーを発表
- SIMD (ベクトル処理) 命令をINSPECT TALLYING or REPLACING命令の一部で利用
  - 1命令で複数データを処理 (SIMD)するベクトル演算命令
- パック10進数データに対してさらにDFP (10進浮動小数点)命令を使用
- OPT(2)とARCH(11)の組み合わせにより、新しいマイクロアーキテクチャー対応の新しい命令スケジューラーを使用

z13: SIMD (ベクトル処理) 命令の利用例

- INSPECT TALLYING命令とINSPECT REPLACING命令

サンプルコード:

```
WORKING-STORAGE SECTION.
 01 VARS.
    02 STR PIC X(255).
    02 C PIC 9(5) COMP-5 VALUE 0.
PROCEDURE DIVISION.
    MOVE ALL 'abc def ghi jkl ' TO STR
    PERFORM 100000000 TIMES
      INSPECT STR TALLYING C FOR ALL ' '
    END-PERFORM

    GOBACK
```

z13: SIMD (ベクトル処理)命令の利用例の計測結果

V5.1
- ARCH(10)

```
         LHI     R0,0xff
         XR      R1,R1
         LA      R12,152(,R8)     #  STR
L0064:   EQU     *
         CLI     0(,R12),X'40'    #
         JNOP    L0066
         LA      R1,1(,R1)        #
L0066:   EQU     *
         LA      R12,1(,R12)      #
         BRCT    R0,L0064
         A       R1,407(,R8)      #  C
         ST      R1,407(,R8)      #  C
```

計測結果 (100 million times in a loop)

```
V5.1 :  46.63 cpu seconds
V5.2 :   1.54 cpu seconds
```

V5.2 で30倍の速さ
97%のCPUタイムの削減

V5.2
- *ARCH(11)*

```
         LHI     R0,0xfe
         XR      R1,R1
         LA      R12,152(,R8)           #  STR
         VREPIB  VRF27,0x40
         VGBM    VRF25,0x0
L0066:   EQU     *
         VLL     VRF24,R0,0(,R12)       #
         AHI     R12,0x10
         VCEQB   VRF24,VRF24,VRF27
         AHI     R0,0xfff0
         VLCB    VRF24,VRF24
         VAB     VRF25,VRF25,VRF24
         JNL     L0066
         VGBM    VRF26,0x0
         VSUMB   VRF25,VRF25,VRF26
         VSUMQF  VRF25,VRF25,VRF26
         VLGVG   R1,VRF25,1(,R1)        #
         A       R1,407(,R8)            #  C
         ST      R1,407(,R8)            #  C
```

z13: DFP (10進浮動小数点)命令の利用例

- パック10進 (COMP-3) データ項目

サンプルコード:

```
WORKING-STORAGE SECTION.
      01 VARS.
        02 A PIC S9(25) Packed-Decimal VALUE +12345678901234567890123456789.
        02 B PIC S9(25) Packed-Decimal VALUE +24680975312468097531246809753124680.
        02 C PIC S9(25) Packed-Decimal VALUE 0.
      PROCEDURE DIVISION.
          PERFORM 100000000 TIMES
            DIVIDE A BY B GIVING C
          END-PERFORM
```

z13: DFP (10進浮動小数点)命令の利用例の計測結果

V5.1
- ARCH(10)

```
XGR     R0,R0
ICMH    R0,X'1',152(,R8)        #  A
L       R0,153(,R8)             #  A
LG      R1,157(,R8)             #  A
CXSTR   FP0,R0
XGR     R0,R0
ICMH    R0,X'1',165(,R8)        #  B
L       R0,166(,R8)             #  B
LG      R1,170(,R8)             #  B
CXSTR   FP1,R0
DXTR    FP4:FP6,FP0:FP2,FP1:FP3
FIXTR   FP0:FP2,9,FP4:FP6
CSXTR   R0:R1,0,FP0:FP2
STCMH   R0,X'1',178(,R8)        #  C
ST      R0,179(,R8)             #  C
STG     R1,183(,R8)             #  C
ZAP     178(13,R8),178(13,R8)   #  C
AHI     R2,0xffff
CIJ     R2,L0034,0,HT(mask=0x2),
```

V5.2
- *ARCH(11)*

```
CXPT    FP0:FP2,152(13,R8),0x8
CXPT    FP1:FP3,165(13,R8),0x8
DXTR    FP4:FP6,FP0:FP2,FP1:FP3
FIXTR   FP0:FP2,9,FP4:FP6
CPXT    FP0:FP2,178(13,R8),0x9
AHI     R2,0xffff
CIJ     R2,L0034,0,HT(mask=0x2),
```

計測結果 (100 million times in a loop)

```
V5.1 :  2.53 cpu seconds
V5.2 :  1.63 cpu seconds
```

V5.2で36% のCPUタイムの削減

参考）オペランドがパック10進のHW制約を越えるDecimal Divideの例

```
1 z14v2 pic s9(14)v9(2).
1 z13v2 pic s9(13)v9(2).
...
Compute z14v2 = z14v2 / z13v2
```

V4

- *ライブラリールーチンを呼び出し*
- *パスレングスは100命令以上*

```
PACK   344(9,13),0(16,2)
PACK   360(16,13),16(15,2)
MVC    376(32,13),59(10)
MVC    398(9,13),344(13)
NI     406(13),X'F0'
MVN    407(1,13),352(13)
L      3,92(0,9)
L      15,180(0,3)
LA     1,146(0,10)
BASR   14,15
NI     431(13),X'0F'
ZAP    431(9,13),431(9,13)
UNPK   0(16,2),431(9,13)
```

V5

- *6命令をインラインに展開*
- *CDZT/CZDTは外部10進とDFP間の変換のためのEC12の新しい命令*
- *ARCH (10) で使用可能*

```
CDZT     FP0,152(16,R8),0x8
CDZT     FP1,168(15,R8),0x8
SLDT     FP0,FP2,2
DDTR     FP0,FP0,FP1
FIDTR    FP0,9,FP0
CZDT     FP0,152(16,R8),0x9
```

計測結果 (100 million in a loop)

```
V5 : 1.08 cpu seconds
V4 : 4.81 cpu seconds
```

V5で78%パフォーマンス改善

Enterprise COBOL V5.2の新機能サマリー

新機能	機能概要
新しいハードウェアへの対応：z13	• SIMD命令の利用 • DFP(10進浮動小数点）命令利用の強化
新規コンパイルオプション	• プログラム開発支援とアプリケーション管理の強化 • 移行支援サポート強化
2002 COBOL言語規格への対応強化	• テーブルSORT命令 • EXIT命令の機能強化（EXIT PERFORM命令など） • COPY REPLACINGおよびREPLACE命令の機能強化
新しいCOBOL言語機能	• ユーザー条件処理を使用するアプリケーションのOPTIMIZE • >>CALLINTERFACEコンパイラー指示 • XML GENERATE命令SUPPRESS指定の強化
JSON serviceへのアクセスサポートの発表	• COBOLからz/OS JSON serviceの利用

Enterprise COBOL V5のコンパイルオプションサマリー

- 以下は新機能に伴い追加・変更・廃止されたオプションを掲載
- 移行支援サポートのためのオプション追加については後述

オプション	V5.2	V5.1	機能概要
ARCH	変更	追加	• コンパイラーが生成するマシンコードのアーキテクチャレベルを指定
SIZE	廃止	変更	• V5.1でMAX指定が廃止されV5.2でコンパイラーによるメモリ管理の改善によりSIZEオプション廃止
QUALIFY	追加		• あいまなデータ項目の修飾が可能
RULES	追加		• 品質の高いコードの開発を支援するためのコーディングルール設定
COPYRIGHT	追加		• アプリケーション管理向上のためオブジェクトに任意の文字列設定
SERVICE	追加		• アプリケーション管理向上のためオブジェクトに任意の文字列設定
OPTIMIZE		変更	• 最適化機能刷新に伴い最適化のレベル指定が0/1/2に変更
STGOPT		追加	• 已参照データの削除を制御
TEST		変更	• デバッグ情報提供機能の刷新に伴いサブオプションの変更・追加
AFP		追加	• 追加浮動小数点レジスター使用時のレジスター内値の保護
HGPR		追加	• 64ビットレジスターの上位32ビット使用時のレジスター内値の保護
SQLIMS		追加	• IMS SQL coprocessorの使用を制御

ARCHコンパイラーオプション

- V5.2ではARCH(6)が廃止され新たにARCH(11)が追加
- 低いレベルのHWに対して、より高いレベルのARCHのオプションを指定しないように、コンパイルしたプログラムをどのHWで稼動させるかを意識する必要がある。

オプション	V5.2	V5.1	前提ハードウェア
ARCH(6)	廃止	○ (Default)	• 2084-xxx models (z990) • 2086-xxx models (z890)
ARCH(7)	○ (Default)	○	• 2094-xxx models (IBM System z9 EC) • 2096-xxx models (IBM System z9® BC)
ARCH(8)	○	○	• 2097-xxx models (IBM System z10 EC) • 2098-xxx models (IBM System z10 BC)
ARCH(9)	○	○	• 2817-xxx models (IBM zEnterprise z196 EC) • 2818-xxx models (IBM zEnterprise z114 BC)
ARCH(10)	○	○	• 2827-xxx models (IBM zEnterprise EC12) • 2828-xxx models (IBM zEnterprise BC12)
ARCH(11)	○	—	• 2964-xxx models (IBM z13)

QUALIFYコンパイラーオプション

- **QUALIFY(COMPAT)**
  - QUALIFY(COMPAT)指定時にはデータ項目アクセス時にはユニークな指定にする必要がある
- **QUALIFY(EXTEND)**
  - QUALIFY(EXTEND)指定時には、データ項目の修飾ルールが拡張され、COBOL言語規格上はユニークと判断されなかった指定がユニークと判断されるように拡張される
  - データ項目の階層構造の全てのレベルを指定して修飾する場合、「完全修飾された」データ項目と呼ぶ
  - 階層構造の上位1つのみを指定して修飾する場合、他の階層構造でも同様の不完全な修飾としてありうる場合でも、該当データ項目へのアクセスとして解釈される

QUALIFYコンパイラーオプション使用例

- QUALIFY(EXTEND)の指定によりあいまいなデータ項目の修飾が可能

```
01 A.
   02 B.
      03 C PIC X.        *> C of A
      03 A PIC X.
   02 C PIC X.           *> Also C of A

Move Z to C of A         *> Refers to 02 level C (unique only with QUA(EXTEND))
Move Z to A              *> Refers to 01 level A (unique only with QUA(EXTEND))
Move Z to C of B of A *> Refers to 03 level C (unique by COBOL std rules)
Move Z to C of B         *> Refers to 03 level C (unique by COBOL std rules)
```

RULESコンパイラーオプション

- PL/I RULESオプションと同様の機能
- 「適切な」コーディングをルールとして指定することが可能
- 多くのお客様からのご要望により機能を追加
- デフォルト値は、RULES(ENDPERIOD,EVENPACK,LAXPERF,SLACKBYTES)

RULESコンパイラーオプションのサブオプション

- NOENDPERIOD (NOENDP):
  - 明示的範囲終了符号（END-*）の代わりにピリオドで終了する条件文に対してメッセージを出力
- NOEVENPACK (NOEVENP):
  - 偶数桁で定義されたパック10進データ項目に対してメッセージを出力
- NOLAXPERF (NOLXPRF)
  - パフォーマンス向上の余地のあるCOBOLの機能に対してメッセージを出力
- NOSLACKBYTES (NOSLCKB)
  - SYNCHRONIZED 指定により遊びバイトが確保されるデータ項目に対してメッセージを出力

RULES(NOLAXPERF)でチェックされるケース

- 以下のようなケースでコンパイルメッセージが出力される
  - 効率のよくないループの場合

    `Perform varying ix1 from 1 by 1 until ix1 > ix1-max`
    - ix1が（パック10進やバイナリーではなく）ゾーン10進として定義されている場合
    - VARYING句にある複数のオペランドがそれぞれ異なるデータ属性の場合
  - バイナリーやパック10進定義でない添字を使用してテーブルにアクセスする場合
  - MOVE命令 (COMPUTE命令, 比較命令)で異なる属性のために数値データの型変換が発生する場合
  - 文字列を別のデータ項目にMOVEする際に（100バイト以上）のパディングが発生する場合
    - 例えば、FROM-FIELD PIC X(10)とTO-FIELD PIC X(32000)のデータ項目が定義されていて、
    - MOVE FROM-FIELD TO TO-FIELDを実行すると、10バイトがMOVEされた残りの32KBのメモリーにはスペースがパディングされる
  - 実行が遅くなるか最適でないコンパイラーオプションが指定されている場合
    - NOAWO, NOBLOCK0, NOFASTSRT, NUMPROC(NOPFD), OPT(0), SSRANGE, TRUNC(STD|BIN), ZONEDEC(MIG)

COPYRIGHT/SERVICEコンパイラーオプション

- COPYRIGHT(‘文字列’) コンパイラーオプション
  - オブジェクトコード内に指定した文字列を埋め込む
  - COBOL実行モジュールを出荷するベンダーの使用を想定
- SERVICE(‘文字列’) コンパイラーオプション
  - オブジェクトコード内に指定した文字列を埋め込む
  - ユーザーのアプリケーション管理での使用を想定
  - 今までは障害解析などでオブジェクトコード内を見ると、コンパイル日付、コンパイラーのバージョン・リリース・モディフィケーションは確認できたが、サービスレベルの情報は確認できなかった
  - 今後は、コンパイル時にサービスレベルを識別する文字列を設定することが可能

Enterprise COBOL V5.2の新機能サマリー

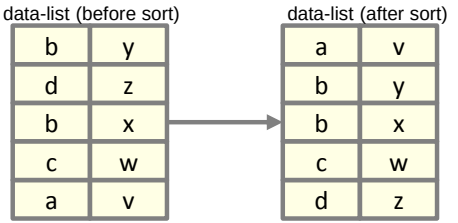
新機能	機能概要
新しいハードウェアへの対応：z13	• SIMD命令の利用 • DFP(10進浮動小数点）命令利用の強化
新規コンパイルオプション	• プログラム開発支援とアプリケーション管理の強化 • 移行支援サポート強化
2002 COBOL言語規格への対応強化	• テーブルSORT命令 • EXIT命令の機能強化（EXIT PERFORM命令など） • COPY REPLACINGおよびREPLACE命令の機能強化
新しいCOBOL言語機能	• ユーザー条件処理を使用するアプリケーションのOPTIMIZE • >>CALLINTERFACEコンパイラー指示 • XML GENERATE命令SUPPRESS指定の強化
JSON serviceへのアクセスサポートの発表	• COBOLからz/OS JSON serviceの利用

テーブルSORT命令(形式2のSORT命令)

- テーブルSORT命令でユーザー指定の順序にSORT
- 指定方法:
  SORT *data-name-2* [ON { ASCENDING | DESCENDING } KEY [*data-name-1*]...
  ]...[WITH DUPLICATES IN ORDER] [COLLATING SEQUENCE IS *alphabet-name-1*]

```
01 mydata.
   03 data-list occurs 5 times.
      05 data-key pic x.
      05 data-nonkey pic x.
```

```
SORT data-list ON ASCENDING KEY data-key
     WITH DUPLICATES IN ORDER
     COLLATING SEQUENCE Standard-1
```



EXIT命令の機能強化

- EXIT [SECTION | PARAGRAPH]
  - EXIT SECTION：現在のSECTIONからぬける
  - EXIT PARAGRAPH：現在の段落からぬける
- EXIT PERFORM [CYCLE]
  - EXIT PERFORM：インラインのPERFORMからぬける
    - PL/Iの`LEAVE`命令や`C`の`break`と同様の機能
  - EXIT PERFORM CYCLE：インラインのPERFORMの現在の繰り返しの残りの処理をスキップ
    - PL/Iの`ITERATE`命令や`C`の`continue`と同様の機能

EXIT命令の機能強化の例

```
working-storage section.
01 n pic 99.
procedure division.
s1 section.
p1.
  display 'In p1'
  if n < 5 then
    exit paragraph  *> proceed to p2
  else
    if n < 10 then
      exit section  *> proceed to s2
    end-if
  end-if
  display 'After if-statement in p1'.
p2.
  display 'In p2'.
```

```
s2 section.
  perform 10 times
    if n = 5 then
      exit perform *> exit perform loop
    end-if
    display 'perform1 n=' n
  End-perform
  *> exit perform passes control here
  display 'N=' n
  perform varying n from 1 by 1 until n = 10
    if n = 5 then
      exit perform cycle *> next iteration
    end-if
    display 'perform2 n=' n
    *> exit perform cycle passes control here
  End-perform
  goback.
```

COPY REPLACINGおよびREPLACE命令の機能強化

- **COPY REPLACING命令の機能拡張**
  - LEADINGとTRAILING句（ワードの一部を変換）
    - REPLACE命令やCOPY命令のLEADING/TRAILING句でソース内やライブラリー内のテキストに含まれるワードの一部を対象にした変換が可能。
      この機能は、データ項目に対する接頭語、接尾語の設定に有効
  - ネストしたCOPY REPLACING命令のサポート（IBM拡張機能）
    - 今までのルール:
      COPY命令はネストさせることが可能だが、ネストしたCOPY命令ではREPLACING句を指定できない。また、COPY REPLACING命令についてはCOPY命令をネストさせることはできない
    - ネストのチェーンの中に1つのCOPY REPLACING命令が指定可能となった
      Pgm AにCOPY Bがあり、Bの中にCOPY Cがあり、Cの中にCOPY D REPLACINGを指定

- **REPLACE命令の機能強化**
  - LEADINGとTRAILING句（ワードの一部を変換）

COPY REPLACING LEADING/TRAILING機能の例

- COPY…REPLACING LEADING/TRAILINGの指定方法:
  COPY <copy-file-name> [...] REPLACING **{LEADING | TRAILING}**
  == *partial-word-1*== BY == *partial-word-2* == … .

```
COPY PAYLIB REPLACING LEADING == DEPT == BY == PAYROLL ==.
```

PAYLIB COPY展開前

```
01 DEPT.
   02 DEPT-WEEK       PIC S99.
   02 DEPT-GROSS-PAY  PIC S9(5)V99.
   02 DEPT-HOURS      PIC S999
      OCCURS 1 TO 52 TIMES
      DEPENDING ON DEPT-WEEK OF
      DEPT.
```

PAYLIBをソースに展開して変換した後

```
01 PAYROLL.
   02 PAYROLL-WEEK       PIC S99.
   02 PAYROLL-GROSS-PAY  PIC S9(5)V99.
   02 PAYROLL-HOURS      PIC S999
      OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF
      PAYROLL.
```

ネストしたCOPY REPLACING命令の例

- 以下はネストしたCOPY REPLACINGの処理の例。以下のプログラムではPAYLIBをincludeしており、PAYLIBがさらにPLAYLIB2をincludeしている。右側が変換が実施された後のイメージ

```
  COPY PAYLIB REPLACING
       LEADING  == DEPT == BY == PAYROLL ==
       TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

PAYLIB

```
01 PAYROLL.
   02 PAYROLL-WEEK PIC S99.
   02 DEPT-GROSS-PAY PIC S9(5)V99.
   02 PAYROLL-HOURS PIC S999 OCCURS 1 TO 52 TIMES
            DEPENDING ON PAYROLL-WEEK OF PAYROLL.
COPY PAYLIB2.
```

PAYLIB2

```
01 PAYROLL2.
   02 PAYROLL2-WEEK PIC S99.
   02 DEPT2-GROSS-PAY PIC S9(5)V99.
   02 PAYROLL2-HOURS PIC S999
           OCCURS 1 TO 52 TIMES
           DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

COPY REPLACING後のソース

```
01 PAYROLL.
   02 PAYROLL-WEEK PIC S99.
   02 PAYROLL-NET-PAY PIC S9(5)V99.
   02 PAYROLL-HOURS PIC S999 OCCURS 1 TO 52 TIMES
           DEPENDING ON PAYROLL-WEEK OF PAYROLL.

01 PAYROLL2.
   02 PAYROLL2-WEEK PIC S99.
   02 PAYROLL2-NET-PAY PIC S9(5)V99.
   02 PAYROLL2-HOURS PIC S999 OCCURS 1 TO 52 TIMES
           DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

Enterprise COBOL V5.2の新機能サマリー

新機能	機能概要
新しいハードウェアへの対応：z13	• SIMD命令の利用 • DFP(10進浮動小数点）命令利用の強化
新規コンパイルオプション	• プログラム開発支援とアプリケーション管理の強化 • 移行支援サポート強化
2002 COBOL言語規格への対応強化	• テーブルSORT命令 • EXIT命令の機能強化（EXIT PERFORM命令など） • COPY REPLACINGおよびREPLACE命令の機能強化
新しいCOBOL言語機能	• ユーザー条件処理を使用するアプリケーションのOPTIMIZE • >>CALLINTERFACEコンパイラー指示 • XML GENERATE命令SUPPRESS指定の強化
JSON serviceへのアクセスサポートの発表	• COBOLからz/OS JSON serviceの利用

ユーザー条件処理を使用するアプリケーションのOPTIMIZE

- 新しく追加されたVOLATILE属性とSERVICE LABELの機能により
  - ユーザー条件処理ルーチンを使用するアプリケーションをOPT(1|2)でコンパイルすることが可能
  - VOLATILEはさらに、STGOPTオプションとの組み合わせでも使用可能
    - 最適化をさせたくない未使用のデータ項目にVOLATILEを指定
      WORKING-STORAGE eye-catcher
      コード管理のためのタイムスタンプ項目　など

LE条件処理を使用したVOLATILEの例

```
identification division.                *> Main program (causes divide-by-zero exception):
program-id. main.
data division.
local-storage section.
77 user-handler procedure-pointer.
77 token    pic S9(9) comp.
01 qty      pic 9(8) binary.
01 divisor pic 9(8) binary value 0.
01 answer   pic 9(8) binary.
01 step     pic 9(8) binary value 0 external volatile.
:
Set user-handler to entry 'handler'
Call 'CEEHDLR' using user-handler, token, null
Compute step = 2                    *> オプティマイザーが「不要なコード」と判断し除去
Compute answer = number / divisor   *> ゼロ割エラーがここで発生して条件処理が起動され
                                    *> 「step」の項目を参照するがコンパイラーは気づかない
Display 'answer = ' answer

Compute step = 3
Display 'step = ' step
Compute answer = qty + 2
```

VOLATILE指定で解決

```
identification division.          *> Condition handler program:
program-id. handler.
data division.
local-storage.
01 step pic 9(8) external.
procedure division.
:

display 'Error: a problem was encountered in step ' step.
```

条件処理プログラムはメインで定義された「STEP」の項目を参照可能だが、「VOLATILE」指定がなければメインプログラムは参照されていることを判断できない。このため間違った値が表示される可能性がある

>>CALLINTERFACEコンパイラー指示

- >>CALLINTERFACE コンパイラー指示の機能
  - CALL命令ごとのにCALL種別(DLL, DYNAM or STATIC) を制御
  - 指定方法:　>>CALLINTERFACE [DLL | DYNAM | STATIC]
    （省略指定は　>>CALLINT　）
  - DLLおよびDYNAMコンパイラーオプションの設定を上書き
  - 次の>>CALLINTERFACEコンパイラー指示標識の指定まで有効。指定がなければプログラムの最後まで有効
- 利点:
  - DLLやDYNAMオプションの設定に関係なくさまざまなCALL方法を簡便に混在させられる
  - どのプログラムでもDLL、動的CALL、静的CALLを混在可能
  - >>CALLINTERFACE指定により、簡便にさまざまなCALL方法を制御可能

>>CALLINTERFACEコンパイラー指示の指定例

- 指定例1:

```
>>CALLINTERFACE DLL
CALL 'SUB1'   *> DLL call
CALL 'SUB2'   *> DLL call
>>CALLINTERFACE DYNAM
CALL 'SUB3'   *> dynamic call
>>CALLINTERFACE
CALL 'SUB4'   *> これ以降は、DLLか
              *> DYNAMオプションの
              *> 設定にしたがって
              *> CALL種別を決定
```

- 指定例2:

```
*> Program compiled with DYNAM
CALL 'SUB5'   *> Dynamic call
>>CALLINTERFACE DLL
CALL 'SUB6'   *> DLL call
>>CALLINTERFACE
CALL 'SUB7'   *> Dynamic call
*> この指定までは全て動的CALL
>>CALLINTERFACE Static
*> この指定の後は全て静的CALL
CALL 'SUB8'   *> Static call
```

XML GEMENRTE命令SUPPRESS指定の強化

- XML GENERATE命令でWHEN句を省略すると、XML生成時に該当データ項目に関する生成を無条件にSUPPRESSすることが可能
  - 該当データ項目がグループ項目であってもWHEN句を省略可能
- 「汎用抑止句」の指定可能なキーワードとしてCONTENTが追加され、TYPE IS CONTENTが指定された項目のみSUPPRESSが限定できるようになった

Enterprise COBOL V5.2の新機能サマリー

新機能	機能概要
新しいハードウェアへの対応：z13	• SIMD命令の利用 • DFP(10進浮動小数点）命令利用の強化
新規コンパイルオプション	• プログラム開発支援とアプリケーション管理の強化 • 移行支援サポート強化
2002 COBOL言語規格への対応強化	• テーブルSORT命令 • EXIT命令の機能強化（EXIT PERFORM命令など） • COPY REPLACINGおよびREPLACE命令の機能強化
新しいCOBOL言語機能	• ユーザー条件処理を使用するアプリケーションのOPTIMIZE • >>CALLINTERFACEコンパイラー指示 • XML GENERATE命令SUPPRESS指定の強化
JSON serviceへのアクセスサポートの発表	• COBOLからz/OS JSON serviceの利用

JSON serviceへのアクセスサポートの発表

- COBOLからz/OS JSON servicesへのアクセス
  - z/OS V2.1でz/OS Client Web Enablement Toolkitを使用するためにはAPAR OA46575のPTF適用が必要
  - まずは、COBOLでのコーディングサンプルを提供
    - 最新のマニュアルやTech Notes、SHAREプレセンテーション資料にて
  - 今後、COBOL言語要素の拡張として提供予定
    - XML PARSE命令やXML GENERATE命令と同様の言語要素として提供

COBOLからJSON service呼び出しの例

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Parses the sample JSON data.                                      *
*                                                                   *
* Services Used:                                                    *
*                                                                   *
* HWTJPARS: Builds an internal representation of the specified      *
*           JSON string. This allows efficient search, traversal,   *
*           and modification of the JSON data.                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
 parse-json-text.
      CALL 'HWTJPARS' USING HWTJ-RETURN-CODE HWTJ-PARSERHANDLE
          json-text-ptr   *> address of JSON text string (input)
          json-text-len   *> length of JSON text string (input)
          HWTJ-DIAG-AREA

     IF (HWTJ-OK)
       DISPLAY 'SUCCESS: JSON data parsed.'
     ELSE
       DISPLAY 'ERROR: Unable to parse JSON data.'
       CALL 'DISPDIAG' USING HWTJ-RETURN-CODE HWTJ-DIAG-AREA
     END-IF
      .
```

Enterprise COBOL V5への移行

移行の主な考慮点

- 以下はEnterprise COBOL V5以前のコンパイラーからV5へ移行する際の主な考慮点の一覧
- 移行支援のための緩和措置については後述

考慮点	概要	緩和措置
2000年対応関連機能（MLE）の廃止	• DATEPROC/YEARWINDOWコンパイルオプションの廃止により2桁年の処理機能の廃止	
ユーザーラベル処理機能の廃止	• DECLARATIVES段落のUSE AFTER LABEL指定の廃止	
旧COBOLプログラムとの相互運用性	• OS/VS COBOLとの連携廃止 • COB II NORESとの連携廃止	
AMODE24プログラムとの相互運用性	• AMODE24プログラムとの静的CALL廃止 • V5プログラムのAMODE31のみのサポート	有
XMLPARSEコンパイルオプションの廃止	• Enterprise COBOL V3のPARSERの廃止	有
NUMPROC(MIG)コンパイルオプションの廃止	• OS/VS COB移行支援サブオプションの廃止	有
可変長ファイルのREADの動作変更	• 実レコード長とプログラム内レコード長の不整合によるfile statusの変更	有
MAPコンパイルオプションの出力相違	• 出力オフセットが10進オフセットに変更	有

移行の主な考慮点

考慮点	概要	緩和措置
潜在的不具合に関する動作変更	• スペック外の動作に関する変更 （可変長テーブルの長さの不正、数値項目への不正データの設定、テーブル定義の範囲外へのアクセスなど）	
コンパイラー出力モジュールのPDSE化	• 出力モジュールはプログラムオブジェクトになるためPDSEへの格納が必須	
コンパイラー環境の変更	• コンパイラー使用リソースの増加 （CPU、メモリーなど）	
デバッグ情報出力の変更	• TESTオプションの変更 • デバッグ情報のモジュール内への取り込み	

- COBOL V5.1からV5.2への移行
  - V4.1から V4.2への移行と同様に容易
- V5以前のコンパイラーからCOBOL V5.2への移行
  - V4.1からV4.2への移行ほど容易ではない
  - V3やV4からV5.1に移行する場合と同様の負荷
  - 十分なリグレッションテストが必要

移行時の制約の緩和

- 制約の緩和のための以下の移行支援機能は、V5.1でPTFとして機能を追加し、V5.2のベースコードにも取り込まれている機能
- V5以前のコンパイラーからの移行を容易にするために改善された機能

考慮点	移行支援機能
AMODE24プログラムとの相互運用性	AMODE24サポート
XMLPARSEコンパイルオプションの廃止	XMLPARSE（COMPAT）コンパイルオプション
NUMPROC(MIG)コンパイルオプションの廃止	ZONEDATA（MIG）コンパイルオプション
可変長ファイルのREADの動作変更	VLR（COMPAT）コンパイルオプション
MAPコンパイルオプションの出力相違	MAP（HEX）コンパイルオプション

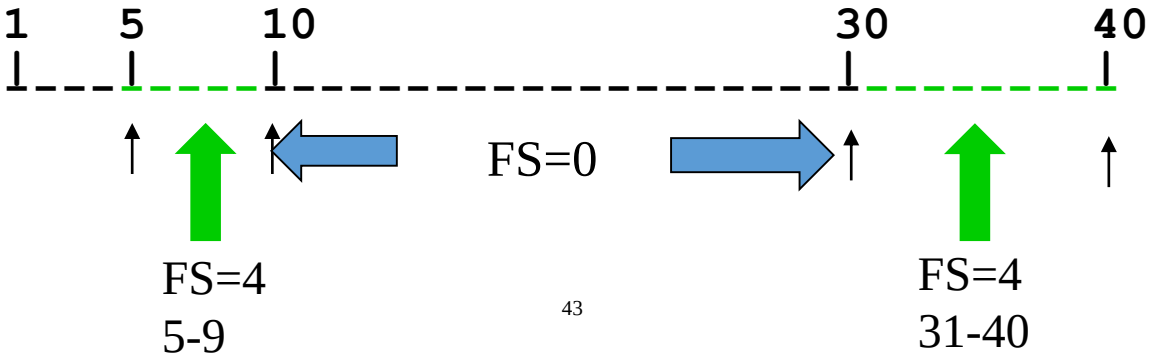
移行支援機能

- AMODE 24 サポート
  - AMODE 24での実行がV3やV4と同様にサポート可能
    - AMODE24のアセンブラープログラムを静的リンク可能
- XMLPARSE(COMPAT)
  - XML PARSE命令への修正なしでV5への移行が可能
- VLR(COMPAT)
  - より安全な言語規格準拠の動作か以前のコンパイラーとの互換性重視の動作かを選択可能
  - 言語規格準拠の動作では、不整合のある可変長レコードREADに対してFS(File Status) 04を返し、プログラムの品質改善に援助する目的でメッセージを追加していた

VLRオプション：不整合のある可変長ファイルREADの例

- COBOL言語規格準拠（VLR（STANDARD）指定時）での可変長ファイルのREADに対するFS（File Status）の設定

```
FD Record Contains 5 To 40 Depending on X.
01 Rec1 PIC X(10).
01 Rec2 PIC X(30).
```



移行支援機能

- ZONEDATA(MIG|PFD)
  - 無効なデータを処理しておりNUMPROC(MIG)を使用しているプログラムを、COBOL V5のNUMPROC(NOPFD)に移行すると、実行結果に差異が発生していた
  - 差異発生の例:

```
WORKING-STORAGE SECTION.
 77  VALUE0    PIC X(4) VALUE '00 0'.          *>  x'F0F040F0'
 77  VALUE1    REDEFINES VALUE0 PIC 9(4).
 77  VALUE3    PIC X(4) VALUE '00A0'. '.       *>  x'F0F0C1F0'
 77  VALUE4    REDEFINES VALUE3 PIC 9(4).

PROCEDURE DIVISION.
   IF VALUE1 = ZERO
      DISPLAY "VALUE1 = ZERO  " VALUE1
   ELSE
      DISPLAY "VALUE1 NOT = ZERO  " VALUE1
   END-IF

   IF VALUE4 = 10
      DISPLAY "VALUE4 = 10  " VALUE4
   ELSE
      DISPLAY "VALUE4 NOT = 10  " VALUE4
   END-IF
```

ZONEDATAEオプション：差異発生の例の実行結果

- COBOL V4およびそれ以前のコンパイラーでNUMPROC(MIG) 指定の場合か、COBOL V5でZONEDATA(MIG)を指定した場合の実行結果:

```
VALUE1 = ZERO   00 0
VALUE4 = 10     00A0
```

- COBOL V4およびそれ以前のコンパイラーでNUMPROC(PFDかNOPFD)指定の場合か、COBOL V5でZONEDATA(PFD)を指定した場合の実行結果:

```
VALUE1 NOT = ZERO   00 0
VALUE4 NOT = 10     00A0
```

移行支援機能

- MAP(HEX)
  - お客様からのご要望により、MAPの出力時に10進のオフセットサポートがV5.1で組み込まれた
  - V4とV5のリストを使用する際に換算する必要があることが問題となった
  - このため、ユーザーは10進オフセットを使用するためにMAP(DEC)を指定するか、
  - 以前のコンパイラーと同様にMAP(HEX)を指定して16進オフセットを使用することが可能
  - MAPのみの指定をすると、MAP(HEX)と解釈される

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。 本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。 パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、Bluemix、DB2、PureData、S/390、System z、Watson、z/OS、およびzEnterprise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。

Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。
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。
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。
Hadoopは、Apache Software Foundationの米国およびその他の国における登録商標または商標です。